Universidad de la República
Facultad de Ingeniería

Sistema de Monitoreo y Control de Cultivo Indoor de Cannabis

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Diego Alonso, Carlos Menoni, Felipe Saravia

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista.

Tutor

Nicolás Pérez ..... Universidad de la República

Tribunal

Pablo Monzón ..... Universidad de la República
Leonardo Steinfeld ..... Universidad de la República
Florencia Blasina ..... Universidad de la República

Montevideo
martes 20 diciembre, 2022

Sistema de Monitoreo y Control de Cultivo Indoor de Cannabis, Diego Alonso, Carlos Menoni, Felipe Saravia.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 232 páginas.
Compilada el martes 20 diciembre, 2022.
http://iie.fing.edu.uy/

Lo que no se mide no se puede controlar, y lo que no se puede controlar no se puede mejorar.

William Thomson

Esta página ha sido intencionalmente dejada en blanco.

Agradecimientos

Al tutor del proyecto, Nicolás Pérez por la dedicación y gran apoyo.

A los consultores Leonardo Steinfeld y Pablo Pérez, por la asistencia y disponibilidad brindada.

Al Departamento de Electrónica de la Universidad Católica del Uruguay, por la asistencia durante el ensamblado de las placas.

A Rufo Martínez por la confianza depositada en el equipo de trabajo y la buena disposición de siempre.

A nuestras familias, amigos y compañeros de facultad por el apoyo durante la elaboración del proyecto y en el desarrollo de nuestra carrera.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

El cultivo de tipo indoor es ampliamente utilizado en el mundo. Consiste en el desarrollo de plantas en un entorno controlado, donde se gestionan de forma artificial los elementos necesarios para su crecimiento. En Uruguay han surgido múltiples emprendimientos aplicando esta técnica de cultivo a partir de la aprobación de la Ley 19.172, que regula la actividad productiva del cannabis.

Este proyecto apunta a la aplicación de un sistema de monitoreo y control del proceso productivo en este tipo de cultivos.

El sistema desarrollado esta conformado por una red inalámbrica en la cual un conjunto de sensores, actuadores y controladores, se encargan de realizar el monitoreo y control. Se cuenta con un SCADA *web-based*, lo cual permite crear una interfaz humano-máquina con el sistema donde es posible configurar parámetros de control, así como visualizar un conjunto de variables de monitoreo.

El sistema monitorea las principales variables del proceso productivo, relacionadas con el entorno ambiental de las plantas: luz, humedad, temperatura, agua. Se busca brindarle al productor información objetiva que le permita conocer la salud de las plantas.

Se implementan también lazos de control que, a partir de la información de monitoreo, permiten automatizar algunas tareas y mejorar las condiciones a las que se someten las plantas. Estos lazos de control inciden en la calidad y cantidad del agua de riego, la temperatura de las salas productivas y la humedad ambiental.

Para la implementación de esta solución, se diseñaron y ensamblaron placas electrónicas, que trabajan con módulos con una radio capaz de comunicarse según el estándar IEEE 802.15.4. Estos sistemas embebidos se utilizaron con el sistema operativo Contiki-NG, diseñado para la implementación del stack de una red 6LoWPAN.

A lo largo de este proyecto se transita por toda la etapa de diseño de un sistema embebido hasta la aplicación del mismo. Se afrontan los desafíos de diseño e integración, utilizando conocimientos de comunicaciones inalámbricas, procesos de control y el concepto de IoT. Por último, se busca desarrollar una aplicación que tenga un valor agregado para el usuario final.

Esta página ha sido intencionalmente dejada en blanco.

Abstract

Indoor type cultivation is widely used over the globe. It consists on growing plants in a controlled environment, where necessary aspects of this process are artificially managed. In Uruguay, many developments under this tecnique have emerged after the 19.172 law approval, which regulates cannabis productive activity. This project aims to apply a control and monitoring system for the productive process of this type of cultivation.

The developed system is formed by a wireless network in which a set of sensors, actuators and controllers, are in charge of carrying out monitoring and control. There is a web-based SCADA that allows to create a human-machine interface with the system with the possibility of configuring control parameters as visualizing a set of monitoring variables. The system monitors the most important variables of the productive process, related to the environmental setting of the plants, such as: light, humidity, temperature, water. It is wanted to provide the grower objective information allowing him to have proper knowledge about plant health. Control loops can, out of monitoring information, automate some chores and embetter conditions plants are submitted to. These control loops impact on wuality and quantity of irrigation water, productive rooms temperature and environmental humidity.

To carry out this solution, electronic boards were designed and assembled. These boards work with System on Chip modules with a radio capable of communicating according to IEEE 802.15.4 standard. These embedded systems were run upon Contiki-NG operative system, designed to implement the stack of a 6LoWPAN network.

Along this project the team has gone through the stages of designing an embedded system, up to the application of it. Designing and integration challenges are faced, using knowledges about wireless communications, control processes and IoT concept. Lastly, it is wanted to arrive to an application with an added value for the final user.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Agradecimientos **5**

Resumen **7**

Abstract **9**

1. Introducción **1**
1.1. Contexto y Motivacion 1
1.2. Descripción del Problema 1
1.3. Objetivos 2
1.4. Resultados 3
1.5. Descripción del documento 4
1.5.1. Capítulos 4
1.5.2. Apéndices 5

2. Estado del Arte **7**
2.1. Investigación de Cultivo 7
2.2. Proceso Productivo 9
2.3. Redes de Sensores Inalámbricos 13
2.3.1. Sistema Operativo 13
2.3.2. Redes Inalámbricas 14

3. Arquitectura **15**
3.1. Control Distribuido 15
3.1.1. Unidad Central 16
3.1.2. Unidades de Control Distribuido 16
3.1.3. Nodos Maceta 18
3.1.4. Actuadores 19
3.1.5. Sensores 19
3.2. Red Inalámbrica 20
3.3. Lazos de Control 21
3.3.1. Control de Humedad de Suelo 21
3.3.2. Control de Nivel de pH 22
3.3.3. Control de Humedad Ambiente 23
3.3.4. Control de Temperatura Ambiente 23
3.4. Sistema SCADA 23

Tabla de contenidos

4. Hardware **25**
4.1. Computadora Raspberry Pi . . . 25
4.2. Placas electrónicas: UCD y NM . . . 26
4.2.1. Módulo SoC . . . 27
4.2.2. Unidad de Control Distribuido . . . 30
4.2.3. Nodos Maceta . . . 32
4.2.4. Ensamblado de Placas . . . 34
4.3. Sensores . . . 35
4.3.1. Humedad y Temperatura Ambiente . . . 36
4.3.2. Iluminación . . . 37
4.3.3. Concentración de CO_2 . . . 38
4.3.4. Humedad de Suelo . . . 39
4.3.5. Temperatura de Suelo . . . 40
4.3.6. Nivel de pH . . . 42
4.3.7. Caudal de Agua . . . 42
4.3.8. Sensor Boya . . . 43
4.4. Actuadores . . . 44
4.4.1. Electroválvulas . . . 44
4.4.2. Bombas . . . 45
4.4.3. Relés . . . 47
4.5. Otros . . . 48
4.5.1. Tableros . . . 48
4.5.2. Llaves Selectoras . . . 48

5. Comunicación **51**
5.1. Características Generales . . . 51
5.2. Red interna - 6LoWPAN . . . 53
5.2.1. Capa Física . . . 53
5.2.2. Capa de Enlace . . . 54
5.2.3. Capa de Red . . . 55
5.2.4. Capa de Transporte . . . 57
5.2.5. Capa de Aplicación . . . 57
5.3. Aplicación . . . 59
5.3.1. Intercambio Cliente-Servidor . . . 60
5.3.2. Mensajes entre UCD y NM, y desde UC hacia UCD-R . . . 65
5.3.3. Mensajes de telemetría hacia UC . . . 65
5.3.4. Mensajes de comando y configuración de parámetros desde UC . . . 66
5.3.5. Adquisición de datos . . . 68

6. Control **71**
6.1. Control de Humedad de Suelo . . . 72
6.1.1. Parámetros Hídricos . . . 72
6.1.2. Rango Óptimo . . . 73
6.1.3. Lazo de Control . . . 73
6.2. Control de Nivel de pH . . . 75

6.2.1. Valores Óptimos 75
6.2.2. Proceso de Ajuste 76
6.2.3. Lazo de Control 77
6.3. Control de Humedad Ambiente 79
6.3.1. Rango Óptimo 79
6.3.2. Lazo de Control 79
6.4. Control de Temperatura Ambiente 80
6.4.1. *Set-points* de Temperatura 80
6.4.2. Lazo de Control 80

7. Sistema SCADA **83**
7.1. Elección SCADA 83
7.2. Thingsboard 84
7.3. Arquitectura 85
7.4. *Scripts* en Python 86
7.5. Acceso remoto 87
7.6. Cadena de reglas 88
7.6.1. Procesamiento de datos y filtrado 89
7.6.2. Alarmas 89
7.7. Paneles 90

8. Puesta en marcha **93**
8.1. Pruebas FAT 93
8.1.1. Unidad Central 93
8.1.2. Nodos Maceta 94
8.1.3. Unidades de Control Distribuido: UCD-V, UCD-F y UCD-S 95
8.1.4. Unidades de Control Distribuido: UCD-R1 y UCD-R2 . . . 97
8.2. Instalación del sistema 100
8.2.1. Instalaciones realizadas 100
8.2.2. Pendiente 101
8.2.3. Mejoras en instalación 102
8.3. Pruebas SAT 103
8.3.1. Pruebas realizadas 103
8.3.2. Pendiente 104

9. Conclusiones **105**
9.1. Dificultades 107
9.2. Trabajos futuros 108

A. Ensayos **111**
A.1. Ensayo Comparativo de Sensores de Humedad de Suelo 111
A.1.1. Introducción 111
A.1.2. Objetivos 111
A.1.3. Desarrollo 111
A.1.4. Resultados 113
A.1.5. Conclusiones 115

A.2. Calibración de Sensores de Humedad de Suelo Capacitivos 115
A.3. Calibración de Sonda de pH 117
A.4. Parametrización de Ajuste de pH 119
A.5. Calibración de Caudalímetros 121
A.6. Calibración de Sensor de CO2 122
A.7. Parametrización de Bomba Peristáltica 123
A.8. Parametrización de Caudal de Riego 124

B. Manual Técnico **127**
B.1. Entorno de desarrollo 127
B.1.1. Hardware 128
B.1.2. Software 128
B.2. Unidad Central 129
B.2.1. Hardware 130
B.2.2. Sistema Operativo 130
B.2.3. SCADA 131
B.2.4. Border Router 132
B.2.5. Router Internet 133
B.3. Nodo Maceta 137
B.3.1. Hardware 137
B.3.2. Software 138
B.4. Unidad de control distribuido 138
B.4.1. Hardware 138
B.4.2. Software 141
B.5. Expandir sistema 142
B.5.1. Nodo 142
B.5.2. SCADA 142
B.5.3. *Script* Python 143
B.6. Desarrollo 144
B.6.1. Thingsboard 144
B.6.2. Contiki-NG 145
B.7. *Troubleshooting* 147
B.7.1. Luego de programar SoC se trancó 147
B.7.2. Programación *bootloader* deshabilitada 147

C. Manual de usuario **153**
C.1. Interfaz SCADA 153
C.1.1. Acceso 153
C.1.2. Navegación 154
C.1.3. Históricos 155
C.1.4. Paneles 156
C.1.5. Ventana de tiempo 159
C.1.6. Alarmas 160
C.1.7. Configuración de parámetros 162
C.2. UCD - Llave Selectora 163
C.3. NM - Indicador de Carga 164

C.4. VNC - Acceso Gráfico Remoto 165
C.5. *Troubleshooting* 166
C.5.1. Los botones de los paneles presentan error 166
C.5.2. No se tiene acceso remoto a la pagina web pero si local . . 166

D. Proyecto de Fin de Curso de RSI **167**

Referencias **203**

Índice de tablas **209**

Índice de figuras **210**

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 1

Introducción

1.1. Contexto y Motivacion

Uruguay cuenta con una de las culturas de cannabis más desarrolladas del mundo, siendo pionero en la legislación de la planta para sus diversos usos. Desde 2013 regula la producción, comercialización y consumo.

La Comisión de Adicciones de la cámara de diputados aprobó el proyecto de ley modificado en 2013, que prevé que las farmacias puedan vender hasta 40 gramos de inflorescencias de cannabis por persona por mes. Esta ley también posibilita el autocultivo por parte de individuos o de clubes de consumidores. Asimismo, se estipula la regulación de permisos a empresas privadas para cultivo de marihuana psicoactiva orientado al uso medicinal o recreativo, cultivo de cáñamo para usos industriales (producción de papel, textiles o combustibles, entre otros) y cultivo con fines de investigación [76].

En resumen, todos los ciudadanos o residentes del país mayores de 18 años, previo registro, pueden adquirir marihuana en farmacias autorizadas hasta 10 gramos por semana, con una tenencia máxima de 40 gramos. Del mismo modo está permitido el cultivo personal de hasta seis plantas, con un máximo de 480 gramos cosechados al año. Se puede cultivar también en clubes con membresía, con un mínimo de 15 socios y un máximo de 45, cosechando un máximo del equivalente a 40 gramos mensuales por socio.

Los clubes cannábicos son asociaciones sin fines de lucro para la producción y distribución legal de cannabis entre un grupo cerrado de usuarios. Estos clubes están integrados por miembros adultos, que cultivan plantas de cannabis colectivamente para satisfacer sus necesidades personales, sin tener que recurrir al mercado ilegal. Son clubes de membresía, y la regulación del funcionamiento de los mismos implica una serie de derechos y compromisos que sus miembros deben asumir.

1.2. Descripción del Problema

El proyecto es planteado a partir de un cliente externo, rol que cumple el club cannábico El Tercer Ojo, habilitado por el Instituto de Regulación y Control del

Cannabis (IRCCA), y el Lic. Juan Martínez en representación del mismo [77].

Este club se dedica al cultivo de tipo *indoor* (cultivo interior en un ambiente controlado), aunque también realiza plantación exterior. Si bien el cultivo de cannabis se puede realizar al aire libre, el cultivo de tipo *indoor* es ampliamente utilizado en el mundo. Consiste en el desarrollo de plantas en un entorno controlado donde se gestionan de forma artificial los elementos necesarios para su crecimiento. Es en esa gestión donde este proyecto tiene su incidencia, buscando implementar o mejorar el diagnóstico y control del proceso productivo, en particular del entorno ambiental del cultivo.

En este proyecto se planteó la preocupación del cliente de optimizar su producción, mejorando la eficiencia de sus procesos. Dentro de las principales preocupaciones del cliente se identifica el consumo energético (proveniente principalmente de la iluminación artificial) y la dificultad de tener un seguimiento de las condiciones de crecimiento de las plantas de cannabis.

Es importante destacar que, previo al inicio del proyecto, no poseía un sistema de monitoreo o control de gran parte de las variables del proceso productivo, siendo este uno de los principales requerimientos del mismo. En particular, el cliente expresa particular interés en la mejora de la planta de agua y el sistema de riego, dado que es el rubro que más costo personal le implica, en horas y esfuerzo.

1.3. Objetivos

El objetivo general de este proyecto es diseñar una red inalámbrica de sensores y actuadores que permita el monitoreo y control de las variables más relevantes involucradas en el ciclo biológico de la planta de cannabis.

Como objetivos secundarios se plantearon los siguientes:

- Automatización del control del pH del agua utilizada para riego: El pH es una medida cuantitativa de la acidez o la basicidad en este caso del agua. Esta variable es importante para lograr un crecimiento óptimo de las plantas.

- Automatización del control del aire acondicionado: Debido al funcionamiento del cultivo *indoor*, las lámparas utilizadas disipan calor, generando la necesidad de regular la temperatura en este caso mediante un equipo de aire acondicionado.

- Automatización del sistema de riego: Se desea lograr una automatización de la tarea más demandante que conlleva el mantenimiento de un club cannábico, como lo es el riego. Esta tarea requiere horas de presencialidad diarias, pudiendo ser un problema significativo dependiendo de la disponibilidad de la persona a cargo.

- Acceso remoto: Se quiere poder monitorear la evolución en tiempo real de todos los factores clave en el desarrollo de la planta.

- Sistema de alarmas: Se quiere que el cliente sea notificado en caso de que alguna parte del sistema a desarrollar entre en estado crítico.
- Trazabilidad: Se busca almacenar en una base de datos toda la información adquirida por el sistema con el fin de mantener una trazabilidad sobre los factores incidentes en el ciclo de vida de cada una de las plantas.

Lograr estos objetivos implica el desarrollo de un sistema de monitoreo y control.

Se busca que la interfaz de usuario sea simple, amigable y accesible de forma remota. Resulta importante contar con una base de datos, que permita tener un registro de las variables monitoreadas.

1.4. Resultados

El sistema desarrollado está conformado por una red inalámbrica en la cual un conjunto de sensores, actuadores y controladores, se encargan de realizar el monitoreo y control. Se cuenta con un SCADA *web-based*, lo cual permite crear una interfaz humano-máquina con el sistema, donde es posible configurar parámetros de control, así como visualizar un conjunto de variables de monitoreo.

El sistema monitorea las principales variables del proceso productivo, relacionadas con el entorno ambiental de las plantas: luz, humedad, temperatura, agua. Se busca brindarle al productor información objetiva que le permita conocer la salud de las plantas.

Se implementan también lazos de control que, a partir de la información de monitoreo, permiten automatizar algunas tareas y mejorar las condiciones a las que se someten las plantas. Estos lazos inciden en la calidad y cantidad del agua de riego, la temperatura de las salas productivas y la humedad ambiental.

Para llevar a cabo estas funcionalidades, el sistema cuenta con tres dispositivos electrónicos. En primer lugar, se tiene la Unidad Central (UC). En esta unidad, se tiene una computadora de placa simple donde se corre el SCADA, además de guardarse las bases de datos y conectarse a WiFi para alcanzar las redes externas. También se tiene un *Border Router* (BR), que cuenta con la tecnología para poder comunicar la computadora con la red inalámbrica interna. Además, en la UC se ejecutan *scripts* de Python que ayudan en la comunicación interna.

Por otro lado, se tienen dos tipos de controladores. Las Unidades de Control Distribuido (UCD) son instaladas en todas las salas de producción y se encargan de comunicarse con sensores, intercambiar información con la UC y ejecutar cálculos y acciones de control de ciertos procesos. El otro controlador se denomina Nodo Maceta (NM). Si bien no realiza cálculos de control, este dispositivo se encarga de manejar los sensores que miden variables en la propia maceta de la planta. Además se comunica tanto con la UC como con algunas UCD.

Tanto las UCD como los NM, son placas electrónicas de circuito impreso, diseñadas y ensambladas por el equipo de proyecto. Están basadas en el módulo

CC2650MODA, fabricado por Texas Instruments (TI), con el que pueden comunicarse en la red inalámbrica interna. A su vez, cada módulo dispone del sistema operativo Contiki-NG, que contiene todas las herramientas necesarias para la implementación del stack de red elegido.

La comunicación entre nodos se realiza mediante una red 6LoWPAN del tipo *mesh*, donde se utiliza el protocolo estándar IEEE 802.15.4 [2] para capas física y de enlace, RPL y 6LoWPAN para capa de red, UDP para capa transporte y CoAP para capa aplicación.

Para el control y monitoreo de todas las variables del sistema se optó por utilizar como SCADA (acrónimo de *Supervisory Control And Data Acquisition*) una plataforma gratuita llamada *Thingsboard*, que permite crear paneles para la visualización de datos y control remoto de dispositivos en tiempo real. Esta plataforma también es capaz de generar alarmas y procesos de filtrado de mensajes recibidos, así como publicar en la red local una página para utilizar como interfaz.

1.5. Descripción del documento

El documento se encuentra organizado en capítulos y apéndices. A continuación se realiza una breve descripción del contenido de cada uno.

1.5.1. Capítulos

Introducción

Este es el primer capítulo, se presenta el contexto y la motivación del proyecto realizado. Se presenta el problema que se busca resolver y los objetivos.

Estado del Arte

El segundo capítulo busca introducir al lector en el contexto más técnico del proyecto, explicando el proceso productivo del club y algunas características técnicas de la solución a implementar.

Arquitectura

El tercer capítulo describe la arquitectura del sistema, se busca dar una visión global, donde se pueda generar una idea macroscópica del rol de cada parte del sistema.

Hardware

El cuarto capítulo se centra en el diseño de las placas y en el hardware utilizado, tanto para la implementación del sistema embebido, como para la formación de las cajas/tableros de control.

Comunicación

El quinto capítulo describe la comunicación de las distinta partes del sistema, explicando en detalle el stack utilizado en la red inalámbrica y los mensajes intercambiados entre cada parte.

Control

El sexto capítulo presenta los lazos de control desarrollados por el sistema y la motivación de los mismos.

Sistema SCADA

El séptimo capítulo busca profundizar en el Sistema SCADA, principalmente en la plataforma utilizada y la función de la misma.

Puesta en marcha

El octavo capítulo describe las instalaciones y pruebas realizadas, tanto en el entorno de desarrollo como en el sitio del cliente.

Conclusiones

El noveno capítulo busca evaluar el trabajo desarrollado en el proyecto, evaluando los resultados obtenidos y comparándolo con los objetivos planteados. Se busca presentar los trabajos a futuro que se podrían desprender de este proyecto.

1.5.2. Apéndices

Ensayos

Este apéndice presenta los ensayos de calibración y modelado de los sensores, y actuadores utilizados en el proyecto.

Manual técnico

El manual técnico busca ser una guía de instrucciones que permita replicar o expandir el sistema de monitoreo y control desarrollado en este proyecto.

Manual de usuario

El manual de usuario tiene como objetivo contener todas las indicaciones e instrucciones necesarias para que el usuario final pueda operar y mantener el sistema de monitoreo y control. Se busca presentar las herramientas para que el usuario pueda sacarle máximo provecho al sistema.

Proyecto de fin de curso de RSI

En el marco del proyecto de fin de curso de Redes de Sensores Inalámbricos, se buscó involucrar la consigna con este proyecto de fin de carrera. En este apéndice se presenta el informe elaborado como parte del proyecto del curso. Se encuentran descripciones detalladas sobre la subcapa de ruteo RPL, ya que uno de los objetivos del proyecto fue analizar RPL Classic vs RPL Lite.

Capítulo 2

Estado del Arte

2.1. Investigación de Cultivo

Como se indica en secciones anteriores, el foco de este proyecto es el monitoreo y control de las variables más importantes del proceso de producción. Se busca brindar herramientas que ayuden al cultivador a tener más información, lo que permite su optimización. Esta producción se basa en el crecimiento de las plantas de cannabis, por lo que resulta esencial estudiar sus características biológicas y factores más importantes.

Según Bruce Bugbee [8], el crecimiento de toda planta se define en función de nueve parámetros principales. Es la gestión de estos parámetros (o algunos de ellos) en lo que consiste la optimización de la producción. Los 9 parámetros son:

1. Luz
2. Temperatura ambiente
3. Humedad ambiente
4. Concentración de dióxido de carbono
5. Movimiento del aire
6. Temperatura del suelo
7. Humedad del suelo
8. Cantidad de nutrientes (fósforo, nitrógeno y potasio)
9. Concentración de oxígeno en suelo

De esta lista, la luz es el parámetro más importante, dado que no solo es el que más incide en el crecimiento de la planta, sino que también lo hace en los demás parámetros. Es decir, su incremento o reducción también implica una variación acorde de los demás para mantener la planta saludable. Además suele ser el parámetro que requiere mayor inversión económica.

Si bien existen distintas opiniones respecto al régimen horario a implementar, la configuración más utilizada es de 18 horas diarias de luz y 6 de oscuridad para la fase de crecimiento vegetativo, mientras que en la fase de floración se igualan las horas diarias de luz y oscuridad [30] [46].

Otro aspecto esencial en la salud de la planta es la temperatura del ambiente. Si bien la planta de cannabis es originaria de las regiones templadas de Asia, hoy en día crece en todo el mundo, adaptándose a los distintos climas. De todas formas, para un mejor rendimiento de la planta se busca que la temperatura del ambiente sea similar a su clima de origen, estando en el rango de los 25-30 °C para las horas de luz y los 15-25 °C en oscuridad.

Este parámetro tiene relación estrecha con la iluminación implementada, siendo que gran parte de la potencia de las lámparas se convierte en calor, aumentando la temperatura de la sala. Es así que se debe tener especial cuidado, dado que puede afectar la salud de la planta e incrementarse el gasto para refrigerar el ambiente [12].

La presencia de un flujo de aire mediante ventilación resulta importante para la salud de la planta. En primer lugar promueve el fortalecimiento de los tallos en su etapa de crecimiento vegetativo, pero también ayuda al control de las variables de humedad y temperatura ambiente y presencia de CO_2 en el aire.

Si bien al realizar la fotosíntesis la planta consume parte del agua absorbida del suelo, la mayoría de ésta es transpirada por las hojas y evaporada al ambiente. La presencia de altas concentraciones de humedad en el aire puede atraer plagas y generar enfermedades en la planta. Se debe entonces mantener la humedad en un valor entre 40 % y 70 %, siendo 60 % el punto óptimo.

También es sabido que el dióxido de carbono es un elemento fundamental en el proceso de fotosíntesis, dado que la planta consume este gas y libera oxígeno, por lo que bajos niveles de CO_2 dificultan el crecimiento. En el caso del cannabis el nivel óptimo de dióxido de carbono en el ambiente es de unos 400 ppm. Si bien controlar correctamente la proporción de este gas en el aire con equipamiento especializado es complejo y costoso, con la ventilación del ambiente suele ser suficiente. Si no se realiza esta tarea, el aire se carga de la humedad evaporada por la planta y se reduce el porcentaje de CO_2, lo que puede llevar a sofocarla.

Respecto al suelo de la maceta, se debe generar un buen entorno para que las raíces cumplan su función de la mejor manera, evitando hojas descoloridas y una producción de menor calidad. Se debe controlar principalmente la temperatura y humedad del suelo, además de los nutrientes y presencia de oxígeno. En el caso de la temperatura, debe ser cercana a la del ambiente.

La humedad del suelo es otro de los parámetros más importantes y se gestiona mediante el riego. El agua a regar depende directamente de la luz que reciba la planta, pudiendo limitar o impulsar su crecimiento. Es así que resulta de gran importancia mantener un control y una planificación del riego, buscando la mayor capacidad de absorción de la planta con la menor cantidad de agua, evitando el

riego excesivo.

Como todo ser vivo, los nutrientes son un aspecto esencial en el crecimiento de la planta, y en particular en la calidad del producto final. Las plantas de cannabis consumen principalmente tres nutrientes: Fósforo, Nitrógeno y Potasio. En general la fertilización se realiza de forma regular disolviendo estos nutrientes en el agua para riego. Para ello se utilizan productos comerciales preparados con distintas proporciones de estos nutrientes [11].

Finalmente, se debe permitir la presencia de oxígeno en el suelo, por lo que se evita compactar demasiado la tierra [46].

En la tabla 2.1 se apuntan los valores y rangos objetivo para cada parámetro.

Parámetro	Rango Óptimo
Luz	Horas en Vegetativo: 18 luz - 6 no luz
	Horas en Floración: 12 luz - 12 no luz
Temperatura Ambiente	Con Luz: 25-30 $°C$
	Sin Luz: 15-25 $°C$
Humedad Ambiente	40 %-70 % (60 % óptimo)
CO_2	400 ppm
Movimiento de Aire	Suficiente para ventilar ambiente
Temperatura de Suelo	Cercana a ambiente
Humedad de Suelo	En rango de confort hídrico de planta
Nutrientes	Proporciones dadas por fabricantes
Oxígeno en suelo	Evitar suelo muy compactado

Tabla 2.1: Resumen de valores objetivo por parámetro.

2.2. Proceso Productivo

Como se menciona en la sección 1.2, nuestro cliente es el club cannábico El Tercer Ojo, ubicado en el departamento Maldonado. Se dedica tanto a la plantación de exterior como al cultivo *indoor*, cumpliendo la demanda de producción mensual y anual. Para el cultivo *indoor* se disgrega el crecimiento biológico de las plantas en 4 etapas.

La primer etapa consiste en la clonación y primer crecimiento de la planta. El proceso biológico comienza con esquejes cortados de plantas maduras, colocados en un equipo de aeroclonación (ver figura 2.1). Este equipo mantiene los esquejes en contacto con el agua, haciendo que crezcan los primeros brotes de raíz.

Figura 2.1: Equipo de aeroclonación.

En la siguiente etapa las plantas comienzan un proceso de crecimiento de los tallos y hojas de la planta, previo a la floración de la misma. El proceso dura alrededor de 4 semanas. Este crecimiento es acompañado por transplantes progresivos a macetas de distintos tamaños (0,3 L, 3 L y 18 L). En la segunda semana, se cortan los esquejes que serán utilizados como clones para la siguiente camada de plantas. En la figura 2.2 se muestra un grupo de plantas adultas, aun sin flores en sus ramas.

Figura 2.2: Sala de crecimiento vegetativo.

La tercera etapa inicia cuando las plantas llegan a un nivel de madurez donde es viable la floración de las mismas. Las plantas son trasladadas a una sala que

cuenta con un tejido de cuerda sintética en forma de rejilla que permite una disposición de tipo ScrOG (Screen Of Green), como en la figura 2.3. Esta disposición busca que las ramas permanezcan lo más separadas posible entre sí, permitiendo que más ramas reciban una cantidad de luz relativamente homogénea. En esta etapa, que dura unas 8 semanas, se busca que las plantas identifiquen el cambio de estación climática mediante la reducción de horas diarias de luz, provocando su florecimiento.

Figura 2.3: Sala de floración. Las plantas se adecúan para una disposición de tipo ScrOG.

Finalmente, la última etapa consiste en la cosecha y secado del cultivo. Para ello se cortan sus ramas, separando las que contienen flores. Estas últimas son colgadas en tendederos durante una semana para eliminar la humedad interna, como se puede apreciar en la figura 2.4. Luego las flores son cortadas de las ramas y envasadas al vacío con silicagel para una mejor conservación.

El proceso completo desde la clonación hasta el envasado del producto final dura aproximadamente 14-15 semanas.

El proceso productivo es implementado en 4 habitaciones, donde dos de ellas son dedicadas al crecimiento, las otras son, una para el secado posterior a la cosecha y otra para el tratamiento de agua y gestión del riego.

La primera sala la denominamos sala de crecimiento vegetativo, dado que en ella se llevan a cabo las primeras dos etapas. En estas etapas se somete a las plantas

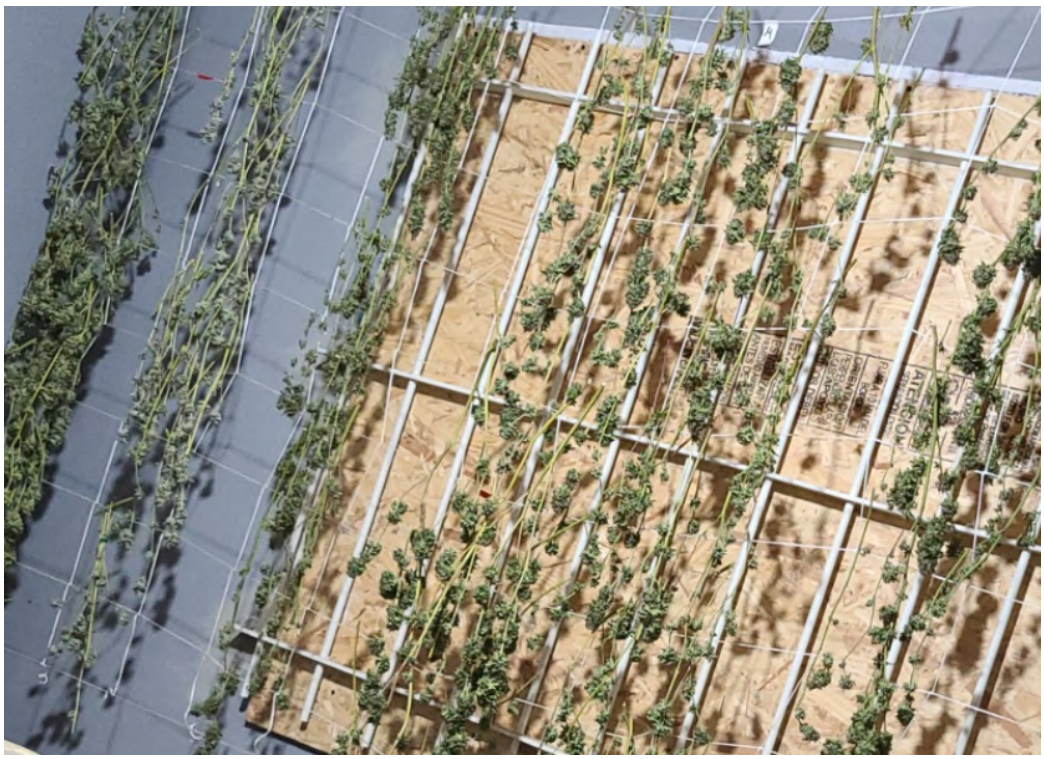


Figura 2.4: Sala de secado.

a un régimen lumínico de 18 horas diarias de luz y 6 de oscuridad. Además, la habitación cuenta con un aire acondicionado que mantiene la temperatura a *22 °C* y ventiladores que generan un flujo de aire. El riego varía según el tamaño y la maceta, siendo que las macetas chicas (0,3 L) están en permanente contacto con el agua, las macetas medianas (3 L) se les riega 0,5 L cada dos días y las grandes (18 L) 3 L cada tres días.

La siguiente sala está dedicada a la etapa de floración. En la sala de floración se cuenta con la infraestrucutura para la disposición ScrOG. El riego se mantiene en 3 L cada dos o tres días por maceta. Además, las plantas reciben las 12 horas de luz diarias necesarias. Para la gestión del ambiente en esta sala se cuenta con aire acondicionado y ventiladores, además de un extractor de aire con filtro de carbono, y orificios en las paredes para la inyección de aire del exterior.

La última etapa se desarrolla en la sala de secado. En esta sala se realiza la cosecha y tendido de las ramas florecidas. En este proceso se utiliza un deshumidifcador, dado que la humedad en el ambiente afecta el secado. Además, se tiene un extractor de aire con filtro de carbono que evita el escape de olores intensos al exterior.

Se dedica una cuarta habitación para la gestión del riego. La planta de cannabis es sensible a las condiciones del agua con la que es regada, por lo que debe ser tratada. El agua utilizada proviene directamente de la red suministro público, que cuenta con un alto nivel de cloro y un pH que pueden afectar a las plantas si no son

corregidos. Actualmente, se utilizan dos tanques de reserva de 100 y 250 L, donde luego de cargarse con agua de OSE diariamente, se la deja reposar para reducir la concentración de cloro por evaporación. Luego de algo más de 20 horas de reposo, se mide el nivel de pH y se realiza un ajuste para dejarlo dentro del rango de 5,8 a 6,2 pH para las salas de crecimiento vegetativo y de floración respectivamente, quedando el agua pronta para el riego. La implementación del riego previa al proyecto, consiste de una bomba sumergible de 1 HP, una manguera que permite alcanzar las habitaciones contiguas y una pistola de riego de uso manual. Por otra parte, una vez por semana se realiza la fertilización, llevada a cabo también en esta sala. Para ello, una vez evaporado el cloro del agua y previo al ajuste de pH, se disuelve en el agua los productos necesarios para la fertilización. Luego se ajusta el pH y se riega normalmente.

2.3. Redes de Sensores Inalámbricos

Una de las principales características del sistema diseñado en este proyecto es la implementación de una red de sensores inalámbricos. Estas redes consisten en un conjunto de nodos con reducida infraestructura física y sin enlaces cableados, que se comunican entre sí mediante ciertos protocolos. Las RSI (o WSN, por su nombre en inglés *Wireless Sensor Networks*), suelen utilizarse en nodos de recursos limitados: tamaño, capacidad de procesamiento, consumo, velocidad de transferencia de datos.

2.3.1. Sistema Operativo

Se conoce un sistema operativo (SO) como el software encargado de proveer los programas de usuario que simplifiquen el modelo de computadora y de gestionar los recursos de la misma [54].

Si bien los sistemas operativos más utilizados a nivel mundial son de la familia Windows y Linux, en los casos de nodos con recursos limitados estos SO resultan imprácticos debido a su enorme huella de memoria (tanto RAM como ROM) y requerimientos de capacidad de procesamiento. Para estos casos existen sistemas operativos de menor huella y menos demandantes, como TinyOS, Contiki-OS o Contiki-NG.

Contiki-NG (de *Next Generation*), al igual que su antecesor Contiki-OS, es un SO multiplataforma y de código abierto para dispositivos dedicados a redes de nodos o *Internet of Things* [17].

Este SO está enfocado a aplicaciones de comunicaciones confiables de bajo consumo y plataformas modernas. La arquitectura de Contiki-NG, en particular el uso de *protothreads*, le permite reducir considerablemente su huella de memoria hasta apenas algunas decenas de kilobytes [18]. Además, el desarrollo de este SO es acompañado por una detallada documentación, tutoriales y guías.

2.3.2. Redes Inalámbricas

Existen múltiples opciones de arquitecturas, protocolos y diseños de red, pero la mayoría de ellas se basan en el estándar IEEE 802.15.4 o IEEE 802.11. El primero de ellos define las especificaciones para la capa física y subcapa MAC para redes con las características de las RSI (conectividad inalámbrica de baja velocidad con dispositivos con limitaciones en consumo energético). En el caso del 802.11, se definen las especificaciones para la capa física y subcapa MAC de las redes inalámbricas de área local (o WLAN, por su nombre en inglés).

A su vez, hay una amplia variedad de protocolos basados en el estándar IEEE 802.15.4, tanto de propietario como de código abierto. En ese sentido, encontramos que los protocolos más ampliamente utilizados y con mayor accesibilidad a documentación y soporte gratuito en internet son 6LoWPAN y ZigBee. Finalmente se elige trabajar con el protocolo 6LoWPAN, dado que ZigBee es de propietario lo cual genera ciertas dificultades de desarrollo.

El protocolo 6LoWPAN [29] (*IPv6 over Low power Wireless Personal Área Network*) surge por el conflicto causado al utilizar IP en redes basadas en el 802.15.4. Si bien el uso del IP es ampliamente aplicado, el tamaño de sus paquetes (MTU, encabezados) no son compatibles con las características de las redes basadas en el estándar IEEE 802.15.4. Dada la importancia del uso de IP, se desarrolla un protocolo que permita modificar los paquetes provenientes de la capa de red para generar tramas acordes a las capas inferiores, y viceversa. Este protocolo, creado por un grupo de trabajo de la IETF (*Internet Engineering Task Force*), define formatos y criterios para adaptar IPv6 y permitir su uso en redes WPAN. Se define entonces una subcapa 6LoWPAN, ubicada entre la capa de red y la de enlace, donde se fragmentan y comprimen los encabezados de IPv6 de forma que puedan ser enviados en tramas acordes al IEEE 802.15.4.

Capítulo 3

Arquitectura

En el presente capítulo se presenta el resultado general del sistema diseñado, describiendo estructura jerárquica, componentes y funcionalidades de todo el sistema.

3.1. Control Distribuido

Se implementa una arquitectura de controladores distribuidos [13] (DCS, por sus siglas en inglés) dado que sus características se adaptan mejor a nuestra aplicación, en contraste con los sistemas de control centralizado (CCS) [33]. La comparación entre ambos se puede resumir en que en el caso de los DCS se necesitan más unidades de control, aunque de menor capacidad de procesamiento y menos cableado. Por otro lado, se genera una independencia ante fallas de los distintos módulos, dado que si un controlador es afectado, los procesos controlados por las demás unidades no se verían perjudicados. Por contraparte, se tiene que los DCS suelen ser más complejos que los CCS, perjuicio que resulta minimizado en nuestra aplicación dado que la magnitud del proceso en cuestión es reducida.

Nuestro proyecto presenta un DCS con una jerarquía de dos niveles: *Nivel de Operación* y *Nivel de Control*.

El primero se caracteriza por ser donde se encuentran los sistemas informáticos para que el operario pueda monitorizar el proceso y en el que se almacena la base de datos. En nuestra aplicación, este nivel es implementado en la Unidad Central (UC).

El otro nivel está compuesto por las diferentes unidades de control a lo largo del proceso productivo, en este caso las Unidades de Control Distribuido (UCD) y los Nodos Maceta (NM). Estas últimas, si bien sus funciones no incluyen realizar cálculos de lazos de control, se asumen en el segundo nivel dado que se comunican directamente con la UC y que la información conlleva un procesamiento previo a enviar el mensaje.

3.1.1. Unidad Central

En el primer nivel encontramos la Unidad Central (UC), implementada en un módulo Raspberry Pi 4. Se encarga de recibir toda la información del monitoreo y el control, correr el *software* SCADA y mantener una conexión vía WiFi con internet, para el acceso remoto al sistema. Además, almacena la base de datos del monitoreo. En la UC también se ejecutan *scripts* de Python, destinado a cumplir ciertas funciones de comunicación entre el SCADA y los demás dispositivos. Todas las funciones de esta unidad se representan en la figura 7.1.

La UC cuenta con un *border-router* conectado a la red interna inalámbrica y con una tarjeta de red inalámbrica WiFi, por lo que puede comunicarse tanto con los controladores distribuidos como con la red global de Internet. El *border-router* es un elemento clave en toda red de sensores, dado que permite vincular dos tecnologías de capa Física diferentes. El *border-router* también separa los dominios de ruteo.

En ese sentido, por un lado esta unidad se comunica mediante una red inalámbrica *6LoWPAN* con los controladores distribuidos, en el nivel inferior inmediato de la jerarquía propuesta. En esta comunicación se reciben los datos de monitoreo e información de fallas, y se envían comandos e información provenientes del SCADA. Por otra parte, vía WiFi se pueden establecer operaciones remotas a través de un portal web, desde el cual pueden visualizarse pantallas, dashboards, botones y enviarse comandos y actualizaciones de set-points.

3.1.2. Unidades de Control Distribuido

Las Unidades de Control Distribuido (UCD) se encuentran en el segundo nivel de jerarquía y realizan el monitoreo y el control propiamente dicho. Estas unidades se encargan de solicitar medidas a los sensores conectados a ellas, además de realizar los cálculos de control y manejar los actuadores. A su vez, las UCD cuentan con llaves selectoras, que permiten activar y desactivar los automatismos desarrollados.

Estas unidades son implementadas en placas electrónicas diseñadas en el marco de este proyecto. Las placas se basan en módulos CC2650MODA de Texas Instruments, consistentes en un SoC (*System on Chip*) con un procesador, una radio (con un transceptor compatible con el estándar IEEE 802.15.4), memoria RAM y FLASH, además de diversos periféricos. Las placas también cuentan con un circuito de alimentación en 24 V_{DC}, elegido por ser una tensión habitual en dispositivos electrónicos en la industria. Se tiene a su vez, reguladores a 5 V_{DC} y 3,3 V_{DC}, cuatro relés DPDT (*Double Pole Double Throw*) y un número de borneras atornillables para acceder los GPIO del SoC y a la alimentación (tanto de 5 V_{DC} como de 3,3 V_{DC}).

Las UCD son instaladas en las cuatro salas relacionadas con el proceso productivo (ver la figura 3.1): sala de crecimiento vegetativo (SV), sala de floración (SF), sala de secado (SS) y sala de riego (SR). Se nombra cada controlador en particular según la sala en la que se ubica: UCD-V, UCD-F, UCD-S y UCD-R. Las funciones

que lleva a cabo cada UCD depende de la sala en la que se encuentra.

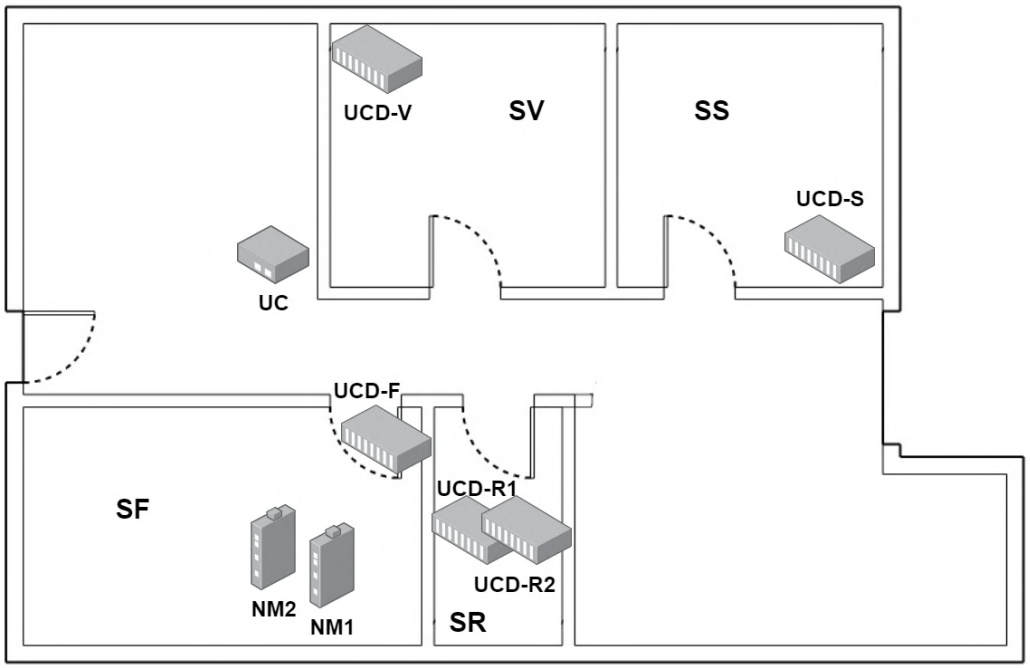


Figura 3.1: Distribución de placas en el establecimiento.

Sala de Crecimiento Vegetativo - SV

En esta sala transcurre la mayor parte de la vida de la planta, donde se impulsa su crecimiento para crear ramas que puedan producir y soportar flores de calidad. Resulta esencial entonces mantener el ambiente en condiciones favorables.

En este caso, la UCD-V toma medidas de humedad ambiente y temperatura ambiente, para lo que utiliza los sensores DHT11, además de estado de luz artificial, con módulos LDR.

Por otra parte, la UCD-V controla el aire acondicionado mediante sus relés y una lógica programada. El control implementado es un lazo cerrado de tipo On/Off, con el equipo establecido en una temperatura fija.

Sala de Floración - SF

Las últimas etapas de vida de la planta consisten en la generación y crecimiento de las flores de las que se cosecha el producto final. En la SF también es importante el estado del entorno.

Al igual que en la SV, la UCD-F monitorea humedad y temperatura ambiente y el estado de la luz artificial (con los mismos sensores), además de la concentración de gas CO_2, con un sensor SEN0219.

En esta sala también se implementa un control de lazo cerrado para la temperatura ambiente, similar al de la SV.

Sala de Secado - SS

En el fin del proceso productivo se encuentra la SS. Luego de cosechadas las plantas, las ramas podadas que cuentan con flores deben ser secadas, por lo que la humedad ambiente de la sala debe ser limitada. En la SS se utiliza el sensor DHT-11 para la medición de la humedad y temperatura ambiente. Además, la UCD-S lleva a cabo un lazo de control relativo a la humedad, donde mediante la gestión On/Off de un equipo deshumidificador mantiene el nivel de humedad alrededor de cierto valor límite.

Sala de Riego - SR

Si bien las plantas no pasan por la SR, esta sala es importante para el proceso productivo. Aquí se instala una planta de agua, desarrollada por el equipo de proyecto, que realiza el tratamiento para un riego saludable.

Las variables monitoreadas en esta sala no están relacionadas con el ambiente, sino con dicha planta. Se tiene entonces una sonda para la medida del nivel de pH, dos caudalímetros (destinados a determinar el volumen de agua utilizada para riego), además de una señal de nivel del tanque mediante sensores de tipo boya.

Por otro lado, las UCD-R implementan dos lazos de control: de humedad de suelo de las macetas en SF y de nivel de pH del agua existente en el tanque. Estos dos lazos de control forman parte del proceso que realiza el riego automático de las plantas.

Debido a la cantidad de actuadores (electroválvulas, bombas, etc) que se deben manejar, se instalan dos UCD en las que se reparten las distintas funciones de monitoreo y actuación.

3.1.3. Nodos Maceta

En el segundo nivel de control distribuido se encuentran los nodos maceta (NM). Consisten en una placa electrónica alimentada por batería, también basada en el módulo CC2650MODA, que controla un grupo de sensores y se comunica de forma inalámbrica con la UC y las UCD. El NM junto con sus sensores conforman un dispositivo individual destinado a monitorear una maceta en particular de forma totalmente inalámbrica, por lo que es instalado en la propia maceta.

Esta placa cuenta con *slot* para una batería de tipo 18650, conexión USB para alimentación y carga de la misma, un circuito de carga de baterías (basado en el integrado TP40546) y resistencias de *pull-up* opcionales (en caso de requerir comunicación por I2C). A diferencia de las UCD, esta placa no implementa lazos

de control, por lo que no cuenta con relés y no maneja actuadores. Además, los sensores asociados son pocos, por lo que la cantidad de borneras accesibles es menor.

Los NM se encargan de tomar la información de los sensores de humedad y temperatura de suelo, realizar cierto procesamiento de los datos y enviarlos directamente a la UC, además de informar el nivel de la batería.

3.1.4. Actuadores

Para llevar a cabo los controles se utilizan dispositivos de actuación que permiten incidir sobre la variable controlada. Los actuadores se reparten en las cuatro salas, de acuerdo a los controles descritos en la sección 3.3. Se muestra en la tabla 3.1 los actuadores utilizados en cada sala y a cuál variable afectan.

Actuador	Variable	Sala
Electroválvulas	Humedad de suelo	SR
	Nivel de Tanque de Agua	SR
Bomba de Dosificación	pH	SR
Relés	Humedad de Suelo (Bomba de Recirculación)	SR
	pH (Bomba de Recirculación)	SR
	Humedad Ambiente (Deshumidificador)	SS
	Temperatura Ambiente	SV
		SF

Tabla 3.1: Detalle de actuadores instalados.

3.1.5. Sensores

Por último, los sensores forman parte del sistema como elementos finales de la arquitectura.

Podemos diferenciarlos en sensores de sala y sensores de maceta. Los sensores de maceta irán conectados por cable al nodo maceta, mientras que los sensores de sala se comunicarán por cable con la UCD de la sala en la que se encuentren. Los tipos de sensores a instalar dependen de la sala y sus funciones.

En la tabla 3.2 se detallan los sensores instalados, la variable que sensan, las salas en la que se encuentran y la unidad con la que se comunican. En la figura 3.2 se ilustra la distribución de los sensores en las salas.

Sensor	Magnitud	Sala	Cantidad	Unidad
DHT-11	Humedad y Temperatura Ambiente	SV	1	UCD-V
		SF	1	UCD-F
		SS	1	UCD-S
		Exterior	1	UCD-V
LDR	Estado de Luz	SV	3	UCD-V
		SF	6	UCD-F
SEN0219	CO_2	SF	1	UCD-F
SEN0193	Humedad de Suelo	SF	2	NM
Sonda DS18B20	Temperatura de Suelo	SF	2	NM
CB-1	Nivel de Agua	SR	2	UCD-R
YFS-201	Caudal de Agua	SR	2	UCD-R

Tabla 3.2: Detalle de sensores instalados.

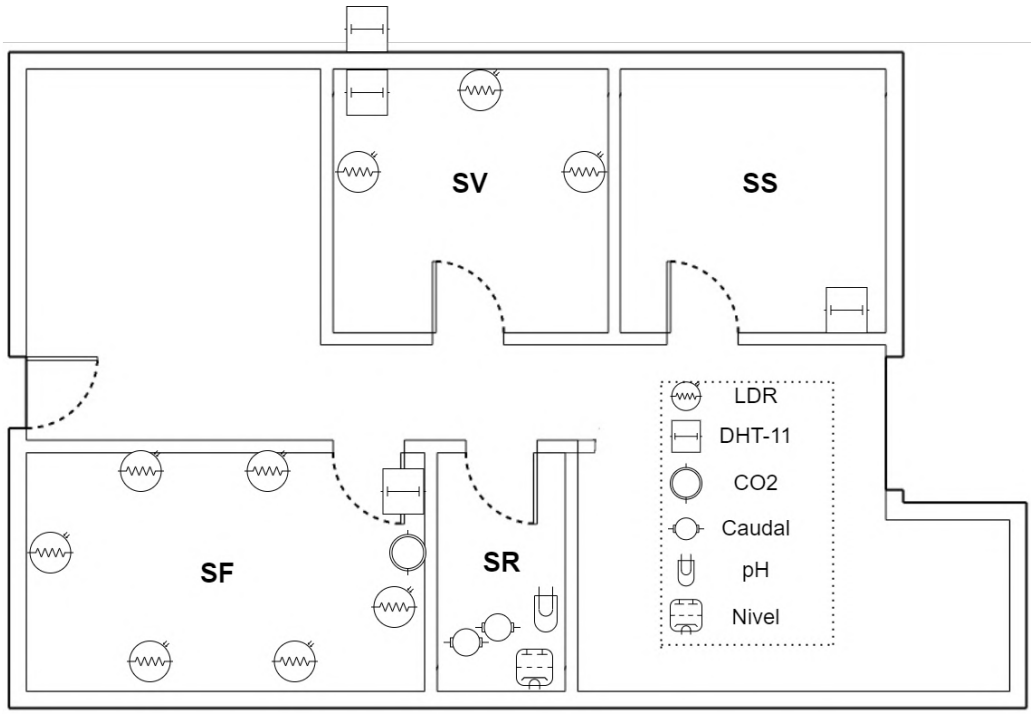


Figura 3.2: Distribución de sensores en las distintas salas.

3.2. Red Inalámbrica

La red a implementar se basa en el protocolo *6LoWPAN*. La red definida es de tipo *mesh*, por lo que todos los nodos pueden vincularse entre sí, eligiendo el mejor camino para enviar la información al destinatario final. Este camino es elegido a partir de una métrica que toma en cuenta aspectos como consumo, velocidad y demoras. De esta forma, tanto las UCD, los NM y la UC, forman parte de la red,

cada uno como un nodo.

A su vez, la UC en particular cumple función de *border-router*, dado que además de pertenecer a la red 6LoWPAN, es el elemento encargado de comunicarse con redes externas.

Tomando en cuenta el modelo de 5 capas, para cada una se definieron protocolos acordes a nuestra aplicación.

En las capas Física y Enlace se aplican los protocolos establecidos en el estándar IEEE 802.15.4, destinado a redes inalámbricas de área personal (WPAN) con bajas tasas de transmisión de datos.

La capa de red se puede dividir en dos subcapas denominadas de Enrutamiento (superior) y de Adaptación (inferior). La capa inferior es la establecida en el estándar 6LoWPAN, destinada a que los paquetes IPv6 puedan ser leídos y procesados por los protocolos del estándar IEEE 802.15.4. Por su parte, en la capa de Enrutamiento se utiliza el protocolo RPL [5] (*Routing Protocol for Low power and Lossy Networks*). En la capa de transporte se elige el protocolo UDP, dado que si bien es un protocolo no orientado a conexión y no confiable, permite una mayor velocidad a cambio de un menor tamaño de segmentos. Además, es el necesario para trabajar con CoAP en capa de aplicación, tema desarrollado en la sección 5.

Por último, en la capa de aplicación se utiliza CoAP (*Constrained Application Protocol*). Este protocolo de tipo REST (cliente-servidor) es un protocolo de transferencia web especializado para su uso en nodos y redes de recursos limitados [51]. CoAP tiene como principal característica la reducción de su *overhead* (relación de bits destinados a la carga útil respecto al mensaje total, incluyendo encabezados). Esto permite un uso optimizado en las redes, mejorando los beneficios del estándar 6LoWPAN. A pesar de ello, CoAP es capaz de implementar una comunicación pedido-respuesta e interactuar con el protocolo HTTP, también de tipo REST y el de uso más extendido, soportando varias funcionalidades de éste.

3.3. Lazos de Control

Como se explicó en secciones anteriores, a lo largo del proceso productivo se implementan cuatro lazos de control.

3.3.1. Control de Humedad de Suelo

El lazo más importante se da en el proceso de riego y busca solucionar el principal problema señalado por el cliente, que radica en la cantidad de horas hombre presenciales que implica esta tarea.

Tomando en cuenta los parámetros hídricos del suelo, se obtiene un rango de valores de humedad que debe contener el suelo para que la planta pueda absorber el agua de forma fácil.

El lazo de control entonces toma como entrada el nivel de humedad de suelo existente y lo compara con el umbral óptimo. En función del nivel de humedad encontrado se toma la decisión del volumen de agua a regar, buscando alcanzar un set-point. Además, se tiene un riego mínimo definido en 0,3 L y uno máximo de 5 L.

Dado que la homogeneización del nivel de humedad de suelo es un proceso muy lento, que puede llevar varios minutos, el valor de la lectura del sensor al momento del riego no puede tomarse como un dato a tiempo real. Por esta razón, este control fue implementado como un lazo cerrado en tiempos discretos, donde se tiene una ventana de tiempo para realizar la comparación con la realimentación.

Se utilizarán para este lazo electroválvulas destinadas tanto al llenado del tanque de agua como al riego de cada grupo de macetas. Además, la medición del volumen de agua vertido se realizará mediante la diferencia de dos caudalímetros (salida y entrada del tanque).

Por razones de alcance y extensión del proyecto, este lazo de control se implementará en macetas de la SF únicamente. Además, se tomarán 2 macetas representativas, que cada una servirá para determinar el riego de un grupo de macetas circundantes, asumiendo estados de humedad similares.

3.3.2. Control de Nivel de pH

Vinculado al proceso de riego, se tiene el control del nivel de pH del agua. Al llenar el reservorio con agua proveniente de la red de suministro, los niveles de cloro y pH no son óptimos. Para eliminar el cloro se remueve el agua cada cierto tiempo (por recirculación) promoviendo su evaporación, luego de lo cual se utilizan químicos para llevar el nivel de pH a un valor conveniente.

El control de pH es el primer lazo a aplicarse una vez iniciado el proceso de riego. Se toma la medición del pH del tanque, en función del valor obtenido se inyecta una dosis de producto acidificante para bajar el pH. Luego de un tiempo, en el que se asume el producto diluido completamente, se toma una nueva medición, prosiguiendo sucesivamente hasta llegar a un valor de pH cercano al set-point deseado.

Dado que la única actuación implementada es la inyección de acidificante, mediante bombeo peristáltico, se busca que la variable se acerque asintóticamente al set-point, evitando sobrepasarlo.

Siendo que si bien existe un tiempo de espera para la dilución del producto, se tiene una realimentación periódica del valor de pH del agua, por lo que en este caso el control también es de lazo cerrado a tiempos discretos, con una ventana de tiempo considerablemente menor.

3.3.3. Control de Humedad Ambiente

Un tercer lazo de control se da en la SS. Para el proceso de secado es importante mantener un nivel de humedad de ambiente bajo. Para ello se utiliza un equipo deshumidificador.

En esta sala se implementa un lazo cerrado simple, donde a través de mediciones periódicas del valor de humedad ambiente, se decide cambiar de estado (encendido/apagado) o no el equipo deshumidificador, hasta llevar la variable alrededor de un set-point definido.

3.3.4. Control de Temperatura Ambiente

Una de las preocupaciones expresadas por el cliente involucraba la temperatura ambiente de las SV y SF, donde actualmente solo se tiene la posibilidad de establecer un único set-point utilizando el equipo de aire acondicionado. Sin embargo, el cliente indica que la situación óptima de las plantas es diferenciar las temperaturas de la noche y del día.

En el lazo de control implementado, se tiene en cuenta que las lámparas son grandes fuentes de calor para las salas y que el ambiente exterior también tiene influencia en la temperatura.

En función de las necesidades del problema y buscando aprovechar el comportamiento natural de la temperatura, se fijan en los equipos las temperaturas deseadas para el período iluminado, mientras que en el período de oscuridad se implementan ciclos de apagados y encendidos de los equipos.

Se implementa entonces un control de lazo cerrado continuo de tipo On/Off, donde la realimentación de temperatura ambiente es a tiempo real y los set-points son actualizados según el ciclo de luz en el que se encuentre la sala.

3.4. Sistema SCADA

Se le llama sistema SCADA (*Supervisory, Control and Data Acquisition*) a cualquier *software* que permita el acceso a datos remotos de un proceso y el control del mismo, a través del uso de las herramientas de comunicación necesarias [45].

Para nuestra aplicación se decide utilizar un *software* SCADA de tipo *web-based* gratuito. Este *software* está dedicado al almacenamiento de los datos, monitorización y visualización de los mismos en tiempo real, chequeo de alarmas y eventos, además de la supervisión de los procesos y eventuales mandos del operario.

El *software* elegido fue Thingsboard, que cuenta con un amplio espectro de funcionalidades. En primer lugar, permite elaborar pantallas de monitoreo, en las que se presenta información del sistema a tiempo real en un formato navegable, además de incluir botones de comandos, avisos de alarmas configurables, gráficas, etc. También implementa una base de datos, en la cual mantendremos registros

históricos de las magnitudes sensadas y demás información. Como interfaz de usuario, se utiliza la página web que este *software* publica en la IP local de la máquina. A su vez, esta IP se publica a internet, de forma de disponibilizar el acceso remoto.

Capítulo 4

Hardware

En este capítulo se describe en detalle las diferentes decisiones tomadas en el proceso de elección o diseño de los componentes hardware del sistema.

En líneas generales podemos discriminar los componentes en cinco grupos distintos: computadora, placas electrónicas, sensores, actuadores y otros componentes. Para cada uno de ellos se estudia la posibilidad de obtener el componente como producto comercial o realizar nuestro propio diseño. Excede el alcance de este proyecto diseñar elementos como sensores o actuadores, así como la computadora, por lo que se adquieren productos del mercado. Respecto a las placas electrónicas de las UCD y NM, dado que su finalidad tiene cierta especificidad se decide realizar un diseño propio que cumpla con los requerimientos.

4.1. Computadora Raspberry Pi

A nivel de operario, el sistema comienza en la interfaz hombre-máquina que implementa nuestro SCADA. Este SCADA necesita una plataforma de procesamiento para ejecutarse, con ciertos requerimientos de rendimiento y capacidad.

Para la elección de la plataforma se busca dentro del universo de las computadoras de placa simple (SBC - *Simple Board Computer*), dado que son equipos compactos y de suficiente capacidad. Se analizan y comparan tres modelos [23].

De la tabla 4.1 podemos observar en primer lugar que la computadora BeagleBone Black resalta como la de menor capacidad, tanto en procesamiento como en memoria RAM. Encontramos también que la Raspberry Pi y la PCDuino son similares en CPU. El microprocesador de PCDuino tiene cierta ventaja en velocidad, mientras que la Raspberry cuenta con un modelo más avanzado de 64 bits, respecto a los 32 bits del A17.

De esta forma, observamos que si bien la computadora BeagleBone Black es la de menor costo, entendemos que sus características podrían no ser suficientes para nuestra aplicación.

Modelo	Raspberry Pi 4-B	PCDuino 9s	BeagleBone Black
Procesador	ARM Cortex A72	ARM Cortex A17	ARM Cortex A8
Arquitectura	64 bits	32 bits	32 bits
Velocidad	1,5 GHz	1,8 GHz	1 GHz
RAM	8 Gb LPDDR4	2 Gb DDR3	512 Mb DDR3
FLASH	32 Gb	8 Gb	4 Gb
Precio (U$S)	150	150	50

Tabla 4.1: Tabla comparativa de computadoras de placa simple.

En cuanto a las dos computadoras restantes, entendemos que la Raspberry Pi es una mejor opción por diversas causas. En primer lugar, vemos que tiene una comunidad amplia en la red, que permite un soporte para un gran número de eventuales problemas. Además, se tiene en cuenta que es una computadora de mayor expansión en el mercado a nivel mundial, por lo que resulta más factible encontrar disponibilidad. Este aspecto no es una ventaja menor, dado que el faltante de chips mundial implica una dificultad en la obtención de computadoras e integrados en general, provocado por la pandemia reciente.

El módulo [10] se caracteriza por contar con un procesador ARMv8 (A72) de 64-bit y cuatro núcleos, de 1,5 GHz de velocidad y 8 Gb de memoria RAM. También incluye una tarjeta SD de 64 Gb clase 10, dedicada a memoria ROM.

Puede comunicarse con el exterior a través de USB (dos puertos 2.0 y otros dos 3.0), HDMI (dos puertos microHDMI) y Ethernet. También cuenta con conexión jack 3.5mm y USB-C para alimentación de 3,5 A.

Se muestra una imagen ilustrativa en la figura 4.1.

4.2. Placas electrónicas: UCD y NM

Uno de los principales elementos de este proyecto son las placas electrónicas. En ellas se lleva a cabo el manejo de los sensores para el monitoreo, los lazos de control, la manipulación de los actuadores y la comunicación de todas las partes entre sí. Por esta razón, se decide que las mismas sean diseñadas y ensambladas por el equipo de proyecto, asegurando así que sus funciones se ajusten correctamente a nuestra aplicación.

Ambas placas fueron diseñadas en el software Eagle, de Autodesk, que permite elaborar el plano esquemático y de placa, además de generar los archivos Gerber requeridos por el fabricante. Con el fin de optimizar y simplificar el diseño, se implementaron dos capas de cobre de 1 $^{oz}/_{ft^2}$ de espesor. La placa UCD tiene unas dimensiones totales de 84x87 mm, mientras que los NM son de 34x80 mm. Las pistas y su aislación tienen un ancho de 8 mils (equivalente a 0,203 mm).

Figura 4.1: Computadora de placa simple Raspberry Pi 4.

4.2.1. Módulo SoC

El componente hardware más importante para implementar la red de comunicación 6LoWPAN es el SoC de las placas, que contiene un microcontrolador, una radio y una antena. Este elemento determina la capacidad de procesamiento, la calidad de la comunicación inalámbrica y el consumo energético de las unidades.

Se elige Texas Instruments como el fabricante del módulo dado que es una marca reconocida a nivel mundial, con un espectro de modelos y opciones muy amplio, además de un soporte extenso (tanto en hojas de datos como en foros y otras fuentes informales) y un desarrollo considerable de este tipo de tecnologías inalámbricas. A su vez, sus productos han sido utilizados reiteradamente en la Facultad de Ingeniería, tanto para proyectos como para investigaciones y aplicaciones de distintos cursos, por lo que se tiene un respaldo de conocimiento y experiencia de parte de fuentes cercanas.

Dentro de las opciones de TI, se buscan por un lado módulos de bajo consumo para los NM, y por otro, módulos con suficiente capacidad de procesamiento y memoria FLASH para las UCD. Además, impusimos como requisito la compatibilidad con Contiki-NG como sistema operativo. De esta forma, la búsqueda queda reducida a la comparación de los chips CC2650 [58], CC1350 [56] y CC2538 [57], resumida en la tabla 4.2.

Lo primero a observar de la tabla, es la igualdad de condiciones en el procesa-

	CC2650	CC1350	CC2538
Procesador	Cortex M3		
FLASH	128 kB		512 kB
RAM	20+8 kB		32 kB
Reloj	48 MHz		32 MHz
GPIOs	31	30	32
ADC	12 bit x8		
Consumo (en TX @0 dBm)	6,1 mA	10,5 mA	24 mA
Bandas de Transceptor	2,4 GHz	2,4 GHz Sub-1 GHz	2,4 GHz

Tabla 4.2: Comparativa de chips

dor, siendo que los tres chips utilizan un ARM Cortex M3, aunque en el caso del CC2538 llega a 32 MHz de reloj, mientras que el CC2650 y CC1350 alcanzan 48 MHz.

En la memoria disponible se observa una diferencia del CC2538 sobre los restantes, dado que el primero cuenta con hasta 512 kB de FLASH y 32kB de RAM, mientras que los otros dos solo incluyen 128 kB de FLASH y 20 kB de RAM, con otros 8 kB de RAM para caché.

En cuanto a los periféricos, las tres opciones cuentan con un convertidor AD de 12 bits y 8 canales, y alrededor de 30 GPIOs disponibles.

Finalmente, se comparan los consumos energéticos durante el proceso de transmisión de radio (contexto de mayor consumo para estos módulos). Se encuentra una marcada desventaja del CC2538 con 24 mA, respecto a los 10,5 mA del CC1350 y los 6,1 mA del CC2650.

En resumen, tenemos pocas diferencias entre el chip CC2650 y el CC1350, mientras que el CC2538 resalta respecto los demas por tener mayor memoria aunque un mayor consumo en transmisión y una velocidad de reloj menor.

De todas formas, dado que el SO elegido (Contiki-NG) posee una reducida huella de memoria, los tres chips satisfacen el único requisito al respecto. Se descarta entonces el CC2538, por lo que finalmente se define como SoC designado el CC2650, dado que además minimiza el consumo energético.

Dado que el *SoC* elegido es la opción de menor consumo, pero mantiene un nivel de procesamiento relativamente potente, se elige tanto para los NM como para los UCD. Además, por precio, flexibilidad y disponibilidad, se decide comprar el módulo CC2650MODA, que consiste en un módulo de reducido tamaño, fácilmente ubicable en una placa electrónica. Se muestra el componente y su pinout en la figura 4.2.

Este módulo se alimenta con una tensión de 3,3 V, consume 1μA en su modo de bajo consumo más profundo y menos de 3 mA estando activo (sin recepción ni transmisión de datos) [58]. También cuenta con una radio con un transceptor de

radiofrecuencia de 2,4 GHz (compatible con el estándar IEEE 802.15.4), además de 15 GPIOs, un ADC incorporado y demás periféricos.

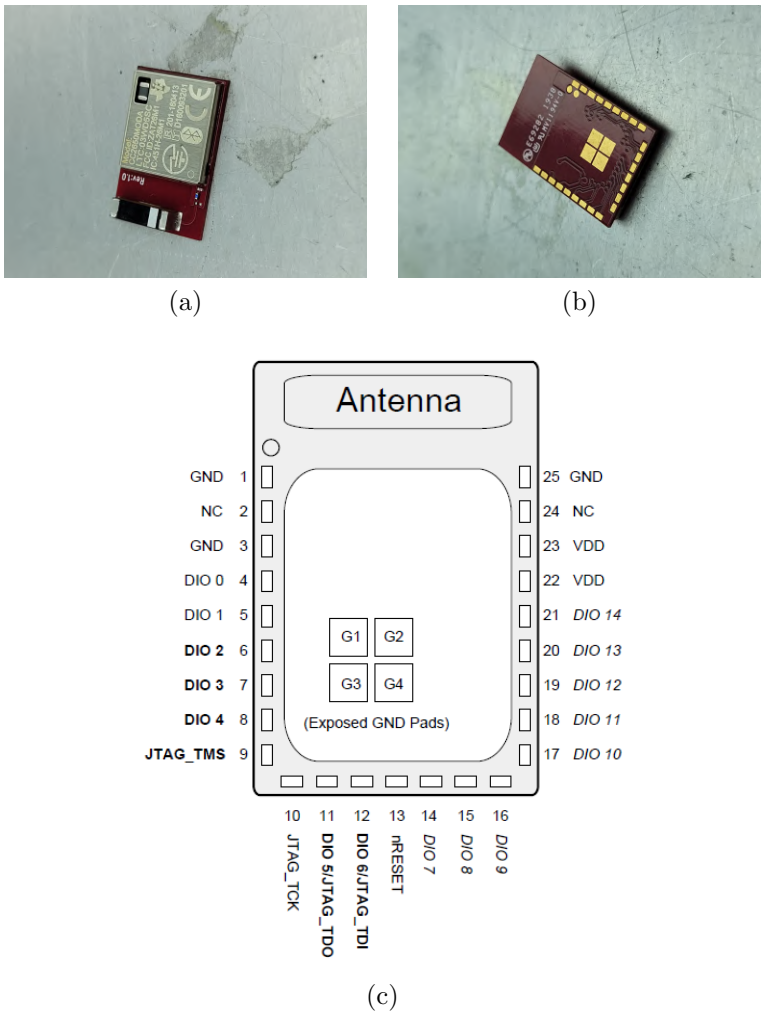


Figura 4.2: Módulo CC2650MODA de Texas Instruments.

Para el ensamblado en la placa, se toman como base las recomendaciones técnicas para el módulo CC2650, incluidas en su hoja de datos. Entre ellas se destaca la sugerencia de dejar un espacio libre de cobre bajo el módulo, a fin de evitar interferencias en su antena.

Se agrega un botón de reset relativo al módulo, con su circuito asociado. El botón consiste de un switch mecánico que permite cortocircuitar la entrada nRESET del microprocesador con tierra. Se incluye un capacitor en paralelo para evitar rebotes y una resistencia de *pull-up* para mantener el estado inactivo.

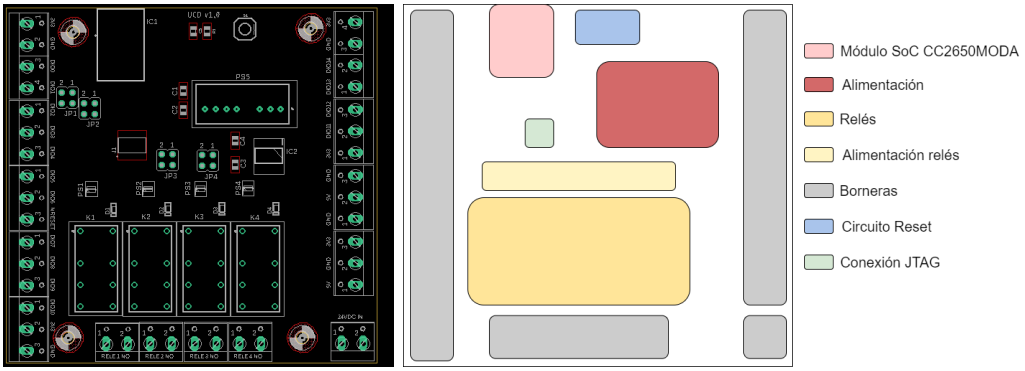
Si bien es más común y fácil programar por *bootloader* (utilizando un periférico UART), se encontraron varios reclamos en internet de que la programación vía

bootloader venía bloqueada de fábrica, debido a un error por parte de Texas Instrument. Se decide entonces incluir un conector FTSH de 10 pines, para programar el SoC mediante comunicación JTAG. Luego, fue necesario limpiar la memoria del SoC por JTAG, para habilitar la programación por *bootloader*, grabando ciertos parámetros en las direcciones de memoria ROM correspondientes [65].

4.2.2. Unidad de Control Distribuido

Como se explica en secciones anteriores, las UCD son placas de circuito impreso basadas en el módulo CC2650MODA de Texas Instruments, dedicadas a ejecutar cálculos de lazos de control y manejar sensores y actuadores. Las placas fueron desarrolladas desde el diseño en el marco de este proyecto, fabricadas mediante un proveedor del exterior y ensambladas, probadas y programadas por el equipo de proyecto.

Para el diseño de la placa se parte de identificar los componentes esenciales de acuerdo a sus funciones, entre los que se encuentran el módulo SoC, relés, alimentación (para el módulo, los sensores adyacentes a la placa y el resto de los componentes de la misma) y borneras atornillables, además de los elementos accesorios a los anteriores. En las figuras 4.3a y 4.3b se muestran los bloques funcionales de la placa y su disposición.



(a) (b)

Figura 4.3: Bloques funcionales de UCD.

Alimentación:

Se define la tensión a introducir en función de los elementos de la placa y auxiliares, además de los reguladores y/o fuentes a colocar. La placa recibe su alimentación a 24 V_{DC} y cuenta con una fuente convertidora 24V-5V y un regulador 5V-3,3V, con sus respectivos capacitores.

Se coloca una fuente SPAN-02B-05, que con una entrada de 24 V_{DC} y un consumo de hasta 90 mA, permite inyectar hasta 500 mA en su salida de 5 V_{DC}, con una eficiencia del 76 % [38]. Por su parte, el regulador es un AMS1117-3.3, con capacidad para casi 1 A [3]. Se agregaron los circuitos de condensadores de entrada y salida de ambos componentes, destinados a regular sus señales.

Relés:

Los relés serán utilizados para el manejo On/Off de ciertos actuadores. En ese sentido, se busca relés de funcionamiento mecánico, cuya bobina permita una alimentación de 3,3 V y sus salidas permitan hasta 230 V. Además, se buscan modelos DPDT, de forma de utilizar un circuito de salida a la tensión del actuador, y otro a tensión 3,3 V destinado a una señal de estado del relé. Por último, debe incluirse un diodo de rueda libre para que al abrirse el circuito se descargue la energía almacenada en la bobina (evitando picos de tensión en el switch), además de un circuito de alimentación para el relé.

Los relés elegidos son los HK19F-DC-3V [27], encontrados en plaza. Estos relés tienen una bobina que permite alimentarse con 3,3 V y consume unos 60 mA mientras se encuentra activa. Ambos circuitos soportan tensiones de hasta 250 V_{AC} y 125 V_{DC}. Debido a que el consumo de la bobina es elevado para la capacidad de los GPIO del SoC, se agregan *load switches*. Se eligen los MIC94090 [40], dispositivos basados en un transistor MOSFET tipo P, que alimentados con 3,3 V entregan hasta 1,2 A.

Dado que son DPDT, se tienen dos circuitos accionados por la misma bobina, cada uno con sus salidas NC y NO. Uno de estos circuitos utiliza su salida NO con alimentación 3,3 V como señal de monitoreo para sensar el correcto funcionamiento del relé mediante nivel alto. El otro de los circuitos tiene tanto su entrada común como su salida NO conectadas a borneras. De esta forma se puede utilizar el circuito para alimentar un dispositivo de cualquier tensión AC (hasta 250 V) o DC (hasta 125 V), o como contacto seco para activar un contactor externo, dejando a elección del instalador esta opción. En la figura 4.4 se describe el circuito de cada relé.

Para controlar y monitorear los cuatro relés se utilizan los pines del DIO1 al DIO4 y del DIO7 al DIO10. De todas formas, mediante el uso de *jumpers*, los relés pueden desconectarse individualmente y dedicar estos pines a otras funciones, a decisión del instalador.

Borneras:

Se decide incluir borneras de tipo atornillable para agregarle versatilidad a la placa, pudiendo así no sólo agregar o quitar sensores y actuadores, sino también cambiar los sensores de bornera de alimentación y de entrada o salida del microprocesador. Se colocan en total 39 borneras repartidas de acuerdo a la tabla 4.3.

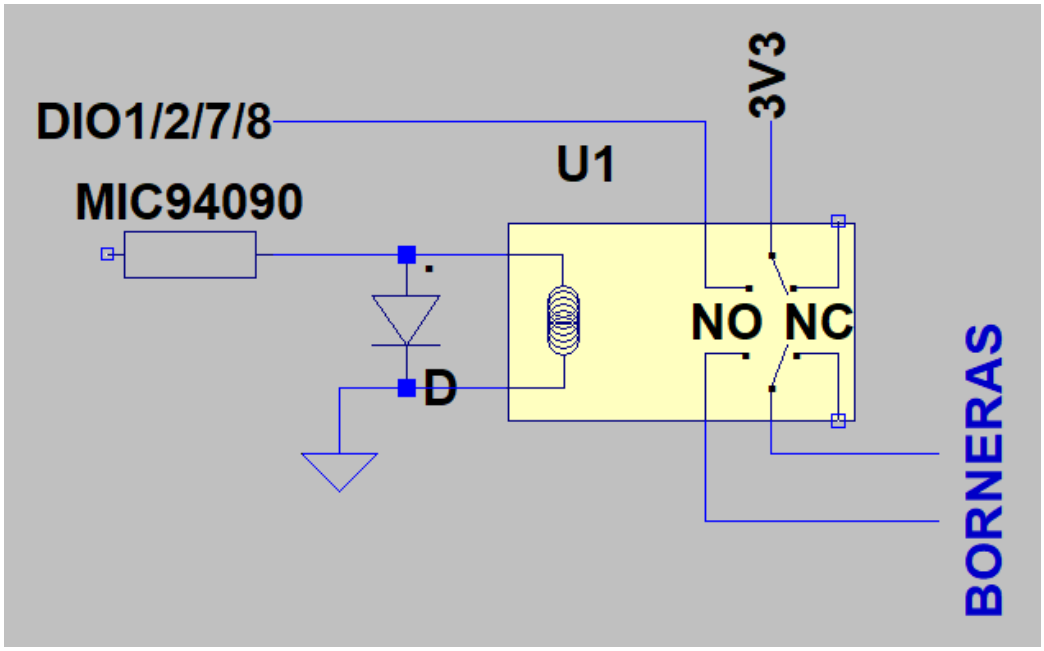


Figura 4.4: Circuito de relés.

Destino	Cantidad
GPIOs	15
Relés	8
Entrada 24 V_{DC}	2
Salidas GND	6
Salidas 5 V_{DC}	2
Salidas 3,3 V_{DC}	5
Pin nRESET	1

Tabla 4.3: Distribución de borneras atornillables según su uso.

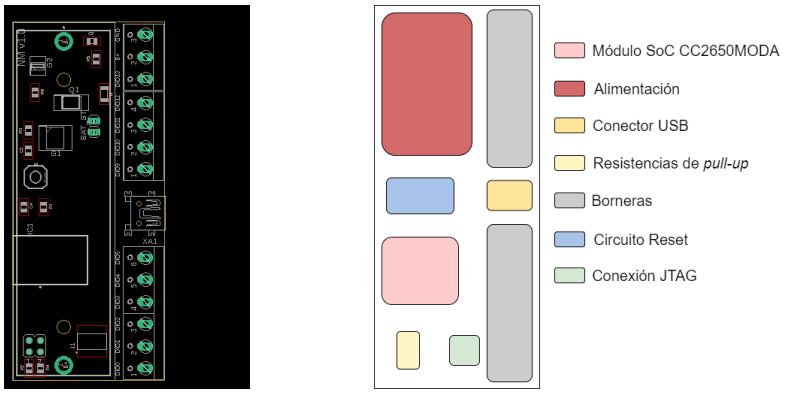
JTAG:

Se incluye un conector FTSH de 10 pines para conexión JTAG, que facilite los procesos de programación de la placa. Siendo que la comunicación JTAG (y por ende los pines involucrados) será utilizada sólo durante la programación, y no en régimen de trabajo normal, los DIO 5 y 6 (utilizados en el conector) son también ruteados a borneras, permitiendo que el instalador los asigne a otros usos.

4.2.3. Nodos Maceta

Esta placa está diseñada para manejar los sensores de humedad y temperatura de suelo. Se apuntó a un diseño versátil, que permita una fácil disposición del equipo en la maceta y una fácil movilidad de la maceta dentro de la sala. En

ese sentido, buscando una independencia física de cableados, se implementa una alimentación por batería. El tamaño de la placa es considerablemente reducido respecto a la UCD, adecuado para incorporar un soporte para baterías y las borneras atornillables suficientes, además del módulo SoC, el circuito de carga de la batería, y otros componentes de la placa. En las figuras 4.5a y 4.5b se muestran los bloques funcionales de la placa y su disposición.



(a) (b)

Figura 4.5: Bloques funcionales de NM.

Alimentación:

En esta placa se implementa una alimentación mediante baterías recargables de tipo 18650. Para ello se incluye un circuito de carga de baterías basado en el integrado TP4056 [28]. Este circuito se alimenta de 5 V y puede cargar una batería a corriente constante y voltaje constante (4.2 V), para lo que se incluye un puerto mini USB de tipo receptáculo. El circuito de carga toma como entrada los bornes B+ y B- de la placa, correspondiente a los polos positivo y negativo de la batería, y devuelve una salida B+ y GND, desde donde se alimentan el resto de los componentes y los elementos externos. Cabe aclarar que, debido a la composición del TP4056, si bien los bornes B+ de entrada y salida coinciden, B- y GND se encuentran separados, pudiendo existir una diferencia de tensión entre ellos.

Borneras:

Las borneras dispuestas permiten conectarse con casi todos los GPIO (DIO del 0 al 13) y acceder a la alimentación desde los bornes B+ y GND. En el caso de los NM, se espera el uso de menos cantidad de sensores, por lo que el número de borneras atornillables se reduce de 39 a 13. Esto también permite reducir el tamaño de la placa, haciéndola más adaptable para ubicarla en una maceta.

Protocolo I2C:

Varios modelos de sensores utilizan el protocolo de comunicación I2C para interactuar con el microprocesador. Por esta razón, se agregan resistencias de *pull-up* en los DIO 2 y 3 junto con sus *jumpers*, dejando al instalador la opción de utilizarlas o no.

4.2.4. Ensamblado de Placas

Luego del proceso de diseño de las placas, se contacta con el fabricante realizando el pedido de 5 unidades de cada modelo con sus respectivos esténciles. También se realiza la compra de los distintos componentes, tanto en el exterior como en plaza. Una vez obtenidos todos los componentes y las placas, se obtiene el permiso del Departamento de Electrónica de la Universidad Católica del Uruguay, para el uso de su laboratorio de Diseño y Ensamblaje de Circuitos Eléctricos (DECE), que cuenta con una máquina de *pick and place* y un horno de reflujo. En la figura 4.6 se observan ambos equipos en el laboratorio.

En este laboratorio se completa el soldado de los componentes de tipo SMD (*Surface Mounted Device*). El proceso comienza aplicando pasta de soldar sobre las placas utilizando el esténcil y una espátula, de forma que la pasta se acumule únicamente en las almohadillas de contacto (figura 4.7). Luego se posicionan cuidadosamente los componentes en sus espacios designados, para finalmente colocarlos en el horno de reflujo. Este horno consiste en una pasarela y una rejilla que mueve las placas hasta el extremo de salida. El horno cuenta con 6 resistencias, con temperaturas configurables de forma individual, que permiten generar una curva de calor sobre la placa a medida que avanza por la pasarela. Una vez culminado el proceso de horneado, se verifica que los componentes hayan quedado bien posicionados y con buen contacto. En las figuras 4.8a y 4.8b se muestran las diez placas con sus componentes SMD soldados.

Finalmente, se procede a soldar los componentes *through hole*, pasando luego a las pruebas de funcionamiento. En la figura 4.9 se muestra el resultado.

En nuestro caso, se dio la situación de no conseguir stock de ciertas medidas de un diodo, por lo que se adquirió un modelo similar. Sin embargo, descubrimos que, a pesar de que la diferencia de tamaño era de 1 mm, el componente no podía ser soldado con el horno de reflujo. Esto se debe a que con el calor, y al fundirse la pasta de soldar, el componente se acomoda sobre las almohadillas. Al no ser compatible con las almohadillas diseñadas, el componente se soldaba solo en uno de los extremos. Para solucionar esta situación, se implementa un puente desde uno de los bornes del diodo hasta el hueco del elemento *through hole* con el que debía conectarse. En las figuras 4.10a y 4.10b se muestra el resultado.

Figura 4.6: Laboratorio DECE de la UCU con los equipos Pick and Place y horno de reflujo.

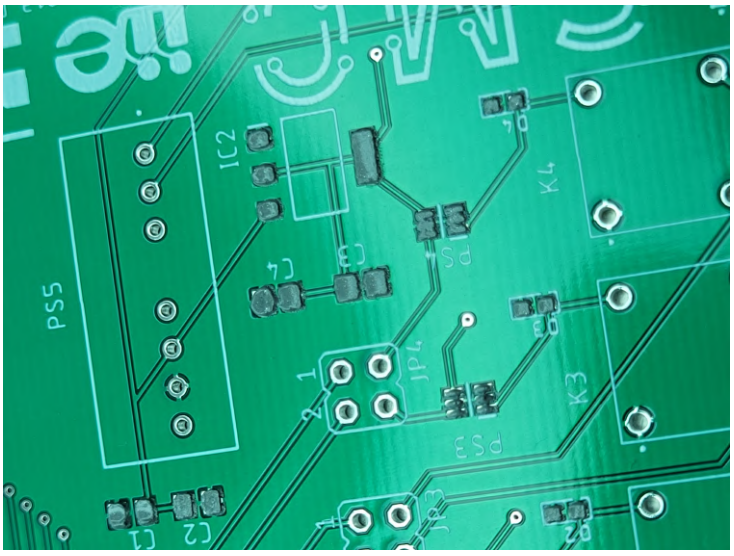
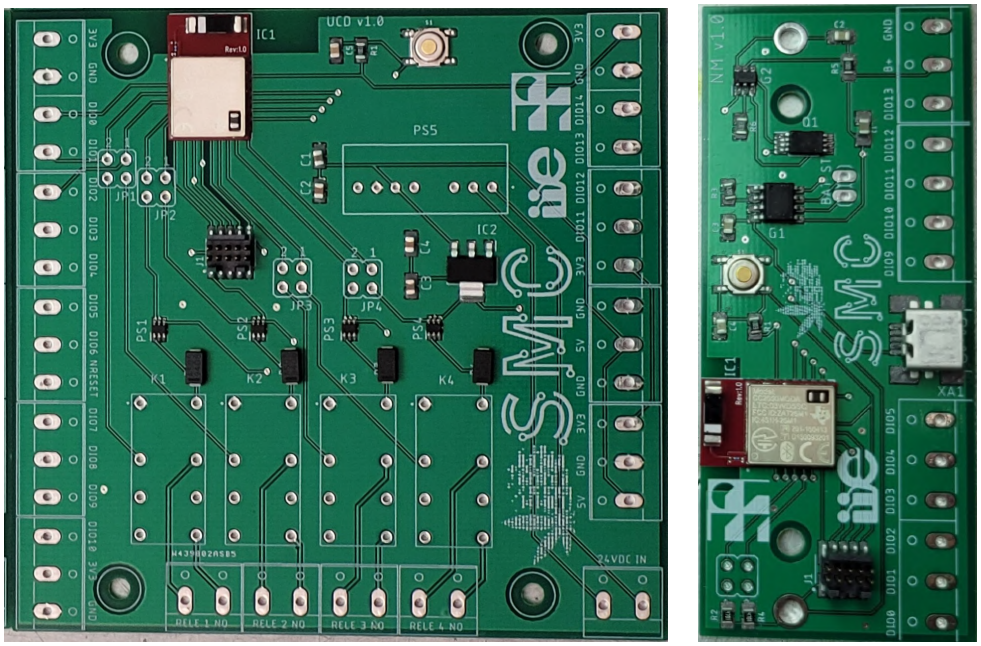


Figura 4.7: Pasta de soldar sobre almohadillas de placa UCD.

4.3. Sensores

De acuerdo a la función que cumple cada sala, se propone un conjunto de sensores por cada una. En líneas generales se prioriza buscar sensores adquiribles en plaza, de fácil instalación y con suficiente información técnica.

En esta sección se describe el modelo utilizado, su elección y detalles de conexión.



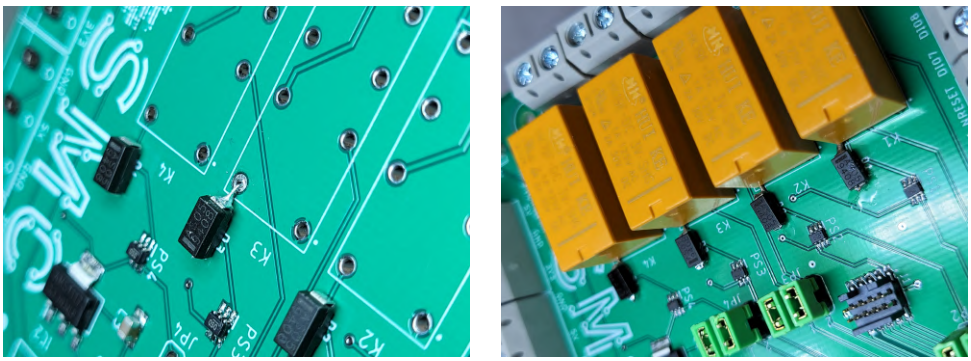
(a) UCD (b) NM

Figura 4.8: Resultado de placas luego del proceso de soldadura de componentes SMD por horno de reflujo.

Figura 4.9: Placa UCD con todos sus componentes soldados.

4.3.1. Humedad y Temperatura Ambiente

Los sensores normalmente utilizados para estas variables son los denominados DHT-11 [6] y DHT-22. Ambos tienen la misma base de funcionamiento, pero presentan diferencias en características de precisión y consumo, como se indica en la



(a) (b)

Figura 4.10: Solución alternativa para conexión de diodo.

talba 4.4. Dado que el modelo DHT-11 es el de menor consumo y su precisión de 0.1 °C es suficiente para nuestra aplicación, es el elegido para este proyecto.

	DHT-11	DHT-2
Precio	256	615
Rango T	0-50 °C	-40-80 °C
Precisión T	±2 °C	±0,5 °C
Rango H	20-90 %	0-99,9 %
Precisión H	±4 %	±2 %

Tabla 4.4: Comparación entre sensores DHT-11 y DHT-22.

El DHT-11 está basado en un termistor de tipo NTC (que cambia su resistencia según la temperatura a la que es sometido) y un sensor de humedad ambiente resistivo. Cuenta con un ADC de 16 bits integrado y un microcontrolador de 8 bits, en cuya memoria se guardan (de fábrica) los coeficientes de calibración.

Este sensor implementa un protocolo de comunicación de propietario, de tipo sincrónico y en un único bus.

Su empaquetado, como se observa en la figura 4.11, viene presentado en una cápsula de plástico, afirmada en una placa donde se encuentran los sensores, su circuito digitalizador y los cuatro pines, que permiten una fácil conexión.

4.3.2. Iluminación

Como se explicara en secciones anteriores, la luz es la variable de mayor importancia en el crecimiento de las plantas. Actualmente el cliente cuenta con un sistema que permite controlar y monitorear encendidos y apagados, consumos y potencia. De todas formas, se incluirán sensores de luz con el fin de corroborar y registrar los horarios y cantidad de horas de iluminación.

En este caso, se utilizan módulos basados en sensores de tipo LDR (*Light Dependent Resistance*) [24], como los de la figura 4.12. Tomando las salidas digitales

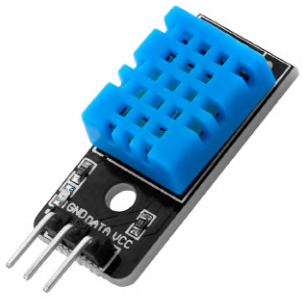


Figura 4.11: Sensor de humedad y temperatura ambiente DHT-11.

de estos, se detecta si las lámparas se encuentran encendidas o apagadas. De esta forma, se puede tener una confirmación de la cantidad de horas de luz y de oscuridad que tiene cada día, consultando los datos en los dashboards de seguimiento.

En cuanto a los sensores, como indica su nombre, consisten en una celda fotorreceptora que cambia el valor de su resistencia según la intensidad de la luz que le incide. El módulo cuenta también con un integrado comparador LM393 y una resistencia variable para ajustar el umbral de comparación.

Los módulos son alimentados con 3,3 V y presentan una salida digital y una analógica. La salida digital indica el resultado del comparador, desactivándose cuando el nivel de luz supera un umbral definido mediante la resistencia variable. La salida analógica corresponde directamente a la caída de tensión en la celda fotorreceptora, resultando entonces en una salida proporcional a la iluminancia, que varía en un rango entre 0 y 3,3 V.

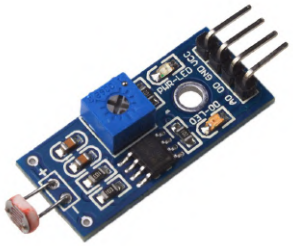


Figura 4.12: Sensor de iluminación basado en LDR.

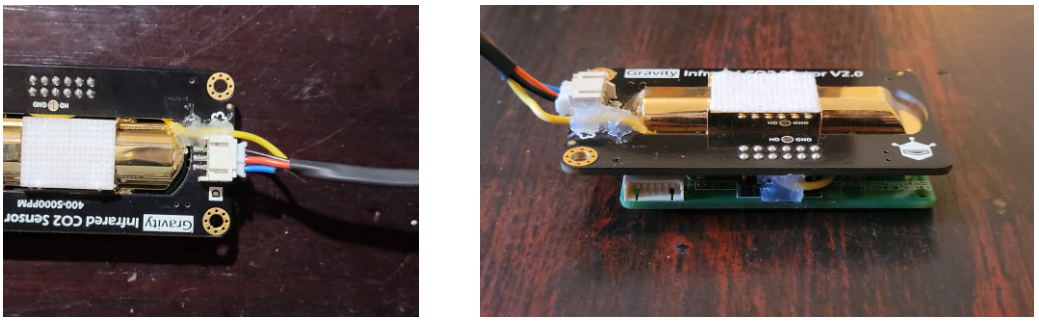
4.3.3. Concentración de CO_2

La concentración de CO_2 en el ambiente es una variable importante para el proceso biológico de la planta, en particular para la fotosíntesis. Para monitorear esta variable se utiliza un módulo sensor denominado SEN0219.

Este módulo utiliza un sensor MH-Z14 [80] con tecnología NDIR (Non-Dispersive InfraRed) para la medida, que le aporta una mayor precisión a la lectura. El módulo tiene un rango de entre 400 y 5000 ppm con una precisión de ± 50 ppm y 3 % de

la lectura. Además, el módulo es de bajo consumo (430mW a 5V), tiene más de 5 años de vida útil estimada, compensación por temperatura y una salida analógica que varía entre 0,4 V y 2 V.

El módulo solo necesita tres pines para conectarse: GND, VCC y Signal, y se alimenta con 5 V. Cuenta con una salida principal correspondiente a una señal de tipo PWM y con salidas secundarias, de tipo analógicas y UART. En nuestro caso utilizaremos la salida analógica, para lo que debimos modificar el módulo. Se implementa un puente cableado entre el pin de salida analógica y el cable de la salida Signal, desconectando este último de la placa. El resultado se observa en la figura 4.13.



(a) (b)

Figura 4.13: Módulo sensor de CO_2 intervenido para obtener la salida analógica.

4.3.4. Humedad de Suelo

Los sensores de humedad de suelo están involucrados directamente en el proceso de riego, indicando el momento y la cantidad de agua que la planta requiere ser regada.

Se buscaron sensores que trabajaran con bajos niveles de tensión (3,3 V), de fácil instalación y una medida relativamente confiable. Se proponen dos modelos de diferente tecnología, siendo uno de ellos capacitivo (SEN0193 [26]) y otro de tipo resistivo (FC-28 [34]). Ambos son ampliamente utilizados en proyectos de tipo Arduino, por lo que cuentan con documentación, soporte en comunidad y ejemplos de aplicación. En el anexo A.1 se plantea un análisis comparativo entre estos dos sensores, a partir del cual se decide utilizar el sensor capacitivo SEN0193.

Este sensor permite alimentarse en el rango de 3,3-5,5 V, con una salida entre 0 y 3 V. El SEN0193 utiliza la humedad de su entorno para hacer variar la capacidad de un condensador, generando una señal de tensión continua, en relación lineal con el valor de humedad. En la figura 4.14 se puede observar un esquemático del circuito interno, donde el integrado TLC555 [59] es un *timer* que genera pulsos a una frecuencia dada, de hasta 2.1 MHz. En la salida de este integrado se encuentra el capacitor variable, que genera una señal de exponenciales periódicas, correspondiente a la carga y descarga del condensador. Luego, mediante la carga de otro

condensador, agregando un diodo para evitar su descarga (y otros componentes de regulación de la señal), se obtiene una salida DC de nivel variable, proporcional a la humedad del suelo.

En la sección A.2 del anexo se detalla el proceso de calibración y parametrización del sensor.

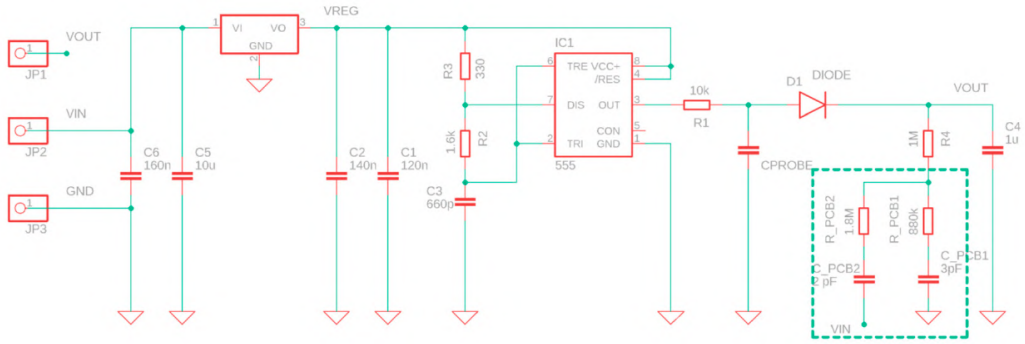


Figura 4.14: Diagrama esquemático del circuito interno del sensor SEN0193.

Como se observa en las figuras 4.15a y 4.15b, el circuito está anexado al sensor, por lo que queda expuesto al agua de riego. Se debe colocar una protección para evitar cortocircuitos y roturas por mojadura de los componentes del circuito. En nuestro caso utilizamos un cerramiento plástico fabricado con impresora 3D, sellándolo con silicona. En la figura 4.15c se observa el resultado.

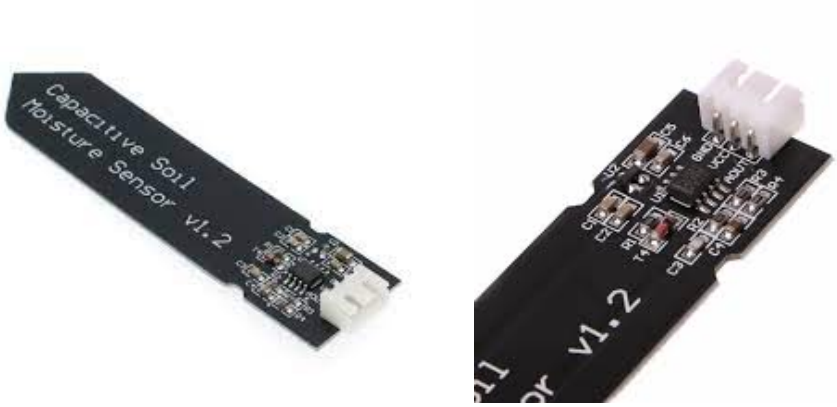
4.3.5. Temperatura de Suelo

Junto con la humedad del suelo, los NM miden temperatura del mismo. Debido a su uso extendido y la disponibilidad de soporte y ejemplos, el sensor elegido es una sonda basada en el integrado DS18B20.

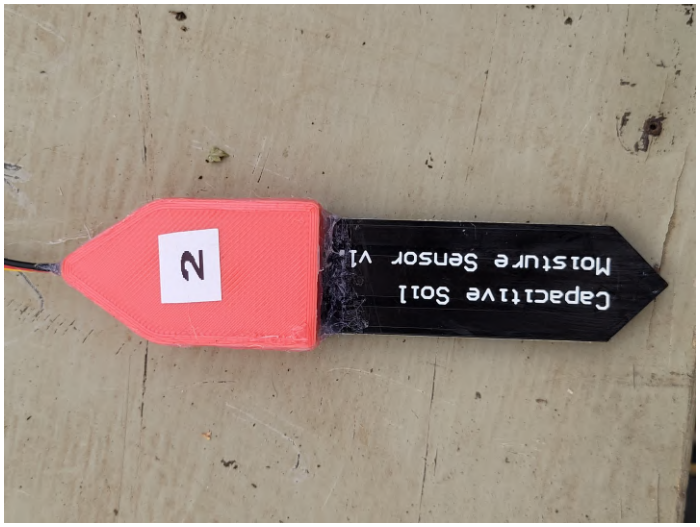
Este sensor consiste en el integrado mencionado, con un empaquetado de tipo TO-92, que cuenta con tres pines: GND, VCC y DATA. Estos pines se extienden mediante un cable tripolar de aproximadamente 1 m. Finalmente, el integrado es cubierto por una cápsula de aluminio y sellado con tubo termocontraíble para evitar el contacto con agua, como se observa en la figura 4.16.

El DS18B20 utiliza One Wire (1W) como protocolo de comunicación por lo que no precisa más que un cable para datos, además de la alimentación. De todas formas, este integrado debe ser operado mediante un microprocesador que sea capaz de enviar los comandos necesarios de forma sincrónica, tarea que realiza el NM.

Este sensor puede ser alimentado con cualquier tensión entre 3,0 V y 5,5 V. Permite una lectura de entre 9 y 12 bits (configurable), en un amplio rango de temperaturas de entre -55 °C y +125 °C, con una apreciación de ±0,5 °C en el rango entre -10 °C y +85 °C.



(a) (b)



(c)

Figura 4.15: Imagen del sensor de humedad capacitivo y su protección.

Figura 4.16: Sensor de temperatura de suelo.

4.3.6. Nivel de pH

Como se explicara anteriormente, el agua recibida de la red de suministro público tiene ciertas condiciones poco saludables para la planta, en particular su nivel de pH, por lo que se debe ajustar esta variable.

El sensor elegido es el SEN0169-V2 de DFRobot, en su versión Pro, que consiste en una sonda de grado industrial (figura 4.17a), junto con una placa de conversión de señal (figura 4.17b).

La sonda en cuestión puede medir todo el rango de pH (0-14) con una precisión de ±0,1 pH a 25 °C. Además, la lectura es rápida, estableciéndose en menos de un minuto, y puede usarse en sumersión continua con una vida útil aceptable, de medio año de extensión.

Por su parte, la placa de conversión de señal cuenta con un conector BNC para leer la salida de la sonda y un circuito para procesar la señal de la misma, reduciendo el ruido, entre otras funciones. La placa utiliza tres pines, tanto para alimentarse (3,3-5 V) como para comunicarse de forma analógica con el microcontrolador correspondiente [25].

(a) (b)

Figura 4.17: Componentes de la medición de pH: sonda y placa de conversión de señal.

En nuestro caso, la sonda será utilizada bajo agua de forma constante, evitando manipulaciones de operarios en cada medición. Para ello se construye un flotador de poliestireno, asegurando que el sensor se encuentre siempre en contacto con el agua a la vez que se mantenga sobre la superficie.

4.3.7. Caudal de Agua

En el manejo de la planta de agua se utilizan dos caudalímetros, correspondientes a la entrada y salida del tanque.

Por disponibilidad en plaza y popularidad en proyectos relativos a Arduino y similares, se tienen en cuenta los sensores YF-S201 e YF-S401. Estos caudalímetros funcionan con un sensor de efecto Hall, que detecta el movimiento de las paletas

del molino interno al ser impulsadas por el agua, y emite pulsos proporcionales al volumen de agua que atraviesa el sensor.

Los dispositivos nombrados son esencialmente iguales, con algunas diferencias funcionales. Básicamente, el modelo 201 es un sensor de mayores dimensiones, tanto en sección de tubo como en rango de caudal. En la tabla 4.5 se detallan las características de ambos caudalímetros [37].

Característica	YF-S201	YF-S401
Rango de alimentación (V_{DC})	5-18	5-24
Rango de medición (L/min)	1-30	1-5
Presión máxima (kg/cm^2)	17,85	
Error (L/min, lectura)	$\pm$0,17 + 5 %	$\pm$0,03

Tabla 4.5: Características de caudalímetros.

Se decide entonces utilizar el sensor YF-S201, ilustrado en la figura 4.18, entendiendo que sus características son las que se ajustan mejor a nuestra aplicación.

De acuerdo al fabricante, este sensor genera entre 480 y 490 pulsos por litro de agua.

Figura 4.18: Caudalímetro YF-S201.

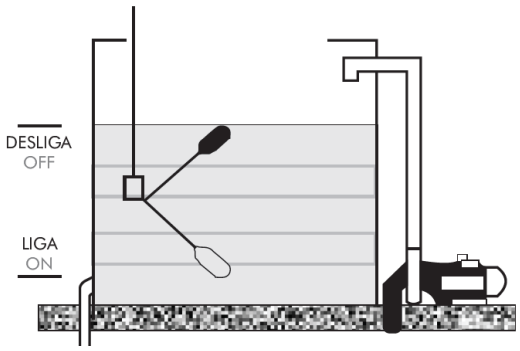
4.3.8. Sensor Boya

Para regular el llenado del tanque de agua, se debe tener certeza del nivel de agua del mismo, evitando derrames. Se incluye entonces un sensor de tipo boya para señalizar el nivel alto del tanque.

El sensor elegido es el denominado CB-1 del fabricante Metaltex [39], disponible en plaza y de fácil instalación y operación. Este dispositivo, mostrado en la figura 4.19b, consiste en un componente flotante con un contrapeso, un interruptor y un par de contactos secos NC y NO. De esta forma, el ángulo que surge entre el

componente flotador y el contrapeso es la variable que gobierna el interruptor, que activa los contactos secos, como se indica en la figura 4.19a.

La boya se conecta a un relé externo de tipo DPDT, que permite duplicar la salida del sensor boya. Uno de los circuitos de este relé se conecta a una de las entrada del microprocesador de la UCD-R, mientras que el otro es conectado en serie en la alimentación de la electroválvula de llenado de tanque. Es así que, además de indicar a la UCD-R el estado del tanque, se dispone un enclavamiento (*interlock*) en la electroválvula.



(a) Esquema de funcionamiento

(b) Sensor de tipo boya CB-1.

4.4. Actuadores

El sistema cuenta con algunos actuadores, dispositivos capaces de generar una acción para modificar una variable en base a una medida previa de la misma.

4.4.1. Electroválvulas

El manejo confiable de los flujos de agua resulta esencial en el proceso de riego. Se necesita para ello poder controlar el pasaje del fluido en ciertos puntos.

En primer lugar, se tiene la válvula destinada al llenado del tanque de agua. Esta válvula es manejada por la UCD-R, que ordena el ingreso de agua desde la red de suministro al terminar cada ciclo de riego. Se cuenta también con un circuito de prevención de derrames mediante la señal del sensor de nivel de agua, que cierra la válvula en caso de ser necesario.

Por otra parte, se tienen las válvulas dedicadas al riego de las macetas. Estas válvulas están incluidas en el circuito de recirculación de la planta de agua, por lo que al activarse permiten que una porción del flujo de agua se desvíe hacia las macetas correspondientes. Se cuenta con dos electroválvulas, cada una encargada de un pequeño grupo de macetas de la SF, definidos por el operario al renovar las plantas de la sala.

Se buscaron modelos disponibles en plaza, de bajo costo y fácil instalación. Además, las válvulas deben ser capaces de soportar la presión ejercida por la bomba. Se analizaron entonces las opciones resumidas en la tabla 4.6.

Modelo	Tensión (V)	Alimentación	Consumo (mA)	Sección (in)	Presión Máx. (bar)
2P025-08	24	DC	125	1/4	7
2P025-08	220	AC	29	1/4	7
2V025-06	24	DC	29	1/8	8
2V025-08	24	DC	125	1/4	8
2V025-06	220	AC	29	1/8	8

Tabla 4.6: Comparación de electroválvulas.

El modelo elegido es el 2P025-08 de 220AC de la figura 4.19, por su costo y calidad, y por su tensión de trabajo, dado que facilita su manejo. Además, tiene un consumo reducido que permite su manejo mediante los relés de la UCD-R.

Figura 4.19: Electroválvula para sistema de riego.

4.4.2. Bombas

Bomba de Recirculación

El cliente dispone de una bomba sumergible Hyundai HYSP750, de 1 HP de potencia, con la que suele realizar el riego manual. Se utilizará esta misma bomba para el circuito de recirculación de agua, mediante el que se hará llegar el agua de riego a las macetas.

La bomba es capaz de generar un caudal de hasta 200 L/min, dependiendo de la altura de la salida de agua. Se indica que pueden ser sumergidas completamente hasta una profundidad de entre 5 y 8 m. En nuestro caso la bomba será utilizada

a nivel del suelo de la sala SR, de forma que la altura máxima a la que trabajará será de 2 m. Además cuenta con salidas para cuatro secciones: 5/4", 1", 3/4" y 1/3".

Se presenta el equipo en la imagen de la figura 4.20.

Figura 4.20: Bomba de recirculación.

Bomba de Dosificación

Para el ajuste del nivel de pH del agua se debe manejar la dosificación del ácido fosfórico, líquido reductor de pH. Para ello, dado que se utiliza poco volumen de este químico, se define utilizar una bomba peristáltica, ilustrada en la figura 4.21. Las bombas peristálticas se utilizan para desplazar fluidos por un conductor flexible, comprimiéndolo y empujando el fluido hacia adelante.

La bomba elegida es ideal para dosificación por su capacidad de bombear pequeños volumenes con buena precisión, además de su tamaño reducido de aproximadamente 5x5 cm. La tensión nominal es de 6 V_{DC} y suele consumir unos 200-300 mA. Es capaz de generar un caudal de entre 0 y 100 mL/min girando entre 0,1 y 100 rpm.

Figura 4.21: Bomba peristáltica de dosificación.

De acuerdo a los ensayos descriptos en el anexo A.7, alimentada con los 5 V de nuestra placa la bomba dosifica aproximadamente 1,18 mL/s en un rango de

al menos hasta 1 minuto. En el ensayo mencionado se determina un modelo lineal al cual se ajusta de buena manera el comportamiento de la bomba. El modelo se representa gráficamente en la figura 4.22.

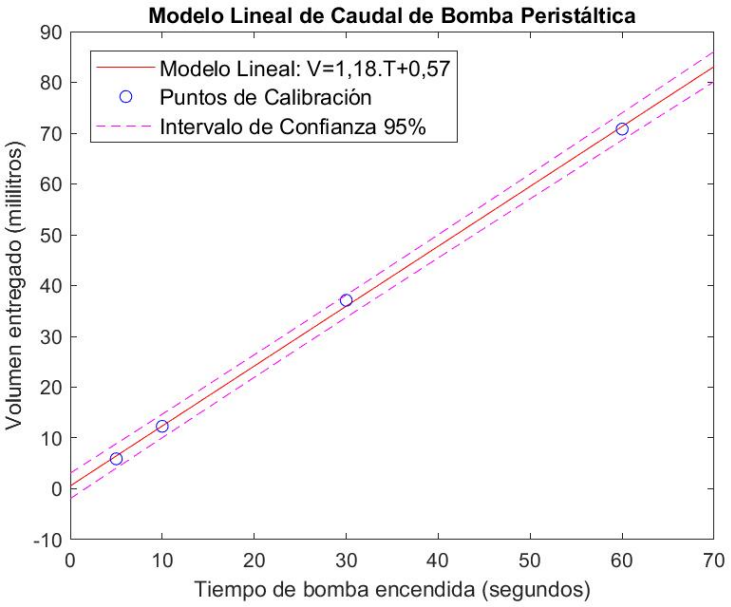


Figura 4.22: Modelo lineal de bomba peristáltica.

4.4.3. Relés

Los relés son dispositivos eléctricos que permiten cerrar o abrir circuitos bajo carga o sin carga. Funcionan activando una bobina para abrir o cerrar contactos. Si bien las UCD cuentan con relés que permiten maniobrar circuitos eléctricos, se requiere de relés con mayor capacidad para manejar equipos de mayor potencia.

Se utilizan módulos de relés premontados de Phoenix Contact de 2 y 4 contactos inversores. Su bobina trabaja a 24 V de tensión y consumen hasta 18 mA para 2 contactos y 42 mA para 4 contactos. Sus bornes admiten tensiones de hasta 250 V_{AC} o V_{DC} y corrientes constantes de hasta 8 y 6 A. Ambos modelos se pueden observar en las figuras 4.23a y 4.23b.

Serán instalados en todas las salas para manejar el encendido y apagado de los aires acondicionados (SV y SF), la bomba de recirculación (SR), y el deshumidificador (SS).

(a) Relé de 2 contactos.

(b) Relé de 4 contactos.

4.5. Otros

4.5.1. Tableros

Si bien todo el equipamiento a instalarse será colocado dentro de las distintas salas, se debe disponer de receptáculos que resguarden los dispositivos. Tanto las placas UCD como la fuente de 24 V, relés y demás componentes, deben protegerse de humedad, polvo y acceso de personas no calificadas.

Las cajas seleccionadas son de tipo estanco, con grado de protección IP65. Su cerramiento es mediante cuatro tornillos ubicados en las esquinas de la caja. En el interior de la misma se afirma la UCD correspondiente a la sala y un riel de tipo Din. Sobre este riel se colocan los siguientes elementos:

- Fusibles: los dos fusibles instalados están destinados a proteger eléctricamente a la placa electrónica y
- Fuente 24 V: la fuente se alimenta del suministro de 220 V_{AC} y los transforma en 24 V_{DC} para alimentar la placa y otros componentes.
- Relés: en algunas salas se requiere el uso de relés externos, de mayor capacidad que los incluidos en las UCD.

Sobre la pared frontal de la caja se instalan las llaves selectoras, de forma que el interruptor queda accesible del lado de afuera, mientras que en el interior se conecta a la UCD y la fuente.

4.5.2. Llaves Selectoras

El sistema desarrollado permite manejar equipos de forma automática, sin embargo cabe la posibilidad de que el operario prefiera el uso manual de alguno de ellos. Se plantea entonces incorporar una herramienta que presente la opción de

desconectar el automatismo y controlar de forma manual el actuador deseado. Para ello se incluyen llaves selectoras en todas las UCD de las salas.

Las llaves selectoras son accionamientos mecánicos que abren o cierran contactos según la posición en la que se encuentren. En este caso se instala una llave selectora de 3 posiciones en cada tablero permitiendo elegir las posiciones *automático*, *manual* y *apagado*.

Los contactos de este tipo de llaves tienen un funcionamiento inverso. Esto es, el contacto de la posición donde se encuentra la llave está cerrado, mientras que los restantes se encuentran abiertos.

De esta forma, se conecta la posición *manual* en serie con la fuente y el equipo a manejar, de forma que al activarse pueda encenderse el equipo sin intervención de la UCD. Por su parte, al seleccionar la posición *automático*, el circuito anterior pasa a estar abierto, por lo que el encendido del equipo depende del control de la UCD. A su vez, los contactos de la posición *automático* se utilizan como señal de estado de la llave, siendo que se sube el nivel (a 3,3 V) cuando se selecciona dicha posición, y se baja a 0 al seleccionar alguna de las posiciones restantes.

Se agregan también dos llaves selectoras de 2 posiciones, con las opciones de *automático* y *apagado*. A diferencia de las de 3 posiciones, los contactos de estas llaves conmutan en conjunto, de forma tal que ambos se encuentran abiertos o cerrados en función de la posición de la llave.

La conexión eléctrica es similar a las llaves anteriores, siendo que los contactos de la posición *apagado* se conectan en serie con la alimentación del equipo, mientras que los de la posición *automático* son utilizados para dar aviso a la UCD del estado de la llave. Es así que cuando la llave se posiciona en *apagado*, los contactos se abren, desconectando el equipo de su alimentación y bajando el nivel de la entrada de la UCD. Por otro lado, al pasar a la posición *automático*, los contactos se cierran, con los consecuentes efectos inversos.

Por cada UCD se instala una llave de 3 posiciones, mientras que en las UCD-R se agregan otras dos llaves de 2 posiciones. Estos últimos están destinados a la bomba de dosificación y la electroválvula de llenado del tanque de agua. Se agregan estos modelos debido a que en caso de preferir la actuación manual, se utilizarían otros equipos (*i.e.*, cuenta-gotas o jeringa y llave de paso de agua).

Los actuadores que controla cada llave se detallan en la tabla 4.7.

Manejar la opción de uso manual le permite mayor disponibilidad de los equipos al cliente. En este caso, si por algún motivo falla el automatismo, el cliente no se quedará sin poder utilizarlos, simplemente lo deberá realizar de forma manual.

Sala	tablero	Posiciones	Equipo Controlado
SV	UCD-V	3	Aire Acondicionado
SF	UCD-F	3	Aire Acondicionado
SS	UCD-S	3	Deshumidificador
SR	UCD-R	3	Bomba de Recirculación
			Electroválvulas de Riego
		2	Bomba de Dosificación
			Electroválvula de Llenado de Tanque

Tabla 4.7: Llaves selectoras utilizadas

Capítulo 5

Comunicación

En este capítulo se detalla todo lo relacionado a la comunicación entre las distintas partes del Sistema de Monitoreo y Control. La comunicación no es solamente esencial en la red inalámbrica entre los controladores, sino que sobre este pilar se logra la integración entre los distintos niveles jerárquicos que conforman este sistema.

5.1. Características Generales

En la sección 3, se presenta la arquitectura del sistema, donde se puede tener una visión global del mismo. A nivel de comunicación, el sistema está compuesto por cuatro tipos de nodos, los cuales se indican en la siguiente lista:

- Unidades de Control Distribuido y Nodos Maceta.
- Border Router.
- SCADA.
- Cliente.

Estos distintos nodos constituyen o forman parte de dos redes:

- Red interna.
- Internet.

En el diagrama de red presentado en la figura 5.1, se indican los diferentes nodos y las redes a las cuales cada uno pertenece.

Las Unidades de Control Distribuido (UCD) y los Nodos Maceta (NM), conforman una red interna de comunicación inalámbrica, implementada en 6LoWPAN,

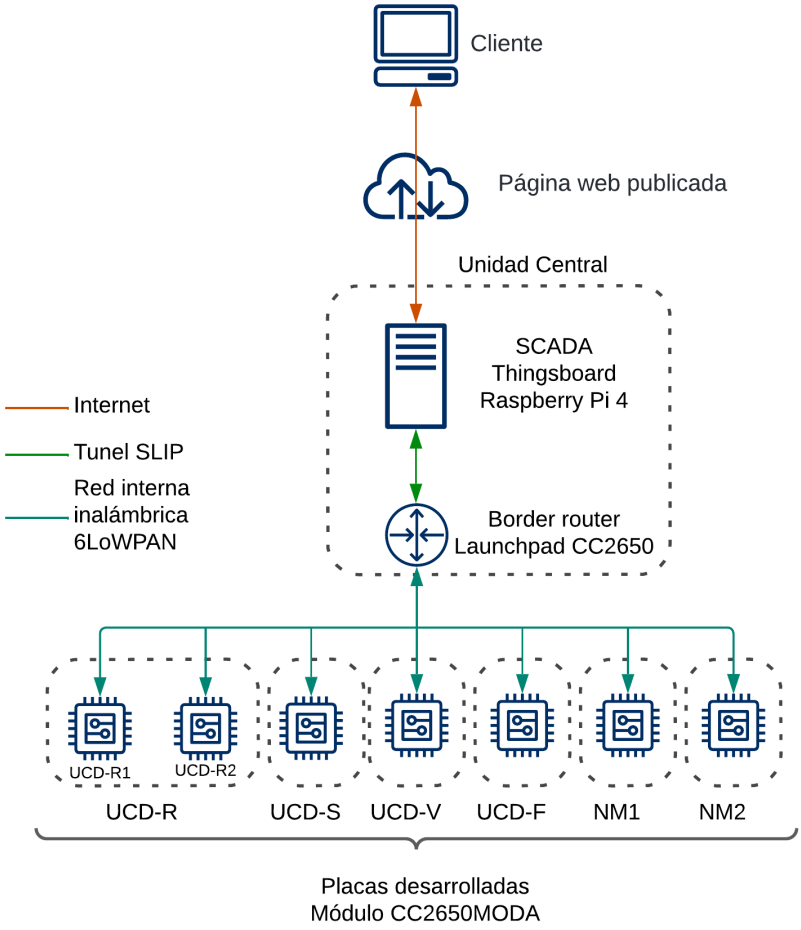


Figura 5.1: Diagrama de red.

con el protocolo CoAP en capa de aplicación. En la sección 5.2, se detalla el stack de protocolos de esta red. Esta red tiene el propósito de permitir la comunicación entre las UCD y los NM, así como integrar el sistema SCADA con estos controladores.

El Border Router (BR) del sistema, además de encargarse del ruteo de la red interna 6LoWPAN, cumple la función primordial de ser el *gateway* entre el servidor SCADA y la red interna de los controladores. Esto es necesario ya que, para formar parte de esta red interna se requiere una radio capaz de comunicarse según el protocolo IEEE 802.15.4 a nivel de capa física, lo cual no es el caso del SCADA.

El SCADA técnicamente se encuentra en una red local, publica su página web a través del *router* de internet del club El Tercer Ojo, y se comunica con los equipos en campo utilizando el BR. Vale aclarar que los equipos que conforman la red de internet (como el *router* de internet), son una solución cerrada cuya composición y funcionalidad excede el alcance de este proyecto. Detalles de cómo se publica la

página web se pueden encontrar en la sección 7.5.

Mediante un navegador (cliente HTTP), el usuario final del sistema puede acceder a la interfaz del SCADA a través de la página publicada. Esto puede realizarse remotamente por internet o directamente al *router*, desde la red local del establecimiento.

5.2. Red interna - 6LoWPAN

La red interna está constituida por los NM, las UCD, el BR y el SCADA. Estos dispositivos se comunican entre sí de forma inalámbrica (a excepción de la comunicación entre el BR y el SCADA), mediante protocolos ajustados al estándar IEEE 802.15.4. En esta sección se detalla cómo está conformado el stack de protocolos de la red interna 6LoWPAN, en base al modelo de 5 capas.

En la figura 5.2, se presenta el Stack de protocolos utilizados en el sistema.

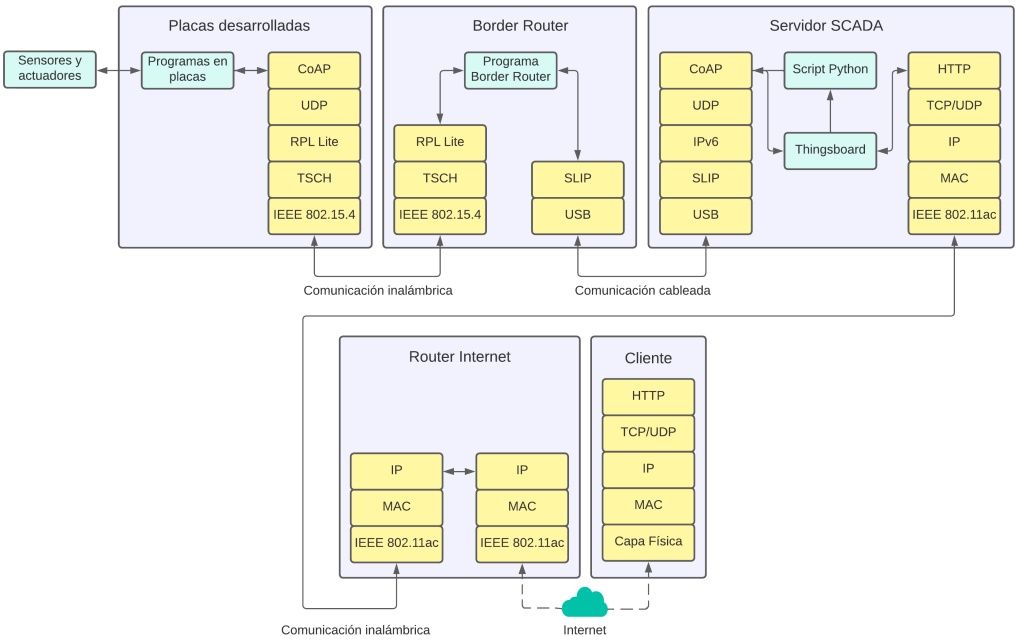


Figura 5.2: Stack de protocolos

5.2.1. Capa Física

En capa física se aplica el estándar IEEE 802.15.4, diseñado para redes inalámbricas de área personal con tasas bajas de transmisión de datos, bajos recursos y bajo consumo [2].

Tanto el BR como las placas desarrolladas UCD y NM, utilizan el SoC CC2650 de Texas Instrument, el cual está diseñado para este tipo de aplicaciones. La radio que utiliza trabaja en la banda de frecuencia 2,4 GHz de uso libre [68].

5.2.2. Capa de Enlace

Como protocolo de capa de enlace se utiliza TSCH (Time-Slotted Channel Hopping), definido por el estándar IEEE 802.15.4e.

Uno de los principales problemas de las redes de comunicación inalámbricas es el comportamiento impredecible del medio, donde se puede dar interferencias en la comunicación. Estas interferencias pueden ser causadas por el intento de más de un dispositivo de comunicarse al mismo tiempo, por ruido aleatorio o por fenómenos físicos como el *Multipath Fading* (interferencia destructiva causada por la propagación de la onda en el medio). Para resolver esto, y proporcionar confiabilidad y latencia a la red, TSCH divide el tiempo y la frecuencia en *slots*. En la figura 5.3, se puede apreciar visualmente esta división [79].

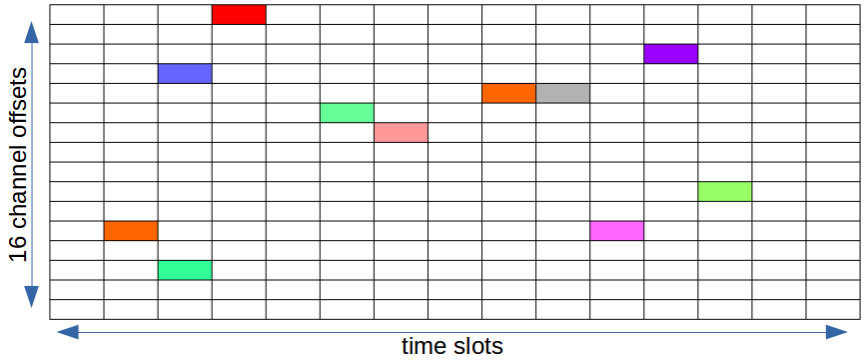


Figura 5.3: *Slots* TSCH. Cada color representa la comunicación entre dos dispositivos [79]

.

Cada *slot* se puede utilizar de forma individual o compartida entre varios nodos. En nuestro caso, se mantuvo la configuración por defecto de Contiki-NG (4 *slots*), donde cada *slot* se utiliza de forma compartida entre todos los nodos de la red [20].

El coordinador de la red (en nuestro caso el BR), se encarga de mantener la red sincronizada. Cuando un nodo busca asociarse al coordinador, realiza saltos aleatorios entre distintos *slots*, hasta encontrarlo. El coordinador envía mensajes breves de *broadcast* (denominados *Enhanced Beacons*), con información sobre la red. Cuando un nodo ve este mensaje, se asocia al coordinador y comienza a enviar *Enhanced Beacons* de forma periódica.

El principal motivo por el cual se decidió utilizar este protocolo es su bajo consumo. El funcionamiento de la red permite mantener apagada la radio, prendiéndola solamente en los momentos necesarios para comunicarse y mantener el sincronismo. El bajo consumo es importante en el caso de los nodos maceta (NM), cuya alimentación es por medio de una batería.
El protocolo CSMA es una alternativa que también se tuvo en cuenta, dado que se utiliza en la configuración de Contiki-NG por defecto. Sin embargo, luego de probarse con los NM se verificó que la batería se agotó en $4,5$ días. Esto se debe

a que en este protocolo la radio se encuentra 100 % del tiempo prendida y el SoC no entra en *Deep LPM*. En el caso del protocolo TSCH, la radio pasa estar 4 % del tiempo prendida y el SoC esta mayoritariamente en *Deep LPM*. Si bien no se comprobó empíricamente la autonomía de la batería con TSCH, se espera que este cambio de protocolo mejore este aspecto.

Desafíos de implementación

Contiki-NG es un sistema operativo que sigue en desarrollo. Si bien en este proyecto se utilizó la última versión publicada a la fecha (versión 4.7), encontramos que TSCH no funcionó correctamente para los SoC CC2650 o CC1350, dado que en primera instancia no fue posible lograr que los SoC se encuentren y se asocien. Se pudo resolver este problema a partir de sugerencias que se encontraron en foros de la comunidad [15] [53].

De esta forma se pudo lograr la comunicación entre SoCs CC2650 y entre SoCs CC1350, pero no entre las dos familias, siendo que los SoC se encuentran y se asocian pero a los pocos segundos pierden la sincronización. Se decide entonces cambiar el *Launchpad* CC1350 con el que se venía trabajando como BR, pasando a utilizar un *Launchpad* CC2650. De este modo se trabaja la misma familia de SoC, evitando los inconvenientes en la comunicación descriptos.

5.2.3. Capa de Red

Esta red utiliza 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) en capa de red. Este protocolo está definido por un grupo de trabajo de la IETF en el RFC 4944 [41], que establece como utilizar IPv6 sobre redes IEEE 802.15.4. Es necesario que la capa de red se adapte a la capa física y de enlace, debido a que estas últimas están diseñadas para dispositivos de bajo consumo, por lo que los paquetes son pequeños. El protocolo 6LoWPAN define cómo realizar la compresión y reensamblado de encabezados, el descubrimiento de vecinos y otras características necesarias para utilizar IPv6.

Dentro de la capa de red, se utiliza el protocolo RPL Lite, en la subcapa de ruteo. Este protocolo es una implementación que surge en Contiki-NG del protocolo RPL (Routing Protocol for Low-Power and Lossy Networks), definido en el RFC 6550 en el año 2012 [5]. Además, es la opción por defecto para la capa de ruteo en Contiki-NG.

El protocolo RPL crea un topología de tipo árbol, denominada *DAG* (Directed Acyclic Graph), como la representada en la figura 5.4. En esta topología, un nodo (o varios) actúa como *DAG root*, al cual conducen todas las rutas (nodo A en la figura). Este DAG puede dividirse en subredes de acuerdo a los distintos *roots* (nodo B en la figura). La subred de cada *root* se llama *Destination Oriented DAG* o simplemente *DODAG*.

Para crear la topología, los nodos intercambian mensajes de control con sus vecinos y determinan cuál vecino es más conveniente para comunicarse con el *root*.

Para ello se utiliza el parámetro *Rank*, que indica la distancia al *root* de cada nodo de acuerdo a cierta métrica definida como *Función Objetivo*. Cuando un nodo elige un vecino mediante el cual comunicarse con el *root*, se dice que lo toma como *padre*. En el ejemplo de la figura 5.4, el nodo G es padre del L, a la vez que el nodo D es padre del G.

A su vez, para cada nodo se define un *Sub-DODAG* donde el *root* es el propio nodo, y en el que están incluidos todos los nodos que lo hayan tomado como padre y el resto de los hijos sucesivos.

El primer mensaje de control enviado por un nodo (no *root*) es el *DODAG Information Solicitation* o *DIS*. En este mensaje, enviado a la dirección *multicast*, se solicita información a los vecinos para así iniciar el proceso de integrarse a un DODAG.

Luego se tiene el mensaje llamado *DODAG Information Object* o *DIO*. Este mensaje es utilizado para crear, formar o mantener actualizado el grafo. Un nodo envía un DIO por múltiples causas: al haber recibido un DIS, al cambiar su *rank*, al cambiar de padre, cada cierto tiempo, etc. En este mensaje se incluyen los campos de configuración del grafo que permiten a un nodo conocer las características del DODAG y unirse de forma correcta. En la figura 5.4, se presenta un diagrama de red a modo de ejemplo [78].

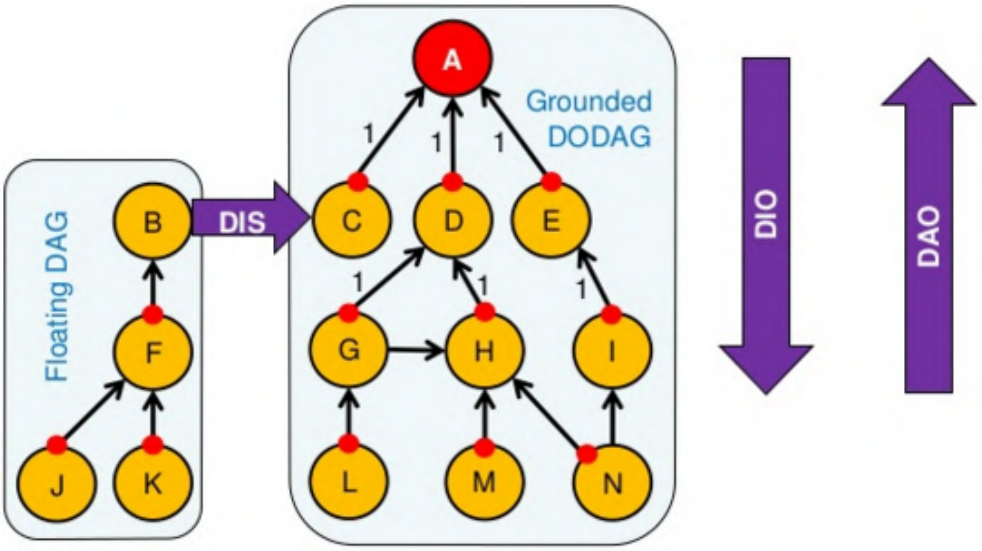


Figura 5.4: Diagrama de red a nivel RPL a modo ejemplo [78].

La implementación RPL Lite, busca enfocarse en implementar una red más estable, simple y robusta. Esta versión solo soporta *non-storing mode*, y remueve la complejidad de manejar multiples instancias y DODAGs. Se logra un mejor rendimiento y un menor uso de memoria ROM.

En el modo *non-storing* solo se almacena información necesaria para llegar al *root*, simplificando la tabla de ruteo y el procesamiento en los nodos. Toda la

comunicación entre los nodos pasa por el *root*.

En nuestro sistema es el BR quien cumple el rol de *root* a nivel RPL.

Como trabajo asociado al presente proyecto en el marco del curso de Redes de Sensores Inalámbricos, se realizó un análisis de RPL Lite enfocado en la comparación con el modo *storing* de RPL Classic. El informe puede encontrarse en el anexo D.

5.2.4. Capa de Transporte

A nivel de capa de transporte se utiliza el protocolo UDP (User Datagram Protocol), documentado en el RFC 768 de la IETF [1]. Es un protocolo no confiable y no orientado a conexión. No se garantiza la entrega de sus mensaje o el orden de los mismos, tampoco se retiene estados de los mensajes enviados a la red. Solamente se añade la información mínima necesaria para la comunicación extremo a extremo. De este modo, se logra un encabezado de tan solo 8 bytes.

Este protocolo es elegido dado que es el único a nivel de capa de transporte compatible con el protocolo que se utiliza en capa de aplicación.

5.2.5. Capa de Aplicación

En capa de aplicación se utiliza el protocolo CoAP (Constrained Application Protocol), definido en el RFC 7252 [51], similar al protocolo HTTP pero diseñado para redes de bajo recursos.

Este protocolo es del tipo REST (Representational State Transfer), por lo que se basa en un intercambio cliente-servidor, donde se dan solicitudes y respuestas utilizando métodos (GET, PUT, POST y DELETE, *et al.*) sobre los recursos del servidor, que son identificados por un URI (Universal Resource Identifier).

Los mensajes pueden ser de uno de los siguientes tipos (ver figura 5.5):

- CON: Confirmable, se espera un respuesta de aceptación del mensaje por parte del receptor. De este modo se logra un protocolo confiable, el emisor espera un mensaje de reconocimiento con el mismo MID (Message ID, identificador del mensaje);
- NON: No confirmable, el emisor no espera un mensaje de reconocimiento.
- ACK: Mensaje de reconocimiento.
- RST: Mensaje de reset, es enviado como respuesta por parte del receptor cuando no es capaz de procesar el mensaje recibido.

En las solicitudes y respuestas se da el concepto de *piggybacked response*, donde en un mensaje ACK se envía la respuesta a la solicitud del cliente, lo que disminuye la cantidad de mensajes necesarios. Sin embargo, también existe la posibilidad de que el servidor conteste la solicitud de forma separada (*separate response*). De

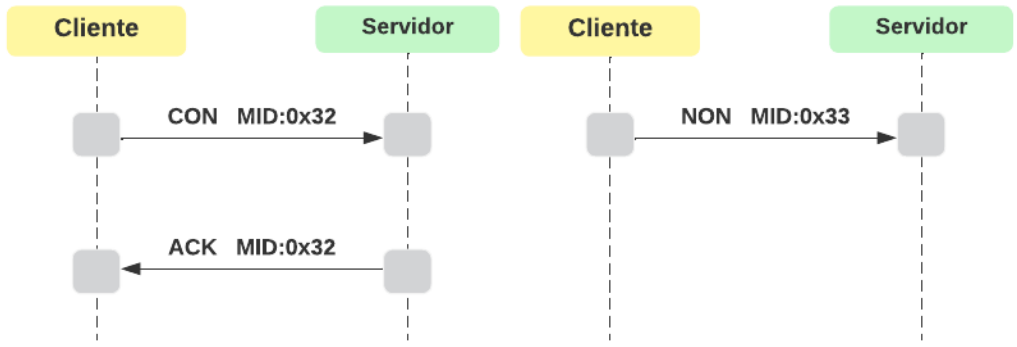


Figura 5.5: Ejemplo tipos de mensajes CoAP.

este último caso surge la necesidad de utilizar un identificador que asocie solicitudes y respuestas, para lo que se utiliza un *token*, como en el ejemplo de la figura 5.6.

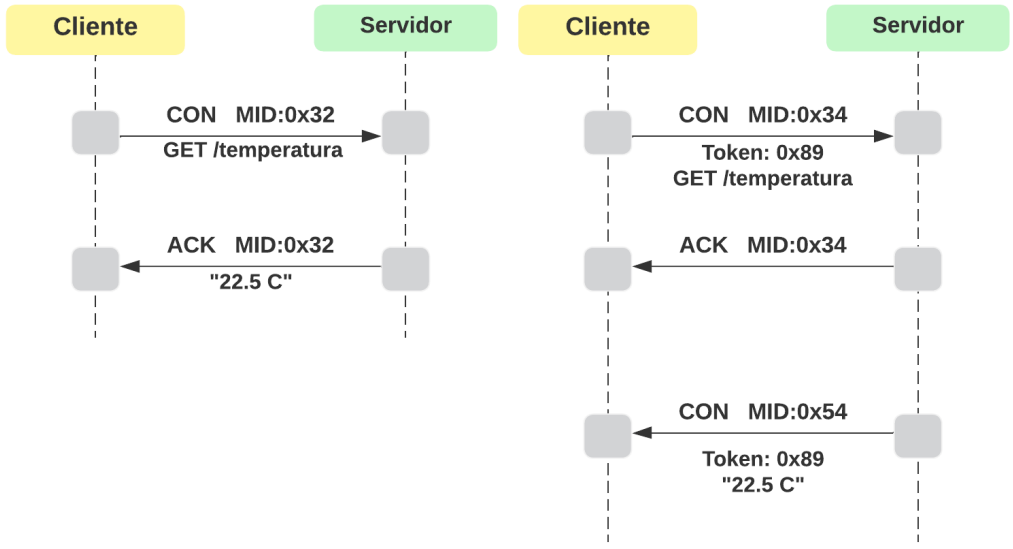


Figura 5.6: Ejemplo *piggybacked* y *separate response*

El protocolo CoAP también admite la posibilidad de que el cliente se suscriba a un recurso del servidor, siguiendo una lógica del tipo *publish/suscriber*. En este caso el servidor notifica al cliente cuando el valor o contenido de ese recurso cambia, o también se puede enviar un mensaje de forma periódica. En la figura 5.7 se presenta un ejemplo.

La plataforma que utilizamos para el servidor SCADA es un servidor CoAP con separate response, por lo que solamente contesta mensajes de forma separada. Además, para enviar comandos hacia los controladores se utiliza CoAP *Observe*, por lo que se requiere que un cliente se suscriba al recurso.

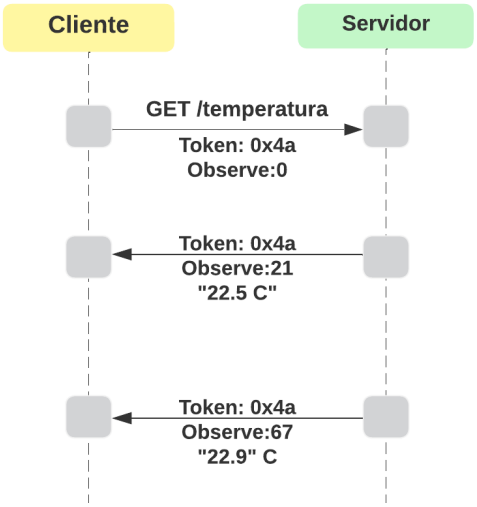


Figura 5.7: Ejemplo CoAP *Observe*

5.3. Aplicación

En esta sección se busca explicar los roles de los distintos nodos que conforman la red respecto a la comunicación.

Como se comentó en la sección anterior, la plataforma Thingsboard que se utiliza como SCADA es un servidor CoAP, por lo tanto no envía mensajes del tipo GET o POST para comunicarse con los NM y UCD. Es así que para lograr enviar los datos de telemetría al servidor SCADA es necesario hacerlo desde clientes CoAP. A su vez, para recibir comandos de la plataforma, es necesario que el cliente se suscriba al recurso de la plataforma, para lo que se utilizan *scripts* intermediarios desarrollados en Python, que cumplen el rol de cliente CoAP. Uno de estos *scripts* se encarga de suscribirse a los recursos y enviar mensajes del tipo POST a los controladores, de forma que las placas que reciben comandos también deben ser servidores CoAP.

De esta manera, los controladores son tanto clientes como servidores, desarrollados en Contiki-NG.

Por otra parte, se tiene también la comunicación entre controladores, llevada a cabo por los clientes y servidores CoAP desarrollados. En la figura 5.8, se indican los clientes y servidores CoAP y el sentido del flujo de mensajes, como ejemplo genérico del intercambio de mensajes.

Desafío de implementación

Como se ha detallado con anterioridad, se utiliza Contiki-NG como sistema operativo de los SoC CC2650. Este sistema operativo cuenta con soporte para varias familias de CPU y plataformas. Se cuenta con código ya desarrollado para

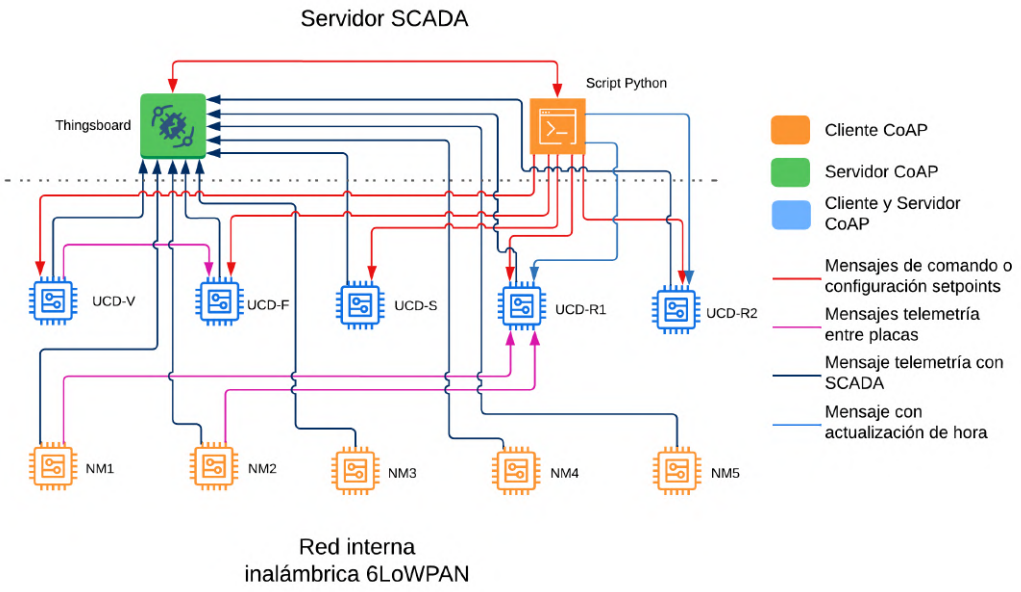


Figura 5.8: Clientes y servidores CoAP

trabajar con la familia de CPU CC2650 RGZ (SoC de 7mm x 7mm). Sin embargo, el módulo CC2650MODA utiliza la versión del chip de 5mm x 5mm, que contiene 15 GPIOs a diferencia de los 31 de la otra versión, además de que el circuito que maneja la radio es distinto. Para poder utilizar este sistema operativo, fue necesario crear una plataforma en el repositorio de Contiki-NG, para lo que nos basamos en las indicaciones de los documentos de Contiki-NG [19].

Lo que se hizo fue utilizar todo el código correspondiente a CPU 7mm x 7mm, y desde la nueva plataforma, que llamamos cc2650moda, se realizó un mapeo de cada GPIO y periférico distinto. Posteriormente se verificó el correcto funcionamiento de la radio y del mapeo.

5.3.1. Intercambio Cliente-Servidor

Normalmente, los modelos REST utilizan recursos que son consultados por el cliente. Sin embargo, en nuestro diseño los recursos son utilizados para retener valores recibidos. Esta implementación fue necesaria para integrar la red interna 6LoWPAN con la plataforma Thingsboard. En ese sentido, las placas que reciben mensajes con valores de telemetría o configuración de parámetros deben ser servidores CoAP, mientras que las placas que envían mensajes con valores de telemetría deben ser clientes CoAP.

Intercambio entre placas

Los mensajes entre placas se envían para que las UCD reciban los valores de telemetría que necesitan para realizar el control. Este tipo de mensajes se envía de

forma periódica, son del tipo CON, método POST y formato texto plano, enviados por la placa que sensa la variable. En la figura 5.9, se ilustra un ejemplo.

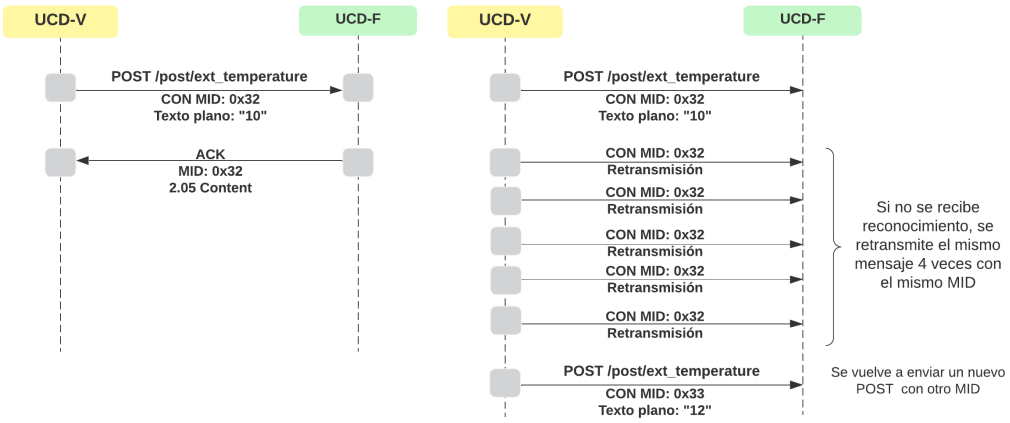


Figura 5.9: Envío de telemetría entre placas

El cliente desarrollado envía un mensaje de forma periódica con la telemetría, para lo que utiliza la función "$COAP_BLOCKING_REQUEST$", desarrollada por Contiki-NG.

Esta función envía un mensaje y se bloquea esperando una confirmación. En caso de que el mensaje enviado sea del tipo CON, mientras no reciba la confirmación realiza 4 retransmisiones separadas por 5, 8, 16 y 30 segundos entre cada una. Si no se recibe ninguna respuesta a las retransmisiones, o si el mensaje enviado originalmente es del tipo NON, el proceso que ejecuta esta función se bloquea. Para evitar esto, se programó otro proceso que funciona tipo *watchdog*. Este último verifica si el código está bloqueado, reiniciándolo nuevamente. El *watchdog* debe actuar antes del último mensaje de retransmisión, caso contrario se bloqueará el proceso por más que el *watchdog* reinicie nuevamente el proceso. Por esta razón, se lo programa para actuar a los 15 segundos de iniciada la transmisión.

Intercambio desde las placas a Thingsboard

Estos mensajes se envían desde clientes CoAP. Los mensajes son del tipo CON, método POST y formato JSON, enviados a una URI configurada en Thingsboard para recibir el mensaje de telemetría. Esto se desarrolló con clientes CoAP utilizando la función "$COAP_BLOCKING_REQUEST$", que funciona de igual manera que en el caso de los mensajes entre placas, ejecutándose en otro proceso y con otro *watchdog*.

En la figura 5.10 se presenta el intercambio de mensajes cuando una placa envía un mensaje a Thingsboard, y cómo se comporta el sistema caso de que no se reciba reconocimiento.

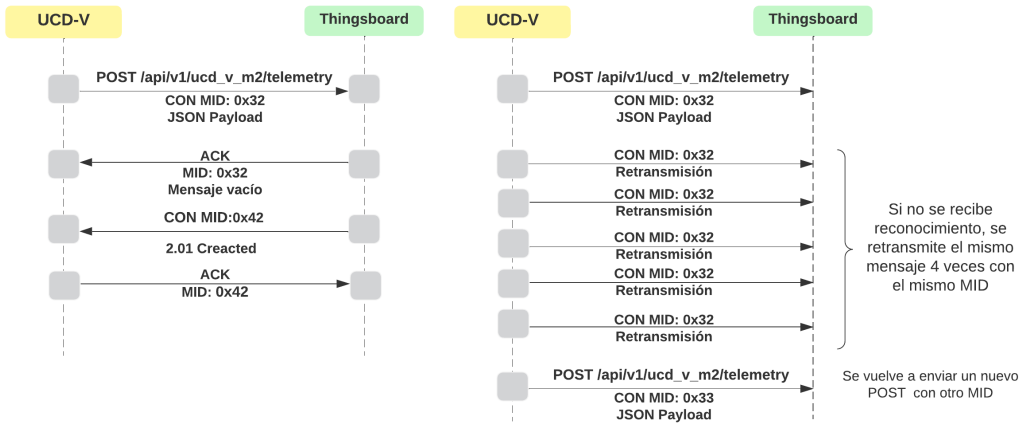


Figura 5.10: Envío de telemetría a Thingsboard

Contiki-NG está principalmente desarrollado para servidores CoAP, pero no tanto para clientes CoAP, por lo que no se logró encontrar un cliente que logre manejar una respuesta separada, dado que si bien es posible contestar mensajes con un *token*, un cliente no genera un *token*. De esta forma, ante un intercambio con respuesta separada, en primera instancia el cliente envia un mensaje CON, recibiendo una respuesta ACK con el mismo MID sin esperar otra respuesta. Luego, al recibir otro mensaje de tipo CON con otro MID y sin un método, el cliente ignora el mensaje, provocando que Thingsboard lo retransmitiera cuatro veces.

Esto se resolvió agregando el caso particular de Thingsboard a la programación de CoAP en Contiki-NG. Se programó que si el mensaje es del tipo CON y proviene de la dirección de Thingsboard, entonces se conteste con un mensaje ACK vacío.

Intercambio entre Thingbsoard y *scripts* Python

Como se comentó en el comienzo de esta sección, se cuenta con *scripts* desarrollados en Python destinados a subsanar problemas de integración en la comunicación de Thingsboard con el resto del sistema. Los *scripts* son ejecutados luego de 3 minutos de iniciado Thingsboard, para asegurar la suscripción al recurso (en este caso no se cuenta con retransmisión caso no se realice la suscripción).

Por un lado, se tiene que Thingsboard admite que un único cliente se suscriba a cada recurso, por lo que se define que los *scripts* cumplan este rol de clientes CoAP y se suscriban a los recursos de Thingsboard (comandos y parámetros de configuración). Esta suscripción tiene duración ilimitada.

Por otra parte, los *scripts* también tienen el propósito de recibir los comandos de Thingsboard y redirigirlos hacía las placas correspondientes, además de enviar un mensaje de actualización de valores a Thingsboard cuando éste lo solicite. Esto último ocurre siempre que un cliente web actualiza la página de visualización de Thingsboard. Para actualizar los valores, al recibirlos son guardados en una base de datos manejada desde Python, consultada cada vez que Thingsboard solicita la

actualización.

En la figura 5.11, se ilustra los mensajes intercambiados entre los *scripts* en Python y Thingsboard. La figura de la izquierda muestra los mensajes cuando un *script* en Python se suscribe a un recurso de Thingsboard, mientras que la figura a la derecha ilustra el intercambio de mensajes cuando Thingsboard solicita una actualización de valores.

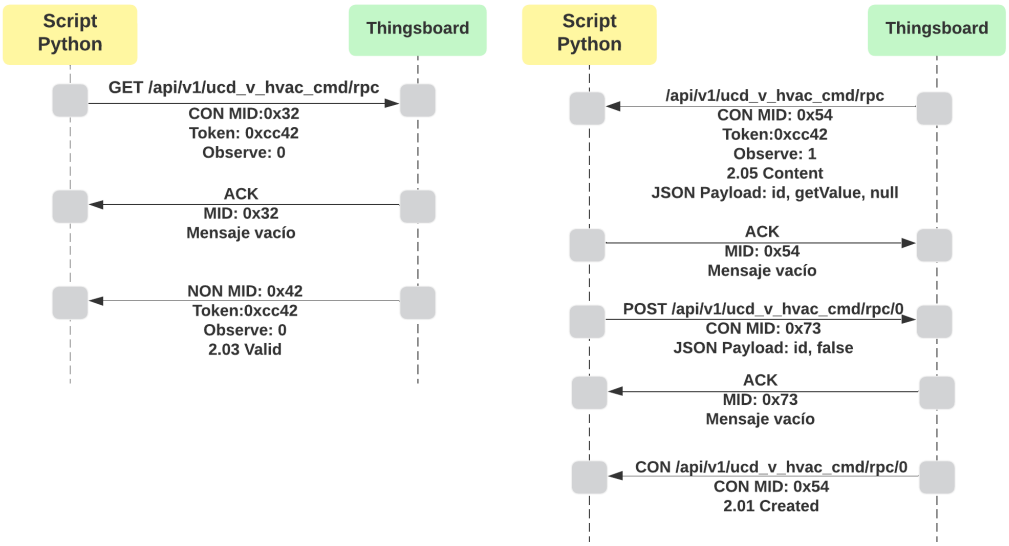


Figura 5.11: Izquierda: suscribirse a un recurso. Derecha: actualización de valores

Si bien las funciones de los *scripts* podrían haber sido implementadas desde Contiki-NG, la solución desarrollada permite robustecer las suscripciones, dado que estos *scripts* se encuentran en la misma máquina que Thingsboard, por lo que se evita el riesgo de pérdida de mensajes y no es necesario una lógica de suscripciones periódicas. A su vez, siendo que los mensajes son internos, se fortalece la independencia entre el nivel de operación y el nivel de controladores, además de contribuir a reducir el tráfico en la red interna.

Intercambio desde Thingsboard hacia placas

Los mensajes de comando o configuración de parámetros, se envían desde Thingsboard hacia las UCD correspondientes. Estos mensajes pasan por los *scripts* en Python, donde se guarda el valor recibido y se transmite el mensaje a la dirección correspondiente. El mensaje desde Thingsboard a los *scripts* en Python se envía como un mensaje CON, en formato JSON y con el mismo *token* utilizado cuando se realizó la suscripción al recurso. Luego el *script* correspondiente envía el mensaje como un mensaje CON, método POST y en formato texto plano a la dirección adecuada. Si en algún punto no se recibe el mensaje de reconocimiento debido, se retransmite el mismo mensaje 4 veces. En la figura 5.12 se muestra un

ejemplo de un mensaje de comando enviado desde Thingsboard hacia un controlador.

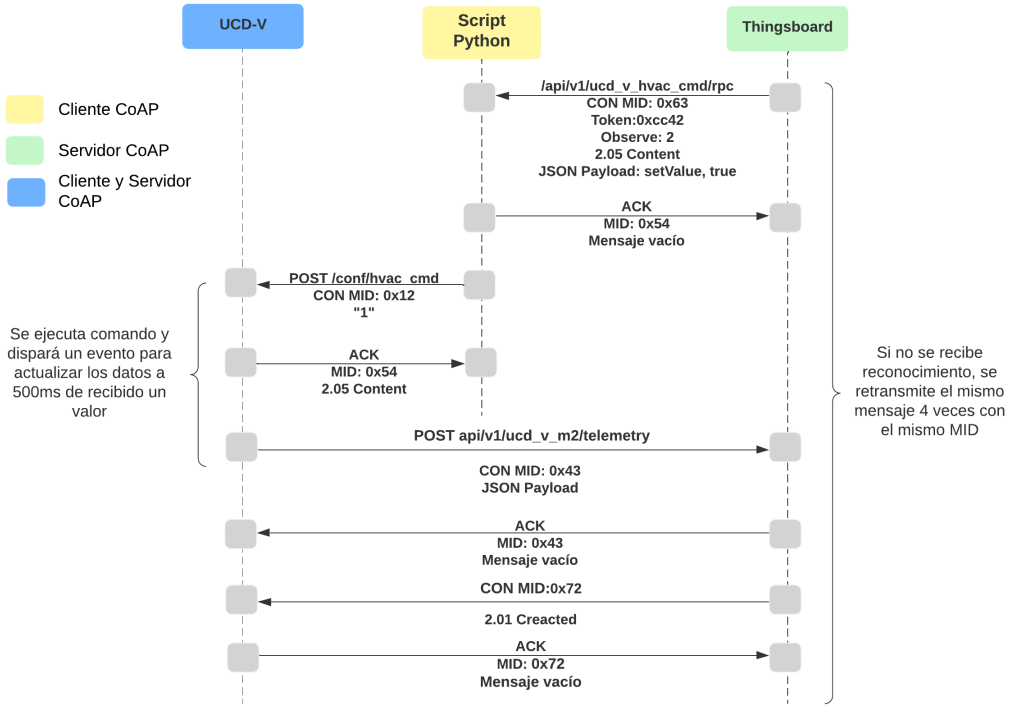


Figura 5.12: Mensaje enviado desde Thingsboard hacia UCD

Cuando el controlador recibe el mensaje dispara un evento para enviar un mensaje de actualización de valores a los 500 *ms* de recibido el comando. Esto se hace para recibir la confirmación del estado, y no tener que esperar hasta el próximo mensaje de telemetría, de este modo el usuario lo ve al instante. Se utiliza este tiempo, ya que en el caso de prender algún equipo, existen algunos *ms* entre que la placa ejecuta el comando y se recibe la señal de realimentación (indicando, por ejemplo, que el último relé encargado de prender este equipo se energizó).

Intercambio desde *script* Python hacia UCD-R

Solamente existe un mensaje que surge desde uno de los *scripts* de Python hacia las UCD-R1 y UCD-R2, con el objetivo de actualizar un valor que no proviene de Thingsboard. Cada 30 minutos, se envía un mensaje de actualización de hora hacia estas UCD, debido a que requieren este dato para realizar el control. Este mensaje es del tipo CON, método POST y se retransmite 4 veces, caso de que no se reciba el reconocimiento.

Se envía de forma periódica para evitar fallas de sincronismo, caso se de que alguna placa se apague y se vuelva a prender, por algún corte eléctrico u otro

motivo. Además, la UC cuenta con conexión directa a internet, por lo que es capaz de mantener la hora actualizada.

5.3.2. Mensajes entre UCD y NM, y desde UC hacia UCD-R

Como se comentó anteriormente, se envían mensajes de telemetría entre las UCD y NM, y desde la UC (script en Python) hacia las UCD-R, para que se pueda ejecutar el control. En esta sección se explica cuáles son los mensajes enviados y el período.

Por un lado, los datos de las variables sensadas son enviadas por un cliente CoAP a un servidor CoAP. Este último almacena dicho valor en un recurso y hace uso del mismo para el control. Para que este control pueda llevarse a cabo, los mensajes se envían de forma periódica.

En la tabla 5.1 se presentan los mensajes y el período de envío. Los mensajes de los NM hacía la UCD-R1, se envían cada 30 minutos, al igual que los mensajes entre UC y las UCD-R. El mensaje entre la UCD-V y la UCD-F se envía cada 40s.

Origen	Destino	URI	Variable	Período
NM1	UCD-R1	/post/m_nm1	Humedad relativa del suelo	30 min
NM2		/post/m_nm2		
UCD-V	UCD-F	/post/ext_temperature	Temperatura exterior	40 s
UC	UCD-R1	/post/current_hour	Envia hora	30 min
	UCD-R2			

Tabla 5.1: Mensajes entre UCD y NM, y desde UC hacia UCD-R.

5.3.3. Mensajes de telemetría hacia UC

Para llevar a cabo el monitoreo, el programa de cada UCD cuenta con dos procesos paralelos independientes. Si bien cada proceso puede tener su propio período de trabajo, actualmente ambos están configurados para ejecutarse cada 40 segundos

Estos procesos se encargan en primer lugar de tomar datos utilizando los drivers de cada sensor. Luego, agrupan los datos generando un mensaje en formato JSON que será enviado a la UC a la plataforma Thingsboard. En el caso de los NM, se cuenta con un único proceso que realiza las mismas tareas.

En las tablas 5.2 y 5.3 se presentan los mensajes de telemetría, los cuales incluyen lectura de sensores, actualización de valores configurados, estado de dispositivos y comandos. HVAC hace referencia al aire acondicionado de la sala correspondiente, mientras que MIC hace referencia al deshumidificador de la sala de secado.

Dado que los sensores DHT-11 y DS18B20 se comunican mediante un protocolo digital, se toman precauciones previendo casos de lecturas erróneas o fallos en el protocolo de comunicación. En el caso del DHT-11, si la lectura es errónea se envía el valor 254, mientras que si el valor excede cierto rango se lo descarta para cualquier lazo de control. Para el DS18B20, se define que la indicación de falla se envíe a la UC luego de 5 datos erróneos consecutivos, evitando dar falsos errores por lecturas fallidas individuales.

NM	Identificador	Variables Incluidas	Período
NM1	nm1	Temperatura y humedad relativa suelo. Alimentación y temp. SoC. Carga batería.	60s
NM2	nm2		

Tabla 5.2: Mensajes de telemetría hacia UC desde NM

5.3.4. Mensajes de comando y configuración de parámetros desde UC

Los mensajes de comando y configuración de parámetros provienen de la UC, más precisamente de la plataforma Thingsboard, enviados por los *scripts* de Python en el instante en que cambian de valor. Se presentan en detalle en la tabla 5.4.

Como se explicara en la sección 5.3.1, al recibir un mensaje de la UC con un comando o configuración de un parámetro se dispara un proceso que se encarga de enviar un mensaje luego de 500 ms, actualizando los valores de la UC. Los mensajes involucrados se presentan en la tabla 5.3, indicados con la palabra *Trigger*.

A su vez, si se trata de la configuración de un parámetro, también se ejecuta otro proceso que se encarga de actualizar la memoria ROM, donde se almacenan los parámetros configurados. Esta funcionalidad vuelve al sistema más robusto. De este modo, caso se apague un controlador por un corte eléctrico u otro motivo, mantendrá almacenado los parámetros de configuración de control y al prenderse podrá ejecutar el control sin problemas.

Desafíos de implementación

Los módulos CC2650MODA no cuentan con una memoria ROM externa destinada a almacenar datos como es el caso del *Launchpad* de CC2650 o CC1350 de Texas Instrument. Contiki-NG cuenta con un gran soporte para los *launchpad*, inclusive existen módulos desarrollados para manejar la memoria externa.

En nuestro caso, para darle robustez al sistema frente apagados no esperados, se utiliza almacenamiento persistente en los nodos. De esta forma, el nodo no debe esperar a recibir un comando de la UC al reiniciarse, sino que cuenta con la última configuración en su memoria. Se cargan entonces los parámetros en la memoria ROM del SoC, para lo que se utilizaron funciones que se encontraron en Contiki-NG, pero que provienen de la programación desarrollada por Texas Instrument.

UCD	Identificador	Variables Incluidas	Período
UCD-V	ucd_v_m1	Temperatura y humedad exterior. Temperatura y humedad sala. Temperatura SoC.	40s
	ucd_v_m2	Alimentación SoC. Estado luces. Estado automatismo hvac y hvac. Parámetros configurados para control hvac.	40s. *Trigger*.
UCD-F	ucd_f_m1	Temperatura y humedad sala, Temperatura SoC.	40s
	ucd_f_m2	Alimentación SoC. Estado luces. Lectura CO_2. Estado automatismo hvac y hvac. Parámetros configurados para control hvac	40s. *Trigger*.
UCD-S	ucd_s_m1	Temperatura y humedad sala. Temperatura SoC.	40s
	ucd_s_m2	Alimentación SoC. Estado automatismo mic y mic. Parámetros configurados para control mic.	40s. *Trigger*.
UCD-R1	ucd_r_m1	Nivel pH. Estado bomba dosificadora. Estado automatismo. Parámetros configuradoras para control pH	40s *Trigger*.
	ucd_r_m2	Alimentación y temperatura SoC. Volumen regado. Estado automatismo riego y bomba. Parámetros configurados para riego.	40s. *Trigger*.
UCD-R2	ucd_r_m3	Alimentación y temperatura SoC. Nivel alto/bajo tanque. Estado automatismo carga tanque.	40s. *Trigger*.

Tabla 5.3: Mensajes de telemetría hacia UC desde UCD

La memoria ROM solamente se puede borrar de a páginas (4kB), por lo tanto, se utiliza la última página de ROM del SoC para almacenar los parámetros recibidos.

Destino	URI	Mensaje
UCD-V	/conf/hvac_cmd	Comando On/Off HVAC
	/conf/aut_hvac_r_s	Comando On/Off automatismo
	/conf/hvac_set_sp	Setpoint temperatura nocturna
	/conf/hvac_set_hist	Histérisis control temperatura
UCD-F	/conf/hvac_cmd	Comando On/Off HVAC
	/conf/aut_hvac_r_s	Comando On/Off automatismo
	/conf/hvac_set_sp	Setpoint temperatura nocturna
	/conf/hvac_set_hist	Histérisis control temperatura
UCD-S	/conf/mic_cmd	Comando On/Off MIC
	/conf/aut_mic_r_s	Comando On/Off automatismo
	/conf/mic_set_sp	Setpoint temperatura nocturna
	/conf/mic_set_hist	Histérisis control temperatura
UCD-R1	/conf/ph_set_sp	Setpoint pH
	/conf/ph_set_hist	Histéresis pH
	/conf/aut_ph_r_s	Comando On/Off automatismo pH
	/conf/aut_r_r_s	Comando On/Off automatismo riego
	/conf/h_nm1_set_sp	Setpoint humedad maceta 1
	/conf/h_nm1_set_hist	Histéresis humedad maceta 1
	/conf/h_nm2_set_sp	Setpoint humedad maceta 2
	/conf/h_nm2_set_hist	Histéresis humedad maceta 2
UCD-R2	/conf/aut_t_r_s	Comando On/Off aut. carga tanque

Tabla 5.4: Mensajes de comando y configuración de parametros desde UC.

Cuando se borra una página cada byte pasa a valer 0xFF. Es necesario borrar la memoria y volver a escribir siempre que se quiera escribir en una dirección que ya contenga un valor escrito, lo cual se realiza siempre que se recibe un valor distinto al almacenado. De todas formas, no se espera que se cambien los parámetros de forma frecuente, por lo que la limitada vida útil de la memoria ROM (de 100 mil ciclos de borrado) no es una restricción que requiera una solución.

Para implementar esto se desarrollaron 2 funciones basadas en ejemplos extraídos de internet [66] [67]. Por un lado, *UpdateSavedValues* utiliza funciones para limpiar y grabar la memoria ROM, mientras que *FlashReadNonCache* utiliza funciones del sistema VIM (Versatile Instruction Memory) para leer la memoria flash. En este último caso, es necesario desactivar la memoria caché del SoC, para no leer el primer valor grabado.

5.3.5. Adquisición de datos

En esta sección se describe la adquisición de los datos de telemetría presentado en las tablas 5.1 y 5.3 por parte de las UCD y NM. Para realizar la adquisición se

requiere de los siguientes sensores:

- DHT-11: Temperatura y humedad ambiente.
- SEN0193: Humedad relativa del suelo.
- DS18B20: Temperatura de suelo.
- SEN0169: Sensor de pH
- SEN0219: Concentración de CO_2.
- YF-S201: Caudalímetro.
- LDR: Estado de lámpara.

No fue necesario desarrollar un *driver* para el sensor DHT-11, ya que viene incluido en Contiki-NG. Los sensores LDR, que monitorean el estado de las lámparas no requieren del desarrollo de un *driver*, ya que simplemente se lee una señal digital utilizando las funciones de Contiki-NG.

Sí fue necesario crear un *driver* para cada sensor analógico (SEN0193, SEN0169 y SEN0219), estos drivers se basan en uno desarrollado por Contiki-NG para lecturas analógicas. Se desarrolló este *driver* para procesar el dato analógico leído y convertirlo de μV a un valor representativo de la variable que sensa. En el caso del sensor de humedad, se adquieren 10 datos consecutivos y se genera un valor puntual a partir del promedio del conjunto.

En el caso del sensor DS18B20 (temperatura de suelo), utiliza el protocolo 1W. El *driver* fue desarrollado a partir de un ejemplo encontrado en internet, incorporando ciertos ajustes de acuerdo a la arquitectura en la cual Contiki-NG define los sensores.

Como parte del procesamiento de datos, el NM antes de enviar la lectura adquirida del sensor verifica si se tiene un caso de error (de 5 mensajes con valor erróneo). En caso de ser afirmativo envía un valor que dispara una alarma de falla en el sensor en la plataforma Thingsboard.

La UCD-R1 tiene dos cuadalímetros de los que adquiere datos, para los que se desarrollaron dos *drivers* idénticos, pero asociados a cada caudalímetro. Esto fue necesario ya que cada *driver* utiliza un proceso propio. Este proceso se encarga de contar la cantidad de pulsos adquiridos en 1 segundo y convertir este dato en mL/min y mL acumulados. Cuando el sensor está activo, cada pulso creciente que se recibe en la entrada digital asociada genera una interrupción. Esta interrupción incrementa una cuenta acumulativa, deshabilita las interrupciones por 300 μs y luego de este tiempo las vuelve a activar. Esta desactivación se realiza para evitar lecturas erróneas, teniendo en cuenta que el caudalímetro no está diseñado para enviar pulsos con menor período.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Control

En este capítulo se describen las funciones de control implementadas en el sistema. Se busca con el mismo automatizar parcialmente algunos procesos de la producción.

Se diseñan entonces cuatro lazos de control destinados a mejorar el entorno de las plantas, donde tres de ellos inciden directamente sobre las mismas mientras que el restante es aplicado al proceso productivo, más precisamente en la planta de agua.

El principal proceso intervenido es el de riego, que involucra dos de los lazos de control diseñados.

El proceso de riego comienza cada día a determinada hora en la madrugada, donde se activa el lazo de control de pH del agua. Este lazo consiste en medir el nivel de pH y dosificar ácido para ajustarlo, con recirculaciones periódicas para la homogeneización del pH en todo el volumen.

Una vez corregido el nivel de pH, se procede con el control de humedad de suelo. En primer lugar se determinan los volúmenes de agua a regar en cada grupo de macetas, en base a la medición de humedad relativa de suelo (parámetro que compara volumen de agua con volumen de tierra) y un umbral óptimo establecido. De esta forma, manteniendo la recirculación del agua del tanque, se manejan las aperturas y clausuras de las electroválvulas de cada grupo de macetas, variando el tiempo de apertura según el volumen de agua requerido.

Por último, a una hora establecida cercana al mediodía para permitir al cliente el riego manual en la mañana, se prepara la planta de agua para un nuevo proceso de riego, llenando el tanque mediante una electroválvula en la entrada de agua. El llenado es implementado utilizando un sensor de nivel de tipo boya, que cuenta con una salida de contactos secos, activados por un *switch*. Utilizando dicha salida y un relé DPDT, podemos tanto dar aviso a la UCD-R del estado del tanque (nivel máximo alcanzado o no), como desactivar la electroválvula mediante un enclavamiento (*interlock*). Además, previendo eventuales fallas en el circuito, el controlador implementa un temporizador de llenado que detiene el proceso luego de una hora, minimizando las posibles pérdidas por desbordes.

El proceso de riego es completado con recirculaciones periódicas dentro del

mismo tanque a lo largo del día, cada una hora y durante 5 minutos, a fin de acelerar la decloración del agua. En la figura 6.1 se muestra un diagrama de bloques representando el proceso de riego completo.

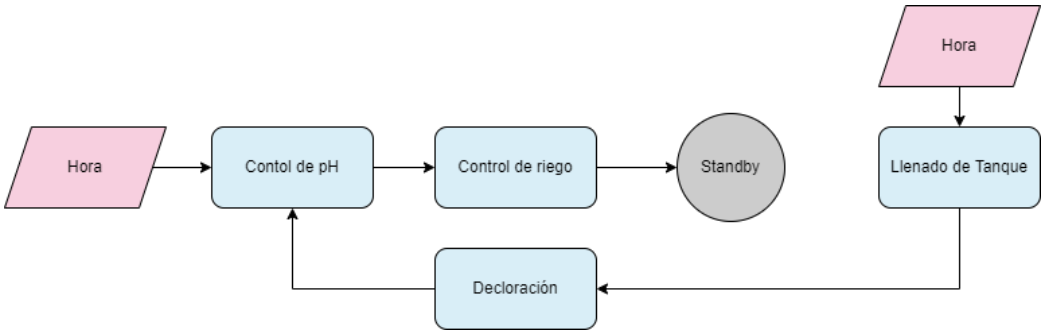


Figura 6.1: Diagrama de flujo del proceso de riego.

6.1. Control de Humedad de Suelo

La humedad del suelo es una de las variables más importantes para el crecimiento de la planta. Las macetas deben ser regadas de forma recurrente, evitando que se seque el suelo y se produzca la marchitez de la planta.

Se monitorea el nivel de humedad del suelo para generar un proceso de riego automático. Esta variable de realimentación permite relacionar el estado de la planta con la cantidad de agua a verter. Además, se busca mantenerlo en un rango óptimo para evitar tanto excesos que lleven a desbordes, como insuficiencias que perjudiquen a la planta. A su vez, el monitoreo de humedad permite al productor relacionar su apreciación visual con valores objetivos.

6.1.1. Parámetros Hídricos

De acuerdo a [81], los suelos pueden contener diferentes cantidades de agua en función de su textura y estructura. Se puede describir un suelo en función de sus parámetros hídricos, mediante los cuales se determina un umbral de cantidad de agua contenida en el suelo que resulta en una óptima absorción de la misma por la planta.

Se tiene en primer lugar la Capacidad de Campo (CC), que corresponde a la cantidad de agua que un suelo bien drenado retiene en contra de las fuerzas gravitatorias. Es decir, la cantidad de agua remanente en el suelo cuando el drenaje ha disminuido significativamente [50]. Este punto es el límite máximo de agua utilizable por las plantas, por lo que representa el máximo nivel de confort hídrico [35], y define nuestro límite superior del umbral óptimo.

Luego, se tiene también el nivel denominado Umbral de Riego (UR). A partir de este punto el agua disponible es difícil de absorber por la planta, por lo que lo tomamos como el límite inferior del umbral óptimo de humedad.

Finalmente, se tiene el Punto de Marchitez Permanente (PMP), que representa la cantidad de agua contenida en el suelo a la que las plantas ya no pueden acceder, provocando su marchitez [50]. Se busca evitar que la humedad del suelo llegue a este nivel.

6.1.2. Rango Óptimo

La optimización del riego consiste entonces en que la cantidad de agua se mantenga en un umbral donde, por un lado, no se desperdicie por el drenaje gravitatorio (removiendo nutrientes del suelo), y por otro, le sea fácilmente accesible a la planta. Este umbral suele definirse entre los niveles de CC y UR.

Por otra parte, se busca también trasladar estos niveles de humedad a magnitudes de volumen de agua. Teniendo en cuenta que los sensores fueron calibrados para identificar el porcentaje de humedad relativa volumétrica, podemos entonces convertirla directamente a un volumen de agua, en función del volumen de tierra existente en la maceta. Esta conversión también aplica para los niveles de CC y UR, cuya diferencia en términos de volumen de agua se denomina Agua Fácilmente Disponible (AFD). El objetivo entonces de la automatización es que el volumen de agua contenido en la maceta se encuentre siempre en el rango de AFD.

6.1.3. Lazo de Control

Debido a que la humedad de suelo requiere cierto tiempo luego del riego para llegar a homogeneizarse en toda la maceta, el lazo de control de riego se implementa como un lazo cerrado en tiempos discretos, aplicando un control de tipo Proporcional-Integral (PI) correspondiente a la ecuación 6.1.

$$V = k_P \times e_H + k_I \times \sum_{i=1}^{j-1} d_{Hi} \tag{6.1}$$

Donde :
$V \rightarrow$ *volumen de agua a regar en L*
$k_P \rightarrow$ *factor de proporcionalidad del lazo*
$e_H \rightarrow$ *error respecto del set-point medido previo al riego*
$k_I \rightarrow$ *factor multiplicativo del integrador*
$d_H \rightarrow$ *error respecto del set-point medido luego de estabilizado el riego*
$j \rightarrow$ *índice de día* ($j-1:$ *ayer*, $j:$ *hoy*)

La variable de entrada es la lectura de humedad relativa de suelo de la maceta, tomando último dato recibido. Se tiene a su vez un *set-point* (correspondiente al nivel de CC) que se desea que alcance esta variable, para lo que se debe calcular el volumen de agua a regar. En base a este *set-point* se construyen los términos

e_H y d_H, que corresponden al error de la humedad (en términos de porcentaje de humedad relativa) respecto de este valor objetivo. Estos términos se diferencian en el momento de evaluar la variable de entrada, siendo que e_H corresponde el valor de humedad al momento de realizar el cálculo (previo al riego), mientras que el d_H es la diferencia del *set-point* con el mayor valor de humedad alcanzado luego del riego.

Por su parte, el factor de proporcionalidad k_P transforma el error evaluado en porcentaje de humedad relativa, a términos de volumen de agua. Se define entonces que $k_P = \frac{V_m}{100}$ siendo V_m el volumen de tierra en la maceta. A su vez, el factor de integración lo definimos como una fracción del anterior, cumpliendo que $k_I = 0,1\ k_P$. Estos factores serán ajustados en base a resultados empíricos luego de instalada la planta de agua.

El término sumador d_H de la ley corresponde a un ajuste periódico, que permite ajustar un eventual *offset* por no idealidades y desvíos de parámetros del modelo.

Para construir este valor, se debe tener en cuenta que una vez completado el riego se debe esperar un tiempo de estabilización para encontrar el valor de humedad que alcanzó la maceta. Puede definirse entonces el valor de salida real del ciclo como el máximo valor de humedad luego de realizado el riego, y previo al siguiente. Partiendo de esta restricción, antes de cada riego, se identifica el error d_H del riego anterior, tomando el valor máximo de los obtenidos desde éste. Dicho error responde entonces a la ecuación 6.2.

$$d_H = H_{SP} - max_{24hs}(H_L) \tag{6.2}$$

Donde :
$H_{SP} \rightarrow$ *set-point de humedad relativa volumétrica de suelo*
$max_{24hs}(H_L) \rightarrow$ *máxima lectura de humedad relativa volumétrica de suelo desde el riego anterior*

Por otra parte, teniendo en cuenta que el riego se hará a caudal fijo, se tiene una relación proporcional entre el volumen de agua y el tiempo que se mantienen las válvulas abiertas. De esta forma, se completa la ley que gobierna el lazo de control. En la figura 6.2 se representa gráficamente esta ley.

Por otro lado, se implementa una medida de contingencia para evitar que errores de cálculos o medición resulten en riegos ilógicos. Para ello se establecen límites mínimo y máximo de volumen a regar, de forma tal que todos los días las plantas sean regadas y que no se desborden. En base a la información brindada por el cliente, se define el riego mínimo en 0,3 L y el máximo en 5 L.

Finalmente, se obtiene una variable temporal $T = \frac{V}{C}$ en función del caudal de riego. El diagrama de bloques de la figura 6.3 representa el lazo de control descripto.

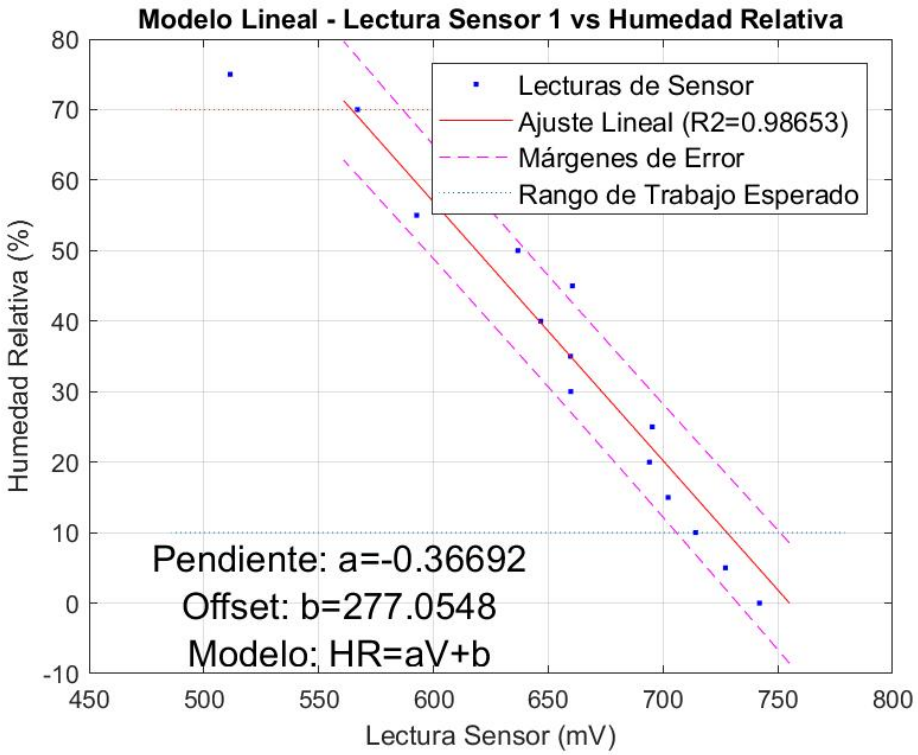


Figura 6.2: Modelo lineal del lazo de control.

6.2. Control de Nivel de pH

Este lazo de control realiza de forma periódica recirculaciones del agua del tanque y mediciones de pH y dosificaciones de producto acidificante (ácido fosfórico). Estas dosificaciones varían en volumen, buscando un acercamiento asintótico por exceso. Se busca una ley de control proporcional debido a que se implementa únicamente la acidificación del agua (reducción del pH), no así su alcalinización (incremento del pH).

6.2.1. Valores Óptimos

Según información oficial de OSE [43], se estima que el pH promedio del agua de la red es 6,9, subiendo a 7,5 luego de la decloración. De acuerdo al cliente, este valor es elevado para el riego de las plantas, afectando su salud, por lo que se busca que el pH se reduzca hasta un valor de 5,8 para SV y 6,2 para SF.

Se implementa entonces un rango de histéresis, definiendo un *set-point* y un umbral, resumidos en la tabla 6.1. Se establece entonces el umbral de aceptación

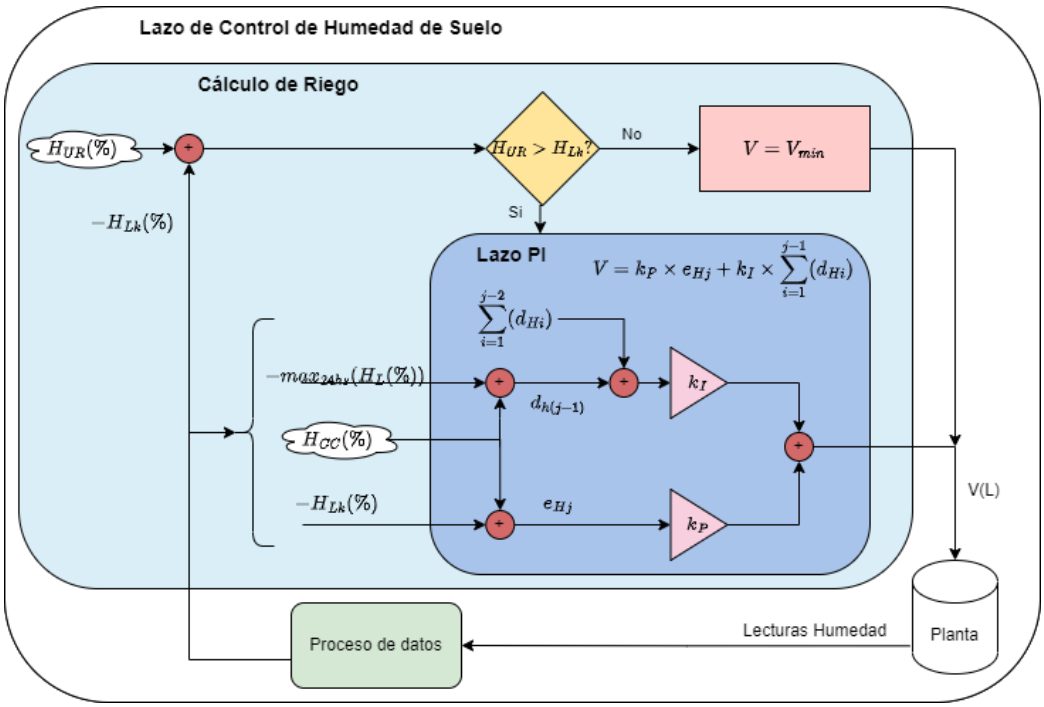


Figura 6.3: Diagrama de lazo de humedad de suelo.

en 5 % por encima del valor de pH buscado, donde se entiende que una vez que el agua se reduce a menos de ese valor, está lo suficientemente cerca del nivel deseado, por lo que se asume como concluido el ajuste.

Nivel	Valor (pH)
Agua inicial	6,9
Agua declorada	7,5
Umbral de Aceptación	6,5
Set-point para SF	6,2

Tabla 6.1: Rangos y valores del ajuste de pH.

6.2.2. Proceso de Ajuste

El ajuste comienza con una primera medición del pH del tanque. Este valor define si se aplica el lazo de control (si el valor obtenido está por encima del umbral de aceptación) o si el agua está en condiciones de riego.

El proceso, representado en la figura 6.4, implica una recirculación del agua del tanque, para acelerar la homogeneización del pH en todo el volumen. Periódicamente se realizan nuevas mediciones de pH, realimentando el lazo de control aplicado. Este lazo tiene como salida un valor de volumen del líquido de ajuste,

que será administrado por la bomba peristáltica. Dado que la bomba funciona a caudal constante, el volumen de líquido vertido es proporcional al tiempo de encendido de la bomba, por lo que ésta será la variable final utilizada por el controlador.

Las mediciones y dosificaciones se repiten hasta que se cruza el umbral de aceptación, donde se asume el pH ajustado y se deja de aplicar el lazo, continuando el proceso de riego.

Se tiene entonces un proceso de tipo *batch*, donde la integración del volumen de ácido se da dentro del tanque de agua.

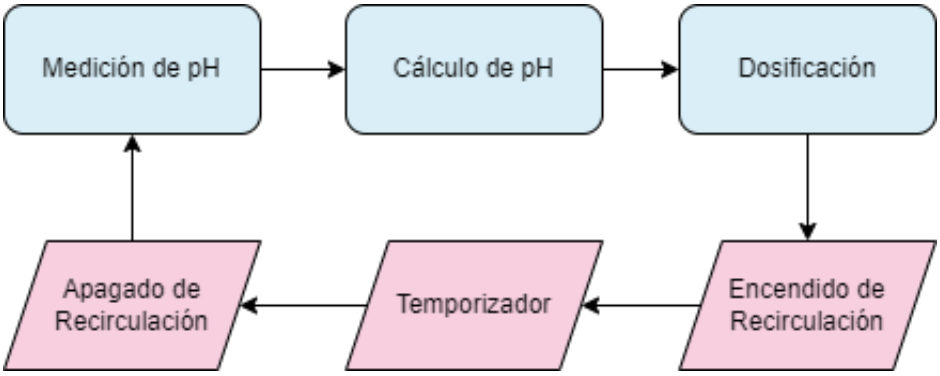


Figura 6.4: Diagrama de flujo de ajuste de pH.

6.2.3. Lazo de Control

Dada la naturaleza del proceso y de las actuaciones posibles, se implementa un lazo de control cerrado en tiempos discretos de tipo proporcional. En la figura 6.5 se presenta el diagrama de bloques del lazo implementado.

Una vez iniciado el proceso de ajuste, se tienen datos de pH que deben transformarse en la dosificación correcta de ácido. La variable de entrada al lazo entonces es la medida de pH del agua.

Para encontrar la ley que gobierne la transformación buscada se toma en cuenta el ensayo que se describe en la sección A del anexo, realizado por el equipo de proyecto. El ensayo permite obtener un modelo que responde a un comportamiento de tercer grado, sin embargo, nuestro rango de trabajo está reducido al intervalo entre 7,5 y 5,5 pH aproximadamente. Se observa gráficamente que en ese rango el comportamiento es casi lineal, por lo que se genera el modelo de la figura 6.6.

De esta forma, con la pendiente obtenida se puede generar el factor de proporcionalidad k_P para la ley de control. Teniendo en cuenta que los decrementos de pH deben ser cortos para un acercamiento lento hacia el *set-point*, se incluye un factor de estabilidad $f_E = 0,20$. Además, se utiliza como constante el volumen de agua del tanque (indicado en 100 L). De esta forma la ley de control queda definida según la ecuación 6.3.

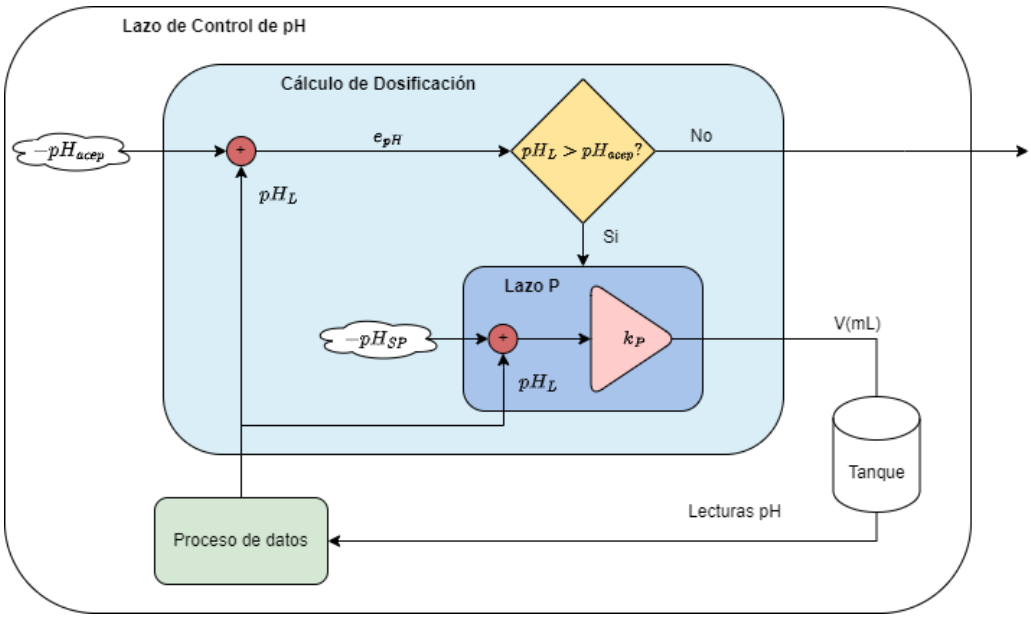


Figura 6.5: Diagrama de bloques del lazo de control de pH.

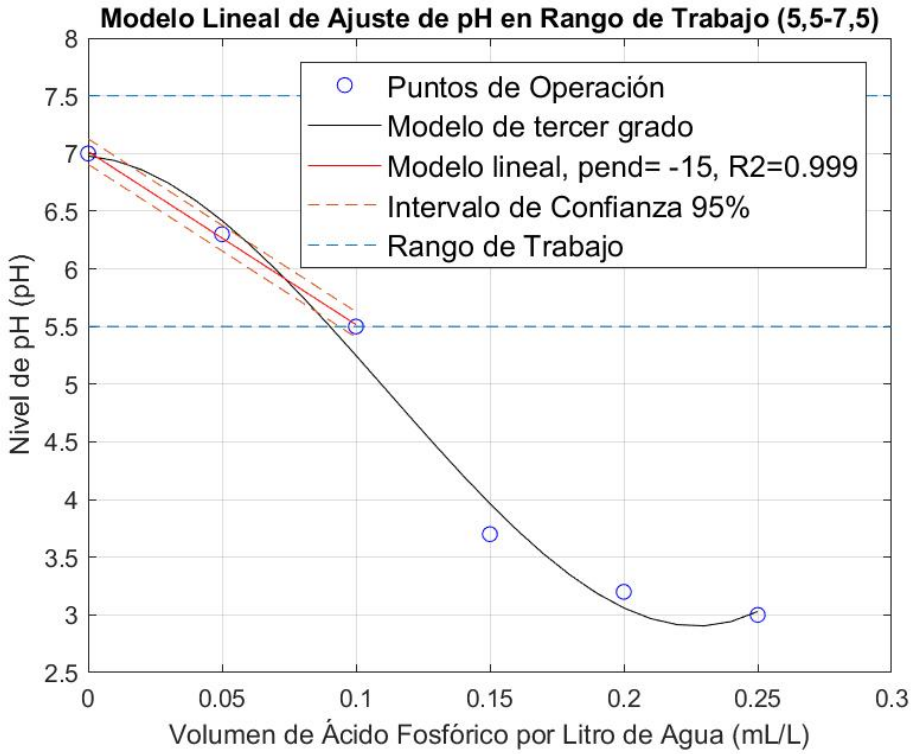


Figura 6.6: Modelo de variación de pH por dosificación de ácido fosfórico.

$$V_{ácido} = f_E \times p \times \Delta pH \times V_{tanque} = k_P \times \Delta pH \tag{6.3}$$

$Donde:$
$V_{ácido} \rightarrow$ *volumen de ácido fosfórico necesario para alcanzar el set-point*
$f_e \rightarrow$ *factor de estabilidad* $(0, 20)$
$p \rightarrow$ *pendiente del modelo lineal (pH en función de volumen de ácido por volumen de agua en tanque)*
$\Delta pH \rightarrow$ *diferencia entre nivel de pH actual y set-point*
$V_{tanque} \rightarrow$ *volumen de agua en tanque*
$k_P \rightarrow$ *factor de proporcionalidad del lazo*

Finalmente, siendo que la bomba peristáltica permite dosificar a un caudal de C=3 mL/s, se tiene que la variable final para el sistema es:

$$T = \frac{V_{ácido}}{C} = \frac{k_P}{3} \times pH \tag{6.4}$$

6.3. Control de Humedad Ambiente

Si bien la humedad ambiente es una variable que se monitoreará en las tres salas que involucran plantas, tiene una importancia particular en la SS. La humedad ambiente afecta directamente al correcto secado de la cosecha, por lo que se decide implementar un control de lazo cerrado.

6.3.1. Rango Óptimo

Para que las ramas cosechadas se sequen de forma adecuada, la humedad ambiente debe reducirse a valores bajos. El umbral, fijado en 60 %, definido por el cliente, pudiendo cambiarlo mediante comandos incluidos en el SCADA.

6.3.2. Lazo de Control

En primer lugar, se toma en cuenta que el monitoreo de la humedad ambiente se realiza de forma constante, y que la percepción del sensor a cambios de la variable es bastante rápida. Además, el actuador utilizado (deshumidificador), incide de forma directa en la humedad ambiente, por lo que no se necesitan cálculos intermedios. Por otro lado, no fue necesario incluir límites en el uso del equipo, dado que la reducción de la humedad ambiente, incluso a niveles mínimos, no deteriora las ramas de la sala.

Se define entonces un lazo cerrado de control con histéresis mediante dos límites. El primer límite indica la activación del actuador cuando la lectura de humedad lo supera. El segundo cumple la función contraria, indicando al sistema el apagado

del deshumidificador cuando la lectura de humedad se encuentra por debajo. Estos límites están fijados en 5 % sobre el umbral y 5 % por debajo del mismo.

Finalmente, el lazo implementado es representado por la figura 6.7.

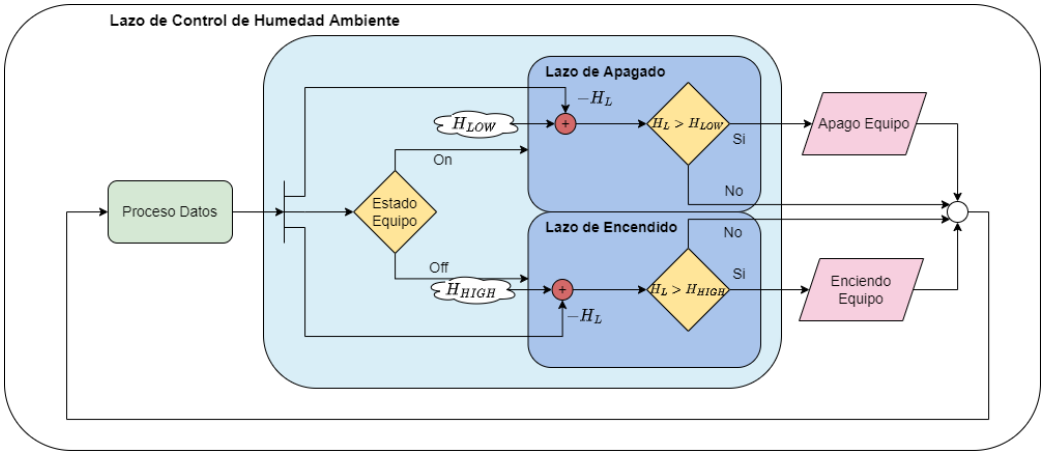


Figura 6.7: Diagrama de lazo de control de humedad ambiente.

6.4. Control de Temperatura Ambiente

Para aportar un mejor acercamiento del entorno artificial al comportamiento natural del ambiente, se decide implementar un control de la temperatura ambiente de las salas SV y SF. Este control busca principalmente diferenciar las temperaturas en los períodos de día (lámparas encendidas) y noche (lámparas apagadas).

6.4.1. *Set-points* de Temperatura

Las temperaturas objetivo varían en función del período lumínico y de la sala en la que se encuentran. En base a recomendaciones del cliente, se definen las temperaturas de la tabla 6.2.

Sala	Lámparas Encendidas (°C)	Lámparas Apagadas (°C)
SV	22	14
SF	19	12

Tabla 6.2: *Set-points* de temperaturas ambiente.

6.4.2. Lazo de Control

El control de esta variable es implementado mediante el manejo del encendido y apagado del equipo de aire acondicionado.

Para definir la operación del equipo, se parte de que las lámparas representan una considerable fuente de calor, cuya incidencia en la temperatura de la sala permite despreciar los efectos de otras fuentes. En cambio, al apagarse las luces esta fuente de calor desaparece, por lo que la principal influencia proviene del ambiente exterior. De esta forma, durante el ciclo de día los aires acondicionados deben enfriar la sala para contrarrestar el efecto de las lámparas, mientras que en el ciclo nocturno, la influencia del exterior puede generar tanto un aumento como un descenso de la temperatura.

Dado que no se cuenta con la capacidad de cambiar la configuración del aire acondicionado de forma directa, se debe definir un único valor de temperatura fijo con el que trabajará el equipo. Teniendo en cuenta el calor emitido por las lámparas, el horario diurno es el de mayor intercambio de energía, convirtiéndolo en el horario más crítico para el trabajo del aire acondicionado. Se espera que el mismo deba trabajar más para evitar que la temperatura aumente, estando la mayor parte del tiempo en estado activo. Por esta razón se decide que la temperatura configurada en el equipo sea el *set-point* de temperatura para el horario diurno. Además, se resuelve mantener el aire acondicionado alimentado, dejando el control de la temperatura a la lógica del propio equipo.

Una vez apagadas las lámparas, se retoma el control por parte del sistema. El funcionamiento del mismo dependerá de la temperatura exterior y su relación con los *set-points* definidos en la tabla 6.2. Se distinguen entonces tres casos posibles:

- Caso 1) Temperatura exterior > *Set-point* diurno > *Set-point* nocturno: En este contexto, el ambiente exterior tiende a aumentar la temperatura de la sala por encima de la temperatura configurada en el aire acondicionado, por lo que el equipo se mantiene encendido siempre que se cumpla esta condición.

- Caso 2) *Set-point* diurno > Temperatura exterior > *Set-point* nocturno: Dado que el equipo solo puede trabajar para llevar la temperatura al *set-point* diurno y que el ambiente exterior no contribuirá a disminuir la misma por debajo del *set-point* nocturno, se mantiene el aire acondicionado apagado.

- Caso 3) *Set-point* diurno > *Set-point* nocturno > Temperatura exterior: Para este caso se implementa el control On/Off completo. Se tiene por un lado que el ambiente exterior incidirá en la reducción de la temperatura ambiente, mientras que el aire acondicionado trabajará para aumentarla hasta el *set-point* correspondiente al día. Bajo estas condiciones entonces, se define un rango de histéresis de ± 2 °C alrededor del *set-point* nocturno, donde el equipo permanecerá apagado. Se encenderá nuevamente si la temperatura de la sala baja por fuera de dicho rango.

Si bien el último es el único caso en el que se logra mantener la temperatura ambiente en un entorno del *set-point* buscado, se entiende que los criterios tomados en los otros dos casos son la solución más efectiva en su contexto, dado que mantienen la temperatura lo más cercana posible al valor deseado.

Unificando las condiciones de los tres casos se resuelve que, si la temperatura exterior es superior al *set-point* diurno el aire acondicionado se mantenga encendido, mientras que si la temperatura exterior es menor a dicho valor, se encienda el equipo solo si la temperatura de la sala es menor a 2 °C por debajo del *set-point* del horario nocturno.

Se presenta en la figura 6.8 un diagrama de bloques representativo del lazo implementado.

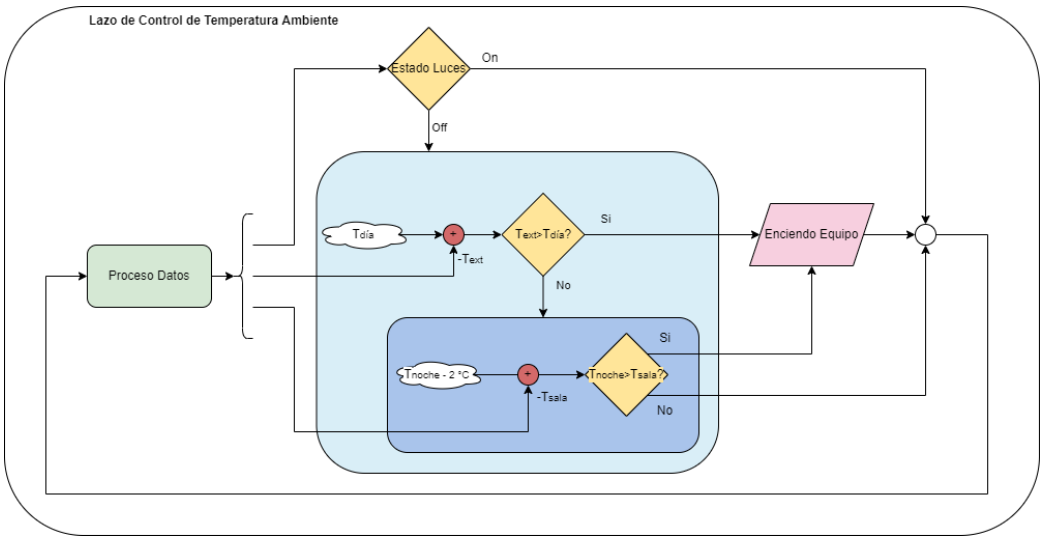


Figura 6.8: Diagrama del lazo de control de temperatura ambiente.

Capítulo 7

Sistema SCADA

7.1. Elección SCADA

Como *software* SCADA se definió elegir uno que cumpla con la característica de ser *web-based* y gratuito. De este modo se facilita el acceso remoto a la interfaz del usuario, ya que no se requiere de un *software* especial, sino que simplemente se puede acceder desde un navegador web, incluso desde un dispositivo móvil.

Esta plataforma debe estar diseñada para permitir la visualización de las variables monitoreadas y permitir el envío de comandos hacia los controladores. También es importante que cuente con la posibilidad de manejar alarmas y guardar un historial con las variables, permitiendo la trazabilidad de la producción.

Es importante que esta plataforma sea fácil de utilizar, tanto para el usuario final como para el equipo del proyecto que se encargará de realizar la configuración y/o programación necesaria del sistema.

En el marco del proyecto de fin de curso de Redes de sensores inalámbricos (apéndice D), se llegó a probar dos plataformas, Mango OS [36] y Thingsboard [72].

Mango OS no tiene implementada la comunicación mediante protocolo CoAP, por lo que, en el marco de dicho proyecto, se desarrolló un *script* en Python con el propósito de integrar la red inalámbrica 6LoWPAN con este SCADA. Este código permite la comunicación entre los servidores CoAP y Mango OS, cumple la función de ser un cliente CoAP, donde solicita las lecturas mediante mensajes GET, guarda los valores y crea un servidor Modbus TCP/IP, al cual Mango OS se comunica y obtiene los valores de lectura.

Thingsboard soporta CoAP con mensajes JSON, pero cumple el rol servidor CoAP, se describe en la sección 7.2. En esta etapa del proyecto solamente se contaba con el desarrollo de servidores CoAP, por lo que para lograr la integración de la red inalámbrica de controladores a la plataforma se desarrolló un cliente CoAP en Python, que envié mensajes GET a los nodos servidores y envíe mensajes POST

a ThingsBoard con el valor recibido.

En estas pruebas solamente se probó la característica de monitoreo, no la posibilidad de enviar comandos desde el la interfaz de usuario.

Como conclusión de estas pruebas, se llegó a que ambas plataformas presentan dificultad en la integración con la red de controladores y muy probablemente sea necesario desarrollar código en Python para subsanar este aspecto. De todos modos, se podría lograr integrar Thingsboard de forma directa si se desarrollase clientes CoAP en la red de controladores. Mango OS en cambio, requiere de más programación para crear las pantallas, dado que en esta plataforma se programa en JavaScript. Si bien se cuenta con algún widget desarrollado, el desarrollo de pantallas no es tan trivial como en el caso de Thingsboard.

Se definió seguir adelante con la plataforma Thingsboard, lo cual implicó enfocarse en el desarrollo de clientes CoAP en Contiki-NG. Esto se decidió debido a que esta plataforma cuenta con mayor soporte en internet, se cuenta con una documentación vasta y se encontró más experiencia de terceros utilizándola.

Por otro lado, esta plataforma también resultó ser más intuitiva y fácil de utilizar, logrando crear pantallas de forma más rápida y sin mayores inconvenientes

7.2. Thingsboard

ThingsBoard es una plataforma IoT de código abierto destinada al monitoreo, procesamiento, visualización, administración de dispositivos de datos y control remoto de dispositivos en tiempo real. Permite la conectividad a dispositivos a través de algunos de los protocolos más utilizados en la industria: MQTT, CoAP y HTTP y admite tanto implementaciones en la nube como locales. A continuación se detalla cada una de estas funcionalidades [70] [69].

- **Monitoreo:** Implementa servidores MQTT, CoAP y HTTP entre otros. A través de estos puertos cualquier tipo de dispositivo que tenga la capacidad, puede enviar y recibir mensajes.
- **Procesamiento:** Thingsboard permite crear *devices*, o dispositivos. Estas son entidades que poseen una ID y se utilizan para representar dispositivos reales. De esta forma si un nodo A de la red envía un mensaje a su entidad asociada A, la plataforma ya sabe que todo mensaje enviado a esa entidad proviene del nodo A sin necesidad de ningún procesamiento extra. A estos dispositivos se les puede asignar un perfil donde se puede configurar alarmas y condiciones de disparo de las mismas. También posee un motor de reglas que permite crear *rule chains*, o cadena de reglas. Este es un sistema altamente personalizable y configurable para el procesamiento de eventos complejos. Con el motor de reglas, todo mensaje entrante originado por dispositivos reales y activos debe atravesar una cadena de reglas la cual puede filtrar, enriquecer y transformar los mensajes. Estas cadenas de regla son

configurables mediante una interfaz gráfica que permite arrastrar y conectar *rule nodes* o bloques de procesamiento que brinda por defecto la plataforma.

- **Visualización**: Se brinda la capacidad de crear y administrar *dashboards*. Esto no es más que un panel o pantalla con el fin de presentar información. Cada panel puede contener varios *widgets* (módulos de interfaz gráfica) que proporcionan funciones para el usuario final. Dichas funciones abarcan visualización de datos, control remoto de dispositivos, gestión de alarmas y visualización de contenido HTML personalizado entre otras. Una vez que se crea el panel, al igual que con los dispositivos, se pueden asignar a uno o varios clientes.

- **Históricos:** Esta plataforma utiliza una base de datos para almacenar entidades (dispositivos, clientes, paneles, etc.) y datos de telemetría (lecturas de sensores de series temporales, estadísticas, eventos). Todo el manejo de la base de datos es realizado de forma autónoma por la misma plataforma para ser utilizada por los mismos widgets y demás entidades. Se admiten tres opciones de base de datos, SQL, NoSQL o una combinación de ambos. Sin embargo NoSQL está obsoleto por lo que para el proyecto se eligió implementar la opción de SQL.

- **Control Remoto:** ThingsBoard permite enviar llamadas de procedimiento remoto (RPC por sus siglas en ingles *Remote Procedure Calls*) desde aplicaciones del lado del servidor a dispositivos y viceversa. Básicamente, esta función permite enviar comandos a/desde dispositivos y recibir los resultados de la ejecución de los mismos. También permite asignarle atributos a los dispositivos y modificarlos activamente mediante widgets. Con esto se logra enviar un comando hacia un dispositivo. Estos fueron los dos métodos empleados para lograr la comunicación desde Thingsboard hacia los demás dispositivos de la red.

7.3. Arquitectura

Thingsboard ofrece diversos tipos de planes, para este proyecto se utilizó la versión Community, que es gratuita y requiere ser instalada en una equipo local. Se utilizó la UC como equipo local que cuenta con una computadora Raspberry Pi con sistema operativo Raspberry Pi OS basado en Debian-Linux. En la figura 7.1, se presenta la arquitectura de Software de la computadora.

La instalación de Thingbsoard se llevó a cabo en el Sistema Operativo Raspberry Pi OS, en donde se ejecuta como un servicio. Se programó para que se pueda acceder remotamente a la computadora mediante protocolo SSH o VNC. Para lograr manejar distintas ventanas de línea de comando desde SSH se utilizó el software *tmux*, se programó para que se levante Thingsboard, se abra el *Tunslip* (tunel SLIP con BR) y se ejecuten todos los *scripts* desarrollados en Python que

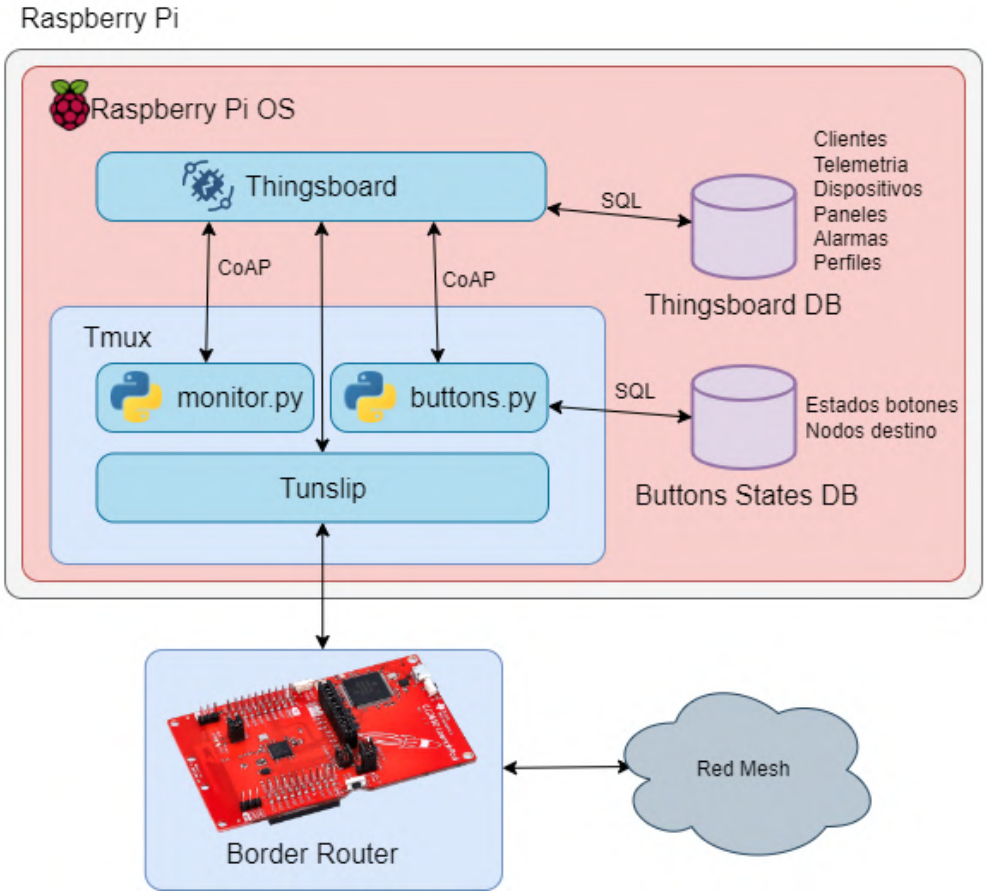


Figura 7.1: Arquitectura de Software Raspberry

se encargan de la comunicación con Thingsboard.

Tunslip permite conectar la computadora con la red inalámbrica 6LoWPAN, sin la necesidad de tener una radio apta para el protocolo IEEE 802.15.4.

7.4. *Scripts* en Python

En la sección 5.3 se explica de forma general el propósito por el cual se desarrollan *scripts* en Python, y se explican los mensajes intercambiados entre Thingsboard y estos *scripts*, así como la necesidad de que estos sean clientes CoAP y se suscriban a los recursos asociados a comandos de Thingsboard.

Se ejecutan tres *scripts* de Python con distintos propósitos. El *script buttons.py* surge para resolver una característica de funcionamiento de la plataforma, ya que no existe ningún Widget de comando que recuerde el último valor de comando. Al recargar un panel la plataforma consulta hacia el dispositivo suscrito cual es el estado. El sistema diseñado debe proveer la capacidad de controlar ciertos dispo-

sitivos activos (aire acondicionado, deshumidificador, entre otros), enviando tanto comandos On/Off para el uso manual del dispositivo, como para activar o desactivar el automatismo. Para ello, se le debe presentar al usuario final una interfaz gráfica con botones para la interacción.

Para poder contestar las consultas de Thingsboard, se utiliza un módulo de Python llamado *tinydb* que brinda la posibilidad de tratar un archivo con estructura de datos JSON como si fuera una base de datos permitiendo utilizar el lenguaje SQL para leer, escribir o modificar valores en el mismo. Cuando se recibe un mensaje asociado a algún botón, se guarda este valor en la base de datos.

Otra función primordial que ejecuta este *script* es enviar a la UCD correspondiente un mensaje con el comando recibido, se tiene un rol de intermediario en ese sentido.

Por otro lado, el *script monitor.py* cumple una función similar al anterior, pero asociado a parámetros de configuración del sistema de control. Otro de los elementos gráficos que se debe presentar al usuario final es uno capaz de ingresar un valor numérico y enviarlo hacia las UCD. Los parámetros que se configuran son del tipo setpoints y histéresis asociados a cada lazo de control implementado. Para ello existe un widget con un campo de entrada y un botón de envío incluido. Sin embargo, este botón al ser presionado, no solo ofrece la opción de enviar una llamada de procedimiento remoto con un mensaje conteniendo el valor ingresado en el campo de entrada, sino que también ofrece la opción de actualizar un atributo con el valor ingresado. Un atributo no es más que una variable que se guarda en Thingsboard.

En conclusión, ambos *scripts* actúan como intermediarios entre Thingbsoard y la red inalámbrica 6LoWPAN, pero únicamente en los mensajes asociados a elementos activos de widgets.

Finalmente, se tiene un tercer *script* de nombre *time.py* que tiene como objetivo únicamente enviar la hora de forma periódica a UCD-R1 y UCD-R2. Las UCD de la sala de riego luego utilizan esta información en los lazos de control de la planta de agua.

7.5. Acceso remoto

En el Sistema Operativo de la computadora se instaló el *software* No-IP, el cual se encarga de tomar la dirección IP dinámica del *router* de Antel cada 5 minutos y enviársela periódicamente a los servidores de No-IP. Estos servidores relacionan esa dirección IP que cambia cada 12 horas y la relacionan con un nombre de dominio fijo. Es decir mantienen actualizado un registro DNS de tipo A. De esta forma, consultando dicho nombre de dominio fijo, es posible conocer la dirección IP dinámica del router de la instalación en todo momento.

Se asocia dicha IP a un *hostname* gratuito, en este caso eligió la URI *smc.webhop.me*. Se debió utilizar este programa debido a que como no se cuenta con una IP estática, no era posible publicar a internet la web local generada por Thingsboard como

interfaz de usuario. Esta página web se publica en el puerto 8080. Para que esta aplicación tuviera utilidad también se debió redirigir el tráfico de este puerto hacia la dirección IP local de la computadora Raspberry Pi, desde la configuración del *router*. Se explica más en detalle en el apéndice B.

Para conectarse de forma remota a la computadora por SSH o VNC, también se debió redireccionar el tráfico del puerto 22 hacia la dirección IP local. Este acceso es muy útil para el mantenimineto del sistema.

7.6. Cadena de reglas

Como se mencionó anteriormente, Thingsboard posee una cadena de reglas para procesar todo dato entrante/saliente la cual se configuró para el proyecto con una lógica como se presenta en el diagrama de la figura 7.2. En esta sección se busca detallar la cadena que gobierna el registro de telemetría así como el disparo de alarmas.

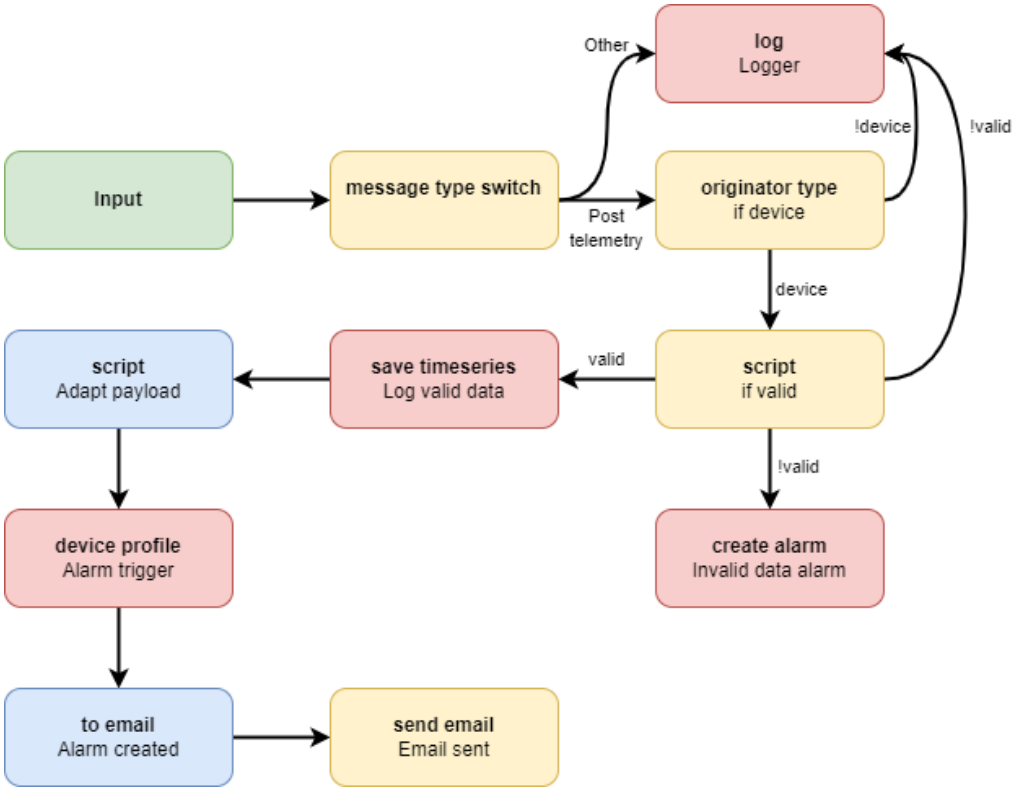


Figura 7.2: Lógica cadena de reglas

Los bloques rojos realizan acciones, los bloques amarillos realizan filtros, los bloques azules aplican algún tipo de transformación y el bloque verde representa la entrada de la cadena.

Cada mensaje entrante ingresa a la cadena por el bloque *Input*. Luego pasa por *message type switch*, se verifica si el mensaje es un mensaje del tipo POST con telemetría, en cuyo caso se encamina el mensaje hacia el bloque *originator type*, o en caso contrario se registra el mensaje en un log.

El bloque *script* ejecuta un código que tiene la función de verificar la validez del mensaje entrante, es un primer filtro para descartar mensajes que contienen valores erróneos detectables. Si el mensaje es válido, sigue hacia el próximo bloque, en caso contrario se registra en un log.

save timeseries simplemente guarda en la base de datos la telemetría contenida en dicho mensaje para ser usada por los Widgets en los paneles. El siguiente bloque *script* ejecuta otro bloque de código con el objetivo de adaptar el contenido del mensaje. Algunos valores de telemetría requieren un procesamiento básico antes de desplegar.

El bloque *device profile* activa las alarmas configuradas en el perfil del dispositivo asociado al mensaje recibido. En caso de que se active alguna alarma, el bloque *to email* genera un cuerpo y encabezado para rellenar en la estructura de un correo, y finalmente *send email* envía el correo generado a la dirección de correo configurada. Se realiza esto para tener un registro de las alarmas disparadas en el mail y además para alertar en tiempo real al usuario final.

7.6.1. Procesamiento de datos y filtrado

Como se comentó en la sección 7.6, una de las funciones que implementa la cadena de reglas es realizar un procesamiento de datos y filtrado antes de guardarlos en el sistema. Parte del procesamiento de datos se implementó en Thingsboard por comodidad ya que podría llevarse a cabo en las UCD o NM. El filtrado se realiza para eliminar valores de telemetría que están fuera de rango, la lectura de los sensores a veces presenta algún pico con un valor fuera de rango. Esto se puede dar por ruido electromagnético o por un mensaje de error por parte del sensor que se representa como un valor fuera de rango. Lo último suele ocurrir con los sensores con protocolo de comunicación (esto no representa un mal funcionamiento en el sensor, podría simplemente deberse a un error en el sincronismo o ruido en la comunicación).

En la tabla 7.1, se ilustra el procesamiento y filtrado llevado a cabo según variable. Se agrupo bajo el nombre 'Digital' a las variables que representan estados, que pueden valer 0 o 1.

7.6.2. Alarmas

En los últimos bloques de la cadena de reglas, se verifica el valor de la telemetría para disparar o no alarmas. Éstas se presentan en la tabla 7.2, donde se puede apreciar la condición de disparo y el mensaje desplegado en el alarmero.

Variable	Filtrado (entre)	Procesamiento
Temperatura suelo	[-5500, 12501]	/100
Temperatura SoC	[-40, 85]	No aplica
Humedad suelo	[0, 15000]	/100
pH	[0, 140]	/10
Alimentación SoC NM	[0, 42000]	/10000
Alimentación SoC UCD	[0, 4200]	/1000
Carga batería	[0, 100]	No aplica
Digitales	[0] o [1]	No aplica

Tabla 7.1: Procesamiento y filtrado según variable

Variable	Alarma	Mensaje
Temperatura suelo	<15	Baja temperatura maceta
	>30	Alta temperatura maceta
	= 125,01	Error sensor de temperatura suelo
Carga batería	5<carga<15	Carga batería <15 % maceta
	<5	Carga batería <5 % maceta
Temperatura SoC	<10 o >30	Temperatura SoC fuera de rango
Falla HVAC o MIC	[1]	Falla estado
Alimentación UCD	<3 o >3.6	Alimentación UCD fuera de rango
Parámetros de configuración	255	Parámetro sin definir

Tabla 7.2: Condición de disparo de alarma y mensaje desplegado

7.7. Paneles

Se crearon distintos paneles para el proyecto, los cuales se detallan a continuación:

- **General:** Este panel tiene como objetivo dar un vistazo general del estado de las variables más relevantes de cada sala del club, así como mostrar todas las alarmas activas y sin reconocer del mismo.

- **Floración:** Se muestra las variables relacionadas con la sala de floración, es decir temperatura y humedad suelo de los NM, batería de los NM, temperatura y humedad aire, concentración de CO_2, estado y ciclo de trabajo de las lámparas. También es posible realizar el control del equipo de aire acondicionado de la sala. Este último involucra el control del automatismo del mismo, la configuración del setpoint y de la histéresis.

- **Vegetativo:** Se presentan las últimas lecturas de temperatura, humedad aire y estado de las luces. Permite realizar el control del equipo de aire

acondicionado de la sala, así como la configuración de los parámetros de control.

- **Secado:** Acá se muestra la temperatura y humedad ambiente de la sala. Este panel también cuenta con los widgets necesarios para realizar el control del equipo deshumidificador de la sala y configuración de parámetros de control.
- **Riego:** Este panel tiene como objetivo permitir la visualización de las variables de riego y configurar los parámetros de control vinculados a la planta de agua.
- **Interno:** En este panel se pueden observar aquellas series de datos relacionados con el mantenimiento del sistema. En particular se puede observar la temperatura interna y la tensión de alimentación de cada SoC CC2650.

En la figura 7.3 se muestra, a modo de ejemplo, el panel de la sala vegetativo. El resto de los paneles se pueden encontrar en el manual de usuario del apéndice C.

Figura 7.3: Panel Vegetativo.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 8

Puesta en marcha

En este capítulo se busca detallar la puesta en marcha del sistema, partiendo de las pruebas realizadas en el ambiente de desarrollo del sistema, las instalaciones realizadas en el club cannábico y las pruebas realizadas en sitio.

8.1. Pruebas FAT

Las pruebas FAT (Factory Acceptance Test) son un proceso de aceptación de un equipo en fábrica, antes de ser trasladado a las instalaciones del cliente. Estas pruebas tienen el objetivo de validar el diseño y funcionamiento del sistema y es un proceso que debe ser aprobado por el cliente. En nuestro caso, llevamos a cabo pruebas funcionales en nuestro ambiente de desarrollo, pero no se realizó la verificación con el cliente, por lo que la validación fue dada por el equipo de proyecto.

Si bien fue necesario realizar pruebas intermedias durante las distintas etapas de desarrollo, es importante la validación del sistema en su completitud, dado que en este punto pueden surgir problemas de integración entre las distintas partes. Llamamos pruebas FAT a la verificación del funcionamiento del sistema completo, dividido en bloques funcionales. Estos bloques corresponden a cada una de las placas desarrolladas, junto con sus componentes y dispositivos adyacentes.

8.1.1. Unidad Central

Para realizar las pruebas funcionales tanto de la Unidad Central como de los nodos maceta se planteó el sistema presentado en la figura 8.1.

Para la unidad central se realizaron las siguientes pruebas funcionales:

- Acceso remoto: Se verificó que el acceso remoto funcionaba correctamente, y era posible acceder al sistema por protocolo SSH y por VNC. También se comprobó el acceso remoto a la interfaz del SCADA, accediendo a la IP que la UC publica.

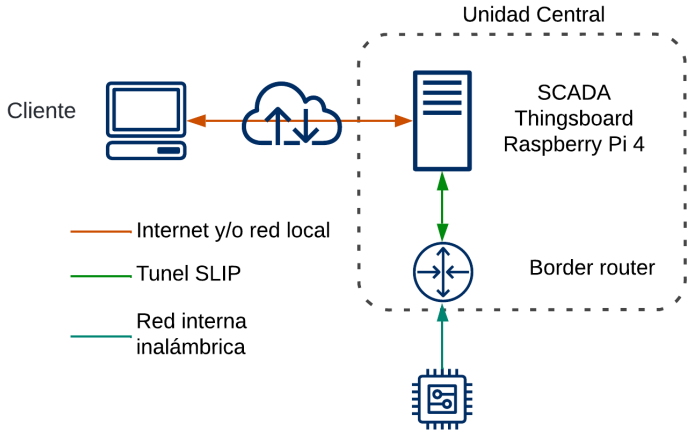


Figura 8.1: Sistema armado para probar Unidad Central

- Inicialización del sistema: Al enchufar la UC, se verificó que ésta se encendía correctamente y se ejecutaba el *script* encargado de inicializar el sistema.
- SCADA: Se verificó que la plataforma Thingsboard inició correctamente y que los *scripts* de Python se asociaron a los recursos correspondientes.
- Comunicación: Se verificó que el BR de la UC funcionaba correctamente, logrando comunicar un nodo con el BR.

8.1.2. Nodos Maceta

Mientras que para los nodos maceta, se realizaron las siguientes pruebas funcionales: **??**.

- Comunicación red inalámbrica: Se verificó desde el BR, que los nodos maceta se asocien al mismo. Esto se logró activando los mensajes *LOG* (mensajes de *debug* de Contiki-NG) a nivel de capa de enlace. En una distancia de 10m, las placas demoran apenas algunos segundos en formar la red.
- Comunicación desde NM a UC: Se verificó tráfico de mensajes utilizando el *sniffer* de red o analizador de protocolos Wireshark. Se pudo comprobar que los nodos macetas enviaban mensajes de forma periódica tanto a la dirección de Thingsboard con el formato JSON adecuado, como a la dirección de la UCD-R con el formato texto plano. También se verificó que el intercambio de mensajes cliente-servidor era el adecuado.
- Visualización en SCADA: Se verificó que desde la plataforma Thingsboard se recibieran los mensajes de telemetría correctamente, y se desplegaran en la pantalla correspondiente.

En la figura 8.2, se presenta una foto de un Nodo Maceta.

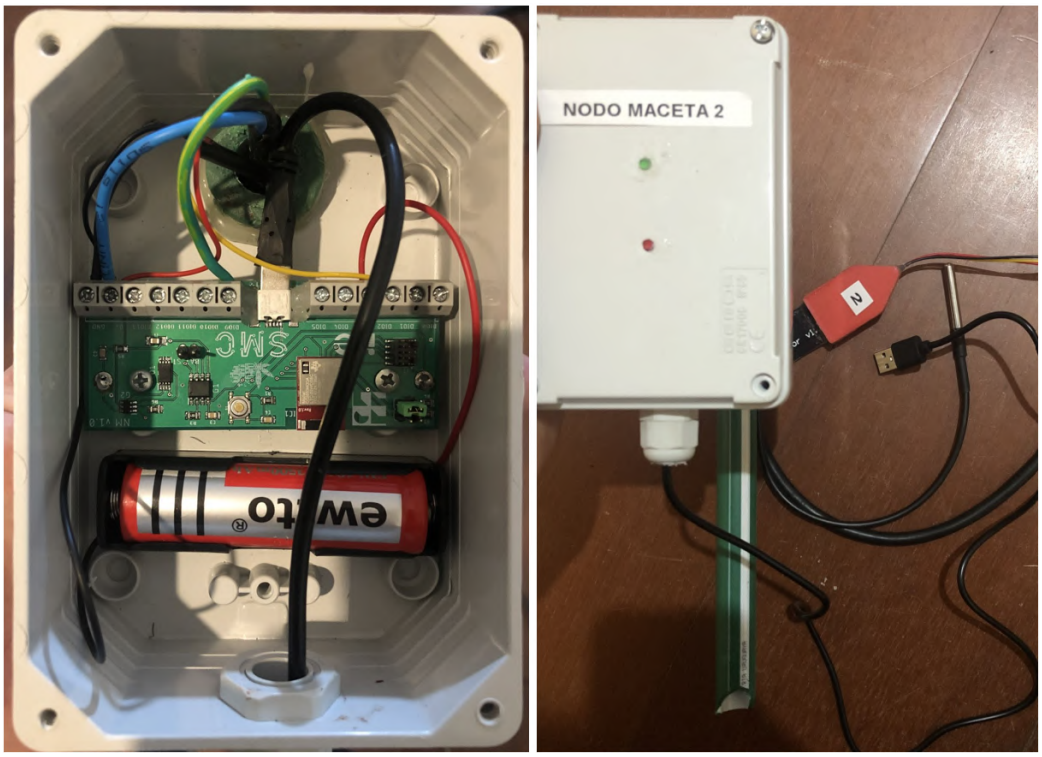


Figura 8.2: Nodo Maceta

Mejoras realizadas

Durante la primera prueba, se verificó que el funcionamiento de los nodos maceta en protocolo CSMA era de 4,5 días. Esta prueba disparó la motivación de cambiar el protocolo en capa de enlace a TSCH como se describió en la sección 5.2.2 obteniendo contra lo esperado una duración similar.

8.1.3. Unidades de Control Distribuido: UCD-V, UCD-F y UCD-S

Tal como se describe en la sección 4, se crearon tres tableros que contienen una placa UCD cada uno. En términos de funcionalidades las UCD-V, UCD-F y UCD-S son muy similares, por lo tanto, se presentarán en conjunto en esta sección. De todas formas, las pruebas se realizaron de forma independiente probando cada tablero por separado.

Se dispone el sistema presentado en la figura 8.3 para realizar las pruebas funcionales. Para simplificarlas se utilizó una laptop en lugar de la UC y se programó la placa para actuar como UCD y como BR. Esto es posible ya que Contiki-NG permite manejar el BR como un módulo que se ejecuta en paralelo. Además, ya se había verificado anteriormente que las placas no presentaban dificultad en asociarse al BR.

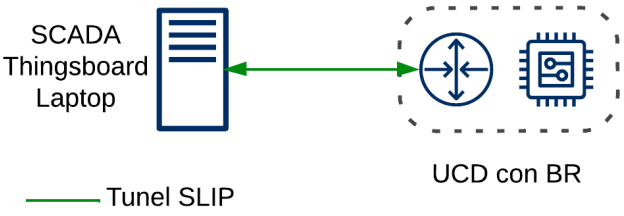


Figura 8.3: Sistema armado para probar las UCD.

A seguir se listan las pruebas realizadas:

- Comunicación desde UCD a UC: En esta etapa pudo verificarse correctamente que las UCD enviaban los mensajes tanto a Thingsboard como a otra UCD, con el formato esperado.
- Comunicación entre UCD: Se verificó primero tranmismitiendo mensajes desde un cliente CoAP en Linux a las direcciones configuradas en las UCD para recibir valores. Posteriormente, se verificó la comunicación entre las UCD de forma directa (transmisión y recepción), conectados a la UCD mediante UART se pudo verificar los mensajes recibidos.
- Visualización en SCADA: Se verificó que desde la plataforma de Thingsboard se recibían los mensajes de telemetría correctamente y estos se lograran visualizar en las pantallas correspondientes.
- Configuración de parámetros y envío de comandos desde SCADA: Se verificó el correcto funcionamiento de los *scripts* desarrollados en Python que actúan de intermediarios. Se pudo validar que la UCD recibía correctamente los mensajes provenientes del SCADA.
- Pruebas de control: Se verificó el funcionamiento de la lógica de control, tanto del comando apagado, manual y automático desde la selectora local, como del comando recibido por el SCADA. Se verificó que la programación respecto a los lazos de control funcionara de forma correcta.

Mejoras realizadas

Durante las primeras pruebas se identificó la molestia que representaba al usuario la espera para verificar en el SCADA el mensaje de confirmación de un comando o parámetro recibido. Esta espera surge del período de envío de mensajes de telemetría, configurado en 40 segundos. Esto motivó a desarrollar un proceso que envía un mensaje de telemetría a los 500 ms de recibir un comando o configuración de parámetro. En la figura 5.12, se ilustra un mensaje de este tipo.

Por otra parte, en primera instancia se estaba trabajando con contenedores de Docker en la UC. Docker es una aplicación que permite crear procesos en segundo plano y de forma auto contenida tan complejos como sistemas operativos enteros. Esto permitía armar el sistema de forma rápida, en caso de que haya que arrancar con la UC de cero. Además, agregaba flexibilidad al encapsular el entorno de trabajo, ya que no importaba en que máquina se trabajara. En este Docker se había instalado Thingsboard, Contiki-NG y se ejecutaban los *scripts* en Python. Analizando con Wireshark el intercambio de mensajes se pudo visualizar que Thingsboard no estaba funcionando correctamente, dado que a veces perdía registro de los suscriptores de su recurso (lo cual significaba que no era posible enviar comandos), y además se generaban retransmisiones. Investigando en internet, se comprobó que varias personas habían tenido problema al utilizar Thingsboard en un contenedor [71]. Se procedió a verificar el funcionamiento con todo el sistema instalado en la raíz de un Sistema Operativo y se pudo validar que en estas condiciones el sistema funciona correctamente, no se dan retransmisiones y la suscripción a los recursos tiene tiempo ilimitado.

8.1.4. Unidades de Control Distribuido: UCD-R1 y UCD-R2

La UCD-R1 y UCD-R2 están contenidas en un mismo tablero (UCD-R). En esta etapa no se contaba con UC, ya que ésta ya estaba instalada en el establecimiento del cliente, por lo que en su lugar se utilizó una laptop.

Para realizar las verificaciones se tiene el sistema presentado en la figura 8.4, donde UCD-R1 también cumplía el rol de BR, al igual que en el caso descrito en 8.1.3. Cabe destacar que el tablero UCD-R está conectado a un block de riego, que contiene el circuito y los actuadores necesarios para el sistema de riego. El block de riego se describe en el final de esta sección.

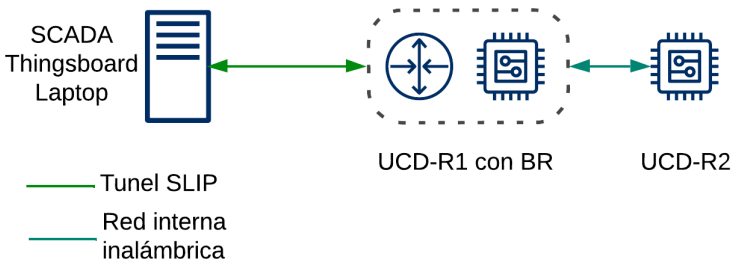


Figura 8.4: Sistema armado para probar UCD-R1 y UCD-R2

Se realizaron las siguientes pruebas funcionales:

- Comunicación desde UCD a UC: Se verificó que los mensajes de telemetría se enviaban a la dirección correcta con el formato adecuado.

- Configuración de parámetros y envío de comandos desde UC: Se comprobó que enviando comandos desde un cliente CoAP se registraban en la UCD de forma correcta.

En las figuras 8.5 y 8.6, se presentan respectivamente fotos de la UCD-R y del block de riego.

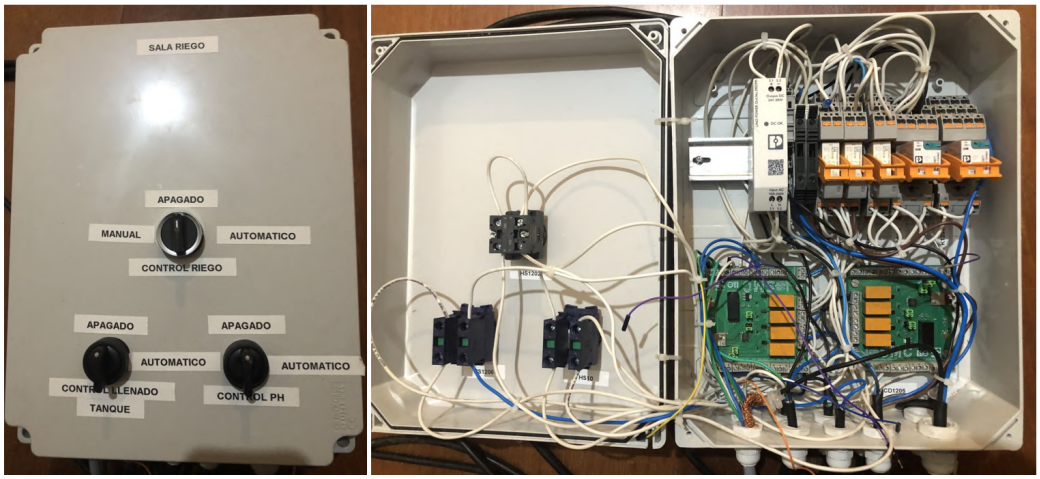
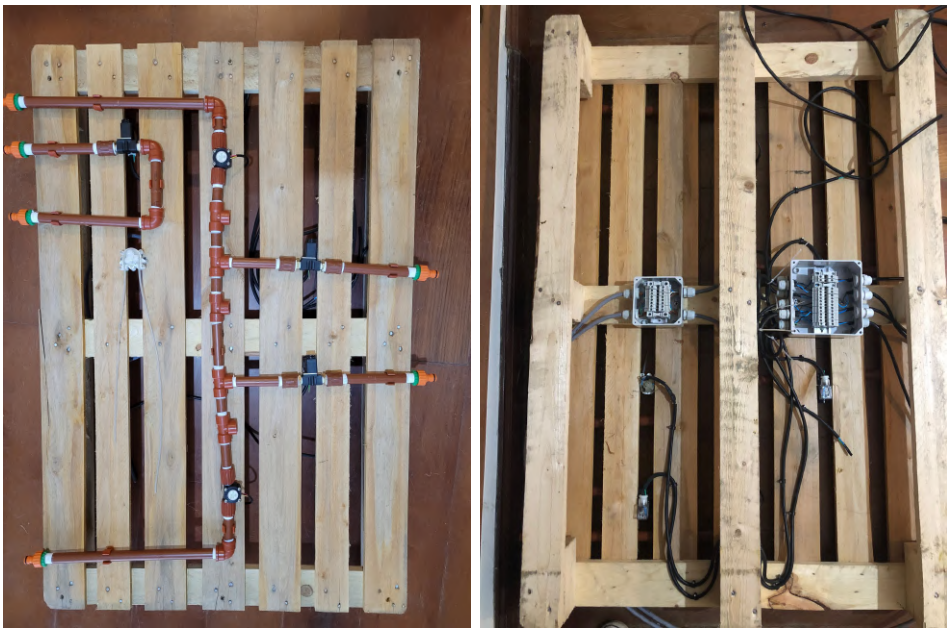


Figura 8.5: Tablero UCD-R



(a) Vista frontal. (b) Conexiones eléctricas.

Figura 8.6: Block de riego

Pendiente

Queda pendiente realizar una prueba funcional de la lógica de control del sistema de riego, que permita validar el correcto funcionamiento del sistema. Se llegó a probar las selectoras y se hizo una primer prueba funcional de la programación, donde enviando la hora desde un cliente CoAP se logró ejecutar el ciclo de control y verificar que los actuadores respondían correctamente.

Quedó pendiente realizar la prueba desde la pantalla final de Thingsboard, validar el correcto funcionamiento de los *scripts* desarrollados en Python, así como el registro de la telemetría por parte de Thingsboard.

Block de Riego

El proyecto incluye la implementación de una planta de agua, destinada al tratamiento del agua, su circulación, el monitoreo de algunas variables y el riego. Esto implica la instalación de varios dispositivos cercanos entre sí y un circuito hidráulico, en una sala que carece de espacio físico. Para facilitar la instalación, pruebas y uso de los componentes involucrados, se diseña un block de riego donde los mismos puedan ser afirmados y manipulados de forma sencilla.

El block de riego consiste en un pallet de 120x80 cm, sobre el cual se instalan los sensores, actuadores y caños y conexiones del circuito hidráulico. Los caños, de polipropileno y cortados a medida para la aplicación, son ajustados con grampas del mismo material, y conforman un circuito que permite el ingreso de agua desde la bomba de recirculación, el desvío hacia las macetas y el retorno al tanque para completar el recorrido. Además, se agregan caños para un mínimo recorrido (independiente del anterior), destinado al llenado del tanque. Se incluyen también acoples rapidos para mangueras de 1/2", en todas las entradas y salidas del circuito.

Conectados a estos caños se instalan los dos caudalímetros. Entre ellos se cuenta con cinco conexiones de tipo T, para generar cinco desvíos en el recorrido. De esta forma, si bien se colocarán dos electroválvulas de riego, se tiene previsto lugar para cinco de ellas. Además de los desvíos mencionados, se coloca una tercera electroválvula en el circuito independiente, a fin de controlar el ingreso de agua al tanque. Este circuito consiste en una vuelta en U, necesaria para afirmar la electroválvula al block.

Se incluye también la bomba peristáltica de dosificación. Esta bomba cuenta con un tubo siliconado de suficiente extensión para alcanzar, por un lado, el recipiente de ácido fosfórico, y por otro, el tanque de agua.

Finalmente, en el reverso del pallet, se cablean todos los dispositivos a cajas estanco con borneras de tipo riel din, para una conexión eléctrica segura, siendo que va a ser ubicado en un entorno de riesgo de mojaduras.

Tanto el sensor de pH como el sensor de nivel, no son incluidos en este block dado que deben ser ubicados dentro del tanque de agua.

El resultado se puede observar en las imágenes de la figura 8.6.

8.2. Instalación del sistema

La instalación se llevó a cabo en varias visitas. Se realizó una planificación basada en el avance secuencial sala por sala.

Como criterio general se buscó optimizar los tiempos paralelizando las distintas tareas, y asignándolas por rubro. Esto es, se designa una persona encargada de amurar los tableros, una segunda persona se encarga del tendido de los cables y ductos para los dispositivos, etc., distribuyendo así las distintas actividades.

Luego de cada instalación, se procede a realizar las conexiones eléctricas para continuar con las pruebas SAT (Site Acceptance Testing).

8.2.1. Instalaciones realizadas

En la figura 8.7 se puede visualizar la Unidad Central, mientras que la 8.8 ilustra el tablero de la UCD-V, ubicado en SV. La figura 8.9 presenta el tablero UCD-S, ubicado en SS. En la figura 8.10, es posible apreciar un sensor de temperatura y humedad ambiente (DHT-11) de la SV y el enchufe que se actúa para controlar el deshumidificador de la SS.

Figura 8.7: Unidad Central

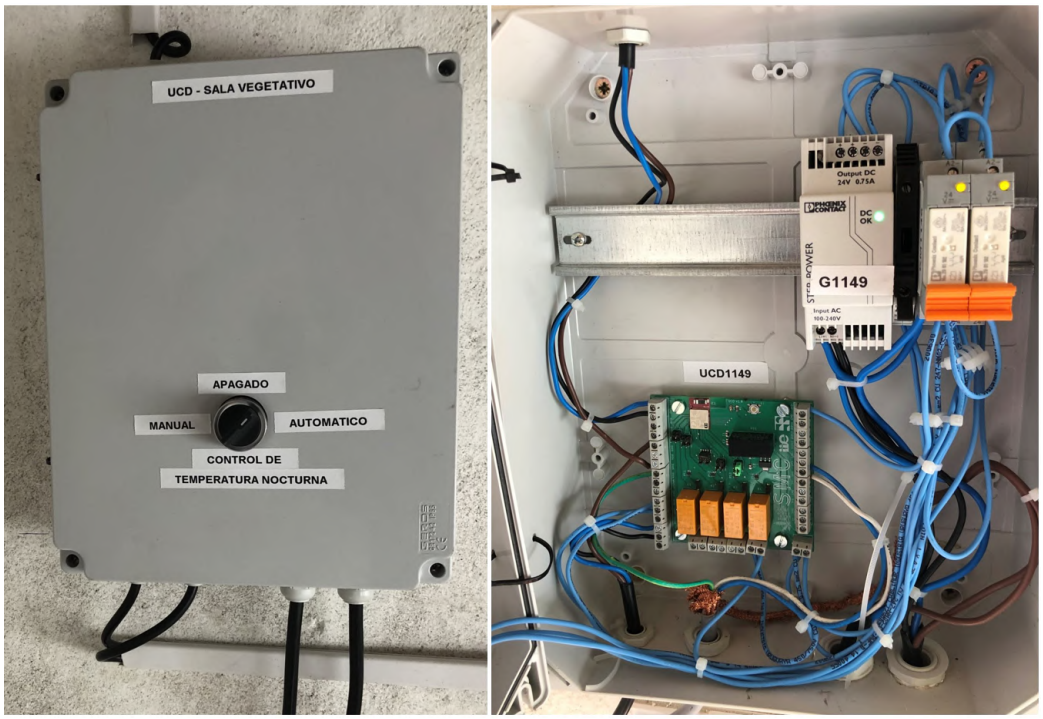


Figura 8.8: Tablero UCD-V

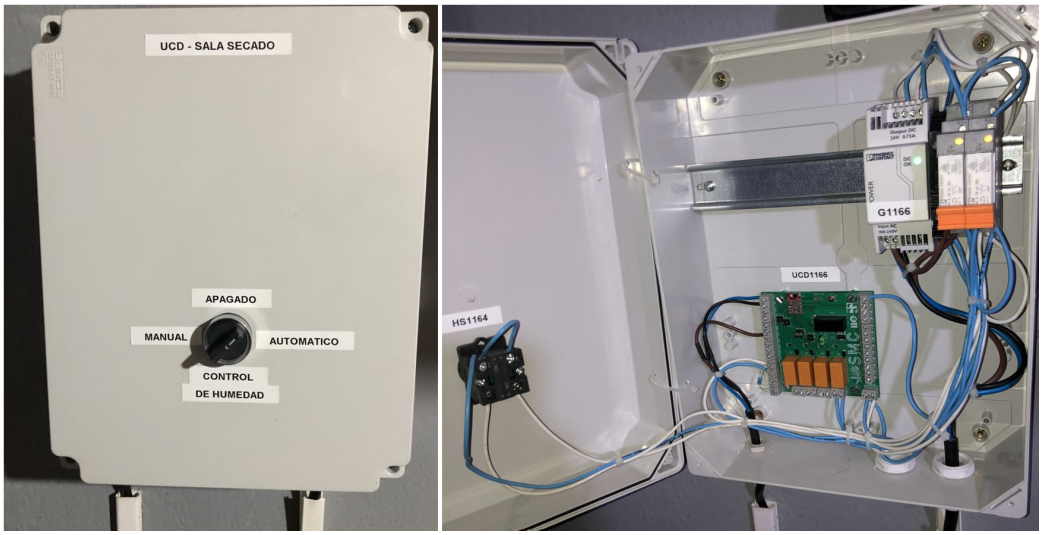


Figura 8.9: Tablero UCD-S

8.2.2. Pendiente

Queda pendiente de instalar lo referido al riego y planta de agua. En este caso, la modificación física de instalación conlleva muchas horas, y no se pudo lograr implementar en el tiempo esperado.

Figura 8.10: Izquierda: Sensor DHT-11. Derecha: Enchufe controlado Deshumidificador SS

También queda pendiente dejar en la instalación del cliente los nodos maceta. Estos aún no se encuentran instalados en la maceta, pero se instalarán junto con el riego y la planta de agua.

Hasta el momento se tienen instalados un sensor de luz por sala, a fin de verificar el funcionamiento del sistema. Se espera completar la instalación según el diseño en instancias próximas.

8.2.3. Mejoras en instalación

Se pudo identificar que el conexionado eléctrico de los sensores requiere de mejoras. Para acceder a sus salidas y bornes de alimentación, los sensores instalados cuentan con pines de un material que no permite una buena soldadura directa a un cable. A su vez, los conectores correspondientes a estos pines no encajan de

forma confiable y firme. Estas condiciones presentan riesgos de falsos contactos o desconexiones de los sensores, que pueden comprometer la disponibilidad del sistema y requerir mantenimientos o reparaciones inesperadas.

8.3. Pruebas SAT

Las pruebas SAT (Site Acceptance Test) también son un proceso de aceptación, en este caso llevado a cabo en el sitio a instalar el equipo. Con estas pruebas se valida la correcta integración del sistema al sitio, por lo que se realizan con el cliente final.

8.3.1. Pruebas realizadas

Las pruebas realizadas no entraron tanto en detalle como las pruebas FAT, debido a que detalles de mensajes y comunicación ya se habían verificado, simplemente se procedió a:

- Chequeo de conexión eléctrica: Lo primero que se hizo fue verificar con un multímetro que las conexiones eléctricas se hayan realizado de forma correcta, con el equipo desenergizado. Luego se procedió a energizar el sistema y se verificó que los niveles de tensión en los distintos puntos del circuito eran los correctos.

- Verificar conexión de UCD a UC: Se verificó desde la UC, que la UCD se asociaba de forma correcta al BR, y que el SCADA recibía mensajes de telemetría.

- Configuración de parámetros y verificación funcional del sistema: Primero se realizaron pruebas de funcionamiento del sistema, donde se pudo comprobar el sistema de control. Posteriormente se dejó el sistema configurado y se validó la puesta en marcha.

A modo ejemplo, en la figura 8.11 se ilustra una de las pantallas, la cual permite visualizar parte del funcionamiento de la lógica de control de la temperatura nocturna de la SV. Mirando con atención la figura, se puede notar que la temperatura de la sala (señal azul) disminuye cuando el aire acondicionado está apagado (señal gris pasa de 1 a 0). La temperatura desciende durante 6 h, llegando al rango de trabajo deseado (rango indicado con líneas horizontales rojas). A las 23 h se prenden las lámparas de la sala, pasando de noche a día, por lo que se prende también el aire acondicionado, llevando la temperatura al valor deseado para el horario diurno. En este caso operativo, la temperatura exterior es menor a la temperatura de la sala y el aire acondicionado termina siendo apagado durante las 6 horas del intervalo nocturno en su completitud.

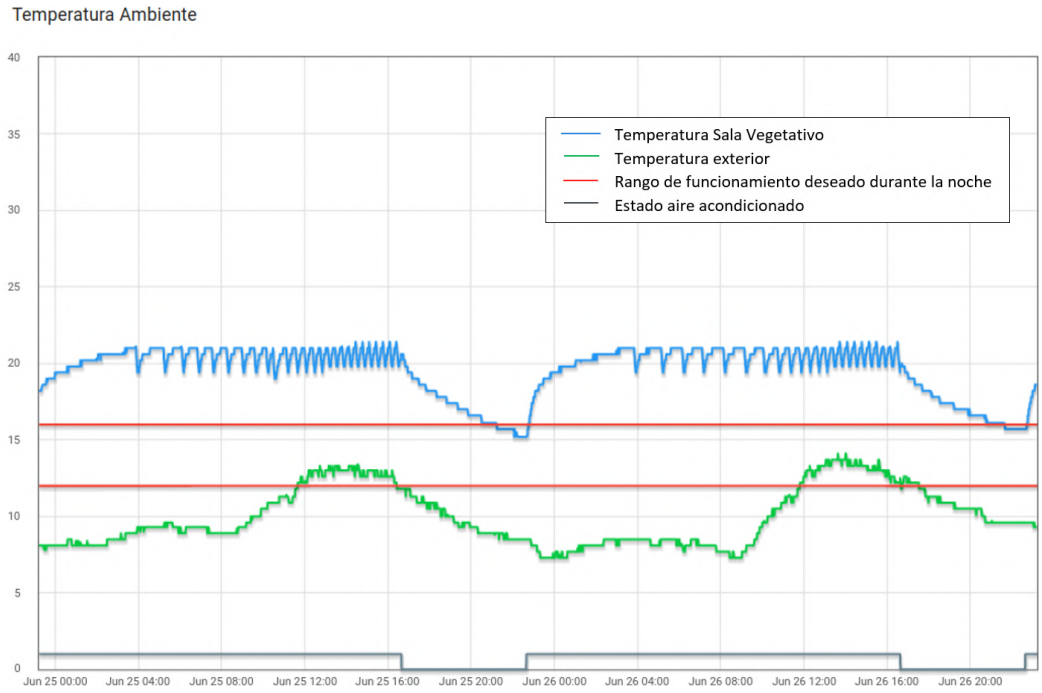


Figura 8.11: Tempearatura ambiente Sala Vegetativo

8.3.2. Pendiente

En cuanto a pruebas SAT, queda pendiente lo referente al riego y la planta de agua, supeditado a su instalación. Una vez completadas estas tareas, se podrá proceder también con la determinación de parámetros (*i.e.* caudal de riego, etc.) y calibración de sensores, como los caudalímetros.

Capítulo 9

Conclusiones

Se logró diseñar e implementar un sistema de monitoreo y control, permitiéndole al usuario final tener un registro de las principales variables del proceso productivo. Esto le da trazabilidad a su producción, proporcionándole información para explorar mejoras en producción.

Se logró diseñar cuatro lazos de control que aportaron al proceso eficiencia, fiabilidad y lo acercó a la independencia de la operación humana. Los lazos de control solucionan buena parte de las inquietudes del cliente, en especial respecto al riego.

El proceso de desarrollo de las placas de los controladores fue notoriamente positivo para el equipo. Se diseñaron dos placas de considerable tamaño y diversas funciones, optimizando espacios, funcionalidades, esfuerzos y costos. Todos los componentes fueron elegidos mediante decisiones fundadas, procurando la mejor opción, en un sentido integral. Se recorrió un período considerable de investigaciones, correcciones internas y consultas, que contribuyeron al enriquecimiento de los conocimientos técnicos del equipo y al desarrollo de la experiencia personal. Se obtuvo no solo información específica en relación a la electrónica de circuitos, sino también conceptos y buenas prácticas.

En el caso de las placas, también se tuvo en cuenta la escalabilidad del proyecto. Se incluyeron más borneras de las necesarias, buscando acceder a la mayor cantidad de entradas/salidas del SoC posibles. Se incluye por ejemplo la opción de desactivar cualquiera de los cuatro relés de la UCD de forma de liberar un puerto y utilizarlo para otro sensor o dispositivo. Las UCD también se instalaron en gabinetes estanco de un tamaño mayor al requerido, posibilitando la inclusión de otros componentes.

Además, entendemos que el resultado técnico de las placas electrónicas fue muy bueno, teniendo en cuenta que las 10 unidades ensambladas funcionaron correctamente desde un primer diseño, sin perjuicio de las mejoras que se pudieran realizar. Se completó el ciclo de desarrollo de una placa electrónica, desde el diseño a la aplicación, incluyendo el pedido de fabricación, la definición, compra y soldado de componentes y las pruebas eléctricas.

Durante el proceso de diseño en general, se tuvieron en cuenta las diversas eventualidades que pudieran surgir y se incorporaron contingencias que le aportan robustez al sistema.

Se tiene que los parámetros de configuración no se pierden en caso de que se apague o reinicie el sistema, sea voluntaria o involuntariamente. Tampoco se pierden mensajes en la comunicación, debido a las retransmisiones. Por otro lado, la utilización de los mensajes periódicos, así como el envío de estados y parámetros configurados, minimizan los problemas de sincronismo que puedan surgir. Además, en caso de desconexiones de la red inalámbrica, los lazos de control se continúan ejecutando, tomando el último valor guardado en los casos que dependan de variables sensadas por otros controladores.

El diseño final prioriza al usuario final, buscando que las funcionalidades y la operación del sistema sean una solución y no un problema.

Los controles implementados tienen la opción de desactivarse de forma local (utilizando la selectora correspondiente) o remotamente (a través del SCADA). Esto permite al usuario realizar el control de forma manual, sea por fallas puntuales o por decisiones de operación.

Además, el sistema presenta una interfaz amigable e intuitiva, lo cual permite al usuario final aprender con rapidez y poder manejar las funciones sin mayores inconvenientes.

Por otra parte, manteniendo el usuario en el foco, se implementó un sistema de alarmas que notifica al cliente ante cualquier evento extraordinario. Se logró también integrar estas alarmas con una cuenta mail para realizar las notificaciones. Dependiendo del motivo de su activación, el sistema de alarmas genera un encabezado y un cuerpo acorde para rellenar el correo electrónico enviado, de forma que el origen de la alarma sea entendible al recibir el correo.

El sistema forma una red *mesh*. Esta arquitectura tiene la ventaja de que permite incorporar nuevos nodos de forma rápida y sencilla dado que cada nodo puede recibir y enviar información de los demás nodos. También se obtiene redundancia, volviéndolo un sistema más robusto ante fallos de desconexión. Además, si bien en principio no fue necesario para el proyecto debido a las características de la arquitectura, en el caso de las redes *mesh*, el rango de la red es fácilmente ampliable a diferencia de una red de tipo estrella. A su vez, uno de los criterios tomados al desarrollar la red buscaba reducir el tráfico para asegurar disponibilidad en la red. Por esta razón se agrupan varios datos en cada mensaje, reduciendo la cantidad de envíos.

Se eligió un SCADA llamado Thingsboard con capacidades de procesamiento y manejo de datos por encima de lo necesitado en el proyecto. El mismo permite la visualización y control de una cantidad de dispositivos que sobrepasa el limite máximo de los que podría llegar a necesitar el club. Este SCADA provee una versión de uso gratuito y presenta una interfaz más amigable e intuitiva que Mango OS, siendo ésta la otra opción contemplada para el proyecto.

En cuanto a la instalación, encontramos como primer reflexión que la estima-

ción de los tiempos no fue acertada, siendo que las tareas insumieron más días de lo planeado. Esto repercutió en el cronograma, generando atrasos y sobreesfuerzos. Creemos que se debió tener mayor especificidad en las tareas para mejorar la precisión de la estimación.

De todas formas, tanto las pruebas FAT como las pruebas SAT realizadas resultaron correctas, por lo que, entendemos que la instalación fue correcta, a la espera de las terminaciones pendientes.

Como conclusión final, si bien a lo largo de toda la duración de este proyecto se dieron contratiempos, retrasos, errores y demás problemas, entendemos que el equipo fue capaz de cumplir los objetivos planteados y desarrollar un sistema funcional.

9.1. Dificultades

La principal dificultad a la cual se enfrentó el equipo fue vinculado a la gestión del proyecto. La planificación de tareas, estimación de tiempos y reparto de trabajos resultó una práctica compleja. Entendemos que la falta de experiencia en muchas de las tareas nos llevó a definir estimaciones inexactas y generar contratiempos en las ejecuciones.

Por otro lado, el proyecto tenía una etapa inicial de definición, junto al cliente, que dejaba un margen amplio para diferentes implementaciones del objetivo. Esta situación generó que el equipo de proyecto, junto con el cliente y el tutor, dedicaran un tiempo considerable en definir los resultados concretos a los que se quería llegar.

A su vez, durante el transcurso del proyecto, el mismo fue evolucionando, con cambios en el foco y en los caminos a seguir, buscando satisfacer los requerimientos del cliente y los intereses académicos del equipo. Un ejemplo notorio se da en la red de comunicaciones, dado que se había definido en un principio incorporarla de un proyecto externo. Sin embargo, debido a que las opciones analizadas no colmaron las expectativas, se decide que el equipo desarrolle la red.

Como se describe en los capítulos anteriores, la red inalámbrica interna está basada en el protocolo 6LoWPAN. Si bien es una tecnología que ha ganado popularidad en los últimos años, sigue siendo un protocolo reciente, por lo que no resulta sencillo encontrar soporte para todos los contratiempos que se presentaron durante el desarrollo de la red.

El sistema operativo utilizado, Contiki-NG, es una evolución de otro sistema operativo (Contiki), por lo que también se da esta circunstancia, con sus consecuentes problemas.

En ocasiones se tuvo que invertir muchas horas de investigación y/o consultas a docentes capacitados para conseguir información que permita solucionar el inconveniente o descartar (de forma argumentada) el camino tomado. En otros casos, se debió desarrollar las soluciones alternativas, como fue el caso de los *scripts* de Python, explicados previamente.

Otra dificultad se da en la adquisición de componentes y materiales. Desde el brote de la pandemia, la disponibilidad de chips y componentes electrónicos se vio comprometida, llegando incluso a afectar industrias de gran porte, como la automotriz. En nuestro caso, se generaron situaciones de demoras o desabastecimiento de productos y componentes. En particular, tuvimos demoras en la compra de la computadora Raspberry Pi, dado que nos era imposible encontrar un proveedor con stock. En el caso de los componentes para las placas PCB, se tuvieron que hacer varias compras, en distintos tiempos y a distintos proveedores, para obtener algunos de ellos.

9.2. Trabajos futuros

A lo largo del desarrollo del proyecto se manejaron diversas funcionalidades, características y métodos que fueron descartándose, tanto por inviabilidad como por alcance.

En ese sentido, se destacan las variables que no son monitoreadas. Se presentaba al inicio del documento una descripción de las variables más importantes para el crecimiento de las plantas. De esas 9, actualmente se monitorean 5: concentración de dióxido de carbono y humedad y temperatura, de suelo y ambiente. Una de las mejoras más interesantes que se plantean consiste en agregar la intensidad de luz como variable monitoreada, dado que es la que más incide en la planta. Es recomendable también medir nutrientes del suelo.

Se puede plantear un razonamiento análogo referido a los lazos de control. Por ejemplo, monitoreando los nutrientes del suelo se puede implementar un lazo de control de fertilización automática. Por otro lado, sería interesante controlar la concentración de dióxido de carbono con un equipo emisor de este gas.

También se consideró un método de control de la temperatura ambiente distinto al implementado. Se indica en el capítulo de control que se tienen dos *set-points* de temperatura, uno para el horario con las luces encendidas, y otro para el ciclo nocturno. Puede resultar útil implementar una comunicación IR directa con el aire acondicionado, de forma de cambiar la temperatura configurada en el equipo al cambiar de ciclo.

La amplitud de este proyecto implicó que varias de las funcionalidades se instalarán a modo de prueba de concepto, por lo que se tienen varias oportunidades de escalabilidad.

Resalta en ese sentido el número de macetas controladas. En este proyecto se propone la instalación de dos NM, que si bien supone el riego de más macetas, no se alcanza la totalidad disponible. Se estima que con 6 NM en la sala de floración y otras 4 en la sala de crecimiento vegetativo, se puede alcanzar el 100 % de las plantas regadas de forma automática.

Otro punto propuesto es un análisis de intensidad luminosa en las salas de floración y vegetativo. Una de las preocupaciones del cliente previo al inicio del

proyecto era el costo que conlleva mantener los focos de luz encendidos entre 12 y 18 horas por día en las salas. El cliente plantea la posibilidad de optimizar la ubicación y potencia de las lámparas de luz, buscando un menor consumo sin modificar la productividad.

Respecto al diseño de los PCB se deja como trabajo a futuro mejorar el circuito de carga de las baterías de los nodo maceta. Actualmente se cuenta con un circuito basado en el integrado TP4056, que carga las baterías a una tensión de 4,2 V). Esta tensión es superior a la máxima soportada por el móduo CC2650MODA, fijado en 4,1 V según su manual. Dado que se excede en solo 2 %, y que dicho exceso dura solo lo que demore en cargarse la batería, se mantiene la configuración actual. De todas formas, se debe corregir este aspecto en eventuales futuras versiones.

En cuanto a la programación de los controladores del sistema, se plantea como trabajo a futuro mejorar la implementación del cliente CoAP con el fin de no tener que recurrir al uso de *scripts*. Durante el proyecto, si bien se desarrolló un cliente CoAP, se utilizaron *scripts* escritos en lenguaje Python para parte de la integración entre Thingsboard y los controladores. Respecto a estos *scripts* se debe también mejorar la robustez y confiabilidad de los mismos. Desarrollar algún método que verifique si por algún motivo se trancó el código o se perdió la suscripción con Thingsboard.

Respecto al SCADA, la plataforma también cuenta con un aplicación para celular. Se podría considerar como una tarea a futuro vincular la plataforma actual con la aplicación y que el usuario pueda utilizar un dispositivo móvil como interfaz con el sistema.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice A

Ensayos

A.1. Ensayo Comparativo de Sensores de Humedad de Suelo

A.1.1. Introducción

El experimento se enmarca en el proceso de diseño del proyecto de fin de carrera SMC. En dicho proyecto se plantea la monitorización de la humedad de la tierra en macetas de plantas de cannabis, y la posibilidad del control de riego correspondiente.

Para realizar el monitoreo de humedad, se presentan dos opciones de sensores de humedad en tierra: el sensor SEN0193, de tecnología capacitiva, y el FC-28, de tipo resistivo.

A.1.2. Objetivos

Con este experimento se busca comparar el comportamiento de los sensores con el fin de evaluar la mejor opción a instalar.

Se definen 9 escalones de riego (que se corresponden con 9 niveles de humedad de suelo) para los cuales se tomarán mediciones de masa del sistema y de humedad con todos los sensores. Para esto se partirá de un nivel bajo de humedad y se procederá a regar el sistema con un volumen determinado de agua en cada escalón.

A.1.3. Desarrollo

-Suposiciones

- Se utilizarán dos sistemas similares, cada uno dedicado a un modelo de sensor.
- Se asume que se alcanza el nivel de 0 % de humedad relativa en el proceso de secado de tierra.

- Se asume homogeneidad de la humedad del sistema, suponiendo que no hay diferencias de valores por ubicación del sensor.
- Se desprecia el volumen de agua perdido por evaporación.
- Se desprecian los efectos producidos por el ambiente exterior (diferencias de temperatura, humedad ambiente, etc).

-Variables

Variable de entrada: masa de agua vertida, traducida en humedad relativa.
Variable de salida: lectura de sensor.

-Instrumentos

- Recipientes (masa 36 g).
- Tierra proporcionada por el cliente, con las mismas características que la usada en el cultivo.
- Horno doméstico común.
- Sensor de humedad de suelo modelo SEN0193.
- Sensor de humedad de suelo modelo FC-28.
- Balanza electrónica, apreciación 1 g.
- Placa de desarrollo NodeMCU (microcontrolador ESP-8266).
- Protoboard, conexiones.
- Computadora.

-Proceso

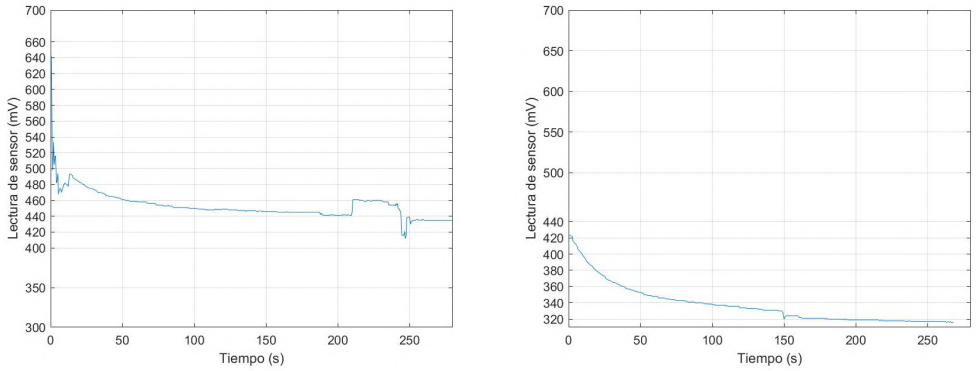
- Secado de tierra
  - Se precalienta el horno a potencia mínima (no se cuenta con termostato).
  - Se coloca una cantidad suficiente de tierra en un recipiente para horno plano.
  - A lo largo del secado se mueve la tierra para favorecer la evaporación homogénea.
  - Para verificar el punto de 0 % de humedad relativa se mide la caída de la masa cada cierto tiempo. Se define el punto de 0 % de humedad cuando se identifica que la masa se mantiene constante en tres mediciones consecutivas.

- Preparación
  - Se miden las masas de los recipientes y los sensores.
  - Luego de aproximadamente 5 horas en horno, se retira la tierra, colocándola en los recipientes.
  - Se verifica una temperatura inicial de aproximadamente 50°C, por lo que se deja reposar durante 1 hora, tapando el recipiente para evitar contaminación del ambiente.
  - Se observa un descenso de temperatura hasta 30°C, permitiendo iniciar las mediciones.
  - Se realizan las conexiones necesarias para la comunicación de cada sensor con las respectivas placas de desarrollo y la computadora.
- Riego y mediciones
  - Primer escalón
    - Se mide la masa, registrando una cantidad de 193 g de tierra seca.
    - Se activa la recepción de medidas del sensor.
    - Se detiene la toma de datos luego de unos minutos.
  - Siguientes escalones
    - Se agrega una cantidad de masa de agua determinada (20-50 g para los primeros 7 escalones).
    - Se mezcla la tierra para forzar la homogeneización de la humedad.
    - Se introduce el sensor correspondiente.
    - Se activa la recepción de datos del sensor.
    - Se detiene la toma de datos luego de unos minutos.

A.1.4. Resultados

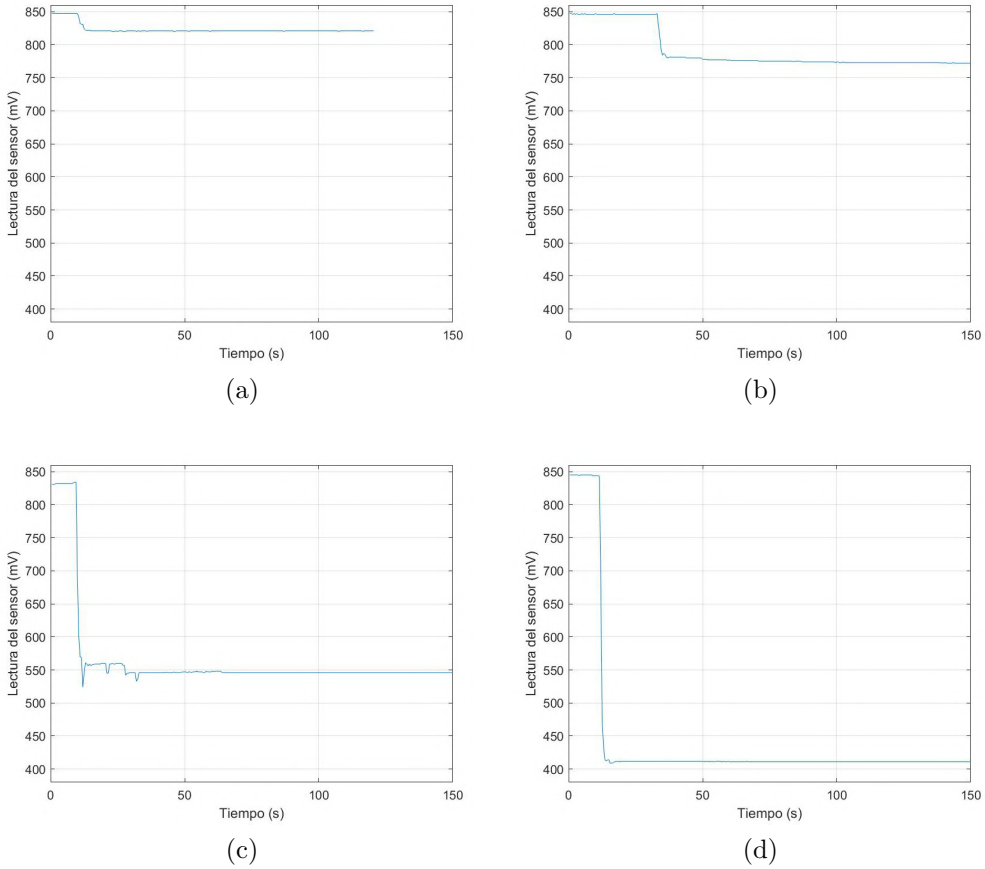
En primer lugar, al trabajar con el sensor FC-28, encontramos que resultaba difícil determinar un valor final de lectura dada la inestabilidad de la medida. Al analizar los datos tomados, se verifica esta situación. En las figuras A.1a y A.1b, se grafica la evolución de la lectura del sensor en el tiempo, para tres escalones de riego distintos. En estas gráficas se observa que la lectura parece no terminar de estabilizarse en un valor, sino que por lo contrario se mantiene con un decrecimiento durante varios minutos.

Este comportamiento del sensor dificulta la lectura de la humedad en tierra, debiendo establecer rangos de incertidumbre más grandes de lo esperado. En este sentido, se observa un contraste con el comportamiento del sensor capacitivo. En las figuras A.2a y A.2b, se muestran gráficas similares a las anteriores. En ellas se observa a simple vista que el sensor SEN0193 fija su valor de lectura en pocos segundos, manteniéndose en el mismo por varios minutos.



(a) (b)

Figura A.1: Evolución en tiempo de lectura de sensor resistivo, ante saltos de humedad.



(a) (b)

(c) (d)

Figura A.2: Evolución en tiempo de lectura de sensor capacitivo, ante saltos de humedad.

A.1.5. Conclusiones

Se completó el experimento de forma satisfactoria, realizándolo dentro del tiempo esperado, cumpliendo las precauciones dispuestas y obteniendo resultados de interés.

En cuanto a la elección de los sensores, si bien a priori no se esperaban grandes diferencias entre una y otra opción, se encuentra en la práctica una característica que marca su disparidad. El comportamiento del sensor capacitivo se aventaja sobre el del sensor resistivo, dado que requiere de un tiempo de establecimiento de la lectura considerablemente menor. Esto concluye en que el sensor elegido sea el modelo SEN0193.

En la figura A.3 se comparan dos de los ejemplos anteriores observando con claridad la dificultad de estabilización del sensor resistivo (en rojo), en contraste con el capacitivo (en azul).

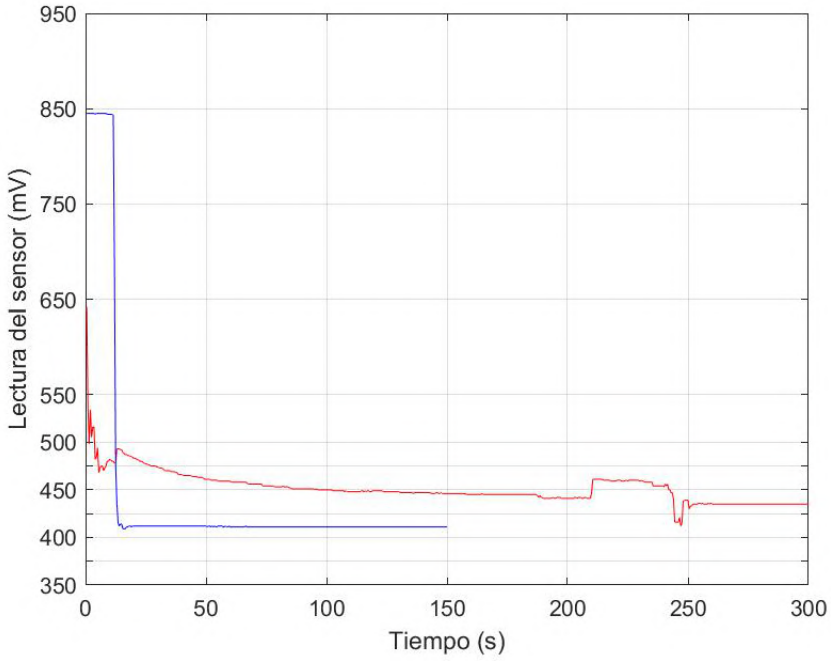


Figura A.3: Evolución en tiempo de lecturas de sensores resistivo (rojo) y capacitivo (azul).

A.2. Calibración de Sensores de Humedad de Suelo Capacitivos

Los sensores de humedad de suelo capacitivos utilizan una capacidad variable para transformar la humedad de su entorno en una señal de tensión continua. El fabricante indica que esta transformación se da de forma lineal, pero no aporta un modelo genérico. Por esta razón, se debe identificar empíricamente la ecuación

que relaciona estas dos magnitudes. Además, durante la investigación al respecto de estos sensores, encontramos en múltiples oportunidades la recomendación de realizar una calibración individual para cada sensor, debido a que sus comportamientos suelen ser diferentes, incluso para un mismo fabricante.

Se propone entonces un procedimiento similar al aplicado en A.1. Se parte de un volumen de tierra seca, es decir con humedad relativa (al volumen de tierra) cercana a 0 %. Luego, se incrementa este valor en saltos, agregando agua de un volumen fijo (saltos constantes), tomando las medidas del sensor luego de un tiempo de establecimiento.

Para obtener un modelo preciso, se define que los saltos sean del 5 % de humedad relativa, entre el 5 % y el 75 % (rango que supera ampliamente el esperado en la aplicación), completando así 15 puntos de medida.

Finalmente, se procesan los datos en Matlab y se construyen y grafican los modelos de los sensores que se observan en las figuras A.4a a A.4e.

Como primer resultado, pudimos confirmar la diferencia de comportamiento entre distintos sensores, encontrando no solo *offsets* diferentes, sino también pendientes del modelo lineal distintas.

Se observa también una variabilidad en la medida, donde algunos puntos se alejan del rango de incertidumbre. De todas formas, podemos afirmar que cuantitivamente no hay un gran apartamiento del modelo, siendo que los coeficientes de determinación son cercanos a 1.

De esta forma, obtenemos un modelo del tipo $H = a \times V + b$ para cada sensor, con los coeficientes de la tabla A.1. Además, podemos observar gráficamente los diferentes modelos en la figura A.5.

Sensor	Coeficiente a	Coeficiente b
1	-0,3669	277,05
2	-0,2505	315,54
3	-0,3055	260,66
4	-0,2282	270,26
5	-0,3720	296,15

Tabla A.1: Coeficientes del modelo lineal de cada sensor de humedad de suelo.

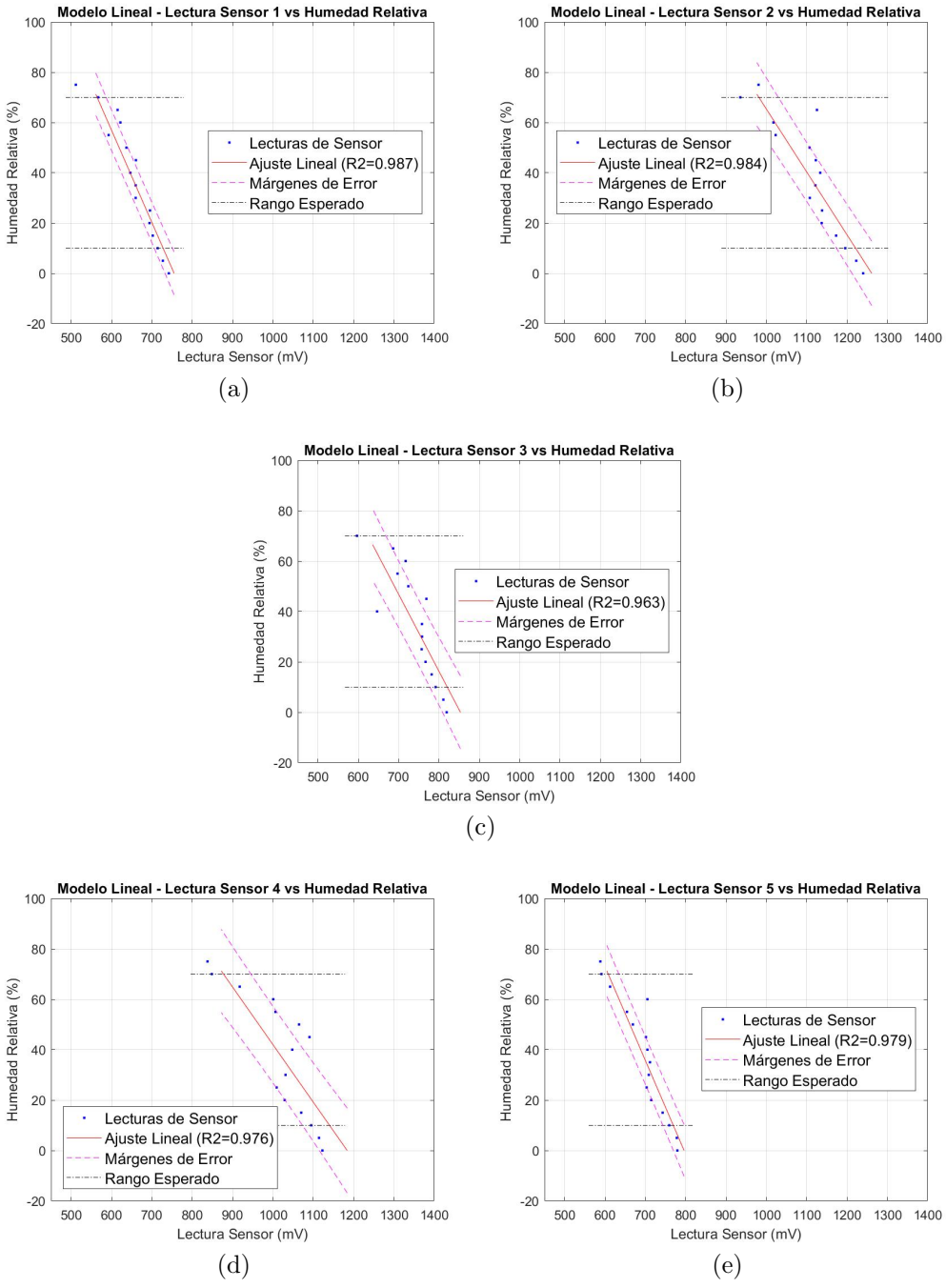


Figura A.4: Modelos lineales de sensores de humedad de suelo.

A.3. Calibración de Sonda de pH

Con la calibración de la sonda de pH se busca elaborar un modelo que permita transformar de forma simple las lecturas analógicas del sensor en un valor de pH.

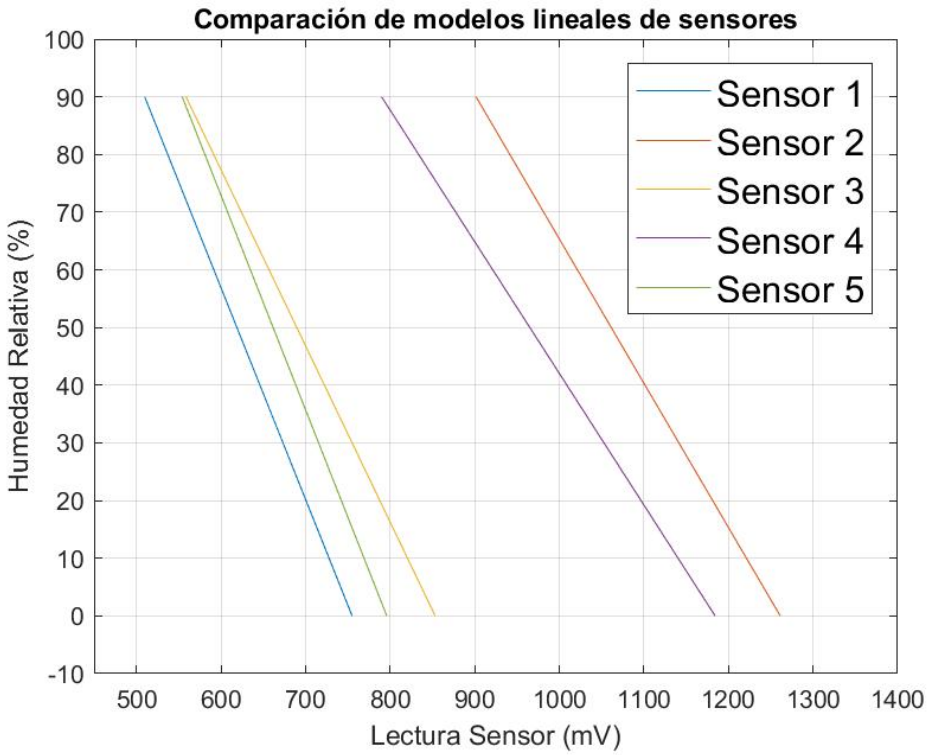


Figura A.5: Comparación de Modelos Lineales de Sensores.

Se llevó a cabo un proceso estándar de calibración de sonda, que implica la medición de líquidos de pH conocido, denominados líquidos *buffer*. Se utilizan *buffer* de pH 4,0 y 7,0, destinados a construir un modelo lineal a partir de dos puntos de calibración.

A lo largo del proceso se utiliza agua destilada y papel para limpiar la sonda e independizar las medidas al cambiar de líquido.

Se encuentra entonces que la sonda arroja los valores de la tabla A.2 al medir los líquidos buffer.

Líquido Buffer (pH)	Lectura de Sonda (mV)
4,0	1299,5
7,0	936,2

Tabla A.2: Medida al utilizar buffer

Con este par de puntos de operación, se puede generar la recta que permita extrapolar las lecturas a todos los valores de pH. Para ello se utiliza la ecuación A.1.

$$pH = a \times Lectura + b \tag{A.1}$$

$$a = \frac{Buffer\ 7\ -\ Buffer\ 4}{Lectura\ 7\ -\ Lectura\ 4} = -0,0083 \tag{A.2}$$

$$b = Buffer\ 7\ -\ a \times Lectura\ 7 = Buffer\ 4\ -\ a \times Lectura\ 4 = 14,7308 \tag{A.3}$$

Con los parámetros del modelo definidos, podemos representar gráficamente el resultado, como se ve en la figura A.6.

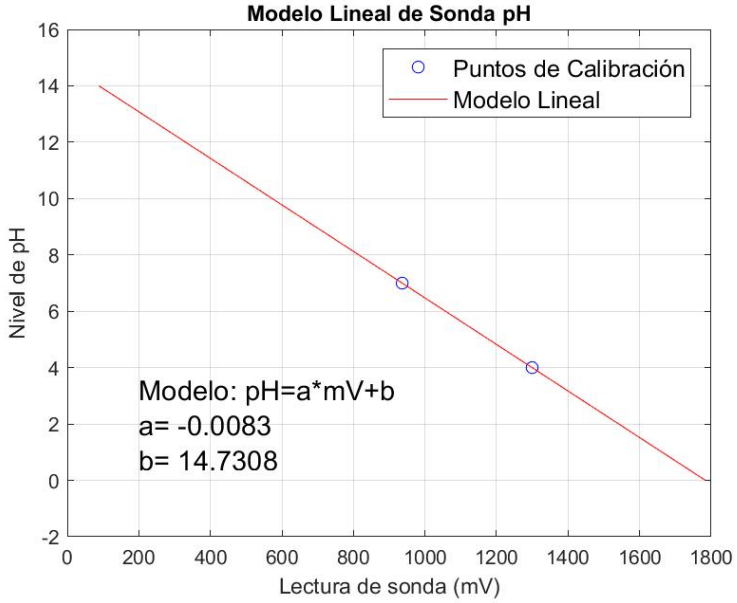


Figura A.6: Modelo lineal para sonda de pH.

A.4. Parametrización de Ajuste de pH

El pH es una medida que representa la concentración de iones de hidrógeno presentes. Para obtener un modelo confiable de la reaccion química al mezclar agua y ácido fosfórico, se elabora un ensayo de respuestas al escalón, estudiando así el cambio en el nivel de pH ante inyecciones de ácido.

El ajuste de pH para el agua de riego requiere tanto de una lectura confiable del pH inicial, como de un correcto cálculo para alcanzar el valor final deseado. Se realiza un ensayo de parametrización del ajuste a fin de elaborar el modelo matemático que transforme diferencias de pH en volumen de ácido a inyectar.

Se toma una muestra de 10 L de agua de la red de suministro para utilizar como prototipo. Para reducir el pH se utilizará ácido fosfórico, siendo esta la sustancia definida por el cliente.

El prototipo, como se ve en la figura A.7, consiste en un balde con el agua recién tomada del suministro, la sonda de pH con su flotador de poliestireno, un agitador manual y una jeringa para dosificar el ácido fosfórico.

Figura A.7: Prototipo utilizado para la parametrización.

El procedimiento consistió en la medición continua del pH del líquido introducido en el balde y la inyección de escalones de ácido fosfórico de volumen 0,5 mL. Luego de cada inyección prosigue un tiempo de agitación de 2 minutos para la homogeneización de la solución.

Observando las respuestas a escalones de la figura A.8, vemos que la misma es considerablemente rápida, estableciéndose al nuevo valor en apenas algunos segundos. Si bien se debe tener en cuenta que el volumen del prototipo es reducido (aproximadamente una décima parte del tanque de la planta de riego), este resultado permite dar cuenta del orden de tiempo de recirculación necesario entre mediciones.

Con los datos recabados se procede a elaborar un modelo que represente el cambio de pH ante distintos valores de volumen de ácido dosificado. Teniendo en cuenta el comportamiento químico de los líquidos involucrados, se espera que en el rango de trabajo el modelo responda de forma similar a un polinomio de tercer

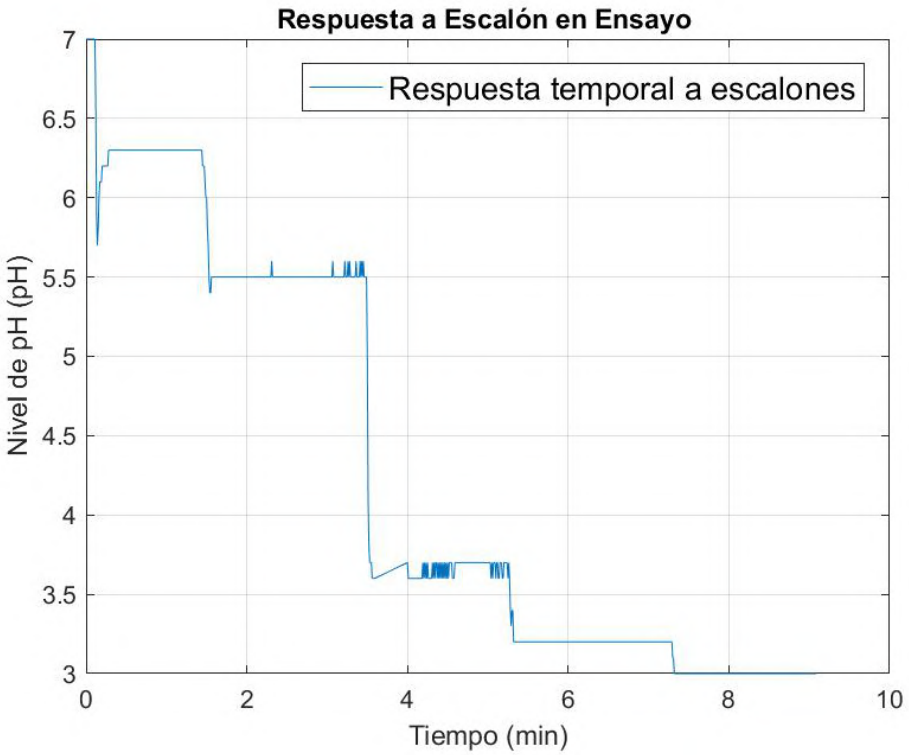


Figura A.8: Evolución temporal del pH durante el ensayo.

grado. De todas formas, se puede simplificar el modelo teniendo en cuenta que en el rango de trabajo (entre 5,5 y 7,5 pH), el comportamiento es aproximadamente lineal.

Se encuentra entonces el modelo de la figura A.9.

A.5. Calibración de Caudalímetros

Si bien los sensores de caudal cuentan con su manual de usuario donde se informa el valor de pulsos por volumen de agua que los caracteriza, se decide comprobar o corregir este valor de forma empírica. En líneas generales, se busca hacer circular un volumen de agua por ambos caudalimetros, verificando la cantidad de pulsos registrados.

Debido a que se tiene pendiente parte de la instalación del sistema en el establecimiento del cliente, esta calibración aun no fue completada. De todas formas, se detalla el plan a seguir.

El circuito hidráulico a considerar consiste en la conexión de la bomba de riego al block de riego diseñado, en serie con ambos caudalímetros y cerrando todas las electroválvulas y aberturas posibles, de forma de dejar como única salida la correpsondiente a la recirculación. A su vez, se coloca un recipiente en la salida de

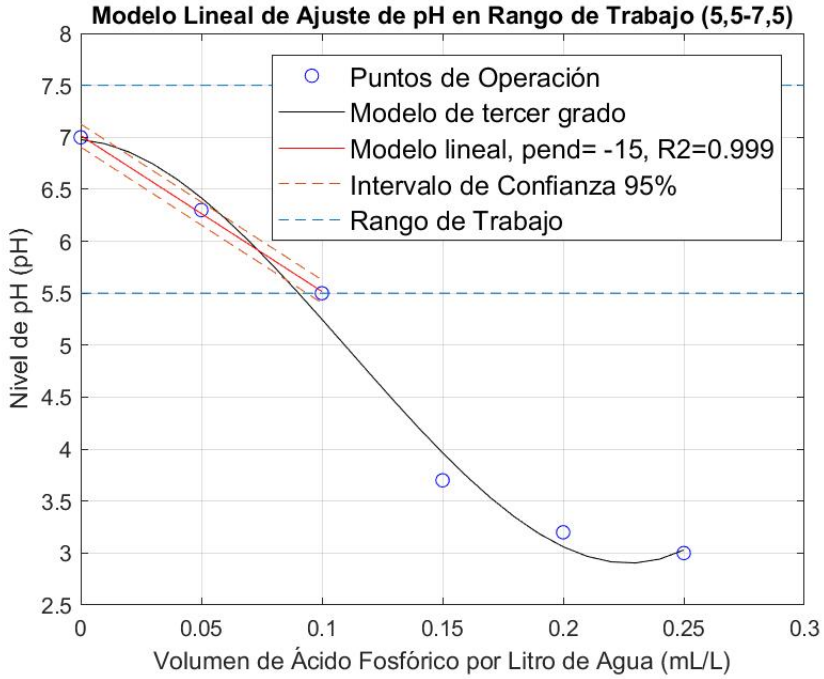


Figura A.9: Modelo de variación de pH por dosificación de ácido fosfórico.

recirculación, con el que se medirá el volumen de agua utilizada.

En primer lugar, se toma cuenta del peso del recipiente, de forma de medir el volumen de agua mediante una balanza.

Se plantean distintos puntos de operación para identificar y descartar sesgos y offsets. Los puntos de operación se definen en tiempo de encendido de la bomba.

El procedimiento entonces consiste en el inicio del registro de datos de ambos caudalímetros, el encendido de la bomba de riego por el tiempo determinado y finalmente el pesaje del recipiente con el agua vertida.

De esta forma, se espera llegar a elaborar un modelo lineal cuya pendiente corresponda al indicador de pulsos por litro de agua.

A.6. Calibración de Sensor de CO2

El sensor de CO2 utilizado tiene un comportamiento lineal, imponiendo en su salida analógica un valor de tensión que se corresponde con la concentración de CO2 de acuerdo a una recta.

El sensor cuenta con la capacidad de autocalibrarse, permitiendo ajustar la recta según las condiciones que detecte. Este proceso se realiza cada vez que se enciende y periódicamente cada 24 horas.

El rango de medición es de 0 a 5000 ppm de CO2. Según el fabricante, la

salida del sensor es de 0,4 a 2 V y cubre completamente el rango mencionado. De esta forma, se caracteriza el sensor con el método de dos puntos, donde 0,4 V corresponden a 0 ppm y 2 V a 5000 ppm.

Finalmente, se determina el modelo graficado en la figura A.10.

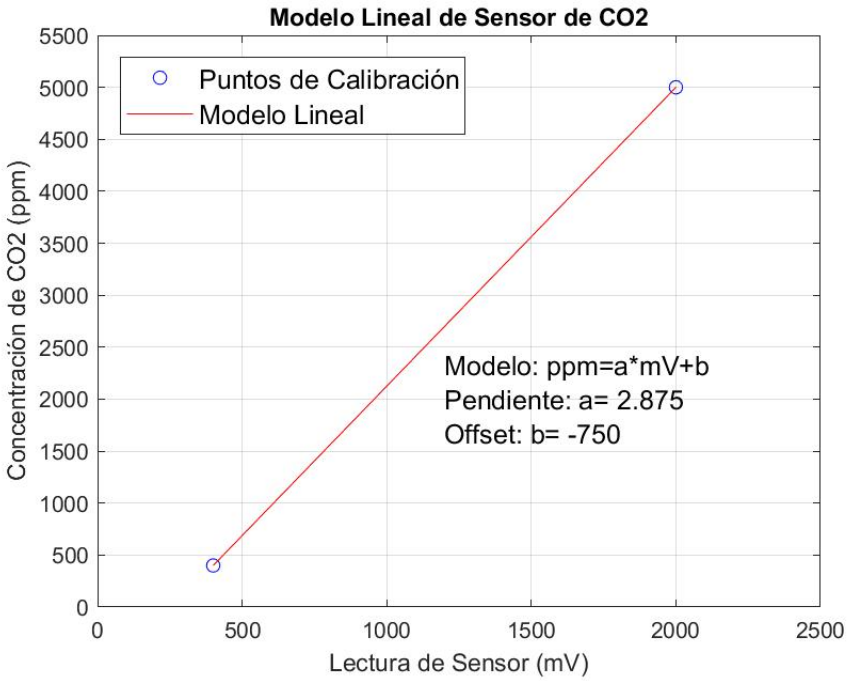


Figura A.10: Modelo lineal del sensor de CO2.

A.7. Parametrización de Bomba Peristáltica

Se espera que las dosificaciones de ácido fosfórico para el ajuste de pH sean de poco volumen, en el orden de mL. Por eso se elige una bomba peristáltica como actuador. A su vez, se debe tener claro el funcionamiento de la bomba, por lo que se parametriza el caudal de la misma.

Se realiza un ensayo de parametrización consistente en la activación de la bomba durante distintas ventanas temporales, midiendo el líquido expulsado.

Se conectan a la bomba una probeta medidora y una jeringa a modo de reservorio, resultando en la configuración de la figura A.11.

La bomba es alimentada (5 V), controlada y monitoreada por una placa UCD.

El procedimiento planteado consiste de 4 puntos temporales de calibración: 5, 10, 30 y 60 segundos, en los que se toman 5 medidas cada uno.

Una vez completadas las medidas, se realiza un análisis estadístico, obteniendo media y desviación estándar de cada punto. En base a estos datos se genera un

Figura A.11: Configuración para ensayo de parametrización de bomba peristáltica.

modelo, observando que se ajustan de buena manera a una aproximación lineal, ilustrada en la figura A.12.

Se concluye entonces que la bomba peristáltica es capaz de dosificar 1,18 mL/s, con un offset inicial de 0,57 mL.

A.8. Parametrización de Caudal de Riego

Para el control del riego es necesario conocer el volumen de agua por segundo que el sistema es capaz de entregar. Para identificar este valor es necesario realizar pruebas con la planta de agua instalada en su totalidad, dado que tanto la bomba como el circuito hidráulico y los demás componentes, influyen en el resultado final. Siendo que la planta de agua aun no está completamente instalada, queda pendiente esta parametrización, a realizarse en cuanto la planta de agua se complete.

De forma similar a la calibración de los caudalímetros, el ensayo consiste en el manejo de las electroválvulas en distintas ventanas de tiempo y la medición del agua vertida.

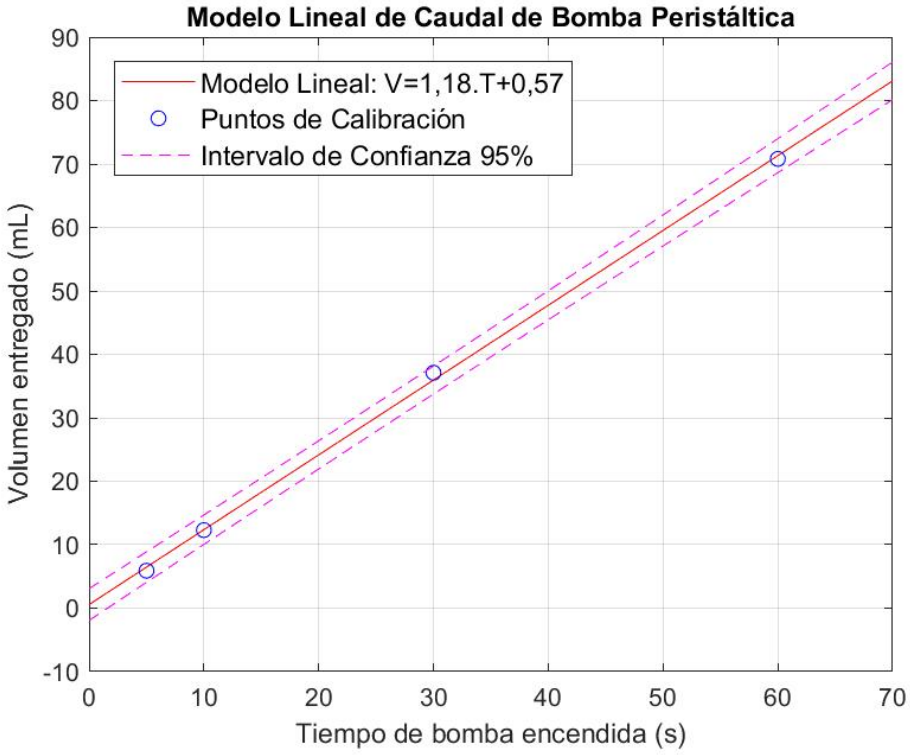


Figura A.12: Modelo lineal de bomba peristáltica.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice B

Manual Técnico

Este manual tiene como objetivo dar información técnica sobre el sistema de monitoreo y control planteado en el proyecto de fin de carrera SMC. Se explica cómo armar la Unidad de Control (UC), con las configuraciones pertinentes para lograr acceso remoto a esta unidad y a la interfaz del SCADA. También se detalla de forma genérica cómo implementar un Nodo Maceta (NM) y una Unidad de Control Distribuido (UCD). Por último, se presentan instrucciones de forma genérica para expandir el sistema y soluciones a posibles problemas que puedan surgir.

El objetivo de armar un entorno de desarrollo es poder contar con los programas para programar las placas, desarrollar pantallas del SCADA y poder realizar pruebas de forma local

B.1. Entorno de desarrollo

Es importante contar con un entorno de desarrollo para poder contar con las herramientas para elaborar y cargar los programas a los NM y UCD, desarrollar pantallas del SCADA y poder realizar pruebas de forma local. Se podría utilizar la UC como entorno de desarrollo, aunque esto no resulta práctico y es poco recomendable una vez se puso en marcha el sistema. En esta sección se presentan recomendaciones de cómo armar este entorno.

Se busca armar un entorno de desarrollo que permita:

- Cargar programas desarrollados en Contiki-NG a los NM y UCD.
- Desarrollar pantallas y configuraciones del SCADA.
- Probar el sistema.

B.1.1. Hardware

Para el entorno de desarrollo se necesita una computadora con la capacidad de instalar un sistema operativo con entorno gráfico. Los programas a instalar presentan muy pocos requisitos de hardware. Thingsboard es el software con mayor demanda, requiere 1 GB de memoria RAM.

Se recomienda fuertemente utilizar una laptop, ya que al ser portable permite al usuario desplazarse hacia donde se encuentran los NM y UCD.

Se recomienda a su vez, trabajar en una máquina virtual y utilizarla solamente para este propósito. Esto proporciona un entorno fácilmente exportable a otro hardware, y a su vez este entorno contendría solamente lo necesario para su uso, evitando otro problemas que puedan surgir a partir de configuraciones u otros programas para otras aplicaciones.

B.1.2. Software

El entorno de desarrollo utilizado en el proyecto SMC, se armó sobre el sistema operativo Ubuntu 20.04 LTS.

Paquetes de herramientas Contiki-NG

Contiki-NG cuenta con un contenedor en Docker, el cual ya cuenta con todas las herramientas necesarias para compilar y programar placas. Sin embargo, se recomienda instalar todo en la raíz, al trabajar con el contenedor a veces el puerto serial que se abre para *tunslip* queda abierto y se requiere cerrar y volver a abrir el contenedor.

Antes de descargar el repositorio de Contiki-NG, conviene instalar las herramientas recomendadas por los desarrolladores. Contiki-NG cuenta con un tutorial para esto y para descargar el repositorio [16].

En la siguiente lista se detallan los puntos necesarios para este entorno de desarrollo:

- *Install development tools for Contiki-NG:* Además de contener programas necesarios para compilar Contiki-NG, contiene Wireshark, que puede ser utilizado para probar el sistema. Puede ser necesario cambiar la librería python-serial por python3-serial.
- *Install ARM compiler:* Necesario para compilar el código.
- *Install a CoAP client:* Útil para pruebas.
- *User access to USB:* Necesario para programar.

El repositorio de Contiki-NG está incluido en el repositorio del proyecto SMC, por lo que alcanza con descargar este último.

Thingsboard

La plataforma presenta instrucciones detalladas paso a paso de como realizar la instalación [74], por lo que se recomienda seguir esta guía. En nuestro entorno de desarrollo se instaló la opción con PostgreSQL.

Thingsboard también presenta la posibilidad de utilizarse desde un contenedor de Docker, esto no se recomienda, ya que es posible que se den problemas de comunicación entre la plataforma y la red de controladores [71].

Python

Para la ejecución de los *scripts* desarrollados en Python, es necesario contar con las librerías *aiocoap* y *tinydb*.

Desde la terminal de Linux se puede instalar módulo *tinydb* utilizando los siguientes comandos:

```
pip install tinydb
```

La instalación de la librería *aiocoap*, se puede realizar ejecutando los siguientes comandos:

```
git clone https://github.com/chrysn/aiocoap
cd aiocoap && ./setup.py install
```

Repositorio del proyecto SMC

Es necesario descargar el repositorio del proyecto SMC ya que contiene los códigos de programación necesarios para la UC, NM y UCD. Esto se logra ejecutando los siguientes comandos desde la terminal de linux:

```
git clone https://gitlab.fing.edu.uy/carlos.menoni/pfc-smc.git
git submodule update --init --recursive
cd Codigos/Python
chmod +x dockerrun.sh
```

B.2. Unidad Central

Esta sección busca explicar como armar la Unidad Central (UC). Esta contiene al SCADA, se encarga de recibir la información de monitoreo y enviar comandos a las UCD. Este equipo también cuenta con acceso remoto.

B.2.1. Hardware

Esta conformado por una computadora Raspberry Pi 4 Model B y un Launchpad CC2650 (Border Router). Para armar esta unidad solamente se necesita conectar el Border Router a cualquier puerto USB de la Raspberry. En la figura B.1, se muestra una foto de este equipo.

Figura B.1: Unidad Central

B.2.2. Sistema Operativo

El sistema operativo utilizado en la Raspberry es Raspberry Pi OS, un SO Linux basado en Debian. Este sistema está instalado en una memoria SD de 64 GB.

Raspberry Pi Foundation explica cómo instalar el sistema operativo en una memoria SD y dejarlo *booteable* para que al iniciar el equipo, inicialice el sistema operativo [48]. Siguiendo este tutorial ya se puede dejar habilitado el acceso por ssh. Este sistema operativo ya contiene instalado vnc.

B.2.3. SCADA

Thingsboard

El software utilizado como SCADA es Thingsboard. Se debe instalar la versión sugerida para este sistema operativo [75].

Contiki-NG

Se debe instalar las herramientas de Contiki-NG y descargar su repositorio, de este modo se logra abrir el tunslip (tunel serial IP). Se requiere de esto para lograr la comunicación con el Border Router.

En este caso, solamente se requiere instalar los siguientes puntos indicados en el tutorial de Contiki-NG [16].

- *Install development tools for Contiki-NG:* Puede ser necesario cambiar la librería python-serial por python3-serial.
- *User access to USB*
- *Clone Contiki-NG*

Python

Se deberá instalar las librerías de Python detalladas en la sección B.1.2, se puede seguir las instrucciones detalladas en esa sección.

Automatización del inicio

Se quiere que en caso de corte de alimentación del equipo, este levante los programas necesarios y se restaure de forma automática. Para lograr este cometido se debe editar el archivo *.bashrc* en el directorio *home* del sistema operativo. Es necesario agregar las siguientes líneas de código:

```
echo "Running at boot"
sudo /usr/local/bin/noip2
sudo sh /home/pi/pfc/dockerrun.sh
```

Instalación y Configuración de No-IP

Dado que las direcciones IP del Router de Antel es dinámica, es necesario implementar un mecanismo que le permita al usuario desde cualquier parte del mundo conocer la dirección IP del Router, para que de este modo se pueda acceder a la página web local de Thingsboard. Para ello se utiliza el software No-IP. Este programa toma la dirección IP dinámica y la registra en un servidor DNS de

tipo A. Primero es necesario crearse una cuenta en su página web [42].

Durante el registro de la cuenta se presenta la opción de crear un nombre de dominio. Una vez finalizado se debe instalar el programa No-IP. Esto se puede lograr con los siguientes comandos desde la línea de código de Linux:

```
wget https://www.noip.com/client/linux/noip-duc-linux.tar.gz
tar vzxf noip-duc-linux.tar.gz
cd noip-2.1.9-1
sudo make
sudo make install
```

Al finalizar la instalación, se le pedirá al usuario las credenciales de su cuenta de No-IP y finalmente se le pedirá que ingrese el período de actualización de la dirección IP del registro A de No-I. Para el proyecto SMC se configuró 5 minutos.

IP Raspberry Pi en red local

Es necesario configurar la IP de la Raspberry para que sae una IP fija y que no le solicite la IP al Router. Para ello se debe editar el archivo *dhcpcd.conf*, ubicado en el directorio *etc*. Al final del archivo se debe agregar:

```
interface wlan0
static ip_address=192.168.1.151
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Donde de arriba hacia abajo representan, la interfaz a la cual se desea fijar la dirección IP local, la dirección IP local que se desea fijar, la dirección IP local del Router y la dirección IP local del servidor DNS que en este caso coincide con la del Router puesto que el mismo dispositivo cumple con ambas funciones.

B.2.4. Border Router

El Border Router es un Lauchpad CC2650 de Texas Instrument, este ejecuta un código desarrollado por Contiki-NG, el cual se encuentra en la dirección *Codigos/Contiki-NG/example/rpl-border-router* en el repositorio del proyecto SMC [52].

Se recomienda programar el Border Router con el entorno de desarrollo. Se debe contar con el repositorio del proyecto SMC, ya que se configuró para que compile con TSCH stack en capa de enlace. Además, se agregaron algunas líneas de código a tsch.c para resolver un problema de compatibilidad [15] [53].

Programar Border Router

El Launchpad CC2650 entra en MODO BSL, manteniendo pulsado el botón 1 y luego el botón de *reset*, como se indica en la figura B.2. Es posible que la programa-

ción por *bootloader* este bloqueado, en este caso, se puede desbloquear siguiendo los pasos indicados en la sección B.7.2 (sin conectar otra placa, ni retirar los *jumpers*).

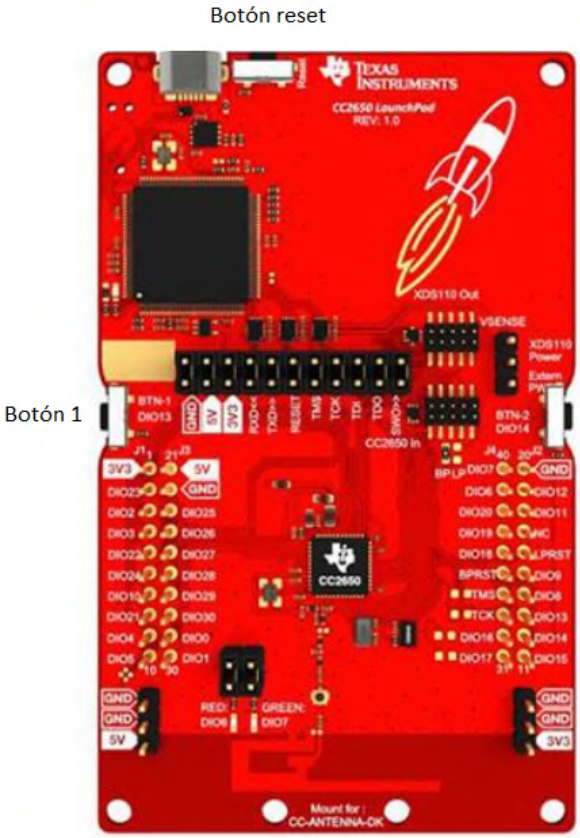


Figura B.2: Launchpad cc2650

Para cargar el programa al Border Router, se debe abrir la terminal desde la carpeta *Codigos/Contiki-NG/example/rpl-border-router* y utilizar los siguientes comandos:

```
make archivo.upload TARGET=cc26x0-cc13x0
BOARD=launchpad/cc2650 PORT=/dev/ttyACM0
```

B.2.5. Router Internet

Thingsboard publica una página web con la IP local y el puerto 8080. Para poder acceder a la misma desde afuera de la red local, es necesario configurar una ruta. En el marco del proyecto SMC se utilizó el Router ZTE F680 de Antel.

Al acceder a la página web local del Router se pueden ver dos campos para ingresar un usuario y una contraseña. Se debe ingresar con las credenciales correspondientes.

En la página principal, en la parte superior existen varias pestañas según la configuración que se desee realizar. Se debe navegar a la pestaña que dice Internet, como se indica en la B.3.

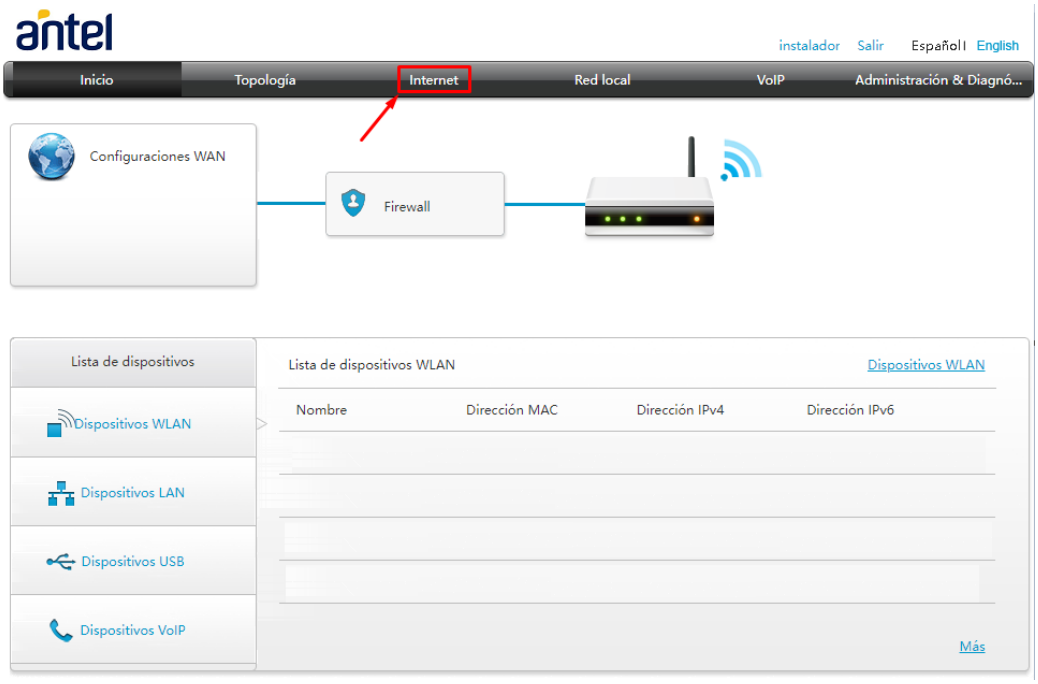


Figura B.3: Página principal del Router ZTE F680

En la pestaña Internet, del lado izquierdo se ve un menú en forma de lista, se debe presionar sobre la opción Seguridad, como se indica en la B.4.

En esta pestaña se debe presionar donde dice Crear Nuevo para agregar una ruta, ver figura B.5.

Finalmente se debe rellenar los campos como se muestra en la figura B.6 y presionar el botón de aplicar cambios.

Para el acceso remoto, se deberá repetir estos pasos y configurar las siguientes rutas (ver figura B.7 y B.8).

Figura B.4: Página Internet del Router ZTE F680

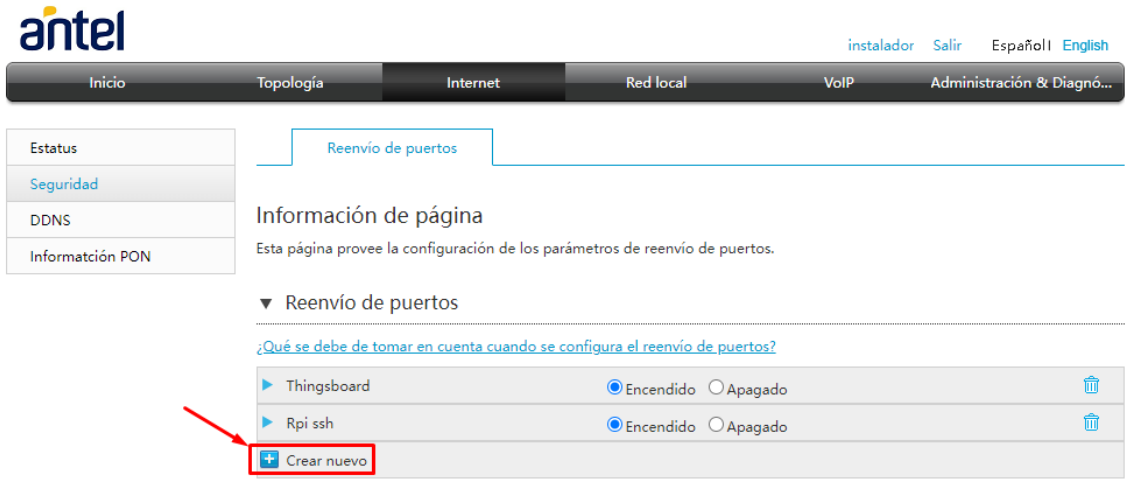


Figura B.5: Página Seguridad del Router ZTE F680

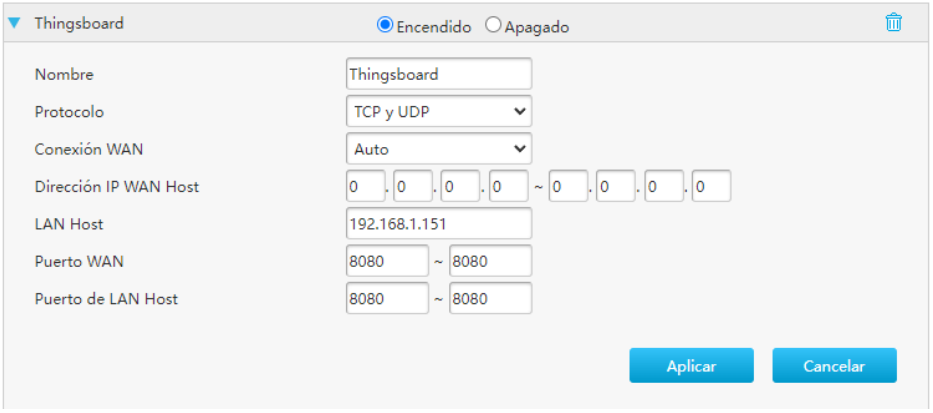


Figura B.6: Configuración de ruta para Thingsboard

Figura B.7: Configuración de ruta para SSH

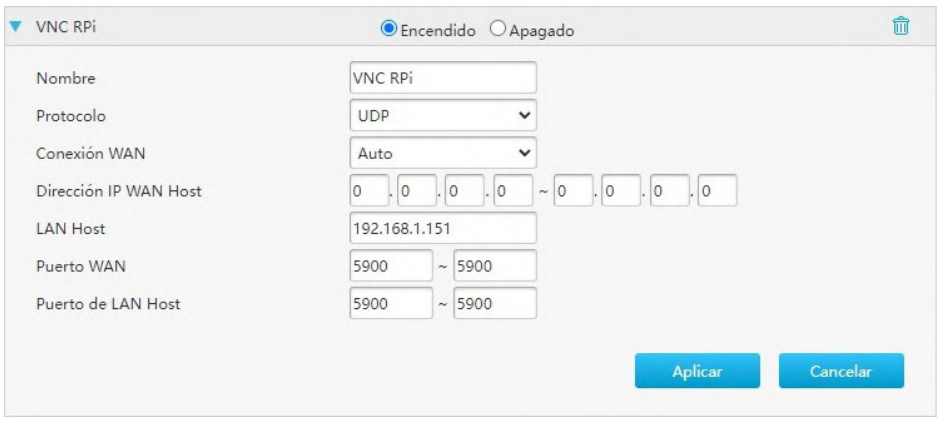


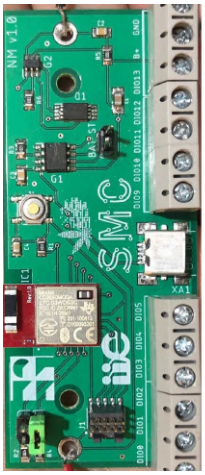
Figura B.8: Configuración de ruta para VNC

B.3. Nodo Maceta

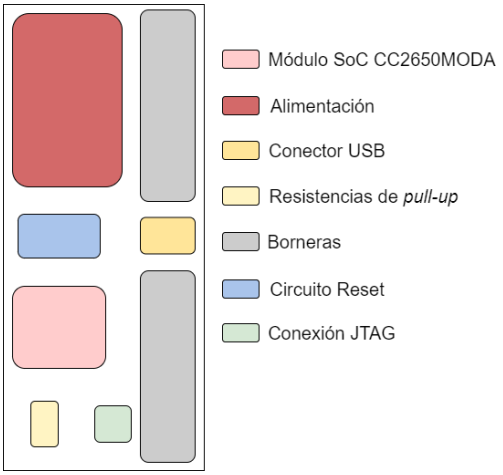
El nodo maceta es una placa basada en el módulo CC2650MODA de Texas Instrument. Este nodo tiene el propósito de ser utilizado para monitoreo, cuenta con un sensor de humedad de suelo y un sensor de temperatura de suelo. Esta placa esta diseñada para alimentarse por una batería de litio de 4.2V.

B.3.1. Hardware

En la figura B.9a, se presenta la placa la cual conforma el Nodo Maceta. La figura B.11 ilustra las pistas de esta placa.



(a) Placa NM.



(b) Bloques funcionales NM

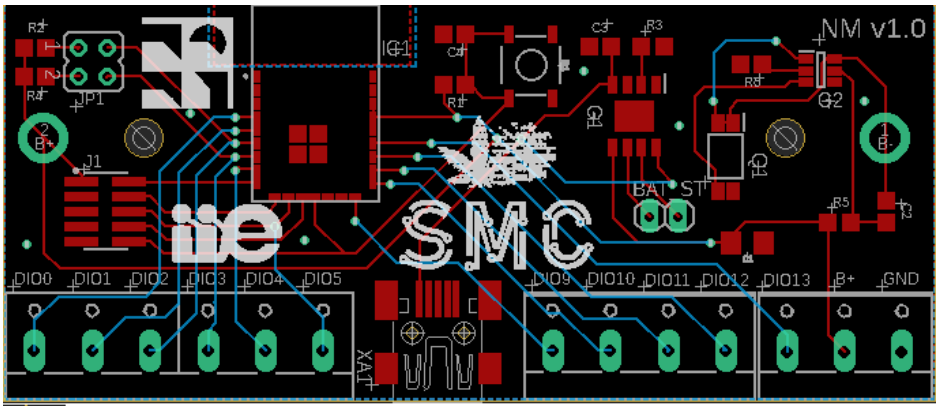


Figura B.9: Ruteo NM

Esta placa cuenta con 2 *jumpers* (JP1.1 y JP1.2) los cuales al cortocircuitarlos activan una resistencia de *pull-up* para las entradas DIO2 y DIO3. Además cuenta

con un conector JTAG y un botón de *reset*. El conector USB se utiliza solamente para el circuito de alimentación de la batería. En la figura B.9b se indican los bloques funcionales que componen esta placa.

La batería que se utilice no puede superar los 4.2V. Los pines BAT y ST se utilizan para monitorear el estado del circuito de alimentación de la batería. La DIO14 del módulo esta conectada a B+, de modo de monitorear la tensión de alimentación del módulo. Los planos eléctricos de la placa se encuentran en el repositorio del proyecto SMC [52].

En la siguiente lista se detallan como se pueden utilizar las entradas de la placa:

- Periférico UART: DIO0 y DIO1.
- Utilizar protocolos que requieran resistencias de *pull-up*: DIO2 y DIO3.
- Entradas y salidas digitales: DIO0 a DIO5, DIO9 a DIO13.
- Entradas analógica: DIO9 a DIO13.
- Alimentación para circuitos externos: B+ (positivo de la batería). GND (Tierra)

.

B.3.2. Software

Para programar la placa se puede seguir las instrucciones detalladas en la sección B.6.2.

Estos equipos llevan el programa que se encuentra en el carpeta *Codigos/Contiki-NG/example/SMC/NM*. El archivo *nm.h*, define el período entre mensaje de telemetría y la dirección a la cual enviar la telemetría.

B.4. Unidad de control distribuido

Al igual que el NM, esta placa esta diseñada para utilizar el módulo CC2650MODA de Texas Instrument. Se cuenta con 4 relés disponibles.

B.4.1. Hardware

En la figura B.10, se presenta la placa la cual conforma la UCD. La figura B.12 ilustra las pistas de esta placa.

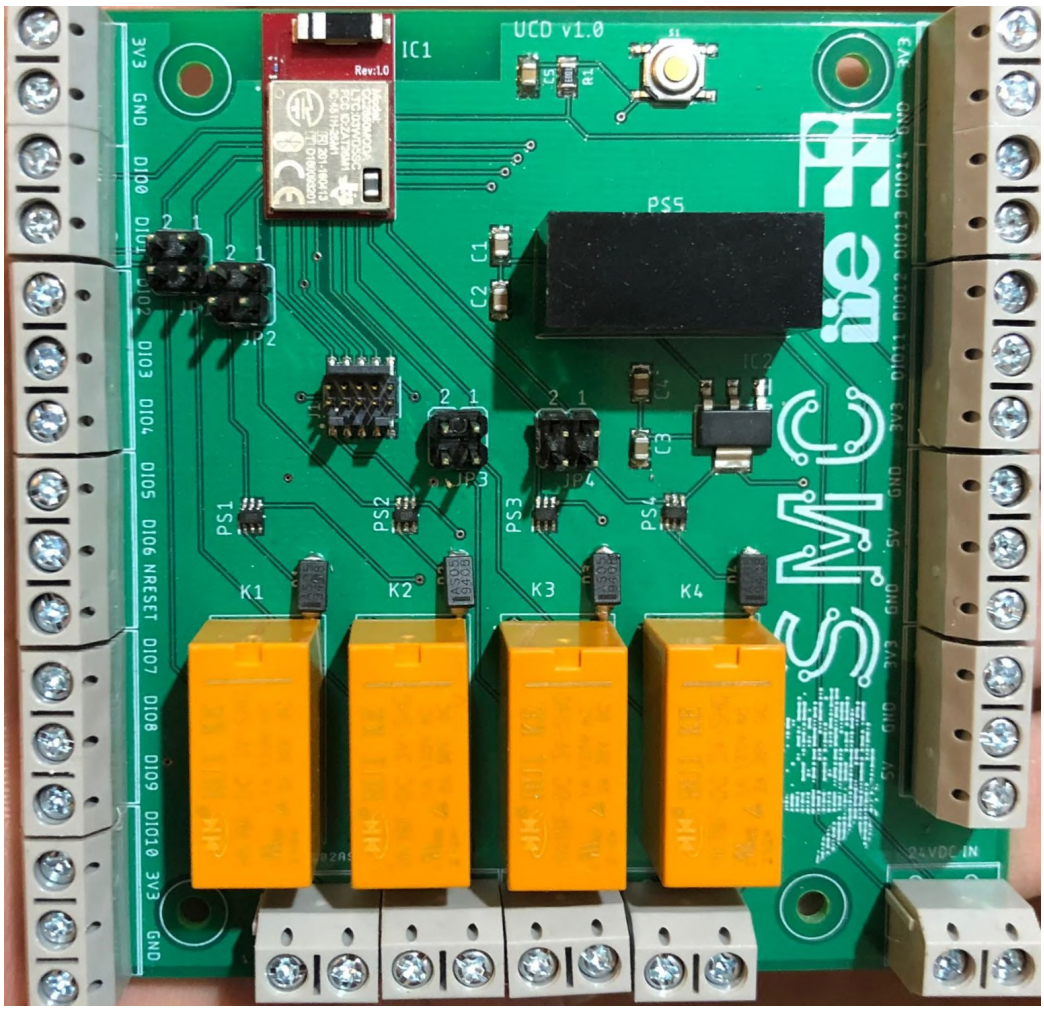


Figura B.10: Placa UCD

Esta placa cuenta con 8 *jumpers* los cuales al cortocircuitarlos conectan las entradas y salidas del módulo CC2650MODA con los relés. Además cuenta con un conector JTAG y un botón de *reset*. En la figura B.9b se indican los bloques funcionales que componen esta placa.

Los planos eléctricos de la placa se encuentran en el repositorio del proyecto SMC.

En la siguiente lista se detallan como se pueden utilizar las entradas de la placa:

- Periférico UART: DIO0 y DIO1.
- Entradas y salidas digitales: DIO0 a DIO14.
- Entradas analógica: DIO7 a DIO14.

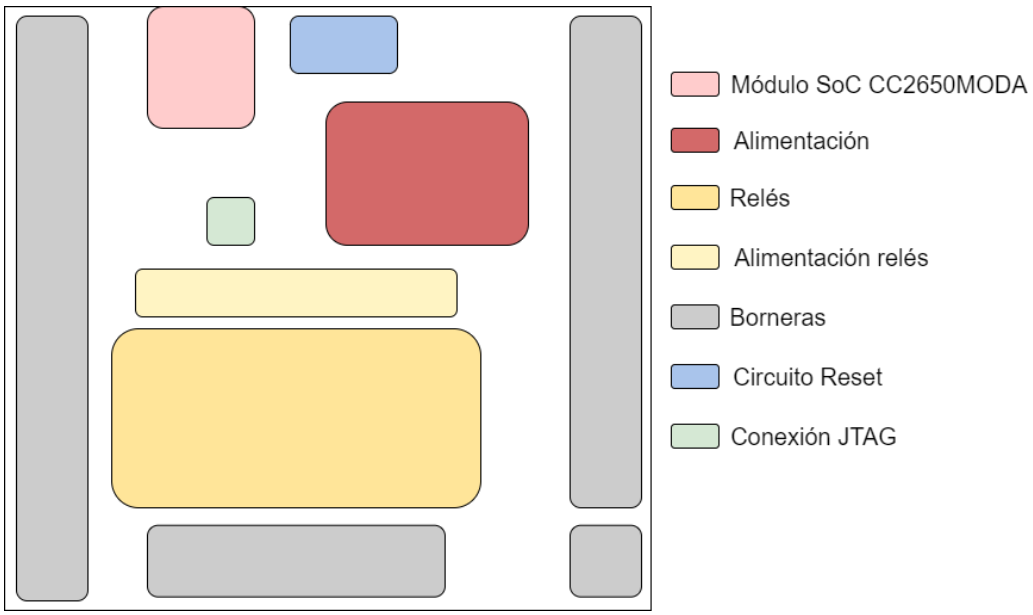


Figura B.11: Bloques funcionales UCD

- Alimentación placa: 24 VDC.
- Alimentación para circuitos externos: 5V, 3V3 y GND.
- Señal de *reset*: NRESET
- Contactos de normales abiertos relé: RELE 1 NO a RELE 4 NO.

.

Configuración de *jumpers*

Esta placa cuenta con 8 *jumpers* agrupados de a 2, en la siguiente lista se indican el rol de cada uno:

- JP1.1: Conecta realimentación relé 1 a entrada DIO1.
- JP1.2: Conecta realimentación relé 2 a entrada DIO2.
- JP2.1: Conecta salida DIO3 con alimentación relé 1.
- JP2.2: Conecta salida DIO4 con alimentación relé 2.
- JP3.1: Conecta realimentación relé 3 a entrada DIO7.
- JP3.2: Conecta realimentación relé 4 a entrada DIO8.
- JP4.1: Conecta salida DIO9 con alimentación relé 3.
- JP4.2: Conecta salida DIO10 con alimentación relé 4.

En caso de utilizar los *jumpers*, no se deberá utilizar las entradas y salidas para otro propósito.

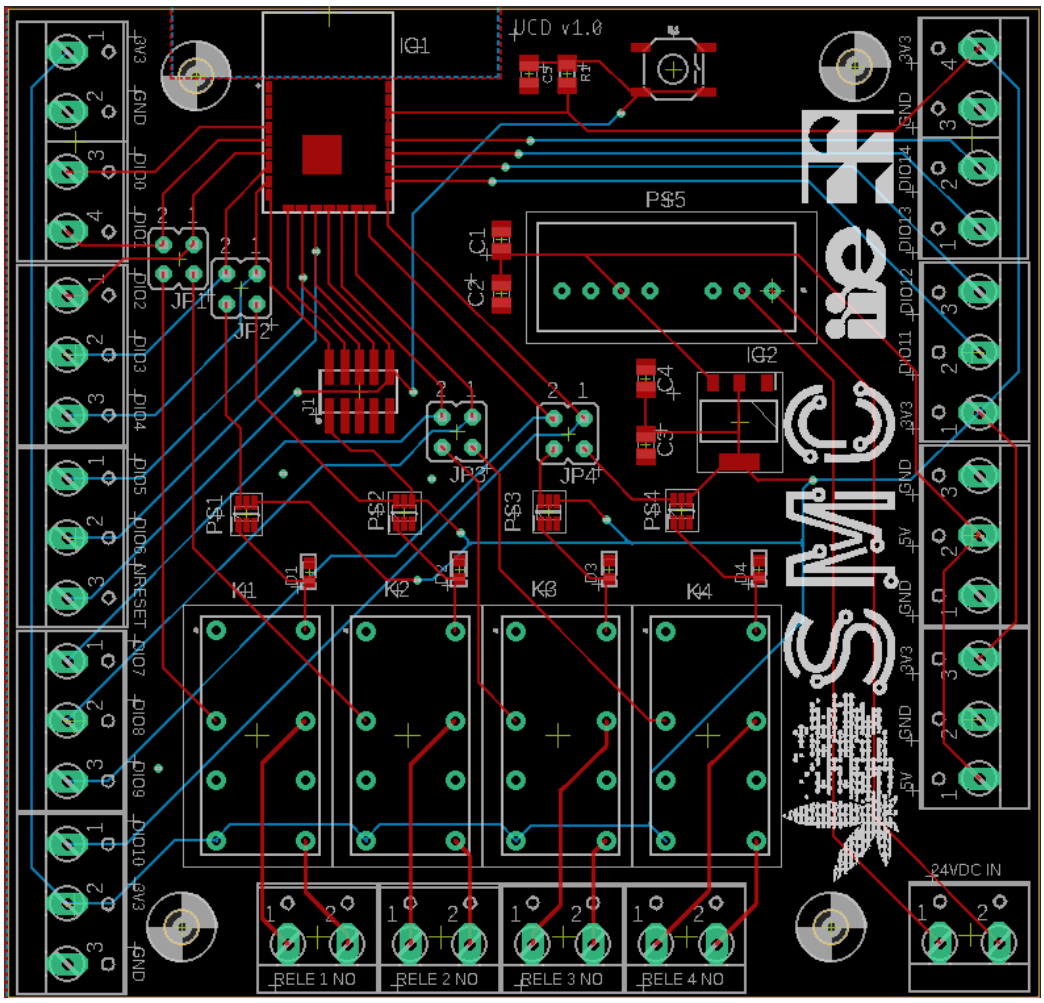


Figura B.12: Pistas UCD

B.4.2. Software

Para programar la placa se puede seguir las instrucciones detalladas en la sección B.6.2.

En la carpeta *Codigos/Contiki-NG/example/SMC*, se encuentran las carpetas con los programas de cada UCD. Adentro de estas carpetas se encuentra el archivo *ucd.h*, donde sedefine el período entre mensaje de telemetría, período del ciclo de control y se asignan las entradas y salidas.

B.5. Expandir sistema

B.5.1. Nodo

En caso de querer expandir el sistema, se recomienda partir de los ejemplos de código ubicados en la carpeta *Codigos/Contiki-NG/examples/SMC*.

Al compilar un programa, se debe tomar en cuenta que es necesario agregar las siguientes líneas en el archivo *project-conf.h*.

```
#define TB_CONNECTION      1
#define TSCH_CONF_ASSOCIATION_POLL_FREQUENCY  1000
```

También es necesario agregar la siguiente línea de código al archivo *makefile*, esto configura el stack de capa de enlace para utilizar el protocolo TSCH.

```
MAKE_MAC = MAKE_MAC_TSCH
```

El nodo a agregar deberá enviar mensajes a la IP fd00::1, ruta URI *api/v1/ TOKEN/telemetry*, donde *TOKEN* debe sustituirse por el *Token* asignado en Thingsboard.

B.5.2. SCADA

Para agregar la telemetría de un nuevo nodo, es necesario realizar ciertas modificaciones en el SCADA.Para agregar un nuevo dispositivo se debe acceder a Thingsboard con un usuario con permisos de administrador. En la figura B.13, se indica como añadir un dispositivo, primero se debe navegar a la pestaña *Dispositivos* (1). Luego se debe crear un dispositivo nuevo que represente el nodo agregado, para ello se debe presionar el botón con forma de cruz (2). En el menú que se despliega, se deberá seleccionar la opción de *Agregar nuevo dispositivo* (3). Luego se abrirá otra ventana (4) donde se pedirá ingresar un nombre para el dispositivo (5). Para elegir el *TOKEN* se debe presionar el botón que dice *siguiente* (6) con lo que se abrirá una ventana distinta (número 7 en la figura B.14) y se le deberá asignar un código de identificación al dispositivo o *TOKEN* (8). Finalmente se debe presionar el botón que dice *Agregar* (9).

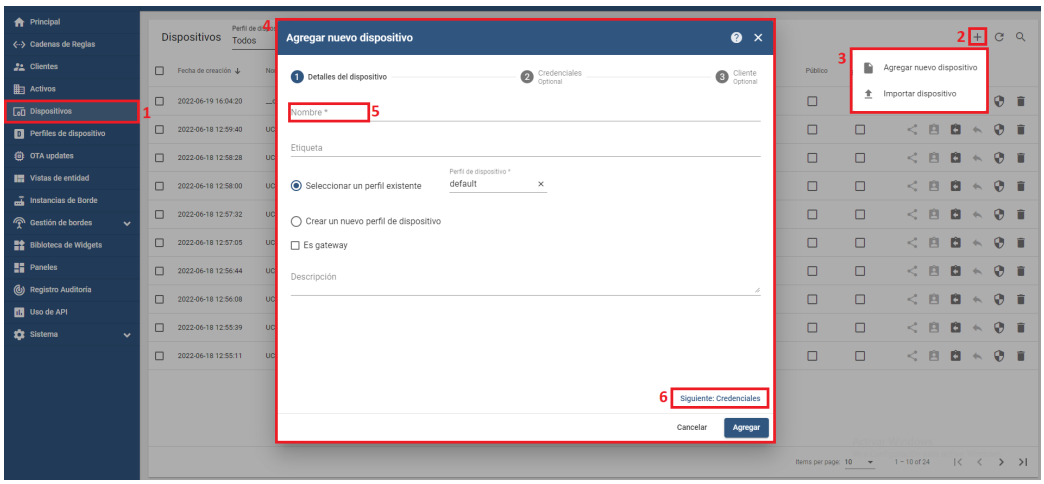


Figura B.13: Agregar Nodo SCADA

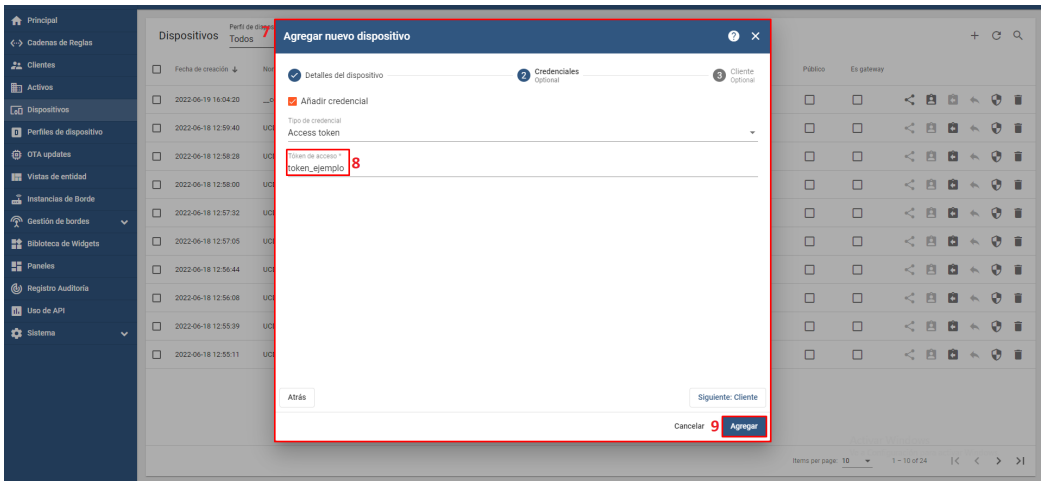


Figura B.14: Agregar Nodo SCADA

Con esta configuración, Thingsboard pasará a registrar los mensajes de telemetría que reciba a la dirección indica en la sección B.5.1. En la página de Thingsboard se cuenta con tutoriales y ejemplos de como crear pantallas y configurar el sistema [73].

B.5.3. *Script* Python

Si se quisiera agregar *widgets* al SCADA que envíen parámetros o comandos hacia los nodos, también se deben modificar los *scripts* desarrollados en Python. En caso de que el mensaje que se desee enviar sea del tipo ON y OFF se debe editar la base de datos local, es decir el fichero *todo_ db.json*, la misma puede verse en la figura B.15.

Este fichero es una tabla con 5 columnas, contiene el número de fila, el *token*

Figura B.15: Base de datos local

de Thingbsoard del originario del mensaje, en este caso el botón, la dirección IP del destinatario, el contenido del mensaje y la ruta URI al cual se esta enviando en mensaje POST.

Si se quisiera mandar mensajes numéricos desde el SCADA hacia un nodo se debe utilizar un *widget* de tipo *Update Server Attribute*, en cuyo caso se debe modificar el diccionario del *script* setvalues.py, indicado en la figura B.16

Figura B.16: Diccionario setvalues.py

Este *script* es un pasamanos entre el SCADA y el nodo destino, el diccionario debe tener en cada fila el *Token* del dispositivo originario y la dirección IP del nodo destino.

B.6. Desarrollo

En esta sección se explica como se puede desarrollar, se recomienda utilizar el entorno de desarrollo para esto, detallado en la sección B.1.

B.6.1. Thingsboard

Se puede desarrollar pantallas y configuraciones de la plataforma Thingsboard desde el entorno de desarrollo, utilizando la interfaz de usuario publicada en la

dirección local de esta computadora.

Los siguientes comandos permiten iniciar, detener y ver el estado de Thingsboard:

```
service thingsboard start
service thingsboard stop
service thingsboard status
```

Thingsboard permite la opción de exportar e importar pantallas y configuraciones, de este modo es posible desarrollar en este entorno y luego importar el sistema a la UC. Esto es útil ya que permite trabajar en este ambiente local, se pueden realizar pruebas y validar el correcto funcionamiento del sistema.

B.6.2. Contiki-NG

Una de las funciones que busca cumplir el entorno de desarrollo es brindar las herramientas necesarias para desarrollar la programación de los controladores.

En la carpeta *Codigos/Contiki-NG/examples/SMC* se encuentran los códigos utilizados para la programación de los NM y UCD del proyecto SMC. Se puede partir de estos códigos como base para desarrollar.

Utilización de la plataforma

Se creo una plataforma en Contiki-NG, llamada *cc2650moda* que se debe utilizar cuando se quiera programar un nodo, esto fue creado basándose en la plataforma *launchpad/cc2650* desarollada por la comunidad de Contiki-NG.

En el archivo *board.h*, se definen las particularidades del chip CC2650 de 5 mm x 5 mm, como parámetros de configuración para la radio, y a su vez en este archivo se definen contantes asociada a los pines de la placa para ser utilizadas por el desarrollador durante la programación.

Los pines de las entradas y salidas digitales se definen con el nombre *BOARD_IOID_DIOX*, donde X debe sustituirse por el deseado, pudiendo usarse:

- Para UCD: 0 al 15
- Para NM: 0 al 5, y 9 al 13.

Los pines de las entradas digitales se definen como *BOARD_IOID_AIX*, donde X debe sustituirse por el deseado:

- Para UCD: 7 al 14
- Para NM: 9 al 13.

En el caso de las UCD, para las señales asociadas a los relés se definen, *BOARD_RELAYX* la señal que actua el relé y *BOARD_RELAYX_FEEDBACK*

la señal de realimentación del estado del relé X, X debe sustituirse por el relé deseado, pudiendo usarse del 1 al 4.

En la figura B.10, se puede apreciar las entradas y salidas de la placa, y se explica como se deben de usar los *jumpers*.

Cargar programa

Las placas admiten programación por JTAG y UART, este último es el protocolo utilizado al programar vía *bootloader*. Si es la primera vez que se va a programar el SoC, es posible que *bootloader* este deshabilitado, en la sección B.7.2 se explica como habilitar este método de programación.

Previo a programar la placa deseada, se debe conectarla a la computadora a través de algún dispositivo que cumpla el rol de UART a USB. En caso de querer utilizar el Launchpad de Texas Instrument, se debe realizar la conexión indicada en la figura B.17.

Primero recomendamos retirar todos los *jumpers* del launchpad, dejando solamente el GND y 3V3. Se debe conectar la DIO0 y DIO1 de la placa (NM o UCD) al pin TXD y RXD (respectivamente), esto conecta la UART de la placa a la UART del launchpad. Se debe conectar 3V3 a B+ (3V3 en el caso de la UCD) y GND a GND, de este modo se alimenta la placa, no se debe utilizar otra alimentación, ya que es importante mantener los niveles de tensión para la comunicación por UART.

Por comodidad puede conectarse el DIO13 de la placa al DIO13 del launchpad, de este modo se puede utilizar el botón 1 del launchpad (BTN-1).

Para entrar en modo BSL (bootstraploader) y cargar un programa, se debe mantener pulsado el botón 13 del launchpad y luego pulsar el botón de *reset* de la placa, caso no se quiera utilizar el botón 13, se puede colocar la DIO13 a GND y luego pulsar el botón de *reset*.

Para compilar un programa de Contiki-NG primero se debe abrir la terminal en el directorio en el cual se encuentra el programa. En ese mismo directorio debe haber un Makefile, un archivo de configuración (project-conf.h) y uno o varios archivos .c y .h.

Para compilar se utiliza el comando:

```
make archivo.c TARGET=cc26x0-cc13x0 BOARD=cc2650moda
```

Para compilar y cargar el programa:

```
make archivo.upload TARGET=cc26x0-cc13x0
BOARD=cc2650moda PORT=/dev/ttyACM0
```

Puede ser útil el siguiente comando, este borra las compilaciones anteriores. Esto es necesario cuando se cambia el Makefile.

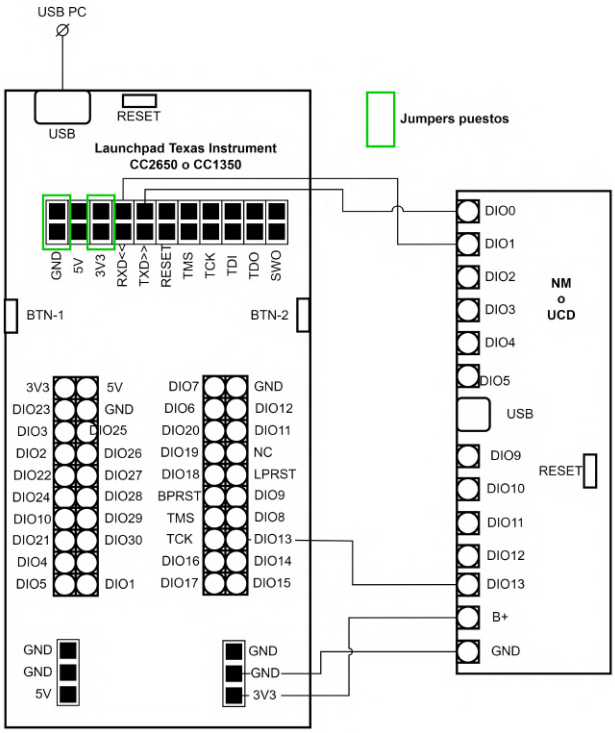


Figura B.17: Conexión Launchpad a NM o UCD para programar

```
$ make distclean
```

En el caso de programar el Border Router ver sección B.2.4, donde se detalla como se debe hacer en este caso.

B.7. *Troubleshooting*

B.7.1. Luego de programar SoC se trancó

Se ha comprobado que aveces al programar el SoC por *bootloader*, es posible que este quede trancado. Esto es frecuente cuando las entradas y salidas del SoC están siendo utilizadas. En el caso contrario, cuando solamente se utiliza el DIO 0 y DIO 1 para la comunicación por UART, no se da este problema.

La solución a este inconveniente es desconectar todo lo que este conectado a los bornes de la placa, dejando solamente lo necesario para la programación.

B.7.2. Programación *bootloader* deshabilitada

Es posible que la programación por *bootloader* no este habilitada desde el SoC, por lo que es necesario habilitarla si se desea utilizar este método de programación.

Esto podría realizarse de las siguientes formas:

1. Programar código compilado de Contiki-NG (de forma predeterminada ya viene habilitado *bootloader*) a través de JTAG.

2. Conectarse al SoC por JTAG y cambiar registro en memoria CCFG (Custom configuration area) del SoC, que configura programación *bootloader*.

Para la implementación de cualquiera de estos dos métodos, será necesario utilizar un conector JTAG 10 pines a JTAG 10 pines y conectar la salida XDS110 OUT de un Launchpad de Texas Instrument (puede ser CC1350 o CC2650) al conector JTAG de la placa. Es necesario desconectar todos los *jumpers* del Launchpad. En la figura B.18, se ilustra como se debe realizar la conexión.

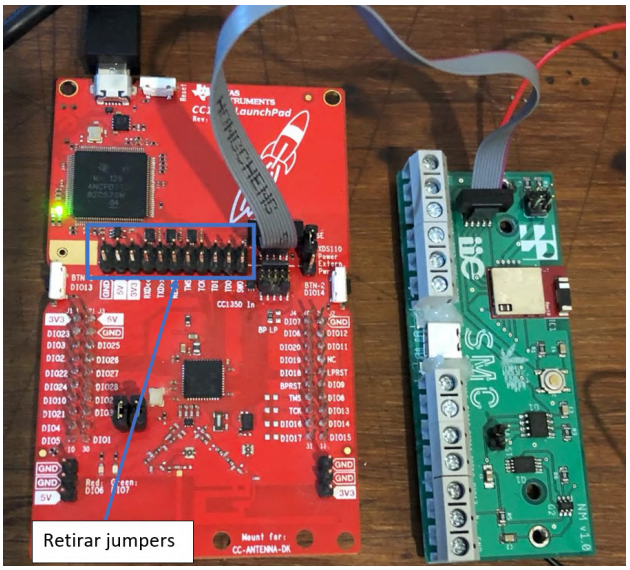


Figura B.18: Conexión Launchpad a placa

El método uno requiere de la instalación del programa Uniflash de Texas Instrument [62]. Puede ser necesaria la instalación del *driver* del XDS110 del Launchpad [64].

En la figura B.19, se indica la configuración que se debe seleccionar, esto es, SoC CC2650F128 y XDS110 USB Debug Proce. Luego se puede pulsar en Start. Una pantalla similar a la de la figura B.20 aparecerá y se deberá hacer *click* en la opción *Load Image* y seleccionar el archivo .bin generado al compilar el proyecto en Contiki-NG para esta placa, este archivo figurará en la carpeta del proyecto, bajo la dirección *build/cc26x0-cc13x0/cc2650moda*.

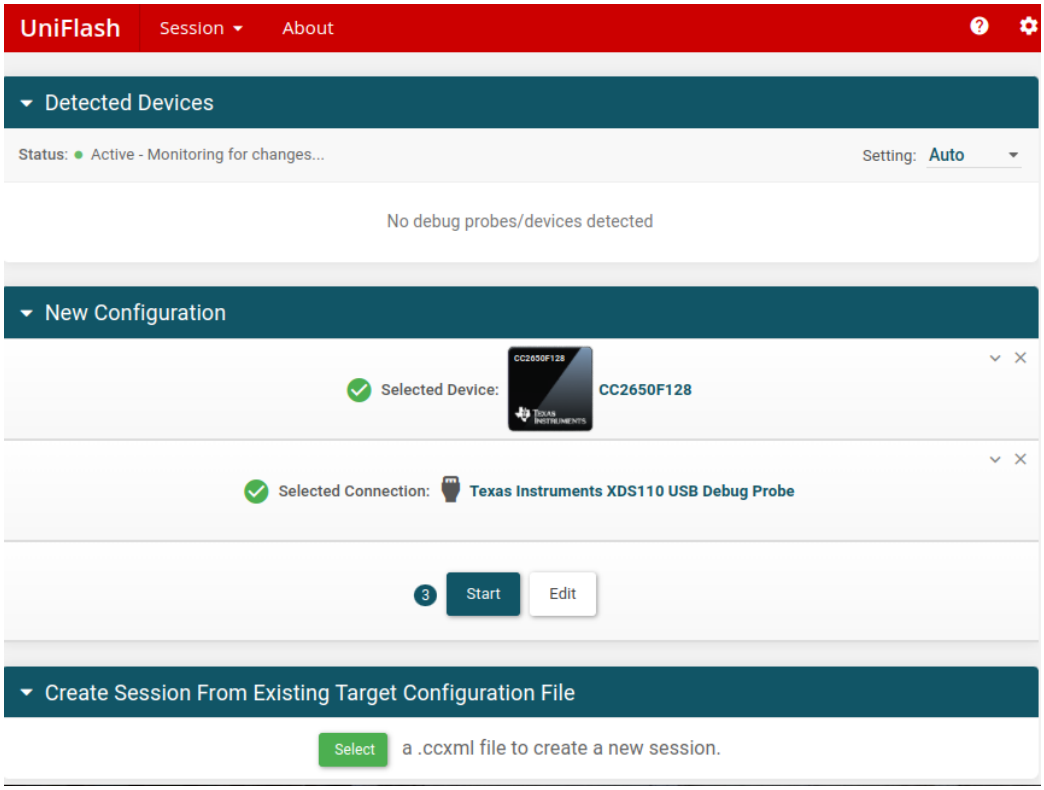


Figura B.19: Uniflash seleccionar SoC

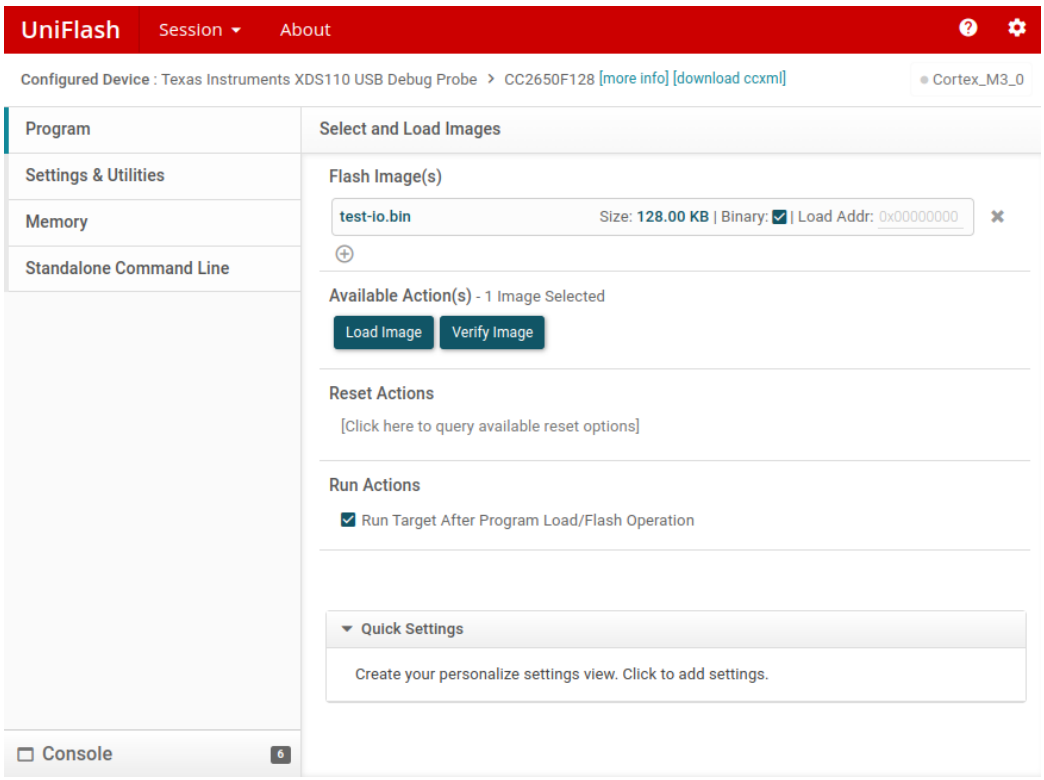


Figura B.20: Uniflash cargar programa

El segundo método requiere del programa Flash Programmer 2 de Texas Instrument, este solo soporta Windows. Al igual que en el método 1, puede ser necesaria la instalación del *driver* del XDS110.

En el programa, se debe conectar al SoC CC2650, y posteriormente leer la memoria CCFG. Luego se debe buscar el registro 0x1FFD8 y cambiar los siguiente 4 Bytes por el valor C5 0D FE C5. Estos valores habilitan el *bootloader* y configuran el SoC para que entre en modo BSL (*bootstraploader*) al colocar la DIO 13 a tierra y activar el botón de *reset*.

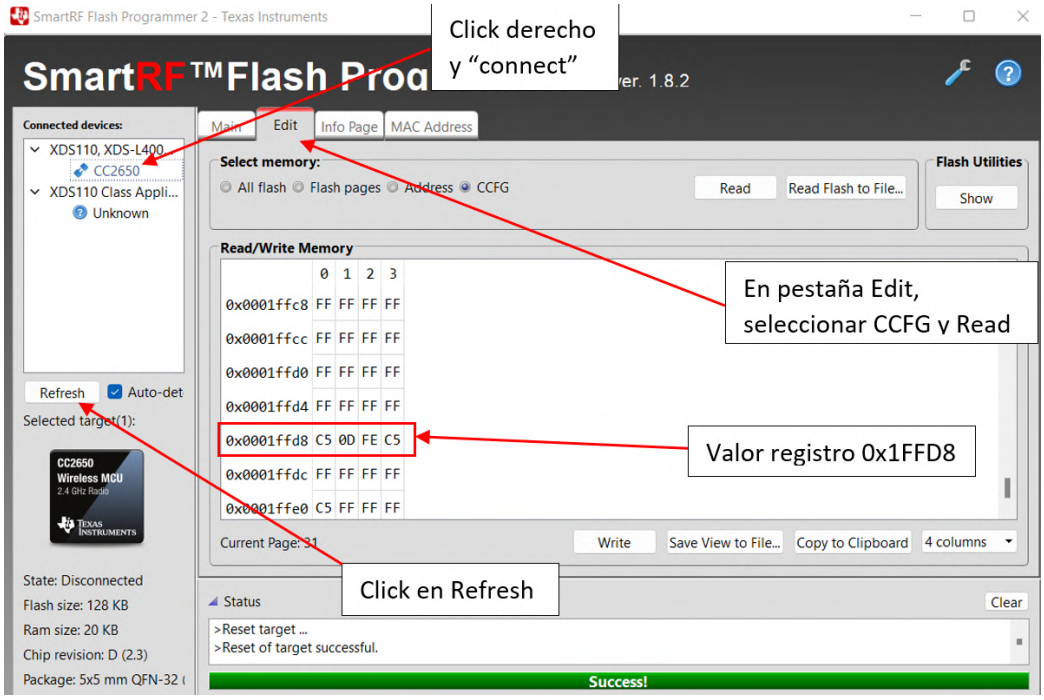


Figura B.21: Flash Programmer 2, conexión a SoC por JTAG

5

Esta página ha sido intencionalmente dejada en blanco.

Apéndice C

Manual de usuario

Este manual de usuario tiene el propósito de ser una guía de como utilizar el sistema de monitoreo y control desarrollado en el proyecto Sistema de Monitoreo y Control de Cultivo Indoor de Cannabis.

C.1. Interfaz SCADA

C.1.1. Acceso

La Unidad Central publica en internet la página web de interfaz de Thingsboard, la misma se puede acceder desde cualquier navegador ingresando la dirección *http://smc.webhop.me:8080/*. Al ingresar la página web solicita credenciales, tal como se indica en la figura C.1

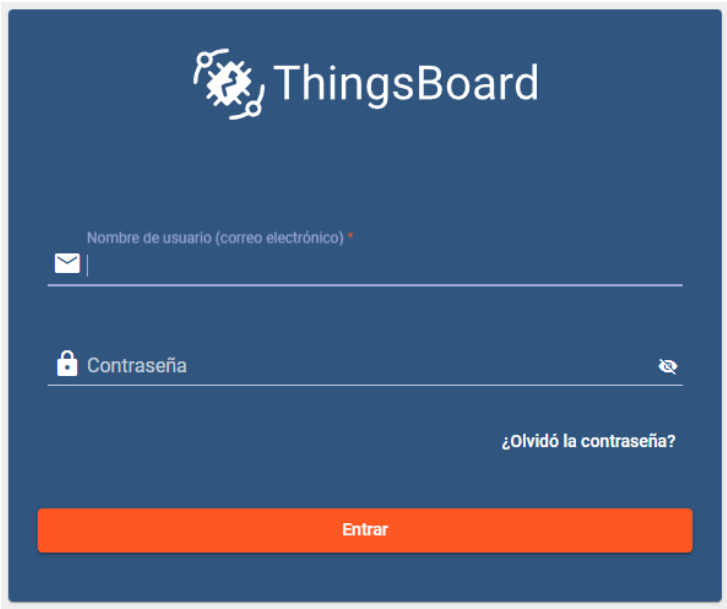


Figura C.1: Acceso web SCADA

Una vez ingresadas las credenciales, el usuario verá el panel General del sistema. Este panel muestra un pantallazo general de las variables más relevantes de cada sala (figura C.2)

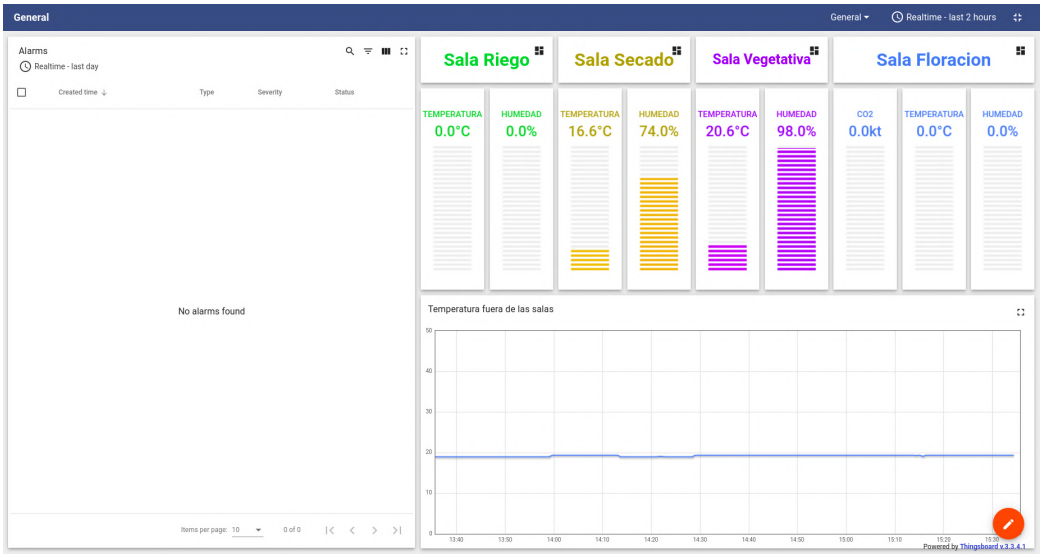


Figura C.2: Panel General del SCADA

C.1.2. Navegación

Mediante el menú de navegación superior es posible acceder al resto de los paneles del sistema: Floración, Vegetativo, Secado, Riego e Interno. Este último será accesible únicamente para aquellos usuarios con permisos de administrador dado el contenido de carácter técnico que presenta. Ver figura C.3.

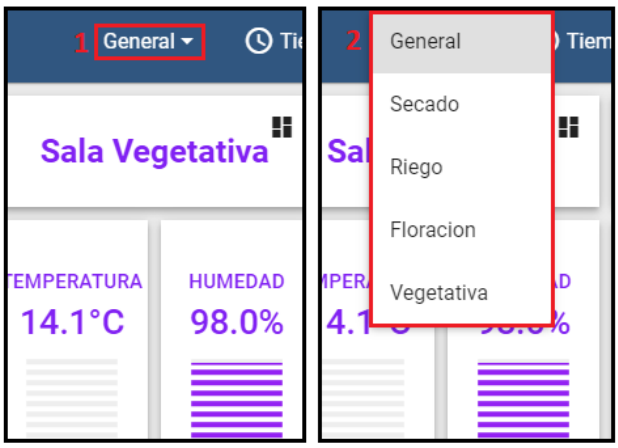


Figura C.3: Menú de navegación

Otra forma de navegar desde el panel General hacia los demás paneles es presionando el icono indicado en la figura C.4.

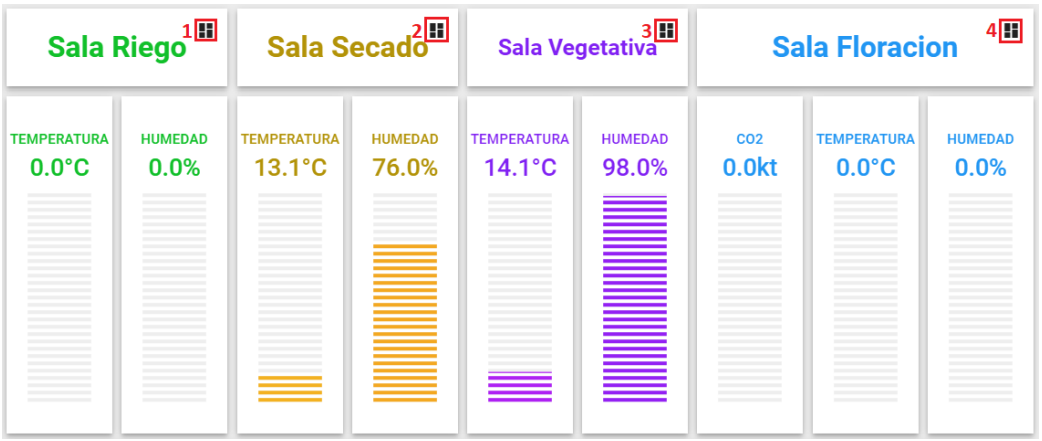


Figura C.4: Menú de navegación

De este manera se tienen dos métodos para navegar desde el panel General hacia los demás paneles del SCADA. Por otro lado, en cada panel distinto del General se encuentra un botón que permite retornar al panel General. Ver figura C.5.

Figura C.5: Botón de regreso

C.1.3. Históricos

Algunos *widgets* ya de por sí muestran el histórico de la variable, mientras que otros se debe acceder a una pantalla donde se utilizan *widgets* para mostrar el histórico, estos *widgets* presentan íconos que puede ser pulsados para acceder a los datos históricos de la variable que representan. En particular, esta opción se encuentra implementada para los estados de las luces y las variables relacionadas con los nodos maceta: temperatura del suelo, humedad del suelo y batería del nodo. Ver figura C.6.

En la figura C.7, se ilustra el histórico del estado de las lámparas de la Sala Floración.

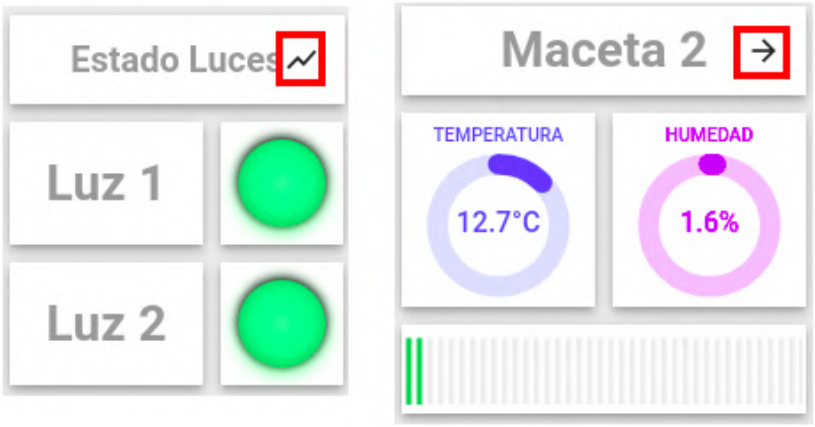


Figura C.6: Acceso a históricos

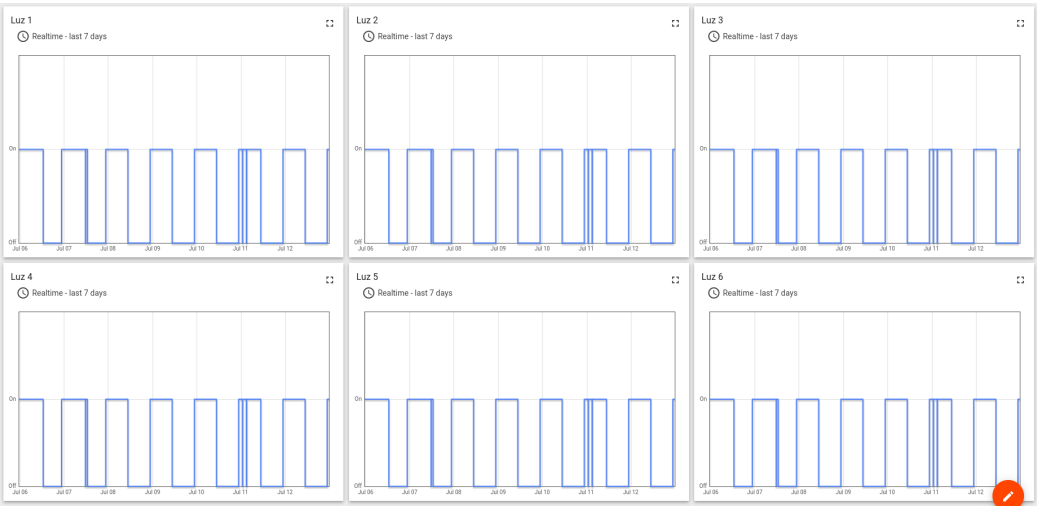


Figura C.7: Histórico luces Sala Floración

Es posible configurar la ventana de tiempo de las gráficas, esto se detalle en la sección C.1.5.

C.1.4. Paneles

Se cuenta con un total de seis paneles. En la figura C.8, se muestra una captura del panel de Floración.

Figura C.8: Panel Floración del SCADA

La figura C.9 presenta el panel de la Sala Vegetativo.

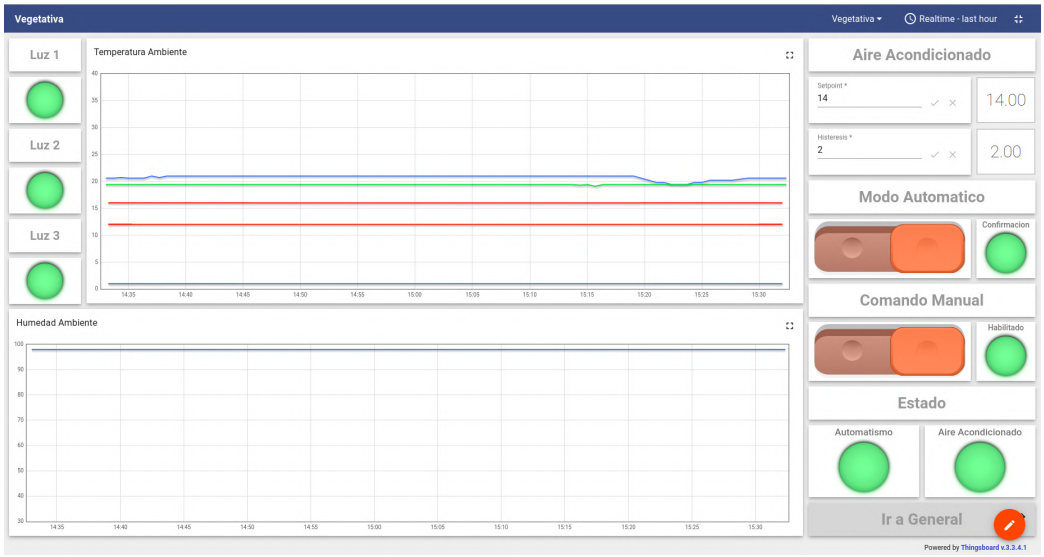


Figura C.9: Panel Vegetativo del SCADA

El panel de Secado se muestra en la figura C.10.

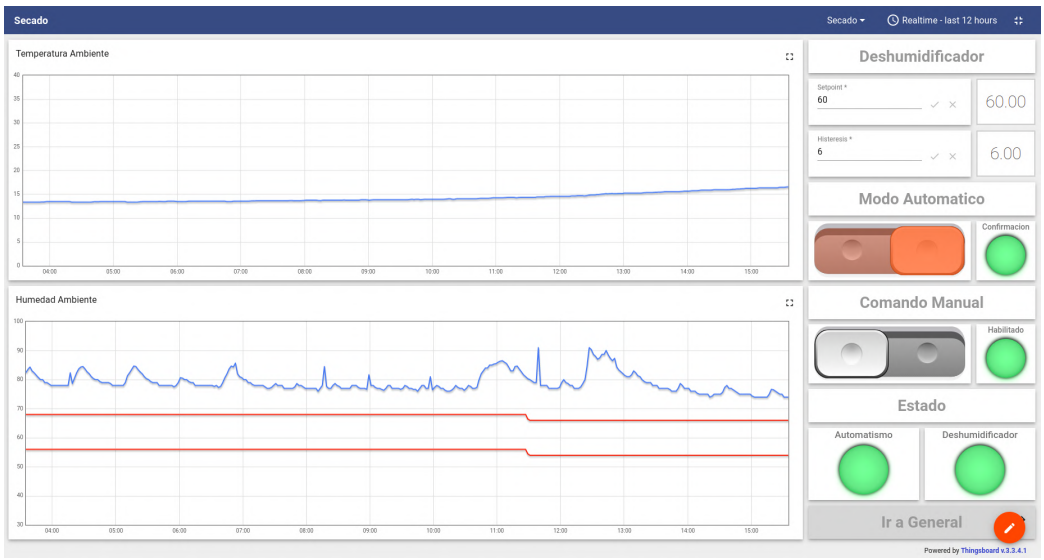


Figura C.10: Panel Secado del SCADA

La figura C.11, presenta el panel de Riego.

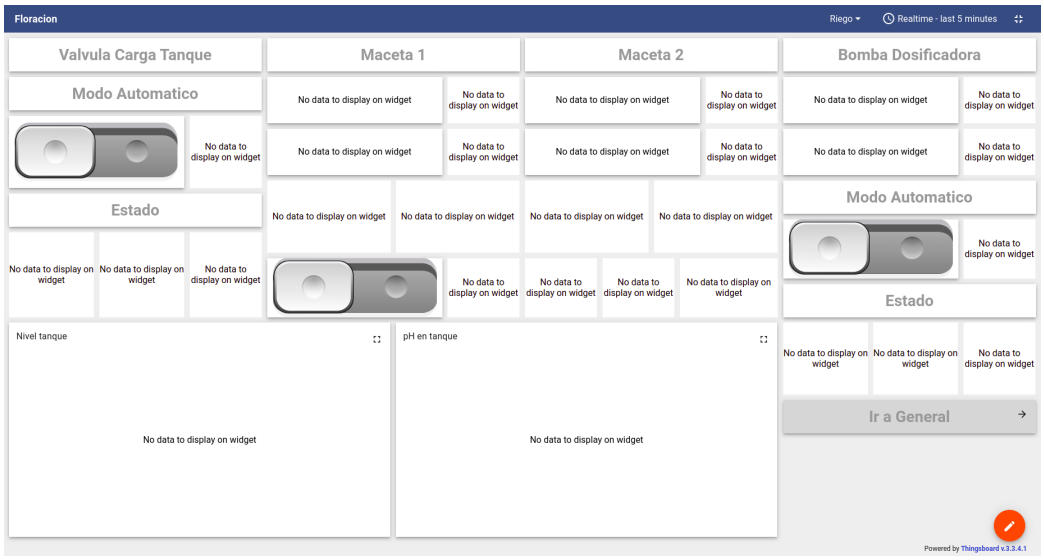


Figura C.11: Panel Riego del SCADA

Finalmente se tiene el panel Interno, restringido a usuarios con permisos de administrador debido al tipo de contenido técnico que presenta. Ver figura C.12.

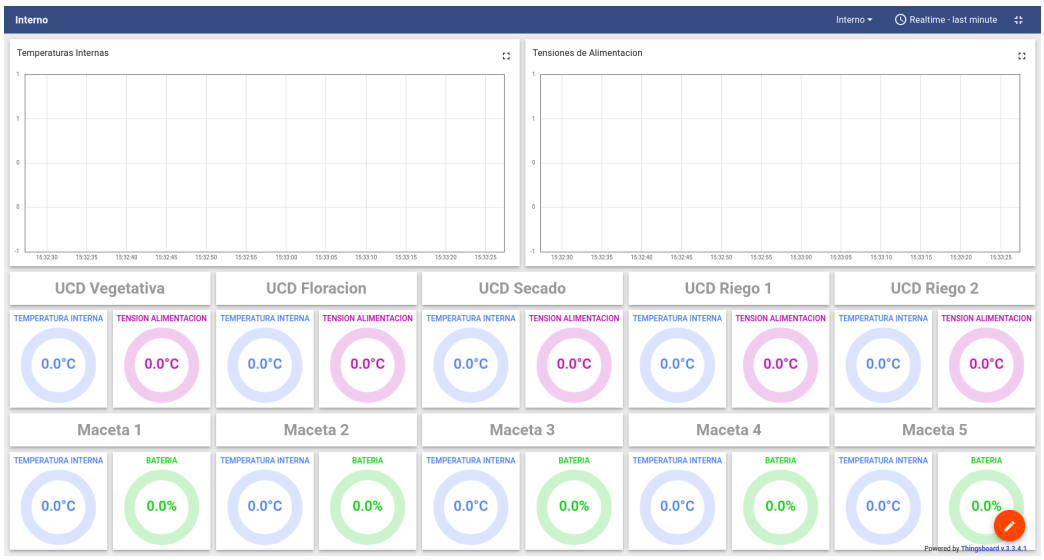


Figura C.12: Panel Interno del SCADA

C.1.5. Ventana de tiempo

En los diferentes paneles se pueden encontrar gráficas que muestran la evolución de una o varias variables durante un determinado intervalo en el tiempo. Es posible modificar este intervalo presionando el botón con icono de reloj en la esquina superior derecha, tal como se indica en la figura C.13.

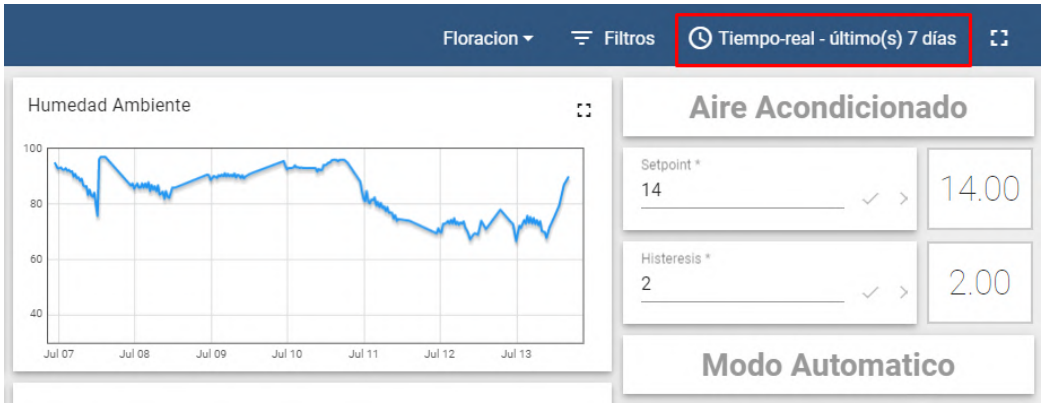


Figura C.13: Intervalo de tiempo

Al presionarlo se abrirá el menú de la figura C.14

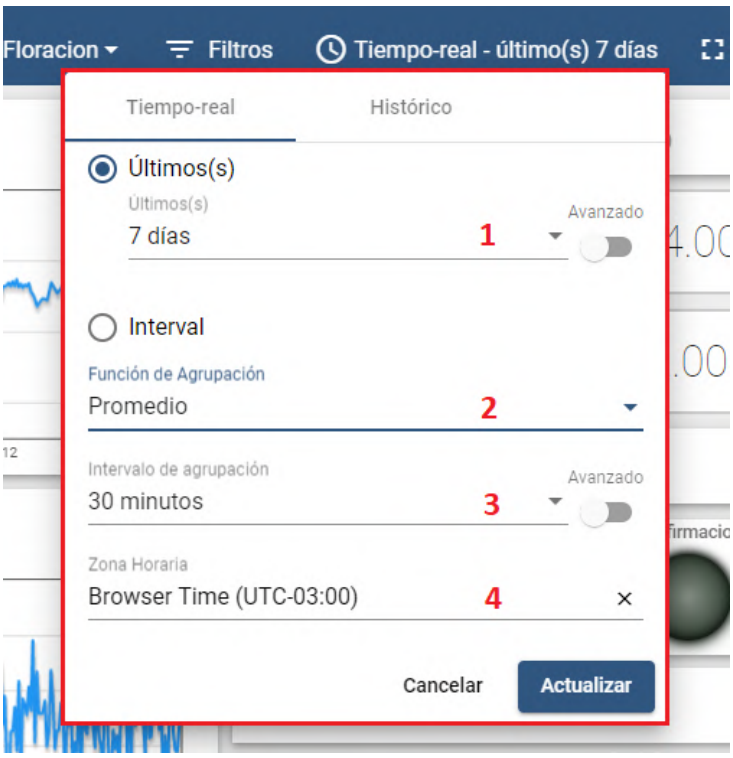


Figura C.14: Popup de configuración

En este menú se tienen varias opciones de configuración, a seguir se listan:

1. Primero: En esta entrada se puede seleccionar entre varios períodos de tiempo preestablecidos, desde un segundo hasta 30 días. Si en vez de Últimos se selecciona Intervalo se tienen más opciones de intervalos para elegir.

2. Segundo: En esta entrada se puede elegir como agrupar las últimas muestras. Se tiene las opciones de promediar, quedarse con el máximo, quedarse con el mínimo, sumarlas, entre otras.

3. Tercero: Esta opción está directamente relacionada con la anterior. Aquí se selecciona el intervalo de tiempo de muestras que se agrupará con la opción anterior. Por lo tanto, toma valores de tiempo desde 10 segundos hasta 5 minutos.

4. Cuarto: Ésta opción permite modificar el huso horario utilizado por el SCADA.

C.1.6. Alarmas

En el panel General se tiene acceso a un *widget* que provee la lista de las alarmas (ver figura C.15 y C.2). El *widget* presenta 5 columnas:

- *Created time:* Indica la fecha y hora de creación de cada alarma.

- *Type:* Indica el tipo de alarma, esta es la columna clave para identificar el origen de la misma.
- *Severity:* Indica la severidad de la alarma: Indeterminada, menor, mayor, crítica o peligro.
- *Status:* Indica en que estado se encuentra la alarma.
- La última columna muestra dos botones, uno con forma de visto que se utiliza para reconocer la alarma y otro con forma de cruz que se utiliza para desactivar la alarma manualmente en caso de ser necesario.

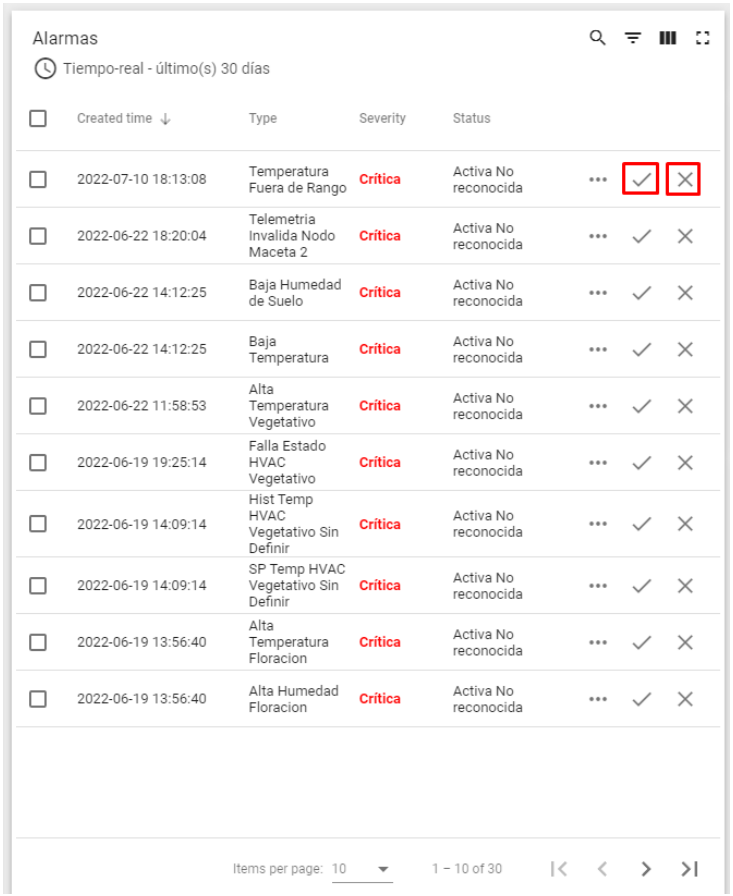


Figura C.15: *widget* de Alarmas

Para cada alarma se distinguen cuatro posibles estados de la misma:

- Activa no reconocida: la alarma no se ha desactivado ni reconocido todavía.
- Activa reconocida: la alarma no se ha desactivado, pero ya se reconoció.
- Inactiva no reconocida: la alarma ya se desactivó, pero aún no se reconoció.
- Inactiva reconocida: la alarma ya se desactivó y reconoció.

Alerta de alarma

Cada vez que se activa una alarma, además de mostrarse en el *widget* que se mencionó en la sección anterior, se le notificará al usuario a través de un correo electrónico (figura C.16).

Figura C.16: Bandeja de Entrada

C.1.7. Configuración de parámetros

Existen dos tipos de *widgets* que son utilizados para configurar parámetros en los nodos. Cada uno de ellos es acompañado de otro *widget* que indica si la configuración fue exitosa o no. En la figura C.17, se presenta un botón.

Figura C.17: Botón

Este tipo de *widgets* se utilizan para enviar mensajes de prender y apagar o activar y desactivar a una UCD. Cuando el nodo reciba un mensaje, responderá un mensaje a los 500ms, en mensaje contiene el valor de confirmación. En este caso, tanto el botón como la luz (confirmación), deben encenderse y apagarse en conjunto.

Esto no ocurren en el caso de comando manual (figura C.18), ya que la indicación luminosa que se utiliza tiene el propósito de indicar si el comando manual esta habilitado desde la selectora en campo, la selectora debe estar en automático para que se pueda operar desde el SCADA, esto se detalla en la sección C.2.

Figura C.18: Comando manual

El otro *widget* se llama *Update Server Attribute* y se utiliza cuando se quiere mandar a un nodo un mensaje conteniendo un numero. Ver figura C.19.

Figura C.19: *Update server attribute*

El funcionamiento de la confirmación es análogo, pero como lo que se envía y recibe en el mensaje es un numero, se utiliza una etiqueta que simplemente muestra lo que recibe de la confirmación desde el nodo.

C.2. UCD - Llave Selectora

Como se mencionó anteriormente, las cajas que protegen y cubren a las UCD dejan al descubierto llaves selectoras como la que se observa en la figura C.20.

El principal propósito de estas llaves es disponibilizar el actuador que se controla para el usuario. En caso de que el control no este funcionando correctamente, el usuario podrá utilizarlo de forma manual, desactivando el automatismo desde la llave selectora o desde el SCADA, la llave tiene prioridad sobre el SCADA, de este modo, en caso no funcione el SCADA, el operario no perderá el control del actuador.

Las llaves se pueden encontrar con dos o tres posiciones. Las de tres posiciones, son de izquierda a derecha: manual (actuador permanece energizado), apagado (actuador permanece desenergizado) y automático (el sistema realiza el control). Cuando está en automático, se habilita tanto el control de lazo cerrado que ejecuta el sistema, como el control manual pero remoto (desde panel del SCADA).

Las llaves de dos posiciones, funcionan igual, solamente no presentan la opción manual, por lo que el sistema controla el actuador (automático), o el actuador permanece desenergizado (apagado).

Figura C.20: Llave Selectora UCD

C.3. NM - Indicador de Carga

Los nodos macetas presentan dos luces led (ver figura C.21) que se pueden utilizar para conocer el estado de la batería del nodo. El led rojo se encenderá siempre y cuando la carga de la batería caiga por debajo del 15 %. De lo contrario este led se encontrará apagado.
Por otro lado, el led verde se encenderá únicamente cuando la batería del nodo maceta se encuentre cargándose y se apagara una vez se encuentre completamente cargada.

Figura C.21: Leds indicadores de carga

C.4. VNC - Acceso Gráfico Remoto

Puede ser de interés acceder de forma remota al escritorio gráfico de la UC. Para ello, la Raspberry cuenta con un servidor VNC configurado, al cual se puede conectar mediante la aplicación VNC viewer. Esta aplicación puede ser instalada tanto en SO basados en linux como en Windows [49].

Una vez instalado se abrirá una ventana como la que se muestra en la figura C.22. En la barra de búsqueda superior se debe ingresar la IP pública del *router* de Antel. En el manual técnico se explica como se debe configurar la UC para que funcione el acceso remoto.

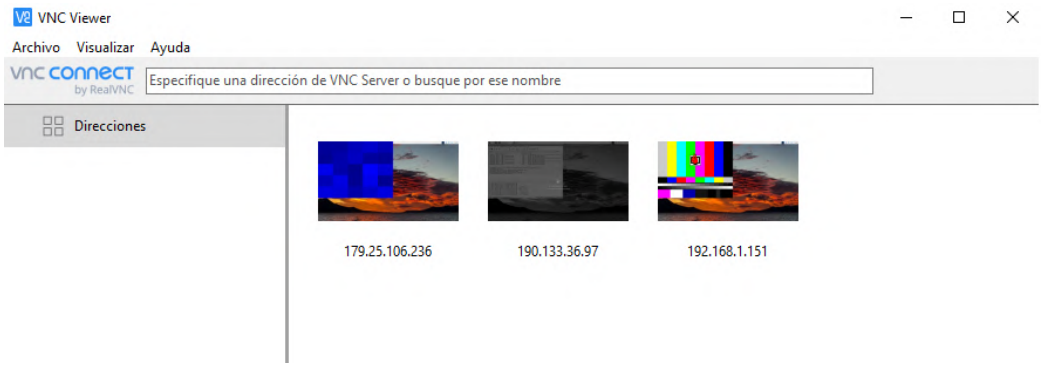


Figura C.22: Aplicación VNC Viewer

C.5. *Troubleshooting*

C.5.1. Los botones de los paneles presentan error

Puede suceder, que se pierda la comunicación entre los *widgets* de los paneles y el resto de la red. Cuando esto sucede, al actualizar, abrir un panel o presionar un botón se podrá ver al cabo de unos segundos un cartel rojo que aparece sobre el mismo *widget* con un mensaje de *Request Timeout* y un botón para cerrar la notificación, tal como se indica en la figura C.23

Figura C.23: Botón timeout

Cuando esto ocurre, se resuelve reiniciado la UC. Se recomienda desenchufar, esperar un minuto y volver a enchufar el equipo.

C.5.2. No se tiene acceso remoto a la pagina web pero si local

Es posible que se tenga acceso a la pagina web del SCADA de forma local pero no de forma remota. A seguir se listan algunas causas y posibles soluciones:

- La dirección IP cambio: La UC envía la IP a No-IP cada 5 minutos. En caso de que cambie la IP enseguida se envío IP anterior a No-IP, no se podrá acceder a la página web del SCADA por 5 minutos.
- El *router* perdió conexión a internet: Si por alguna razón el *router* de Antel perdiera el acceso a internet, no sería posible conectarse a la página web de forma remota. En este caso, se puede probar desenchuar el *router*, esperar un minuto y volver a enchufar.
- El servicio de No-IP se cayó: Si se tiene acceso a internet a través del *router* y no se tiene acceso a la página web hace más de 5 minutos, es posible que el servicio de No-IP no se encuentre ejecutandose en la UC. Para solucionarlo, se recomienda desechufar, esperar un minuto y volver a enchufar el equipo.

Apéndice D

Proyecto de Fin de Curso de RSI

En este apéndice se adjunta el informe final del proyecto de fin de curso de Redes de Sensores Inalámbricos. En este proyecto se probó la plataforma Thingsboard y Mango OS, además se realizó un análisis sobre el protocolo RPL, comparando RPL Lite y RPL Classic.

Proyecto de Fin de Curso
Redes de Sensores Inalámbricos
Año 2021

Implementación de red de sensores inalámbricos y análisis de protocolo RPL

Grupo 1

Autores

Diego Alonso - 4.585.687-0 - diego.alonso.ruiz@fing.edu.uy

Carlos Menoni - 4.632.018-5 - carlos.menoni@fing.edu.uy

Felipe Saravia - 4.720.555-8 - felipe.saravia@fing.edu.uy

Tutor

Leonardo Steinfeld

Facultad de Ingeniería - Universidad de la República

15 de julio de 2022

Índice

1. Introducción **1**
- 1.1. Objetivos . . . 1
- 1.2. Alcance . . . 1

2. Entorno de Trabajo **3**

3. Pila de Red **4**

4. Subcapa de Ruteo - RPL **5**
- 4.1. RPL CLassic . . . 5
  - 4.1.1. Grafos . . . 5
  - 4.1.2. Mensajes de Control . . . 5
  - 4.1.3. Modos de Operación . . . 6
  - 4.1.4. Rutas Hacia Arriba . . . 7
  - 4.1.5. Rutas Hacia Abajo . . . 7
- 4.2. RPL Lite . . . 7
  - 4.2.1. Mensajes de Control . . . 7
  - 4.2.2. Modo de Operación . . . 8
  - 4.2.3. Source Routing Header . . . 8
- 4.3. Simulaciones Cooja . . . 9
  - 4.3.1. Formación de la Red . . . 10
  - 4.3.2. Rutas Hacia Arriba . . . 11
  - 4.3.3. Rutas Hacia Abajo . . . 12
- 4.4. Comparación Final . . . 16

5. Aplicación **17**
- 5.1. Arquitectura del sistema . . . 17
- 5.2. Drivers . . . 18
- 5.3. Nodos - Servidores CoAP . . . 18
- 5.4. Border Router . . . 19
- 5.5. Aplicación final . . . 19
  - 5.5.1. Mango OS . . . 19
  - 5.5.2. ThingsBoard . . . 19

6. Conclusiones del Proyecto **21**

7. Bibliografía **22**

8. Anexo **24**
- 8.1. Entorno de Trabajo . . . 24
  - 8.1.1. Instalar Contiki-NG . . . 24
  - 8.1.2. Instalar Code Composer Studio . . . 24
  - 8.1.3. Compilar un proyecto y utilizar Debug . . . 27

1. Introducción

Este documento presenta las características principales del Proyecto de Fin de Curso de Redes de Sensores Inalámbricos (RSI). En ese sentido, se busca investigar y desarrollar una implementación práctica de los temas expuestos en dicho curso.

El curso de RSI se centra en la descripción y estudio de una pila de comunicación inalámbrica, analizando cada una de las capas del stack y algunas de las tecnologías y protocolos involucrados. Es así que durante el curso se vieron temas relativos al OS Contiki, los protocolos CoAP y MQTT, el protocolo RPL de capa de red, el estándar 6LoWPAN, y características del estándar IEEE 802.15.4 para las capas MAC y PHY.

Por otra parte, los estudiantes autores conforman un grupo de trabajo llevando a cabo el Proyecto de Fin de Carrera (PFC). De esta forma, se busca también que los objetivos de esta instancia se relacionen con el PFC.

El presente proyecto se nutre de diversas publicaciones, destacándose el libro en el que se basa el curso de RSI [1], la documentación del repositorio de Contiki-NG [2], el PFC BeeSmart [3], además de otros tutoriales y recomendaciones de páginas de fabricantes, detalladas más adelante.

1.1. Objetivos

El objetivo general de este proyecto es implementar una red de sensores inalámbricos, basados en el OS Contiki-NG.

Se toma como objetivo particular el desarrollo de la red RPL en los modos *non-storing* y *storing* entre los nodos, y así comparar el desempeño de la red. Se trabaja también a nivel de capa de aplicación elaborando los programas necesarios para que éstos se comuniquen correctamente.

También se implementa una aplicación de monitoreo de algunos sensores. Para lograr esto se deben desarrollar los drivers necesarios.

1.2. Alcance

Se estudia el manejo de Contiki-NG, a fin de adaptar lo estudiado en el curso de RSI y definirlo como OS de nuestro diseño, que también se utilizará en el PFC. Se buscará integrar Contiki-NG al software Code Composer Studio de Texas Instruments, y aprovechar las herramientas de Debug desarrolladas por Texas Instruments.

Se trabaja a nivel de capa de red, estableciendo una red RPL en modo *non-storing* y otra en *storing*, con al menos tres nodos.

Luego de implementada la red RPL, se evalúan y comparan las opciones *RPL Classic* (en su modo de operación storing) y *RPL Lite*.

También se desarrollan los programas a nivel de capa de aplicación que implementen la comunicación de los nodos. En esta capa se utiliza el protocolo CoAP.

Se elaboran los drivers necesarios para los sensores de humedad ambiente y temperatura, y humedad del suelo.

Las pruebas se realizan tanto a nivel de simulación con el programa Cooja (visto en el curso) como en hardware. Para esta última opción, se utilizan launchpads con el microcontrolador CC1350 de Texas Instruments.

También se integra la aplicación final al software SCADA que se utilizará en el PFC, MangoOS.

A seguir, se presenta un diagrama indicativo de la red:

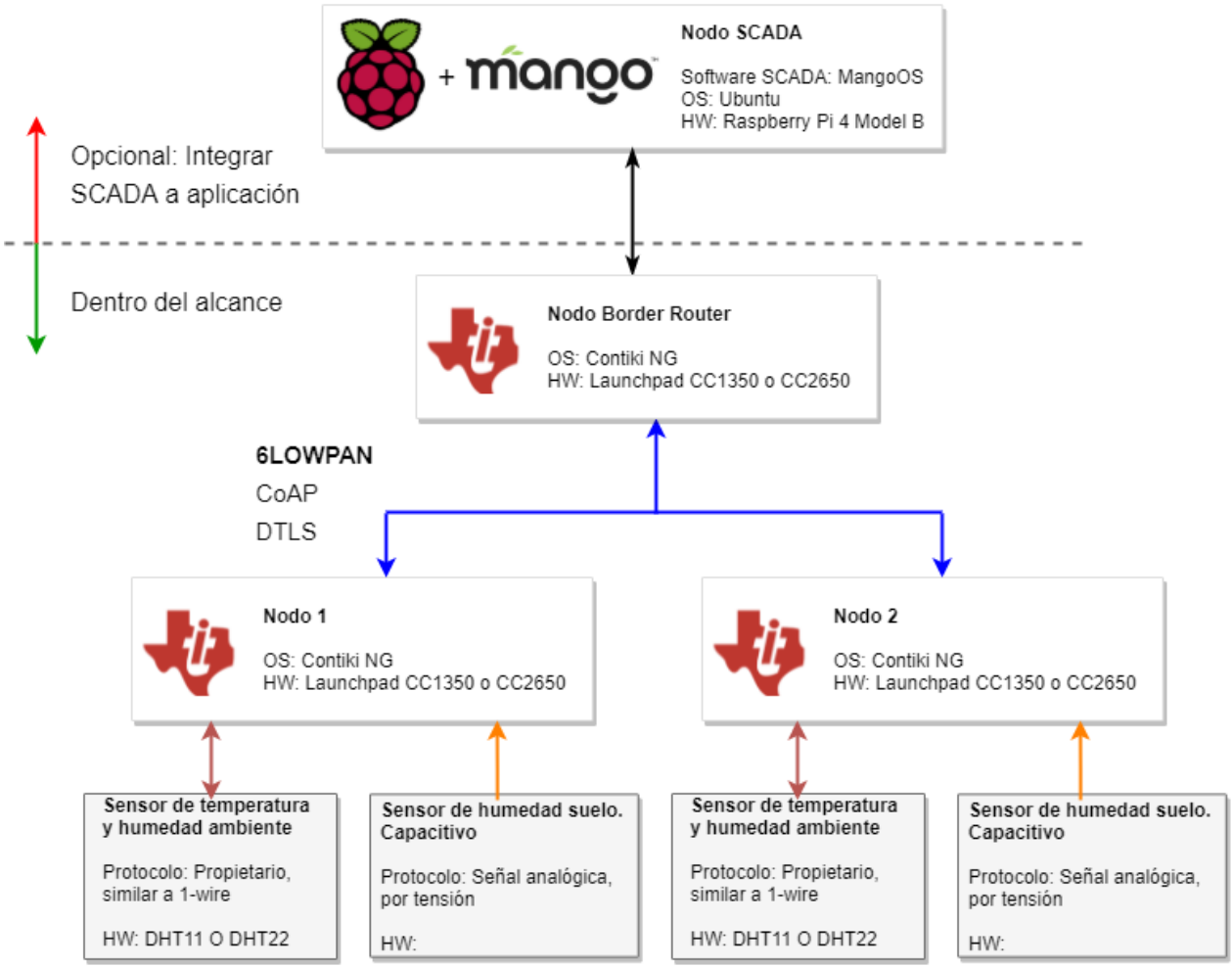


Figura 1.1: Diagrama de la red

2. Entorno de Trabajo

Al inicio del proyecto se intentó trabajar utilizando la máquina virtual utilizada durante el curso de RSI. Sin embargo, esto no fue posible debido a que en Contiki-NG la herramienta Cooja esta basada en una arquitectura de 64-bits y la máquina virtual estaba basada en 32-bits. Por lo tanto, se decidió crear una máquina virtual de 64 bits.

Se deseaba integrar Contiki-NG a Code Composer Studio (CCS), para así facilitar la depuración del sistema, utilizando la herramienta Debug de CCS. Se decidió utilizar CCS 11, la versión más actual a la fecha. CCS 11 soporta la distribución Ubuntu 20.04 de 64 bits. Por lo tanto, se armó una máquina virtual donde se instalaron las siguientes herramientas:

- Ubuntu 20.04.
- Contiki-NG v4.7.
- Wireshark.
- CCS 11.
- XDS Emulation Software (software para que CCS logre conectarse a la placa utilizando el SoC XDS110 que viene integrado en el launchpad CC1350.)
- UniFlash v7.0 (programa para conectarse con la placa y programar archivos .bin y .hex al SoC.)
- Firefox versión 55, con la extensión Copper.
- Python 3 (para ejecutar scripts desarrollados para integrar la aplicación final con la capa de aplicación. Ver sección 5, donde se describen las librerías necesarias.)

En el Anexo 8.1, se describe como instalar Contiki-NG y CCS 11, así como las librerías necesarioas para estos programas. También se explica como utilizar la herramienta debug de CCS 11. Integrar Contiki-NG a CCS, fue una de las tareas que demandó más tiempo.

Nos pareció importante utilizar Copper, ya que nos facilita la lectura y envío de mensajes CoAP. Se intentó instalar Copper a la última versión de Firefox, pero esta no es compatible. Se logró instalar la extensión Copper Cu4Cr para Chrome, sin embargo no logramos utilizar esta extensión por distintos problemas. Por último, se instaló Firefox v55, la cual es compatible con Copper, y se logró utilizar con éxito la extensión.

3. Pila de Red

Este proyecto se basa fuertemente en la pila de red estudiada durante el curso, la cual parte del modelo de 5 capas de redes. La misma está compuesta por protocolos que son los más comunes a la hora de trabajar con redes LLN (Low Power and Lossy Networks) y WPAN (Wireless Personal Area Networks).

La primer capa, se le llama capa física. Corresponde a los medios físicos que interconectan los nodos y a las transformaciones realizadas a las secuencias de bits para transmitirlas de un medio a otro. En nuestro caso, la capa física está regida por el estándar IEEE 802.15.4.

La siguiente capa es la de enlace, que se encarga del direccionamiento, acceso al medio, control de flujo, etc. Para este proyecto mantenemos lo estudiado durante el curso, bajo el estándar IEEE 802.15.4.

La tercer capa, denominada capa de red, la dividimos en dos subcapas: la subcapa de ruteo y la subcapa de adptación. En la primer subcapa utilizaremos el protocolo RPL (Routing Protocol for LLNs). Se estudiarán dos versiones de este protocolo, el RPL Classic y el RPL Lite. Para la subcapa de adaptación se aplicará el estándar 6LowPAN (IPv6 over Low Power WPANs), que posibilita el uso del protocolo IPv6 en redes basadas en IEEE 802.15.4.

La siguiente capa es la de transporte, en la que este proyecto no profundiza demasiado. Se utilizará el protocolo UDP, comúnmente usado para este tipo de aplicaciones y estudiado en el curso.

Por último, en la capa de aplicación utilizaremos CoAP (Constrained Application Protocol). Es un protocolo de tipo REST, que trabaja en un formato de cliente-servidor similar a HTTP, con ciertas restricciones para aplicaciones similares a las de este proyecto.

En la figura 3.1, se presenta el Stack de protocolos.

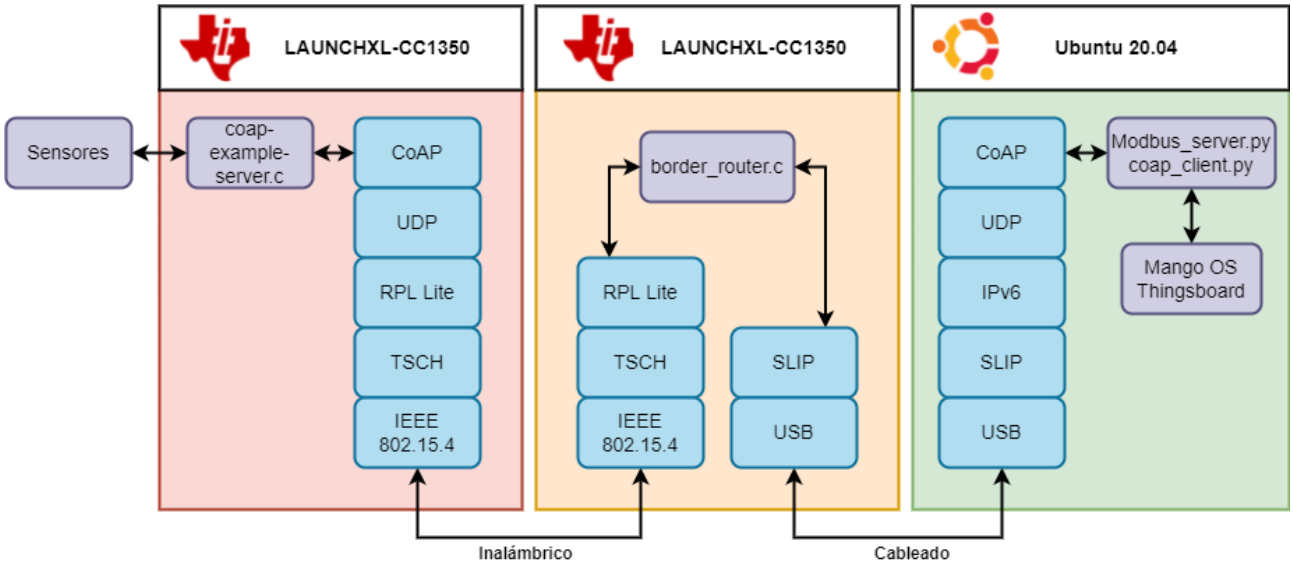


Figura 3.1: Stack de Protocolos.

4. Subcapa de Ruteo - RPL

Para la subcapa de ruteo (dentro de la capa de red IPv6), se elige el protocolo RPL (*Routing Protocol for Low-Power and Lossy Networks)*, definido en la RFC 6550 en el año 2012 [4].

Si bien este protocolo está dirigido a aplicaciones con nodos de recursos limitados, para el año 2017 surge la versión RPL Lite, incluyendo solo las características mas importantes de la original. En ese sentido, esta nueva versión mejora en cuanto a complejidad de implementación, así como reduce la huella de memoria en el nodo.

A los efectos de este trabajo, se propone comparar la implementación de RPL Classic con RPL Lite, buscando identificar, enumerar y contrastar sus diferencias, analizando la relación costo-beneficio de cada una y sus posibles aplicaciones. Para este análisis, nos apoyaremos en [5], [2] y [1].

4.1. RPL CLassic

El protocolo RPL original es un protocolo de enrutamiento para redes inalámbricas proactivo, basado en vectores de distancia. Se implementa bajo IPv6 e IEEE 805.15.4, mientras que admite variabilidad en capa de aplicación y transporte.

4.1.1. Grafos

RPL crea un topología de tipo árbol, denominada *DAG* (Directed Acyclic Graph), en la cual un nodo (o varios) actúa como *DAG root* al cual conducen todas las rutas.

A su vez, este DAG puede dividirse en subredes de acuerdo a los distintos roots. La subred de cada root se llama *Destination Oriented DAG* o simplemente *DODAG*.

Para crear la topología, los nodos intercambian mensajes de control con sus vecinos y determinan cuál vecino es mas conveniente para comunicarse con el root. Para ello se utiliza el parámetro *Rank*, que indica la distancia al root de cada nodo de acuerdo a cierta métrica definida como *Función Objetivo*. Cuando un nodo elige un vecino mediante el cual comunicarse con el root, se dice que lo toma como *padre*.

A su vez, para cada nodo se define un *Sub-DODAG* donde el root es dicho nodo y en el que están incluidos todos los nodos que lo hayan tomado como padre y el resto de los hijos sucesivos.

En RPL, los nodos de una red pueden pertenecer a múltiples instancias. Esto puede ser útil para tráficos con distintos requierimientos, perimitiendo implementar varias funciones objetivo o tener distintos roots. En estos casos se utilizan los parámetros *DODAGID* y *RPLInstanceID*.

4.1.2. Mensajes de Control

Para configurar los grafos, los nodos intercambian mensajes de control. En estos mensajes cada nodo comparte información con sus vecinos para elegir la topología óptima.

El primer mensaje de control enviado por un nodo (no root) es el *DODAG Information Solicitation* o *DIS*. En este mensaje, enviado a la dirección multicast, se solicita información a los vecinos para así iniciar el proceso de integrarse a un DODAG.

El mensaje llamado *DODAG Information Object* o *DIO*, sirve para crear, formar o mantener actualizado el grafo. Un nodo envía un DIO por múltiples causas: al haber recibido un DIS, al

cambiar su rank, al cambiar de padre, cada cierto tiempo, etc. En este mensaje se incluyen los campos de configuración del grafo mostrados en la figura 4.1, que permiten a un nodo conocer las características del DODAG y unirse de forma correcta.

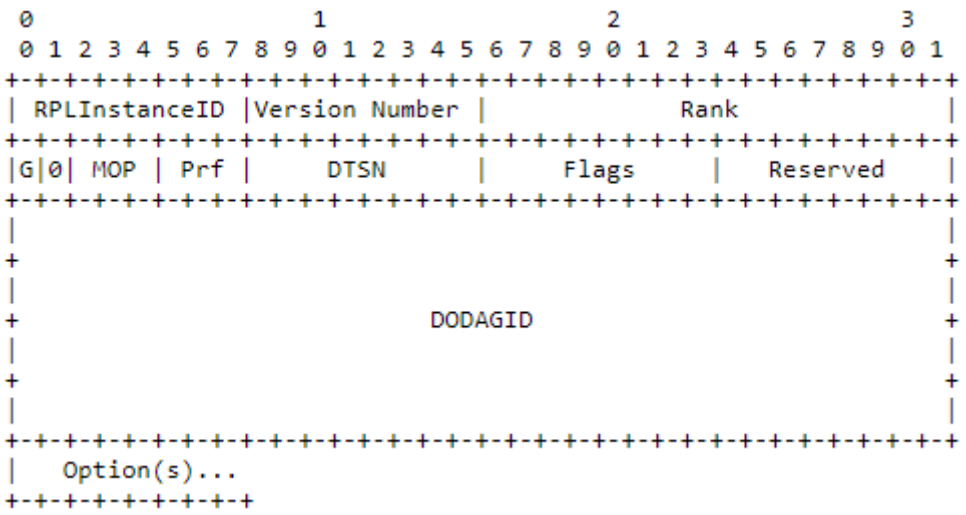


Figura 4.1: Campos de mensaje de control DIO.

El procesamiento de la información enviada en un DIO depende de si el nodo pertenecía o no al DODAG. En caso que fuera parte del DODAG, el nodo agrega el remitente como posible padre y reelige su padre preferido. Si de esta elección surge un nuevo padre, el nodo envía un DIO multicast para avisar a sus vecinos y un mensaje DAO al nuevo padre. Por otra parte, si el nodo no formara parte del DODAG, se agrega al remitente como padre preferido (comunicándoselo mediante un DAO), se crea una dirección IP (si no la tenía) y se resetea el timer de envío de mensajes DIO.

Para propagar información sobre destinos, se utiliza un *Destination Advertisement Object* o *DAO*. Este mensaje es enviado cuando se cambia de padre o se recibe un DIO de padre. Al recibirlo se debe actualizar la tabla de ruteo (en caso de trabajar en storing mode). Además, este mensaje es sucesivamente reenviado a niveles superiores hasta el root, completando las tablas de ruteos de los nodos en el camino.

4.1.3. Modos de Operación

El protocolo RPL presentado en el RFC 6550, permite cuatro modos de operación.

El primer modo de operación no da soporte a tráfico "hacia abajo" de ningún tipo. Es decir, los nodos no pueden enviar mensajes a sus hijos.

Se tiene también el modo de operación denominado *non-storing mode*. Este modo permite el tráfico hacia abajo pero sin que los nodos mantengan tablas de ruteo. En este caso, la información de ruteo viaja en los mensajes y se va actualizando en cada salto.

Luego se tiene el modo *storing mode without multicast*, que implementa tablas de ruteo actualizables en cada nodo. Sin embargo, en este modo solo es posible enviar mensajes de este tipo a un nodo en particular.

Finalmente, existe el modo *storing mode with multicast*, que a diferencia del anterior, permite también enviar mensajes multicast.

4.1.4. Rutas Hacia Arriba

En RPL se definen las rutas "hacia arriba" como los caminos que toman los mensajes en direccion (aunque no necesariamente destino) al DODAG root. Estos caminos se construyen directamente desde los nodos, cada uno eligiendo su padre preferido por el cual enviarán sus mensajes.

Luego de intercambiar mensajes DIO con su entorno, cada nodo tiene información de sus nodos cercanos, lo que se denomina el conjunto de vecinos candidatos. De este conjunto, se destaca un grupo de ellos con un valor de Rank menor o igual al del propio, lo que indica que se encuentran más cerca del root. Este grupo se le llama conjunto de candidatos a padre. De ellos, el nodo elige al menos uno (o más) como padre, y le avisa de su decisión mediante un mensaje DAO.

De esta forma, cuando el nodo necesita enviar un mensaje al root, se lo enviará al padre elegido, quien a su vez lo retransmitirá a su propio padre, sucesivamente hasta llegar al root.

4.1.5. Rutas Hacia Abajo

Las rutas "hacia abajo" se implementan gracias al uso de los mensajes DAO. Estos permiten a los nodos saber cuales nodos los tienen como padre. El manejo de estas rutas depende del modo de operación utilizado.

En el caso de usar el modo storing, todos los nodos mantienen las tablas de ruteo actualizadas, por lo que son conscientes de la topología total del DODAG. De esta forma, el nodo fuente buscará el destino en su sub-DODAG. En caso de no encontrarlo, enviará el mensaje a su padre preferido, el cual realizará la misma verificación. Este proceso se repite sucesivamente hasta encontrar un ancestro en común (pudiendo ser el root) que redirija el mensaje hacia abajo.

Si el modo de operación utilizado es non-storing, el único nodo que cuenta con tabla de ruteo es el root del DODAG, por lo que inevitablemente el mensaje debe alcanzarlo. A su vez, dado que los nodos no poseen información del grafo, el ruteo debe indicarlo el nodo root y transmitirlo a los nodos por los que tenga que pasar el mensaje. Para lograr esto, se agregan ciertos campos al encabezado del mensaje, donde se incluye información de ruteo.

4.2. RPL Lite

De acuerdo a numerosas evaluaciones respecto al rendimiento de RPL Classic, han surgido oportunidades de mejora para el protocolo. En particular, se encuentra por ejemplo que en la mayoría de las aplicaciones no se utilizan múltiples instancias de RPL, múltiples funciones objetivo, DODAGs flotantes ni locales, además de ciertas características de seguridad.

Esta situación abre el camino para una renovación del protocolo, quitando características innecesarias o prescindibles, y modificando ciertas operaciones.

4.2.1. Mensajes de Control

Para esta versión de RPL, se mantienen los mensajes de control descriptos anteriormente. La diferencia en este caso radica en los campos incluidos en ellos. Por ejemplo, dado que se elimina el uso de RPLInstanceID y DODAGID, estos campos ya no aparecen en los mensajes de control.

4.2.2. Modo de Operación

En RPL Lite se simplifica el aspecto de los modos de operación, dado que simplemente se da soporte para el modo non-storing. Esto permite una reducción importante en el uso de memoria de los nodos, dado que no deben guardar tablas de ruteo. Además, modifica el proceso de las rutas hacia abajo, mientras que las rutas hacia arriba se ven indistintas.

4.2.3. Source Routing Header

Como se explicara anteriormente, para establecer un ruteo hacia abajo en modo non-storing, el root debe hacerse cargo de las direcciones a las cuales enviar el mensaje. Dado que los nodos no guardan tablas de ruteo y solo tienen información de sus vecinos cercanos, el nodo root debe indicarles hacia donde retransmitir el mensaje recibido. Para ello se implementa el protocolo Source Routing Header, presentado en el RFC 6554 [6].

El SRH es un conjunto de campos que se agregan al encabezado de capa de red de un mensaje, con el fin de realizar el enrutamiento. Dicho enrutamiento puede ser interno al DODAG o con otros DODAG, incluso con otras instancias de RPL. Si la comunicación es interna al DODAG, el SRH se incluye en el encabezado IPv6 del mensaje original, mientras que para comunicarse con el exterior se debe implementar una tunelización IPv6-in-IPv6 e incluir el SRH en el encabezado IPv6 exterior.

Entre otras características particulares, este encabezado introduce métodos de compresión para las direcciones asociadas al camino enrutado. En concreto, las direcciones que coinciden en prefijo con la dirección indicada como destino en el encabezado IPv6 son truncadas, eliminando los octetos coincidentes. Esto resulta favorable para el caso de RPL Lite, dado que las aplicaciones donde se implementa RPL suelen tener coincidencias de prefijos.

De acuerdo al diagrama del RFC 6554, los campos del SRH son los indicados por la figura 4.2.

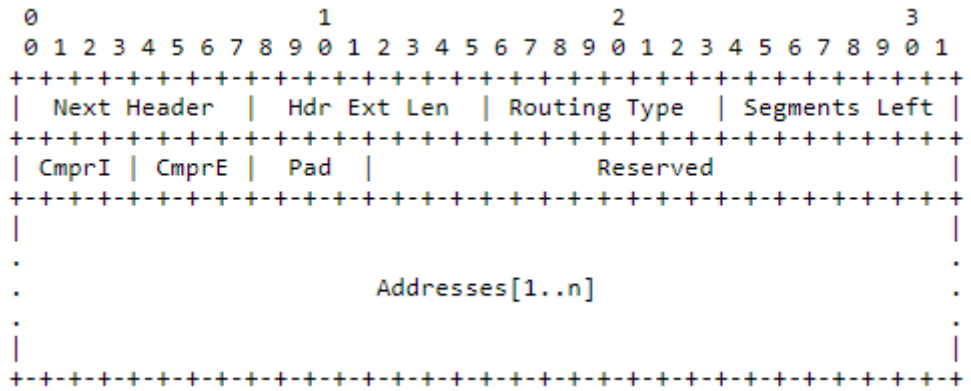


Figura 4.2: Campos SRH.

-Next Header: indica de qué tipo es el encabezado que sigue al actual.

-Hdr Ext Len: indica el largo en octetos del SRH, sin contar los primeros 8 octetos.

-Routing Type: identifica el tipo de ruteo al que pertenece el encabezado. En el caso de SRH es 3.

-Segments Left: contador que marca la cantidad de nodos por los que resta pasar.

-CmprI: indica la cantidad de octetos del prefijo de las primeras n-1 direcciones del encabezado, que son eliminados.

-CmprE: indica la cantidad de octetos del prefijo de la última dirección del encabezado, que son eliminados.

-Pad: indica la cantidad de octetos que son utilizados para relleno. Coincide con CmprI+CmprE.

-Address[1..n]: en este campo se guardan las direcciones de los nodos por los que debe saltar el mensaje para llegar a destino.

4.3. Simulaciones Cooja

Para observar el comportamiento de los protocolos de enrutamiento, se implementan dos simulaciones de Contiki-NG en la herramienta Cooja. Las mismas coinciden en cuanto a la configuración de los nodos y topología de la red, difiriendo únicamente en la versión de RPL compilada.

Se toma entonces el ejemplo "RPL-UDP" incluido en el repositorio de Contiki-NG, donde se pueden compilar nodos cliente o servidores. En nuestro caso, construiremos una red que incluya cuatro nodos cliente y un nodo servidor, que hará las veces de border router. En cuanto a la topología, los nodos serán dispuestos de forma tal que el border router tengo un único hijo, el cual sea padre de otros dos nodos y uno de ellos sea padre de un último nodo. Tal distribución se muestra en la figura 4.3.

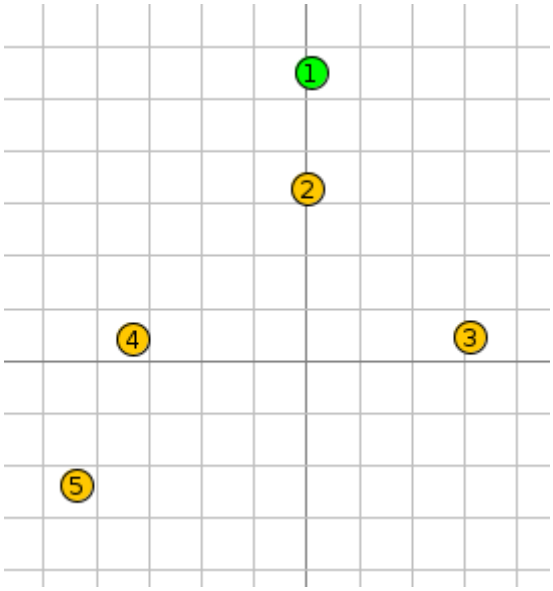


Figura 4.3: Distribución de nodos para las simulaciones.

Como último ajuste, se utiliza la herramienta "Radio messages", activando la ocpión "6LowPAN Analyzer with PCAP". Esta herramienta permite observar los mensajes intercambiados,

generando también un archivo de extensión *.pcap* que luego podrá ser analizado en profundidad por un programa analizador de protocolos. En nuestro caso dicho programa será Wireshark.

4.3.1. Formación de la Red

Dado que RPL Lite fue diseñado buscando estabilidad y rendimiento, encontramos diferencias en el proceso de formación de los grafos.

En el caso de RPL Classic, se puede observar en la figura 4.4 el intercambio de mensajes de control. La formación del grafo se inicia con el mensaje DIO que el nodo 1 (root) envía con destino multicast. Este mensaje es recibido por el nodo 2 (dado que es el único al alcance) y segundos después se dispone a enviar su DAO al nodo 1, indicándole su intención de ingresar al DODAG. Luego, es el nodo 4 el que envía su DAO al nodo 2, buscando integrarse al grafo. A partir de este mensaje DAO, el nodo 2 además de actualizar su tabla de ruteo, lo reenvía a su nodo padre. Se puede observar el mismo proceso para los nodos 3 y 5, donde a partir de un mensaje DIO se unen al DODAG a través de un mensaje DAO, que será reenviado a los nodos padres hasta el root.

Time	Source	Destination	Proto	Length	Info
0.00…	fe80::202…	ff02::1a	ICMP…	27	RPL Control (DODAG Information Solicitati
0.47…	fe80::205…	ff02::1a	ICMP…	27	RPL Control (DODAG Information Solicitati
2.44…	fe80::201…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
6.05…	fe80::202…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
6.16…	fe80::202…	fe80::201…	ICMP…	76	RPL Control (Destination Advertisement Ob
8.18…	fe80::204…	fe80::202…	ICMP…	76	RPL Control (Destination Advertisement Ob
8.22…	fe80::202…	fe80::201…	ICMP…	76	RPL Control (Destination Advertisement Ob
8.68…	fe80::204…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
8.71…	fe80::203…	fe80::202…	ICMP…	76	RPL Control (Destination Advertisement Ob
8.73…	fe80::202…	fe80::201…	ICMP…	76	RPL Control (Destination Advertisement Ob
8.77…	fe80::201…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
9.13…	fe80::203…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
9.13…	fe80::204…	fe80::202…	ICMP…	102	RPL Control (DODAG Information Object)
9.15…	fe80::204…	fe80::202…	ICMP…	102	RPL Control (DODAG Information Object)
10.5…	fe80::202…	fe80::204…	ICMP…	102	RPL Control (DODAG Information Object)
11.4…	fe80::203…	fe80::202…	ICMP…	102	RPL Control (DODAG Information Object)
12.1…	fe80::205…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
12.4…	fe80::202…	fe80::201…	ICMP…	76	RPL Control (Destination Advertisement Ob
13.0…	fe80::205…	fe80::204…	ICMP…	76	RPL Control (Destination Advertisement Ob
13.0…	fe80::204…	fe80::202…	ICMP…	76	RPL Control (Destination Advertisement Ob
13.1…	fe80::204…	fe80::202…	ICMP…	76	RPL Control (Destination Advertisement Ob
13.1…	fe80::202…	fe80::201…	ICMP…	76	RPL Control (Destination Advertisement Ob

Figura 4.4: Mensajes de control RPL Classic en la formación del grafo.

En el caso de RPL Lite, si bien el proceso de formación del grafo es similar, requiere más pasos. En general, dado que se busca encontrar una mejor estabilidad de los enlaces, los nodos intercambian varios mensajes DIO antes de elegir un padre. Esto se puede observar en las figuras 4.5, 4.6 y 4.7.

Time	Source	Destination	Proto	Length	Info
0.00…	fe80::201…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
1.81…	fe80::202…	fe80::201…	ICMP…	102	RPL Control (DODAG Information Object)
2.71…	fe80::202…	fe80::201…	ICMP…	102	RPL Control (DODAG Information Object)
5.35…	fe80::202…	fe80::201…	ICMP…	102	RPL Control (DODAG Information Object)
6.00…	fe80::202…	fe80::201…	ICMP…	102	RPL Control (DODAG Information Object)
6.63…	fe80::201…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
9.67…	::202:2:2…	::201:1:1…	ICMP…	85	RPL Control (Destination Advertisement Ob
9.69…	::201:1:1…	::202:2:2…	ICMP…	43	RPL Control (Destination Advertisement Ob

Figura 4.5: Integración del nodo 2 al grafo en RPL Lite.

Time	Source	Destination	Proto	Length	Info
12.3…	fe80::202…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
15.4…	fe80::203…	fe80::202…	ICMP…	102	RPL Control (DODAG Information Object)
15.6…	fe80::204…	ff02::1a	ICMP…	27	RPL Control (DODAG Information Solicitati
16.2…	fe80::204…	fe80::202…	ICMP…	102	RPL Control (DODAG Information Object)
18.0…	fe80::204…	fe80::202…	ICMP…	102	RPL Control (DODAG Information Object)
18.0…	fe80::204…	fe80::202…	ICMP…	102	RPL Control (DODAG Information Object)
19.1…	fe80::203…	fe80::202…	ICMP…	102	RPL Control (DODAG Information Object)
19.3…	fe80::202…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
21.9…	fe80::204…	fe80::202…	ICMP…	102	RPL Control (DODAG Information Object)
22.8…	fe80::201…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
22.9…	fe80::203…	fe80::202…	ICMP…	102	RPL Control (DODAG Information Object)
25.5…	fe80::203…	fe80::202…	ICMP…	102	RPL Control (DODAG Information Object)
27.1…	fe80::205…	ff02::1a	ICMP…	27	RPL Control (DODAG Information Solicitati
27.4…	fe80::202…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
27.8…	::204:4:4…	::201:1:1…	ICMP…	93	RPL Control (Destination Advertisement Ob
27.8…	::204:4:4…	::201:1:1…	ICMP…	94	RPL Control (Destination Advertisement Ob
27.8…	::201:1:1…	::204:4:4…	ICMP…	51	RPL Control (Destination Advertisement Ob
27.8…	::201:1:1…	::204:4:4…	ICMP…	60	RPL Control (Destination Advertisement Ob
30.5…	fe80::204…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
31.4…	::203:3:3…	::201:1:1…	ICMP…	93	RPL Control (Destination Advertisement Ob

Figura 4.6: Integración de los nodos 3 y 4 al grafo en RPL Lite.

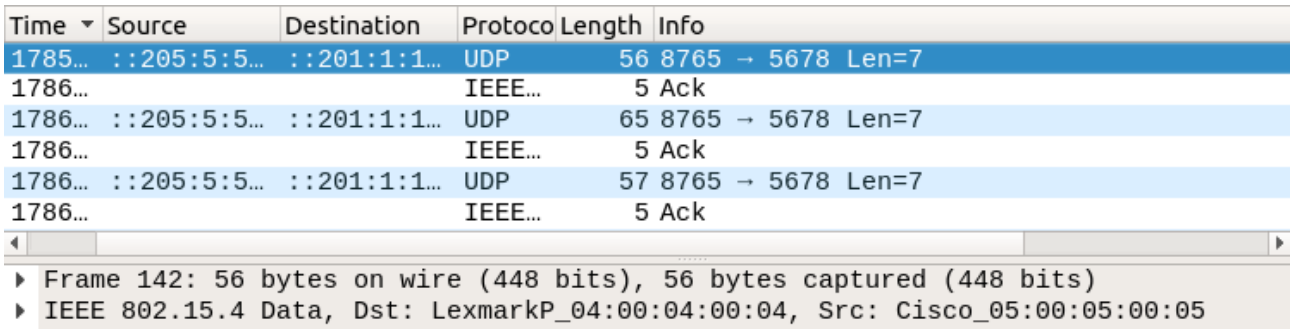
Time	Source	Destination	Proto	Length	Info
36.1…	fe80::204…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
39.0…	fe80::205…	fe80::204…	ICMP…	102	RPL Control (DODAG Information Object)
41.3…	fe80::205…	fe80::204…	ICMP…	102	RPL Control (DODAG Information Object)
41.8…	fe80::203…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
43.4…	fe80::202…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)
44.5…	::205:5:5…	::201:1:1…	ICMP…	93	RPL Control (Destination Advertisement Ob
44.5…	::205:5:5…	::201:1:1…	ICMP…	102	RPL Control (Destination Advertisement Ob
44.5…	::205:5:5…	::201:1:1…	ICMP…	94	RPL Control (Destination Advertisement Ob
44.6…	::201:1:1…	::205:5:5…	ICMP…	59	RPL Control (Destination Advertisement Ob
44.6…	::201:1:1…	::205:5:5…	ICMP…	68	RPL Control (Destination Advertisement Ob
44.6…	::201:1:1…	::205:5:5…	ICMP…	68	RPL Control (Destination Advertisement Ob
47.1…	fe80::205…	ff02::1a	ICMP…	97	RPL Control (DODAG Information Object)

Figura 4.7: Integración del nodo 5 al grafo en RPL Lite.

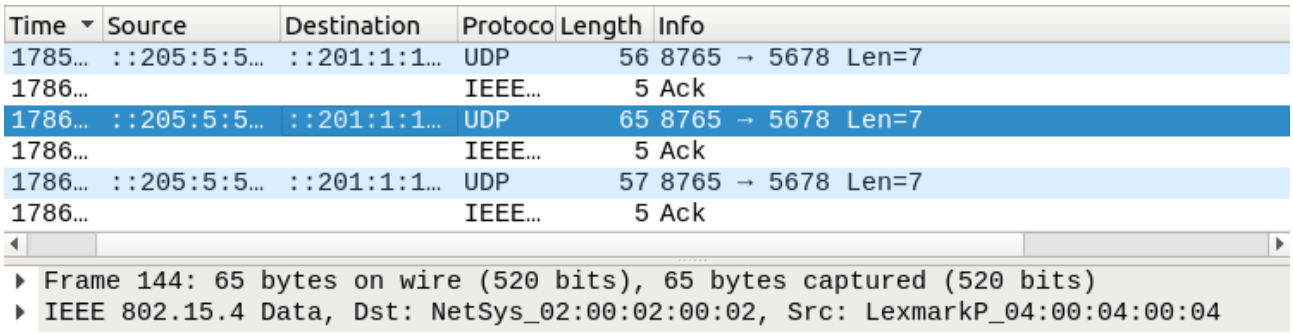
4.3.2. Rutas Hacia Arriba

En el caso de los mensajes enviados hacia arriba, ambas versiones del protocolo RPL funcionan de la misma manera. Esto es debido a que se basan en el sistema de nodos padres

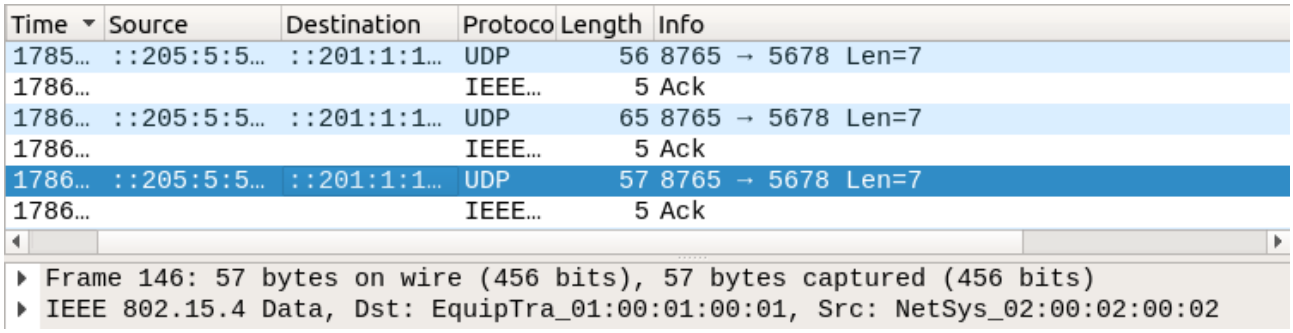
que implementan ambos. Cuando un nodo quiere comunicarse con el root o con otro nodo en posición de ancestro respecto al nodo fuente, simplemente envía su mensaje a su nodo padre, el cual lo reenviará al suyo, y así sucesivamente hasta llegar a destino. Este proceso se puede observar en las siguientes figuras, donde en el encabezado de capa de enlace vemos el cambio de dirección de destinatario.



(a) Salto desde el nodo 5 al nodo 4.



(b) Salto desde el nodo 4 al nodo 2.



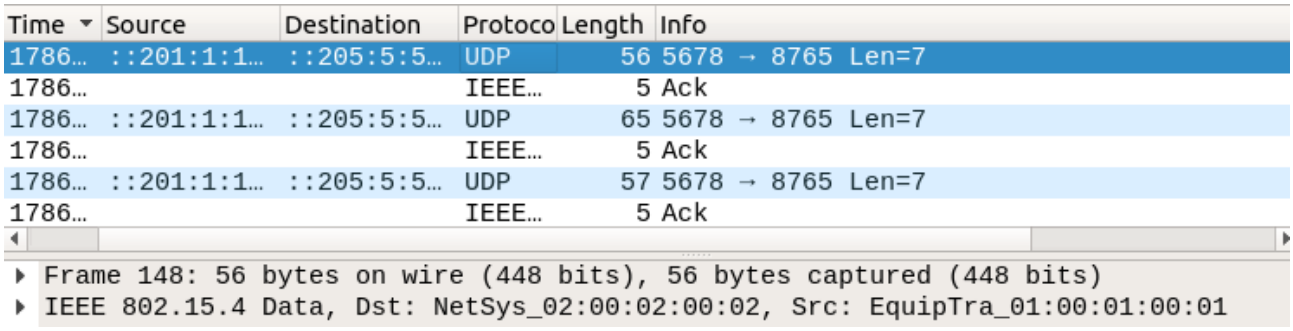
(c) Salto desde el nodo 2 al nodo 1.

Figura 4.8: Mensaje UDP enviado desde el nodo 5 con destino al nodo 1.

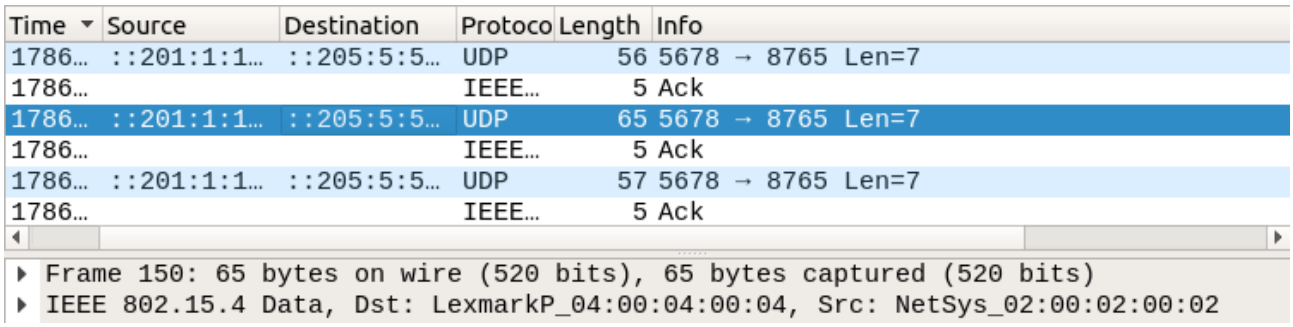
4.3.3. Rutas Hacia Abajo

Cuando los mensajes son enviados hacia abajo, se plantea una diferencia entre RPL Classic y RPL Lite.

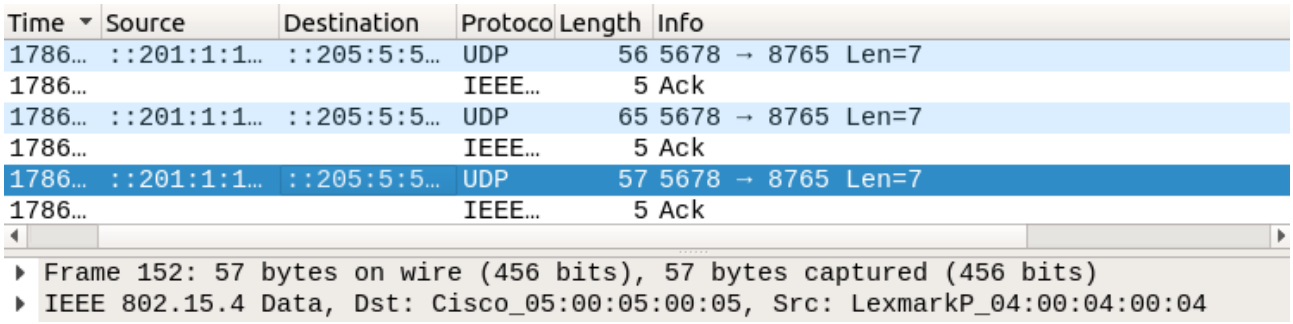
En el caso del primero, el hecho de que cada nodo mantenga su tabla de ruteo permite tener la información suficiente para que el nodo sepa hacia donde enrutar el mensaje. De esta forma cada nodo reenvía el mensaje recibido en la dirección correspondiente, llegando a su destino de forma correcta. Se observa en la figura 4.9 que el proceso es similar al enrutamiento hacia arriba. La diferencia es que en este caso los nodos no mandan su mensaje a su padre preferido, sino que lo mandan en función de su tabla de ruteo.



(a) Salto desde el nodo 1 al nodo 2.



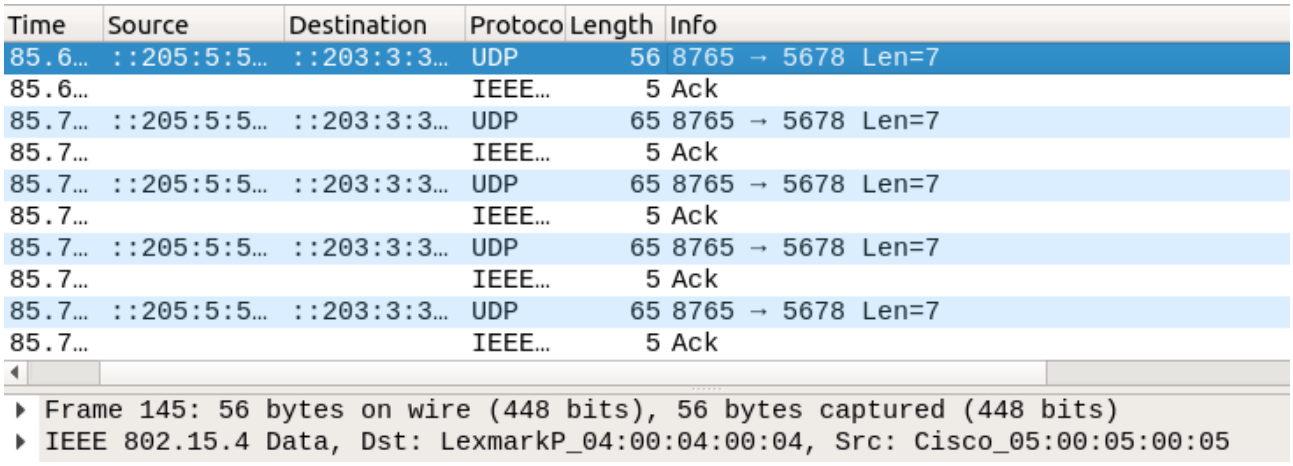
(b) Salto desde el nodo 2 al nodo 4.



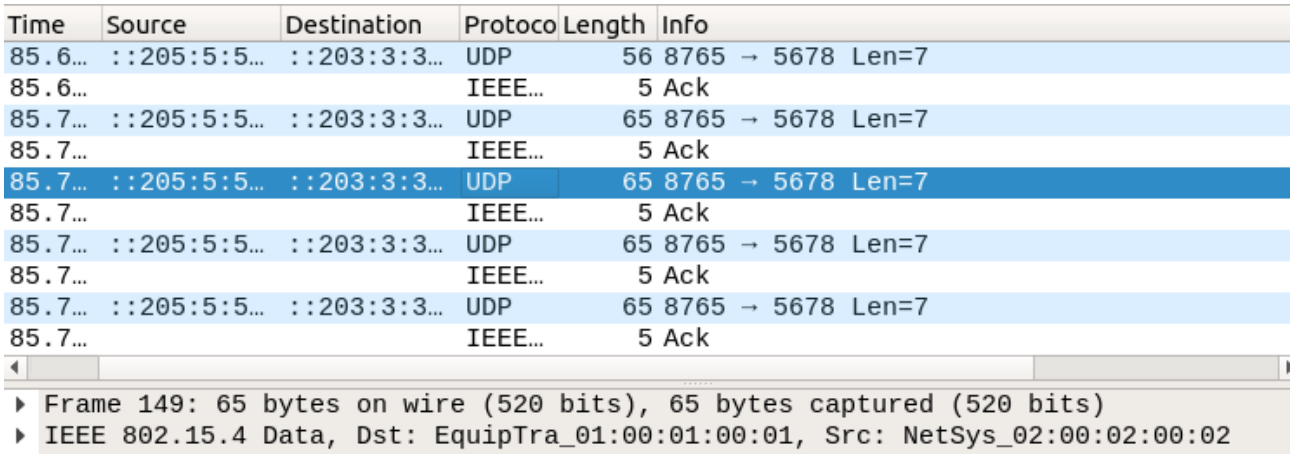
(c) Salto desde el nodo 4 al nodo 5.

Figura 4.9: Mensaje UDP enviado desde el nodo 1 con destino al nodo 5.

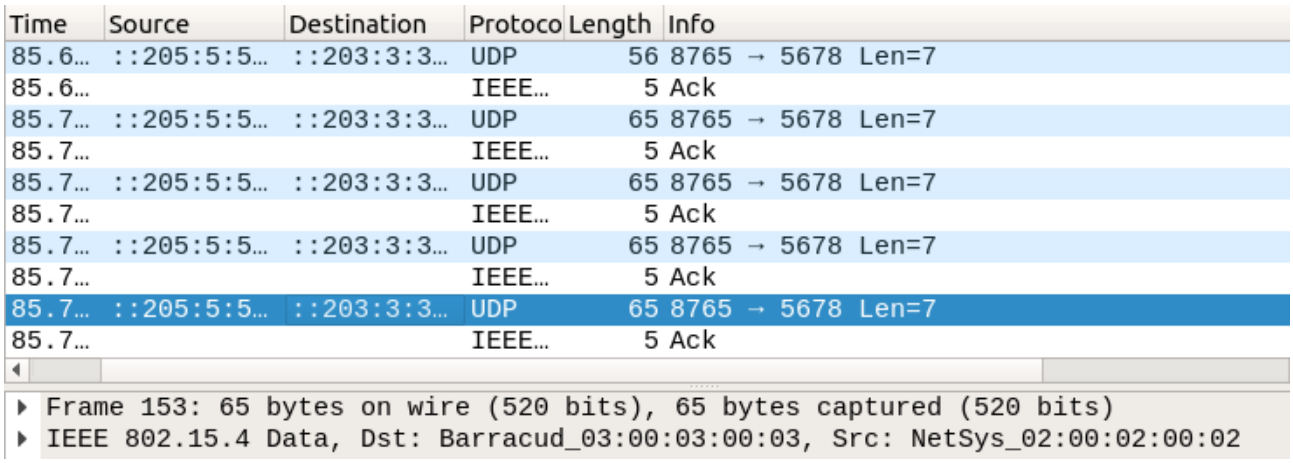
Por otra parte, en el protocolo RPL Lite, solo el nodo root mantiene tablas de ruteo, por lo que es quien debe realizar el enrutamiento. Para ello el mensaje debe enviarse hasta este nodo en primer lugar, para luego descender hasta el nodo deseado. Esta situación se verifica en la figura 4.10, donde el nodo 5 pretende comunicarse con el nodo 3. Se observan entonces los mensajes reenviados desde el 5 hasta el 1 (pasando por el 4 y el 2), y luego descendiendo hasta el 3.



(a) Salto desde el nodo 5 al nodo 4.



(b) Salto desde el nodo 2 al nodo 1.



(c) Salto desde el nodo 2 al nodo 3.

Figura 4.10: Mensaje UDP enviado desde el nodo 5 con destino al nodo 3.

En este caso podemos dividir el proceso en dos partes: los mensajes hacia arriba (primeros tres) y los mensajes hacia abajo (últimos dos).

Para los tres primeros mensajes, rige el uso de nodos padres como forma de enrutar el mensaje, como se explicara anteriormente.

Para enviar los mensajes hacia abajo el root utiliza el SRH, indicando por cuales nodos debe pasar el mensaje. En la figura 4.11, se observa dicho encabezado, donde ya habiendo llegado al

nodo 2, se retira su dirección del mismo, quedando las direcciones siguientes. En este caso se observa únicamente la dirección del nodo 3, el último salto del mensaje para llegar a destino. Se puede verificar también en el campo Segments Left que al mensaje le queda un solo salto, correspondiendo con la cantidad de direcciones restantes en el encabezado.

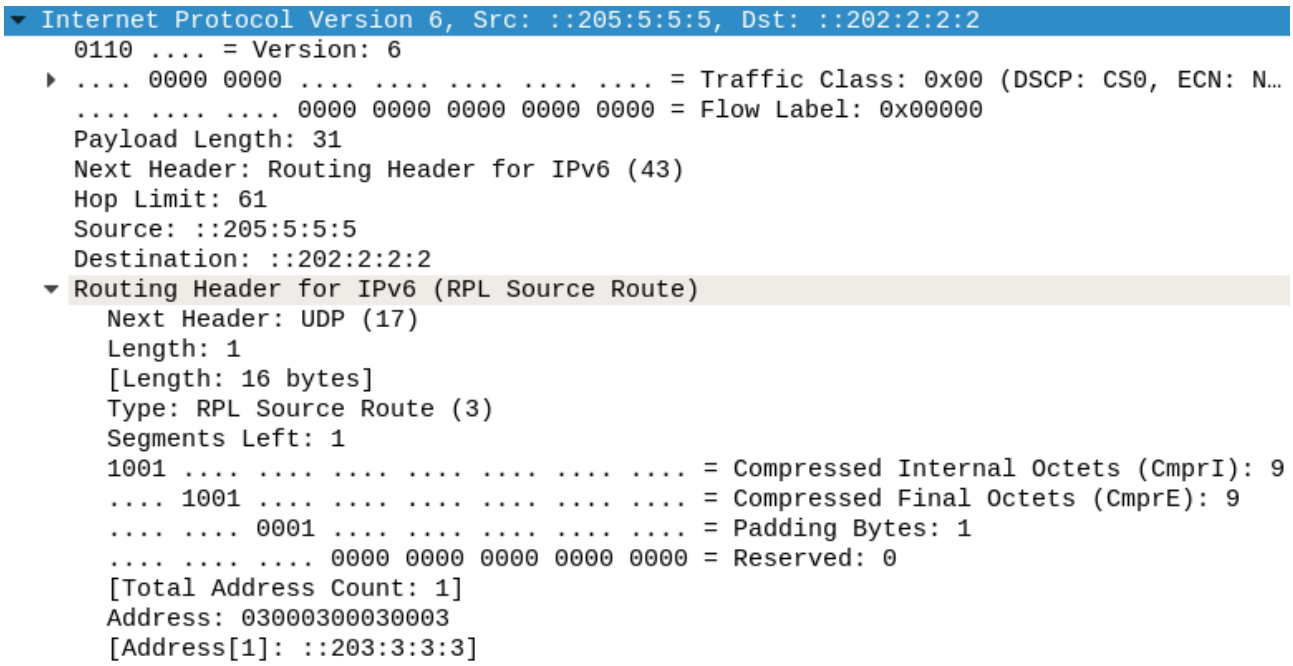


Figura 4.11: Detalle de Source Routing Header para envío de mensajes hacia abajo.

En el ejemplo de la figura 4.12 se tiene una situación similar. En este caso el nodo 1 pretende enviar un mensaje al nodo 5, por lo que en el mensaje recibido por el nodo 2 se tiene un SRH con las direcciones de los nodos 4 y 5. Se observa también que el campo Segments Left indica 2 saltos restantes.

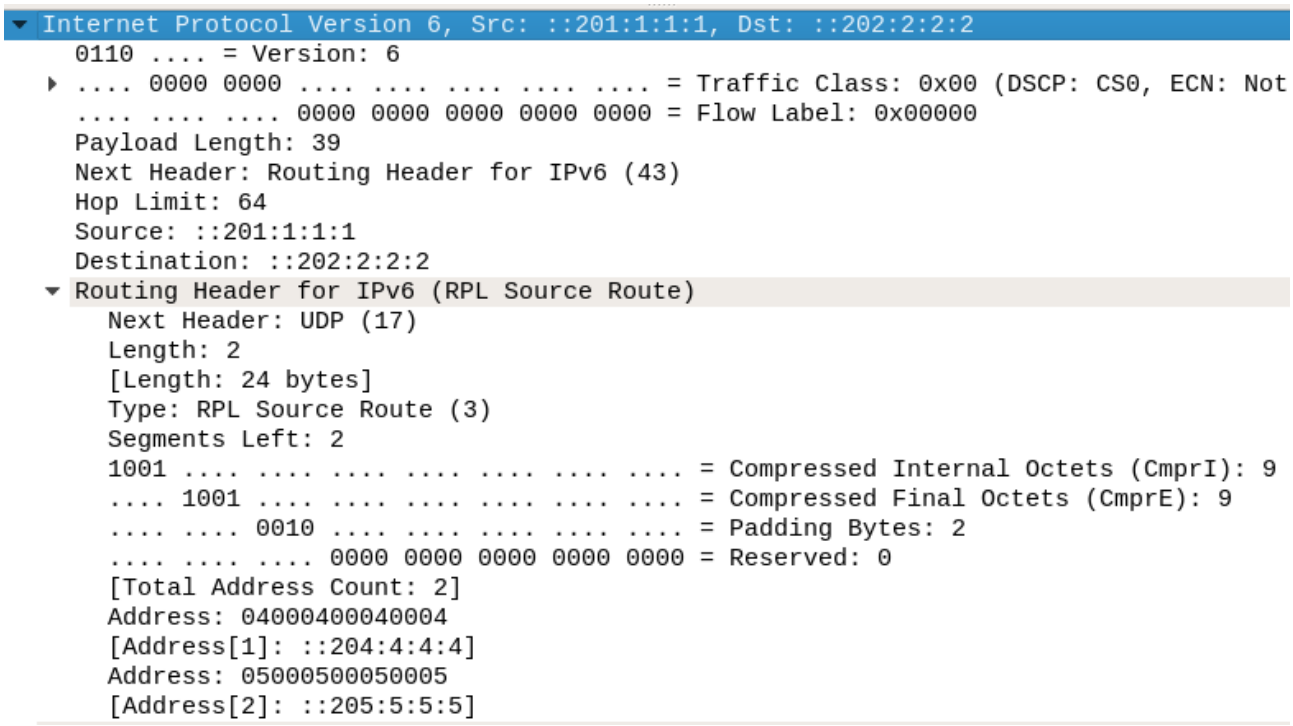


Figura 4.12: Otro ejemplo de detalle de Source Routing Header.

4.4. Comparación Final

Habiendo verificado las principales diferencias entre ambas versiones del protocolo RPL, podemos establecer ciertas conclusiones respecto a sus ventajas, desventajas y aplicaciones óptimas.

En primer lugar, podemos decir que RPL Lite es una versión mejorada de RPL Classic, en la que se profundizan conceptos como estabilidad de enlaces, sencillez de implementación y simplicidad de funcionalidades. A estas metas acompañan los resultados de menor huella de memoria ROM y mejor rendimiento. Vale aclarar que estos resultados son a costa de una peor interoperabilidad con otras implementaciones, que por ejemplo podrían necesitar el uso del modo storing.

A niveles prácticos, encontramos que RPL Lite deslinda a los nodos de la necesidad de mantener y actualizar tablas de ruteo, lo que redunda en un menor uso de memoria RAM y de procesamiento. Vimos también que RPL Classic utiliza menos cantidad de mensajes de control para formar sus redes, traduciéndose en menos tiempo de formación de la red y menor consumo (por menos cantidad de transmisiones). Por último, al no tener tablas de ruteo en todos los nodos, RPL Lite utiliza Source Routing Header para el enrutamiento de los mensajes, lo que implica paquetes más grandes, mayor procesamiento y mayor consumo.

Podemos asegurar entonces que ambas versiones son útiles y beneficiosas, dependiendo de la aplicación a implementar. Se puede asegurar que existe una relación de *trade-off* entre consumo y memoria (tanto ROM como RAM). Si las restricciones son mayores respecto al consumo, sería conveniente el uso de RPL Classic, mientras que si los nodos en cuestión se encuentran más limitados en cuanto a memoria, lo mejor sería implementar RPL Lite. Otro aspecto a incluirse en la decisión es la cuestión de la interoperabilidad, para lograr una buena combinación con el resto del *stack* de red.

5. Aplicación

Como se describe en la sección 1.1, parte del objetivo de este proyecto es implementar un sistema de monitoreo. A nivel de capa de aplicación se trabajó con el protocolo CoAP, creando 2 servidores CoAP. La aplicación final es un software tipo Web-based (Aplicación Web), similar a un SCADA o web de dashboards, donde se visualiza las variables monitoreadas. Para integrar la capa de aplicación con la aplicación final fue necesario utilizar un Border Router y crear un cliente CoAP en Python. Además, se debió desarollar un driver para uno de los sensores.

Otro de los objetivos planteados era utilizar el protocolo DTLS para CoAP, conformando lo que se conoce como CoAPs. En la documentación de Contiki-NG, se presenta que la implementación actual soporta tanto CoAP como CoAPs[7]. Sin embargo, tras investigar en internet como utilizar DTLS, encontramos varios problemas recientes debido a que las últimas modificaciones que se han realizado en Contiki-NG no se han integrado a la librería de DTLS, por lo que se puede concluir que la implementación actual de CoAP no soporta DTLS.

Los códigos modificados o desarrolados, se pueden encontrar en el siguiente repostorio: https://gitlab.fing.edu.uy/carlos.menoni/proyecto-rsi-grupo-1

5.1. Arquitectura del sistema

En la figura 5.1, se puede apreciar la arquitectura del sistema. El nodo 1, representa el sensado de un nodo ubicado en una sala (Sala Vegetativo), mientras que el nodo 2 representa estar ubicado en otra sala (Sala Floración). Se utilizaron dos sensores en cada nodo: DHT11 para humedad y temperatura ambiente, y SEN0193 para humedad de suelo.

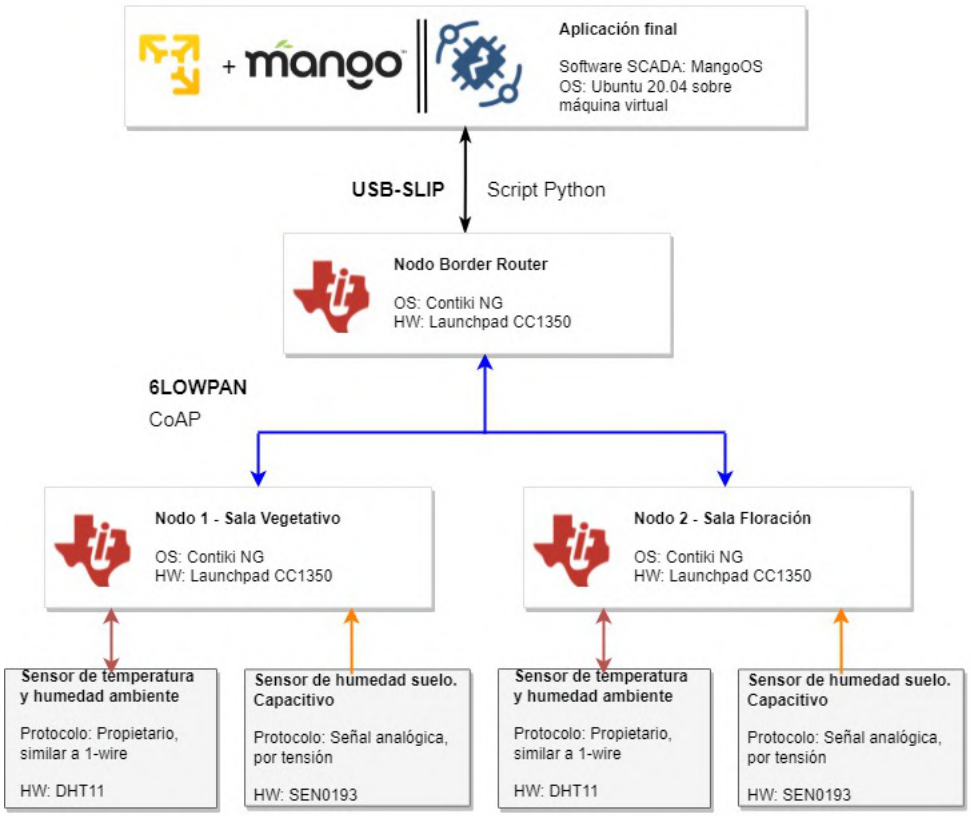


Figura 5.1: Arquitectura del sistema

Estos nodos y el Border Router (BR) forman la red inalámbrica en 6LowPAN. El BR se comunica con la computadora con máquina virtual Ubuntu, mediante la herramienta *tunslip6* de Contiki-NG, utilizando el puerto USB.

Sobre la máquina virtual se armaron dos servidores web locales, Mango OS y ThingsBoard. Estas aplicaciones web son del tipo SCADA o Dashboard, en este caso, se utilizaron para visualizar las variables monitoreadas. Se desarrollaron dos scripts en Python (uno para cada servidor) para lograr crear una pasarela de datos entre la aplicación final y los nodos servidores CoAP.

5.2. Drivers

Para que los nodos logren monitorear los sensores, además de conectar los mismos a la placa fue necesario aprender a utilizar los drivers que proporciona Contiki-NG y desarrollar uno de ellos.

Para el sensor DHT11 de humedad y temperatura ambiente, Contiki-NG cuenta con un driver ya desarrollado y un ejemplo. Antes de solicitar los valores de lectura, se debe configurar el driver para utilizar el puerto y pin de la GPIO al cual se conectó el sensor, además de agregar el módulo al Makefile y activar el sensor.

Si bien no se cuenta con un driver para el sensor analógico SEN0193, sí se tiene uno para utilizar el ADC como sensor. Este driver utiliza el ADC en su tensión de refencia normal ($4,3\ V$) y devuelve en microvolts la tensión sensada en el canal. Para utilizarlo es necesario activar el sensor y configurar el channel del ADC. A partir del driver del ADC se desarrolla uno para el sensor SEN0193. Este driver permite configurar el channel y devuelve un valor que se corresponde con la humedad del suelo en porcentaje. multiplicada por 10. Para calcular este ajuste, se utilizaron datos de un experimento realizado para el proyecto de fin de carrera.

Para validar la lectura de los sensores, se desarrolló un código de ejemplo que realiza la lectura e imprime el valor recibido, similar al desarrollado en el laboratorio 1 del curso.

5.3. Nodos - Servidores CoAP

Los nodos servidores tienen como archivo main el código ejemplo “coap-example-server”, proporcionado por Contiki-NG.

Se agregaron los siguientes recursos:

- res-dht11: Recurso que responde request de GET, con la lectura de temperatura (^{o}C) y humedad ambiente (%).
- res-sen0193: Responde request de GET, con la humedad del suelo (%).
- res-unix-time: Se puede solicitar y setear el tiempo, en unidades de segundos. Se migró el recurso desarrollado en los laboratorios de RSI a Contiki-NG. Este recurso no se utiliza actualmente para la aplicación final, pero tiene el propósito de servir para indicar el timestamp o para monitorear los nodos.

De forma predeteminada se envía la lectura en formato JSON, aunque también se implementaron texto plano y XML, tal como los otro recursos implementados por Contiki-NG.

Para validar el funcionamiento de los recursos desarrollados, utilizamos la extensión Copper, donde fácilmente podemos enviar mensajes GET y POST a los servidores.

5.4. Border Router

Para el nodo Border Router, se utilizó el código de ejemplo "rpl-border-router" []. Tal como se realizó en los laboratorios, nos conectamos por tunslip6 al BR para lograr acceder a la red inalámbrica.

5.5. Aplicación final

Como aplicación final se probaron dos aplicaciones web locales, Mango OS[8] y ThingsBoard[9].

5.5.1. Mango OS

Esta aplicación no tiene implementada la comunicación mediante protocolo CoAP. Por lo tanto fue necesario desarrollar un script en Python con el propósito de crear una pasarela de datos entre los servidores CoAP y Mango OS. El script cumple la función de CoAP client, donde solicita las lecturas mediante mensajes GET, guarda los valores y crea un servidor Modbus TCP/IP, donde Mango OS se conecta y obtiene los valores de lectura. Para desarrollar este script se utilizó la librería *aiocoap*[10] y *pyModbusTCP*[11].

El resultado final se observa en la gráfica de la figura 5.2, generada en Mango OS con los valores de lectura obtenidos.

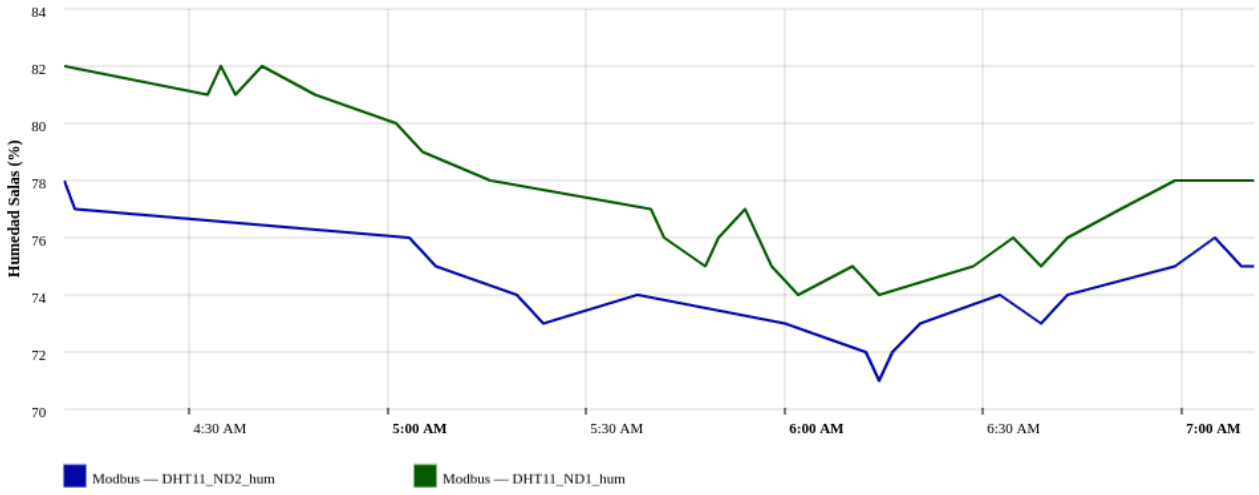


Figura 5.2: Comparación humedad ambiente medida por sensores DH11

5.5.2. ThingsBoard

También se utilizó la aplicación web local ThingsBoard. Esta aplicación soporta CoAP con mensajes JSON, pero cumple el rol de CoAP Server. Por lo tanto, fue necesario desarrollar un script en Python que envie mensajes GET a los nodos servidores y envíe mensajes POST a ThingsBoard con el valor recibido. Para realizar este CoAP client, se utilizó la librería *aiocoap*.

Se desarrollaron 3 pantallas, una para cada sala y una de comparación. En la figura 5.3, se presenta una de las pantallas.

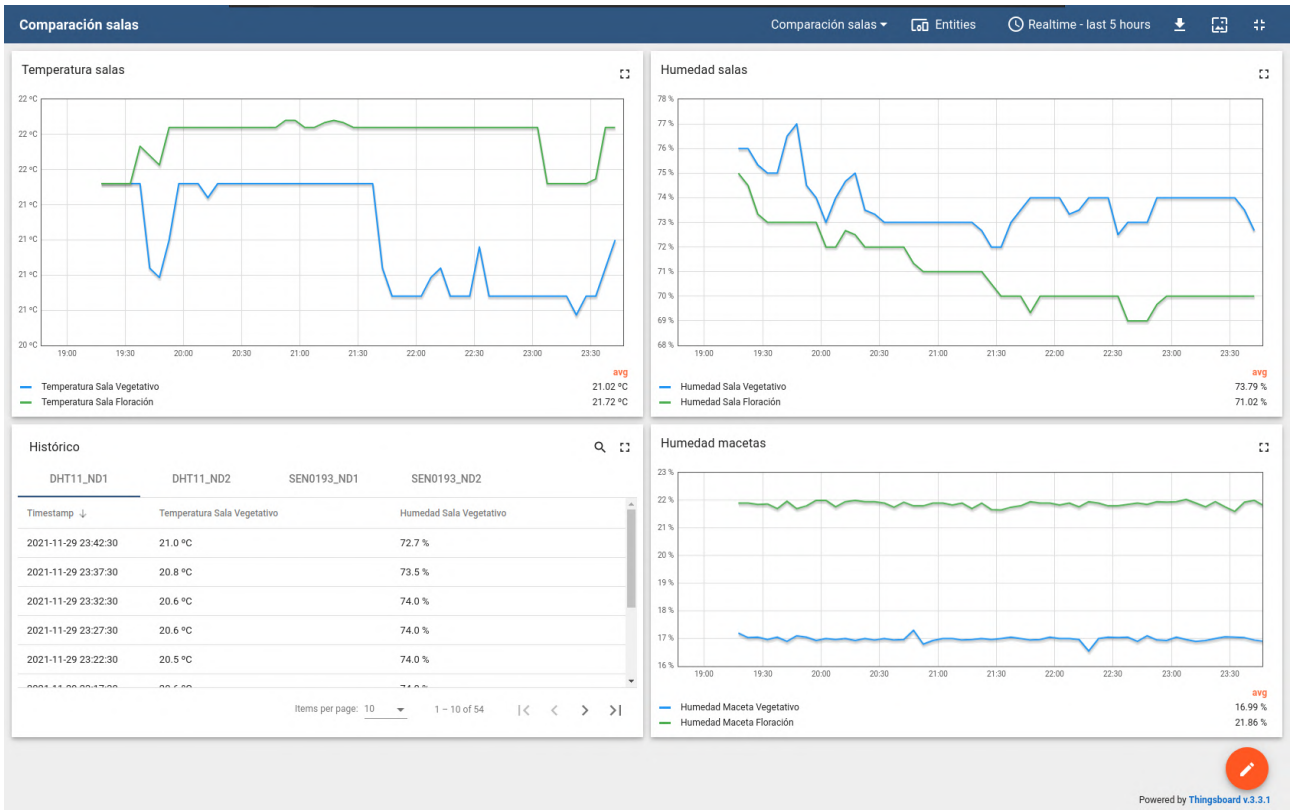


Figura 5.3: Comparación sensores de ambas salas

6. Conclusiones del Proyecto

Una vez finalizado el proyecto encontramos que pudimos completar los objetivos trazados al principio e incluir todo el alcance propuesto. En tal sentido, se logra implementar una red de sensores inalámbricos con el OS Contiki-NG, implementando una red con enrutamiento RPL, tanto en su versión Classic como Lite. Además se completó la pila de red desarrollando la aplicación CoAP y utilizar un SCADA para la publicación de información en dashboards editables.

Más en particular, se implementaron dos versiones de protocolo de enrutamiento RPL. Se analizaron las características de ambas, encontrando diferencias que permitieron describir las aplicaciones óptimas de cada una.

A nivel de los sensores, se elaboraron drivers a partir de ejemplos para manejar los mismos desde los nodos, tomando datos y comunicándolos al concentrador correspondiente.

Se trabajó también con dos opciones de SCADA, ThingsBoard y MangoOS. Esto permitió estudiar sus diferencias y elegir la mejor opción para el Proyecto de Fin de Carrera asociado.

En cuanto a la ejecución del proyecto, encontramos que si bien hubieron etapas de mayor o menor avance, se pudo cumplir a grandes rasgos con la planificación planteada. En particular, encontramos una demora importante que no esperabamos, al momento de preparar el entorno de trabajo, así como superamos nuestras expectativas al desarrollar los drivers de los sensores, gracias a los ejemplos encontrados en el repositorio de Contiki-NG.

Finalmente, valoramos positivamente haber afirmado la mayoría de los conceptos dados en el curso y profundizado algunos de ellos. Creemos que este proyecto fue de gran ayuda para complementar el Proyecto de Fin de Carrera, tanto por su contenido técnico como por la experiencia en gestión y trabajo.

7. Bibliografía

Referencias

[1] A. Liñán, A. Vives, M. Zennaro, A. Bagula, and E. Piertosemmoli, *IoT in Five Days*, 2016. [Online]. Available: https://github.com/marcozennaro/IPv6-WSN-book/raw/master/Releases/IoT%20in%20five%20days%20-%20v1.1%2020160627.pdf

[2] G. Oikonomou et. al. (2020, December) Contiki-ng documentation. [Online]. Available: https://github.com/contiki-ng/contiki-ng/wiki

[3] F. Lima, J. Navarro, and M. Porto, "BeeSmart," Proyecto de fin de Carrera, Universidad de la República (Uruguay). Facultad de Ingeniería., 2021. [Online]. Available: https://iie.fing.edu.uy/publicaciones/2020/LNP20/LNP20.pdf

[4] R. Alexander, A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, and T. Winter, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, Mar. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6550.txt

[5] A. Parasuram, "An Analysis of the RPL Routing Standard for Low Power and Lossy Networks," Master of Science in Computer Science, University of California, Berkeley., 2016. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-106.html

[6] D. Culler, J. Hui, J. Vasseur, and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)," RFC 6554, Mar. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6554.txt

[7] Contiki Developers. (2018, Mayo) Documentation: Coap. [Online]. Available: https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-CoAP

[8] Mango OS Developers. Web page. [Online]. Available: https://mango-os.com/

[9] Thingsboard Developers. Web page. [Online]. Available: https://thingsboard.io/

[10] Aiocoap Developers. Web page. [Online]. Available: https://aiocoap.readthedocs.io/en/latest/api.html

[11] pyModbusTCP Developers. Web page. [Online]. Available: https://pypi.org/project/pyModbusTCP/

[12] Contiki-NG Developers. (2020, December) Contiki-ng toolchain installation on linux. [Online]. Available: https://github.com/contiki-ng/contiki-ng/wiki/Toolchain-installation-on-Linux

[13] Texas Instruments. Ccs v11 linux host support. [Online]. Available: https://software-dl.ti.com/ccs/esd/documents/ccsv11_linux_host_support.html

[14] Texas Instruments. Descargar ccs v11. [Online]. Available: https://www.ti.com/tool/download/CCSTUDIO

[15] Texas Instruments. Xds emulation software. [Online]. Available: https://software-dl.ti.com/ccs/esd/documents/xdsdebugprobes/emu_xds_software_package_download.html

[16] ——. Developing with contiki-ng in ccs. [Online]. Available: https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/158/contiki_2D00_ccs_2D00_debugging.pptx

8. Anexo

8.1. Entorno de Trabajo

En esta sección se explicará como instalar Contiki-NG, Code Composer Studio (CCS) y como utilizar la heramienta Debug de CCS. Este manual se realizó sobre una máquina virtual con Ubuntu 20.04 64 bits.

8.1.1. Instalar Contiki-NG

Antes de descargar el repositorio de Contiki-NG, se recomienda instalar las herramientas recomendadas por los desarrolladores. Contiki-NG cuenta con un tutorial para instalar estas herramientas y para descargar el repositorio [12].

En el primer paso, se recomienda cambiar la librería python-serial, por python3-serial.

El resto del tutorial se puede seguir paso a paso, y no ocurrirá ningún problema.

8.1.2. Instalar Code Composer Studio

Primero se debe seguir la guía desarrollada por Texas Instrument, para instalar las librerías necesarias para CCS [13]. Se debe descargar CCS v11 [14] y extraer los archivos comprimidos. Luego, en el archivo de setup de extensión *.run*, se debe entrar en las propiedades y habilitar la casilla descrita como "Allow executing file as program". Esto se puede ver en la figura 8.1

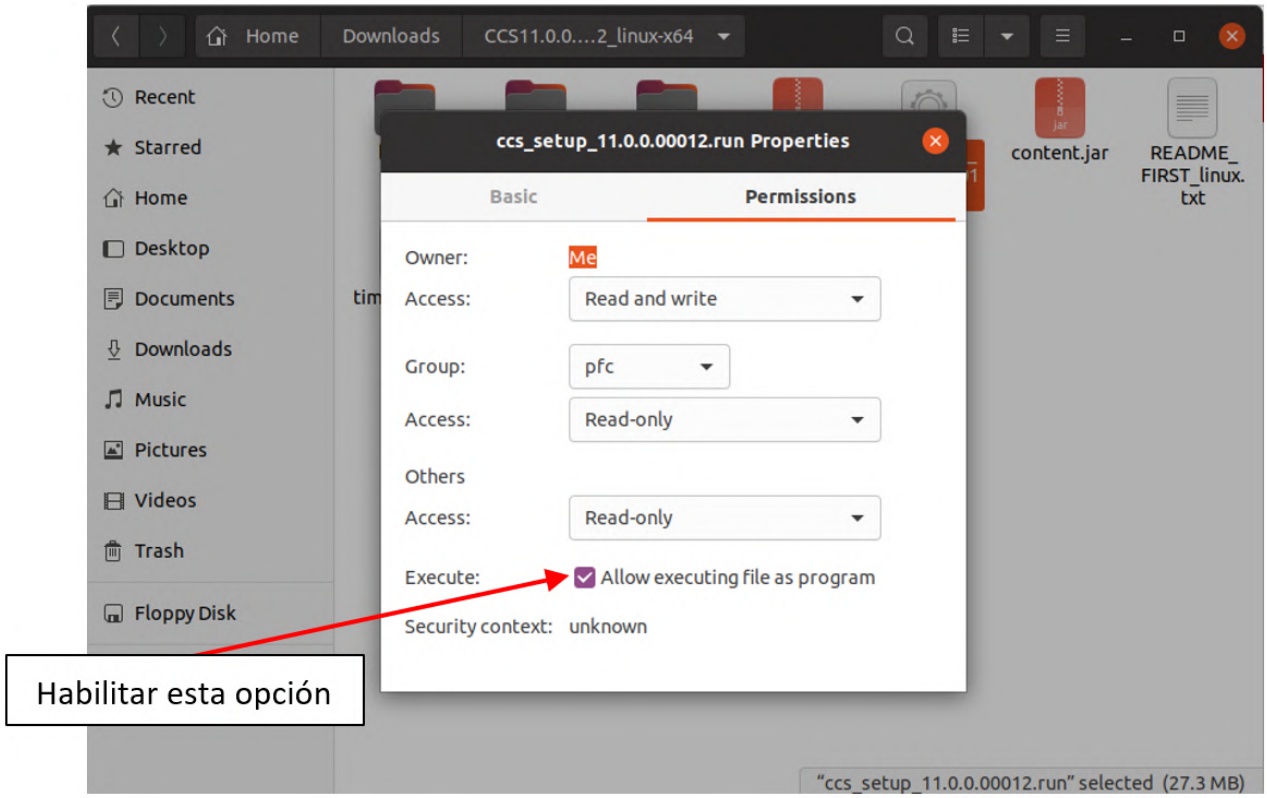


Figura 8.1: Propiedades ccs setup

Luego se podrá abrir el instalador dando doble click sobre el archivo. Al avanzar en la instalación, si se instalaron todas las herramientas necesarias para CCS, debería aparecer la siguiente pantalla (figura 8.2).

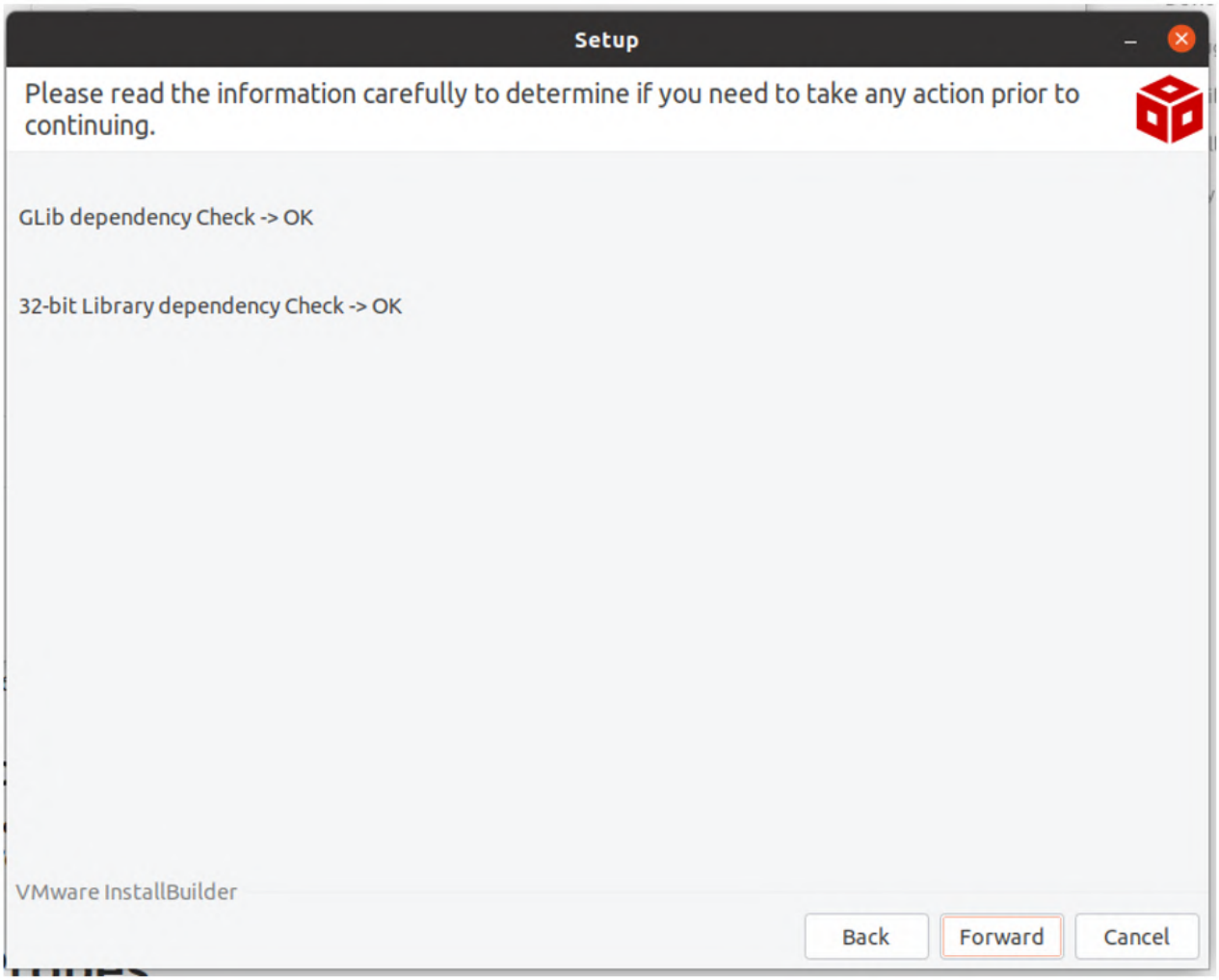


Figura 8.2: Pantalla de verificación de herramientas necesarias

En nuestro caso particular, solamente instalamos el paquete "Simplelink CC13xx and CC26xx Wireless MCU" y "Spectrum Digital Debug Probes and Boards", ya que contamos con un launchpad del SoC CC1350.
Recordar que luego de instalar CCS, se deben instalar los drivers, tal como se indica en el instructivo de Texas.
Para lograr conectarlos al launchpad CC1350, debemos instalar el programa XDS Emulation Software, ya que el Launchpad CC1350 cuenta con un XDS110 embebido. Para esto debemos descargar XDS Emulation Software [15]. Al avanzar en la instalación, elegir la carpeta /{CARPETA DE INSTALACIÓN CCS}/ccs1100/ccs/. Al finalizar la instalación, se recomienda reiniciar el sistema operativo.

Ejecutar CCS y abrir CCS App Center, como se indica en la figura 8.3.

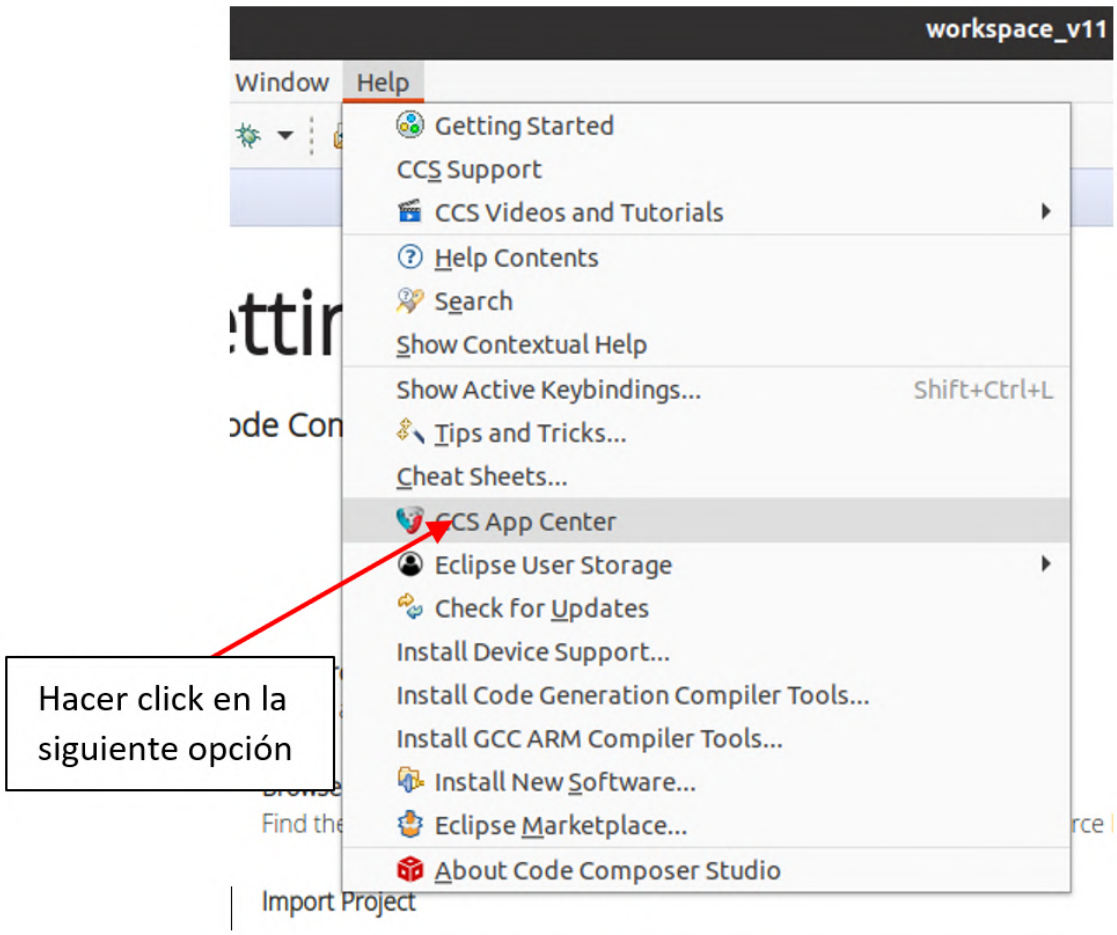


Figura 8.3: Abrir CCS App Center

Instalar TI ARM Compiler en los Add-ons, como se indica en la figura 8.4.

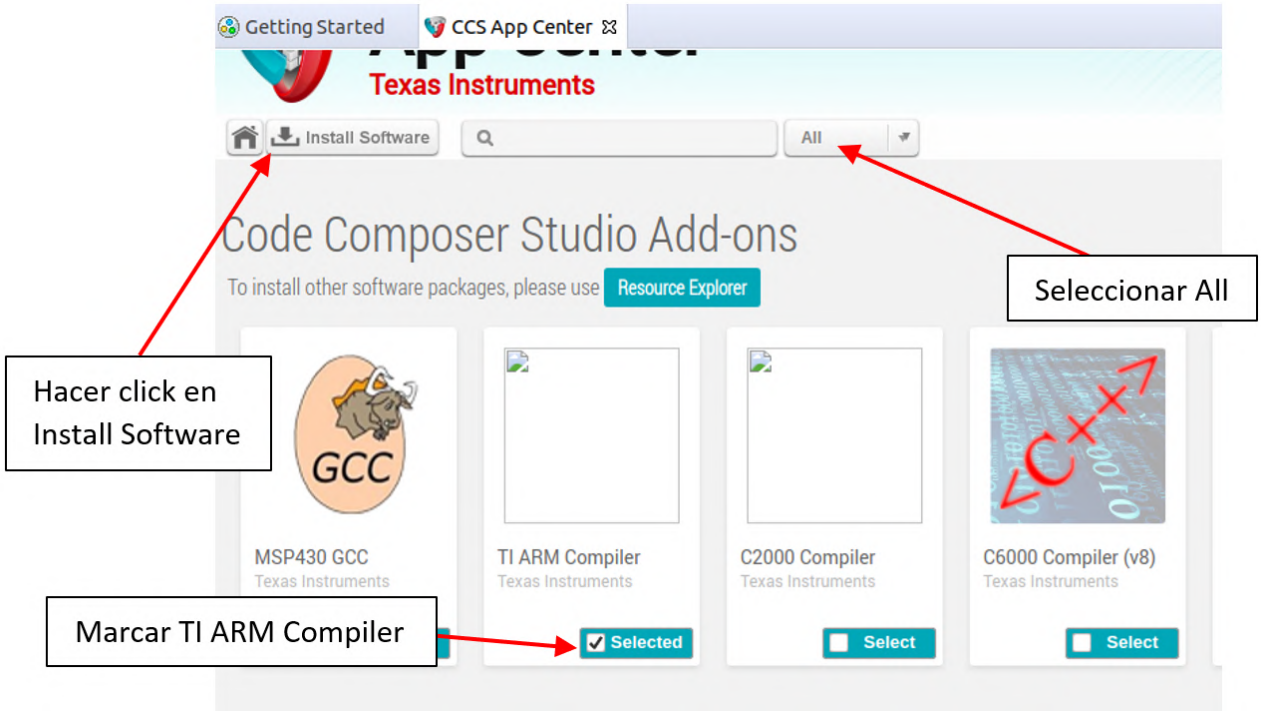


Figura 8.4: Instalar TI ARM Compiler

Armar un nuevo proyecto en CCS y verificar conexión con SoC, tal como se puede ver en la figura 8.5.

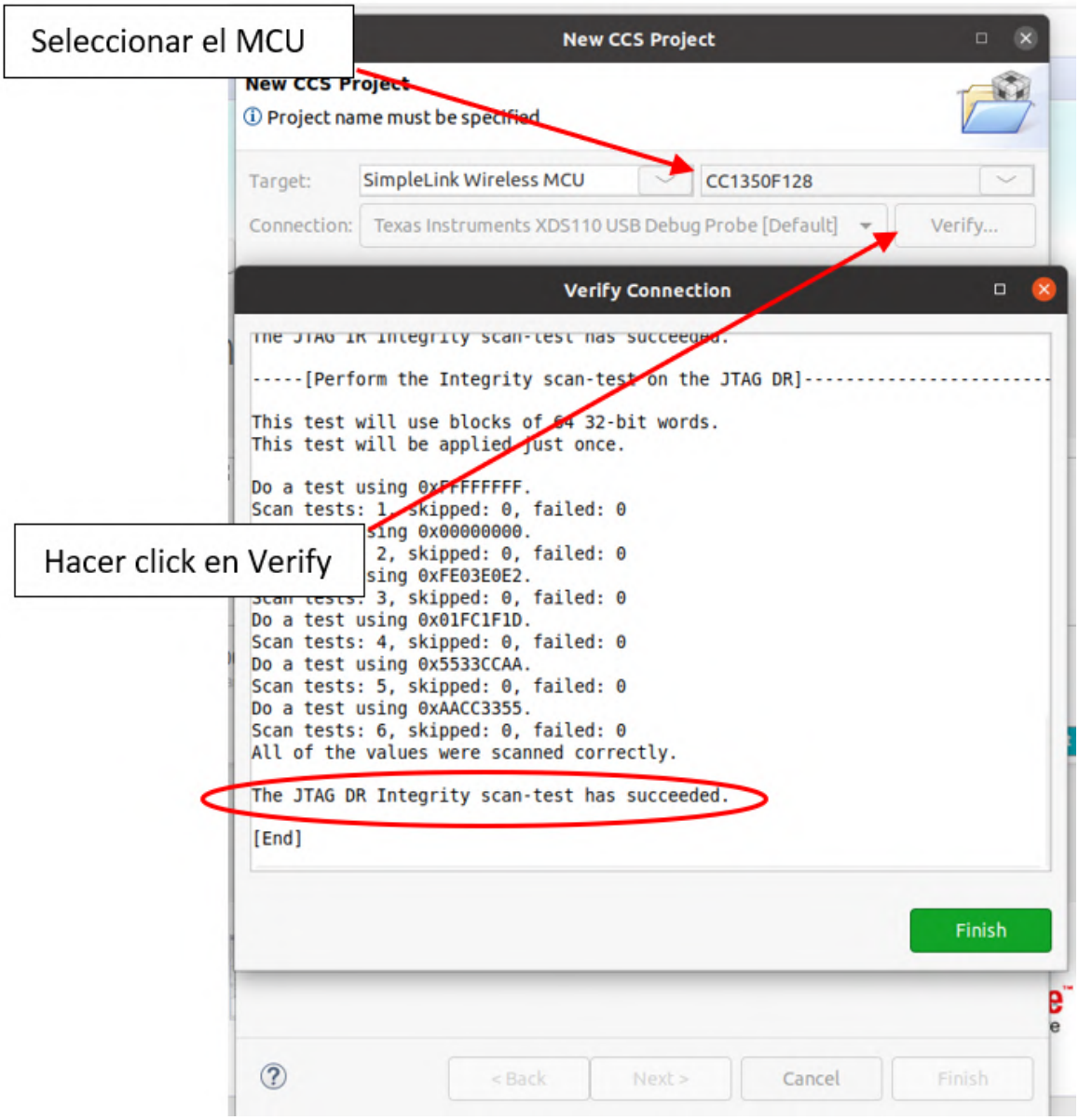


Figura 8.5: Verificar conexión

.

8.1.3. Compilar un proyecto y utilizar Debug

Esta parte del tutorial se basó en el tutorial de Texas Instruments [16]. Se decidió realizar este tutorial, debido a que los pasos indicados por el tutorial de Texas Instrumentos no son suficientes, y al seguirlos surgen errores.

Empezaremos por armar un nuevo proyecto. Se debe seleccionar el SoC, el compilador GNU y la opción Empty Project, tal como se indica en la figura 8.6.

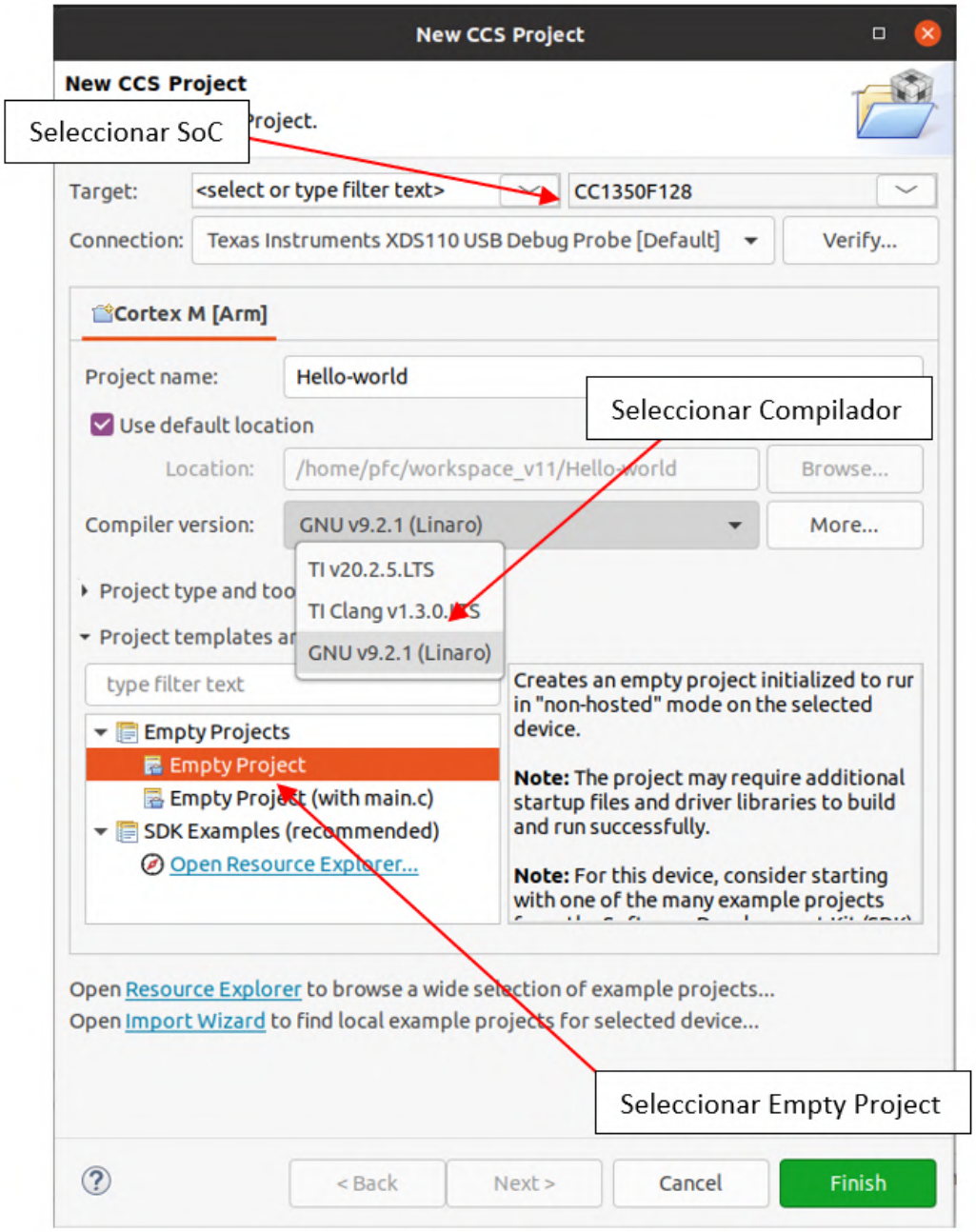


Figura 8.6: Armar nuevo proyecto

Procederemos agregando el directorio de Contiki-NG al Path. Haciendo click derecho sobre el proyecto armado y seleccionando la opción Properties podremos agregar un nuevo Path, seleccionando Linked Resources en Resources, New y colocando la dirección de Contiki-NG. Ver figura 8.7.

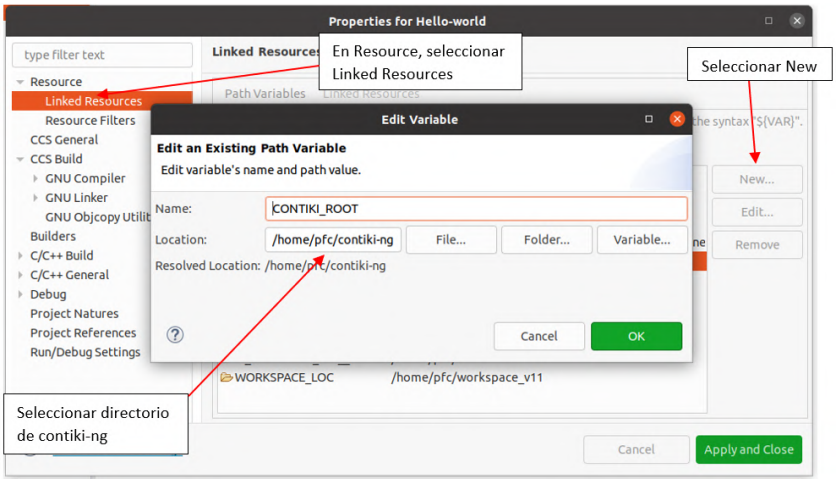


Figura 8.7: Agregar Contiki-NG al Path

Luego haciendo click derecho sobre el proyecto armado, podemos agregar una nueva carpeta, seleccionando New y luego Folder. Esta carpeta la vincularemos a Contiki-NG, como se indica en la figura 8.8.

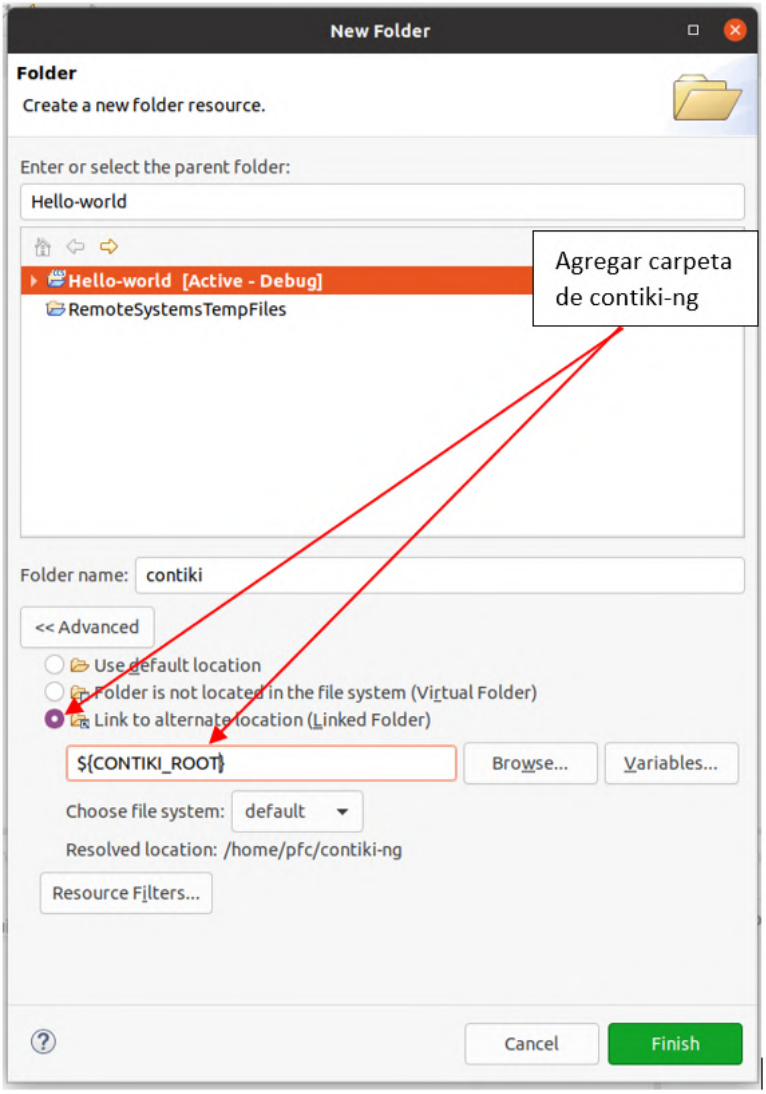


Figura 8.8: Agregar carpeta Contiki-NG

Luego debemos configurar el CCS Build. Entrando de nuevo en la opción de Propiedades, seleccionamos CCS Build, donde desmarcaremos la opción de Makefile generation y Auto build. Debemos seleccionar la carpeta donde se encuentra el código principal del proyecto, en este caso /contiki-ng/examples/hello-world. Ver figura 8.9.

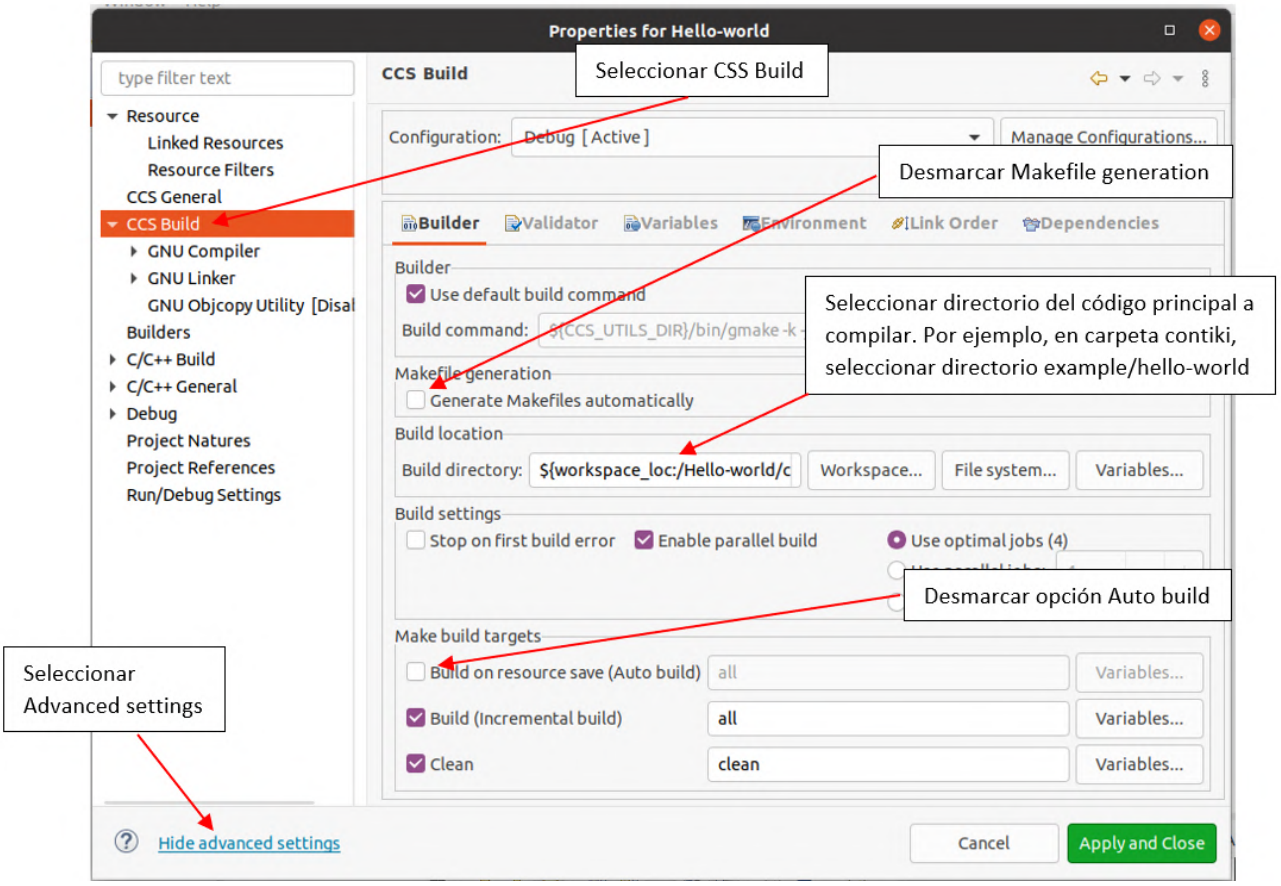


Figura 8.9: Configurar CCS Build

Se debe agregar al Makefile el TARGET, BOARD y CFLAGS. Al agregar CFLAGS se debe colocar el comando -g, que cumple la función de agregar el código necesario para el debug, tal como se indica en la figura 8.10

Se identifica que existen 2 posibles targets desarrollados para la compilación de Contiki-NG en el launchpad CC1350. Se puede configurar el target como:

- TARGET=simplelink
- TARGET=cc26x0-cc13x0.

No hemos investigado las posibles diferencias. En foros de Texas Instruments se recomienda utilizar el target cc26x0-cc13x0 si se desea utilizar el debugger de CCS.

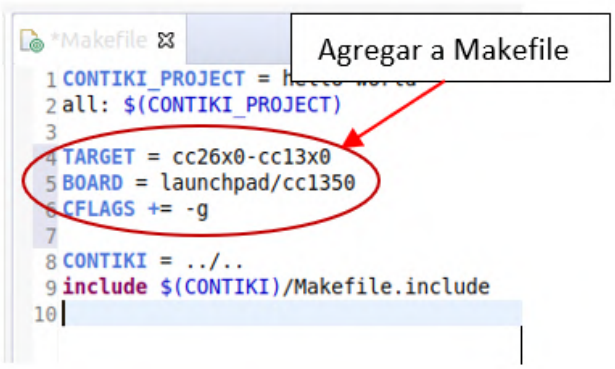


Figura 8.10: Makefile

Se debe comentar la inicialización del watchdog, en contiki-main.c. Este código se encuentra en la carpeta OS de contiki.

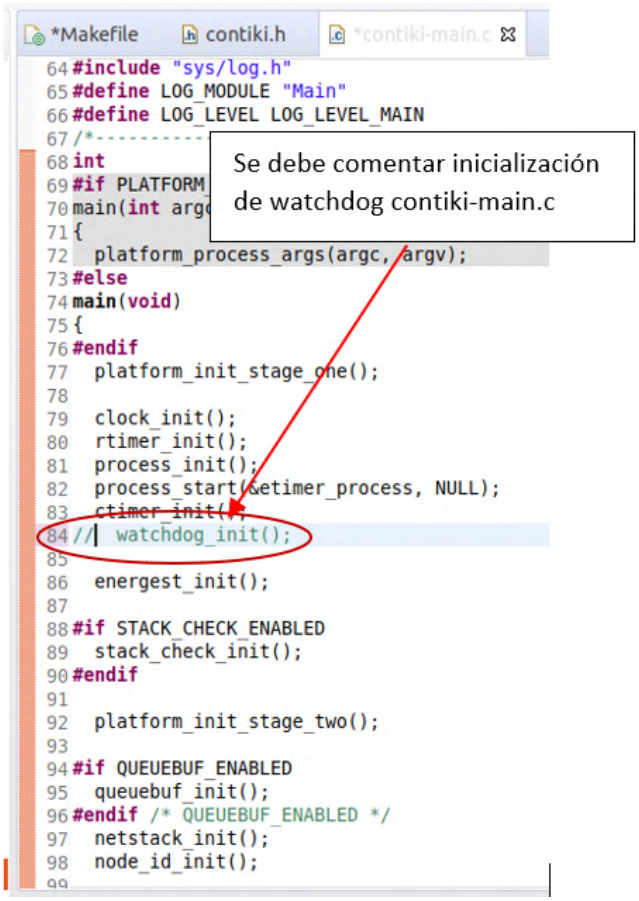


Figura 8.11: Código contiki-main.c

Se debe compilar el código y posteriormente realizar click en Debug, tal como se indica a seguir.

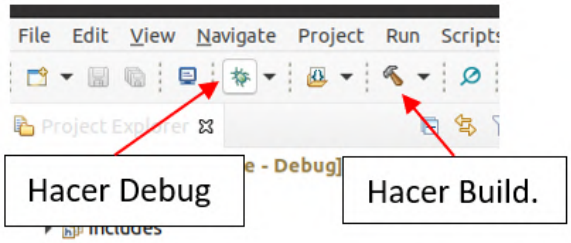


Figura 8.12: Compilar y debug

Es de esperar que se genere un error, pero es necesario realizar este paso, para que se generen los archivos necesarios para realizar el Debug. El error que generalmente aparece se indica en la figura 8.13.

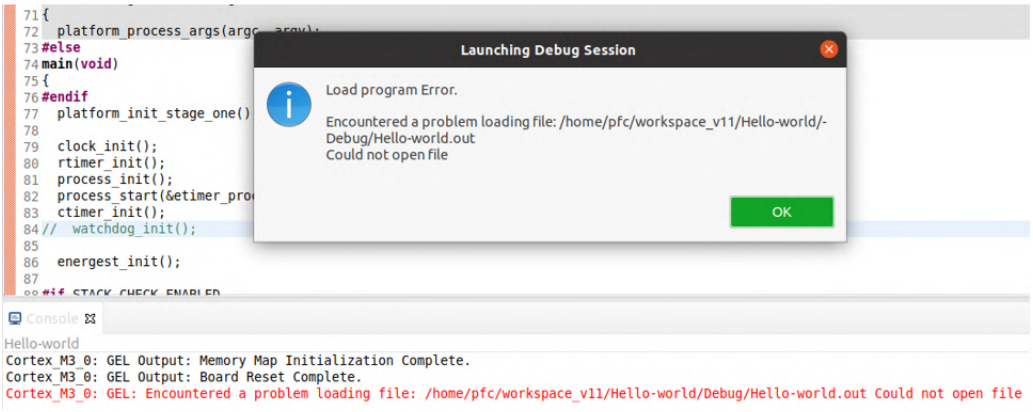


Figura 8.13: Error luego de hacer click en Debug.

Luego se necesita entrar a las propiedades del proyecto, seleccionar Run/Debug Settings y seleccionar Edit, como se muestra en la figura 8.14.

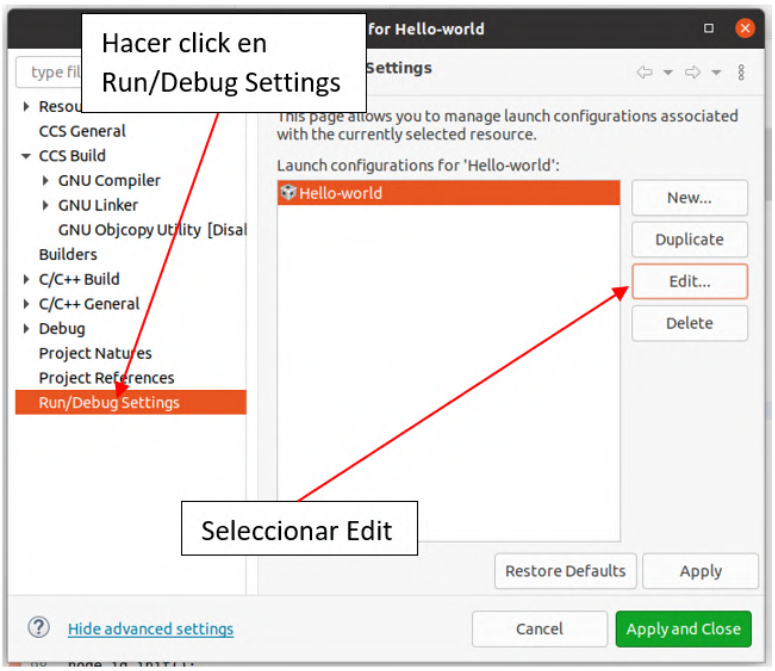


Figura 8.14: Editar Run/Debug Settings

Seleccionar la opción Program, y en File System, seleccionar el archivo .elf, ubicado en las subcarpetas que se generan en la carpeta donde se ubica el código principal.

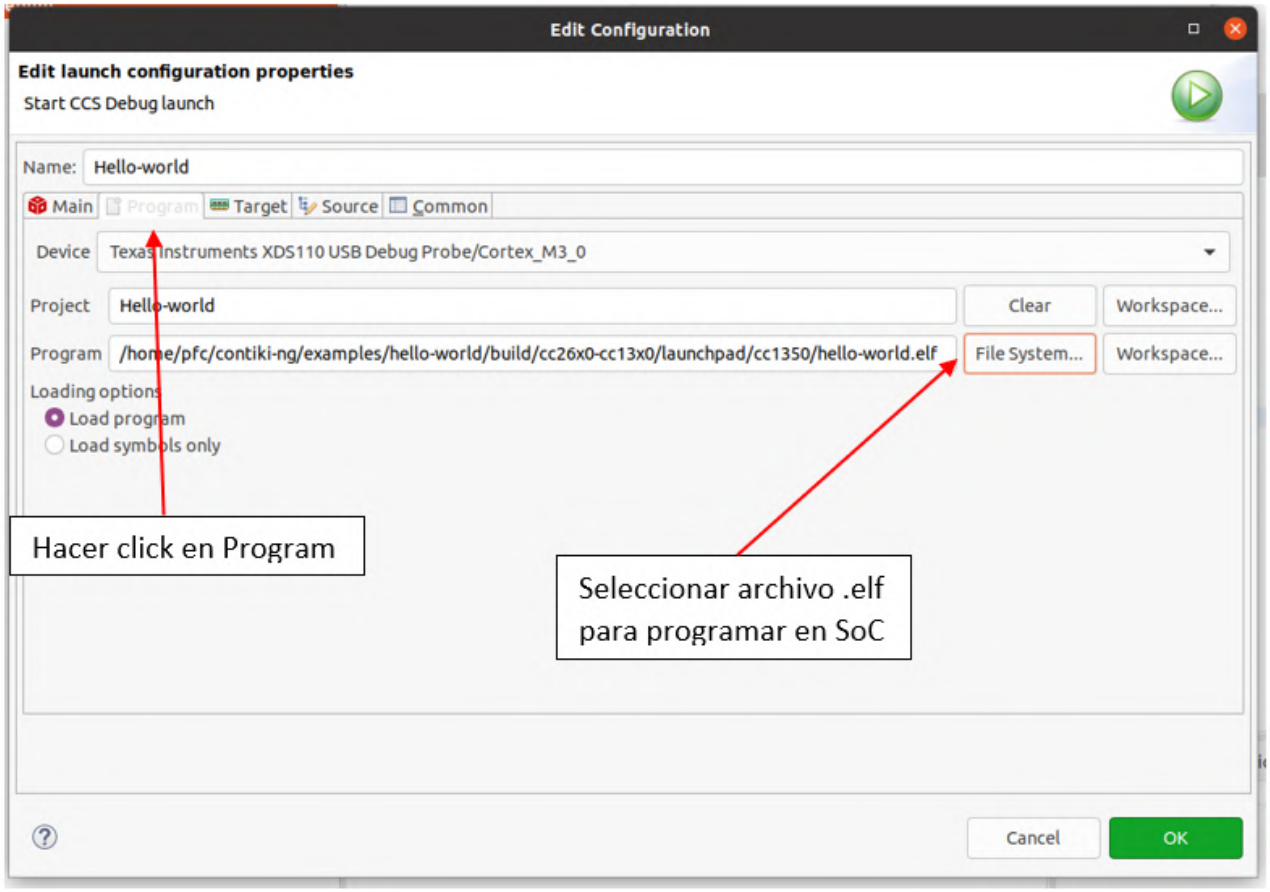


Figura 8.15: Seleccionar archivo a cargar en SoC

Ahora se puede volver a hacer click sobre la opción Debug. Una vez se cargue el programa al SoC, se deben realizar los pasos que se indican en la figura 8.16

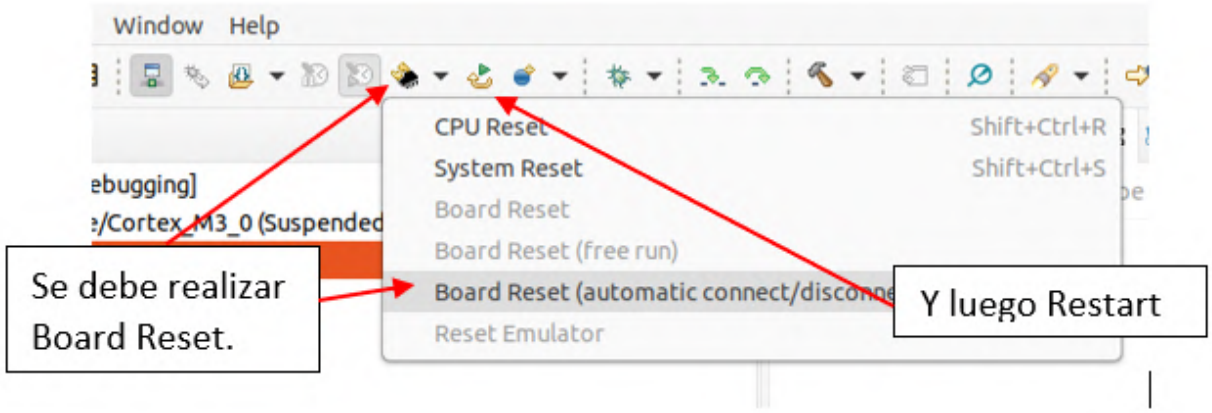


Figura 8.16: Reset y Restart

Al realizar click en Restart, ya se puede empezar a trabajar en el Debug.

Referencias

[1] RFC 768 - User Datagram Protocol. https://www.rfc-editor.org/info/rfc768, August 1980.

[2] IEEE Standard for Low-Rate Wireless Networks. *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)*, pages 1–800, 2020.

[3] Advanced Monolithic System. Datasheet AMS1117-3.3 [Online]. Available:. http://www.advanced-monolithic.com/pdf/ds1117.pdf.

[4] Aiocoap Developers. Web page. https://aiocoap.readthedocs.io/en/latest/api.html.

[5] Roger Alexander, Anders Brandt, JP Vasseur, Jonathan Hui, Kris Pister, Pascal Thubert, P Levis, Rene Struik, Richard Kelsey, and Tim Winter. RFC 6550 - RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. https://rfc-editor.org/rfc/rfc6550.txt, March 2012.

[6] Aosong. Datasheet DHT-11 [Online]. Available:. https://components101.com/sites/default/files/component_datasheet/DHT11-Temperature-Sensor.pdf.

[7] Author. Title. *Journal*, 10(2):1–2, 2001.

[8] Bruce Bugbee. Indoor hemp production. [Online]. Available:. https://www.youtube.com/watch?v=0I6awuKSp2c, 2022.

[9] Bruce Bugbee. Maximizing cannabis yields with dr bruce bugbee. https://youtu.be/JwtkHxv_3pU, Marzo 2020.

[10] Canakit. Datasheet - Raspberry Pi 4 Canakit [Online]. Available:. https://www.canakit.com/raspberry-pi-4-starter-max-aluminum-kit.html.

[11] Jorge Cervantes. *Indoor Marijuana Horticulture*. 2002.

[12] Khan IA Elsohly MA. Chandra S, Lata H. Photosynthetic response of Cannabis sativa L. to variations in photosynthetic photon flux densities, temperature and CO2 conditions. *National Center for Natural Product Research, School of Pharmacy, University of Mississippi.*, 23572895, 2008.

[13] Ronald Cohn. *Distributed Control System*.

[14] Contiki Developers. Documentation: CoAP. https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-CoAP, Mayo 2018.

[15] Contiki-NG Community. TSCH issue - Forum. [Online]. Available:. https://github.com/contiki-ng/contiki-ng/issues/1795.

[16] Contiki-NG Developers. Contiki-NG Toolchain installation on Linux. https://github.com/contiki-ng/contiki-ng/wiki/Toolchain-installation-on-Linux, December 2020.

[17] Contiki-NG Documentation. Contiki-NG [Online]. Available:. https://github.com/contiki-ng/contiki-ng.

[18] Contiki-NG Documentation. Contiki-NG [Online]. Available:. https://github.com/contiki-ng/contiki-ng/wiki/More-about-Contiki%E2%80%90NG.

[19] Contiki-NG Documentation. Porting Contiki-NG to new platforms. [Online]. Available:. https://github.com/contiki-ng/contiki-ng/wiki/Porting-Contiki%E2%80%90NG-to-new-platforms.

[20] Contiki-NG Documentation. TSCH and 6TiSCH. [Online]. Available:. https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-TSCH-and-6TiSCH.

[21] David Culler, Jonathan Hui, JP Vasseur, and Vishwas Manral. RFC 6554 - An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL). https://rfc-editor.org/rfc/rfc6554.txt, March 2012.

[22] D. Dang, A. Lele, and E. Wakefield. Designing an Ultra-Low-Power (ULP) Application With SimpleLink MSP432 Microcontrollers. *Texas Instruments Incorporated. All Rights Reserved.*, SLAA668A, 2015.

[23] Desconocido. Arduino vs Raspberry Pi vs BeagleBone vs PCduino [Online]. Available:. https://randomnerdtutorials.com/arduino-vs-raspberry-pi-vs-beaglebone-vs-pcduino/.

[24] Desconocido. Photosensitive Sensor Module [Online]. Available:. http://www.energiazero.org/arduino_sensori/photosensitive_sensor_module.pdf.

[25] DFRobots. Datasheet SEN0169-V2 [Online]. Available:. https://www.dfrobot.com/product-2069.html.

[26] DFRobots. Datasheet SEN0193 [Online]. Available:. https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0193_Web.pdf.

[27] Ever-way Industry. Datasheet HK19F-DC-3V [Online]. Available:. https://datasheetspdf.com/pdf-file/1015118/EVER-WAYINDUSTRY/HK19F-DC3V/1.

[28] Handson Technology. Datasheet módulo TP4056 [Online]. Available:. http://www.handsontec.com/dataspecs/module/18650-Lithium%20charger.pdf.

[29] IETF Group. IPv6 over Low power WPAN (6lowpan) [Online]. Available:. https://datatracker.ietf.org/wg/6lowpan/charter/.

[30] Pamela C. Korczynski, Joanne Logan, and James E. Faust. Mapping monthly distribution of daily light integrals across the contiguous united states. *HortTechnology horttech*, 12(1):12 – 16, 2002.

[31] F. Lima, J. Navarro, and M. Porto. BeeSmart. Proyecto de fin de carrera, Universidad de la República (Uruguay). Facultad de Ingeniería., 2021.

[32] A. Liñán, A. Vives, M. Zennaro, A. Bagula, and E. Piertosemmoli. *IoT in Five Days*. 2016.

[33] Richard L.Shell and Ernest L.Hall. *Handbook of Industrial Automation*. 2000.

[34] Luis Llamas. Que es un higrómetro FC-28? [Online]. Available:. https://www.bolanosdj.com.ar/MOVIL/ARDUINO2/MedidorHumedadSuelo.pdf, 2019.

[35] R. Hayashi P. Morales M. García Petillo, L. Puppo. Metodología para determinar los parámetros hídricos de un suelo a campo.

[36] Mango OS Developers. Web page. https://mango-os.com/.

[37] Mantech. Datasheet YF-S201 [Online]. Available:. http://www.mantech.co.za/datasheets/products/yf-s201_sea.pdf.

[38] Mean Well. Datasheet SPAN-02B-05 [Online]. Available:. https://www.meanwell-web.com/content/files/pdfs/productPdfs/MW/SPAN02/SPAN02,DPAN02-spec.pdf.

[39] Metaltex. Datasheet CB-1 [Online]. Available:. https://www.metaltex.com.br/assets/produtos/pdf/cb-1.pdf.

[40] Microchip. Datasheet MIC94090 [Online]. Available:. https://ww1.microchip.com/downloads/en/DeviceDoc/mic9409x.pdf.

[41] Gabriel Montenegro, Jonathan Hui, David Culler, and Nandakishore Kushalnagar. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. https://www.rfc-editor.org/info/rfc4944, sep 2007.

[42] No-Ip. Webpage [Online]. Available:. https://www.noip.com/es-MX.

[43] OSE. Consultas sobre calidad de agua [Online]. Available:. https://consultalab.ose.com.uy/hingreso.aspx.

[44] A. Parasuram. An Analysis of the RPL Routing Standard for Low Power and Lossy Networks. Master of science in computer science, University of California, Berkeley., 2016.

[45] Aquilino Rodríguez Penin. *Sistemas SCADA*. 2012.

[46] Roger Pertwee. *Handbook of Cannabis*. 2014.

[47] pyModbusTCP Developers. Web page. `https://pypi.org/project/pyModbusTCP/`.

[48] Raspberry Pi Foundation. Guía de instalación Raspberry Pi OS en SD [Online]. Available:. `https://www.raspberrypi.com/software/`.

[49] Real VNC. Webpage [Online]. Available:. `https://www.realvnc.com/es/connect/download/viewer/`.

[50] Dirk Raes Martin Smith Richard G. Allen, Luis S. Pereira. *Evapotranspiración del cultivo*. 2006.

[51] Zach Shelby, Klaus Hartke, and Carsten Bormann. RFC 7252 - The Constrained Application Protocol (CoAP). `https://www.rfc-editor.org/info/rfc7252`, June 2014.

[52] SMC. Repositorio proyecto [Online]. Available:. `https://gitlab.fing.edu.uy/carlos.menoni/pfc-smc.git`.

[53] Source Forge community. TSCH issue - Forum. [Online]. Available:. `https://sourceforge.net/p/contiki/mailman/message/36005109/`.

[54] A. Tanenbaum. *Modern Operating Systems*. 1992.

[55] Texas Instruments. CCS v11 Linux Host Support. `https://software-dl.ti.com/ccs/esd/documents/ccsv11_linux_host_support.html`.

[56] Texas Instruments. Datasheet CC1350 [Online]. Available:. `https://www.ti.com/lit/ds/symlink/cc1350.pdf`.

[57] Texas Instruments. Datasheet CC2538 [Online]. Available:. `https://www.ti.com/lit/ds/symlink/cc2538.pdf`.

[58] Texas Instruments. Datasheet CC2650 [Online]. Available:. `https://www.ti.com/lit/ds/symlink/cc2650.pdf`.

[59] Texas Instruments. Datasheet TLC555 [Online]. Available:. `https://www.ti.com/lit/ds/symlink/tlc555.pdf`.

[60] Texas Instruments. Descargar CCS v11. `https://www.ti.com/tool/download/CCSTUDIO`.

[61] Texas Instruments. Developing with Contiki-NG in CCS. https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/158/contiki_2D00_ccs_2D00_debugging.pptx.

[62] Texas Instruments. Uniflash. [Online]. Available:. https://www.ti.com/tool/UNIFLASH.

[63] Texas Instruments. XDS Emulation Software. https://software-dl.ti.com/ccs/esd/documents/xdsdebugprobes/emu_xds_software_package_download.html.

[64] Texas Instruments. XDS110 Driver. [Online]. Available:. https://software-dl.ti.com/ccs/esd/documents/xdsdebugprobes/emu_xds_software_package_download.html.

[65] Texas Instruments Community. Texas Instruments - Forum. [Online]. Available:. https://e2e.ti.com/support/wireless-connectivity/bluetooth-group/bluetooth/f/bluetooth-forum/572442/cc2650moda-factory-image-and-bootloader.

[66] Texas Instruments Community. Texas Instruments - Forum. [Online]. Available:. https://e2e.ti.com/support/wireless-connectivity/sub-1-ghz-group/sub-1-ghz/f/sub-1-ghz-forum/797800/cc1310-error-write-to-internal-flash-contiki-ng.

[67] Texas Instruments Community. Texas Instruments - Forum. [Online]. Available:. https://e2e.ti.com/support/wireless-connectivity/zigbee-thread-group/zigbee-and-thread/f/zigbee-thread-forum/506551/cc2650-write-to-internal-128kb-flash.

[68] Texas Instruments Incorporated. *CC13x0, CC26x0 SimpleLink™ Wireless MCU Technical Reference Manual.* https://www.ti.com/lit/ug/swcu117i/swcu117i.pdf.

[69] Thingbsoard. Thingsboard Documentation User Guide. [Online]. Available:. https://thingsboard.io/docs/user-guide.

[70] Thingsboard. Thingsboard Web Page. [Online]. Available: . https://thingsboard.io/.

[71] Thingsboard Community. CoAP observation drops[Online]. Available:. https://github.com/thingsboard/thingsboard/issues/5263.

[72] Thingsboard Developers. Web page. https://thingsboard.io/.

[73] Thingsboard Documentation. Getting Started with Thingsboard [Online]. Available:. https://thingsboard.io/docs/getting-started-guides/helloworld/.

[74] Thingsboard Documentation. Guía de instalación para Ubuntu [Online]. Available:. https://thingsboard.io/docs/user-guide/install/ubuntu.

[75] Thingsboard Documentation. Guía de instalación Thingsboard en Raspberry Pi OS [Online]. Available:. https://thingsboard.io/docs/user-guide/install/rpi/.

[76] Wikipedia. Cannabis en Uruguay. [Online]. Available:. https://es.wikipedia.org/wiki/Cannabis_en_Uruguay.

[77] Wikipedia. Growroom. [Online]. Available:. https://en.wikipedia.org/wiki/Growroom.

[78] Wikipedia. Protocolo de enrutamiento RPL. [Online]. Available:. https://es.wikipedia.org/wiki/Protocolo_de_enrutamiento_RPL.

[79] Wikipedia. Time Slotted Channel Hopping. [Online]. Available:. https://en.wikipedia.org/wiki/Time_Slotted_Channel_Hopping.

[80] Winsen. Datasheet MH-Z14 [Online]. Available:. https://www.winsen-sensor.com/d/files/MH-Z14.pdf.

[81] L. Zotarelli, Michael Dukes, and Kelly Morgan. Interpretación del contenido de la humedad del suelo para determinar capacidad de campo y evitar riego excesivo en suelos arenosos utilizando sensores de humedad. *EDIS*, 2013, 02 2013.

Índice de tablas

2.1. Resumen de valores objetivo por parámetro. 9

3.1. Detalle de actuadores instalados. 19
3.2. Detalle de sensores instalados. 20

4.1. Tabla comparativa de computadoras de placa simple. 26
4.2. Comparativa de chips 28
4.3. Distribución de borneras atornillables según su uso. 32
4.4. Comparación entre sensores DHT-11 y DHT-22. 37
4.5. Características de caudalímetros. 43
4.6. Comparación de electroválvulas. 45
4.7. Llaves selectoras utilizadas 50

5.1. Mensajes entre UCD y NM, y desde UC hacia UCD-R. 65
5.2. Mensajes de telemetría hacia UC desde NM 66
5.3. Mensajes de telemetría hacia UC desde UCD 67
5.4. Mensajes de comando y configuración de parametros desde UC. . . 68

6.1. Rangos y valores del ajuste de pH. 76
6.2. *Set-points* de temperaturas ambiente. 80

7.1. Procesamiento y filtrado según variable 90
7.2. Condición de disparo de alarma y mensaje desplegado 90

A.1. Coeficientes del modelo lineal de cada sensor de humedad de suelo. 116
A.2. Medida al utilizar buffer 118

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

2.1. Equipo de aeroclonación. 10
2.2. Sala de crecimiento vegetativo. 10
2.3. Sala de floración. Las plantas se adecúan para una disposición de tipo ScrOG. 11
2.4. Sala de secado. 12

3.1. Distribución de placas en el establecimiento. 17
3.2. Distribución de sensores en las distintas salas. 20

4.1. Computadora de placa simple Raspberry Pi 4. 27
4.2. Módulo CC2650MODA de Texas Instruments. 29
4.3. Bloques funcionales de UCD. 30
4.4. Circuito de relés. 32
4.5. Bloques funcionales de NM. 33
4.6. Laboratorio DECE de la UCU con los equipos Pick and Place y horno de reflujo. 35
4.7. Pasta de soldar sobre almohadillas de placa UCD. 35
4.8. Resultado de placas luego del proceso de soldadura de componentes SMD por horno de reflujo. 36
4.9. Placa UCD con todos sus componentes soldados. 36
4.10. Solución alternativa para conexión de diodo. 37
4.11. Sensor de humedad y temperatura ambiente DHT-11. 38
4.12. Sensor de iluminación basado en LDR. 38
4.13. Módulo sensor de CO_2 intervenido para obtener la salida analógica. 39
4.14. Diagrama esquemático del circuito interno del sensor SEN0193. . . 40
4.15. Imagen del sensor de humedad capacitivo y su protección. 41
4.16. Sensor de temperatura de suelo. 41
4.17. Componentes de la medición de pH: sonda y placa de conversión de señal. 42
4.18. Caudalímetro YF-S201. 43
4.19. Electroválvula para sistema de riego. 45
4.20. Bomba de recirculación. 46
4.21. Bomba peristáltica de dosificación. 46
4.22. Modelo lineal de bomba peristáltica. 47

5.1. Diagrama de red. 52

5.2. Stack de protocolos 53
5.3. *Slots* TSCH. Cada color representa la comunicación entre dos dispositivos [79] 54
5.4. Diagrama de red a nivel RPL a modo ejemplo [78]. 56
5.5. Ejemplo tipos de mensajes CoAP. 58
5.6. Ejemplo *piggybacked* y *separate response* 58
5.7. Ejemplo CoAP *Observe* 59
5.8. Clientes y servidores CoAP 60
5.9. Envío de telemetría entre placas 61
5.10. Envío de telemetría a Thingsboard 62
5.11. Izquierda: suscribirse a un recurso. Derecha: actualización de valores 63
5.12. Mensaje enviado desde Thingsboard hacia UCD 64

6.1. Diagrama de flujo del proceso de riego. 72
6.2. Modelo lineal del lazo de control. 75
6.3. Diagrama de lazo de humedad de suelo. 76
6.4. Diagrama de flujo de ajuste de pH. 77
6.5. Diagrama de bloques del lazo de control de pH. 78
6.6. Modelo de variación de pH por dosificación de ácido fosfórico. . . . 78
6.7. Diagrama de lazo de control de humedad ambiente. 80
6.8. Diagrama del lazo de control de temperatura ambiente. 82

7.1. Arquitectura de Software Raspberry 86
7.2. Lógica cadena de reglas 88
7.3. Panel Vegetativo. 91

8.1. Sistema armado para probar Unidad Central 94
8.2. Nodo Maceta 95
8.3. Sistema armado para probar las UCD. 96
8.4. Sistema armado para probar UCD-R1 y UCD-R2 97
8.5. Tablero UCD-R 98
8.6. Block de riego 98
8.7. Unidad Central 100
8.8. Tablero UCD-V 101
8.9. Tablero UCD-S 101
8.10. Izquierda: Sensor DHT-11. Derecha: Enchufe controlado Deshumidificador SS 102
8.11. Tempearatura ambiente Sala Vegetativo 104

A.1. Evolución en tiempo de lectura de sensor resistivo, ante saltos de humedad. 114
A.2. Evolución en tiempo de lectura de sensor capacitivo, ante saltos de humedad. 114
A.3. Evolución en tiempo de lecturas de sensores resistivo (rojo) y capacitivo (azul). 115
A.4. Modelos lineales de sensores de humedad de suelo. 117

A.5. Comparación de Modelos Lineales de Sensores. 118
A.6. Modelo lineal para sonda de pH. 119
A.7. Prototipo utilizado para la parametrización. 120
A.8. Evolución temporal del pH durante el ensayo. 121
A.9. Modelo de variación de pH por dosificación de ácido fosfórico. . . . 122
A.10.Modelo lineal del sensor de CO2. 123
A.11.Configuración para ensayo de parametrización de bomba peristáltica.124
A.12.Modelo lineal de bomba peristáltica. 125

B.1. Unidad Central 130
B.2. Launchpad cc2650 133
B.3. Página principal del Router ZTE F680 134
B.4. Página Internet del Router ZTE F680 135
B.5. Página Seguridad del Router ZTE F680 135
B.6. Configuración de ruta para Thingsboard 136
B.7. Configuración de ruta para SSH 136
B.8. Configuración de ruta para VNC 136
B.9. Ruteo NM 137
B.10.Placa UCD 139
B.11.Bloques funcionales UCD 140
B.12.Pistas UCD 141
B.13.Agregar Nodo SCADA 143
B.14.Agregar Nodo SCADA 143
B.15.Base de datos local 144
B.16.Diccionario setvalues.py 144
B.17.Conexión Launchpad a NM o UCD para programar 147
B.18.Conexión Launchpad a placa 148
B.19.Uniflash seleccionar SoC 149
B.20.Uniflash cargar programa 150
B.21.Flash Programmer 2, conexión a SoC por JTAG 151

C.1. Acceso web SCADA 153
C.2. Panel General del SCADA 154
C.3. Menú de navegación 154
C.4. Menú de navegación 155
C.5. Botón de regreso 155
C.6. Acceso a históricos 156
C.7. Histórico luces Sala Floración 156
C.8. Panel Floración del SCADA 157
C.9. Panel Vegetativo del SCADA 157
C.10.Panel Secado del SCADA 158
C.11.Panel Riego del SCADA 158
C.12.Panel Interno del SCADA 159
C.13.Intervalo de tiempo 159
C.14.Popup de configuración 160
C.15.*widget* de Alarmas 161

C.16.Bandeja de Entrada 162
C.17.Botón 162
C.18.Comando manual 163
C.19.*Update server attribute* 163
C.20.Llave Selectora UCD 164
C.21.Leds indicadores de carga 165
C.22.Aplicación VNC Viewer 165
C.23.Botón timeout 166

Esta es la última página.
Compilado el martes 20 diciembre, 2022.
http://iie.fing.edu.uy/