Universidad de la República
Facultad de Ingeniería

Habitación Educada

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

Abraham Rébori, Andrés Grignola, Juan Alvarez

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

Tutor

Ing. Leonardo Barboni, PhD ..... Universidad de la República

Tribunal

Ing. Gonzalo Casaravilla, PhD ..... Universidad de la República
Ing. Julio Pérez Acle, Msc ..... Universidad de la República
Ing. Guillermo Antunez ..... Universidad de la República
Ing. Leonardo Barboni, PhD ..... Universidad de la República

Montevideo
5 de junio, 2017

Habitación Educada, Abraham Rébori, Andrés Grignola, Juan Alvarez.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 128 páginas.
Compilada el martes 18 julio, 2017.
http://iie.fing.edu.uy/

No nos quedemos sólo con los resultados para valorar lo que se hace, el éxito no son solo los resultados, sino las dificultades que se pasan para obtenerlos y el espíritu de plantearse desafíos, y también la valentía para superarlos. El camino es la recompensa.

Mtro. Oscar Wáshington Tabárez

Decía Bernardo de Chartres que somos como enanos a los hombros de gigantes. Podemos ver más, y más lejos que ellos, no por la agudeza de nuestra vista ni por la altura de nuestro cuerpo, sino porque somos levantados por su gran altura.

Juan de Salisbury

Esta página ha sido intencionalmente dejada en blanco.

Agradecimientos

Al País y la oportunidad que brinda la Educación Pública, mas allá de lo perfectible que sea.

A la barra que empezó siendo un grupo de estudio y se convirtió en compañeros de trinchera de esta guerra sin cuartel.

Todos.

A mis padres por inculcarme la curiosidad por aprender, y a mi hermana por la compañía y escucha de estos últimos tiempos.

Abraham.

A Valentina por su paciencia y amor, a Maite por la motivación y a mis viejos por insistir en ésto.

Andrés.

A toda mi gran familia, en especial padres, abuelos, hermana y tía Zulema por darme el apoyo fundamental para lograr esto y a Carolina por tanta comprensión y por formar parte de este camino.

Juan Francisco.

Esta página ha sido intencionalmente dejada en blanco.

A todos los que nos conocen.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

El proyecto se desarrolla en el marco de la automatización hogareña y la eficiencia energética, específicamente, se trabaja sobre la gestión de la demanda y la optimización de recursos energéticos generados localmente, mediante paneles fotovoltaicos.

En particular, se definen dos ámbitos de investigación y desarrollo: la gestión de la demanda, y la microgeneración y uso de la energía generada.

Se plantearon los siguientes objetivos:

- En el plano de la gestión de la demanda:
  - Desarrollar e implementar la infraestructura para la obtención de las variables ambientales de la habitación, la actuación sobre elementos de consumo del sistema y el registro del consumo.
  - Las características de la infraestructura desarrollada son; comunicación inalámbrica sobre WiFi, un nodo central, implementado en una Single Board Computer, con sistema operativo basado en linux y nodos periféricos basados en microcontroladores.
  - En los nodos periféricos se da la interacción entre el entorno y el sistema; se definen sensores que relevan las variables ambientales y energéticas y actuadores con los cuales se actúa sobre el sistema.
  - En el nodo central se ejecutan las rutinas de optimización del sistema y el modelo de planta fotovoltaica offgrid.

- En el plano de la microgeneración y uso de energía:
  - Desarrollar e implementar el modelo de un sistema de generación fotovoltaico offgrid, para ésto se modelan en python por un lado, la generación del arreglo de paneles solares, en base a la irradiancia y la temperatura, y por otro, el banco de baterías.
  - Desarrollar e implementar una rutina de optimización para el uso de la energía generada, en base a lógica difusa.

- Comparar el costo asociado al uso de la energía por parte de la rutina de optimización desarrollada con el costo asociado al uso del inversor sin "inteligencia".

En conclusión, se implementó el sistema con un nodo central y nodos periféricos, se probó su funcionamiento durante un mes y medio, reportando las variables medidas, se desarrolló la rutina de optimización y se modeló un sistema de generación fotovoltaica con almacenamiento local en baterías. Es destacable que se obtuvo resultados alentadores para distintas series de generación (catalogadas como alta, media y baja según el Mapa Solar del Uruguay). En promedio mostró un 9,4 % de ahorro respecto al sistema convencional fotovoltaico off-grid.

Se deja el registro en esta memoria de las pruebas realizadas, los resultados obtenidos, los procedimientos y herramientas utilizadas, así como también una descripción detallada de los elementos de Hardware utilizados, su configuración y el código desarrollado, para poder continuar las investigaciones en la línea de este trabajo.

Prefacio

El lector recibe en sus manos, o en su monitor, el resultado de más de un año de trabajo, con encuentros y desencuentros, logros y golpes contra la pared.

Este trabajo es fruto del recorrido de un camino sinuoso que partió de una idea nuestra y se convirtió en un producto tangible y útil. Quizás el camino fue más sinuoso de lo necesario, pero al partir de una idea, no del pedido de un "cliente", los objetivos, el alcance y el trabajo está mucho menos definido que otro tipo de proyectos. Ésto tiene sus pro y sus contra. En cuanto a los pros, está la sensación de libertad y la movilidad a la hora de hacer foco en los distintos problemas que surgen en el trayecto. Por el lado de las contras, está la desazón debida a las incógnitas que van apareciendo y el tiempo que puede llevar sortear los obstáculos, ya que no hay nadie abajo con la red, con la visión global, sino que es uno mismo el que elige las batallas y busca las herramientas para ganarlas.

Queremos aprovechar para destacar el desafío que supuso para nosotros este proyecto y el aprendizaje que nos llevamos en áreas que no son trabajadas específicamente en la carrera; Optimización, Python, Microgeneración, Energía fotovoltaica, etc. Fue un proyecto muy enriquecedor.

Los autores.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 1

Introducción

1.1. Objetivo General

El proyecto tiene como objetivo el estudio e implementación de un sistema inalámbrico de sensores y actuadores que permita minimizar el gasto que origina el consumo eléctrico de un ambiente, agregando al sistema la capacidad de gestionar el uso de la energía generada localmente, y la posibilidad de optimizar el consumo de la habitación.

1.2. Objetivos Específicos

Para alcanzar el objetivo general se trabaja en objetivos concretos, definidos como:

- Diseño e implementación de módulos periféricos para medición de las variables que definen el confort de la habitación.
- Diseño e implementación de un módulo que permita obtener y definir la curva de demanda energética de la habitación.
- Análisis y definición del modelo matemático de un sistema fotovoltaico off-grid, incluyendo el panel solar, el almacenamiento y el inversor, en base a un funcionamiento MPPT[1].
- Diseño e implementación de un módulo exterior para la medición de las variables necesarias para obtener la generación y almacenamiento, en tiempo real, del sistema modelado.
- Estudiar la posibilidad de que la ejecución de modelos y rutinas de optimización se haga en forma local o remota (en servidores externos).

[1]Maximum Power Point Tracking, modo de funcionamiento del bus de continua en un arreglo de paneles solares que perite extraer el máximo de potencia del sistema.

- Definir la topología del sistema, tal que permita la integración de todos los sensores necesarios, así como la ampliación de la cantidad de los mismos.
- Estudio y desarrollo de una rutina de optimización del consumo energético del ambiente en base a la curva de demanda energética, la curva de generación, la energía almacenada y el costo del kWh, que defina el origen de la energía consumida en tiempo real.
- Implementar la rutina de optimización con los datos obtenidos por el sistema y análisis de resultados.
- Como último objetivo el sistema debe ser armado y sometido a pruebas de funcionalidad en un caso prototipo.

Para sentar las bases de trabajo de ahora en más, se establece la definición de un sistema fotovoltaico offgrid, estos son sistemas de generación de energía a partir de paneles solares los cuales están pensados para su uso en forma aislada de la red macro de abastecimiento como el término OFFGRID así refiere. Es decir, la energía generada deberá ser almacenada o utilizada localmente en el abastecimiento de un conjunto de cargas definidas las cuales están conectadas a la salida del sistema. Como se ve en la un sistema fotovoltaico offgrid típico está compuesto de la siguiente manera:

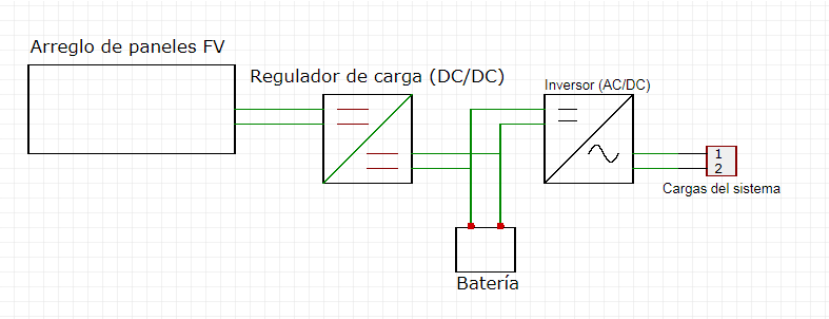


Figura 1.1: Diagrama de un sistema fotovoltaico offgrid típico.

- Arreglo de paneles solares fotovoltaicos: tienen la función de generar energía eléctrica a partir de radiación solar, esto se logra dado que los mismos están compuestos de silicio el cual presenta un efecto fotoeléctrico el cual consiste en la emisión de electrones al incidir sobre el material una radiación electromagnética como es el caso de la luz. Cabe destacar que la corriente generada es continua, y que los arreglos pueden contener determinada cantidad de paneles en serie o en paralelo buscando tener determinado rango de salida tanto en corriente como en voltaje.
- Baterías para almacenamiento de la energía: la energía generada por los paneles debe ser almacenada para su posterior utilización, tanto en DC como en AC, para esto el sistema dispone de baterías, las mismas suelen ser del

tipo de ciclo profundo dado que poseen la cualidad de ser utilizadas casi en la totalidad de su capacidad a diferencia de las baterías convencionales.

- Regulador de carga: este componente es básicamente un convertidor DC/DC que utiliza electrónica para transferir corriente continua cambiando su nivel de tensión. Esto es utilizado utilizando como entrada el arreglo de paneles para obtener a la salida una fuente regulada en tensión, la cual se conecta a las baterías dado que estas no pueden ser cargadas a cualquier nivel de voltaje ya que sino puede producirse un daño a las mismas.
- Inversor de onda pura: este convertidor DC/AC es utilizado para tomar energía desde las baterías y transformarla en energía AC de 50 Hz en el caso de Uruguay la cual puede ser utilizada por el conjunto de cargas conectadas al sistema, ocasionalmente pueden ser conectadas cargas en DC directamente a la salida de las baterías, pero esto es menos usual.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 2

Hardware

2.1. Introducción

En este capítulo serán precisados los aspectos referentes al hardware utilizado, haciendo énfasis en las razones para el uso del mismo y sus principales características técnicas.

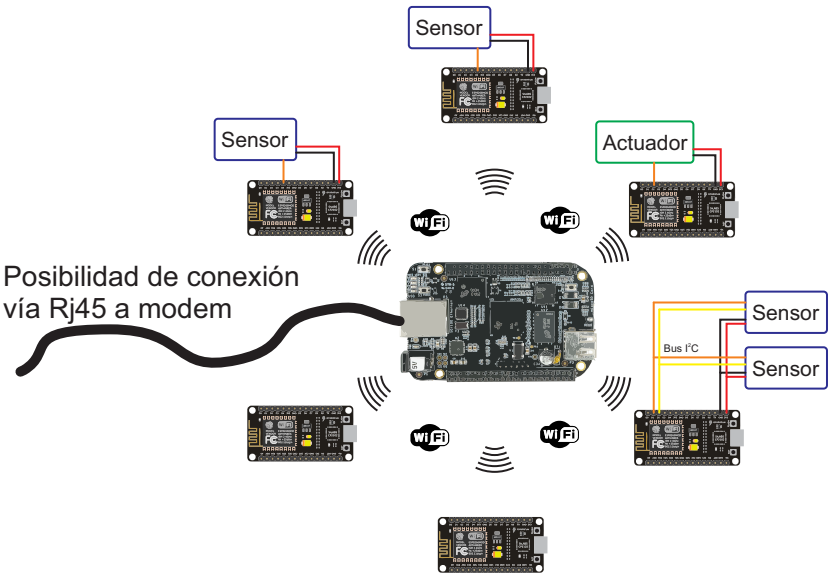


Figura 2.1: Diagrama General del Sistema; como nodo central la placa BeagleBone Black, como periféricos los ESP8266, y cableados a éstos últimos; los sensores y actuadores

2.2. Funciones a Resolver

El Hardware se divide en las siguientes categorías:

- Sensores analógicos y digitales para adquirir variables físicas.

- Actuadores para tomar acciones sobre el sistema.
- Módulos de comunicación entre sensores y nodo central, para descentralizar los sensores y actuadores antedichos.
- Nodo central del sistema, dónde se ejecuta la optimización. Debe procesar y almacenar los datos recogidos por los distintos módulos y comunicarse eventualmente con un servidor externo para alojar datos en él.

En la Figura 2.1 se puede ver un diagrama del sistema, con la interconexión de cada uno de los elementos nombrados anteriormente.

2.2.1. Sensores

En todos los casos, los sensores seleccionados cumplen la condición de ser de fácil adquisición y bajo costo, para simplificar su reposición en caso que sea necesario. Asimismo, estos criterios facilitan la replicación del proyecto.

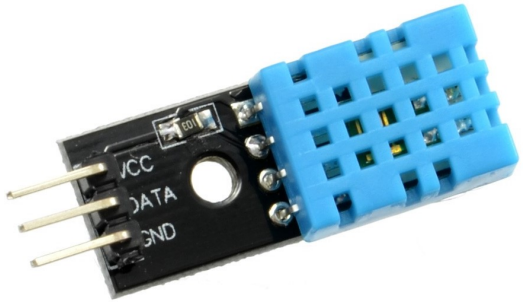


Figura 2.2: Sensor de temperatura DHT11

Sensor de Temperatura y Humedad DHT11

El proyecto utiliza para la medición de temperatura y humedad el sensor DTH11 (ver Figura 2.2). Éste brinda la información en formato digital, considerando que la ESP8266 tiene solo un pin analógico y once digitales, es una característica muy útil.

Tiene un rango de medición de temperatura de 0 °C a 50 °C, con una resolución de 1 °C y un error típico de ±1 °C, este sensor será usado para medir la temperatura interior y exterior[1] con este sensor. Respecto al rango de medición tanto la temperatura exterior e interior, en ambos casos, tienen un rango menor.

En el CD adjunto a la memoria se encuentra la hoja de datos de este sensor[2].

[1] En particular, la teperatura exterior será relevante en horas del día, por lo que no es necesario medir grados bajo cero.

[2] CD-HEduc\Datasheets\dht11.pdf

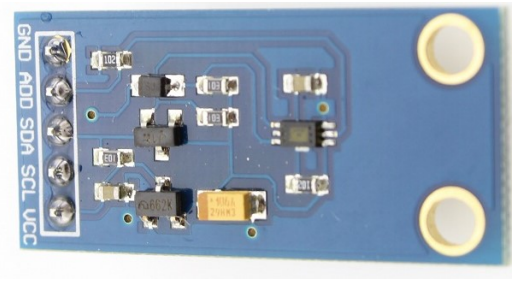


Figura 2.3: Sensor de luminosidad BH1750FVI

Sensor de Luminosidad BH1750FVI

El sensor BH1750FVI (ver Figura 2.3), se destaca por tener amplio rango y buena resolución en la medición, con la salida en formato digital, a través de un bus I^2C.

Sus características principales son; rango de medida desde 1 lx a 65535 lx, resolución de 1 lx en modo de alta resolución o 4 lx en baja. Mide luminosidad dentro de la habitación, en particular, sobre el escritorio, donde se espera medir un nivel de iluminación de 500-600lx.

Las características técnicas en detalle de este sensor se encuentran en el CD adjunto a la memoria[3].

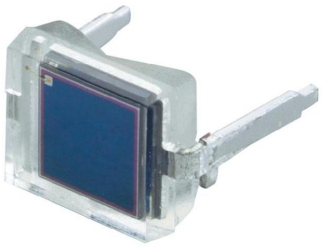


Figura 2.4: Fotodiodo de Silicio BPW34

Bus I^2C

Algunos sensores se comunican con la ESP8266 a través de un bus con el protocolo I^2C.

[3]CD-HEduc\Datasheets\bh1750fvi.pdf

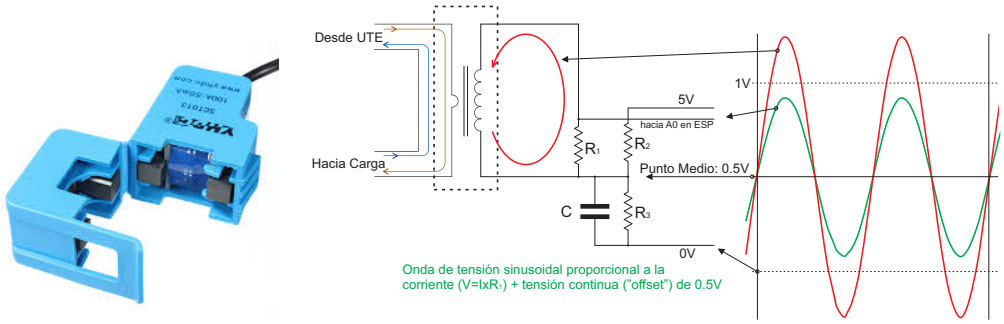
El protocolo I^2C consiste en un bus de datos en serie que permite la comunicación entre circuitos integrados a través de una infraestructura de dos cables, donde uno es la señal de reloj y, el otro es la señal de datos. El bus funciona como activo bajo (la señal activa es un cero no un uno), los datos se transmiten generando ceros por los elementos conectados. Como la comunicación es bidireccional, a través de salidas open-collector, es necesario utilizar resistencias como "pull-up", paralelas a los componentes conectados en las líneas del circuito.

El protocolo funciona con un único maestro, los esclavos se identifican con una dirección individual para cada uno.

Para información mas detallada, en el CD adjunto a la memoria se encuentra un documento sobre este protocolo de comunicación[4].

Fotodiodo de Silicio BPW34

El Fotodiodo de Silicio BPW34 (ver Figura 2.4) es, en los hechos, una celda fotovoltaica y permite obtener la medida de la irradiancia (en W/m^2), que se utiliza para el modelo del panel fotovoltaico. Se conecta a la entrada analógica de la ESP8266 a través de un circuito adicional sencillo, que convierte la corriente generada por la incidencia de la luz en el diodo en tensión, a través de una resistencia. Para disminuir la impedancia vista en la entrada analógica, se utiliza un Amplificador Operacional LM358. En el CD adjunto a la memoria se encuentra la hoja de datos de este sensor[5].



(a) Transformador de Corriente de Nucleo Partido SCT-013-000

(b) Diagrama explicativo del circuito complementario

Figura 2.5

Transformador de corriente de núcleo partido SCT-013-000

Con el fin de medir el consumo eléctrico de la habitación para modelar la demanda y definir el origen de la energía consumida, el sistema mide en tiempo

[4]CD-HEduc\Datasheets\I2C.pdf
[5]CD-HEduc\Datasheets\BPW34.pdf

real usando un transformador de corriente (ver Figura 2.5b). La pinza genera una señal analógica que se acondicionó para utilizar la entrada analógica de la ESP8266.

Para acondicionar la señal, se tuvieron en cuenta los siguientes aspectos funcionales:

- El transformador tiene una relación de vueltas de 1:2000, lo que es lo mismo a decir que con una corriente entrante de 100A, tiene una salida de 50mA.
- La entrada analógica tiene un rango de 0 a 1V, con un conversor analógico a digital de 10 bits.
- La corriente a medir es alterna.

Estas consideraciones imponen la necesidad de un hardware complementario, para que la señal a la entrada de la ESP8266 sea un voltaje y tenga la mayor resolución posible.

La solución planteada y la señal de entrada resultante se explican en la Figura 2.5b. Consiste en colocar una resistencia en paralelo con el bobinado secundario del transformador (R_1 en el diagrama, de 56 Ω) para obtener una tensión proporcional a la corriente que atraviesa la pinza. Se define el valor de la resistencia para que la tensión de pico a la entrada del pin analógico de la ESP8266 no supere 1V cuando la corriente consumida por la habitación sea 20A[6].

Como la entrada analógica no admite valores de tensión negativos, se realizó un divisor resistivo para centrar la tensión sinusoidal en 0,5V. Para ello se utilizaron dos resistencias en serie (R_2 de 56KΩ y R_3 de 10KΩ). A los efectos de reducir las fluctuaciones en este punto, se dispuso de un condensador en serie de 10μF. En el CD adjunto a la memoria se encuentra la hoja de datos de la pinza de corriente[7].

Sensor de Distancia

Se utiliza también un sensor de distancia HC-SR04 con el objetivo de detectar la presencia de personas próximas al espacio de trabajo del Nodo Interior. El módulo HC-SR04 (ver Figura 2.6) basado en pulsos de ultrasonido, resuelve esta necesidad en el marco de este proyecto. El mismo tiene un rango de medición de distancia de 2 cm a 4 m, con una resolución de 3 mm.

En el CD adjunto a la memoria se encuentra la hoja de datos de este sensor[8].

2.2.2. Actuadores

Módulo Dimmer

Para el control de la luminosidad de la habitación se dispone de un módulo, desarrollado para este proyecto, con la capacidad de dimmerizar luces. Partiendo de

[6]Se considera que el consumo de la habitación está limitada por protecciones térmicas de 20A.

[7]CD-HEduc\Datasheets\SCT013.pdf

[8]CD-HEduc\Datasheets\HC-SR04.pdf

Figura 2.6: Sensor de Distancia por Ultrasonido HC-SR04

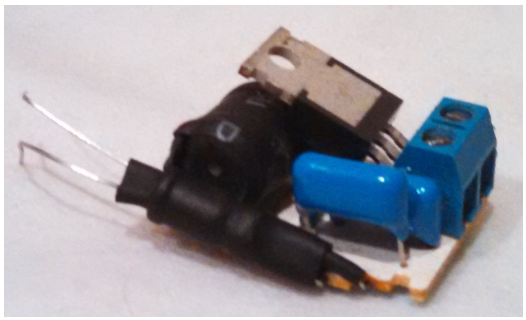


Figura 2.7: Modulo Dimmer

un módulo de Dimmer convencional, basado en el triac BT137-800, se sustituyó el potenciómetro por una fotoresistencia junto a un led en una vaina termocontraíble, con lo cual se obtuvo un módulo de dimmer aislado eléctricamente del sistema, controlable a través de un pin de la ESP8266 mediante una señal PWM.

(a) LED Emisor IR

(b) Receptor IR

Emisor y Receptor IR

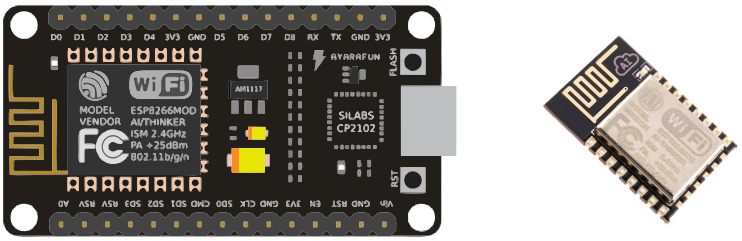
A los efectos de interactuar con el equipo de Aire Acondicionado se utiliza un LED emisor infrarrojo, en este caso, un IR333-A (ver Figura 2.8a). Para obtener las señales del control remoto, se dispuso de un receptor IR VS1838 (ver Figura 2.8b) para hacer la ingeniería inversa y descubrir los mensajes asociados a cada estado de funcionamiento.

El LED emisor cumple con la tarea de enviar el mensaje de 112 bits sobre una onda portadora de 38kHz. En la sección 4.3.4 se detalla como se desgrabó el control y como se emite el mensaje.

En ambos casos, los dispositivos se conectaron a pines digitales de la placa, en el caso del receptor, al recibir una señal IR transmite una salida en alto hacia la placa. En el caso del emisor, funciona con una salida PWM.

En el CD adjunto a la memoria se encuentra la hoja de datos del LED IR utilizado[9].

2.2.3. Módulo para Comunicación: ESP8266



(a) Placa de desarrollo NodeMCU v1.0 basada en la ESP8266 12E

(b) Placa ESP8266 12E

El módulo ESP8266 surge inicialmente como un adaptador SoC ("System on a Chip") de bus serial a WiFi para proyectos de hardware de bajo costo (Arduino, Raspberry Pi, etc) con un firmware que permitía, mediante comandos AT, configurar la red.

Se volvió muy popular debido a que el Chip es completamente programable, con lo cual se tiene un microcontrolador con conectividad WiFi y varios GPIO por un muy bajo costo (a partir de dos dólares).

El proyecto utiliza el kit llamado NodeMCU, basado en la ESP8266 12E. Éste tiene un formato compatible con la protoboard e incluye en el mismo PCB un conversor Serial a USB y un regulador de voltaje (para suministrar los 3V a los que funciona la ESP8266), facilitando el conexionado a la PC para cargar el firmware. Adicionalmente, la ESP 12E tiene 11 pines GPIO[10], lo que permite conectar varios

[9] CD-HEduc\Datasheets\LED-IR.pdf

[10] En el momento de iniciar el Proyecto, ésta era la ESP8266 con más pines GPIO disponibles, a la fecha de publicación ya existe en el mercado la placa ESP32 con mayor capacidad. Ver `https://espressif.com/en/products/hardware/esp32/overview`.

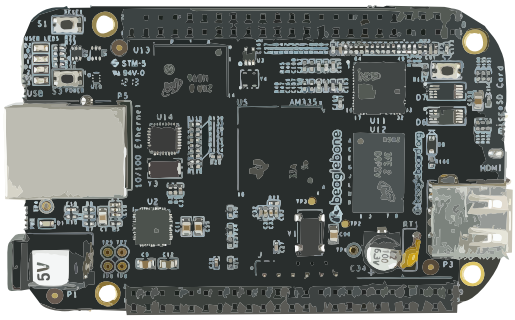


Figura 2.10: Placa de desarrollo BeagleBone Black; nodo central del sistema

sensores a una misma placa. Es destacable también que la ESP permite utilizar el protocolo I2C.

Las características de la ESP 12E son las siguientes:

- Frecuencia: 80MHz
- Memoria Flash / SRAM: 512KB / 64KB
- GPIO: 11 Digitales (11 PWM) + 1 Analógica
- Otras: WiFi 802.11 b/g/n, 1.5 UART, compatible con SPI/I2C

El Kit NodeMCU viene pre-configurado para programarlo en lenguaje "LUA", nativo para este hardware. Sin embargo a través del IDE de Arduino, con un plugin específico se puede cargar el firmware. Esta segunda opción resulta más ventajosa, debido a que permite utilizar el abanico de librerías disponibles en la comunidad Arduino.

Para esta aplicación, la placa ESP8266 resulta muy conveniente, ya que permite la distribución de los sensores en la habitación, comunicándose con el nodo central vía WiFi. A su vez, tiene un tamaño reducido, lo que permite en un futuro desarrollar los nodos periféricos en PCB a tamaños pequeños y es barato comparado con microcontroladores con características similares (WiFi, varios GPIO y capacidad de procesamiento).

2.2.4. Nodo Central del Sistema.

Definición previa: Servidor local o remoto

Un elemento a definir al inicio del proyecto era el lugar donde realizar el procesamiento de las variables, el modelado de sistemas y la toma de decisiones. Se analizaron dos posibilidades; en forma local o remota, alojado en servidores externos.

En un primer momento, se consideró la posibilidad de utilizar el entorno de Thingspeak[11]. Si bien la plataforma permite trabajar y desarrollar gráficos, históricos y funciones sobre los datos almacenados, así como actuar sobre el sistema,

[11] Ver `https://thingspeak.com/`>

requiere una conexión permanente a internet, que no es posible asegurar. En este caso en particular, se descartó el servidor remoto por esto último, ya que la intención es que este trabajo sea fácil de implementar en otros contextos. En caso de no contar con una buena conexión a internet, el uso de un servidor externo llevaría a que se pierdan las acciones coordinadas desde thingspeak hacia los actuadores del sistema. Implica problemas respecto al tipo de conexión, teniendo que utilizar protocolos de comunicación seguros entre el nodo (y la red) local y el servidor remoto. Además la plataforma thingspeak presenta limitaciones a la hora de operar con los datos obtenidos, lo que lleva a descartar su uso para realizar en ella los algoritmos de optimización. De todas maneras queda abierta la posiblidad de alojar datos ya procesados en la plataforma, con el fin de generar bases de datos.

En particular, esta última funcionalidad fue la utilizada para el registro de los valores de consumo y generación del sistema sobre los que luego se ejecuta la optimización.

Por otra parte, desarrollar el sistema en forma local supone las siguientes ventajas:

- Permite utilizar una red WiFi local, exclusiva para el sistema o no, independientemente de si la habitación en cuestión tiene un servicio de internet. Esto mejora la comunicación entre periféricos y central ya que reduce los elementos de comunicación del sistema a lo mínimo indispensable, disminuyendo la incidencia de fallas externas (no se requiere de proveedor de internet, conexión internacional para alcanzar el servidor, etc). Adicionalmente, resuelve posibles problemas de seguridad en la red local, utilizando protocolos WPA2 para autenticarse en la red de automatización, sin necesidad de protocolos SSL o similares para la comunicación vía internet.

- Permite a su vez definir el protocolo de comunicación sobre la red de manera especifica con el fin buscado, en este caso MQTT[12] (ver 4.2.2). De forma de disminuir el código en las ESP8266 (Módulos Periféricos; ver 2.2.3) en comparación al código necesario para comunicarse a través de los servidores de Thingspeak.

- Vuelve al sistema sencillo de implementar en habitaciones distintas. Considerando la posibilidad de que en un futuro se pueda utilizar en Viviendas u Oficinas, bastaría con cambiar la configuración de los nodos dispuestos para la nueva situación en la BeagleBone para adecuarla a las condiciones de ese nuevo entorno.

- Por último, la BeagleBone Black tiene un sistema operativo preinstalado basado en Linux llamado Angstrom, que incluye Python. Python es una herramienta muy versátil, sobre la cual es posible programar y ejecutar en tiempo real, tanto los modelos del sistema FV como las rutinas de optimización, además de configurar el broker de MQTT (ver 4.2.4)

[12]Message Queue Telemetry Transport, se explicará en detalle mas adelante este protocolo de comunicación.

BeagleBone Black como nodo central del sistema

La placa de desarrollo BeagleBone Black Figura 2.10 es el nodo central en la red de sensores inalámbricos planteado. Esta placa tiene la particularidad de contar con un sistema operativo Debian 8.6 "Jessie", el cual tiene las herramientas necesarias para el desarrollo del software necesario para esta aplicación. El lenguaje de programación base utilizado es Python, el cual viene incluido con Linux. Es un lenguaje de programación interpretado, de tipado dinámico y multiplataforma, y cuenta con una gran variedad de módulos y funcionalidades disponibles para usar.

El entorno para la programación es un IDE pre-instalado en la placa llamado Cloud9,[13] se trata de una plataforma para programación de código abierto, que se comunica con la placa utilizando el protocolo ssh[14].

Las principales características de la placa BeagleBone Black se detallan en la Tabla 2.1.

Procesador	ARM 1Ghz
RAM	512MB DDR3
Memoria	4GB on board exapndible vía Micro SD
Audio	Estéreo sobre HDMI
GPIO	65 pines
HDMI	Micro HDMI
Periféricos	Puerto USB y Ethernet 10/100
Alimentación	Micro USB o 5VDC

Tabla 2.1: Principales características de la Placa BeagleBone Black

Como la BeagleBone no tiene conexión WiFi nativa, para comunicarse con las ESP8266 se dispuso en este caso de un router Belkin AC1900 (cualquier router puede cumplir la función). La conexión entre la BeagleBone y el router se realizó mediante cable RJ45 y desde el router se generó la red de WiFi a la cual se conectan las ESP8266.

[13] ver `https://c9.io/`

[14] Secure SHell, es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red

Capítulo 3

Optimización

3.1. Introducción y caso de estudio

En este capítulo se introducen todos los conceptos y marco teórico referidos al problema de optimización que se trata en el proyecto. El sistema propuesto consta de un modelo de habitación con las siguientes características:

- Se supone una potencia instalada en la habitación de 2,2 kilowatts.
- La misma cuenta con 4 paneles solares fotovoltaicos que forman un arreglo con una potencia pico de generación de 1 kilowatt.
- Los paneles solares son utilizados, mediante un regulador de carga, para cargar un banco de almacenamiento a nivel 12v con una capacidad de 3,6 killowatts/hora.
- Se coloca a la salida del banco de almacenamiento un inversor de onda pura con una potencia de 2 kilowatts y funcionamiento MPPT[1].
- Se posee un mecanismo de control de forma de poder seleccionar el abastecimiento para el conjunto de cargas, ya sea desde la red de UTE como del conjunto baterías e inversor.
- Se considerará el costo de la tarifa triple horario que proporciona UTE (ver subsección 4.4.9), aunque el sistema permite ingresar una tarifa dinámica.

3.2. Objetivos puntuales del algoritmo de Optimización

Dentro de los objetivos definidos para el proyecto se encuentra el minimizar el gasto que genera el consumo eléctrico de la habitación. Para esto el algoritmo

[1]MPPT: Maximum Power Point Tracker, el sistema funciona extrayendo el máximo de potencia posible en cada momento. Para lograrlo, establece el nivel de tensión de continua del string de paneles para funcionar en el punto de máxima potencia. Se verá en detalle en la Fig 4.11a.

de optimización utilizado tiene como fin seleccionar desde dónde proviene el abastecimiento para las cargas de la habitación, siendo las opciones; desde baterías a través de un inversor de onda pura o desde la red de energía eléctrica.

Las entradas utilizadas como insumo para la toma de decisión son: la evolución horaria de la tarifa de consumo de energía desde la red (ver subsección 4.4.9), la energía disponible en batería (ver subsección 4.4.4), la curva de consumo de la habitación, y además el pronóstico a futuro para la generación (ver subsección 4.4.7).

Para esto se estudian dos métodos utilizados en el área de optimización: lógica difusa y programación dinámica. Dadas las características del caso de estudio es necesario un método que permita el modelado de variables físicas y que permita introducir las subjetividades propias del sistema, tales como los niveles de aporte de generación, el gasto en consumo y la energía restante en la batería.

3.3. Lógica difusa

En esta sección se presentarán los rasgos generales sobre lógica difusa.

La lógica difusa o "fuzzy logic" es una extensión de la lógica booleana habitual. Fue formulada inicialmente en el año 1965, por el ingeniero azerbaiyano Lofti Zadeh. La misma utiliza sets de pertenencia difusos, que se diferencian de los sets de pertenencia exactos (utilizados en lógica booleana) por no ser binarios. Los sets difusos permiten introducir nociones subjetivas o no totalmente completas, asociándoles una función de pertenencia a cada set.

Las funciones de pertenencia (o de membresía) son funciones continuas que toman valores entre cero y uno, correspondientes al grado de pertenencia de un elemento al conjunto. Esto expresa el carácter difuso del sistema, ya que toma relevancia en qué grado pertenece el elemento al conjunto, y no solamente si pertenece o no al mismo. Esta función es la que define la subjetividad o ambigüedad del conjunto o atributo.

Las reglas que vinculan a los sets de entrada con los sets de salida, siguen la heurísticas, de la forma *SI (antecedente) ENTONCES (consecuencia)*. Donde los antecedentes corresponden a la combinación de los sets de entrada, y la consecuencia al set de salida.

Para profundizar en los conceptos presentados anteriormente, se propone el siguiente ejemplo[2], utilizado para entender mejor la lógica difusa.

[2]Este ejemplo en particular, fue presentado para explicar a su vez, las herramientas que provee Matlab y Python para desarrollar controladores de lógica difusa. Ver sección 4.4.8

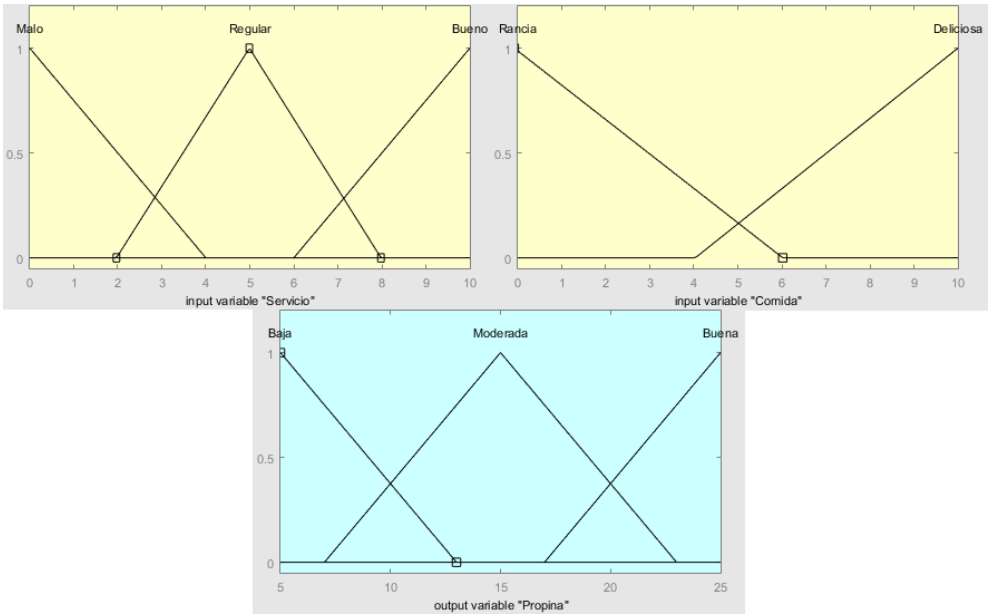


Figura 3.1: Las dos gráficas superiores corresponden a las funciones de pertenencia de entradas: servicio y comida. La gráfica inferior a la función de pertenencia de la salida.

Suponga que se va a cenar a un restaurante, y se plantea el problema de decidir cuál va a ser la propina que se le dará al mozo. Las dos variables de decisión que se presentan son: el buen servicio del mozo, y lo apetecible de la comida. El rango de la propina va de 5 % al 25 % del costo de la cena, clasificándose en baja, moderada y buena. El servicio del mozo se clasifica del 0 al 10, en malo, regular y bueno. La comida se clasifica del 0 al 10, en rancia o deliciosa. La figura 3.1 muestra las funciones de pertenencia para el servicio, la comida y la propina.

Las reglas que se proponen son:

- *Si* el servicio es malo *o* la comida es rancia *entonces* la propina es baja.
- *Si* el servicio es regular *entonces* la propina es moderada.
- *Si* el servicio es bueno *o* la comida es deliciosa *entonces* la propina es buena.

Se presentan las siguientes dos situaciones, ilustradas en la figura 3.2:

Imagine la primera situación. El servicio del mozo es bueno (servicio = 9) pero, si bien el plato principal estuvo delicioso, el postre no (lo cual a su gusto es desabrido, comida = 5). Entonces la propina es de $18,9\,\%$.

O imagine la segunda situación. La comida es deliciosa (comida = 9,5). Pero el servicio del mozo es malo (servicio = 0.5). Entonces la propina es de $15\,\%$.

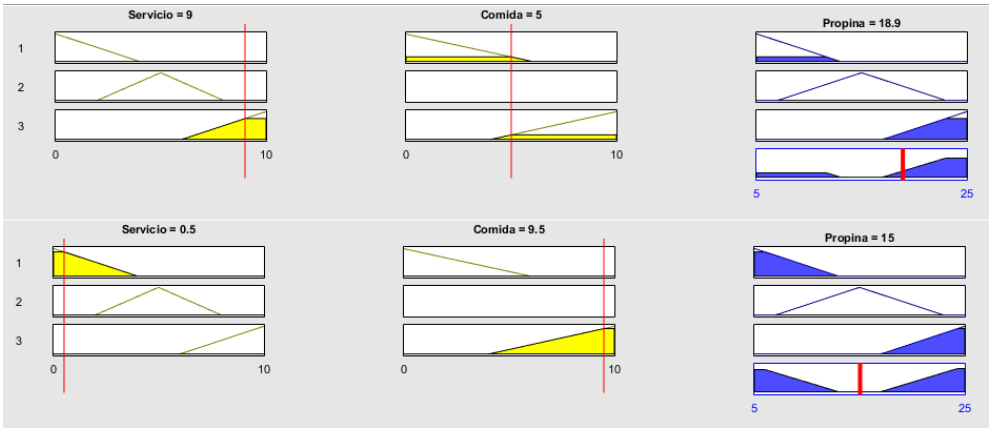


Figura 3.2: Situaciones del ejemplo; a la izquierda se presentan las entradas y a la derecha la salida.

Para concluir, se destaca que la operación para evaluar una entrada en los sets se denomina fusificación. Mientras que la operación para obtener la salida, luego de aplicar las reglas a las entradas, se llama defusificación. Obsérvese que la forma de desfusificar es hallando el centroide (el área azul se divide a la mitad por la linea roja).

3.4. Programación dinámica

En esta sección se establecen aspectos generales del algoritmo de programación dinámica.

El algoritmo de programación dinámica es un método de programación vinculado a la teoría de control, cuyo objetivo central el minimizar el costo (función matemática definida que rige el problema). Este costo como función matemática discretizada (ejecutada en pasos) tiene como variable de entrada el estado del sistema en el momento o paso actual. El algoritmo contiene, a su vez, la información del pasado relevante para el futuro del sistema, una variable de control definida para el momento, y también un parámetro aleatorio. Al estar afectada la función por un parámetro aleatorio, la salida de la misma no es totalmente determinística, lo que dota de una volatilidad importante al método.

El método tiene alta interrelación entre los distintos pasos temporales, sumado a la alta dependencia a los estados del pasado y que la perturbación también es ajustada por el pasado, hacen al sistema fuertemente realimentado[3].

[3]Como se menciona en el libro del ingeniero griego Dimitri Bertsekas "Dynamic programming and optimal control", el cual fue utilizado como base para estudiar la viabilidad de introducir este algoritmo al proyecto, "A key aspect of such situations is that decisions cannot be viewed in isolation since one must balance the desire for low cost with the indersirability of high future costs".

3.5. Elección del método a utilizar en el proyecto

Las características de la lógica difusa la hacen recomendable para aplicaciones en procesos altamente no lineales y con conocimiento no estrictamente definido, como es el caso del problema que se abarca en este proyecto.

Ademas de la opción de formar un algoritmo mas determinístico y estacionario, en el cual la riqueza del problema de optimización se obtiene dotando de mayor complejidad a los sets de pertenencia y a las reglas difusas que lo vinculan, fueron las razones que inclinaron la balanza a favor de la lógica difusa. Por lo expuesto se decidió su inclusión en el proyecto por encima de la programación dinámica.

También se desestima a la programación dinámica, por la necesidad de tener una gran base de datos, en búsqueda de hacer más fiel al sistema optimizado (dadas sus características de sistema realimentado y dependiente del pasado).

3.6. Análisis de antecedentes de la aplicación del método al problema

Luego de haber realizado el estudio general de los métodos, como se describe en la sección 3.5, se prosiguió a realizar una búsqueda de artículos que abordaran problemas similares al planteado en este proyecto [4], que utilizaran lógica difusa.

Fueron consultados en esta búsqueda los siguientes artículos:

- *Home Energy Management System based on Photovoltaic System.* Publicado por Leehter Yao, Chien-Chi Lai, Wei Hong Lim, del departamento de ingeniería eléctrica de la National Taipei University of Technology. [1]
- *Fuzzy Logic-Based Energy Management System Design for Residential Grid-Connected Microgrids.* Escrito por los miembros de la IEEE Diego Arcos-Aviles, Julio Pascual, Luis Marroyo, Pablo Sanchis y Francesc Guinjoan. [2]
- *DC Microgrid Power Coordination Based on Fuzzy Logic Control.* Publicado por Rashid Al Badwawi, Walid Issa, Tapas Mallick y Mohammad Abusara, pertenecientes al Departamento de Sostenibilidad y Medio Ambiente de la Universidad de Exeter. [3]
- *Energy Management Optimization of Integrated Generation Systems by Fuzzy Logic Control.* De los autores Francesco Bonanno y Giovanni Patani, del Departamento de Ingeniería Electrónica y Matemática Aplicada de la Universidad de Regio Calabria. [4]

[4]Similares en cuanto a optimización de recursos relacionados con energías renovables, utilización de tarifas diferenciadas en torno al horario del día, uso de banco de almacenamiento y realización de predicciones del aporte energético de generación

- *Optimal Fuzzy Logic EMS Design for Residential Grid-Connected Microgrid with Hybrid Renewable Generation and Storage.* Publicado por D. Arcos-Aviles (Universidad de las Fuerzas Armadas ESPE Sangolquí Ecuador), J. Pascual, L. Marroyo, P. Sanchis (Universidad Pública de Navarra), F. Guinjoan y Martin P. Marietta (Universidad Politécnica de Catalunya). [5]
- *Fuzzy Logic Approach for Short Term Solar Energy Forecasting.* Artículo perteneciente a Ayushi Chugh (Gautam Buddha University), Priyanka Chaudhary (Delhi Technological University) y M. Rizwan ((Delhi Technological University). [6]
- *Energy Management Strategy for a grid-tied Residential Microgrid based on Fuzzy Logic and Power Forecasting.* Publicado en colaboración por los siguientes autores D. Arcos-Aviles (Universidad de las Fuerzas Armadas ESPE Sangolquí Ecuador), J. Pascual, L. Marroyo, P. Sanchis (Universidad Pública de Navarra), F. Guinjoan y Martin P. Marietta (Universidad Politécnica de Catalunya). [7]

3.6.1. Artículo: *Home Energy Management System based on Photovoltaic System*

El estudio del artículo referido propone realizar un sistema de gestión de la energía utilizando lógica difusa. Se plantea una estrategia de respuesta de la demanda y de despacho de energía para un sistema con tarifa de electricidad dinámica y aporte de energía fotovoltaica.

Se define el concepto de respuesta de la demanda como los cambios en el patrón de consumo de electricidad por los usuarios en respuesta a: cambios en el precio de la energía en el tiempo, el pago de incentivos para reducir el consumo en momentos de pico de consumo, o cuando el funcionamiento del sistema se ve comprometido.

El hecho de controlar la respuesta de la demanda produce ventajas tanto para los usuarios finales como para los agentes encargados de generar y distribuir la energía eléctrica. Para los usuarios finales las ventajas son fundamentalmente el hecho de utilizar tarifas de menor costo. Para los agentes, las ventajas son el hecho de liberar capacidad en transformadores de distribución, bajar pérdidas y tener que abastecer un pico de demanda menor.

Como generalmente se produce desfasaje entre el periodo donde se produce el pico de generación y el rango donde la tarifa es más alta, se establece la necesidad de gestionar el almacenamiento, de forma de asegurar que la energía se vuelque al consumo en el horario más conveniente.

Como paréntesis, el artículo hace énfasis en el concepto IoT (Internet of Things). Destacando la capacidad de proporcionar información y de realizar interacciones en tiempo real.

La red está dividida en tres capas:

- La capa física: en donde se ubican los elementos de medida, utilizados para obtener la información referente al estado de carga del almacenamiento, tarifa del servicio eléctrico, potencia generada por los paneles solares y consumo de energía del conjunto de cargas.
- La capa de red: encargada de la transmisión de información entre los componentes del sistema. En este caso combina comunicaciones tanto inalámbricas, como cableadas.
- La capa de aplicación: cumpliendo el fin de almacenar, analizar y procesar todo el flujo de información. Ésta usará un control con lógica difusa para decidir sobre la estrategia de despacho de energía.

Retornando con el sistema de gestión de la energía hogareña, los autores plantean las siguientes ventajas de la lógica difusa, frente a otras técnicas de optimización:

- La lógica difusa es una mejor herramienta para modelar problemas complejos, dada su capacidad para emular el razonamiento humano y la toma de decisiones.
- Un controlador fuzzy proporciona una solución más robusta, dado que puede responder a las ambigüedades del lenguaje humano, las cuales no pueden ser representadas en términos precisos y discretos.

El controlador fuzzy consiste en tres módulos:

- El primero son las funciones de membresía de las entradas. Éstas son: el estado del almacenamiento de energía, el consumo de las cargas y la tarifa. Las funciones de membresía se clasifican en: muy bajo, bajo, medio, alto, y muy alto.
- El segundo es la funcion de membresía para la salida. En este caso es: la cantidad de energía que debe ser despachada desde la red. Nuevamente las funciones se clasifican en: muy bajo, bajo, medio, alto, y muy alto.
- El último módulo corresponde a las reglas; el sistema utiliza las 125 reglas posibles[5].

El caso de estudio planteado en el artículo fue el siguiente: la capacidad de almacenamiento fue de 48V 300Ah, la red fue de 110V; la tarifa fue establecida en un rango dinámico; como fuente de energía renovable se tiene un arreglo de

[5]Éstas se establecen al definir un sistema con tres entradas, cada una clasificada en cinco posibles niveles. Entonces hay 5^3 posibles reglas.

paneles fotovoltaico con una potencia de pico es 4KW; y por último, la potencia instalada es 3KW.

Se prueban días contiguos, un día soleado y uno nublado. Los resultados fueron los siguientes:

- Se observa que durante el primer día (precisamente desde las 6:45 hasta 16:30), el uso de energía de la red es bajo dado que el aporte solar es suficiente. También durante este período el almacenamiento de energía va aumentando gradualmente (pasando de un 70 % a un 100 %).
- Pasado dicho período hay unas tres horas en las que el precio de la tarifa toca su techo, donde además la demanda crece. Es aquí donde se da el aporte diferencial del sistema, dado que el mismo tiene energía suficiente para abastecer el consumo (disminuye el almacenamiento de energía a un 15 %).
- Por último, en el horario comprendido entre las 20:15 y las 04:30 del día siguiente, la cantidad de energía demandada a la red aumenta, dado que no hay generación. Pero no hay mayores consecuencias, ya que el precio de la tarifa es medio/bajo (entre un 50 % y un 70 % más barato que en la hora pico de tarifa).

3.6.2. Artículo: *Energy Management Strategy for a grid-tied Residential Microgrid based on Fuzzy Logic and Power Forecasting*.

En este artículo se introduce un sistema híbrido con aporte fotovoltaico y eólico. El estudio se centra en un escenario ON-grid con una red pequeña, persiguiendo el objetivo de controlar la cantidad de energía demandada a la red, para disminuir la factura de electricidad y también las eventuales sobrecargas del sistema.

El sistema está modelado como: un aporte de generación fotovoltaica de 6kWp; una turbina eólica de 6KW; un banco de almacenamiento de 72kWh; y un conjunto de cargas con una potencia máxima de 7kW.

Predicciones de generación y consumo:

- El pronóstico de generación solar y eólico fue realizado utilizando las variables climáticas provenientes de un portal de pronóstico. Se realiza una curva de generación esperada para todo el día, en el artículo se presentan las diferencias entre la generación esperada y la efectivamente medida. El error obtenido, salvo pequeños picos cortos en el tiempo, se mantiene dentro de los márgenes aceptables definidos por los autores.
- Respecto del consumo esperado, el sistema asume que la curva de consumo diaria será idéntica a la del día anterior. Para ésto toma como insumo el consumo que tuvieron las cargas, el cual es medido en el sistema.

La estrategia de control del controlador de lógica difusa consiste en medir el error de potencia generada que abastecería al banco de almacenamiento y el pronóstico para las tres horas previas, con la finalidad de mantener el banco en un 75 % de su carga máxima.

Entonces la potencia de red es la suma de:

- La potencia generada por el sistema. Se llega a esta potencia con el promedio de la potencia generada en las 12 horas anteriores y el pronóstico de generación en las 12 horas siguientes.
- La potencia consumida en ese momento. De forma similar consulta el consumo del pronóstico planteado.
- De la potencia que pueda otorgar o recibir la batería, en base a la decisión del controlador fuzzy.

El bloque de control por lógica difusa está compuesto por dos entradas, una salida y veinticinco reglas. Las funciones de membresía de ambas entradas consisten en 5 funciones triangulares, que recorren todo el rango posible para cada una de las mismas. Por otra parte, la salida contiene 9 funciones triangulares en el intervalo señalado.

Las simulaciones fueron realizadas a partir de datos reales tomados en un año en la microgrid de la Universidad Pública de Navarra.

Los resultados de las simulaciones muestran que la demanda de energía a la red permanece dentro de valores bajos (menores a 2kW de valor absoluto). Esto es consecuencia de un equilibrio entre la potencia de generación y de las cargas, además del hecho de que se logra mantener el banco siempre cercano al 75 %, de manera que siempre haya disponibilidad de compensar las posibles fluctuaciones del sistema.

3.6.3. Artículo: *Optimal Fuzzy Logic EMS Design for Residential Grid-Connected Microgrid with Hybrid Renewable Generation and Storage*

Como el artículo resumido en la subsección 3.6.2, en este caso se estudia un sistema conectado a la red el cual contiene aportes de generación fotovoltaica 4KWp, generación eólica 6KW de potencia instalada y un banco de almacenamiento de 72KWh (aunque por motivos de preservarlo solo se utilizará el 50 %), los mismos funcionan con un sistema de cargas que tiene una potencia de 7KW.

Dado que el objetivo principal de implementar un control con lógica difusa es lograr una red con bajas fluctuaciones, se estudia para establecer un punto de comparación, un sistema conectado a la red sin banco de almacenamiento y por

tanto, sin control de energía. Se obtiene como resultado que la red tiene altas fluctuaciones (el máximo valor demandado a la red es de 5,75KW y el máximo valor volcado es 6,45KW, esto teniendo en cuenta el conjunto de cargas es de 7KW, por lo tanto el suministro puede experimentar una potencia demandada máxima de 7KW).

El sistema utiliza en este caso un filtro pasabajo para eliminar los componentes de alta frecuencia de la potencia de la red. Y luego aplica una estrategia SMA (media móvil simple) con una ventana de un día, bajando las fluctuaciones de la potencia un 18 %. El rango de fluctuaciones para la potencia siguen siendo altas (con máximos de 4,71KW y 2,40KW respectivamente).

El sistema difuso consta de 2 entradas, una es la energía del banco de almacenamiento y la otra es la derivada de la potencia promedio que se obtiene al aplicar la estrategia SMA. Ambas son clasificadas como funciones de membresía en 5 rangos: negativo alto, negativo bajo, cero, positivo bajo y positivo alto, de los cuales los 3 intermedios son triangulares y los 2 ubicados en los extremos del rango son trapezoidales. La salida es calculada utilizando las 25 reglas posibles y está valorada bajo 9 funciones de membresía posibles.

Aplicado el controlador fuzzy se obtiene que las fluctuaciones bajan más que aplicando la estrategia SMA (comparado al sistema sin control, se reduce un 67 %). Con máximos de 1,89KW para la potencia demandada y 2,04KW para la potencia entregada a la red.

3.6.4. Artículo: *DC Microgrid Power Coordination Based on Fuzzy Logic Control*.

En este artículo se analiza la implementación de un sistema fotovoltaico con banco de almacenamiento. El control con lógica difusa realizado tiene la particularidad de intervenir en el sistema, sacando de funcionamiento cargas que son consideradas de baja prioridad. Otro aspecto del trabajo realizado es que trabaja con una red en DC, a diferencia de los demás artículos analizados.

Para hacer mas simple el control con lógica difusa, el mismo se divide en dos subsistemas, el primero controla el umbral superior del estado de carga (SOC) y la potencia de entrada a las baterías, mientras que el otro controla que no se disminuya del SOC mínimo y que la potencia demandada por las cargas sea menor que el máximo establecido.

Las entradas del primer sistema son: la diferencia entre el SOC actual y el SOC máximo y la diferencia entre la potencia suministrada desde el sistema renovable y la potencia de entrada máxima. La salida de este sistema es una señal que controla la potencia del sistema fotovoltaico a través de un power shifter implementado en

el sistema.

Las entradas del segundo sistema son: la diferencia entre el SOC y la potencia saliente del almacenamiento para el momento actual; y sus umbrales definidos. La señal de salida tiene como fin sacar cargas de funcionamiento, para este fin las cargas se agrupan en tres categorías (vitales, esenciales y normales), sobre las que el control puede tomar decisión. Es decir, puede sacar de funcionamiento todo el grupo de cargas normales, luego, de ser necesario todo el grupo de cargas esenciales y por ultimo deshabilitar el grupo de cargas vitales. Los umbrales definidos para el sistema permiten al banco de baterías moverse entre el 95 % y el 40 % de su capacidad y limitan tanto la potencia de entrada como de salida en 1000W (la potencia del conjunto de cargas es 1140W).

Las funciones de membresía tanto para las entradas y salidas de ambos subsistemas contemplan niveles bajo, medio y alto para cada variable. Siendo el nivel medio una función triangular y los valores bajos y altos una función trapezoidal. La única excepción es la salida del segundo subsistema que se compone de 3 funciones triangulares.

Para realizar la prueba de dicho sistema, se realizan simulaciones partiendo de diferentes valor para el estado de carga del almacenamiento, donde presentan 4 casos con el SOC tomando los valores 95 %, 80 %, 50 % y 42,88 %. Comprobando que el sistema realice una buena gestión de los grupos de cargas y de la potencia demandada al sistema de energía solar. Según los resultados exhibidos el funcionamiento del bloque de control es correcto, manteniendo el nivel de batería dentro de los margenes deseados.

3.6.5. Artículo: *Fuzzy Logic Approach for Short Term Solar Energy Forecasting*.

Este artículo se centra en las posibilidades de realizar un pronostico de una hora adelante de la energía que un sistema FV aportará.

El sistema se centra en los valores de irradiancia de las horas anteriores para la realización del pronostico y tiene 4 entradas, correspondientes a los valores de irradiancia medidos en W/m^2. Para el análisis difuso las entradas son normalizadas y ubicadas en el rango 0,1 a 0,9 normalizado y luego son calificadas en 6 rangos (muy bajo, bajo, medio, medio alto, alto y muy alto).

El estudio fue realizado en base a un mes de muestras y los resultados exhibidos muestran que el error en términos absolutos lo largo del mes fue de 1,052 %, el cual se plantea aceptable. Por lo que los autores concluyen que realizar un pronostico basado en lógica difusa es viable y efectivo.

3.6.6. Artículo: *Fuzzy Logic-Based Energy Management System Design for Residential Grid-Connected Microgrids*.

En este artículo se proporciona un sistema de gestión de energía en una red microgrid que incluye fuentes de generación renovables y capacidad de almacenamiento[6]. En este caso el sistema asume que no es posible controlar ni la generación, ni la demanda de energía. El objetivo de la investigación es implementar un sistema de control con lógica difusa, que permita reducir las fluctuaciones de potencia en la red, mientras se mantiene el estado de carga del almacenamiento dentro de los niveles seguros.

El trabajo implementa un sistema de control por lógica difusa con 2 entradas y una sola salida, el cual consiste de 25 reglas. El sistema tiene además la particularidad de introducir como entradas el ratio de cambio de la energía en la microgrid, como una forma de anticipar el comportamiento del sistema.

El sistema de control fue implementado en la práctica, para este fin fue utilizada la microgrid residencial de la Universidad de Navarra. La misma cuenta con un arreglo FV de 4kWp, una turbina eólica de 6KW, mientras que el sistema de cargas tiene una potencia instalada de 7KW, para el almacenamiento se cuentan con baterías con una capacidad de 72kWh. Para comprobar el éxito en el sistema se realizaron comparaciones con otros métodos mediante simulaciones.

El sistema de control propuesto funciona de la siguiente manera: un bloque de control difuso recibe como entradas el estado de carga del almacenamiento y la pendiente de la potencia en los últimos dos tiempos de muestra. La salida de este bloque es un valor llamado P_{flc} que está destinado a modificar el valor de energía entregado o demandado a la red en orden de abastecer la demanda y mantener el almacenamiento por encima del nivel mínimo aceptable. El bloque de control genera las funciones de membresía para las entradas, catalogándolas en cinco categorías, cuyo grado de pertenencia a cada una está distribuido dentro del rango de acción de la entrada. Los mismos son triangulares para los atributos ubicados en el centro del rango y trapezoidales para los ubicados en los dos extremos. La salida está catalogada del mismo modo, pero con nueve atributos posibles en vez de cinco.

La tipología del sistema da lugar a 25 reglas, las cuales son utilizadas en su totalidad. Para caracterizar el resultado del sistema se tienen en cuenta los picos máximos demandados y entregados a la red y el valor máximo de la derivada de la potencia. Según los resultados presentados, en todo los valores se logran mejoras sustanciales aplicando este método.

[6]Se define microgrid como un sistema de bajo voltaje compuesto de cargas, generación distribuida y bancos de almacenamiento que se puede conectar a la red de abastecimiento en un solo punto.

3.6.7. Artículo: "Energy Management Optimization of Integrated Generation Systems by Fuzzy Logic Control"

Este artículo incluye un sistema con generadores a gasolina (particularmente 2 con potencias de 30KVA y 50KVA respectivamente), junto con turbinas eólicas con una potencia total de 50KW, y un sistema fotovoltaico con 29KW de potencia de pico. Por último el sistema cuenta con un banco de almacenamiento de 135KWh y una carga de desvío. El hecho de contar con generadores diésel en el sistema abre un abanico de posibilidades.

En cuanto al control que se puede realizar sobre los generadores diésel, se destacan:

- La regulación de la potencia que pueden entregar.
- La minimización de la cantidad de ciclos de arranque (lo que ayuda a alargar su vida útil).
- La programación de las tareas de mantenimiento, en los momentos en que la demanda esté cubierta por las energías renovables.

El proyecto propone un sistema de control por lógica difusa, que consta de 4 entradas: potencia eólica, potencia fotovoltaica, estado de carga del almacenamiento y potencia demandada por las cargas. Y 4 salidas: potencia en cada uno de los dos generadores, potencia intercambiada con el almacenamiento y potencia enviada a la carga de desvío.

El sistema utiliza funciones de membresía triangulares y utiliza 130 reglas, las cuales se apoyan en los siguientes 4 puntos:

- Siempre que sea posible, abastecer las cargas desde la energía renovable generada.
- Estrategia de generadores diésel, pensada en reducir el consumo de combustible y minimizar la cantidad de ciclos de arranque.
- Evaluar el estado de carga del almacenamiento.
- Gestionar la operación de carga y descarga de la batería.

Para realizar las simulaciones del sistema, se trabaja con las tres entradas de potencia en el sistema, cada pasos de una hora. Para ésto, se tomaron mediciones durante una temporada de primavera en una isla del mediterráneo. Luego se realizaron simulaciones, con el fin de generar los perfiles de potencia entregada por las fuentes de energía renovable, y para la potencia demandada por el conjunto de cargas. Así, el sistema trabaja con los valores de velocidad de viento y radiación para cada hora, devolviendo la potencia generada.

Las baterías comienzan con un estado de carga de 100 % al inicio de la simulación, y se limita la cantidad de energía que puede ser extraída en cada paso de una hora en 10 % de la capacidad.

Los resultados de las simulaciones proporcionan información precisa acerca de valores relevantes para el sistema, como lo son: la energía disponible proveniente de cada fuente; el consumo de combustible; la cantidad de ciclos de arranque de los generadores; la generación de energía renovable en el sistema; así como el flujo de energía en el almacenamiento y estado de carga del mismo. Los resultados exhibidos muestran una disminución del consumo de aceite de 5 % y una disminución de la cantidad de ciclos de un 22 %.

3.6.8. Conclusiones de la investigación

La investigación realizada confirmó el uso de la lógica difusa como herramienta de optimización para este proyecto. A continuación se detallan las conclusiones relevantes:

- **Es conveniente el uso de la lógica difusa para la gestión de sistemas con energías renovables.**

  La primera conclusión de la investigación remarca la cantidad de problemas de gestión de sistemas con energías renovables que implementan la lógica difusa. Observando en los artículos la diversidad de tipologías tratadas, siendo el único aspecto fundamental para todos los casos el hecho de contar con banco de almacenamiento.

- **La lógica difusa permite aplicarse para variedad de objetivos.**

  El segundo objetivo remarca la variabilidad de los objetivos planteados en los artículos, los cuales son resueltos utilizando lógica difusa. Entre los más destacados se encuentran: minimizar el costo económico del abastecimiento; lograr un uso inteligente del almacenamiento prolongando su vida útil; gestionar cargas del sistema haciendo uso de estrategias de respuesta a la demanda o cargas de desvío.

- **Combinación de controladores con la lógica difusa.**

  Respecto del uso de la lógica difusa se observa que se opta en su mayoría por sistemas con pocas entradas (en general no más de cuatro) y una salida. La cantidad de reglas utilizadas es, en varios casos, la totalidad de las generadas por las funciones de membresía.

  Se observa también que, para afrontar los problemas de gestión se combinan diferentes bloques de control con la lógica difusa, por ejemplo, utilizando otras estrategias como filtro pasabajos y estimadores.

  Esta última conclusión fue una de las razones que motivó la elección de combinar la lógica difusa con otro tipo de algoritmos, como se explica en la sección 3.7.2, se utilizó un árbol de decisión.

3.7. Algoritmo Heduc de optimización

3.7.1. Introducción

Se presenta en esta sección la implementación del algoritmo para este proyecto. Como se mencionó, el propósito del algoritmo es controlar, de forma inteligente, el uso de la energía de la batería del sistema. Para eso se definieron las siguientes entradas al algoritmo: el estado actual de la batería, los pronósticos de generación solar y consumo del hogar para el siguiente intervalo de tiempo y la tarifa (en este caso, la tarifa triple horario).

Otra definición a tomar fue la frecuencia con que el algoritmo se iba a ejecutar, definiéndose dos intervalos de tiempo, uno de quince minutos y otro de una hora. Los fundamentos de por qué aplicar uno u otro se aclaran en las secciones siguientes, pero se justifica la cota superior de una hora para acotar el error de la ejecución del problema en una transición de la tarifa, y porque los pronósticos tienden a ser menos precisos en períodos más largos.

3.7.2. Árbol temporal Heduc

El controlador fuzzy Heduc recibe el estado actual de la batería, el pronóstico de generación, consumo en los próximos 15 minutos y la tarifa en ese momento. Sobre esas variables decide con qué posibilidad tomaría la energía de la batería para ese intervalo de tiempo.

Una vez obtenido el resultado numérico a la salida del bloque Heduc fuzzy, no se lo defusifica, sino que se opera con este valor variando entre 0 y 100 para obtener las respectivas posibilidades, donde la posiblidad de consumir de batería coincide con este número de salida y la posibilidad de consumir de red es el complemento del intervalo de entrada [7]

Por la forma como fue definido el controlador fuzzy, se observó que la decisión se basa en el primer intervalo de tiempo, perdiendo la posibilidad de tomar una decisión mas eficiente observando los siguientes intervalos de tiempo. Se presenta un ejemplo de ésto a continuación.

Si nos situaramos en el nodo inicial de la figura 3.3 se tiene una carga media en la batería, en un momento donde el costo está en su franja media y el consumo es alto, el controlador fuzzy indicaría consumir de batería, ya que es la opción más eficiente (el controlador fuzzy indica la posibilidad $\mathbf{p_B}$). Pero esta decisión podría no ser la más eficiente mirando hacia adelante. Si nos movemos hacia el nodo B del primer piso de la figura, en el siguiente intervalo la tarifa se sitúa alta, y el consumo se mantiene. En la situación anterior, el controlador fuzzy indicaría

[7] para poner en detalle, la posibilidad de consumir de red se obtiene como 100 menos el valor de salida del bloque fuzzy.

p_B	60 %	p_{BB}	40 %	$p_B \cdot p_{BB}$	24 %
		p_{BR}	60 %	$p_B \cdot p_{BR}$	36 %
p_R	40 %	p_{RB}	95 %	$p_R \cdot p_{RB}$	38 %
		p_{RR}	5 %	$p_R \cdot p_{RR}$	2 %

Tabla 3.1: Tabla para el primer ejemplo, donde p_B, p_R, p_{BR}, p_{RB} son las posibilidades resultantes del controlador fuzzy.

consumir de red (posibilidad de $\mathbf{p_{BR}}$), decisión tomada porque la batería no logra abastecer todo el consumo.

Si en lugar de consumir de batería en el primer intervalo, se hubiese consumido de red, en el segundo intervalo nos encontraríamos en el nodo R de la figura, donde el controlador fuzzy indicaría consumir de batería (posibilidad de $\mathbf{p_{RB}}$). Como se ve en la tabla 3.1 la opción más eficiente es no consumir de batería en el primer intervalo de tiempo, para consumirla en el segundo ($\mathbf{p_{RB}}$).

Analizando el ejemplo, si se observan dos intervalos de tiempo, ponderando las dos situaciones, la opción de decidir no utilizar batería en el primer intervalo de tiempo y utilizarlo en el segundo es del $\mathbf{p_R \cdot p_{RB}}$ contra un $\mathbf{p_B \cdot p_{BR}}$ de decidir lo contrario.

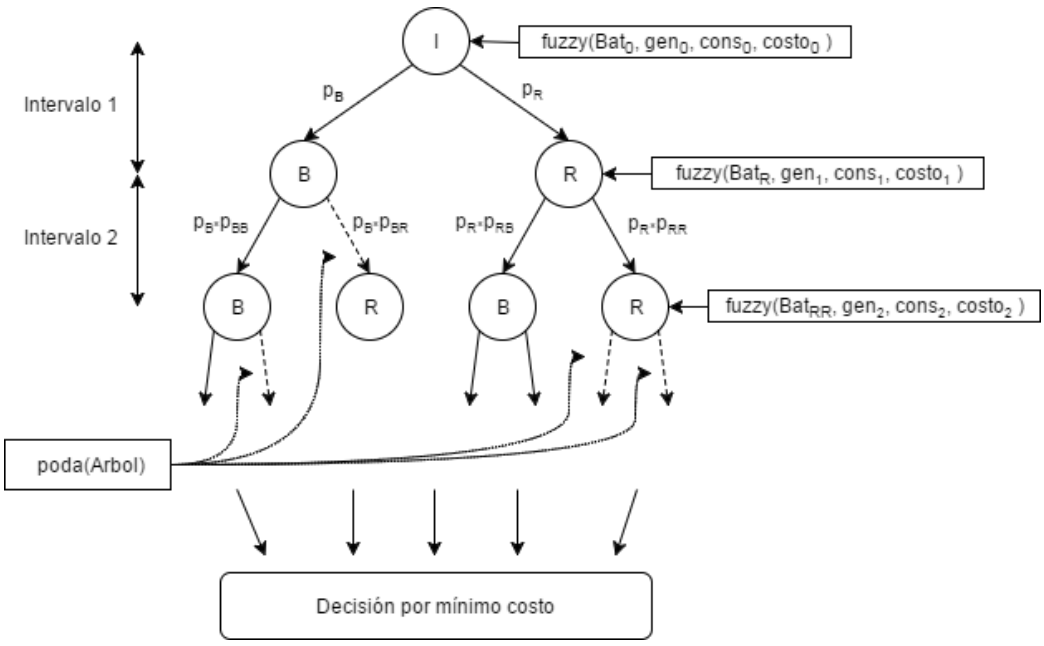


Figura 3.3: Árbol de decisión presentado para el algoritmo implementado en el proyecto. Se observan las funciones poda y decisión de costo.

Para mitigar esta situación, se propuso la implementación de un árbol, el cual

se construye a través de decidir si se utiliza o no batería, observando su evolución en varios intervalos de tiempo. La figura 3.3 muestra un árbol de tres intervalos de tiempo donde en cada nodo de éste se ejecuta el controlador fuzzy, el cual recibe como entrada a la energía generada y consumida en el próximo intervalo de tiempo, el costo del kWh en ese momento, y el estado de la batería con el que el sistema llegó a ese nodo.

Se observa también una función llamada **poda** y el método de decisión para elegir la rama a seguir, los cuales se explican en la siguiente sección.

3.7.3. Función de Poda y decisión por mínimo costo

La función de poda se encarga de eliminar aquellos nodos que tienen una probabilidad muy pequeña, comparada con el nodo de mayor probabilidad. Observando la figura 3.3, si la probabilidad $p_B \cdot p_{BR} \leqslant x_{\%} \cdot \mathbf{max}(p_B \cdot p_{BB}, p_R \cdot p_{RB}, p_R \cdot p_{RR})$ ($x_{\%}$ corresponde al umbral de decisión), entonces el nodo *IBR* se elimina de los posibles candidatos, descartando las ramas que pudieran surgir de él.

Adicionalmente, la función está pensada para economizar recursos, eliminando aquellos nodos que comienzan con una probabilidad muy baja en el entendido de que no podrán generar nodos hijos que puedan ser candidatos a elegirse. En particular, se da que dentro de las ramas de baja probabilidad se encuentran las que resultan menos costosas, ya que contemplan, por ejemplo, consumir siempre de baterías. Ésto no es viable, debido a que la energía almacenada en las mismas es menor al total consumido en la suma de los intervalos, con lo cual se descartan antes de tomar la decisión por costo.

El árbol de decisión termina de crearse luego del último intervalo de tiempo considerado, la cantidad máxima de nodos es de 2^n donde n es el número de dichos intervalos. Sobre los últimos nodos obtenidos, filtrados por la función poda[8], se toma la decisión en base al mínimo costo, esto es, eligiendo el nodo cuyo costo acumulado a lo largo de los n intervalos sea el menor.

p_B	52 %	p_{BB}	23 %	$p_B \cdot p_{BB}$	12 %
		p_{BR}	77 %	$p_B \cdot p_{BR}$	40 %
p_R	48 %	p_{RB}	83 %	$p_R \cdot p_{RB}$	40 %
		p_{RR}	17 %	$p_R \cdot p_{RR}$	8 %

Tabla 3.2: Tabla para el segundo ejemplo presentado, donde p_B, p_R, p_{BR}, p_{RB} son las probabilidades con las que sale el controlador fuzzy.

Se presenta un ejemplo a continuación, el cual por hipótesis en los dos intervalos de tiempo presentados tiene: consumos similares, un costo medio en los dos y la batería sólo puede abastecer a uno de los consumos. En la tabla 3.2 se muestran

[8]La función poda considera la probabilidad ponderada a lo largo de los n intervalos

los datos de este ejemplo, si se elige $x_{\%} = 1/3$ la función poda elimina a los nodos con la probabilidad ponderada $p_B \cdot p_{BB}$ e $p_R \cdot p_{RR}$.

Observando los porcentajes ponderados de los nodos *IBR* e *IRB*, se llega a que la probabilidad de consumir energía desde baterías en el primer intervalo de tiempo o en el segundo es igual, entonces, aplicando la decisión por mínimo costo se opta por el más económico.

Como ultima observación, si la poda no se ejecutaba, el nodo elegido por mínimo costo hubiera sido el *IBB* (consumiendo desde baterías en los dos intervalos), violando las hipótesis del ejemplo.

3.8. Heduc fuzzy

3.8.1. Introducción

En la presente sección se explica el diseño del sistema fuzzy que se implementó para la optimización de este proyecto. Primero se van a definir las funciones de membresía de las entradas que atienden a las variables: estado de la batería (de ahora en mas *batería*), predicción en el próximo intervalo de tiempo de consumo (*consumo*) y costo del kWh(*costo*), y los resultados del estado de la batería más la generación(*batGen*) y el estado de la batería más la generación menos el consumo(*batGenCon*).

Luego se explicará la función de membresía de la salida que define si se utiliza la energía almacenada en baterías o no. Y finalmente se van a definir las reglas que involucran a estas variables.

3.8.2. Funciones de membresía para las entradas y la salida

La figura 3.4 muestra a las funciones de membresía de la *batería* y del *consumo*, en particular, se observan las funciones para una entrada baja, media o alta. El rango de entrada de la *batería* es de 3600 Wh, mientras que en el *consumo* es de 2000 Wh.

La figura 3.5 muestra a las funciones de membresía de las entradas *batGen* y *costo*, se observan también las funciones diferenciales para cada entrada (baja, media y alta). El rango de *batGen* es de 5100 Wh (suma de la capacidad más la máxima generación) y en el caso del *costo* es de 9 pesos uruguayos [9].

La figura 3.6 muestra las funciones de membresía de la entrada *batGenCon* y de la salida del sistema. Se definieron para la entrada dos funciones, las cuales fuerzan a la batería a salir de servicio en caso que quede por debajo del umbral de descarga máxima o a que se consuma de ella en caso que la generación prevista sea mayor al margen disponible de almacenamiento. La salida de las reglas tiene cinco funciones de membresía, tres son salida baja, media y alta, que acompañan

[9]para cubrir las tres franjas de la tarifa tripe horario; la máxima es $ 8,507

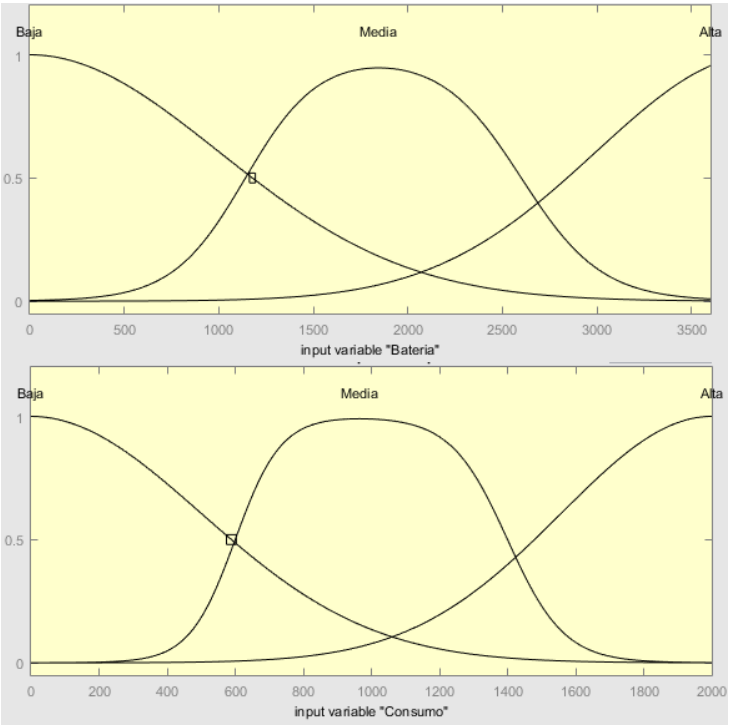


Figura 3.4: Funciones de membresía de la batería y el consumo.

a las reglas formuladas con el cruzamiento del *costo* con las tres primeras entradas presentadas, y otras dos funciones de membresía que acompañan a las reglas formuladas con la última entrada presentada.

Las funciones de membresía de las tres primeras entradas se implementaron con las funciones gaussiana y diferencial sigmoide[10]. Las funciones de membresía de las dos últimas entradas utilizaron funciones trapezoidales. La salida implementó todas las mencionadas anteriormente.

3.8.3. Reglas

A continuación se presentan las reglas que involucran a las entradas y la salida. En total son once reglas: las primeras seis involucran a la *batería* y *batGen* contra el *costo*;las siguientes tres reglas involucran *consumo* contra *costo* y la *batGenCon*; las últimas dos involucran solo a la entrada *batGenCon*.

1. ***if*** *batería* **is** Alta **&** *costo* **is** Alto ***then*** *salida* = Alta
2. ***if*** *batería* **is** Media **&** *costo* **is** Medio ***then*** *salida* = Media
3. ***if*** *batería* **is** Baja **&** *costo* **is** Bajo ***then*** *salida* = Baja

[10]Ambas funciones se explican en el Capítulo 4.4.8

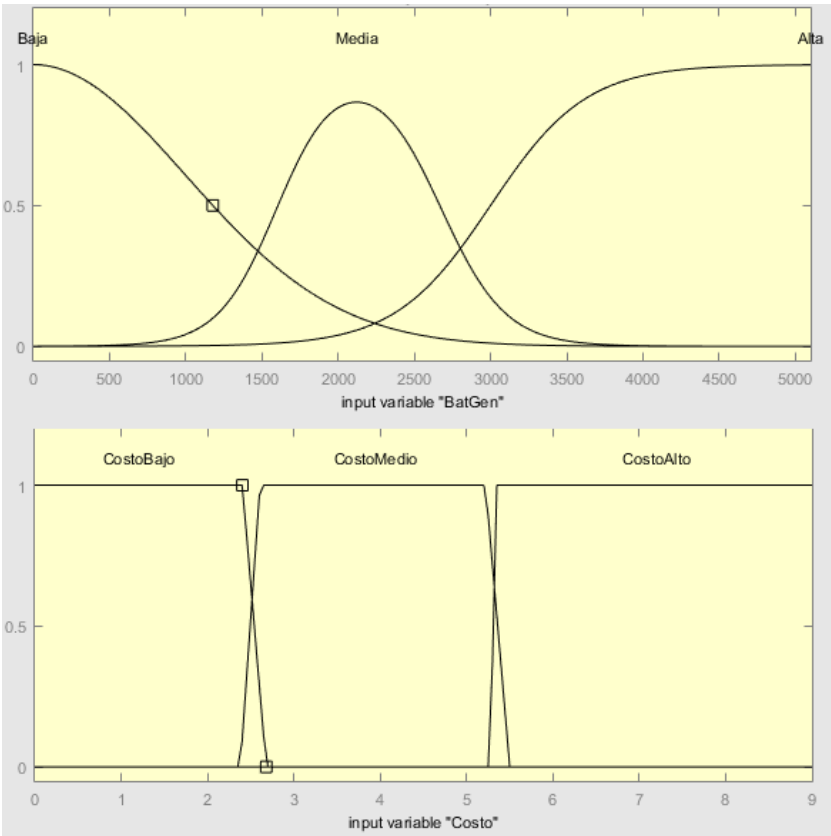


Figura 3.5: Funciones de membresía de la batGen y el Costo.

4. ***if*** *batGen* **is** Alta **&** *costo* **is** Alto ***then*** *salida* = Alta
5. ***if*** *batGen* **is** Media **&** *costo* **is** Medio ***then*** *salida* = Media
6. ***if*** *batGen* **is** Baja **&** *costo* **is** Bajo ***then*** *salida* = Baja
7. ***if*** *consumo* **is not** Baja **&** *costo* **is** Alto **&** *batGenCon* **is not** Baja ***then*** *salida* = Alta
8. ***if*** *consumo* **is** Baja **&** *costo* **is** Alto **&** *batGenCon* **is not** Baja ***then*** *salida* = Media
9. ***if*** *consumo* **is** Baja **&** *costo* **is** Bajo ***then*** *salida* = Baja
10. ***if*** *batGenCon* **is** Baja ***then*** *salida* = OFF
11. ***if*** *batGenCon* **is** Alta ***then*** *salida* = ON

El diseño de las reglas fue hecho en base al comportamiento esperado del sistema para situaciones específicas. Las seis primeras reglas contemplan a la batería en el estado presente y luego de un intervalo de tiempo (si no se consume de ella), buscando que la preferencia del uso de la batería se dé acorde al costo. Estas reglas

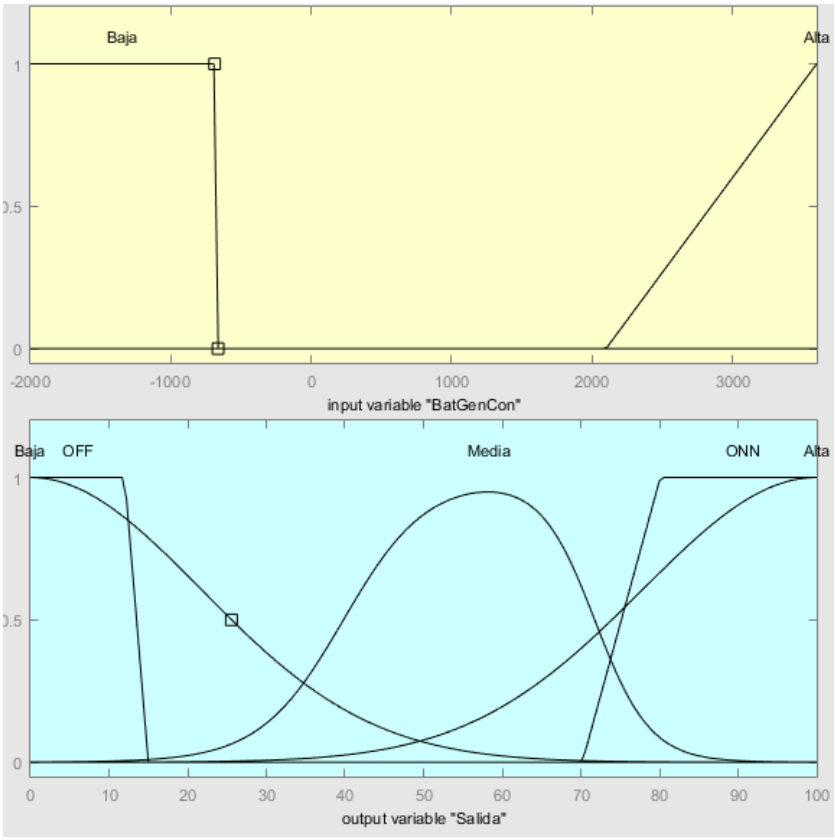


Figura 3.6: Funciones de membresía de la batGenCos y la salida.

no cubren todo el espectro de posibilidades, como por ejemplo batería alta y costo bajo, porque no contemplan una salida única. A su vez, la falta de decisión en estos casos le da importancia a las siguientes reglas.

Las reglas 7 y 8, contemplan al consumo dado el costo alto, éstas buscan consumir de batería con mayor o menor grado dependiendo de el nivel de consumo. Éstas reglas están penalizadas por la entrada *batGenCos*, dado que si el consumo es mayor que lo que podría dar la energía almacenada en batería más la energía generada en el intervalo, entonces las reglas pesan menos en el resultado.

La novena regla también contempla al consumo; busca penalizar el uso de la batería si tiene poca energía almacenada y el costo es bajo, evitando el gasto de energía almacenada en pequeños consumos cuando se está en horarios de bajo costo.

Las dos últimas reglas fueron diseñadas para mantener la energía de la batería dentro un de un rango de uso, con el objetivo de no perder energía generada por falta de capacidad y la de no llevar a consumir por debajo del mínimo de energía disponible en la batería.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 4

Software

4.1. Introducción

En este capítulo se detallan las rutinas de Software desarrolladas tanto para la comunicación entre los nodos, así como las específicas de cada sensor y los modelos del sistema fotovoltaico.

4.2. Comunicación entre nodos

En la presente sección se presenta el protocolo de comunicación MQTT explicando su funcionamiento y cómo es implementado en los nodos, tanto los periféricos (ESP8266), como el central (Beagle Bone Black). Pero antes se presentará el primer protocolo que se desarrolló antes de ser remplazado por el MQTT[1].

4.2.1. Primera implementación, los sockets

Inicialmente la comunicación entre los diferentes nodos fue desarrollada en base a la implementación de sockets, definiéndose socket como la forma de intercambiar datos entre dos programas corriendo en dos dispositivos diferentes, a través del protocolo TCP/IP.

Esta implementación requería que cada nodo levantara un servidor, obligando a que el resto de ellos tuviera que tener definida la dirección IP de cada uno. Para simplificar, todas las comunicaciones se hacían hacia el nodo central, el cual se encargaba de gestionar el destino final de los mensajes. En los Anexos se encuentra la librería implementada para los nodos periféricos A.12.2 y el módulo A.12.1 implementado en Python para el nodo central.

[1]Message Queue Telemetry Transport

4.2.2. Implementación definitiva, protocolo MQTT

Finalmente, el protocolo anteriormente mencionado fue reemplazado por el protocolo MQTT (Message Queue Telemetry Transport), el cual fue ideado originalmente por IBM y está enfocado en la conectividad Machine to Machine (M2M). Su uso es extendido en aplicaciones que requieren un ancho de banda pequeño, además posee la ventaja de ser un método de comunicación de muy bajo consumo que requiere pocos recursos para su funcionamiento. Su arquitectura consiste en un nodo central o broker que se encarga de gestionar los diferentes mensajes que transitan por él.

MQTT implementa el concepto de tema (o topic en inglés) con lo cual se organiza la comunicación. Básicamente, hay nodos que publican en diferentes topics, mientras que otros se subscriben a éstos.

También es posible dotar de jerarquía a los topic, de manera que cuando un cliente se suscribe a uno, recibe no solo información de éste, sino también de todos los subtopics asociados. Para poder implementar el protocolo, se implementó Mosquitto[2] como broker, corriendo en la BeagleBone. Este servidor cumple la función de gestionar las colas de mensajes y los publicadores/suscriptores participantes.

Se definió este cambio por la sencillez del uso del MQTT, que permite enviar todo tipo de datos en formato de texto, sin tener que configurar individualmente cada servidor y cliente, como hay que hacer si se utilizara sockets.

4.2.3. Protocolo MQTT en nodos periféricos

El cambio a MQTT implicó la migración de todo el código de envío de información, para lo cual se utilizó la librería *PubSubClient.h*[3] en los nodos periféricos. De dicha librería se utilizaron en particular las siguientes funciones en sus versiones mas simples:

- ***setServer(IP_broker, port)***: se utiliza para configurar la dirección del broker y el puerto de comunicación del mismo, por defecto el puerto es el 1883.
- ***setCallback(function_Callback)***: define la función a ejecutar al recibir un mensaje en algún topic al que se está suscrito.
- ***connect(ID, user, password)***: permite conectarse con el broker con una ID individual para cada nodo. Adicionalmente, permite autenticarse si el

[2]Ver `https://mosquitto.org/`.

[3]Se puede encontrar en el repositorio https://github.com/knolleary/pubsubclient con licencia MIT

broker fue configurado para requerir usuario y contraseña; no fue implementada esta funcionalidad, pero es relevante si se quiere hacer al sistema más seguro.

- ***publish(topic, message)***: permite publicar mensajes, requiere que previamente el nodo este conectado al broker. Básicamente se establece en qué topic se quiere publicar el mensaje y cuál es el mismo.

- ***subscribe(topic)***: requiere que esté conectado al broker, permite suscribirse a un topic, para que luego, al recibir un mensaje nuevo, el programa ejecute la *function_Callback*. En caso de querer suscribirse a todos, basta con poner un # en el campo topic.

4.2.4. Protocolo MQTT en nodo central

El nodo central por el cual se optó es una Beagle Bone Black. Para poder trabajar con los nodos periféricos se puso a todos estos bajo la misma red. Para el caso de la Beagle Bone se trataron dos posibles métodos.

El primero fue mediante un dongle WiFi, para el cual se tuvo que instalar el driver correspondiente, mediante el dongle se pudo configurar tanto un punto de acceso WiFi como conectarse a una red existente.

El segundo fue a través de un cable Ethernet conectado entre la Beagle Bone y el router que oficiaba de punto de acceso, eliminando el dongle WiFi. Se terminó eligiendo esta opción para implementar el sistema.

Para trabajar con el protocolo MQTT se instaló un broker local en la Beagle Bone llamado Mosquitto[4], un broker open Source que implementa las versiones 3.1 y 3.1.1 del protocolo.

La librería utilizada por python para trabajar con el protocolo MQTT es *paho-mqtt*[5], la cual implementa una clase llamada *client()*, los métodos que implementa están descritos en la API de la librería, pero a continuación se van a describir los que se utilizaron.

- ***connect(host, port=1883)***: esta función recibe como parámetros el host (dirección IP) del broker y el puerto por el cual transitan los mensajes.

- ***on_conect(client, userdata)***: esta función es llamada cuando el broker responde a una petición de conexión desde el cliente, por lo que permite a este último, por ejemplo, suscribirse a los tópicos que solicite.

[4]Ver `https://mosquitto.org/`, en la sección documentation de dicha página, se encuentran varias herramientas para debuggear el funcionamiento de mosquitto: en particular mosquitto_sub y mosquitto_pub de utilidad para aprender de su funcionamiento.

[5] ver pypi.python.org/pypi/paho-mqtt/1.1

- ***on_message(client, userdata, message)***: esta función es llamada cuando un mensaje llega al tópico en que el cliente está suscrito, permitiendo por ejemplo, guardar ese dato para su posterior uso.

- ***loop_forever()***: esta función implementa un loop que se queda a la escucha durante tiempo indefinido, salvo que se rompa la conexión.

Además de implementar la clase *client*, paho-mqtt implementa un módulo para publicar en los tópicos, el cual es importado a través de *paho.mqtt.publish*, el método que se utilizó de este módulo se llama ***single()*** el cual publica un único mensaje en un tópico definido.

4.2.5. Plataforma Thingspeak

Como se menciona en la sección 2.2.4, en las etapas iniciales de este proyecto fue estudiado el uso de esta plataforma para resolver la comunicación entre los nodos periféricos y el nodo central, y si bien esto fue descartado por los motivos que se exhiben en la sección, se contempla la posibilidad de usar la plataforma como respaldo de datos. Es decir, una vez que los datos obtenidos en los periféricos son recibidos en la beaglebone, o bien, los resultados son obtenidos de los módulos de pyhton correspondientes a la optimización del uso de energía eléctrica, pueden ser enviados a la plataforma a modo de respaldo. Esto además permite tener un monitoreo remoto del estado del sistema.

La plataforma thingspeak es una solución abierta para los productos y servicios de internet de las cosas y ofrece la posibilidad de funcionar a través de servicios de host gratuitos o servidores personales.

4.3. Módulos ESP8266

4.3.1. Introducción

En esta sección se introduce y detalla cada módulo periférico, y en particular las rutinas y librerías utilizadas para ellos.

4.3.2. Nodo Exterior

El nodo exterior tiene el objetivo de relevar las variables necesarias para la simulación del panel fotovoltaico: Irradiancia y Temperatura. Para ello, se dispuso de una placa NodeMCU (ver sección 2.2.3), un sensor de temperatura DHT11, un fotodiodo de silicio y un circuito adicional para acondicionar la señal analógica del fotodiodo. El diagrama de conexión del módulo se puede ver en la Figura 4.1.

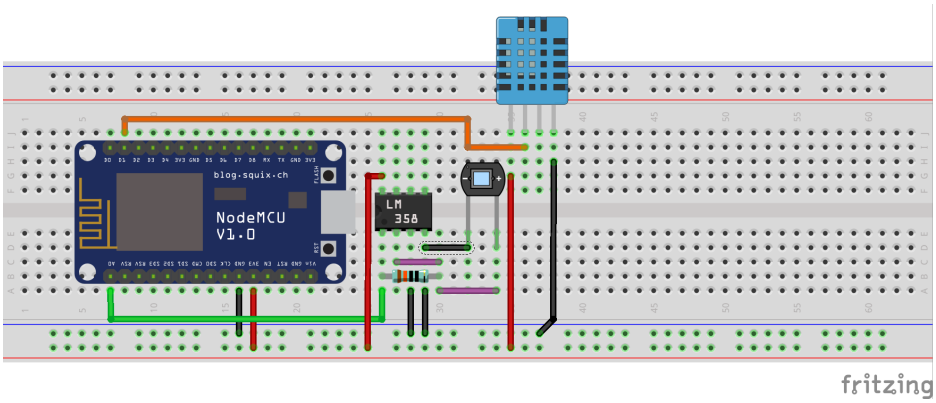


Figura 4.1: Diagrama de conexionado del Nodo Exterior.

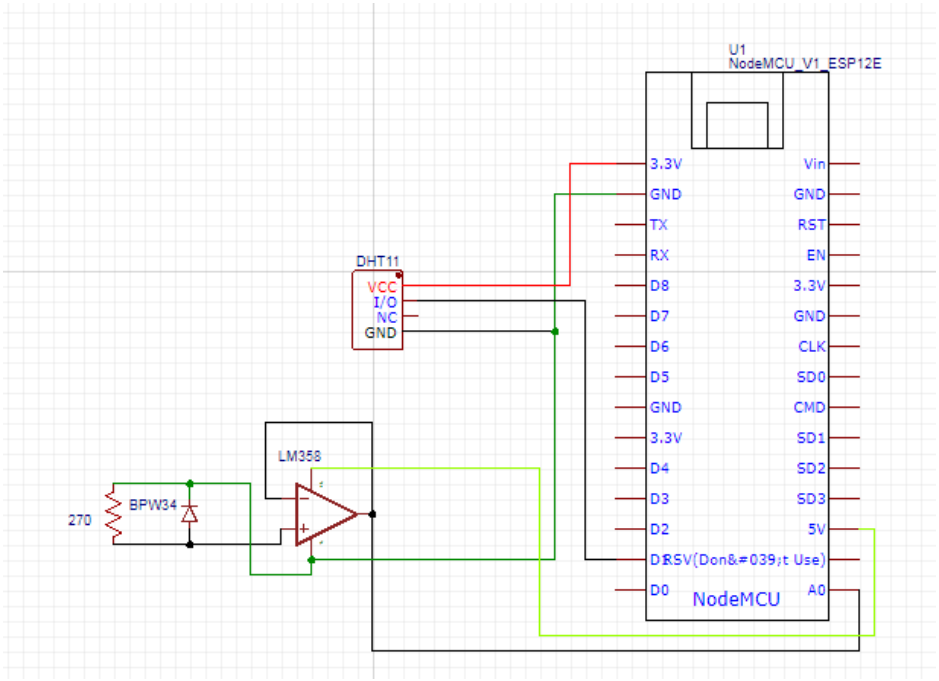


Figura 4.2: Esquematico del Nodo Exterior.

Medición de Temperatura Exterior

Para la medición de temperatura se utiliza el sensor DHT11, descrito en el capítulo 2.2.1. Dicho sensor se alimenta desde la placa con 3V, y su salida digital se conecta al pin D1. Para la comunicación entre la placa y el sensor se utiliza la librería *rDHT11.h*, disponible en la página de Arduino[6]. De dicha librería se utilizan las siguientes funciones:

- ***DHT11(pin)***: inicialmente se define el pin en el que está conectado el sensor.

[6]Más precisamente en `http://arduino.cc/playground/Main/DHT11Lib`

- ***update()***: esta función vacía el buffer de comunicación con el sensor y solicita y decodifica una nueva medición, convirtiendo la señal serial del sensor en datos numéricos de temperatura y humedad.
- ***getCelsius()***: esta función retorna el valor de temperatura almacenado en el buffer en grados Celsius.

Medición de Irradiancia

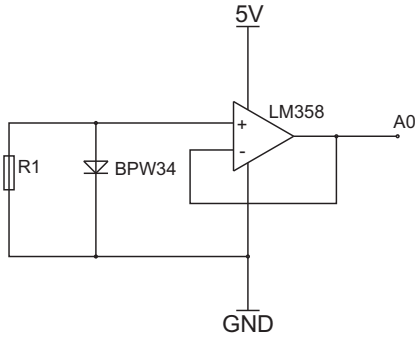


Figura 4.3: Diagrama de conexión de Amplificador Operacional como seguidor funcionando como adaptador de impedancias

Por otro lado, para medir la irradiancia, se utiliza el diodo BPW34, cuyas características se describieron en el capítulo 2.2.1. Este diodo genera una corriente inversa entre sus pines proporcional a la irradiancia que incide en su superficie. Para poder convertir dicha corriente en una señal medible por la ESP se utiliza una configuración de hardware adicional (ver figura 4.3) que acondiciona la señal para poder ingresar un voltaje en el rango admisible del pin analógico de la placa.

En particular, para el acondicionamiento de la señal, primeramente se utiliza una resistencia en serie con el diodo para lograr una tensión proporcional a la corriente generada. Como el máximo voltaje a la entrada del pin digital de la ESP es 1V y la corriente máxima generada por el diodo por la incidencia del sol es de 0,0034A (Según hoja de datos; $E_{sol-max}$=1000W/m^2), el valor de la resistencia sería de 294Ω. Para no perder los datos de máxima potencia, se utilizó una resistencia de 270Ω±1 %.

Para mitigar el efecto de las impedancias vistas a la entrada de la placa y a la salida del conjunto diodo-resistencia, se utilizó un Amplificador Operacional LM358 en configuración de seguidor de tensión. Esta configuración con seguidor permite eliminar los efectos de la impedancia de salida del conjunto diodo-resistencia y la impedancia de entrada a la ESP.

Es relevante destacar que se optó por utilizar el Diodo BPW34 en base al estudio realizado por M.Čekon, R.Slávik y P.Juráš; Obtainable Method of Measuring

the Solar Radiant Flux Based on Silicone Photodiode Element [8].

En dicho estudio, se comparan las medidas de un piranómetro vertical y otro horizontal de una estación climática en Brno (República Checa) con las lecturas realizadas por un sistema de cuatro diodos BPW34. Finalmente, en la publicación se concluye que es una buena forma de medir la irradiancia, especialmente en la medida horizontal, que es relevante en este proyecto.

4.3.3. Nodo Interior

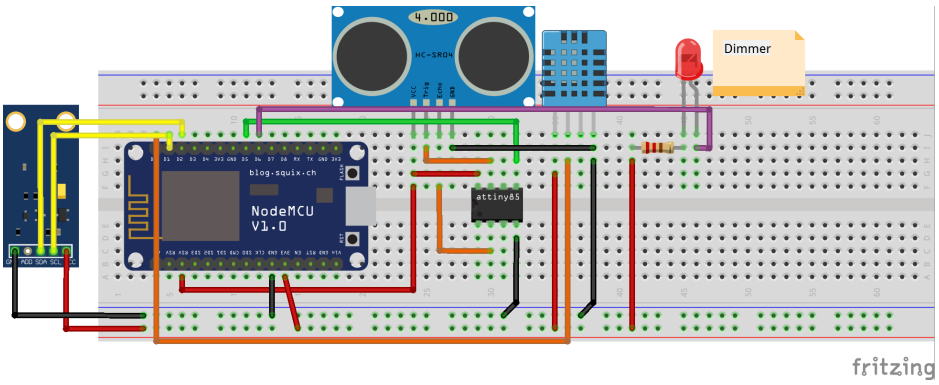


Figura 4.4: Diagrama de conexionado del Nodo Interior.

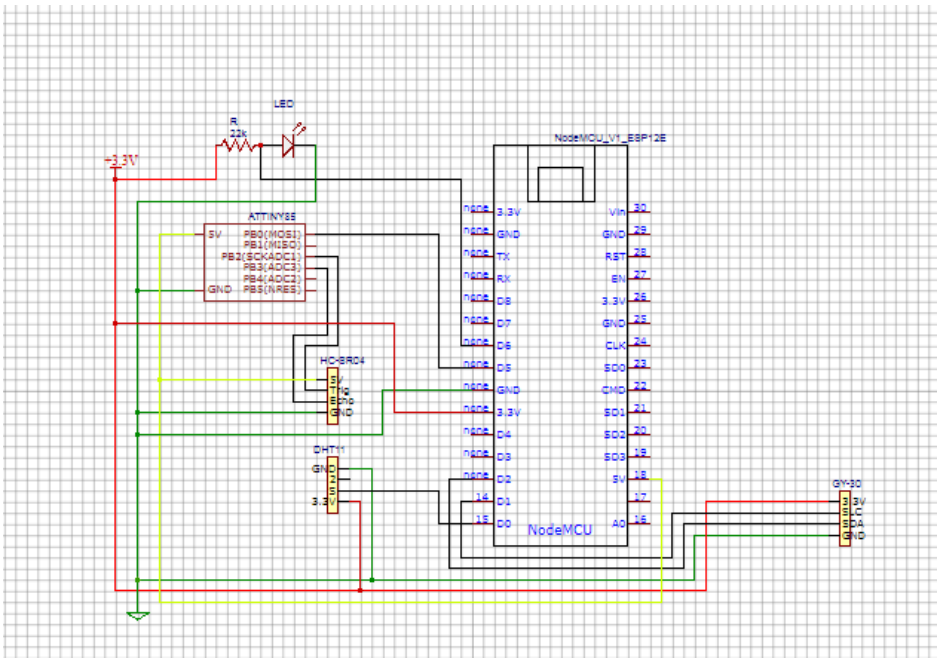


Figura 4.5: Esquematico del Nodo Exterior.

En el interior de la habitación se encuentra el nodo que registra la luminosidad del ambiente y la temperatura interior, y controla, mediante el dimmer, una lámpara sobre un escritorio. Adicionalmente, controla la presencia de personas en el escritorio con un sensor de distancia controlado por un Attiny 85. El diagrama de conexión del módulo se puede ver en la Figura 4.4

Temperatura Interior

Para la medición de la temperatura interior se utiliza un sensor DHT11 similar al utilizado para la medición de la temperatura exterior. Para controlarlo se utiliza la librería antes detallada (ver 4.3.2).

Detección de Presencia

Para definir si hay alguien en el escritorio se utiliza el sensor de distancia por ultrasonido HC-SR04, el mismo se controla desde un ATtiny85, programado a través de la IDE de Arduino, con la librería *Ultrasonic.h*[7].

De dicha librería se utilizan las siguientes funciones:

- ***ultrasonic(trig, echo)***: con este comando se crea y se inicializa el objeto *Ultrasonic*, y se definen los pines de *Trigger* y *Echo* donde está conectado el sensor.
- ***ultrasonic.Ranging(cm)***: esta función devuelve la distancia obtenida por el sensor en centímetros.

Se resolvió utilizar un microprocesador adicional para el control del sensor de distancia, debido a que al implementar todos los sensores y actuadores en una misma ESP, había conflictos entre la medida de distancia y la salida PWM para el dimmer, ya que ambas utilizaban el mismo timer de la placa. Se definió que fuera un Attiny85 por la cantidad de pines necesarios y el tamaño del integrado.

La interacción entre el ATtiny y la ESP se realiza por medio de señales digitales; el ATtiny envía un flanco de subida si la distancia medida es menor a 20 cm mediante la función ***digitalWrite(pin, stat)***, donde pin refiere al pin de salida y stat al estatus del mismo; *HIGH* o *LOW*.

Control de luminosidad

Para la medición exacta de la luminosidad en el plano de trabajo del escritorio se utilizó el sensor BH1750FVI, para lo cual se crea la librería *Lumen.h* (ver Anexo A.3.1) basada en la librería *Wire.h*[8] para la comunicación sobre el protocolo I2C. En la librería creada se utilizan las siguientes funciones:

[7]Creada por ITead, disponible en `https://github.com/hemalchevli/ultrasonic-library/blob/master/Ultrasonic.h`

[8]incluida por defecto en el IDE de Arduino

- ***Wire.beginTransmission(address)***: esta función inicia la transmisión con la dirección especificada. En este caso la dirección del sensor por defecto es 0x23 en hexadecimal.
- ***Wire.requestFrom(address, request)***: esta función solicita al sensor la información asociada al requerimiento *request* a la dirección *address*.
- ***Wire.read()***: esta función recibe la información a través del protocolo I2C y la devuelve a razón de un carácter.
- ***Wire.endTransmission()***: esta función finaliza la comunicación con el sensor, dejando el bus libre para comunicarse con otro sensor conectado en el mismo.

Control de Dimmer

El funcionamiento del Dimmer es en base a una señal PWM sobre un LED cuya luz incide en la fotorresistencia del circuito de potencia. Este funcionamiento permite además la separación eléctrica entre los circuitos, funcionando el led y la fotorresistencia como un optoacoplador.

Para definir la señal PWM el programa principal evalúa primero el pin de entrada desde el ATtiny; si está en alto mide la luminosidad del plano de trabajo. Si dicha luminosidad es menor a 550lx, la rutina comienza a emitir y aumentar la señal PWM gradualmente, evaluando paso a paso la luminosidad del plano, hasta alcanzar el valor objetivo. Los intervalos de evaluación y aumento de señal PWM están definidos cada 500ms.

La salida PWM se resuelve utilizando la función ***AnalogWrite(pin, value)***, donde *pin* es la salida hacia el dimmer y *value* varía entre 1024 y 898, ya que ese intervalo es el que genera variaciones en la luminosidad del LED relevantes para el circuito del dimmer. El programa completo se encuentra en el Anexo A.3

4.3.4. Nodo Consumo e IR

En el interior de la habitación también se encuentra el nodo que registra el consumo y que a su vez tiene el emisor IR para controlar el equipo de Aire Acondicionado. Se instaló también un display LCD a través de un conversor I2C para poder ver el consumo en el momento. El diagrama de conexión del módulo se puede ver en la Figura 4.6.

Consumo de energía

Para la medición de consumo se utilizó la pinza y el hardware descrito en la sección 2.2.1, para la interpretación de la entrada analógica, se utilizó la librería

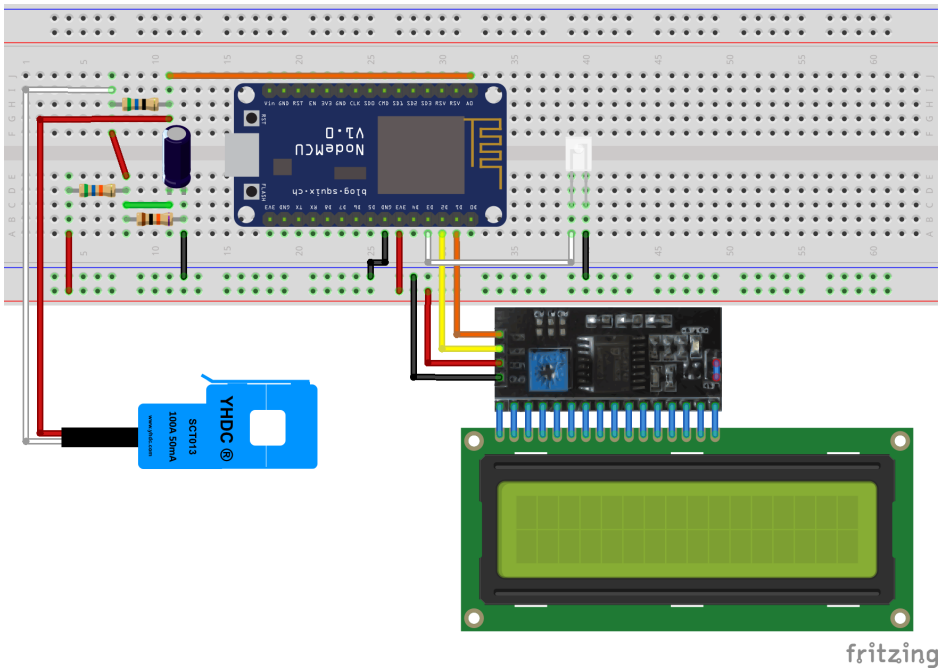


Figura 4.6: Diagrama de conexionado del Nodo de medición de consumo.

EmonLib.h[9].

De dicha librería se utilizan las siguientes funciones:

- ***current(input pin, calibration)***: esta función inicializa la medida de corriente, en la misma se define el pin en que se conecta la pinza amperimétrica y la calibración de la medida. Para definir el valor de la calibración, se comparó la medida de la pinza amperimétrica con la medida de una pinza Fluke 376 de un mismo conductor a diferentes corrientes. Se obtuvo una divergencia máxima entre las muestras en los extremos del rango de medida[10] del 3 %.
- ***calcIrms(cantidad_de_muestras)***: esta función es la responsable de tomar la medida de la corriente RMS, para ello se define una cantidad de muestras determinada. A mayor cantidad de muestras, más exacta es la medida. En este proyecto se toman 1480 medidas de corriente para calcular la Irms.

Para la medida de la corriente, la librería funciona de la siguiente manera: para cada una de las muestras tomadas a través de la función *analogRead(input_pin)* se rectifica la medida restándole la tensión de Offset (tensión en continua utilizada

[9]Creada por Trystan Lea, adaptando la librería original (disponible en https://openenergymonitor.org) para funcionar con un ADC de 10bits.

[10]Se consideró un rango de 0 a 25 A, coincidiendo con el máximo de corriente del interruptor general.

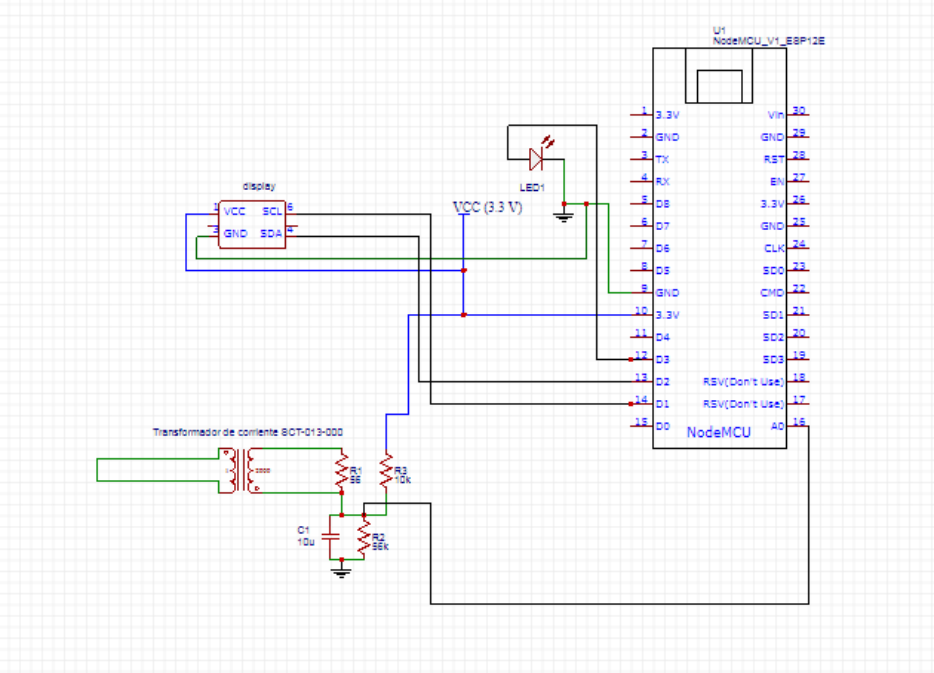


Figura 4.7: Esquematico del Nodo de medición de consumo.

para que la entrada analógica sea positiva, ver sección 2.2.1). Luego, se eleva la medida al cuadrado y se suman todas las medidas. Se procede entonces a calcular la raíz de la sumatoria y se la divide entre la cantidad de muestras. Finalmente se multiplica el valor obtenido por los parámetros de correspondencia; el valor de calibración, el rango de la entrada digital y se divide entre la resolución del ADC. Se obtiene entonces la corriente consumida.

Para calcular la potencia se multiplica la corriente por 230V. A los efectos del proyecto, se asume que la variación en la tensión de la red es despreciable.

Control IR

Para el control remoto del equipo de Aire Acondicionado, se utilizó una librería basada en la librería *IRremote.h* creada originalmente por Ken Shirriff y editada varias veces para su aplicación en la placa ESP8266[11].

Las modificaciones realizadas fueron consecuencia del proceso de ingeniería inversa sobre el control del equipo de Aire Acondicionado, e incluyen cambios en el largo del mensaje y en los tiempos específicos de las máscaras de ceros y unos del mensaje. Se trabajó sobre el equipo de Aire del laboratorio de Medidas.[12]

[11] En particular, la versión utilizada como base es la que se encuentra en el repositorio `https://github.com/markszabo/IRremoteESP8266`, con licencia GNU

[12] Se trata de un equipo Split marca Mabe de 9000BTU

El objetivo de un control remoto es enviar un vector de información, en formato binario, a través de una onda portadora, en este caso de 38kHz. Para eso se definen distintos intervalos de tiempo (que se miden en cantidad de ciclos de la onda de 38kHz) que corresponden a ceros y unos a las señales de inicio o fin del mensaje. Cada uno de estos intervalos están separados por un intervalo de tiempo constante. A partir del análisis del control remoto utilizado, se pudo identificar los intervalos de tiempo de cada elemento del mensaje, en particular se obtuvieron los siguientes valores:

Elemento	Tiempo
Señal de Inicio del Mensaje	3300μs
Señal de Espacio al inicio del Mensaje	1550μs
Uno	950μs
Cero	300μs
Señal de Espacio entre Bits	550μs

Tabla 4.1: Referencia de Tiempo correspondiente a cada elemento de la señal IR

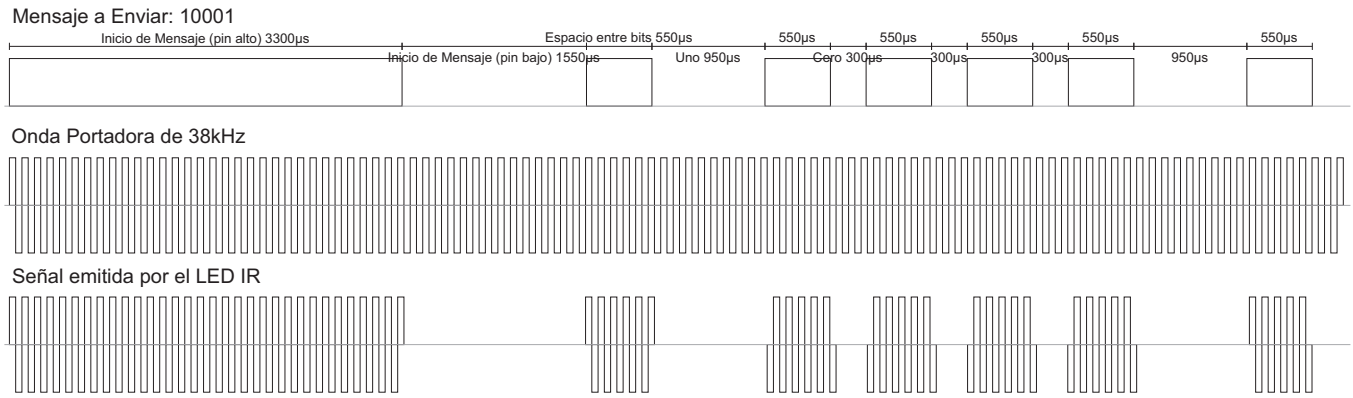


Figura 4.8: Ejemplo de generación de señal IR. Mensaje enviado: 10001.

Para construir el mensaje en el caso estudiado, se debe enviar un mensaje compuesto por una señal de inicio del mensaje, seguida de una de espacio de inicio del mensaje, y a continuación, el vector binario con una señal de espacio entre cada bit (Ver Figura 4.8 con ejemplo). La señal de salida emitida por el LED IR, es en definitiva, una señal cuadrada, con los flancos de subida y bajada descritos anteriormente, modulada como una señal cuadrada de 38kHz.

En los hechos, la generación de la señal portadora, así como la máscara para el mensaje binario se definen en la librería *IRremote.h*. La señal NEC incluida en la misma tiene un formato similar a la señal del Aire Acondicionado, por lo que se utiliza dicho formato como punto de partida. Se modificaron los intervalos de tiempo para coincidir con los obtenidos del control remoto y se amplió el tamaño del mensaje, ya que el protocolo NEC utiliza palabras de 16 bits y el Aire Acondicionado de 112 bits.

Realizando los cambios mencionados, la librería es compatible con el equipo de Aire Acondicionado. En el nodo se utilizan las siguientes funciones de *IRremote.h*:

- ***irsend(IR_pin)***: en esta función se define el pin donde se conecta el LED IR.
- ***irsend.begin()***: esta función inicializa el objeto *irsend*.
- ***irsend.sendNEC(vector_binario,bits)***: esta función es la que convierte el mensaje binario en el código explicado anteriormente sobre la onda portadora de 38kHz.

Comando	Mensaje (112 bits)
Apagado	0xC4D364800004C050A000000000E2
21°C Frío	0xC4D364800024C050A000000000D2
24°C Frío	0xC4D364800024C0E0A00000000012
24°C Caliente	0xC4D36480002480E0A00000000062
28°C Caliente	0xC4D3648000248020A000000000C2

Tabla 4.2: Mensajes desgrabados del control remoto del Aire Acondicionado

Los mensajes relevantes que se desgrabaron del control del Aire Acondicionado se detallan en la Tabla 4.2

Display LCD

Para poder realizar la lectura de la potencia consumida en tiempo real, como un servicio al usuario, se utilizó un display LCD de 2x16 con un conversor I2C.

El protocolo de comunicación I2C, explicado con anterioridad (ver 2.2.1), permite la comunicación serial entre circuitos integrados. En este caso, se utiliza un conversor I2C que simplifica la comunicación entre la ESP8266 y el Display, asociado a la librería *LiquidCrystal_I2C.h*[13]. En particular, de dicha librería se utilizaron las siguientes funciones:

- ***lcd.begin (16,2)***; esta función inicializa el display, en este de caso de 16x2 caracteres.
- ***lcd.home ('Inicializando...')***: esta función permite mostrar en pantalla un mensaje al inicio al comenzar una conexión con el display.
- ***lcd.print('NodoConsumo')***: esta función imprime los caracteres en el display a partir de la posición en que se encuentre el cursor. En este caso imprime "NodoConsumo"
- ***lcd.setCursor (0, 0)***: esta función modifica la ubicación del cursor, en el caso de las coordenadas (0,0) se sitúa en la posición arriba a la izquierda.

[13]Creada por Francisco Malpartida bajo licenica Creative Commons 3.0. Disponible en `https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/Home`

4.4. Módulos en la BeagleBone

4.4.1. Introducción

En esta sección se introducen y detallan los módulos creados en el nodo central, así como las librerías necesarias para el funcionamiento de los mismos.

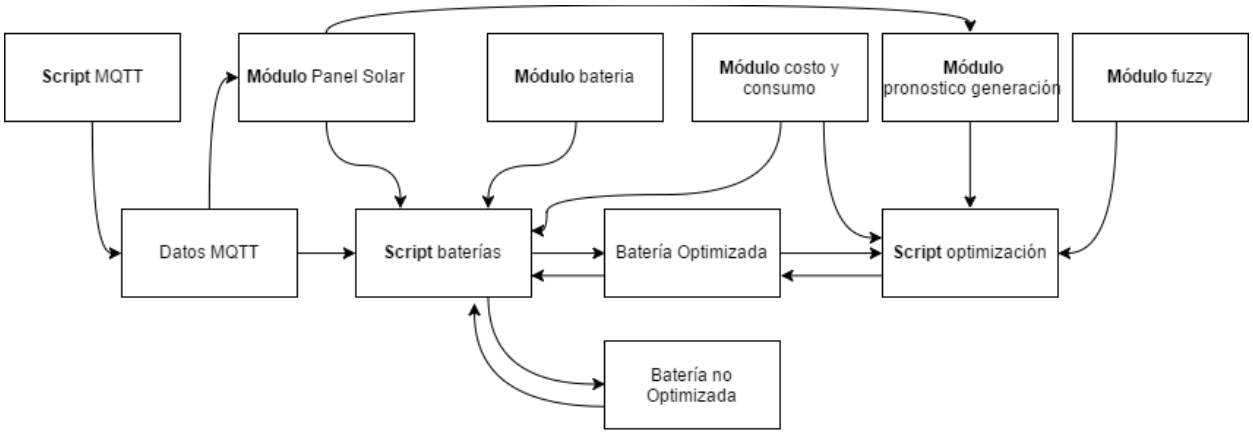


Figura 4.9: Diagrama de archivos que forman parte del nodo central.

La figura 4.9 muestra los diferentes archivos que integran a la implementación del algoritmo de optimización. Cuentan con cinco módulos que brindan funcionalidades a los tres scripts que corren sobre el nodo central, y cuenta con tres archivos donde se perduran en formato *Json* las variables utilizadas por estos scripts. Los archivos van a ser explicados en las siguiente subsecciones. El lenguaje utilizado para todos estos módulos y script es python.

4.4.2. Script de optimización

Este script implementa el algoritmo presentado en la sección de optimización 3.7.2.En el anexo A.4 se detalla el código explicado a continuación. Primero se definen dos clases que se utilizarán en el script, estas son *nodo* y *podaArbol*.

La primera define a un nodo englobando las variables de: estado de la batería (*bateria*), intervalo de tiempo asociado (*tiempo*), probabilidad ponderada (*probabilidad*), costo acumulado (*costoAcumulado*) y un estado que muestra si es un nodo activo o no (*estado*).

La clase podaArbol define la variable *maximaProbabilidad*, y los métodos:

- ***maxProb(prob)***: se encarga de guardar la mayor probabilidad ponderada de aquel nodo que en un intervalo de tiempo la posea.
- ***setmaxProb()***: vuelve a cero el valor de la variable *maximaProbabilidad*.

- ***poda(nodo)***: aplica el algoritmo de poda al nodo que entra, llevando su *estado* a False si la *probabilidad* es menor al 20 % de la *maximaProbabilidad*.

La función ***heducAlgoritmo()*** es la encargada de implementar el algoritmo. Primero se define un objeto de la clase *podaArbol*, luego se trae el estado de la batería del módulo *Batería optimizada*, y los vectores de pronóstico de generación, consumo y costo del kWh en los próximos ocho intervalos de tiempo (se terminó eligiendo este número para las pruebas que se hicieron) de los módulos *Costo y consumo* y *pronóstico de generación*.

El árbol fue creado implementando un diccionario de Python, donde la clave es un String con el nombre del nodo y el valor es un objeto *nodo*. La iteración en los diferentes intervalos de tiempo consiste en aplicar en los nodos activos de ese intervalo el controlador fuzzy, luego se crean los nodos hijos de cada uno, donde además el objeto *poda* guarda el valor *maximaProbabilidad* de todos ellos. Finalizada la creación de los hijos se procede a podar, descartando los resultados que estén por debajo del máximo mencionado anteriormente.

Finalizada la iteración, se procede a buscar aquel nodo del último intervalo de tiempo que tenga el menor costo posible. Y en base a esta elección, se decide si consumir o no energía desde las baterías en el próximo intervalo de tiempo.

El script consiste en aplicar esta función una vez por intervalo de tiempo, el cual se definió de 15 minutos (o una hora).

4.4.3. Modelo de Panel Fotovoltaico

Electrical Characteristics

STC	LNSE-245P
Optimum Operating Voltage (Vmp)	30.0 V
Optimum Operating Current (Imp)	8.17A
Open Circuit Voltage (Voc)	37.1V
Short Circuit Current (Isc)	8.74A
Maximum Power at STC (Pmax)	245W
Cell Efficiency	17.11%
Operating Module Temperature	-40 °C to +85 °C

Figura 4.10: Principales características del panel modelado; Luxen LNSE-245P

El fin de este módulo es poder modelar la potencia que generaría un panel fotovoltaico de la manera más sencilla posible.

Se resolvió utilizar como modelo un panel marca Luxen, modelo LNSE-245P, en el CD adjunto a la memoria se encuentra la hoja de datos del mismo[14]. Por

[14]En particular en CD-HEduc\Datasheets\Solar panel Luxen 245P.pdf

otro lado, los valores más notables utilizados en el modelo se reflejan en la figura 4.10.

I_1, V_1	Los puntos medidos experimentalmente.
I_{MR}	Es la corriente de cortocircuito medida del dispositivo de referencia.
I_{SR}	Es la corriente de cortocircuito del dispositivo de referencia en el estándar u otra irradiación deseada.
K	Factor de corrección de la curva.
I_2, V_2	Los puntos aproximados de la corrección.
P_{max}	Potencia máxima de salida
I_{SC}	Corriente de cortocircuito a 25° C y 1000W/m^2
I_{ph}, I_{SR}	corriente fotoeléctrica, corriente de saturación inversa.
I_{max}	corriente máxima ideal, cuando $V = -$ ínf en STC.
I_{MPPT}, V_{MPPT}	corriente y tensión óptimos para extraer la máxima potencia del panel.
V_{OC}	tensión de circuito abierto (.ºpen-cicuit").
V_{max}	tensión de circuito abierto a 25° C y 1000W/m^2, condición de STC.
V_{min}	tensión de circuito abierto a 25° C y 200W/m^2.
R_S, R_P	Resistencia serie y paralelo
T	temperatura del panel solar.
T_N	temperatura nominal; 25°C.
TCi	coeficiente de temperatura para I_{SC}, (%/°C).
TCV	coeficiente de temperatura para V_{OC}, (V/°C).
E_i	irradiancia solar, ($\frac{W}{m^2}$).
E_{iN}	irradiancia solar nominal, $1000W/m^2$.
b	constante característica de la curva I(V), se halla con los valores $I_{OP} y V_{OP}$.
STC	"Standard Test Conditions", es una definición de las condiciones de ambiente para realizar los ensayos de paneles FV, definida en la IEC 60891, implica realizar los mismos con una irradiancia de $1000W/m^2$ y a una temperatura del panel de 25°C .

Tabla 4.3: Referencias para Ecuaciones

Para definir el modelo matemático a emplear, se estudiaron los modelos utilizados en la actualidad para calcular la potencia generada por un panel solar. En particular, se evaluaron los siguientes métodos (ver referencias de ecuaciones en la tabla 4.3):

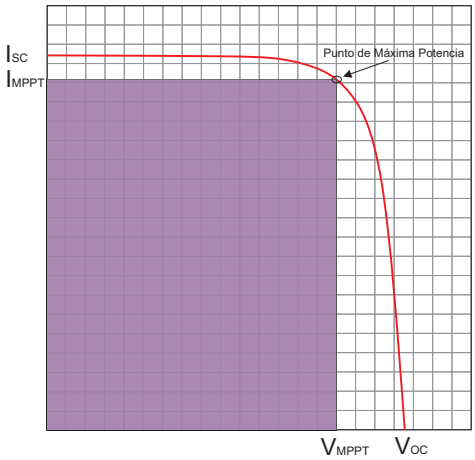
- El estudio de la celda solar como semiconductor en base a la siguiente expresión:

$$I(V) = I_{Ph} - I_{SR}\left(exp(\frac{q.V}{k.T}) - 1\right)$$

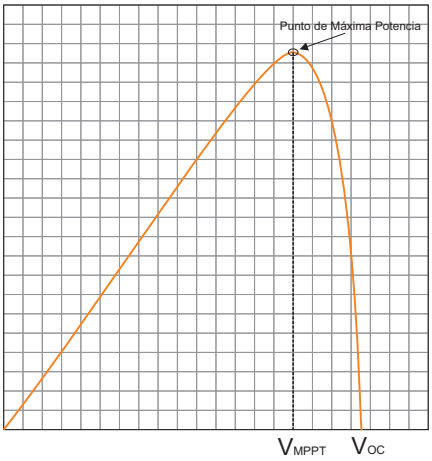
- El estudio según las fórmulas propuestas en la IEC-891 [9]:

$$I_2 = I_1 + I_{SC}\left[\frac{I_{SR}}{I_{MR}} - 1\right] + TCi.(T_2 - T_1)$$

$$V_2 = V_1 - R_S(I_2 - I_1) - KI_2(T_2 - T_1) + TCV(T_2 - T_1)$$



(a) Gráfica de Corriente en función de voltaje, el área señalada en violeta es la potencia máxima que puede generar el panel.



(b) Gráfica de Potencia en función de voltaje en un panel Fotovoltaico

Figura 4.11: Ambas gráficas son del panel LNSE-245P, donde $I_{SC} = 8{,}74A, I_{MPPT} = 8{,}17A, V_{MPPT} = 30{,}0V, V_{OC} = 37{,}1V$

Finalmente, se trabajó en base a una publicación de la IEEE, de Eduardo Ortiz-Rivera del año 2005 [10], donde se plantea un modelo que utiliza como entradas la temperatura, la irradiancia y los valores brindados por el fabricante en la hoja de datos del panel modelado. De dicha publicación se extrajo la ecuación base para el modelo desarrollado:

$$P(V) = \alpha.I_{max}.\tau_i - \alpha.I_{max} - \tau_i.exp\left(\frac{V}{b.(\gamma.\alpha + 1 - \gamma).(V_{max} + \tau_V)} - \frac{1}{b}\right)$$

Donde:

$$\alpha = \frac{E_i}{E_{iN}}$$

$$I_max = \frac{I_{SC}}{1 - exp(-1/b)}$$

$$\gamma = 1 - \frac{V_{min}}{Vmax + \tau_V}$$

$$\tau_i = 1 + \frac{TCi}{100}.(t - T_N)$$

$$\tau_V = TCV - (T - T_N)$$

Se asume que el inversor funciona en todo momento en el máximo punto de potencia[15], por lo que, luego de tener la ecuación de la potencia en función de la tensión y en función de la Irradiancia y la Temperatura, resta hallar la tensión a la que se obtiene la máxima potencia (ver gráficas 4.11a y 4.11b). Para ello, se calcula la derivada de $P(V)$ y se halla V_{MPPT} tal que $\frac{dP}{dt}(V_{MPPT}) = 0$. Esto se resuelve utilizando la función **brentq((f, a, b)** de la librería *scipy* de Python, que permite hallar las raíces de la función f (si existen) en el intervalo definido *[a,b]*. El código utilizado para el modelo Fotovoltaico se encuentra en el anexo A.5

El procedimiento para calcular la potencia generada es el siguiente:

- Inicialmente, se reciben los valores de irradiancia y temperatura exterior desde el Nodo exterior (Ver sección 4.3.2) vía MQTT. Para recibir los mensajes se utiliza la librería Paho.MQTT de Python (ver sección 4.2.4).
- Con los valores antedichos, se define la ecuación P(V) para las condiciones actuales de funcionamiento y se halla la raíz; V_{MPPT}.
- Finalmente, conociendo V_{MPPT} se calcula $P(V_{MPPT})$, lo cual da como resultado la potencia máxima que se podría extraer de un panel LNSE-245P en el estado actual.
- Para calcular la potencia total generada, basta con multiplicar la potencia generada por un panel por el total de paneles que conforman la planta simulada.

4.4.4. Módulo modelo batería

Este módulo implementa funcionalidades para trabajar con un modelo de batería simple, y trabaja con el script de batería para mantener actualizados los estados de las dos baterías implementadas, la que trabaja bajo las reglas de la optimización y la que no.

Este módulo utiliza: el método **now()**, del módulo *datetime*, el cual se encarga de traer (en un formato especificado) la hora actual; el módulo *json* del cual se

[15]MPPT por Maximum Power Point Tracker, la mayoría de inversores para sistemas fotovoltaicos del mercado funcionan de esta manera.

utilizan los métodos **load()** y **dump()** encargados de leer y escribir un objeto json; y el método **parse()** del módulo *dateutil.parser*, que se encarga de convertir un dato del tipo string al tipo datetime[16].

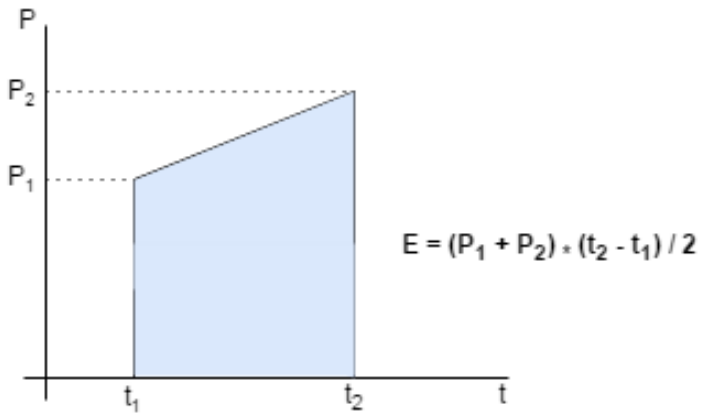


Figura 4.12: Visualización gráfica del cálculo de la energía en un intervalo de tiempo.

El módulo modelo batería creado contiene las siguientes funciones:

- ***decremento_bateria(potConsumida, tiempo)***: esta función recibe como entrada los datos de la potencia instantánea consumida y tiempo asociado a esta medida. Si se está consumiendo desde batería se decrementa la energía almacena restándole la energía integrada entre la última potencia consumida y la actual. En la figura 4.12 se muestra en detalle el cálculo realizado de la energía restada. Además calcula el ahorro que se hace o el gasto dada la energía calculada en el intervalo de tiempo, el costo del Kwh del momento y si se está consumiendo energía desde baterías o no.
- ***incremento_bateria(potGenerada, tiempo)***: de forma análoga a la función anterior, esta función recibe la potencia instantánea generada por el panel solar y el tiempo asociado a esta medida. La función incrementa la energía almacenada, sumándole la energía integrada entre la última potencia generada y la actual, si la energía sobrepasa el límite establecido por la batería se guarda este limite. En la figura 4.12 se muestra en detalle el cálculo realizado.
- ***estado_actual()***: devuelve la energía con la que actualmente cuenta la batería.
- ***bandera(estado)***: modifica la bandera asociada a la habilitación del consumo de la batería, si el estado es positivo se consume de ella. En el caso que el estado anterior de la bandera fuera positivo y llegara la modificación para que no se consuma más de ella, se calcula la energía consumida hasta ese momento, utilizando el dato guardado de la potencia consumida.

[16]Para guardar en json, los datos del tipo datetime se convierten a string

Los datos que se guardan en formato Json son: la energía de la batería, la potencia consumida con su tiempo asociado, la potencia generada con su tiempo asociado, el estado de la bandera que regula el uso de la batería, el ahorro y el gasto efectuado. El código se puede ver en el anexo A.6

4.4.5. Script MQTT

Este script es el encargado de guardar los datos provenientes de los nodos periféricos, los cuales son: irradiancia, temperatura exterior (necesarios en el modelo del panel solar) y potencia consumida. Utiliza el modulo *json*, en particular los métodos **load()** y **dump()** encargados de leer y escribir un objeto json.

El script implementa los métodos explicados en el apartado del modulo MQTT para python: **on_conect()**, donde se subscribe al tópico *"HEDUC/#"*, y **on_message()**, mediante el cual guarda en formato json las variables anteriormente mencionadas en el archivo *DatosMQTT*.

Luego crea una instancia *client()* pasándole los métodos anteriores, se conecta al servidor donde está ubicado el broker (en este caso es la misma Beagle Bone) con la función **conect()**, y finalmente se queda a la escucha de nuevos mensajes llamando a la función **loop_forever()**. El código se puede ver en el anexo A.7

4.4.6. Script baterías

Este script se encarga de mantener actualizados los estados de los dos bancos de baterías (la óptima y no óptima), ejecutando las funciones explicadas en el módulo de batería con los datos del archivo *DatosMQTT*. El código se puede ver en el anexo A.8

4.4.7. Módulo Pronóstico de generación solar

Introducción

Para el modelado del pronóstico de generación solar, se investigó sobre el estado del arte con los referentes nacionales en el área. En tal sentido, hubo reuniones con el Ing. Rodrigo Alonso del Laboratorio de Energía Solar (L.E.S) y con el Ing. Ruben Chaer, Gerente de la Administración del Mercado Eléctrico (A.D.M.E.).

Hoy en el país, la generación eléctrica en sistemas fotovoltaicos se pronostica en base a imágenes satelitales del pronóstico de nubosidad, filtradas por algoritmos que corren en servidores de gran capacidad de procesamiento. El resultado da un pronóstico fiable, pero al ser un proceso complejo, el acceso a la información tiene un costo elevado. Si bien se nos planteó la posibilidad de acceder a dichos pronósticos en el marco de este proyecto, como el objetivo del mismo es que sea

replicable, decidimos generar localmente el pronóstico.

Para ello, se partió de las ideas que surgieron en las reuniones, acerca de posibles pronósticos que predijeran la generación solar en un período de tiempo del orden de una hora, priorizando el acceso a la información sobre la exactitud.

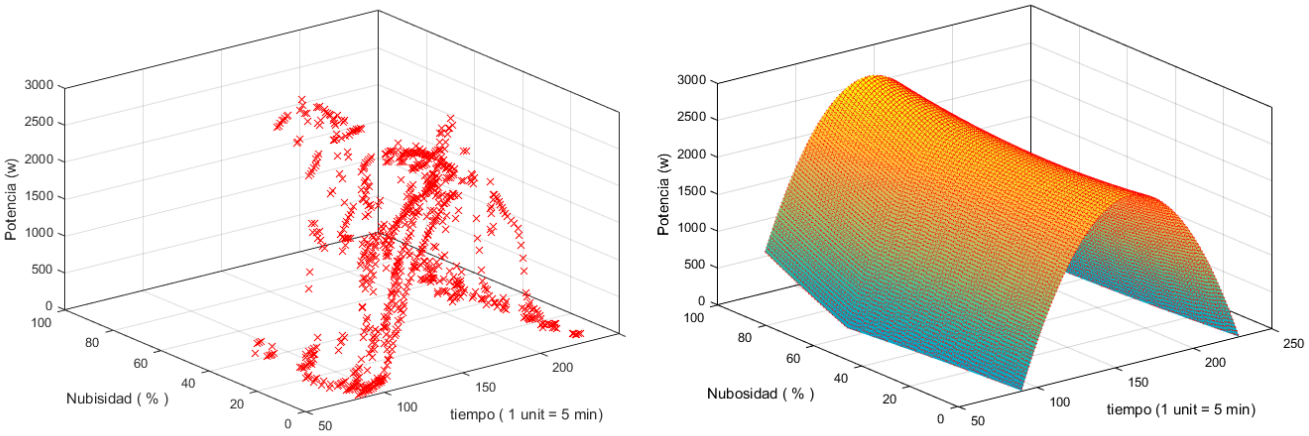


Figura 4.13: Datos recolectados de potencia contra nubosidad y tiempo. Se registraron 812 datos.

Una de las ideas consiste en el cruzamiento de los datos de nubosidad obtenidos de un portal del meteorología (Accuweather) contra la generación de una planta fotovoltaica a las afueras de Montevideo[17]. La implementación del modelo de pronóstico fue hecha con datos recolectados durante 7 días. La figura 4.13 muestra la potencia generada por la planta contra los datos de nubosidad y tiempo, la figura de la derecha es la función a la que se aproximaron estos datos, corresponde a un polinomio de segundo grado de las variables nubosidad y tiempo. La función, que se aproximó por el método de regresión lineal, no aportó una clara evidencia de que la potencia se viera atenuada por la nubosidad.

Las causas de la intrascendencia de este pronóstico fueron discutidas, llegándose a la conclusión de que la nubosidad no era representativa del lugar geográfico de la planta, por lo que las muestras no tienen correlación.

Conclusiones y Pronóstico implementado

De la experiencia anterior se concluye que es posible el diseño de un pronóstico de generación pero estudiando la posibilidad de incorporar otras variables al modelo, como por ejemplo, la temperatura o la generación que se viene dando en

[17]Se utilizaron los datos de una planta existente ya que en ese entonces no estaba aún operativo el Nodo Exterior, ver 4.3.2. En la configuración final, los datos de generación se pueden sacar del propio sistema.

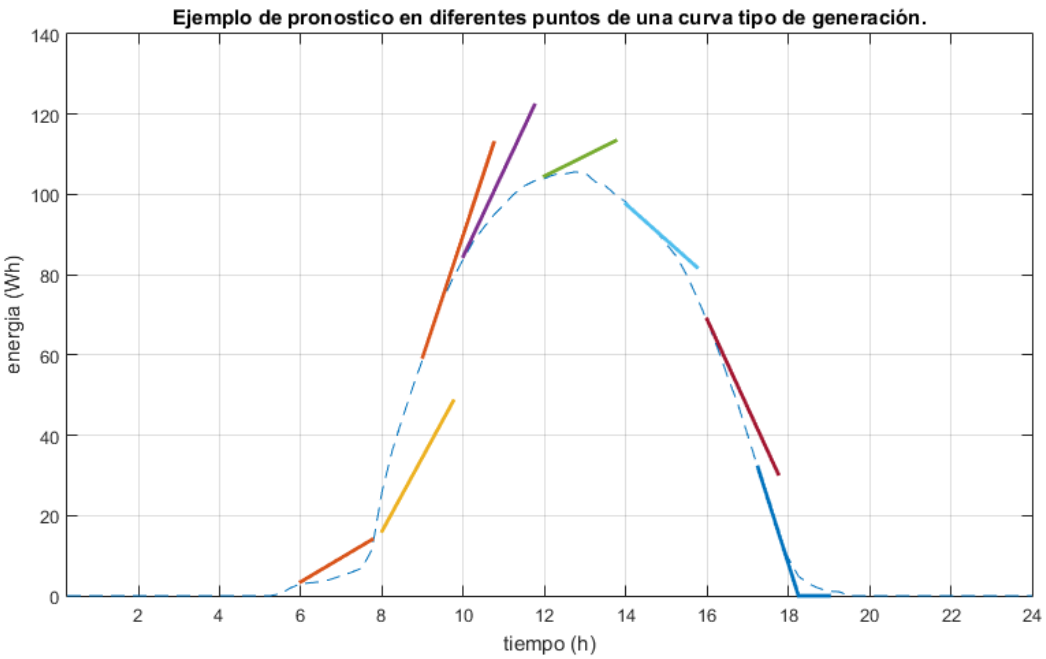


Figura 4.14: Curva de generación con ejecuciones del pronóstico en diferentes intervalos.

el tiempo anterior inmediato. Con ésto se presenta el segundo modelo, el cual fue implementado y utilizado en las simulaciones. A diferencia del pronóstico anterior, éste no fue diseñado con un histórico de datos, sino que tan solo utiliza la energía generada en los dos intervalos de tiempos anteriores al momento de pronosticar.

Entonces, tomando los dos valores de energía producida en los intervalos anteriores, se aproxima a la generación de los siguientes ocho intervalos por la recta que pasa por estos valores. La siguiente fórmula describe lo anterior ($\tilde{g}$ es la generación pronosticada).

$$\begin{cases} a \cdot t_{-1} + b = g_{-1} \\ a \cdot t_{-2} + b = g_{-2} \end{cases} \Rightarrow \begin{cases} \tilde{g}_i = a \cdot t_i + b \\ si\ \tilde{g}_i < 0 \ \Rightarrow \ \tilde{g}_i = 0 \end{cases} \quad \forall\ i = 0.,7$$

La figura 4.14 muestra un ejemplo donde las rectas son los pronósticos en esos intervalos de tiempo. La generación, en este ejemplo, corresponde a un día soleado. Y en el anexo A.9 aparece la implementación.

4.4.8. Módulo Fuzzy Logic

Matlab: Fuzzy Logic Designer

Antes de empezar a describir el módulo de Python en el cual se implementó el controlador fuzzy, se hablará de la herramienta que se utilizó de Matlab, el Fuzzy Logic Designer. Esta herramienta brinda la posibilidad de modelar las funciones de membresía, tanto para las entrada del controlador, como para la salida; el diseño

de las reglas de la lógica difusa y la posibilidad de elegir bajo qué tipo se aplican las funciones lógicas and, or y defuzificación.

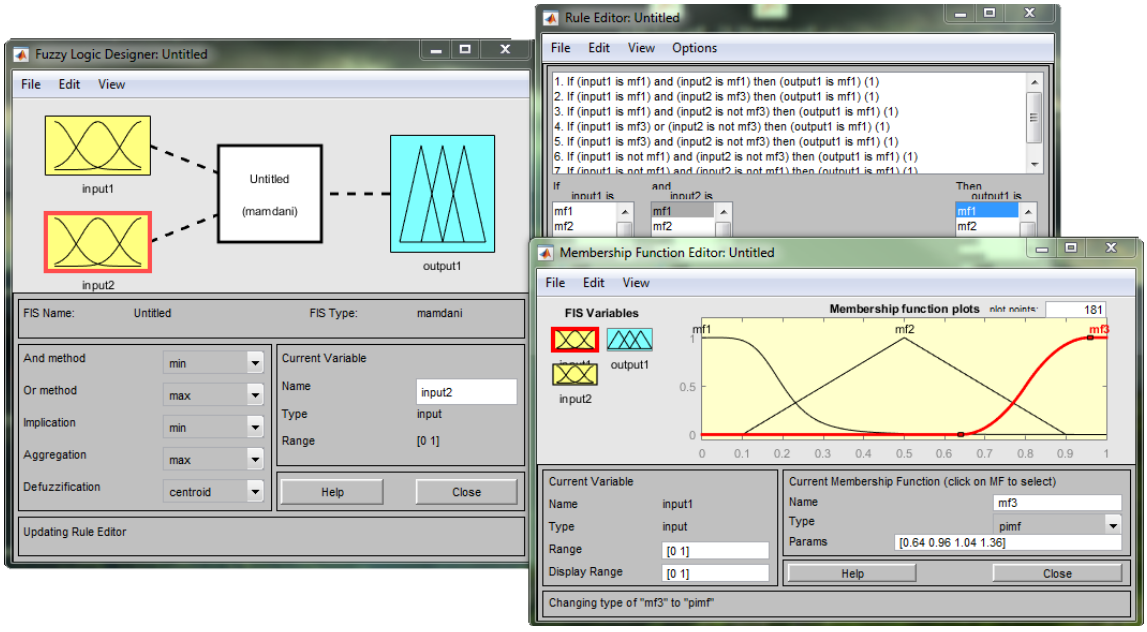


Figura 4.15: Ventanas de la herramienta de Matlab: Fuzzy Logic Designer.

La figura 4.15 muestra las tres ventanas más importantes de la herramienta: a la izquierda se ve la ventana principal, en la que se definen la cantidad de entradas y salidas de controlador, y el tipo de funciones lógicas que se van a utilizar; la ventana izquierda superior, corresponde al bloque de las reglas que se van a diseñar; y por último la ventana izquierda inferior, corresponde al diseño de las funciones de membresía que se utilizan para fuzificar las entradas y defuzificar las salidas.

El uso que se le dio a esta herramienta fue para aprender, con algunos ejemplos sencillos, cómo trabaja la lógica difusa. Se utilizó además para el diseño de las funciones de membresía.

Librerías de Python para el fuzzy

La librería que se utilizó para implementar el fuzzy en Python es skfuzzy[18], el cual provee un conjunto de herramientas para el diseño de este tipo de controladores. Esta librería está hecha para utilizarse en conjunto con las herramientas que provee el ecosistema basado en Python de herramientas matemáticas Scipy[19].

[18] skfuzzy: `www.pythonhosted.org/scikit-fuzzy/` link a la documentación, donde se incluyen ejemplos de uso y la API.

[19] ver Scipy: `www.scipy.org/`.

A continuación se describirán las funciones que se utilizaron de la librería skfuzzy, explicando primero las que implementan las funciones de membresía y luego las que se utilizan bajo el sub-módulo control (el cual encapsula las funciones que integran el diseño del controlador):

- **gaussmf(x, μ, σ)**: implementa la gaussiana, definiéndose a ésta como la siguiente función: $f(x) = e^{\frac{-(x-\mu)^2}{2\sigma^2}}$.
- **dsigmf(x, b_1, c_1, b_2, c_2)**: implementa la diferencia de dos sigmoides, donde se define a la sigmoide como la siguiente función: $f(x) = \frac{1}{1+e^{-c(x-b)}}$.
- **trapmf(x, a, b, c, d)**: implementa la función trapezoidal, definiéndose a ésta como la siguiente función: $f(x) = max(min(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}), 0)$.
- **Antecedent(universe, label)**: Primera clase del sub-módulo control, con ella se definen las variables de entrada al controlador. Los parámetros de entrada que se le pasan son: un arreglo que define el rango de entradas (universe), y un string con el nombre de la variable (label).
- **Consequent(universe, label)**: Análogamente con la clase anterior, esta clase define las variables de salida del controlador y los parámetros de entrada son los mismos.
- **Rule(antecedent=None, consequent=None, label=None)**: En esta clase se define una de las reglas que unen a las entradas (antecedent) con las salidas (consequent).
- **ControlSystem(rules=None)**: Esta clase crea un contenedor para todas las reglas definidas que se van a utilizar en el controlador.
- **ControlSystemSimulation(control_system)**: Esta clase es la encargada de ejecutar el controlador definido en la clase anterior. Los métodos que se usaron de ella fueron: **input(input_dict)** que recibe el valor de las diferentes entradas; **compute()** que computa estas variables; **output(output_dict)** que devuelve el resultado de la salida seleccionada.

La función heducFuzzy

La función heducFuzzy implementa el controlador diseñado para este proyecto, explicado en la sección 3.8. Recibe como entrada a las cinco variables definidas como entrada del controlador, estas son: el estado actual de la batería (**bateria**), el consumo pronosticado en el intervalo de tiempo definido (**consumo**), el costo del wH en ese rango de tiempo (**costo**), la suma del estado de la batería y generación (**batGen**), y la suma del estado de la batería más la generación menos el consumo (**batGenCon**).

Entonces observando el código presentado en el anexo A.10:

- Primero se declaran las entradas y la salida con sus rangos de trabajo y sus funciones de membresía asociados.
- Luego se definen las reglas que asocian a éstas y se crea el controlador fuzzy.
- Finalmente se crea la función ***heducFuzzy(bateria, consumo, batGen, batGenCon, costo)*** que se encarga de ejecutar el controlador, y devuelve con qué porcentaje se usa la batería.

4.4.9. Módulo de costo y consumo

Como se explica en la sección 4.4.2, el algoritmo de optimización utiliza pasos discretos de quince minutos para la toma de decisiones, particularmente toma en cuenta las condiciones a futuro en ocho pasos de quince minutos (o dos horas). Para esto es necesario cargar el costo de la tarifa y el consumo estimado para cada uno de los siguientes ocho pasos en función del momento actual en un vector. Ver en anexo A.11 la implementación.

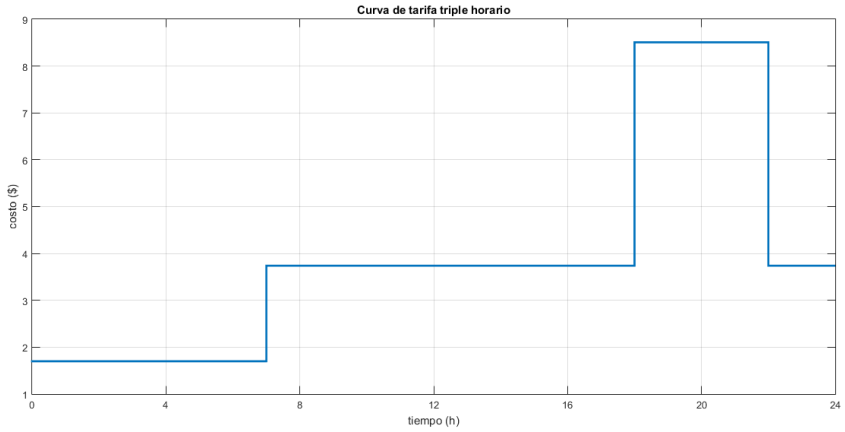


Figura 4.16: Curva de la tarifa triple horario.

- **vector_costo()**: esta función devuelve un vector con los costos en las próximas dos horas, en intervalos de quince minutos. Para esto utiliza la librería datetime, utilizando la función *datetime.now()* que devuelve la hora actual del sistema, con la cual define el costo de las dos horas siguientes en base a la tarifa triple horario, (ver figura 4.16).
- **vector_consumo()**: esta función al igual que la anterior, devuelve un vector con el pronóstico de consumo en las próximas dos horas en intervalos de quince minutos. Utiliza también la función *datetime.now()* y cruza ese dato con un vector de consumo representativo (éste tiene el consumo en intervalos de quince minutos en las 24 horas del día) o con un vector de consumo real obtenido del modulo periférico que contiene a la pinza de corriente.

La tarifa utilizada en el proyecto es la definida en el pliego tarifario de UTE como MC1. La misma plantea tres costos de kWh en cuatro franjas horarias. Si bien para contratar esta tarifa la potencia contratada debe ser mayor a los 10kW, se eligió esta forma de facturación porque se acerca mas a una tarifa dinámica y brinda mayor complejidad al sistema de optimización.

La tarifa se compone por tres franjas horarias:

- La franja denominada "Valle" comprende desde el comienzo del día hasta las 7:00 am y tiene un costo de $ 1,7 por kilowatt-hora consumido. Éste es el costo más bajo de las tres bandas horarias.
- La siguiente franja es denominada "Llano" y consta de dos intervalos horarios comprendidos de 7:00 a 18:00 y de 22:00 a 24:00. El costo asociado a estos horarios es de $ 3,739 por kilowatt-hora consumidos.
- La última franja cuyo nombre es "Punta", es la de costo más caro y queda definida como el horario de 18:00 a 22:00 horas, con un costo de $ 8,507 por kilowatt-hora de energía consumida.

Capítulo 5

Pruebas de funcionamiento del sistema

5.1. Introducción

En este capítulo se describen todas las pruebas realizadas sobre el sistema, particularmente se explicitan las condiciones en las que las mismas fueron realizadas, con qué objetivo se hicieron, los resultados obtenidos y en algunos casos modificaciones que surgieron al sistema. Por último se presentan las conclusiones extraídas de las pruebas.

5.2. Prueba de funcionalidad de los nodos periféricos

Como se define anteriormente, el sistema cuenta con tres nodos periféricos (ver 2.2.3).

Para evaluar su funcionalidad, se dejaron los nodos funcionando en una habitación de prueba con el nodo central en línea, comprobando que el nodo central fuera capaz de recibir el flujo de información a través de la red MQTT, y a su vez, que los nodos periféricos pudieran recibir órdenes, desde el nodo central. El período de prueba dispuesto fue de 15 días, para tener un control de la transferencia de datos entre los nodos se creó un script en Python que enviaba cada dato recibido a la plataforma Thingspeak (ver capítulo 4.2.5).

Cabe destacar que durante esta prueba no se realizaron pruebas sobre la calidad de las medidas recogidas, ya que este proceso fue realizado durante el diseño de cada nodo, con esta prueba solamente se buscaba confirmar que flujo de información era posible recibir sin tener problemas de comunicación.

El resultado de ese período de prueba fue que, bajo la red de comunicación definida, es posible disponer de todos los datos necesarios en el nodo central en un tiempo de muestreo de 30 segundos.

5.3. Prueba del algoritmo de optimización

El resultado del ahorro para el sistema de microgeneración depende de dos factores: uno es la precisión del pronóstico de generación, mientras que el otro es el resultado de las decisiones que toma el algoritmo de optimización. Para realizar la prueba del algoritmo se individualizó el funcionamiento de éste por separado de las función de pronóstico de generación.

Para verificarlo, se utilizaron muestras de generación obtenidas a partir del módulo en Python del panel fotovoltaico (ver 4.4.3) y series de consumo de dos tipos: una a partir de las muestras tomadas por la pinza amperimétrica en la habitación de prueba, y por otro lado, series de consumos diarios que representan diferentes dinámicas de consumo.

Para evaluar el éxito del funcionamiento del sistema se comparó el resultado obtenido con éste, en contraposición con un sistema no optimizado[1].

5.3.1. Datos introducidos y parámetros del sistema

Para realizar las pruebas se consideraron tres series de generación de un largo de 6 días, cuyos datos fueron adquiridos en el período del 24 de febrero al 31 de marzo. Cada serie se eligió en correspondencia a una generación baja, media y alta. Para generar las muestras se toma el promedio de potencia generada en los 96 períodos de 15 minutos a lo largo del día. Luego se divide el promedio entre cuatro para obtener el valor de la energía generada en Watts/hora.

Los valores de generación total a lo largo de los 6 días para cada caso fueron los siguientes: 13,64kWh, 24,86 kWh y 37,18 KWh para el caso de generación baja, media y alta, respectivamente. Las gráficas con los valores de potencia generada se pueden ver en la Figura 5.1.

Las referencias utilizadas para catalogar cada muestra de generación fueron obtenidas del mapa solar del Uruguay versión 1[2], del cual se desprende que la radiación promedio que impacta sobre cada metro cuadrado en un día tiene un máximo, en el mes de Enero, con 6,4 kWh al día y un mínimo, en Junio, de 1,9 kWh al día. Por lo tanto, haciendo un cálculo primario en el cual se multiplican los valores de radiación por la eficiencia del panel, el área de éstos y la cantidad de paneles presente en el arreglo, se obtiene que:

- La generación promedio en Junio tiene un valor de 2,18kWh al día, y se asemeja en buena medida a 2,26kWh diarios obtenidos para la serie de baja generación.

- La generación promedio en un día de Enero es de 7,33kWh, en comparación a los 6,19kWh que se obtienen en la serie para alta generación.

[1] El sistema no optimizado fue modelado con un funcionamiento tal, que siempre que el SOC del banco de baterías sea mayor que la tensión mínima de descarga se da prioridad a la batería, consumiendo del banco la energía necesaria.

[2] Ver `https://www.fing.edu.uy/if/solar/msu-miem-v1.pdf`

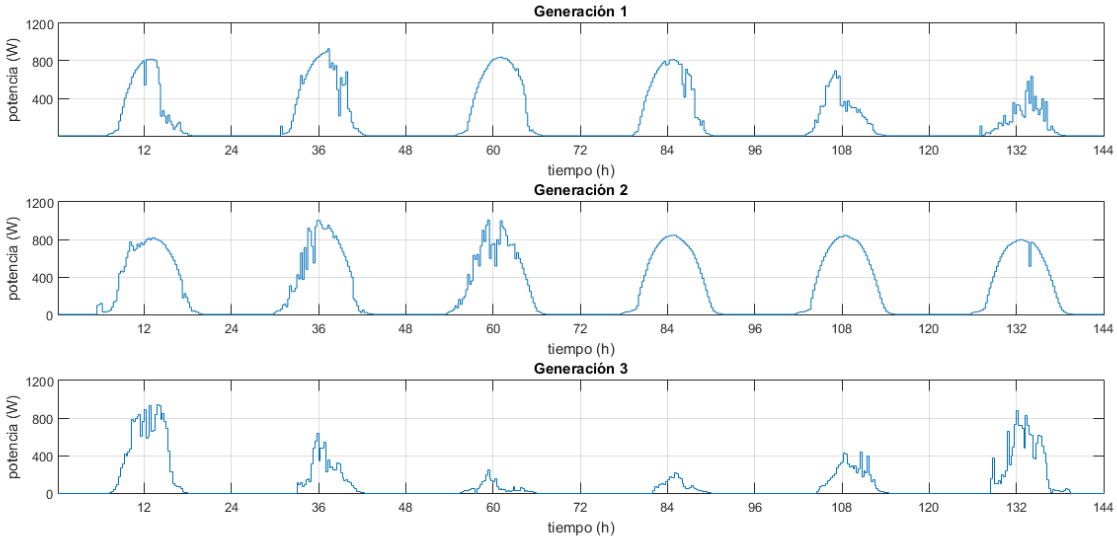


Figura 5.1: Gráficas de potencia eléctrica generada en función del tiempo para las series media (1), alta (2) y baja (3).

Cabe destacar que en el cálculo no se consideran pérdidas ni la trayectoria diaria, dado que solo se conoce el promedio de energía que impacta sobre cada metro cuadrado. También se deja de lado el efecto de la inclinación de los paneles, por lo que los valores expresan una cota superior a lo que puede esperarse. Por último se debe tener en cuenta que los valores de radiación utilizados corresponden al departamento de Montevideo.

En cuanto al consumo (ver 5.2), se consideraron tres series en base a un consumo diario fijo, que se repite seis veces a lo largo de la simulación. Si bien el consumo total varía de una a otra[3], el objetivo es variar la distribución del consumo dentro de las distintas franjas horarias, cambiando así el costo que tiene abastecer la habitación, para verificar el comportamiento del bloque Fuzzy y del árbol de decisión. Finalmente, se tomó una muestra del consumo en la habitación de prueba entre los días 14 y 19 de marzo de 2017. El consumo de la habitación de prueba a lo largo de los seis días es de 47,1kWh, con un promedio de 7,84kWh al día.

Los parámetros del sistema definidos para realizar las pruebas de funcionamiento del sistema de optimización fueron los siguientes:

- Pasos de 60 minutos.
- Capacidad del banco de baterías de 3600 Watts/hora.

[3]Con valores de 8,65 KWh, 4,68 KWh y 3,22KWh

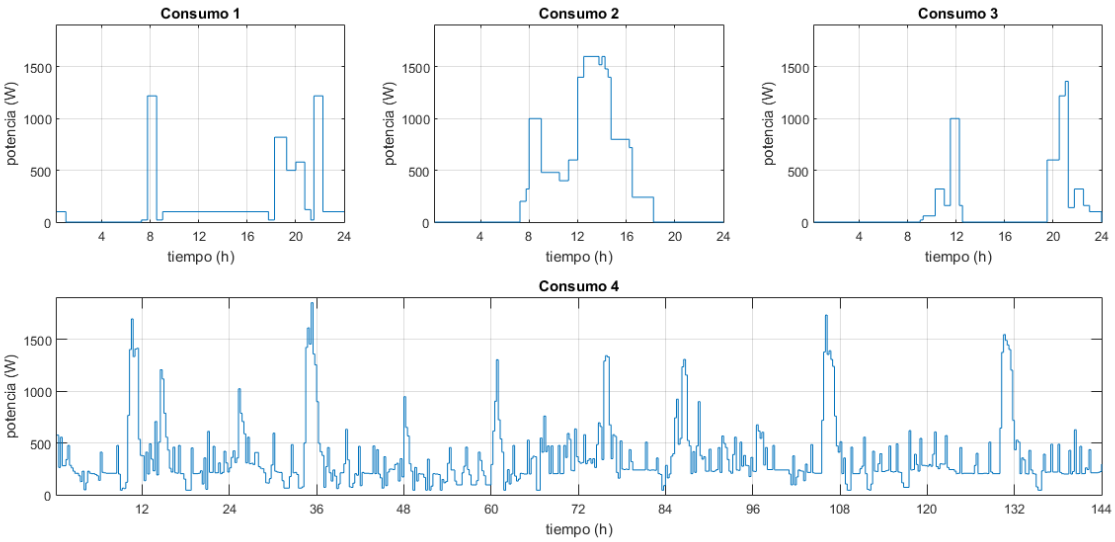


Figura 5.2: Gráfica de potencia, en función del tiempo, para los consumos utilizados en las pruebas del algoritmo de optimización

5.3.2. Primera Prueba: Variación de Consumo

La primera prueba fue realizada con el fin de ver el comportamiento del bloque fuzzy ante variaciones en el consumo. Se realizaron tres simulaciones, dejando fija la generación en su serie baja (ver 5.1), utilizando los valores de demanda modelada a lo largo de 6 días (ver primeros tres consumos de la Figura 5.2).

Los resultados obtenidos fueron los siguientes; en el caso del Consumo 1, el ahorro fue del 6,59 %, para el Consumo 2, un 0,14 % y para el Consumo 3 un 11,57 %. Estos resultados son positivos, ya que indican que el controlador fuzzy pudo optimizar el sistema en todos los casos. En particular, es relevante destacar que en el caso dónde el ahorro fue menor al 1 % (Consumo 2), se da que tiene una demanda de energía que no es propicia para el uso del bloque de optimización, ya que no hay consumo en la franja de horario más caro donde la gestión de la energía almacenada pesa más en la ecuación del gasto. Igualmente el sistema tuvo un resultado favorable, ya que en condiciones adversas el método es mínimamente superior al sistema no optimizado.

Caso 1: Consumo 1 con Generación Baja

En el primer caso de estudio (ver Figura 5.3), se observa un comportamiento similar de ambos sistemas tanto en el día 1 como en el día 6[4], donde se llega prácticamente a tener toda la capacidad de los bancos de batería disponible. Esto hace que el margen de ahorro del sistema sea menor, en comparación con los siguientes casos, debido a que el sistema no optimizado puede abastecer la demanda

[4]En el día 1 tiene un comportamiento idéntico entre las 11 y las 24, mientras que el día 6 desde las 9 a las 22 tienen una diferencia constante de 163Wh.

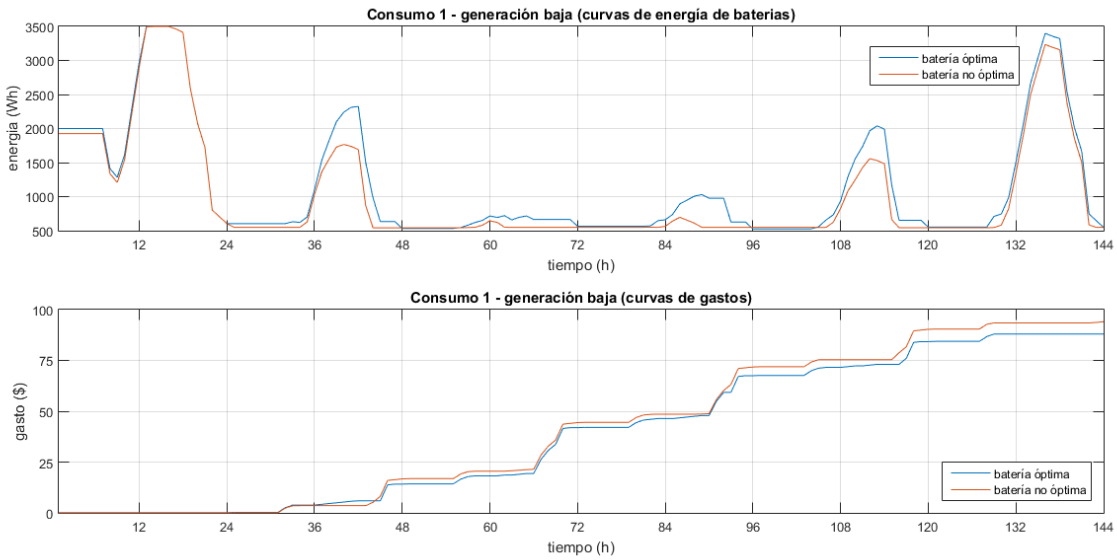


Figura 5.3: Gráficas de estado de carga de batería y gasto a lo largo de la simulación para el Consumo 1 y la serie de generación baja.

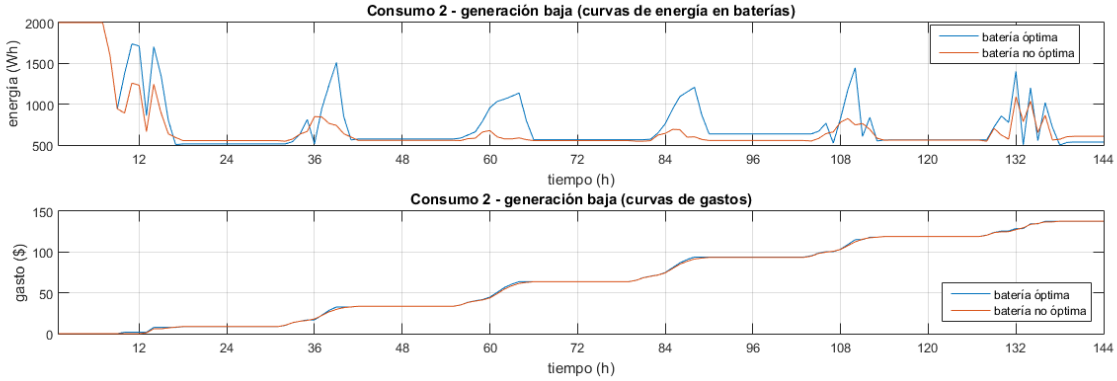


Figura 5.4: Gráficas de estado de carga de batería y gasto a lo largo de la simulación para el Consumo 2 y la serie de generación baja.

y a su vez acumular energía para más adelante. Igualmente, como el consumo en las horas siguientes es elevado, permite que en los días sucesivos se originen diferencias a favor del sistema optimizado.

Caso 2: Consumo 2 y Generación Baja

En las gráficas correspondientes a la dinámica de Consumo 2 (ver Figura 5.4), se observa como, si bien las trayectorias para el estado de carga tienen diferencias, no se establece ninguna distancia significativa entre los gastos que genera cada sistema. Ésto se debe a que las diferencias en el almacenamiento no son sostenidas sino que se equilibran rápidamente y siempre dentro del mismo rango de tarifa.

Por último, cabe destacar que el resultado obtenido es idéntico, tanto en el almacenamiento final como en el gasto originado. Esto se explica por el hecho de que el consumo se da casi totalmente superpuesto con la generación (ver Consumo 2 en Figura 5.2), lo que hace que el recurso de almacenamiento pierda peso en el sistema.

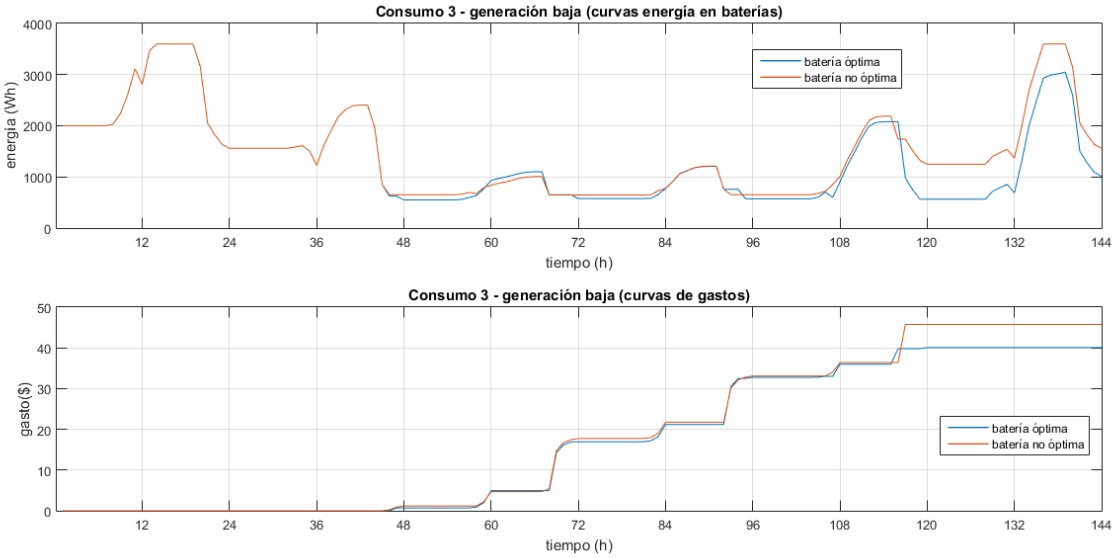


Figura 5.5: Gráficas de estado de carga de batería y gasto a lo largo de la simulación para el Consumo 3 y la serie de generación baja.

Caso 3: Consumo 3 y Generación Baja

En la gráfica para el consumo 3 (ver Figura 5.5)se puede ver como durante las primeras 46 horas de la simulación, la evolución del estado de carga es totalmente igual y el gasto es nulo. Esto se debe a que el sistema no optimizado puede abastecer la habitación sin llegar nunca a una descarga total de baterías y que el sistema de optimización decide consumir siempre desde batería. La diferencia en performance se da en el momento que la generación aumenta moderadamente, a partir del quinto día de la simulación, donde las decisiones del sistema llevan a un funcionamiento más eficiente.

5.3.3. Segunda Prueba: Variación de Generación

En una segunda instancia se buscó comprobar el funcionamiento del controlador fuzzy, variando las series de generación, con un consumo fijo, en este caso, el de la habitación de prueba. Nuevamente la duración de cada simulación es de 6 días.

En las condiciones de esta prueba, el ahorro promedio se sitúa en 13,4 % donde se observa que el gasto para ambos sistemas aumenta conforme baja la generación.

Por otro lado, analizando los valores finales de la energía acumulada en baterías, en el caso de generación baja, se observa una diferencia de 4Wh en favor del sistema optimizado, mientras que para los demás casos, se obtienen diferencias a favor del sistema no optimizado de 128Wh para el caso medio, y 418Wh para el caso alto. Esta energía almacenada se tomó en cuenta para el análisis de la gestión de la energía en cada caso.

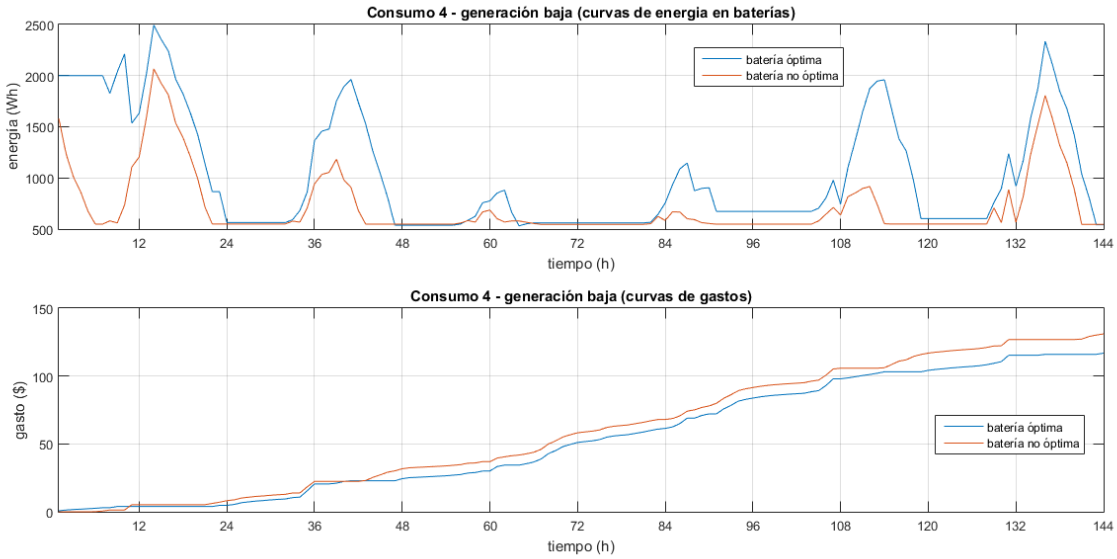


Figura 5.6: Gráficas de estado de carga de batería y gasto a lo largo de la simulación para el consumo de la habitación de prueba con una generación baja.

Caso 1: Baja Generación

Primeramente se analizará el caso con la serie de generación baja (ver Figura 5.6). En la gráfica se observa cómo, según los mecanismos de decisión, se determina que en el rango de tarifa media a lo largo de los días el estado de carga del sistema optimizado permanece siempre por encima de la batería no optimizada. Esto permite que en los días 2, 5 y 6 el sistema optimizado llegue con energía suficiente al horario de tarifa de punta, pudiendo abastecer la demanda de la habitación, mientras que el sistema no optimizado sólo llega a abastecer dos de esas cuatro horas. Finalmente, se establece una diferencia de $17,44 en favor del sistema optimizado, siendo el ahorro total $14,18.

Caso 2: Generación Media

Pasando al análisis del caso con generación media (ver Figura 5.7), en la gráfica se observa cómo ambos almacenamientos comienzan con una carga de 2000Wh, y mientras el sistema no optimizado consume la energía inicial en las primeras 6 horas, el sistema no optimizado reserva dicha energía, ya que el modelo se encuentra

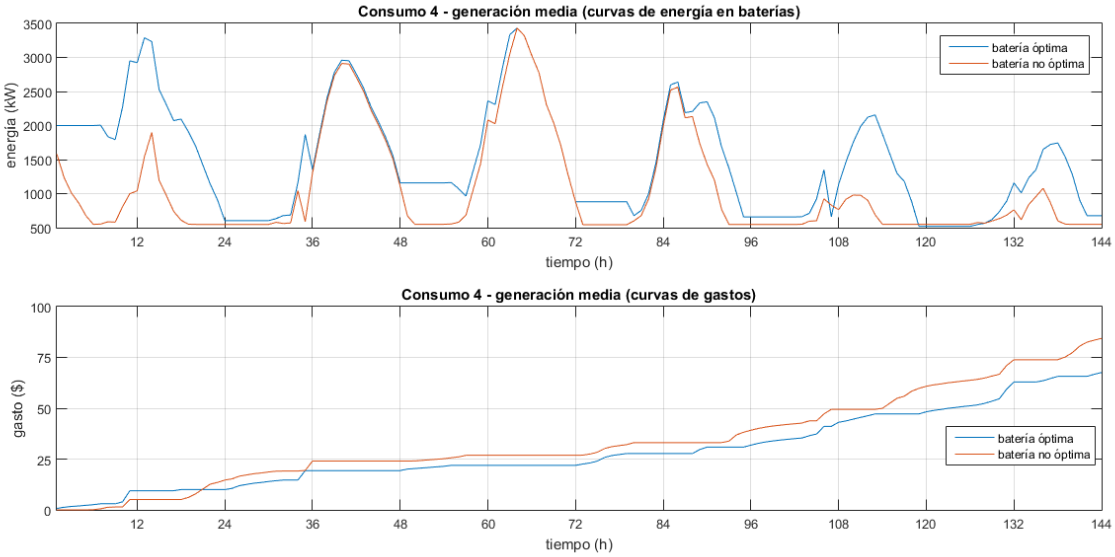


Figura 5.7: Gráficas de estado de carga de batería y gasto a lo largo de la simulación para el consumo de la Habitación de prueba con una generación media.

en el período de tarifa más bajo. En este caso, a diferencia de la simulación con generación baja, el sistema no optimizado queda sin energía disponible antes del horario de tarifa de punta, con lo cual, para la hora 21:00, el gasto del sistema óptimo pasa a ser menor.

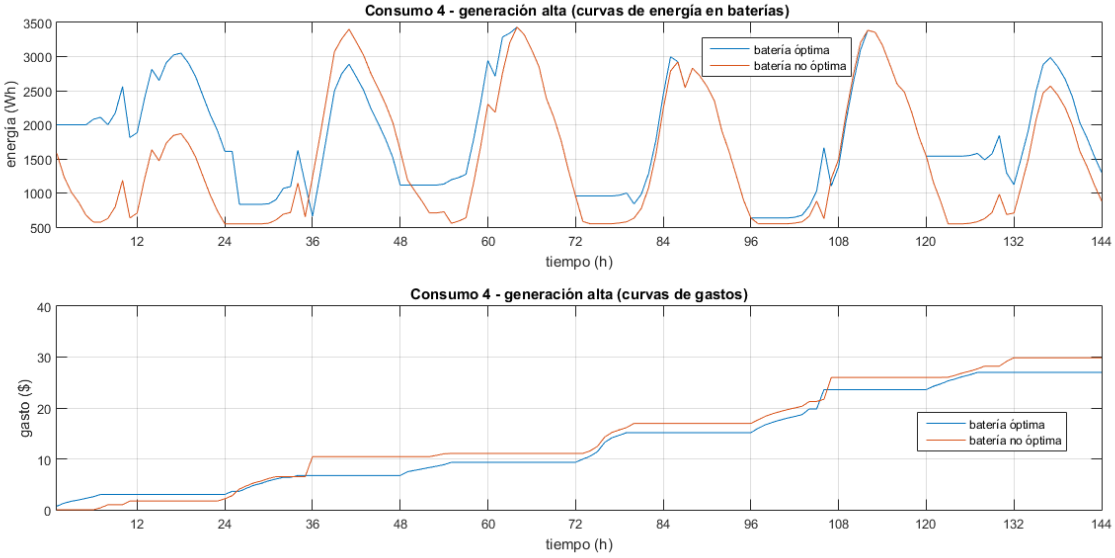


Figura 5.8: Gráficas de estado de carga de batería y gasto a lo largo de la simulación para el consumo de la habitación de prueba con una generación alta.

Caso 3: Generación Alta

Por último, respecto al caso con generación alta (ver Figura 5.8) se observa que el sistema no optimizado logra cubrir la demanda del rango de punta en su totalidad. Esto sucede debido al aumento de la generación respecto a las otras simulaciones.

Otro dato relevante es que en los días 2, 3 y 5 de la simulación, el banco no optimizado alcanza la máxima capacidad de almacenamiento. Ésto, sumado a que en los tres días la generación supera al consumo en el tiempo que los paneles se encuentran generando, deja al sistema con poco margen para optimizar.

Si nos encontráramos en una situación donde la generación supera permanentemente al consumo, el caso de estudio perdería sentido, ya que el sistema no optimizado no generaría gasto alguno. Sin embargo, el resultado obtenido permite inferir que existe una combinación de paneles y batería óptima para este consumo[5].

5.4. Prueba del sistema con pronóstico

5.4.1. Introducción

Luego de realizar una prueba aislada sobre el algoritmo de generación, se procede a evaluar el pronóstico de generación planteado.

5.4.2. Implementación del pronóstico

Como primera medida para la implementación de un pronóstico para la generación de los paneles fotovoltaicos, se procede a modificar el paso del árbol de decisión de 60 minutos a 15 minutos. La razón para este cambio es que mantener dentro de los margenes deseados el error introducido por el pronóstico, el cual es menor pronosticando dos horas hacia adelante que ocho horas.

Entonces, para realizar la comparación de resultados con y sin pronóstico, se realiza un caso de referencia, en las condiciones de la prueba 2, con la única variación de que cada paso es de 15 minutos.

El resultado será menos eficiente que en la Prueba 2 debido principalmente a que en el caso de los pasos de 60 minutos, el controlador fuzzy tiene información exacta de las próximas 8 horas, mientras que en el paso de 15 minutos tiene solamente 2 horas por delante de información, igualmente exacta. Sin embargo, una vez introducido el pronóstico, los pasos más cortos hacen que el pronóstico introduzca menos error (por la forma en que fue definido[6]). Adicionalmente, como la

[5]Definiendo el consumo diario, se podría especificar las características de un sistema fotovoltaico que consiga la máxima eficiencia de consumo con el mínimo costo, teniendo en cuenta la generación a lo largo del año.

[6]Al ser una interpolación lineal entre los últimos dos valores, la tendencia es que diverja de la curva de generación, ya que esta última es generalmente parabólica. Por lo tanto, cuanto más adelante en el tiempo se esté pronosticando, mayor es el error introducido. Ver

rutina de optimización se ejecuta cada 15 minutos, rápidamente se hace un nuevo pronóstico de generación y se toma una nueva decisión por parte del controlador, lo que hace que de introducirse un error en el pronostico podrá ser corregido mas rapidamente por el sistema, afectando a este por menos tiempo.

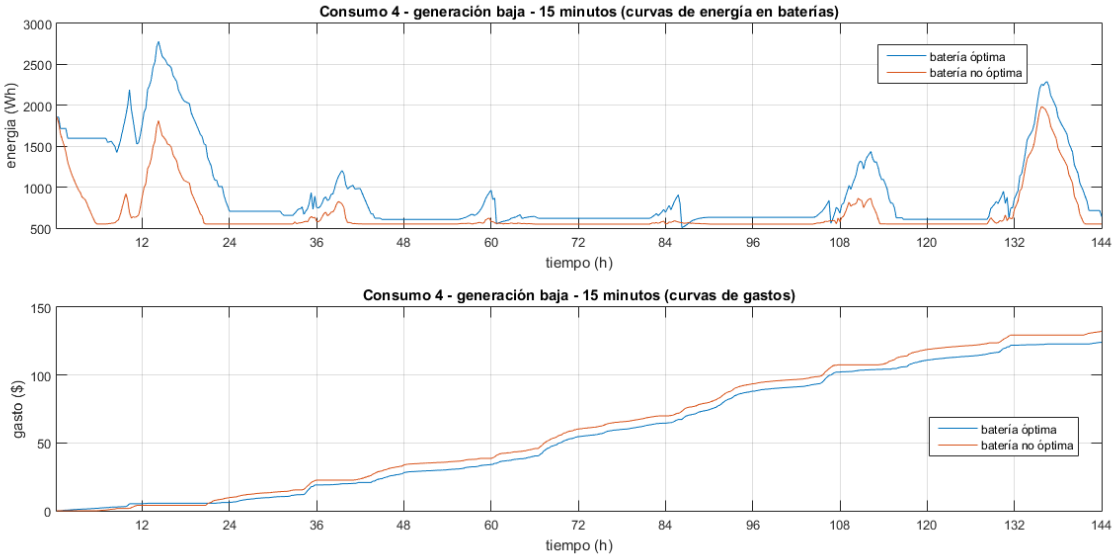


Figura 5.9: Gráficas de estado de carga de batería y gasto a lo largo de la simulación con paso de 15 minutos para el Consumo 4 con generación baja.

Comparando los resultados obtenidos de las simulaciones del consumo de la habitación de prueba[7] con pasos de 60 minutos (Figuras 5.6 y 5.7) contra pasos de 15 minutos(Figuras 5.9 y 5.10), se observa una disminución del ahorro de 10,8 % a 6,3 % en el caso de baja generación y de 19,8 % a 12,7 % en el caso de generación media.

Al comparar las simulaciones antedichas con el pronóstico (ver Figuras 5.11 y 5.12), se obtiene que el ahorro en la generación media se reduce de 12,7 % a 12,1 %, mientras que para la generación baja, aumenta de 6,3 % a 6,7 %, obteniéndose un ahorro promedio de 9,4 % frente al 9,5 % obtenido previo a la optimización, por lo que el pronóstico definido es bueno para esta aplicación.

Para analizar estos resultados nos centraremos en dos aspectos:

Primeramente, el resultado del pronóstico en comparación con la generación real en el mismo intervalo tiene un error relativo de 23 %, aumentando a partir de una hora después del momento que se está evaluando. Pero el impacto de este error se minimiza ejecutando una nueva optimización luego de transcurridos 15 minutos,

Capítulo 4.4.7, en particular, Figura 4.14

[7]Consumo 4, ver Figura 5.2

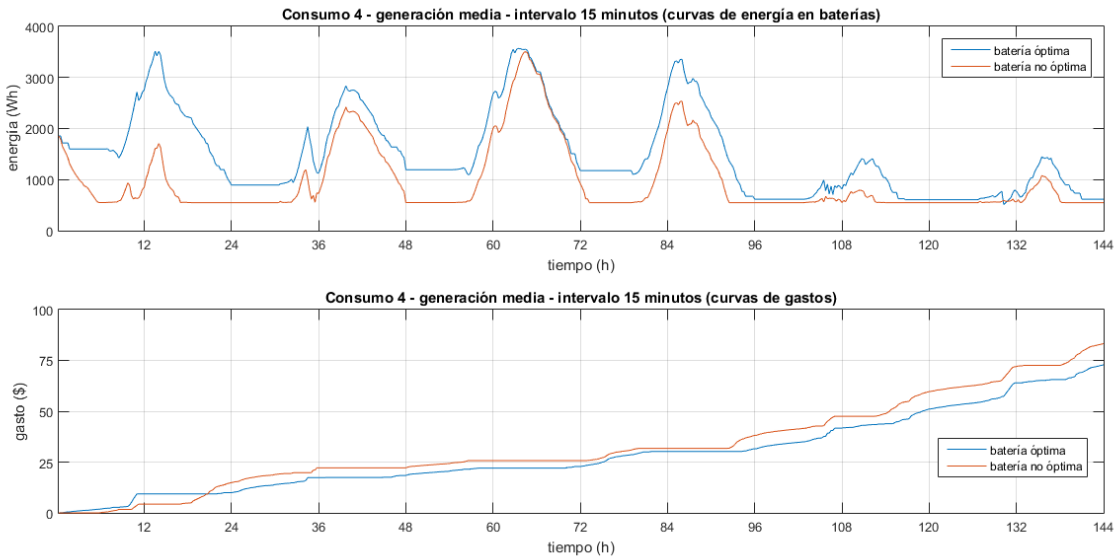


Figura 5.10: Gráficas de estado de carga de batería y gasto a lo largo de la simulación con paso de 15 minutos para el Consumo 4 con generación media.

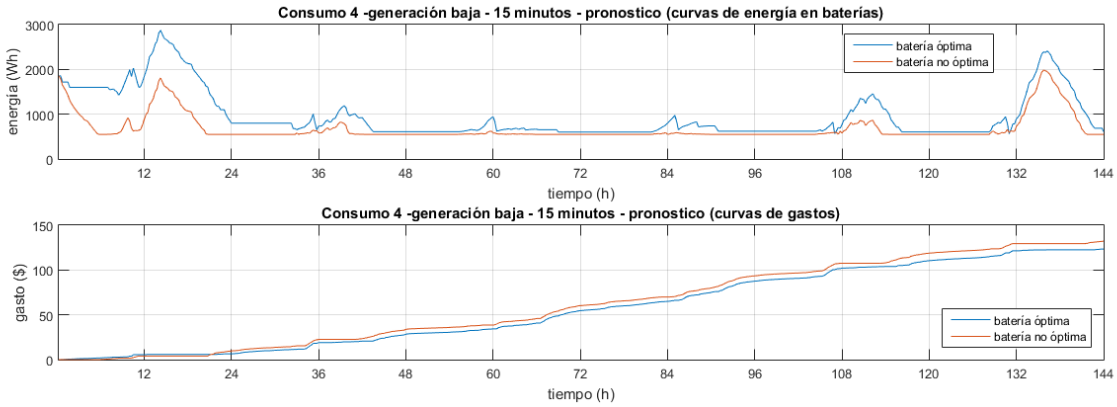


Figura 5.11: Gráficas de la simulación con paso de 15 minutos para el Consumo 4, con generación baja y pronóstico implementado.

con lo cual el tiempo en que el sistema funciona en base a datos del pronóstico menos fiables es reducido.

En segundo lugar, la forma en que se plantea la lógica difusa en el controlador disminuye la incidencia del error. La generación no es en sí misma una entrada del sistema sino que el algoritmo fuzzy recibe una entrada llamada bat+gen que ingresa el valor del pronóstico sumado con el estado actual de la batería[8]. Entonces, si bien el error relativo analizado contra la generación puede ser alto, se ve atenuado a la

[8]lo que intuitivamente sería la energía disponible en los próximos 15 minutos

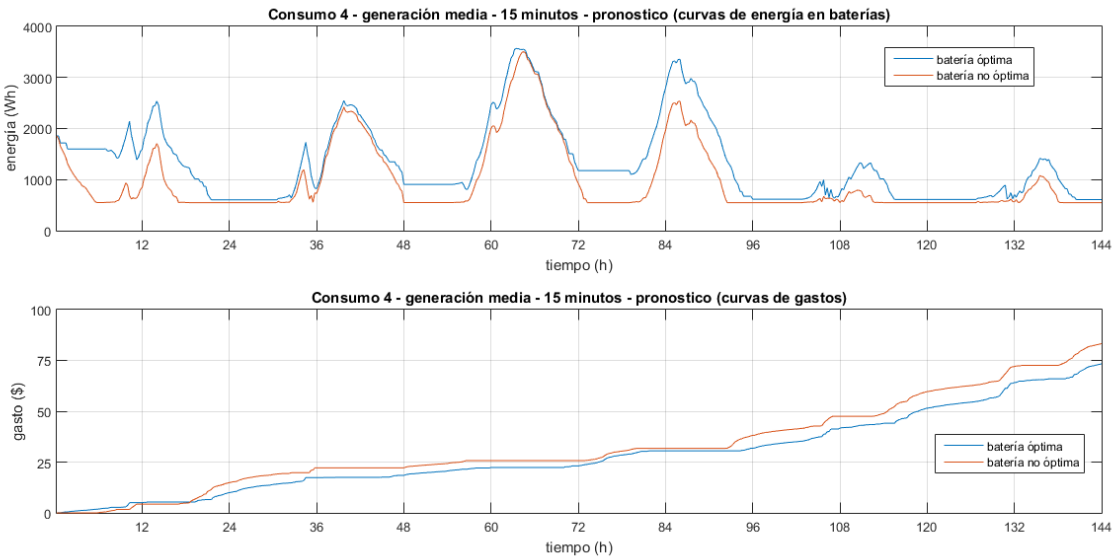


Figura 5.12: Gráficas de la simulación con paso de 15 minutos para el Consumo 4, con generación media y pronóstico implementado.

entrada del algoritmo cuando es sumado con el nivel de carga del almacenamiento[9].

Luego, como se puede ver en la Figura 3.5, la entrada bat+gen se fuzzifica asignándole un valor de pertenencia para tres atributos (baja, media y alta) según la función de membresía. Dicho valor de pertenencia es el utilizado al evaluar las reglas del sistema fuzzy, y por la manera en la que se construye, tiene alta tolerancia a ruidos en la entrada[10].

[9]la capacidad de almacenamiento en baterías es de 3600Wh y, en pasos de 15 minutos, el pico de generación es de 250Wh.

[10]Ante variaciones a la entrada el valor de pertenencia tiene un cambio pequeño y acotado

Capítulo 6

Conclusiones y recomendaciones

6.1. Conclusiones

Se definió e implementó un sistema inalámbrico de sensores y actuadores, mediante el cual es posible muestrear y transmitir los parámetros ambientales de una habitación, así como actuar sobre los elementos eléctricos de la misma. Adicionalmente, sobre la misma infraestructura se obtienen y reportan los datos necesarios para ser utilizados en un algoritmo de optimización en la gestión de recursos energéticos. Por otra parte, se desarrolló el algoritmo antedicho y se probó su funcionamiento en una habitación de prueba, para lo cual se modeló un sistema de generación fotovoltaica con almacenamiento local en baterías, obteniéndose resultados alentadores para distintas series de generación. En promedio, las distintas series mostraron un 9,4 % de ahorro respecto al sistema convencional: fotovoltaico off-grid.

La optimización en la gestión de la energía se basa en el control del almacenamiento de la misma en el banco de baterías del sistema fotovoltaico local, utilizando lógica difusa. El controlador desarrollado utiliza como entradas al sistema el pronóstico de consumo, el pronóstico de generación, la energía almacenada en baterías y el costo del kW en el momento (se utilizó la tarifa triple horario vigente en Uruguay, pero el sistema permite ingresar otras formas de tarifa, incluso tarifa dinámica).

El desempeño del sistema desarrollado es perfectible; como se explica a lo largo de la memoria, hay margen de mejora del ahorro obtenido, principalmente por el lado de la gestión de la demanda. Igualmente, es destacable que el costo del mismo fue de U$S 81 [1], con lo cual, en las condiciones del caso de prueba, se pagaría completamente en 60 meses[2], sin contar el costo ni el ahorro generados por el sis-

[1]Este valor de referencia, contempla el costo de adquirir todos los componentes del sistema al momento de la realización del proyecto. No incluye el costo de un sistema fotovoltaico offgrid, ni el trabajo invertido en horas de desarrollo.

[2]El costo del consumo en la habitación sin optimizar supone $407,32 al mes (prome-

tema de generación Off Grid. Es relevante destacar que si el consumo es mayor, el sistema se paga antes, ya que el costo de implementación es fijo.

Se deja el registro en esta memoria de las pruebas realizadas, los resultados obtenidos, los procedimientos y herramientas utilizadas, así como también una descripción detallada de los elementos de Hardware utilizados, su configuración y el código desarrollado, para poder continuar las investigaciones iniciadas con este trabajo.

6.1.1. Análisis según criterios de éxito

- **Diseñar e implementar una red inalámbrica que permita el intercambio de datos entre la central y los dispositivos periféricos**.

  Se considera que el sistema diseñado cumple en su totalidad con los requisitos planteados originalmente, y funcionó correctamente durante el tiempo de prueba, centralizando los datos de los nodos periféricos y llevando el registro de los valores relevantes en la plataforma Thingspeak.

  Particularmente, dichos valores fueron los que se utilizaron para las simulaciones, descargados de la plataforma en formato CSV.

- **Lograr procesar la información del sistema, con histórico de las variables monitoreadas, que permita caracterizar las mismas**.

  Se considera que también fue alcanzado el nivel de desarrollo necesario para tales fines, en particular, todos los resultados de las simulaciones, así como los modelos de panel FV, baterías y el controlador basado en fuzzy se corrieron en Python, en la placa BeagleBone.

- **La definición de una rutina que implemente, basándose en la estimación climática, las condiciones del sistema y el costo del kW hora, la toma de decisiones en tiempo real sobre el uso de las fuentes de energía: red y solar.**

  Se consideran cumplidos estos requerimientos, se desarrolló un controlador que implementa una rutina de optimización basada en lógica difusa, con las entradas antedichas. Adicionalmente, se probó un método de pronóstico desarrollado en el proyecto con buenos resultados para esta aplicación.

- **La implementación y optimización de la rutina anterior.**

  Se implementó la rutina de optimización y se probó el funcionamiento de la misma con diferentes parámetros de entrada, variando la generación, el

diando los costos en las tres series de generación), por lo que el ahorro supone unos $38,28 en igual período, lo que equivale a US$ 1,35 por mes.

consumo y el pronóstico. En todos los casos, el sistema logró disminuir los costos.

6.2. Recomendaciones y trabajo a futuro

A fin de mejorar aspectos puntuales del proyecto o profundizar en algunos que fueron tratados superficialmente se proponen las siguientes acciones:

- Mejorar la precisión del pronóstico de generación, ya sea extendiendo la técnica utilizada, buscando considerar más puntos que los dos actuales o tomando una forma de ajuste más adecuada que una recta. O bien definir y utilizar un método distinto.
- Probar el sistema con otros métodos de microgeneración y la optimización del consumo respecto a más de una fuente de generación en simultáneo.
- Utilizar los actuadores y dispositivos implementados para realizar la gestión de la demanda. Adicionalmente, se puede realizar una rutina que optimice la gestión. Dicha rutina se puede desarrollar utilizando lógica difusa.
- Realizar una prueba del sistema utilizando como entrada una tarifa dinámica, en principio con posibilidad de variar hora a hora, para verificar el comportamiento del sistema frente a una entrada de este tipo. Podría ser con el formato que utilizará UTE para este tipo de tarifa.
- Estudiar con mayor criterio la implementación del controlador fuzzy, por ejemplo: estudiar la posibilidad de agregar mas reglas, ya que se utilizaron 11 de 512 reglas posibles.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice A

Anexos: Software

A.1. NodoExterior.ino

```
#include "FotoDiodo.h"
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "rDHT11.h"

int rDHT11pin = 5;
rDHT11 DHT11(rDHT11pin);
int result;

const char* ssid = "ssid";
const char* password = "password";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg1 = 0;
long lastMsg2 = 0;
long now;
int i=0;
char msg[10];
char msg1[10];
char msg2[10];

int Temp = 0;
float Irra;
int Irradiancia;

void setup() {
  Serial.begin(115200);
  setup_wifi();
```

```
  client.setServer("m10.cloudmqtt.com", 13579);
  client.setCallback(callback);
}

void setup_wifi() {

  delay(10);
  Serial.println();
  Serial.print("Conectando con ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println("Direccion IP: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Mensaje arrivado [");
  Serial.print(topic);
  Serial.print("] ");
  for(i=0; i<length; i++) {
    msg[i] = payload[i];
  }
  msg[i] = '\0';
  String msgString = String(msg);
  Serial.println("Payload: " + msgString);
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Intentando coneccion con MQTT.");
    if (client.connect("ESPClient", "userMQTT", "passwordMQTT")) {
      Serial.println("conectado");
      client.publish("outTopic", "Nodo Exterior Conectado");
      client.subscribe("#");
    } else {
      Serial.print("fallo , rc=");
      Serial.print(client.state());
```

```
      Serial.println("tratar de nuevo en 5 seg.");
      delay(5000);
    }
  }
}
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  now = millis();
  if (now - lastMsg1 > 2800) {
    lastMsg1 = now;
    Irra = irradiancie();
    Irradiancia = (int)Irra;
    while( (result = DHT11.update()) != OK);
    Temp = DHT11.getCelsius();
    snprintf (msg1, 75,"%d", Irradiancia);
    snprintf (msg2, 75,"%d", Temp);
    Serial.print("Publicar mensajes: ");
    Serial.print(Irra, DEC);
    Serial.println(Temp);
    client.publish("HEDUC/Irradiancia", msg1);
    client.publish("HEDUC/TempExt", msg2);
  }
}
```

A.1.1. FotoDiodo.h

```
int irradiancie(void);
```

A.1.2. FotoDiodo.cpp

```
#include "FotoDiodo.h"
#include <math.h>
#include <Arduino.h>

float Cte = 16.276*0.623 ;
int irradiancie(void){
        int val = analogRead(A0);
        Serial.println(val);
        float radiance = (float)val * Cte;
        Serial.print(radiance);
        Serial.println( " W/m2");
```

```
        return (int) radiance;
}
```

A.2. NodoConsumo.ino

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "IRremoteESP8266.h"
#include <Wire.h>
#include "LCD.h"
#include "LiquidCrystal_I2C.h"
#include "EmonLib.h"

#define I2C_ADDR    0x27

const char* ssid = "ssid";
const char* password = "password";

String apiKey = "1234567890";
const char* server = "api.thingspeak.com";

EnergyMonitor emon1;

IRsend irsend(0);

LiquidCrystal_I2C lcd(I2C_ADDR,2, 1, 0, 4, 5, 6, 7);

WiFiClient espClient;
WiFiClient client2;
PubSubClient client(espClient);

long lastMsg = 0;
long now;


int value = 0;
char msg1[20];
char msg2[20];
int potencia;

void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer("m10.cloudmqtt.com", 13579);
  client.setCallback(callback);
```

```
  emon1.current(0, 35);
  irsend.begin();
  lcd.begin (16,2);
  lcd.setBacklightPin(3,POSITIVE);
  lcd.setBacklight(HIGH);
  lcd.home ();
  lcd.print(" HEduc - OnFire");
}

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Conectando con ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println("Direccion IP: ");
  Serial.println(WiFi.localIP());
}

int i=0;
char msg[10];

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Mensaje arrivado [");
  Serial.print(topic);
  Serial.print("] ");

  for(i=0; i<length; i++) {
    msg[i] = payload[i];
  }
  msg[i] = '\0';
  String msgString = String(msg);
  lcd.setCursor ( 0, 0 );
  lcd.print("                ");
  lcd.setCursor ( 0, 0 );
  lcd.print("MQTT ");
  lcd.print(msgString);
```

```
  Serial.println("Payload: " + msgString);
  if (msgString == "21F"){
    Serial.println("21F");
    irsend.sendNEC(0xC4D3, 0x6480, 0x0024,0xC050, 0xA000, 0x0000, 0x00D2, 16);//
  }
  if (msgString == "24F"){
    Serial.println("24F");
    irsend.sendNEC(0xC4D3, 0x6480, 0x0024,0xC0E0, 0xA000, 0x0000, 0x0012, 16);//
  }
  if (msgString == "24C"){
    Serial.println("24C");
    irsend.sendNEC(0xC4D3, 0x6480, 0x0024,0x80E0, 0xA000, 0x0000, 0x0062, 16);//
  }
  if (msgString == "28C"){
    Serial.println("28C");
    irsend.sendNEC(0xC4D3, 0x6480, 0x0024,0x8020, 0xA000, 0x0000, 0x00C2, 16);//
  }
  if (msgString == "OFF"){
    Serial.println("OFF");
    irsend.sendNEC(0xC4D3, 0x6480, 0x0004, 0xC050, 0xA000, 0x0000, 0x00E2, 16);/
  }
}


void reconnect() {
  while (!client.connected()) {
    Serial.print("Intentando conexion con MQTT.");

    if (client.connect("ESP8266Client", "userMQTT", "passwordMQTT")) {
      Serial.println("conectado");
      client.publish("outTopic", "hello world");
      client.subscribe("HEDUC/IR");
      client.subscribe("HEDUC/LED");
    } else {
      Serial.print("fallo , rc=");
      Serial.print(client.state());
      Serial.println("tratar de nuevo en 5 seg.");
      delay(5000);
    }
  }
}

void loop() {

  if (!client.connected()) {
```

```
    reconnect();
  }

  client.loop();
  now = millis();

  if (now - lastMsg > 15000) {
    now = lastMsg;
    double Irms = emon1.calcIrms(1480);
    potencia = Irms*230;
    snprintf (msg1, 75,"%d", potencia);
    Serial.print("Publicar mensaje: ");
    Serial.println(potencia);
    client.publish("HEDUC/Consumo", msg1);
    lcd.setCursor ( 0, 1 );
    lcd.print("Consumo: ");
    lcd.print(potencia) ;
    lcd.print(" W");
    lcd.print("  ");

    if (client2.connect(server,80)) { // "184.106.153.149" or api.things
      String postStr = apiKey;
      postStr +="&field3=";
      postStr += String(potencia,DEC);
      postStr += "\r\n\r\n";

      client2.print("POST /update HTTP/1.1\n");
      client2.print("Host: api.thingspeak.com\n");
      client2.print("Connection: close\n");
      client2.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
      client2.print("Content-Type: application/x-www-form-urlencoded\n"
      client2.print("Content-Length: ");
      client2.print(postStr.length());
      client2.print("\n\n");
      client2.print(postStr);

      Serial.print("Irradiancia: ");
      Serial.print(potencia, DEC);
      Serial.println(" enviar a Thingspeak");
      }
      client2.stop();

    delay (1000);
  }
}
```

A.3. NodoEscritorio.ino

```
#include "distheduc.h"
#include <Wire.h>
#include "Lumen.h"
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "rDHT11.h"
#include "HeducWifi.h"

IPAddress mqtt_server(192, 168, 1, 2);

WiFiClient espClient;
PubSubClient client(espClient);

void reconnectMqtt() {
  while (!client.connected()) {
    Serial.print("Intentando coneccion con MQTT.");
    String clientId = "ESP8266Client-lumenAndHT";
    if (client.connect(clientId.c_str())) {
      Serial.println("conectado");
    } else {
      Serial.print("failed , rc=");
      Serial.print(client.state());
      Serial.println("tratar de nuevo en 5 seg.");
      delay(5000);
    }
  }
}

void callback(char* topic, byte* payload, unsigned int length){
}

int rDHT11pin =   16;
rDHT11 DHT11(rDHT11pin);
long lastCheckHT = 0;
int temp;
int hum;
char msgT[20];
char msgH[20];

long lastCheckDimmer = 0;


void setup() {
```

```
  Serial.begin(9600);
  initDist();
  GY30_Init();
  pinMode(DIMMERPIN, OUTPUT);
  initWifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop() {

  if (!client.connected()) {
    reconnectMqtt();
  }

  reconnetWifi();

  long now = millis();

  if(now - lastCheckDimmer > 500){
    lastCheckDimmer = now;
    if(dist()){
      analogWrite(DIMMERPIN,NivelDimmer());
    }
    else{
      analogWrite(DIMMERPIN,dimmerOff());
    }
  }

  if( now - lastCheckHT > 60000){
    lastCheckHT = now;
    if( DHT11.update() == OK){
      temp = DHT11.getCelsius();
      hum = DHT11.getHumidity();
      snprintf (msgT, 25, "%d", temp);
      snprintf (msgH, 25, "%d", hum);
      client.publish("heduc/indoor/temperatura", msgT);
      client.publish("heduc/indoor/humedad", msgH);
    }
  }

 if(cambioEstado()){
   char msg[4];
   if(estado()){
     snprintf (msg, 25, "ON", 0);
```

```
    }
    else{
      snprintf (msg, 25, "OFF", 1);
    }
    client.publish("heduc/indoor/presencia", msg);
 }

}
```

A.3.1. Lumen.h

```
#define DIMMERPIN 12
#define NIVELLUM 200

void GY30_Init();
int GetLumen();
int NivelDimmer();
int dimmerOff();
```

A.3.2. Lumen.cpp

```
#include <Arduino.h>
#include "Lumen.h"

byte buff[2];
int GY30_address = 0x23;
void GY30_Init(){
  Wire.begin();
  Wire.beginTransmission(GY30_address);
  Wire.write(0x10);
  Wire.endTransmission();
}

byte GY30_Read(int address){
  byte i=0;
  Wire.beginTransmission(address);
  Wire.requestFrom(address, 2);
  while(Wire.available()){
    buff[i] = Wire.read();
    i++;
  }
  Wire.endTransmission();
  return i;
}

int GetLumen(){
```

```
  if(GY30_Read(GY30_address)==2){
    float valf=((buff[0]<<8)|buff[1])/1.2;
    if(valf<0){
      Serial.print("> 65535");
      return 65535;
    }
    else{
      return (int)valf;
    }
  }
}

int Lumen;
int DimmerPIN = DIMMERPIN;
int pLum = 0;
int NivelDim[] = {1024, 1018, 1016, 1014, 1012, 1010, 1008, 1006, 1004,
int Nivellum = NIVELLUM;

int NivelDimmer(){
  Lumen = GetLumen();
  Serial.println(Lumen);
  if(Lumen<Nivellum){
    if(pLum<21){
     pLum ++;
    }
  }
  else{
    if(pLum>0){
      pLum--;
    }
  }
  Serial.println(NivelDim[pLum]);
  return NivelDim[pLum];
}

int dimmerOff(){
  pLum = 0;
  return NivelDim[pLum];
}
```

A.4. HeducAlgoritmo.py

```
from heducFuzzy import heducFuzzy
from modelobateria import bandera, estado_actual
from consumoAndCosto import *
```

```
from PronosticoGeneracion import pronosticoGeneracion
import time

# clase nodo para trabajar con facilidad en el arbol generado
class nodo:
    bateria, tiempo, probabilidad, costoAcumulado, estado = None, None, None,0,
    def __init__(self, bateria, tiempo, probabilidad, costoAcumulado):
        self.bateria = bateria
        self.tiempo = tiempo
        self.probabilidad = probabilidad
        self.costoAcumulado = costoAcumulado
    def estadoNodo(self, estado):
        self.estado = estado

# Funcion fuzzy, dado un nodo y el estado de gen, consumo y costo devuelve la pr
def fuzzy(nodo, x_generacion, x_consumo, x_costo):
    x_batGen = nodo.bateria + x_generacion
    x_batGenCon = nodo.bateria +x_generacion - x_consumo
    return heducFuzzy(nodo.bateria, x_consumo, x_batGen, x_batGenCon, x_costo)

# Clase poda para podar
class podaArbol:
    maximaProbabilidad = None
    def __init__(self):
        self.maximaProbabilidad = 0.0
    def maxProb(self, prob):
        if (prob > self.maximaProbabilidad):
            self.maximaProbabilidad = prob
    def setmaxProb(self):
        self.maximaProbabilidad = 0.0
    def poda(self, nodo):
        if (self.maximaProbabilidad / 5) > nodo.probabilidad:
            nodo.estado = 0

def heducAlgorithmo():
    # Objeto podadora para podar
    podadora = podaArbol()

    #Vectores que alimentan al algoritmo
    estadoBateria = estado_actual() #traigo el estado actual de la bateria
    vGeneracion =pronosticoGeneracion() #traigo el pronostico de generacion
    vConsumo  =vector_consumo() #traigo el pronostico de consumov
    vCosto =vector_costo() #traigo el pronostico de costo

    largo = 8
```

```
    # Arbol del algoritmo
    arbol = {'N': nodo(estadoBateria,0,float(1),float(0))}
    # Construccion del arbol
    for tiempo in range(largo):
        for keYs in list(arbol.keys()):
            if (arbol[keYs].tiempo == tiempo) and (arbol[keYs].estado =
                pBateria = float(fuzzy(arbol[keYs], vGeneracion[tiempo]
                #Rama hacia el uso de la bateri
                newKey = keYs + 'B'
                arbol[newKey] = nodo(arbol[keYs].bateria + vGeneracion[
                podadora.maxProb(arbol[newKey].probabilidad)
                #Rama hacia el uso de la red
                newKey = keYs + 'R'
                arbol[newKey] = nodo(arbol[keYs].bateria + vGeneracion[
                podadora.maxProb(arbol[newKey].probabilidad)
        #poda, entra con el dic auxiliar y devuelve los nodos validos a
        for keYs in list(arbol.keys()):
            if(arbol[keYs].tiempo== tiempo + 1):
                podadora.poda(arbol[keYs])
        podadora.setmaxProb()

    for keYs in list(arbol.keys()):
        if (arbol[keYs].tiempo==largo) and (arbol[keYs].estado == 1):
            nodoOptimo = keYs
            break

    for keYs in list(arbol.keys()):
        if (arbol[keYs].tiempo == largo) and (arbol[keYs].estado == 1) a
            nodoOptimo = keYs

    if nodoOptimo[1] =="B":
        print 'Uso bateria'
        bandera(True)
    else:
        print 'Uso red'
        bandera(False)

while(True):
    heducAlgorithmo()
    time.sleep(60*15)
```

A.5. PanelFV.py

```
import math
```

```
from scipy.optimize import fsolve
from scipy.optimize import brentq

#Panel LNSE/245P

def Pmppt(E, T):

    Vmax = 37.1 #dato, tension Voc,stc@25C,1000W/m2
    Isc = 8.74 #dato
    TCV = -0.0033 #dato %/celsius
    TCi = 0.055 #dato %/celsius

    b = 0.0701 #caracteristica V-I determinada por el punto stc
    Tv = Vmax*TCV*(T-25) #T temperatura actual en celsius
    Vmin = Vmax*(math.log(200)/math.log(1000))
    Imax = Isc/(1-math.exp(-1/b))
    Ti = 1+TCi*(T-25)*0.01
    a = E*0.001 #E = irradiancia en W/m2
    g = 1-Vmin/(Vmax+Tv)

    def dP(V):
        return a*Imax*Ti-a*Imax*Ti*math.exp(V/(b*(g*a+1-g)*(Vmax+Tv))-1/b) - V*(

    V = brentq(dP, 10, 50)
    #print "V=",V
    P = a*Imax*V*Ti-a*Imax*V*Ti*math.exp(V/(b*(g*a+1-g)*(Vmax+Tv))-1/b)
    print "P=", P
    return P
```

A.6. modeloBateria.py

```
from datetime import datetime
import dateutil.parser
import json
from consumoAndCosto import vector_costo

# funcion inversa a isoformat()
def getDateTimeFromISO(s):
    d = dateutil.parser.parse(s)
    return d

def decremento_bateria(potConsumo, timeConsumo):
    f = open('Bateria', 'r')
    data = json.load(f)
```

```
    f.close

    PotOutOld = data['PotOut']
    tiempoOutOld = getDateTimeFromISO(data['tiempoOut'])
    consumoBateria = data['consumoBateria']
    energia_disp = data['energiaDisp']

    # calculo tiempo del paso desde dos valores consecutivos
    min_out = timeConsumo.minute - tiempoOutOld.minute
    seg_out = timeConsumo.second - tiempoOutOld.second
    tiempo_out_paso = min_out * 60 + seg_out
    #calculo la potencia diferencial
    Pot_out_dif = potConsumo - PotOutOld
    # calculo la energia del paso, integral 'triangular'
    energia_paso_out = float((PotOutOld * tiempo_out_paso + Pot_out_dif
    if consumoBateria == True:
        energia_disp = energia_disp - energia_paso_out
        data['Ahorro'] = data['Ahorro'] + abs(vector_costo()[0]*energia
    else:
        data['Gasto'] = data['Gasto'] + abs(vector_costo()[0]*energia_pa
    # Guardo las nuevas variables
    data['PotOut'] = potConsumo
    data['tiempoOut'] = timeConsumo.isoformat()
    data['energiaDisp'] = energia_disp
    f = open('Bateria', 'w')
    json.dump(data, f)
    f.close

def incremento_bateria(potGeneracion, timeGeneracion):
    f = open('Bateria', 'r')
    data = json.load(f)
    f.close

    potInOld = data['PotIn']
    tiempoInOld = getDateTimeFromISO(data['tiempoIn'])
    energia_disp = data['energiaDisp']

    # calculo tiempo del paso
    min_in = timeGeneracion.minute - tiempoInOld.minute
    seg_in = timeGeneracion.second - tiempoInOld.second
    tiempo_in_paso = min_in * 60 + seg_in
    #calculo la potencia diferencial
    Pot_in_dif = potGeneracion - potInOld
    # Ep = P1*tp + (P2-P1)*tp/2 Ws--> Wh
    energia_paso_in = float(potInOld * tiempo_in_paso + Pot_in_dif * tie
```

```
    #calculo energia total
    energia_disp = energia_disp + energia_paso_in
    if energia_disp > data["Capacidad"]:
        energia_disp = data["Capacidad"]
    # Guardo las nuevas variables
    data['PotIn'] = potGeneracion
    data['tiempoIn'] = timeGeneracion.isoformat()
    data['energiaDisp'] = energia_disp
    f = open('Bateria', 'w')
    json.dump(data, f)
    f.close

def estado_actual():
    f = open('Bateria', 'r')
    data = json.load(f)
    f.close
    return data['energiaDisp']

def bandera (estado_bandera):
    f = open('Bateria', 'r')
    data = json.load(f)
    f.close

    PotOutOld = data['PotOut']
    tiempoOutOld = getDateTimeFromISO(data['tiempoOut'])
    energia_disp = data['energiaDisp']

    if estado_bandera != data['consumoBateria']:
        if estado_bandera == True:
            data['tiempoOut'] = datetime.now().isoformat()
        if estado_bandera == False:
            tiempo_bandera = datetime.now()
            tiempo_bandera_paso = (tiempo_bandera.minute - tiempoOutOld.minute)
            energia_disp = float(energia_disp) - float(PotOutOld * tiempo_bander
    data['consumoBateria'] = estado_bandera
    data['energiaDisp'] = energia_disp
    f = open('Bateria', 'w')
    json.dump(data, f)
    f.close
```

A.7. mqtt.py

```
import paho.mqtt.client as mqtt
import json
```

MQTTv31=3

```
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("#")

def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))
    if msg.topic == 'HEDUC/Consumo':
        f = open('datosMqtt', 'r')
        data = json.load(f)
        f.close
        try:
            data['PotConsumida']=float(msg.payload)
            f = open('datosMqtt', 'w')
            json.dump(data, f)
            f.close
        except:
            print 'No se pudo guardar la potencia'

    if msg.topic == 'HEDUC/Irradiancia':
        f = open('datosMqtt', 'r')
        data = json.load(f)
        f.close
        try:
            data['irradiacia']=float(msg.payload)
            f = open('datosMqtt', 'w')
            json.dump(data, f)
            f.close
        except:
            print 'No se pudo guardar la irradiancia'

    if msg.topic == 'HEDUC/TempExt':
        f = open('datosMqtt', 'r')
        data = json.load(f)
        f.close
        try:
            data['TemExterior']=float(msg.payload)
            f = open('datosMqtt', 'w')
            json.dump(data, f)
            f.close
        except:
            print 'No se pudo guardar la temperatura'
```

```
    if msg.topic == 'HEDUC/Consumo':
        f = open('datosMqtt', 'r')
        data = json.load(f)
        f.close
        try:
            data['PotConsumida']=float(msg.payload)
            f = open('datosMqtt', 'w')
            json.dump(data, f)
            f.close
        except:
            print 'No se pudo guardar la consumo'



client = mqtt.Client(protocol= MQTTv31)
client.on_connect = on_connect
client.on_message =on_message
client.connect(locahost, 9800, 60)


client.loop_forever()
```

A.8. scriptBateria.py

```
from modelobateria import decremento_bateria , incremento_bateria
import json
from modelobateriasec import *
from datetime import datetime
import time

while(True):
    f = open('datosMqtt', 'r')
    data = json.load(f)
    f.close
    #Potencia consumida en el momento
    pConsumida = data['PotConsumida']

    #Decremento de la bateria para el modelo fuzzy
    decremento_bateria(pConsumida , datetime.now() )
    #Decremento de la bateria para el modelo no optimo
    decremento_bateriaSec(pConsumida , datetime.now() )
    if estado_actualSec()<=200:
        banderaSec(False)
    elif estado_actualSec()> capacidad()/7:
```

```
        banderaSec(True)
    #Irradiancia y temperatura
    irradiacia = data['irradiacia']
    tempExterior = data['TemExterior']
    #Potencia generada
    pGeneracion = Pmppt(irradiacia , tempExterior)
    #Incremento de la bateria optimizada
    incremento_bateria(pGeneracion , datetime.now() )
    #Incremento de la bateria no optimizada
    incremento_bateriaSec(pGeneracion , datetime.now() )

    timePronosticoG = timePronosticoG +1

    if(timePronosticoG ==15 )
        timePronosticoG = 0
        f = open('datosGeneracion', 'r')
        data = json.load(f)
        f.close

        data['E2'] = data['E1']
        data['E1'] = (pGeneracion + data['gen1'])*(datetime.now() - data
        data['gen1'] = pGeneracion
        data['t1'] = datetime.now()

        f = open('datosGeneracion', 'w')
        json.dump(data, f)
        f.close

    time.sleep(60)
```

A.9. PronosticoGeneracion.py

```
from numpy import polyfit
from generacionSimulador import  ptosPronostico
import json

def pronosticoGeneracion():

    f = open('datosGeneracion', 'r')
    data = json.load(f)
    f.close

    y = [data['E2'], data['E1']]
```

```
    x = [data['t1'],  data['t1'] -1]

    coef = polyfit(x,y,1)
    #print coef
    tf = [x[1]+1+i for i in range(8)]
    #print tf
    v= [int(tf[i]*coef[0]+coef[1]+0.5) for i in range(len(tf))]
    for i in range(len(v)):
        if v[i]<0:
            v[i]=0
    return v
```

A.10. heducFuzzy.py

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

#entradas
bateria = ctrl.Antecedent(np.arange(0, 3600, 1), 'bateria')
consumo = ctrl.Antecedent(np.arange(0, 2000, 1), 'consumo')
costo = ctrl.Antecedent(np.arange(0, 9, 1), 'costo')
batGen = ctrl.Antecedent(np.arange(0, 5100, 1), 'batGen')
batGenCon = ctrl.Antecedent(np.arange(0, 3600, 1), 'batGenCon')

#salida
salida = ctrl.Consequent(np.arange(0, 100, 1), 'salida')

bateria["Baja"] = fuzz.gaussmf(bateria.universe, 0, 1000)
bateria["Media"] = fuzz.dsigmf(bateria.universe, 1140, 0.00518, 2600, 0.00471)
bateria["Alta"] = fuzz.gaussmf(bateria.universe, 3860, 865)

consumo["Baja"] = fuzz.gaussmf(consumo.universe, 0, 500)
consumo["Media"] = fuzz.dsigmf(consumo.universe, 600, 0.0149, 1400, 0.01202)
consumo["Alta"]= fuzz.gaussmf(consumo.universe, 2000, 442.4)

costo["Bajo"] = fuzz.trapmf(costo.universe,  [-3.24, -0.36, 2.404, 2.68])
costo["Medio"] = fuzz.trapmf(costo.universe, [2.38, 2.608, 5.22, 5.5])
costo["Alto"] = fuzz.trapmf(costo.universe, [5.27, 5.35, 9.08, 11.9])

batGen["Baja"] = fuzz.gaussmf(batGen.universe, 0, 1000)
batGen["Media"] = fuzz.dsigmf(batGen.universe, 1600, 0.00518, 2670, 0.00471)
batGen["Alta"] = fuzz.dsigmf(batGen.universe, 3000,0.00319, 15000,0.00148)

batGenCon["Baja"] = fuzz.trapmf(batGenCon.universe,[-13400,-5470, 600, 650])
```

```
batGenCon["Alta"] = fuzz.trapmf(batGenCon.universe, [2100, 3600, 10260,

salida['Baja'] = fuzz.gaussmf(salida.universe,0, 20.72  )
salida['Media'] = fuzz.dsigmf(salida.universe,  39.81, 0.188, 71.8, 0.28
salida['Alta'] = fuzz.gaussmf(salida.universe,  100, 22.13)
salida['OFF'] = fuzz.trapmf(salida.universe, [-90, -10, 12, 15])
salida['ON'] = fuzz.trapmf(salida.universe, [70.2, 80.12, 110, 190])

#Reglas
rule1 = ctrl.Rule(bateria["Alta"] & costo["Alto"], salida['Alta'])
rule2 = ctrl.Rule(bateria["Media"] & costo["Medio"], salida['Media'])
rule3 = ctrl.Rule(bateria["Baja"] & costo["Bajo"], salida['Baja'])
rule4 = ctrl.Rule(~consumo["Baja"] & costo["Alto"] & ~batGenCon["Baja"]
rule5 = ctrl.Rule(consumo["Baja"] & costo["Alto"]& ~batGenCon["Baja"],
rule6 = ctrl.Rule(costo["Alto"] & batGen["Alta"], salida['Alta'])
rule7 = ctrl.Rule(costo["Medio"] & batGen["Media"], salida['Media'])
rule8 = ctrl.Rule(costo["Bajo"] & batGen["Baja"], salida['Baja'])
rule9 = ctrl.Rule(batGenCon["Baja"],salida['OFF'])
rule10 = ctrl.Rule(batGenCon["Alta"],salida['ON'])
rule11 = ctrl.Rule(consumo["Baja"] & costo["Bajo"],salida['Baja'])

#Controlador
tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, ru

def heducFuzzy(bateria, consumo, batGen, batGenCon, costo):
    tipping = ctrl.ControlSystemSimulation(tipping_ctrl)
    tipping.input['bateria'] = bateria
    tipping.input['consumo'] = consumo
    tipping.input['batGen'] = batGen
    tipping.input['batGenCon'] = batGenCon
    tipping.input['costo'] = costo

    tipping.compute()

    return tipping.output['salida']
```

A.11. moduloConsumoCosto.py

```
consumo = [.....]
consumo = consumo*2

def vectorConsumoH(tiempo):
    return consumo[tiempo:tiempo + 8]
```

```
costo15 = [1.7 for i in range(28)] + [3.739 for i in range(44)] + [8.507 for i i
costo = costo15
costo = costo*2

def vectorCostoH(tiempo):
    return costo[tiempo: tiempo + 8]
```

A.12. Sockets

A.12.1. Socket.py

```
import threading
import json , sys, time
import socket #For access to socket communication
import sys #To terminate with error code

def initialize_server(host, port):
    print "Starting server on %s:%s" %(host,port)
    srv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    #Set the socket reuse to 1, so that socket terminates quicker
    srv.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    try:
        srv.bind((host, port))
    except socket.error , msg:
        print "Bind failed. Error: " + str(msg[0]) + "\nMessage: " + msg[1]
        sys.exit(-1)
    srv.listen(0) #Currently only one client at the time
    return srv

def wait_for_client(server_socket):
    (client, c_address) = server_socket.accept() #blocking wait for a client
    print "Received a connection from [%s]" %c_address[0]
    return client


def server_run():
    print "Starting heduc server"

    srv = server_socket.initialize_server("", 7777)

    while True:
        print "Waiting for someone to chat with..."
        # This will be a blocking wait
        client = server_socket.wait_for_client(srv)
```

```
        if handle_client_request_for_esp(client) == MSG_OK:
            msg = "OK"
            client.sendall(msg)
        else:
            try:
                msg = "ERR"
                client.sendall(msg)
            except:
                print "Client no respond"
        print "Close comunication"
        client.close()


# Protocolo de comunicacion para HEDUC
# Se definen las palabras para la comunicacion entre las esp8266 y la B

# Codigo de seguridad (primera palabra que se lee) (4 bytes)

SECURITY_CODE = 1234

# Mensaje error
MSG_OK = 1
MSG_ERROR = 0

def handle_client_request_for_esp(cs):
    try:
        msg = int(string_div(cs))
        if msg == SECURITY_CODE:
            print "Security code is ok"
            type_msg = string_div(cs)
            if type_msg == "PUT":
                var_msg = string_div(cs)
                print var_msg
                value = int(cs.recv(3))
                print value
                new_variable_write( var_msg , value )
            else:
                print "Incorrect command "
        return MSG_OK
    except:
        print "Interrupted communication"
        return MSG_ERROR

def string_div(css):
    msgg = css.recv(1)
```

```
    msgRead = ""
    while msgg != "/" :
        msgRead = msgRead + msgg
        msgg = css.recv(1)
    return msgRead

data = { 'Temperature' : [] , 'Humidity' : [] }

def newVariableToWrite( typeVariable , variableValue ):
    try:
        f = open( 'datos' , 'r' )
        data = json.load( f )
        f.close

        date = time.strftime("%d/%m/%y %H:%M")
        data[typeVariable].append( [date , variableValue] )

        f = open( 'datos' , 'w' )
        json.dump( data , f )
        f.close

    except:
        print "No se pudo guardar el dato"

def variableToRead(typeVariable):
    f = open( 'datos' , 'r' )
    data = json.load( f )
    f.close

    return data[typeVariable]

server_run()
```

A.12.2. socket.h

```
// Debug mode
#ifndef DEBUG_MODE
        #define DEBUG_MODE 1
#endif

#define BAUDIOS 115200

void init_socket();

int clientToAttend();
```

```
String msgRecive();

int msgToTransmit(String msg);
```

A.12.3. socket.cpp

```
#include <ESP8266WiFi.h>
#include "heduc_socket.h"
const int Port = 7777;
const char* host = "192.168.2.3";

WiFiServer server(Port);

void init_socket(){
        server.begin();
        if(DEBUG_MODE){
                Serial.begin(BAUDIOS);
                Serial.println("Server started");
        }
}

String newMsg;

int clientToAttend(){
        int ret;
        WiFiClient clientS = server.available();
        if(clientS){
                if(DEBUG_MODE){
                        Serial.println("new client");
                }
                while(!clientS.available()){
                        delay(1);
                }
                newMsg = clientS.readStringUntil('\r');
                if(DEBUG_MODE){
                        Serial.println(newMsg);
                }
                clientS.print("OK");
                clientS.stop();
                ret = 1;
        }
        else{
                ret = 0;
        }
```

```
        return ret;
}

String msgRecive(){
        return newMsg;
}

int msgToTransmit(String msg){
        int ret;
        WiFiClient clientC;
        if(clientC.connect(host,Port)){
                Serial.println("Connect to server");
                clientC.print(msg);
                delay(100);
                if(clientC.available()){
                        msg = clientC.readStringUntil('\r');
                        if(DEBUG_MODE){
                                Serial.println(msg);
                        }
                        ret = 0;
                }
                else{
                        if(DEBUG_MODE){
                                Serial.println("Connection lose");
                        }
                        ret = 1;
                }
        }
        else{
                if(DEBUG_MODE){
                        Serial.println("Connection failed");
                }
                ret = 2;
        }
        return ret;
}
```

Referencias

[1] L. Yao, C.-C. Lai, and W. H. Lim, "Home energy management system based on photovoltaic system," in *Data Science and Data Intensive Systems (DSDIS), 2015 IEEE International Conference on*, pp. 644–650, IEEE, 2015.

[2] D. Arcos-Aviles, J. Pascual, L. Marroyo, P. Sanchis, and F. Guinjoan, "Fuzzy logic-based energy management system design for residential grid-connected microgrids," *IEEE Transactions on Smart Grid*, 2016.

[3] R. Al Badwawi, W. Issa, T. Mallick, and M. Abusara, "Dc microgrid power coordination based on fuzzy logic control," in *Power Electronics and Applications (EPE'16 ECCE Europe), 2016 18th European Conference on*, pp. 1–10, IEEE, 2016.

[4] F. Bonanno and G. Patane, "Energy management optimization of integrated generation systems by fuzzy logic control," in *Control Applications, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 2, pp. 969–974, IEEE, 1998.

[5] D. Arcos-Aviles, J. Pascual, L. Marroyo, P. Sanchis, F. Guinjoan, and M. P. Marietta, "Optimal fuzzy logic ems design for residential grid-connected microgrid with hybrid renewable generation and storage," in *Industrial Electronics (ISIE), 2015 IEEE 24th International Symposium on*, pp. 742–747, IEEE, 2015.

[6] A. Chugh, P. Chaudhary, and M. Rizwan, "Fuzzy logic approach for short term solar energy forecasting," in *India Conference (INDICON), 2015 Annual IEEE*, pp. 1–6, IEEE, 2015.

[7] D. Arcos-Aviles, F. Guinjoan, M. P. Marietta, J. Pascual, L. Marroyo, and P. Sanchis, "Energy management strategy for a grid-tied residential microgrid based on fuzzy logic and power forecasting," in *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*, pp. 4103–4108, IEEE, 2016.

[8] M. Čekon, R. Slávik, and P. Juráš, "Obtainable method of measuring the solar radiant flux based on silicone photodiode element.," *Applied Mechanics & Materials*, vol. 824, 2016.

[9] I. E. Commission *et al.*, "Iec 891," *Procedures for temperature and irradiance corrections to measured IV characteristics of crystalline silicon photovoltaic devices*, 1987.

[10] E. I. Ortiz-Rivera and F. Z. Peng, "Analytical model for a photovoltaic module using the electrical characteristics provided by the manufacturer data sheet," in *Power Electronics Specialists Conference, 2005. PESC'05. IEEE 36th*, pp. 2087–2091, IEEE, 2005.

Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

Esta es la última página.
Compilado el martes 18 julio, 2017.
http://iie.fing.edu.uy/