



UNIVERSIDAD DE BUENOS AIRES

Facultad de Ingeniería

86.65 Sistemas Embebidos

Memoria del Trabajo Final:

## **E-Save Bluetooth. Ahorro Energético de Climatización.**

**Autor:**

**Sr. Agustín Mariano de Vedia**

Legajo: 102.297

*Este trabajo fue realizado en las Ciudad Autónoma de Buenos Aires,  
entre marzo y junio de 2023.*

## **RESUMEN**

El presente trabajo consiste en un producto que controla el consumo de energía de un aparato eléctrico en base a si hay alguien o no en el ambiente. El *E-Save Bluetooth* posee un sensor de movimiento y si no se detecta movimiento en un intervalo de tiempo, desconecta el equipo de consumo hasta que se vuelva a detectar presencia en el ambiente. El sistema se configura vía Bluetooth Low Energy.

El producto final resulta de gran utilidad frente a otras opciones más limitadas ofrecidas en el mercado y aún así es de bajo costo. Su desarrollo demostró ser una aplicación completa de lo aprendido durante el cuatrimestre en la materia Sistemas Embebidos. Se utilizaron conceptos de máquinas de estado, control de interrupciones, programación y depuración de la placa NUCLEO, reloj interno de tiempo real, utilización de un relay, comunicación serie y Bluetooth.

De esta Memoria, el lector se llevará un entendimiento extenso del proceso de desarrollo de este trabajo así como también comprenderá las decisiones de diseño consideradas y el funcionamiento de las máquinas de estado involucradas.

## **ABSTRACT**

E-Save Bluetooth is a low cost device which turns an electric appliance off if it does not detect presence in its environment for a certain period of time. It can be configured via Bluetooth Low Energy. This product stands out from what the market currently offers due to its low cost and diverse functionality.

The reader will, upon reading this Memoir, have a complete understanding of the project's development process, both implementation and testing stages. It serves as a thorough recap of the Embedded Systems course, having used concepts from state machines, interrupt handling, programming and debugging the NUCLEO board, real time clock, relays, serial communication and Bluetooth.

## **Agradecimientos**

A los profesores de la asignatura por su compromiso y enfoque didáctico de la materia. A Cata por su apoyo constante en mis ideas y no molestarte si saltó la térmica un par de veces.

# Índice General

<b>Registro de versiones</b>	<b>2</b>
<b>Introducción general</b>	<b>8</b>
1.1 Origen del proyecto	8
1.2 Estado del arte	9
1.3 Tiempo de Desarrollo	11
<b>Introducción específica</b>	<b>12</b>
2.1 Requisitos	12
2.2 Casos de uso	14
2.3 Descripción de los módulos	15
2.3.1 Relay	15
2.3.2 Sensor de movimiento	16
2.3.3 Módulo Bluetooth	17
<b>Diseño e implementación</b>	<b>18</b>
3.1 Hardware del E-Save Bluetooth	18
3.1.1 Módulos	19
3.1.2 Sistema completo	24
3.1.3 Listado de componentes	27
3.2 Firmware del E-Save Bluetooth	28
3.2.1 Lógica principal	30
3.2.2 Comunicación BLE	32
3.2.3 Actuador	33
3.2.4 Relay	35
3.2.5 Sensor de movimiento	36
3.2.6 RTC	38
3.2.7 Repositorio	40

<b>Ensayos y resultados</b>	<b>41</b>
4.1 Pruebas funcionales del hardware	41
4.2 Pruebas funcionales del firmware	42
4.2.1 Comunicación serie	42
4.2.2 Actuador	44
4.3 Pruebas de integración	44
4.4 Cumplimiento de requisitos	44
4.5 Comparación con otros sistemas similares	47
4.6 Documentación del desarrollo realizado	48
<b>Conclusiones</b>	<b>49</b>
5.1 Resultados obtenidos	49
5.2 Próximos pasos	49
<b>Bibliografía</b>	<b>51</b>

## **Registro de versiones**

<b>Revisión</b>	<b>Cambios realizados</b>	<b>Fecha</b>
1.0	Creación del documento	27/06/2023
1.1	Redacción del informe	29/06/2023
1.2	Edición	03/07/2023
1.3	Finalización del documento	05/07/2023

# CAPÍTULO 1

## Introducción general

### 1.1 Origen del proyecto

La idea original del proyecto nace de la necesidad de regular el consumo de energía en el hogar, tanto por temas económicos como de seguridad. Dejar un equipo de alto consumo encendido puede traer consecuencias graves en un descuido. Sumado a eso, regular la energía consumida resulta de alta importancia en estos tiempos. Influenciado por querer disminuir el gasto en la boleta de luz y por querer tener un menor impacto en el medio ambiente, se diseñó el E-Save Bluetooth.

La lógica principal del dispositivo consiste en desconectar un aparato de consumo, como un aire acondicionado o una estufa, si no se detecta presencia en el ambiente luego de un determinado intervalo de tiempo. El objetivo es dejar de gastar energía en climatizar un ambiente vacío. Desde un principio se ideó que el aparato sea configurable vía Bluetooth utilizando la aplicación *Serial Bluetooth Terminal*<sup>1</sup>, para dar comodidad al usuario. Por sobre esta funcionalidad básica, se puede fijar un horario de funcionamiento activo, fuera del cual el equipo de consumo permanece conectado sin importar si hay gente o no en el ambiente. En la figura 1.1 se puede observar un diagrama general del sistema. En el mismo se muestra un aire acondicionado como aparato de consumo, conectado al dispositivo el cual a su vez se conecta a la red de alimentación y a un sensor para detectar presencia en el ambiente. El estado del sistema se indica por medio de los seis LEDs que se observan en el diagrama. El detalle de la conexión y estados posibles se puede encontrar en el Capítulo 3: Diseño e implementación.

---

<sup>1</sup> *Serial Bluetooth Terminal* se encuentra disponible en la App Store de Android y en Apple Store.



**Figura 1.1:** Diagrama general del sistema.

El trabajo se realizó en el ámbito de la asignatura Sistemas Embebidos, dada en la Facultad de Ingeniería de la Universidad de Buenos Aires. En la misma se analizan conceptos tales como máquinas de estado, control de interrupciones, reloj interno de tiempo real, utilización de un relay, comunicación serie y Bluetooth. Todo esto aplicado en una placa NUCLEO-F429ZI. Tanto los conocimientos teóricos como la práctica constante de desarrollo y depuración fueron elementos claves en la preparación del proyecto.

## 1.2 Estado del arte

En la etapa inicial de desarrollo se analizó el mercado a fin de entender los productos ofrecidos que cumplen funciones similares. Al observar los dos principales dispositivos de control de consumo eléctrico en el hogar, *Enchufe Inteligente* y *Termostato Inteligente*, el E-Save Bluetooth se destaca principalmente por su bajo costo y por ubicarse con una funcionalidad intermedia entre ambos. Se muestra una comparación de los productos a continuación, en la tabla 1.1.

**Tabla 1.1:** Comparación de los productos ofrecidos por el mercado y el E-Save Bluetooth.

Característica	Enchufe inteligente. <a href="#">Link</a>	Termostato Inteligente <a href="#">Link</a>	E-Save Bluetooth
<b>Conectividad a internet / Wi-Fi</b>	Sí	Sí	No
<b>Conectividad Bluetooth</b>	No	No	Sí
<b>Control manual</b>	Sí	Sí	Sí
<b>Lógica de control</b>	Encendido/Apagado accionado por el usuario vía Wi-Fi	Encendido/Apagado accionado por el usuario vía aplicación móvil y Wi-Fi. También permite configurar timers para encender y apagar.	Encendido/Apagado accionado por el usuario o automáticamente cuando no se detecta presencia por cierto tiempo.
<b>Sensor de movimiento</b>	No	No	Sí
<b>Configuración</b>	Encendido/Apagado accionado por el usuario. No posee configuración de timer.	Vía Wi-Fi.	Vía Bluetooth.
<b>Precio estimado (USD)</b>	\$12	\$360	\$40

El enchufe inteligente permite encender y apagar el equipo de consumo en tiempo real utilizando Wi-Fi. Tiene una funcionalidad mínima y un costo bajo. Frente a esta opción, el producto desarrollado en el presente trabajo provee mayor utilidad a un pequeño incremento de costo. Los *termostatos inteligentes*, a pesar de su multifuncionalidad, no poseen sensores que permitan detectar la presencia de una persona en el ambiente. Responde a requerimientos que no coinciden con los del *E-Save Bluetooth*. Se podrían usar en simultáneo para una completa funcionalidad. El dispositivo diseñado no se ve afectado por las eventualidades de Internet, ya que se configura vía Bluetooth, lo cual es una clara ventaja frente a su competencia.

En tema costo, se percibe una diferencia considerable entre el precio de un *enchufe inteligente* (USD 12) y de un *termostato inteligente* (USD 360). Dentro de este espectro, el *E-Save Bluetooth* se ubica entre ambos con un costo aproximado de USD 40.

El dispositivo desarrollado trae consigo una funcionalidad novedosa que es la de desconectar el consumo de energía si no se detecta gente en los alrededores. Esto se diferencia del accionar de un termostato inteligente que funciona únicamente con timers.

### **1.3 Tiempo de desarrollo**

El tiempo de desarrollo del trabajo fue de aproximadamente 20 horas de trabajo dispersas en 4 semanas. Al planear el proyecto, uno de los objetivos fue que la implementación no consumiera más de 40 horas, valuadas a 6 USD/h. Frente a esta expectativa sobreestimada, se calcula un ahorro de 120 USD.

## CAPÍTULO 2

### Introducción específica

#### 2.1 Requisitos

En la tabla 2.1 a continuación, se listan los requerimientos definidos para el trabajo. Se tuvieron en cuenta los criterios SMART<sup>2</sup> al momento de definirlos. Los requerimientos se agrupan en grupos para una mejor organización.

**Tabla 2.1:** Requerimientos técnicos.

Grupo	ID	Descripción
1. Sensor	1.1	El sistema podrá detectar la presencia de una persona en el ambiente.
	1.2	El sistema podrá detectar cuando no haya nadie presente en el ambiente.
	1.3	El sistema podrá llevar cuenta de cuánto tiempo continuo no detecta presencia en el ambiente.
	1.4	El sistema indicará mediante un LED amarillo si hay alguien en el ambiente.
2. Interfaz	2.1	El sistema tendrá un pulsador que al ser presionado desconecta el equipo de consumo.
	2.2	El sistema tendrá un pulsador que al ser presionado conecta el equipo de consumo.
	2.3	El sistema permite al usuario conectarse vía Bluetooth mediante una aplicación al mismo.
	2.4	El sistema permite enviar mensajes de ayuda al usuario vía Bluetooth mediante la aplicación utilizada.
	2.5	El sistema permite recibir mensajes (instrucciones) vía Bluetooth mediante la aplicación utilizada.
	2.6	El sistema permite configurar el timer de acción en función de la

<sup>2</sup> Criterios SMART: specific, measurable, achievable, relevant, time-bound.

		información recibida vía Bluetooth.
	2.7	El sistema indicará mediante un LED verde que el equipo de consumo se encuentra en conectado a su alimentación
	2.8	El sistema indicará mediante un LED rojo que el equipo de consumo se encuentra desconectado de su alimentación
	2.9	El sistema indicará mediante un LED azul si hay una conexión Bluetooth activa.
3. Reloj	3.1	El sistema tendrá un reloj interno de tiempo real.
	3.2	Mediante la comunicación Bluetooth se podrá configurar el horario del reloj interno del dispositivo.
	3.3	El usuario podrá definir vía Bluetooth intervalos de horarios para los cuales el dispositivo está en funcionamiento activo (sensando y accionando).
	3.4	Dentro del rango horario definido para el funcionamiento activo, el microcontrolador encenderá un LED rojo a fin de indicar el estado activo.
	3.5	Fuera del rango horario definido para el funcionamiento activo, el microcontrolador no desconectará el relay sin importar el estado del sensor.
4. Control	4.1	El sistema accionará un relay para conectar/desconectar la alimentación del aparato de consumo.
	4.2	El sistema desconectará la alimentación del aparato de consumo si no detecta presencia en el ambiente luego de X tiempo.
5. Alimentación	5.1	El sistema se alimentará mediante el uso de un transformador 220 V a 5 V.
	5.2	La placa NUCLEO F429ZI se alimentará con 5 V
6. Costo	6.1	Implementar el proyecto tendrá un costo de materiales menor a 50 USD.
	6.2	Implementar el proyecto tendrá un costo de horas de trabajo menor a 240 USD, a un valor de 6 USD/hora.
7. Tiempo	7.1	El proyecto será finalizado antes del 28 de junio de 2023.
8. Documentación	8.1	Junto al prototipo, se proveerá un listado de partes, el repositorio de código documentado y un manual de uso.

## 2.2 Casos de uso

Antes de comenzar a idear los detalles técnicos, desarrollar el hardware y software, se definieron casos de uso. Los mismos se pueden observar en las tablas 2.2, 2.3 y 2.4. El objetivo fue determinar las funcionalidades básicas y cómo interactuaría el usuario en las distintas situaciones.

**Tabla 2.2:** Caso de uso 1: Equipo se desconecta a causa de agotarse el tiempo de espera.

Elemento	Definición
Disparador	El sistema registra que no hubo nadie en el ambiente por 20 minutos.
Precondición	El sistema debe estar encendido y correctamente instalado. Se debe haber configurado previamente 20 minutos como el tiempo de espera.
Flujo básico	El sistema registra el disparador. Se ejerce la acción de control sobre el Relay. Se desconecta el equipo de consumo. Se enciende el LED rojo y se apaga el LED verde.
Flujos alternativos	1. a. El relay no está correctamente instalado. La acción de control no tiene efecto pero los LEDs cambian igual: se enciende el LED rojo y se apaga el LED verde. 1. b. El relay se encuentra ya apagado porque el usuario lo había apagado manualmente y no ha vuelto a encenderlo. No se producen cambios en los LEDs.

**Tabla 2.3:** Caso de uso 2: Equipo se desconecta por acción manual del usuario.

Elemento	Definición
Disparador	El usuario presiona el pulsador para desconectar el equipo.
Precondición	El sistema debe estar encendido y correctamente instalado. El relay debe estar encendido y el equipo eléctrico de consumo conectado a su alimentación.
Flujo básico	El sistema registra la acción de pulsado. Apaga el relay y se desconecta el equipo eléctrico de su alimentación. Se enciende el LED rojo y se apaga el LED verde.
Flujos alternativos	1. a. El relay no está correctamente instalado. La acción de control no tiene efecto pero los LEDs cambian igual: se enciende el LED rojo y se

	apaga el LED verde. 1. b. El relay estaba ya apagado, el equipo no estaba conectado a la alimentación. No se producen cambios ni en el relay ni en los LED.
--	--

**Tabla 2.4:** Caso de uso 3: Se envía un comando de configuración al dispositivo vía Bluetooth.

<b>Elemento</b>	<b>Definición</b>
Disparador	El usuario envía un comando a través de la aplicación para configurar el tiempo de espera a 15 minutos.
Precondición	El sistema debe estar encendido y con conexión Bluetooth activa con el usuario.
Flujo básico	El sistema recibe el comando vía Bluetooth. Se procesa internamente. Se ajusta el valor del timer a 15 minutos.
Flujos alternativos	1. a. El sistema no está encendido o no está correctamente vinculado vía Bluetooth con el usuario. No se recibe el mensaje. Se ignora.

## 2.3 Descripción de los módulos

En esta sección se presentará una introducción a los módulos utilizados en el *E-Save Bluetooth*. Se explica lo necesario para poder comprender su funcionamiento y cómo interactúan con la placa NUCLEO-F429ZI.

### 2.3.1 Relay

Para el desarrollo del dispositivo se hace uso de un módulo relay que se controla con alimentación de 5 V y una entrada digital que permite alternar entre dos conexiones a la salida. El módulo incluye el relay, pines y bornera para las conexiones y un diodo de protección. Se puede observar el módulo completo en la figura 2.1.



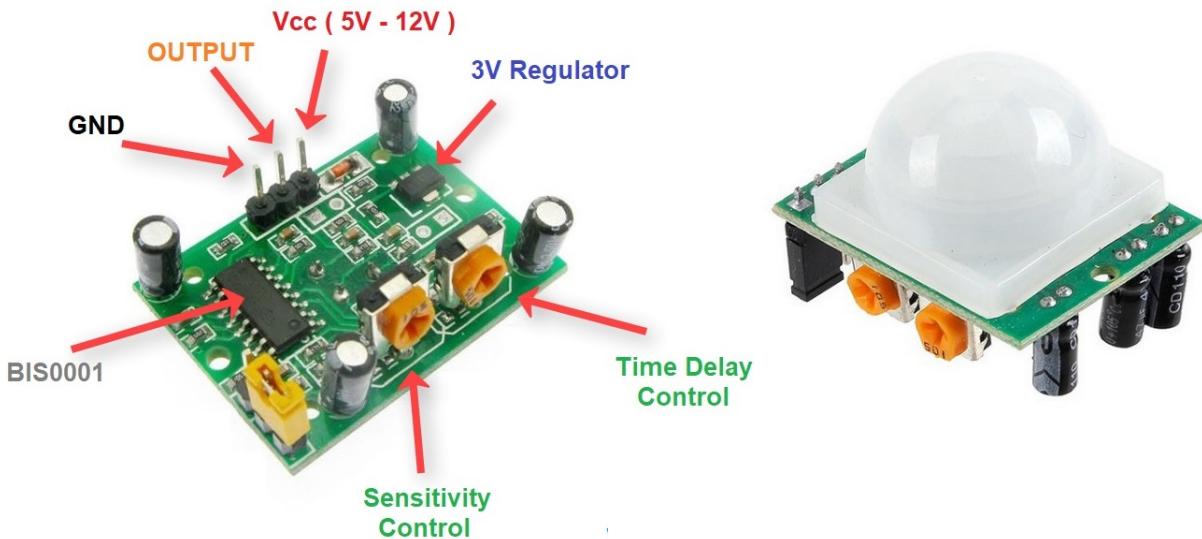
**Figura 2.1:** Módulo relay.

Es importante tener en cuenta que el relay puede demandar un alto consumo y es recomendable alimentarlo directamente de los 5 V del transformador. Más allá del consumo, el switching de las bobinas pueden introducir un ruido muy apreciable al sistema. Es por esto que durante la implementación del prototipo se determinó necesario conectar un buffer entre el pin de salida de la placa y el pin de control relay. El buffer se armó con un BC547 polarizado con una resistencia de 10 kOhm y alimentado a 5 V de igual modo que el relay. Más información al respecto se provee en la sección 3.1 del capítulo 3.

La etapa de salida del módulo tiene 3 bornes: “común” en el centro y “normalmente abierto” y “normalmente cerrado” a los costados. En función de la entrada digital en la etapa de entrada, el común se conectará con alguno de los otros dos bornes.

### 2.3.2 Sensor de movimiento

El sensor PIR de movimiento utilizado es el HC-SR501. El mismo consta de un sensor que detecta cambios de calor utilizando luz infrarroja. El módulo tiene 3 pines, la alimentación de 5-12 V y un pin digital de salida con el cual se envía un ‘1’ lógico si se detecta movimiento. Se provee en la figura 2.2 una imagen del módulo completo con referencias.

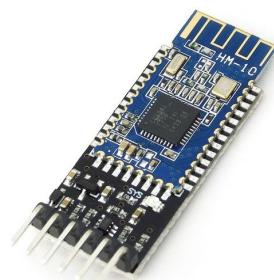


**Figura 2.2:** Módulo sensor de movimiento.

El módulo cuenta con dos potenciómetros con los que se regula la sensibilidad y el retraso temporal entre la detección de movimiento y el envío de la señal digital. Es muy importante su correcta calibración.

### 2.3.3 Módulo Bluetooth

Para lograr que el dispositivo tenga conectividad Bluetooth se hace uso del módulo HM-10. Esta placa está basada en el chip CC2541. Se comunica con la placa NUCLEO-F429ZI vía puerto serie y luego, transparente al NUCLEO-F429ZI, se reenvía la comunicación utilizando el protocolo Bluetooth o Bluetooth Low Energy. Se debe aclarar que en este trabajo se utiliza BLE (*Bluetooth Low Energy*), ya que al no requerir transmitir constantemente grandes cantidades de información resulta más eficiente trabajar con BLE. Una imagen del módulo utilizado se puede apreciar en la figura 2.3.



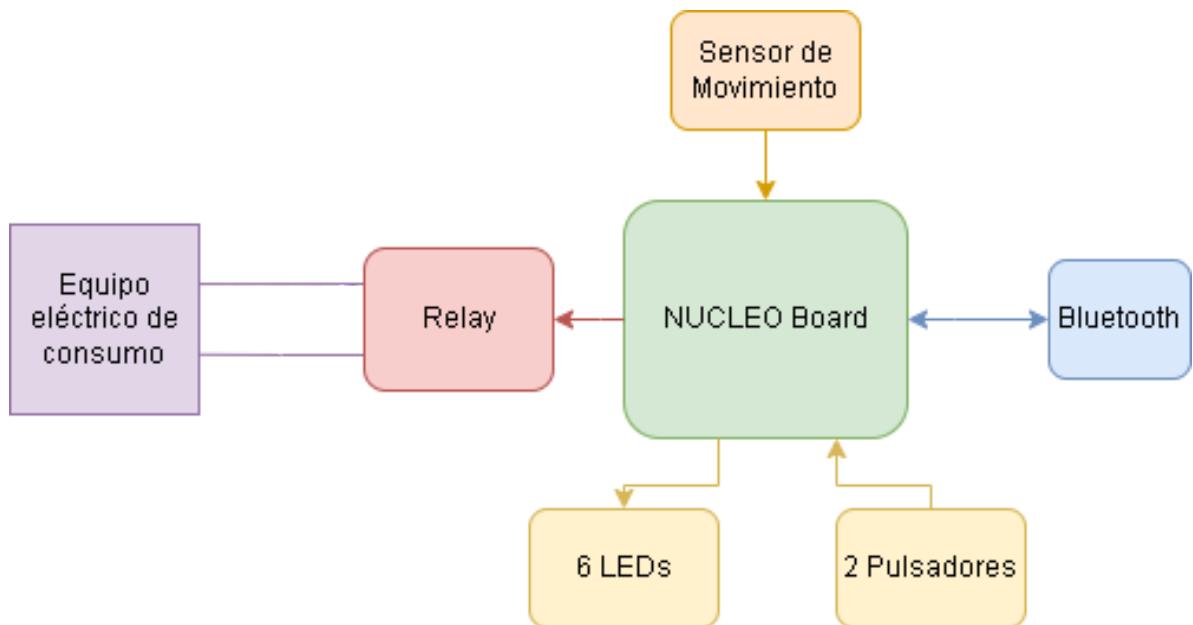
**Figura 2.3:** Módulo sensor de movimiento.

## CAPÍTULO 3

### Diseño e implementación

#### 3.1 Hardware del E-Save Bluetooth

Luego de determinados los requerimientos y casos de uso se diseñó el hardware del dispositivo. En la figura 3.1 se puede observar el diagrama de bloques del trabajo.



**Figura 3.1:** Diagrama en bloques del trabajo.

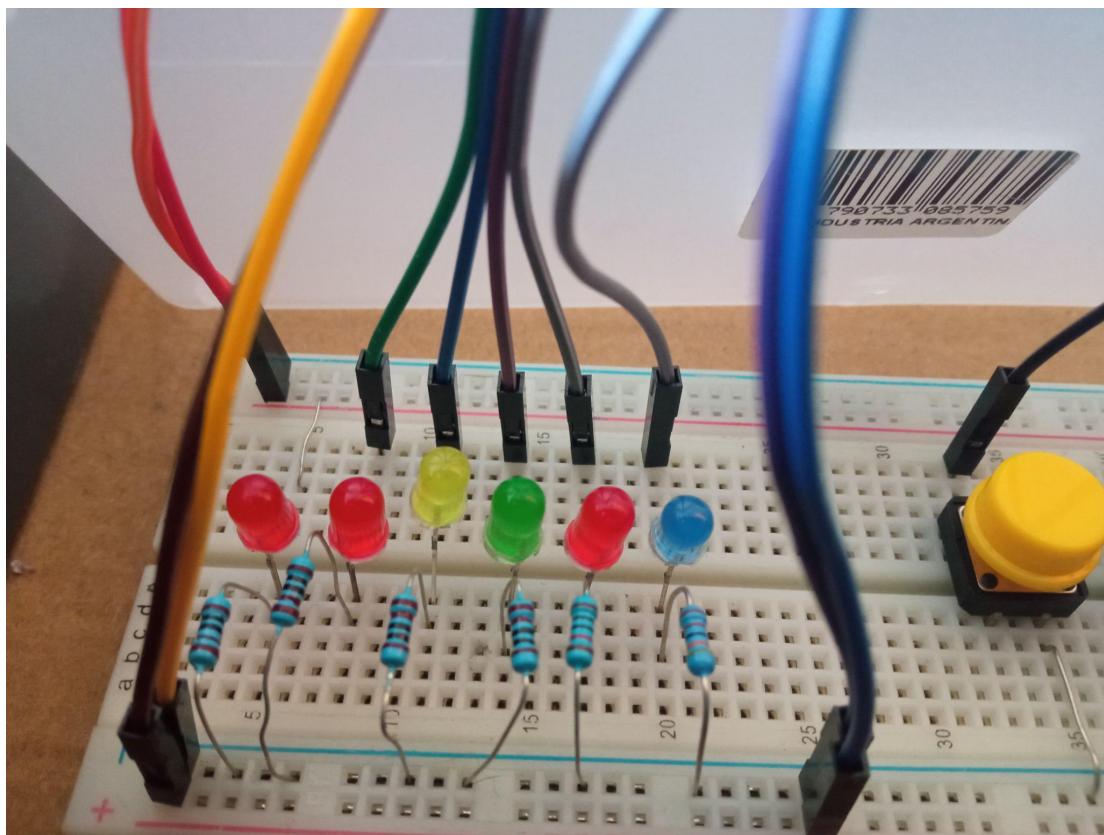
En términos generales, el trabajo consta de una placa NUCLEO-F429ZI conectada con un sensor de movimiento y dos pulsadores como entradas de control y un relay conectado al equipo eléctrico de consumo y 6 LEDs como salidas para accionar y mostrar su estado. En paralelo a esto, el módulo para conexión Bluetooth se comunica vía puerto serie con la placa.

Se decidió utilizar la placa NUCLEO-F429ZI para la implementación del trabajo a fines didácticos y para aprovechar la experiencia de programar y depurar una placa de tal calibre. Otra opción viable era la placa NodeMCU con el microprocesador ESP32, el cual ya viene integrado con conectividad Wi-Fi y BLE. Si se desea desarrollar en masa el producto se recomienda aprovechar el bajo costo y tamaño de una placa con un ESP32. El trabajo es

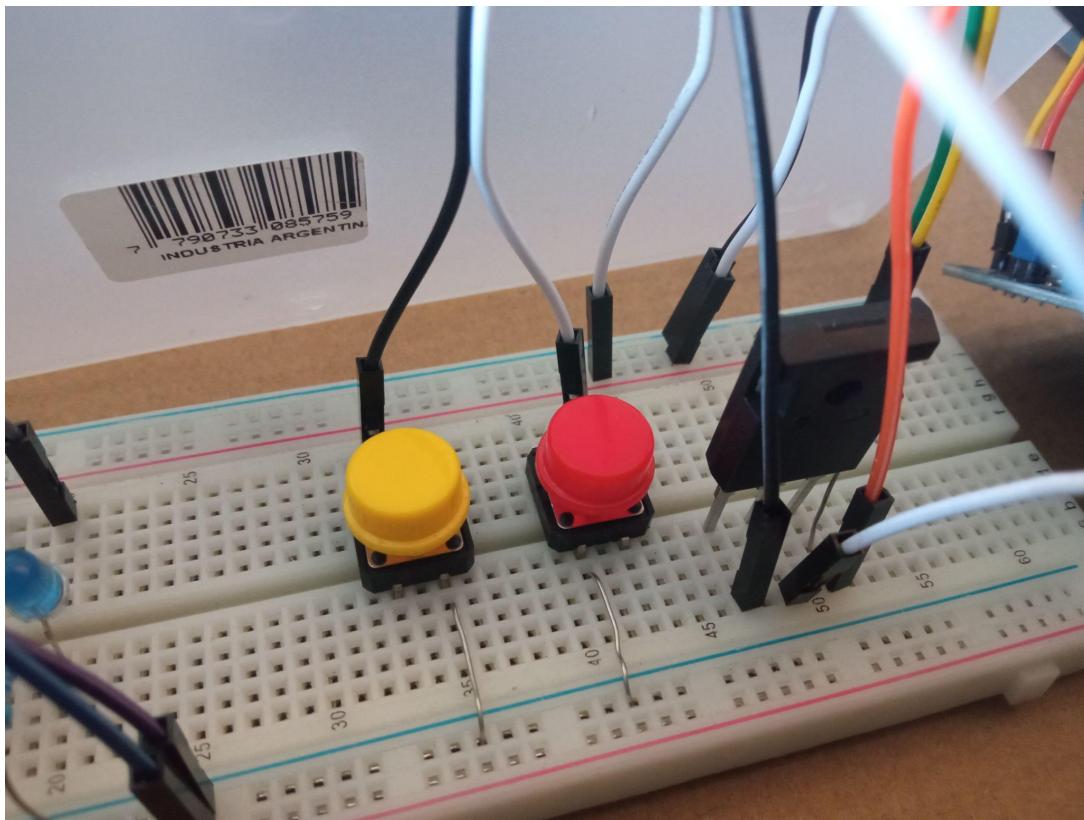
fácilmente escalable para funcionar con ese microcontrolador, sería necesario actualizar el código base pero los módulos periféricos son compatibles.

### 3.1.1 Módulos

Los pulsadores y LEDs utilizados son estándar. Los LEDs se alimentan con 3,3 V y se polarizan con resistores de 150 Ohms. Para los pulsadores se aprovechan las resistencias pull up internas del microcontrolador. Imágenes de los LEDs y pulsadores se pueden observar en las figuras 3.2 y 3.3.



**Figura 3.2:** 6 LEDs utilizados para indicar el estado del sistema.



**Figura 3.3:** Pulsadores utilizados en el trabajo.

El pulsador rojo desconecta el aparato de consumo y bloquea el sistema. Para desbloquearlo se pulsa el botón amarillo. Si ya se encuentra bloqueado, presionar nuevamente el botón rojo no tiene ningún efecto apreciable, lo mismo si el sistema no se encuentra bloqueado y se presiona el botón amarillo.

El módulo preferido para implementar la conectividad BLE es el HM-10, como se mencionó anteriormente. Hay una variedad de modelos similares con costos equiparables. El elegido es de fácil adquisición. El módulo presenta 5 pines. La figura 3.4 indica visualmente la interfaz, luego se presenta una lista con el detalle de los pines.



**Figura 3.4:** Interfaz del módulo HM-10.

- State: Indica si hay una conexión Bluetooth activa.
- RXD: Receptor del módulo.
- TXD: Transmisor del módulo.
- GND: Referencia.
- VCC: Para su alimentación.
- EN: Utilizado para habilitar/deshabilitar la conexión.

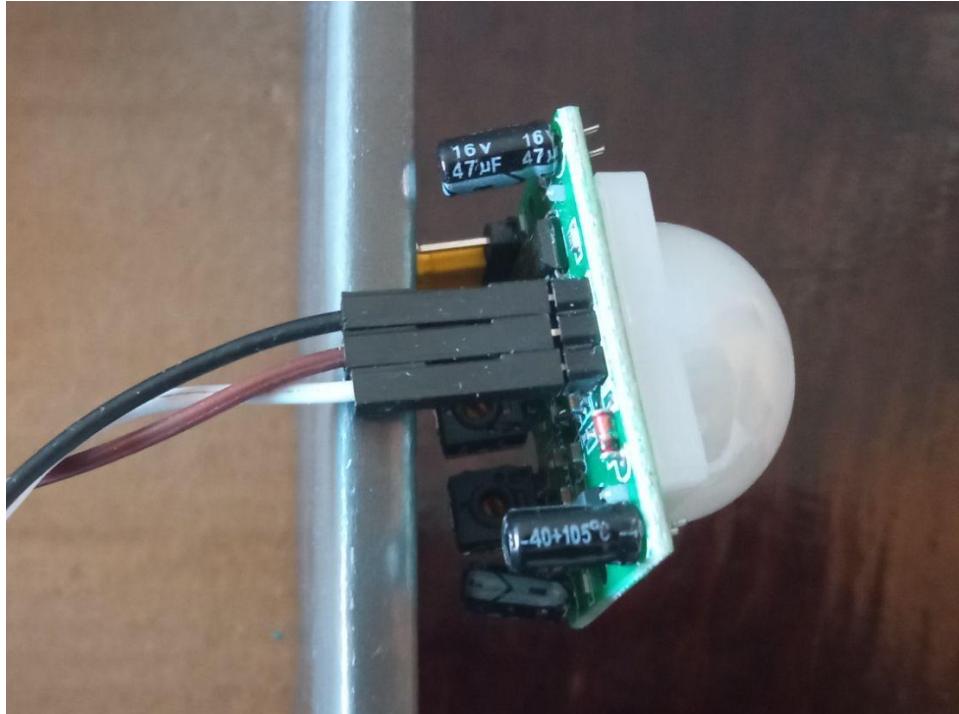
Los pines RXD, TXD, VCC y GND se conectarán a la placa. El pin state se aprovechará para alimentar con 3,3 V el LED azul que indica si hay una conexión Bluetooth activa. El pin enable no se utiliza en este trabajo.

Para la elección del sensor de movimiento se presentan múltiples opciones. En este punto el mercado ofrece productos de diversas características y precios. Para este trabajo, donde se construyó un prototipo, se decidió por el sensor infrarrojo PIR HC-SR501 por su alta disponibilidad y bajo costo. Es compatible con la placa, se alimenta con 5 V y es simple de calibrar gracias a sus dos potenciómetros fácilmente accesibles. Sin embargo, se debe tener en cuenta que hay diversas opciones y para una aplicación específica puede ser mejor elegir uno u otro. Se pueden analizar tres variedades principales en la tabla 3.1.

**Tabla 3.1:** Principales opciones de sensores de movimiento en el mercado.

Producto	Tecnología	Rango	Interfaz	Precio [ USD ]
Hikvision Inalámbrico	PIR	15 m / 85,9°	Inalámbrica vía aplicación.	85
HC-SR501	PIR	7 m / 140°	VCC, Signal, GND	2
IP 44 Osram	PIR	10 m / 180° rotativo	VCC, Signal, GND	15

El sensor se conecta utilizando los 3 pines. La alimentación directamente a 5 V y el pin de señal a la placa. Se puede ver una imagen de cerca en la figura 3.5.



**Figura 3.5:** Sensor PIR HC-SR501 conectado.

Analizando la opción de relay, se encuentra también una variedad amplia en términos de la potencia soportada. Es recomendable determinar la tensión y corriente a la que funciona el aparato de consumo a regular y en base a eso elegir un relay adecuado. Se propone utilizar un módulo que incluya el relay con la bornera y pines de control, así como también el diodo de protección, para facilitar las conexiones. Para el prototipo, se utilizó un módulo de relay controlado con 5 V y que soporta 10 A a 220 Vac, suficiente para funcionar con una estufa de 2 kW conectada a 220 Vac.

Al igual que con el sensor, la alimentación del relay se conecta directamente a 5 V y la señal de control se conecta a la placa a través de un buffer implementado con un BC547. Esto se definió así para evitar propagar ruido del relay a la placa. La conexión se puede observar en la figura 3.6. La base del transistor es el input que se conecta a la placa y a través del buffer, polarizado con un resistor de 10 kOhms, la señal llega a la entrada del relay.

El borne común del relay se conecta al vivo de la red y el borne normalmente abierto al vivo del equipo de consumo. El neutro y tierra se conectan directamente entre el equipo eléctrico y la red. Esta conexión se muestra en el diagrama de la figura 3.7.

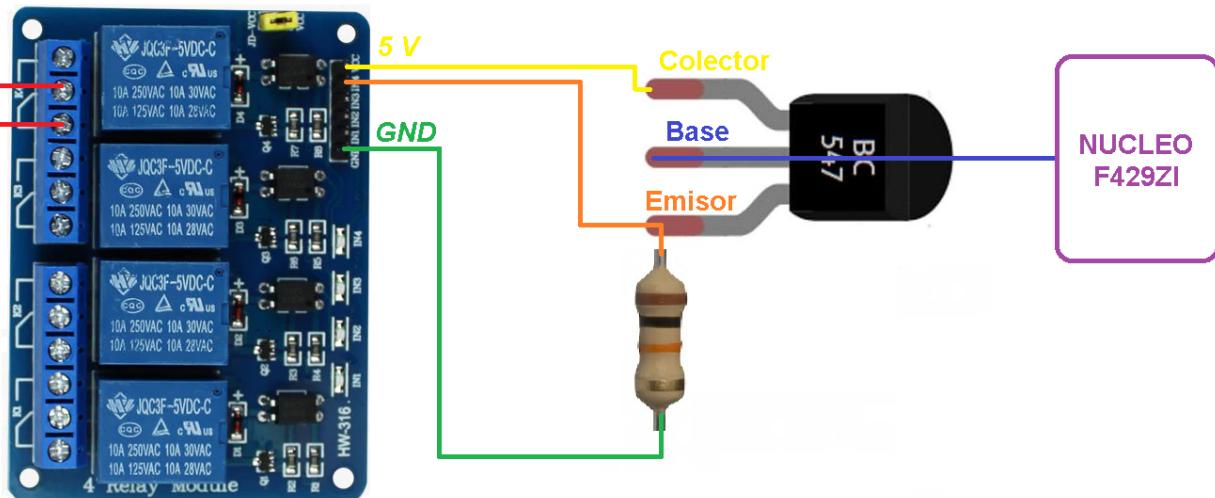


Figura 3.6: Conexión de la placa al relay utilizando un buffer.

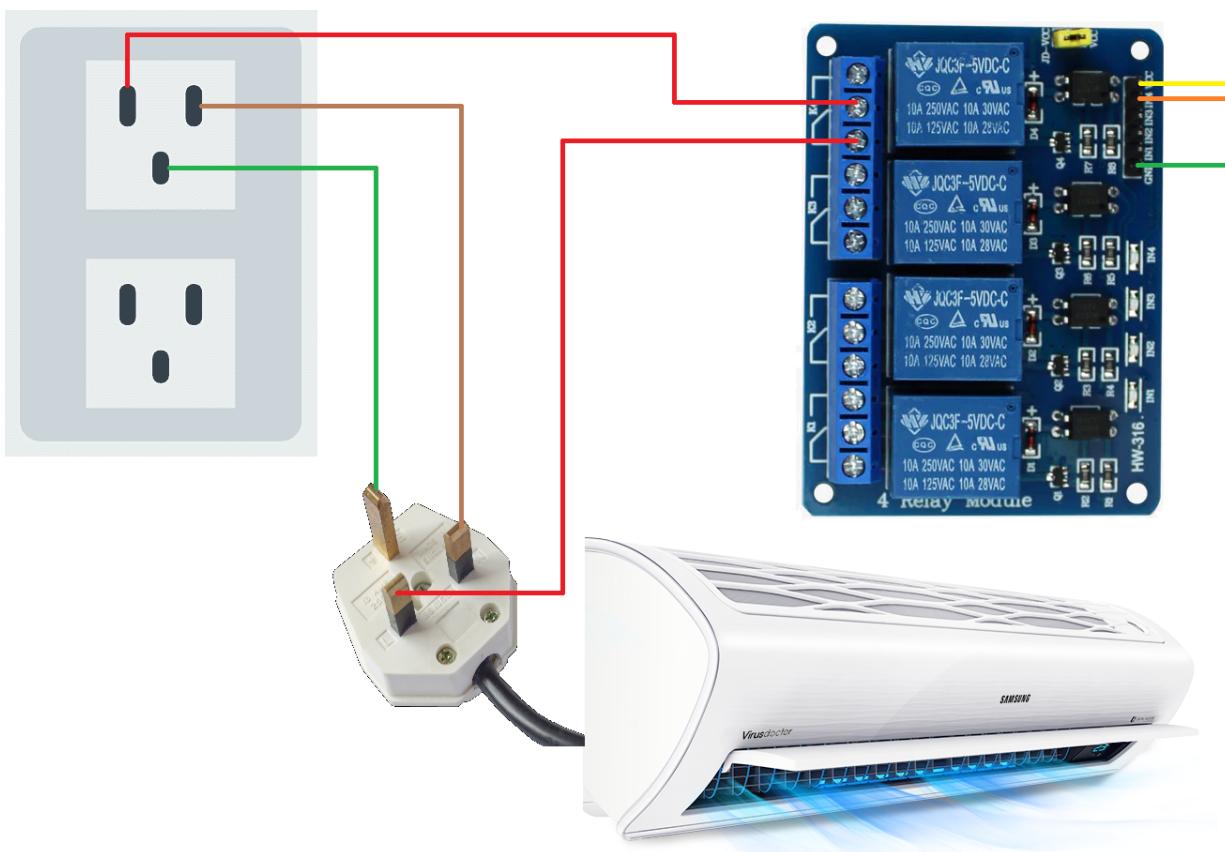
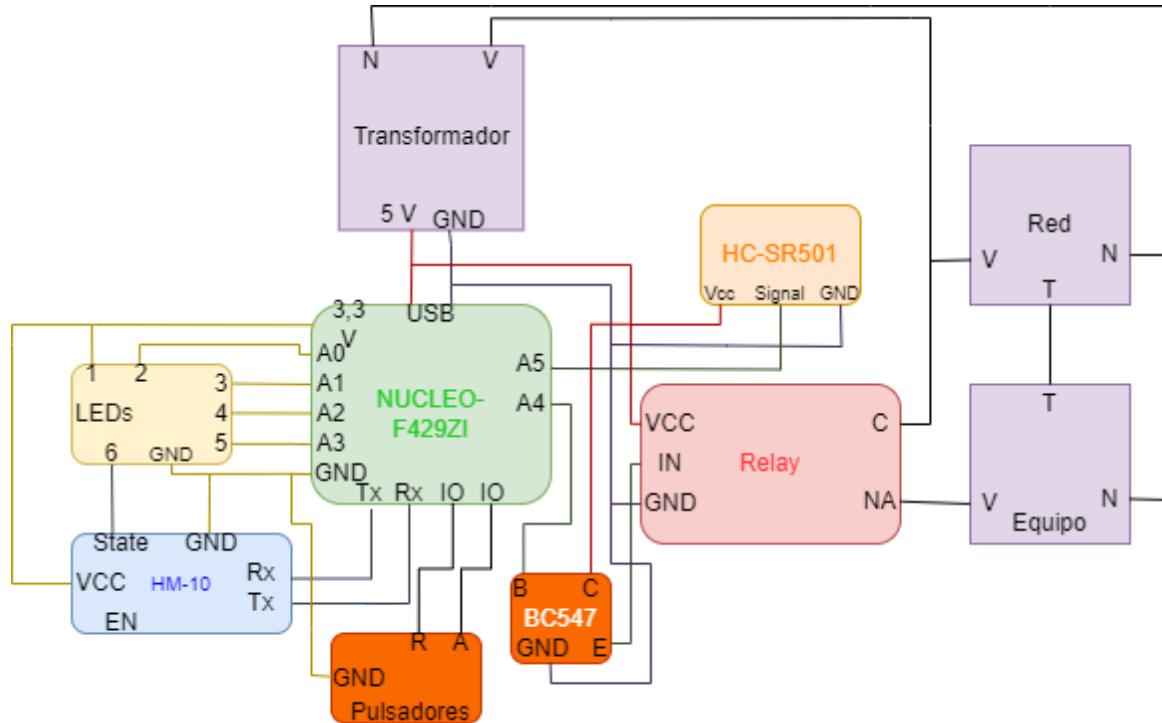


Figura 3.7: Conexión entre el aparato de consumo, el relay y la red.



### 3.1.2 Sistema completo

Habiendo analizado cada módulo, se presenta a continuación el diagrama completo con todas las conexiones del dispositivo. Se puede observar el diagrama en la figura 3.8 y luego en las tablas 3.2 a 3.9 el detalle de cada conexión por componente.



**Figura 3.8:** Diagrama completo del dispositivo.

**Tabla 3.2:** Conexiones de los LEDs.

LED	Conexión
Rojo: encendido.	NUCLEO-F429ZI: 3,3 V.
Rojo: funcionamiento activo.	NUCLEO-F429ZI: A0.
Amarillo: Presencia detectada.	NUCLEO-F429ZI: A1.
Verde: equipo conectado.	NUCLEO-F429ZI: A2.
Rojo: equipo desconectado.	NUCLEO-F429ZI: A3.
Azul: Conexión bluetooth activa.	HM-10: State.
Todos los LEDs al GND de la NUCLEO a través de resistores de 150 Ohms.	

**Tabla 3.3:** Conexiones de la placa NUCLEO-F429ZI.

<b>NUCLEO-F429ZI pin</b>	<b>Conexión</b>
A0	LED rojo de funcionamiento activo.
A1	LED amarillo de presencia detectada.
A2	LED verde de equipo eléctrico conectado.
A3	LED rojo de aparato eléctrico desconectado.
A4	BC547: Base.
A5	HC-SR501: Signal.
UART_RX	HM-10: Tx.
UART_TX	HM-10: Rx.
3,3 V	HM-10: VCC. LED rojo de encendido.
GND	HM-10: GND. Bus de tierra de los 6 LEDs.
USB	Transformador 5 V.

**Tabla 3.4:** Conexiones del HM-10.

<b>Módulo relay pin</b>	<b>Conexión</b>
State	LED azul de conexión BLE activa.
Rx	NUCLEO-F429ZI: UART_TX.
Tx	NUCLEO-F429ZI: UART_RX.
VCC	NUCLEO-F429ZI: 3,3 V.
GND	NUCLEO-F429ZI: GND.
EN	No conectar.

**Tabla 3.5:** Conexiones del equipo eléctrico de consumo.

<b>Equipo eléctrico de consumo</b>	<b>Conexión</b>
Vivo	Módulo relay: NA.
Neutro	Neutro de la red.
Tierra	Tierra de la red.

**Tabla 3.6:** Conexiones de los pulsadores.

<b>Pulsador</b>	<b>Conexión</b>
Rojo	NUCLEO-F429ZI: IO (PG_0)
Amarillo	NUCLEO-F429ZI: IO (PG_1)
Ambos pulsadores conectados directamente al bus de GND. El bus a GND de la NUCLEO-F429ZI.	

**Tabla 3.7:** Conexiones del BC547.

<b>Módulo relay pin</b>	<b>Conexión</b>
Colector	Transformador: 5 V.
Base	NUCLEO-F429ZI: A4.
Emisor	Módulo relay: IN. Transformador: GND a través de un resistor de 10 kOhm.

**Tabla 3.8:** Conexiones del HC-SR501.

<b>Módulo relay pin</b>	<b>Conexión</b>
VCC	Transformador: 5 V.
Signal	NUCLEO-F429ZI: A5.
GND	Transformador: GND a través de un resistor de 10 kOhm.



**Tabla 3.9:** Conexiones del módulo relay.

Módulo relay pin	Conexión
VCC	Transformador: 5 V.
GND	Transformador: GND.
IN	BC547: Emisor.
C	Vivo de la red.
NA	Vivo del equipo eléctrico de consumo.
NC	No conectar.

### 3.1.3 Listado de componentes

A continuación se presenta la tabla 3.10. En esta tabla se encuentran los componentes a utilizar para construir el E-Save Bluetooth. Tener en cuenta que los modelos de relay y sensor de movimiento pueden modificarse y elegirse de manera específica para tener una aplicación más personalizada al tamaño y geometría del ambiente y potencia del equipo de consumo.

**Tabla 3.10:** Listado de componentes.

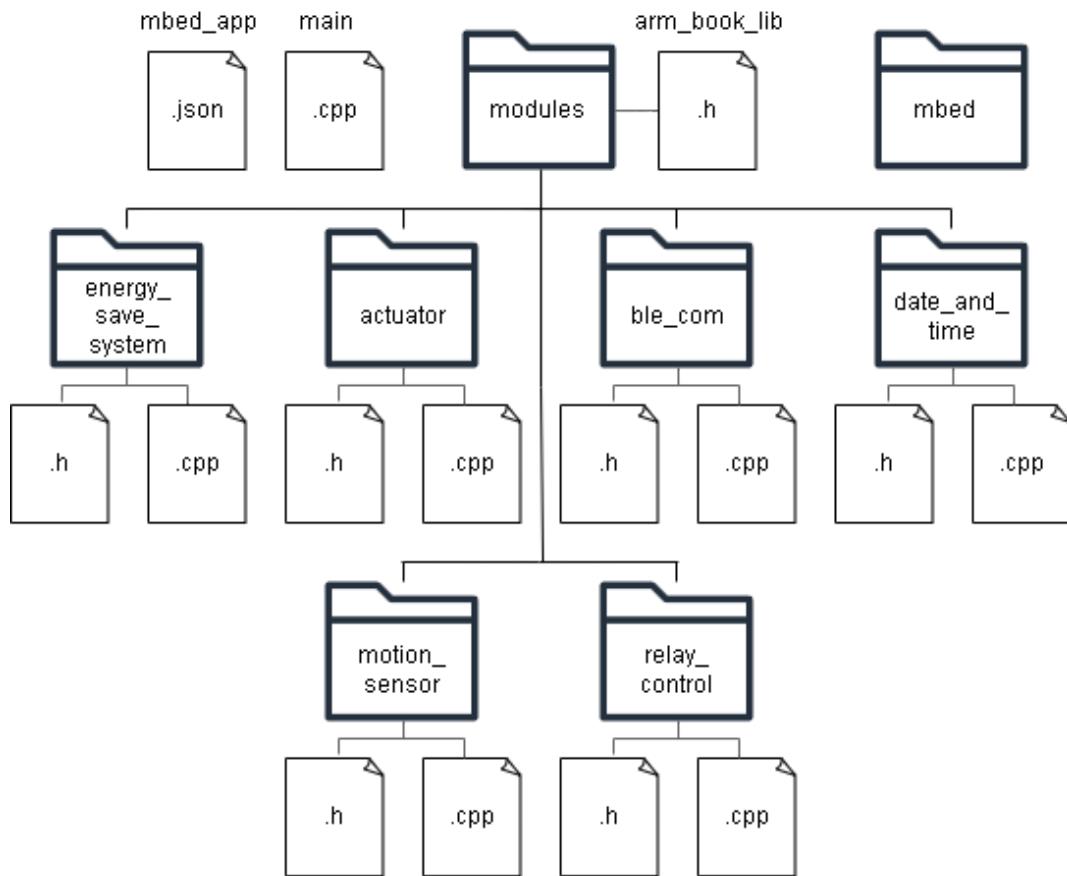
Componente	Cantidad	Costo unitario [ USD ]	Disponibilidad
Placa NUCLEO-F429ZI	1	24	Baja
Módulo relay 220 Vac 10 A	1	4	Alta
Sensor PIR Infrarrojo	1	2	Alta
Módulo serie - Bluetooth HM-10	1	8	Alta
BC547	1	0,1	Alta
Resistor 10 kOhm	1	0,05	Alta
Resistor 150 Ohm	6	0,05	Alta
Pulsador	2	0,1	Alta
LED rojo	3	0,15	Alta

<b>LED amarillo</b>	1	0,15	Alta
<b>LED verde</b>	1	0,15	Alta
<b>LED azul</b>	1	0,15	Alta

Este listado soporta una implementación limitada del producto. El sensor de movimiento infrarrojo PIR podría ser reemplazado por uno de mejor alcance y calidad, también se podrían utilizar varios sensores si la geometría del ambiente es compleja. El relay debe ser dimensionado acorde al consumo del equipo eléctrico conectado. Funcionando a 220 Vac, esta configuración puede funcionar correctamente con un aparato de hasta 2 kW

### 3.2 Firmware del E-Save Bluetooth

Una vez definido el hardware para el dispositivo, se diagrama el software. Los archivos utilizados para el trabajo se organizan según el diagrama de la figura 3.9. Hay un archivo principal, main.cpp, 6 módulos y librerías correspondientes a la placa y microcontrolador. El módulo que encapsula el comportamiento general del programa es el *energy\_save\_system*. Aparte de este, existen los módulos *actuator*, *ble\_com*, *date\_and\_time*, *motion\_sensor* y *relay\_control*.



**Figura 3.9:** Diagrama de archivos del trabajo.

El modo de encarar el desarrollo del firmware fue el siguiente: primero se implementaron de manera simple los drivers `relay_control` y `motion_sensor`, únicamente con la funcionalidad para registrar la señal del sensor y poder accionar el relay. Una vez validado su funcionamiento, se continuó desarrollando el módulo `actuator`, específicamente la lógica para percibir cambios en el sensor y poder accionar el relay de manera automática. En paralelo se agregó el módulo `date_and_time` pero sin comunicación con el actuador.

El módulo `ble_com` maneja la comunicación serie con el HM-10. Inicialmente se implementó para utilizarse con una terminal de la PC y de esta manera se fue diseñando el módulo. Se ayudó de la terminal y este módulo para ir probando de manera segmentada la funcionalidad del actuador y demás módulos. Recién una vez que estaba todo implementado, se conectó el HM-10 y se realizaron los ajustes necesarios para establecer la comunicación BLE efectiva.

En la tabla 3.11 se puede encontrar la funcionalidad y rol de cada módulo.

**Tabla 3.11:** Funcionalidad y roles de cada módulo.

Módulo	Funcionalidad	Rol
<i>energy_save_system</i>	Llamar a inicializar y actualizar los subsistemas involucrados.	Sistema
<i>ble_com</i>	Manejar la interfaz vía BLE con el usuario.	Subsistema
<i>actuator</i>	Accionar lógica de control en función del estado del sistema e input del sensor.	Subsistema
<i>date_and_time</i>	Manejo del RTC ( <i>Real Time Clock</i> ).	Subsistema
<i>motion_sensor</i>	Interpretar la señal del sensor de movimiento y llevar la cuenta del tiempo acumulado de no detectar movimiento.	Driver
<i>relay_control</i>	Accionar el relay.	Driver

### 3.2.1 Lógica principal

La lógica principal del programa puede encontrarse en el archivo *main.cpp* y el módulo *energy\_save\_system*. Sin entrar en detalles de implementación de cada módulo, el programa comienza inicializando el actuador y la comunicación BLE y luego las actualiza cada 1 ms<sup>3</sup>. La implementación de *main.cpp* y el diagrama de flujo que describe esta lógica principal puede observarse a continuación en el código 3.1 y la figura 3.10.

---

<sup>3</sup> Se eligió 1 ms para poder captar correctamente la información recibida vía BLE. Se profundiza sobre esto en la subsección 3.2.2 Comunicación BLE de este mismo capítulo.

```

12 //===== [Libraries] =====
13
14 #include "energy_save_system.h" //> System module.
15
16 //===== [Main function, the program entry point after power on or reset] =====
17
18 /**
19 * @brief Main function. Setup & main loop.
20 *
21 */
22 int main()
23 {
24     // Setup:
25     energySaveSystemInit();
26
27     // Mainloop:
28     while (true) {
29         energySaveSystemUpdate();
30     }
31 }
```

Código 3.1: Implementación del *main.cpp*.

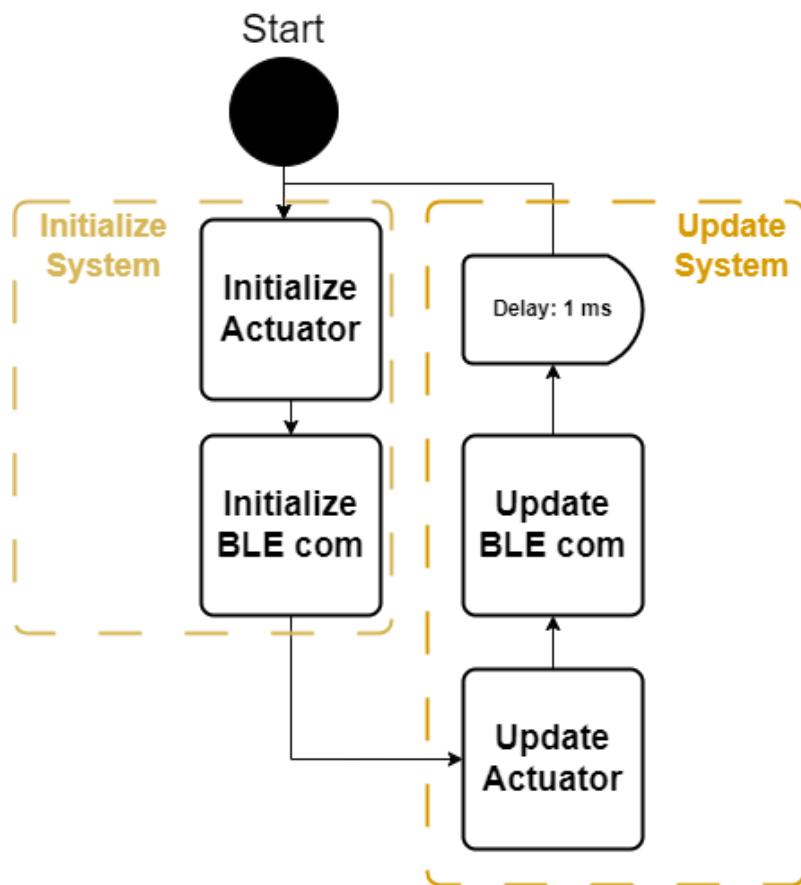
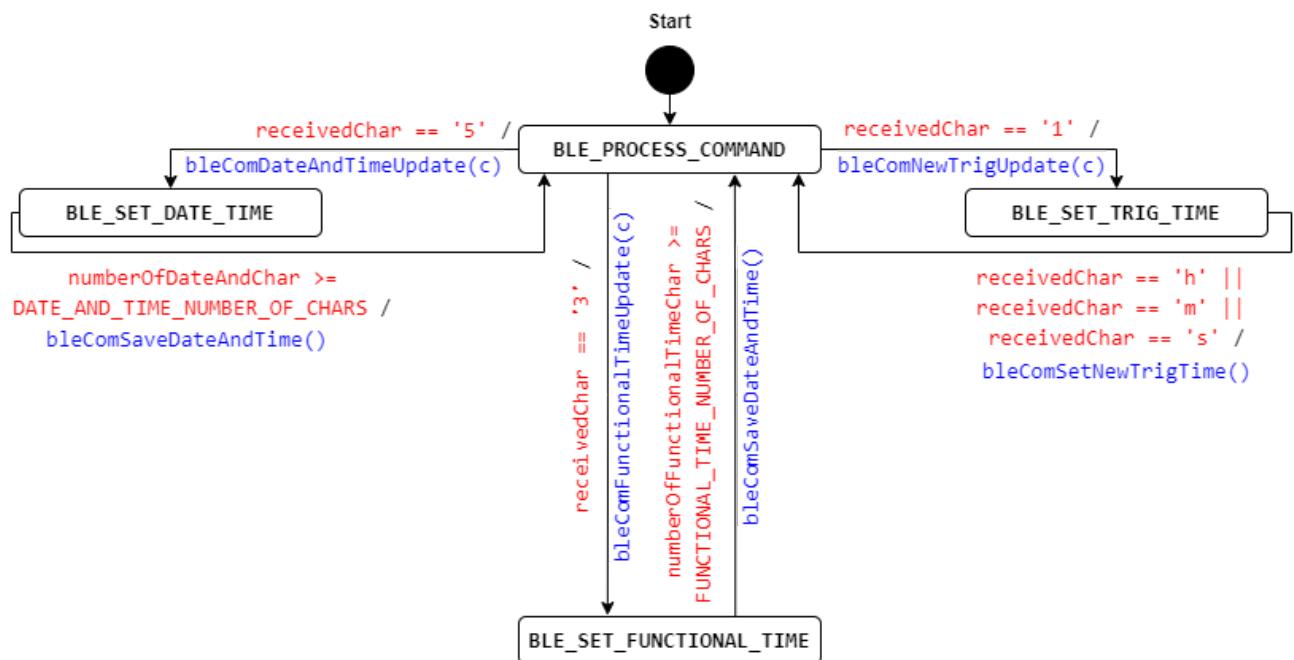


Figura 3.10: Lógica principal del programa.

### 3.2.2 Comunicación BLE

El módulo *ble\_com* encapsula la comunicación vía BLE. Su responsabilidad es verificar si se recibió información del HM-10 y decodificarla. Para poder lograrlo, se diseñó la máquina de estados de la figura 3.11. Este módulo tiene 4 estados de funcionamiento, un estado por defecto de *BLE\_PROCESS\_COMMAND* dentro del cual aguarda recibir un comando del usuario y en cuanto se recibe procede a ejecutarlo. Si uno de estos comandos es '1', '3' o '5', se cambia de estado para poder ejecutarlo. En cuanto se finaliza, se vuelve a *BLE\_PROCESS\_COMMAND*. Los comandos para configurar la fecha y hora, el horario de funcionamiento activo y el tiempo de espera son de mayor complejidad que el resto, por esto tienen sus correspondientes estados. El módulo, en cada uno de estos estados, aguarda input del usuario para actualizar los parámetros, a diferencia de comandos como '2: Get current trig time' según el cual el sistema informa al usuario el actual tiempo de espera y rápidamente finaliza la ejecución del comando, sin aguardar input del usuario, todo mientras se encuentra en el estado *BLE\_PROCESS\_COMMAND*.



**Figura 3.11:** Diagrama de estados del módulo BLE.

Las funciones públicas de este módulo se pueden encontrar en el archivo *modules/ble\_com.h*. Son solo dos y sus detalles se muestran en la tabla 3.11. Son las funciones para inicializar y actualizar el módulo.

**Tabla 3.12:** Funciones públicas del módulo *ble\_com*.

Función	Descripción	Módulos que la utilizan
bleComInit()	Inicializa el módulo. Configura la comunicación serial.	<i>energy_save_system</i>
bleComUpdate()	Actualiza el módulo. Procesa los comandos recibidos vía BLE.	<i>energy_save_system</i>

Al momento de implementar el sistema con el HM-10, se utilizó un analizador lógico para observar la comunicación entre la placa y el periférico. Al inspeccionar las tramas enviadas por el HM-10, se observa un intervalo de 1 ms entre los caracteres enviados. Por este motivo, se decide actualizar el *ble\_com* cada 1 ms, a fines de no perder información.

### 3.2.3 Actuador

El módulo *actuator* es el otro subsistema de gran importancia en el trabajo. El mismo implementa la lógica principal de control de consumo. Presenta 5 estados posibles de acuerdo de si está fuera del horario de funcionamiento activo, si está bloqueado, si se detecta o no movimiento y si el relay está desconectado por no detectarse presencia en el ambiente. El diagrama de estados se puede ver en la figura 3.12.

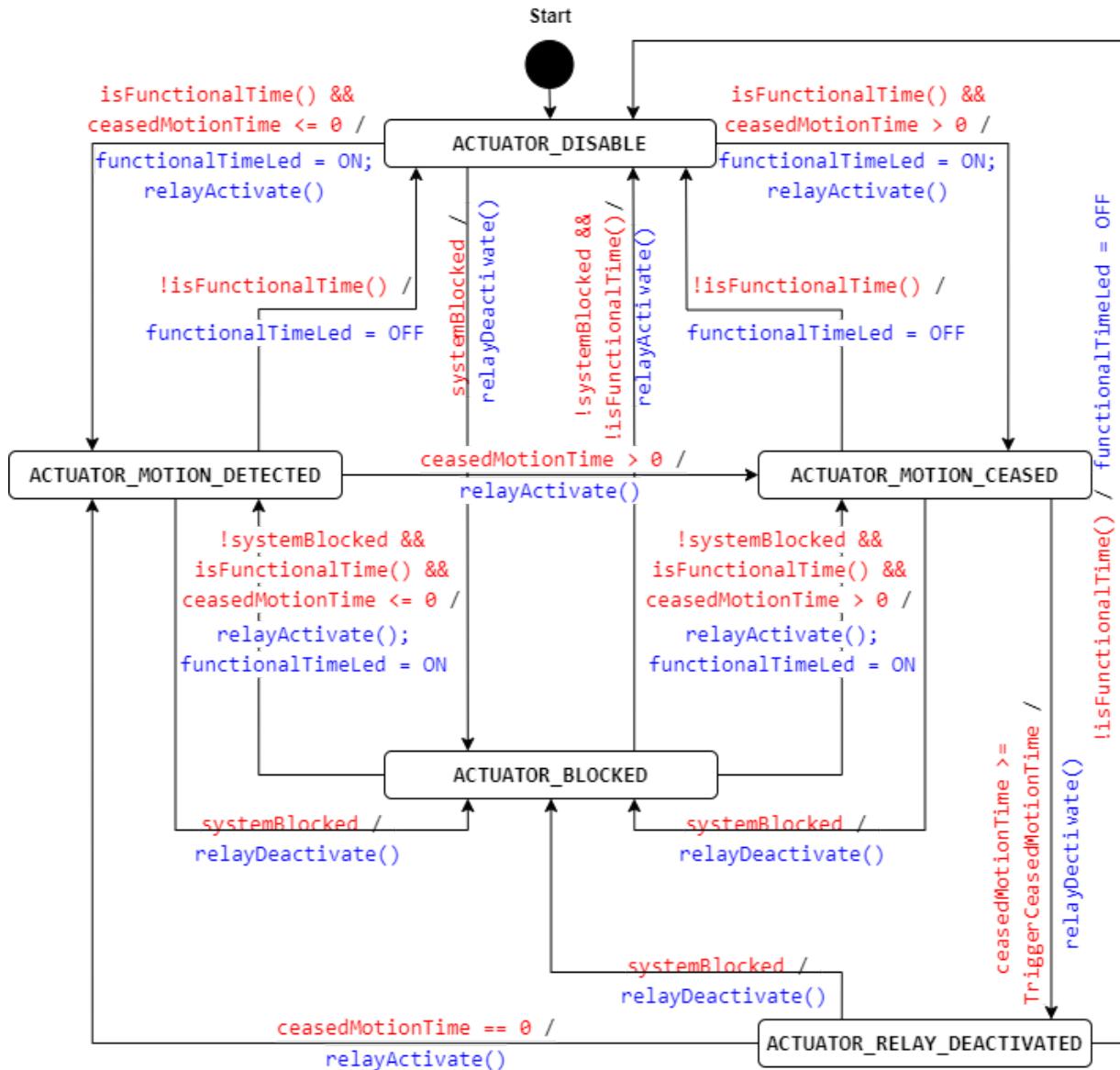


Figura 3.12: Diagrama de estados del módulo *actuator*.

El sistema se inicia por defecto en el estado *ACTUATOR\_DISABLE*, correspondiente al horario fuera de funcionamiento activo. Si se detecta que se encuentra en horario de funcionamiento activo, por consultar al módulo *date\_and\_time*, se mueve a uno de los dos estados: *ACTUATOR\_MOTION\_DETECTED* / *ACTUATOR\_MOTION\_CEASED*, en función del estado indicado por el módulo *motion\_sensor*. Una vez dentro del estado de no estar detectando movimiento y transcurrido el tiempo de espera, se pasa a desactivar el relay transicionando al estado *ACTUATOR\_RELAY\_DEACTIVATED*. De este estado se puede salir si se detecta movimiento o se sale del rango de funcionamiento activo.

En paralelo a esta lógica, se puede ingresar en el estado de bloqueo si se presiona el botón rojo. La única manera de salir es presionando el botón amarillo para desbloquear el sistema.

Durante el desarrollo del proyecto, se comenzó implementando los estados *ACTUATOR\_MOTION\_DETECTED* y *ACTUATOR\_MOTION\_CEASED*. De este modo se integró la funcionalidad del driver del sensor con la lógica principal. Luego se procedió a manejar el relay y agregar el estado *ACTUATOR\_RELAY\_DEACTIVATED*. Una vez completa la funcionalidad básica, se agregaron los otros estados *ACTUATOR\_DISABLE* y *ACTUATOR\_BLOCKED* a los cuales se puede transicionar desde cualquier otro estado.

Se presentan en la tabla 3.13 el detalle de las funciones públicas que tiene este módulo.

**Tabla 3.13:** Funciones públicas del módulo *actuator*.

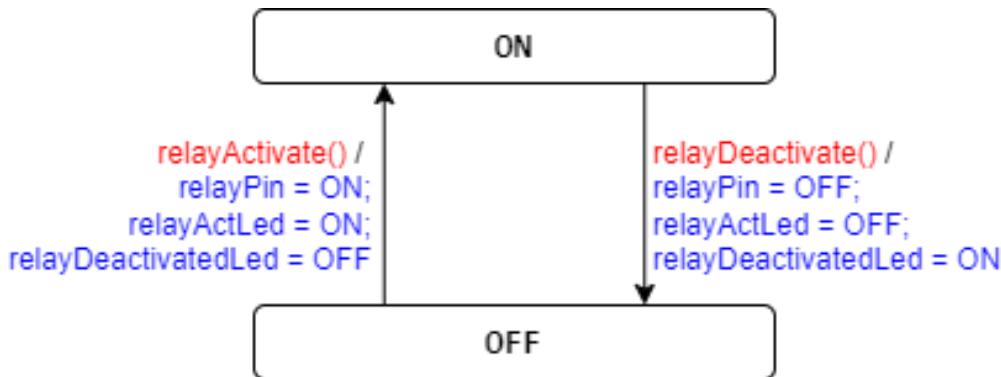
Función	Descripción	Módulos que la utilizan
<code>actuatorInit()</code>	Inicializa el módulo. Configura interrupciones e inicializa los drivers.	<i>energy_save_system</i>
<code>actuatorUpdate()</code>	Actualiza el módulo. Actualiza su estado en función del input del sensor, interrupciones y horario.	<i>energy_save_system</i>
<code>setTriggerMotionCeasedTime_ms(int)</code>	Configura el tiempo de espera en ms después del cual desactivar el relay si no se detecta movimiento.	<i>ble_com</i>
<code>getTriggerTime_ms()</code>	Devuelve el tiempo de espera en ms	<i>ble_com</i>

### 3.2.4 Relay

El funcionamiento del driver del relay es relativamente simple. Presenta dos estados: relay activado y desactivado. La salida para controlar el relay es digital, con un '1' lógico se activa el relay y de esta manera se conecta el aparato de consumo a la red eléctrica. Si la

salida es un '0' lógico, se desactiva el relay y el equipo es desconectado de la red. El pin de control se configura como *DigitalOut*, tiene siempre un '1' o '0' lógico a la salida.

El diagrama de estados se presenta en la figura 3.13. A diferencia de los anteriores módulos, esta máquina de estados no se implementa en código con un tipo enumerativo y un *switch*, dada la simpleza de tener solo dos estados de funcionamiento se hace uso de una variable de estado booleana. Los cambios de estado se realizan cuando se llama a las funciones *relayActivate()* y *relayDeactivate()*.



**Figura 3.13:** Diagrama de estados del driver del relay.

El driver actualiza también el valor de los LEDs verde y rojo que indican visualmente al usuario si el equipo de consumo está conectado a la red o no.

Las funciones públicas de este driver se pueden observar a continuación en la tabla 3.14.

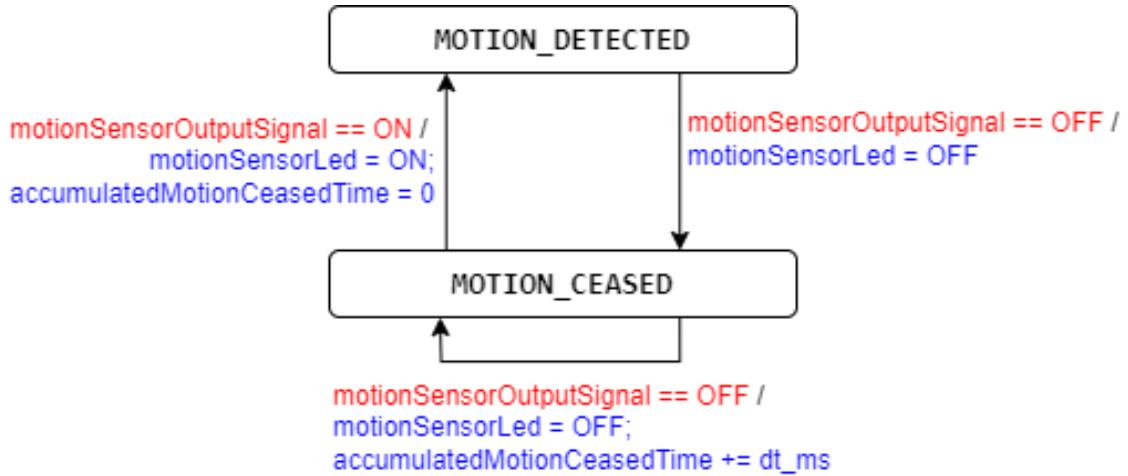
**Tabla 3.14:** Funciones públicas del módulo *relay\_control*.

Función	Descripción	Módulos que la usan
<code>relayControlInit()</code>	Inicializa el pin de control y LEDs de estado.	<i>actuator</i>
<code>relayActivate()</code>	Activa el relay y actualiza los LEDs.	<i>actuator</i>
<code>relayDeactivate()</code>	Desactiva el relay y actualiza los LEDs.	<i>actuator</i>

### 3.2.5 Sensor de movimiento

El segundo driver del trabajo es el del sensor de movimiento. Al igual que el anterior, tiene dos estados posibles: *MOTION\_DETECTED* y *MOTION\_CEASED*. El subsistema *actuator* llama a actualizar al driver y éste actualiza su estado en función del input del

sensor. Se actualiza también en LED correspondiente. El diagrama de estados se puede observar en la figura 3.14.



**Figura 3.14:** Diagrama de estados del módulo *motion\_sensor*.

Rutinariamente se verifica el estado del sensor. Si no se detecta movimiento, se incrementa el tiempo acumulado y en cuanto se detecta movimiento se resetea. La variable 'int accumulatedMotionCeasedTime' se devuelve al módulo *actuator* para que éste luego actúe en consecuencia. Su interfaz con el módulo *actuator* se maneja con las funciones públicas que se muestran en la tabla 3.15 a continuación.

**Tabla 3.15:** Funciones públicas del módulo *motion\_sensor*.

Función	Descripción	Módulos que la usan
<code>motionSensorInit(int dt)</code>	Inicializa el pin de entrada y el LED de estado.	<i>actuator</i>
<code>motionSensorUpdate()</code>	Actualiza su estado en función del input del sensor. Devuelve el tiempo acumulado de no detectar movimiento. Si se está detectando movimiento devuelve 0.	<i>actuator</i>

### 3.2.6 RTC

El Real Time Clock se maneja a través del módulo *date\_and\_time*. Este módulo inicializa el reloj cargando inicialmente una fecha y horario por defecto y luego actualizándose a pedido de *ble\_com* cuando el usuario lo demanda. Aparte de controlar el RTC, este módulo almacena el rango horario de funcionamiento activo definido por el usuario. Por defecto se inicializa con fecha y hora 00:00:00 del 01/01/2000 y un rango de 00:00:00 hs a 00:00:00 hs lo cual implica que nunca está en funcionamiento activo. Una vez configurados la fecha y hora y el rango horario por el usuario, *ble\_com* llama a actualizar estos valores.

Se debe destacar que el módulo no determina si el rango de funcionamiento activo es válido o no. Recae sobre el usuario la responsabilidad de ingresar un valor con sentido. Es una propuesta interesante la de agregar una validación para proteger al sistema de un input incorrecto por parte del usuario y ayudarlo a éste a utilizar mejor el producto.

Junto con almacenar el rango de funcionamiento activo, recae sobre el módulo la lógica de verificación de si el sistema se encuentra dentro de ese rango. Esto se implementa en la función pública 'bool *isFunctionalTime()*'. Se puede observar su implementación en el código 3.2.

Esta función verifica por partes si el horario de funcionamiento activo inicia antes de la hora actual (entre las líneas 71 y 81) y si finaliza después (entre las líneas 83 y 93). Si se verifica que en efecto el rango inicia antes y finaliza después de la hora actual se devuelve *true*. Esta funcionalidad limita los valores admitidos para el rango de funcionamiento activo: el rango debe estar contenido dentro de un mismo día calendario. No sería válido, aunque permisible, configurar un rango de funcionamiento activo que inicie a las 17:00 hs y finaliza a las 4:00 am del día siguiente. Si el usuario ingresa esos valores, el sistema registrará que nunca está en funcionamiento activo. Se propone como mejora en el corto plazo integrar una nueva funcionalidad que permita configurarse de esta manera e incluso fijar horarios de funcionamiento para los distintos días de la semana, involucrarse más con el calendario.

```
63     bool isFunctionalTime()
64     {
65         bool isAfterStartTime = false;
66         bool isBeforeEndTime = false;
67
68         time_t current_time = time(NULL);
69         struct tm * now = localtime(&current_time);
70
71         if(now->tm_hour > startTime.tm_hour) {
72             isAfterStartTime = true;
73         } else if(now->tm_hour == startTime.tm_hour) {
74             if(now->tm_min > startTime.tm_min) {
75                 isAfterStartTime = true;
76             } else if(now->tm_min == startTime.tm_min) {
77                 if(now->tm_sec > startTime.tm_sec) {
78                     isAfterStartTime = true;
79                 }
80             }
81         }
82
83         if(now->tm_hour < endTime.tm_hour) {
84             isBeforeEndTime = true;
85         } else if(now->tm_hour == endTime.tm_hour) {
86             if(now->tm_min < endTime.tm_min) {
87                 isBeforeEndTime = true;
88             } else if(now->tm_min == endTime.tm_min) {
89                 if(now->tm_sec < endTime.tm_sec) {
90                     isBeforeEndTime = true;
91                 }
92             }
93         }
94
95         return isAfterStartTime && isBeforeEndTime;
96     }
```

Código 3.2: 'bool isFunctionalTime()', función pública de *date\_and\_time*.

El módulo *date\_and\_time* utiliza data types y funciones del módulo *time.h* provisto por mbed para el desarrollo en su placa. Se lista a continuación los detalles relevantes.

- **struct tm.** Estructura que almacena en valores enteros la fecha (día, mes y año) y la hora (horas, minutos y segundos). Tiene también un flag para indicar horario de verano<sup>4</sup>.
- **time\_t.** Tipo de variable *unsigned int* que representa el tiempo en segundos desde el primero de enero de 1970, fecha inicial.
- **time\_t time(time\_t \*).** Función que devuelve la fecha y hora actual del RTC como variable *time\_t*. Puede devolverse por argumento si se le pasa una dirección de memoria de una variable de tipo *time\_t*.
- **char \* ctime(time\_t).** Convierte una fecha y hora almacenada en tipo *time\_t* a una string en formato “Fri Sep 16, 17:11:02 2023\n”.
- **time\_t mktime(struct tm \*).** Convierte una fecha y hora almacenada en una estructura *tm* en una variable de tipo *time\_t*
- **void set\_time(time\_t).** Inicializa el RTC en tiempo *time\_t*.

El módulo hace uso de estas variables y funciones para almacenar el rango de funcionamiento activo, actualizar el RTC y verificar si el horario actual cae dentro del rango de funcionamiento activo. Fue vital investigar y comprender la librería *time.h* para el desarrollo de este módulo y su funcionalidad. Por ejemplo, para comparar el horario actual con el rango de funcionamiento activo, se pasó toda variable temporal al tipo struct *tm*.

### 3.2.7 Repositorio

El código fuente se puede revisar en el repositorio público [https://github.com/deve023/E-Save\\_Bluetooth](https://github.com/deve023/E-Save_Bluetooth). Dentro del mismo se encuentra un README.md completo con descripciones similares a las incluidas en este documento y un manual de usuario. Es de acceso público, se invita a descargar y probar su funcionamiento. La estructura de archivos es la misma explicada en la subsección 3.2 Firmware del E-Save Bluetooth de este mismo capítulo.

---

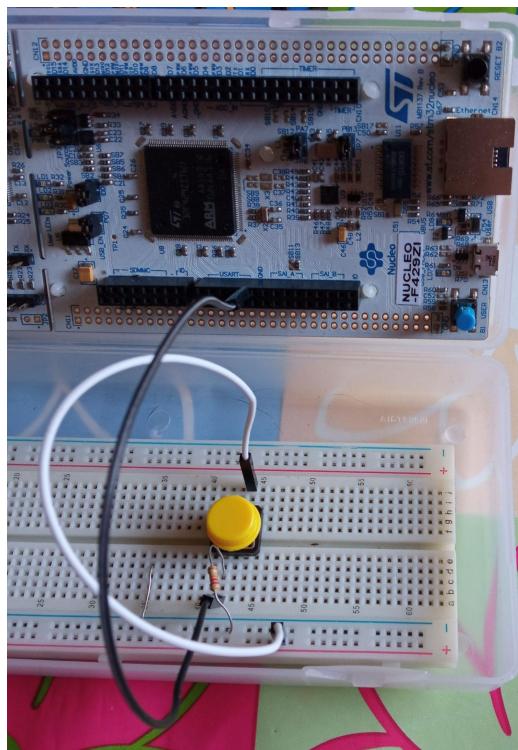
<sup>4</sup> Esto es importante si la región implementa un cambio de horario en el verano como ciertos países para ajustarse al corrimiento de luz del día durante la temporada.

## CAPÍTULO 4

### Ensayos y resultados

#### 4.1 Pruebas funcionales del hardware

A medida que se implementaron los drivers y lógica de control, se realizaron pruebas y ensayos para verificar que todo funcione correctamente. Primero se verificó el funcionamiento de los drivers de manera aislada y emulando los módulos con pulsadores y LEDs. Una vez validado el funcionamiento se continuó con pruebas del sistema pero emulando el sensor y relay con un pulsador y LED de la placa misma. En la figura 4.1 se puede observar un pulsador conectado a la placa para emular el input del sensor de movimiento.



**Figura 4.1:** Pulsador emulando el sensor de movimiento.

## 4.2 Pruebas funcionales del firmware

Luego de verificar que el sistema interpreta correctamente la salida del sensor y puede emitir una salida de control al relay, se profundiza en las pruebas de los subsistemas.

### 4.2.1 Comunicación serie

El módulo *ble\_com* contiene un extenso código aunque una simple máquina de estados. Para no acarrear complicaciones, se implementaron primero los estados posibles y las transiciones entre ellos pero sin realmente actuar en cada uno.

El módulo está fraccionado en los distintos procesos correspondientes a los comandos del usuario. Una vez procesado el comando ingresado, se dispara uno de los procesos. Se muestra en el código 4.1 la decodificación de los comandos.

```

206 static void bleProcessCommand(char c)
207 {
208     switch(c) {
209         case '1':
210             bleCommandNewTrigTime();
211             break;
212         case '2':
213             bleCommandPrintTrigTime();
214             break;
215         case '3':
216             bleCommandNewFunctionalTime();
217             break;
218         case '4':
219             bleCommandPrintFunctionalTime();
220             break;
221         case '5':
222             bleCommandNewDateAndTime();
223             break;
224         case '6':
225             bleCommandPrintDateAndTime();
226             break;
227         case 'h':
228             printAcceptableCommands();
229             break;
230         default:
231             bleComStringWrite("Command not understood. Please review acceptable commands:\r\n");
232             printAcceptableCommands();
233     }
234 }
```

**Código 4.1:** Decodificación de los comandos posibles.

La validación de cada proceso es independiente ya que no interfieren unos con otros. Todos las pruebas iniciales se realizaron vía *CoolTerm*<sup>5</sup> en la PC. Recién en la última etapa de testeo se repitieron las pruebas utilizando BLE y el módulo HM-10.

Pasar de usar la terminal serie del *CoolTerm* en la PC a utilizar el HM-10 y la aplicación *Serial Bluetooth Terminal* en teoría no debía traer inconvenientes ya que el código es transparente. Los protocolos BLE son implementados dentro del HM-10 y lo que ve el micro es comunicación serie a través de sus pines Rx y Tx. Sin embargo, al conectar el módulo externo e iniciar la comunicación se encontraron problemas.

Había fallas en la comunicación entre la placa y el HM-10. Lo que enviaba uno no parecía recibirla el otro, o al menos no se accionaba en consecuencia. Frente al desconcierto, se utilizó un analizador lógico para visualizar la comunicación. Una imagen del mismo se muestra en la figura 4.2. Tiene 8 canales y un muestreo a 24 MHz, aunque para los efectos de estas pruebas se utilizó a 16 MHz.



**Figura 4.2:** Analizador lógico.

Se analizaron las tramas enviadas y recibidas por ambas placas. La información se transmitía. Al enviar uno o varios caracteres por la terminal de la aplicación, las tramas eran captadas por el analizador lógico, y consecuentemente la placa receptora, pero lo que resaltó fue la velocidad de transmisión de caracteres. Los caracteres ingresados se transmitían cada 1 ms. El sistema se actualizaba cada 100 ms. Esta discrepancia imposibilitaba la transmisión. Se ajustó el intervalo de actualización a 1 ms y el problema desapareció. La comunicación era efectiva. Se tuvo que ajustar también un detalle de configuración del Serial Bluetooth Terminal: no enviar caracteres '\n' y '\r' al final de cada envío ni mostrar en pantalla lo que se va enviando. De esta manera es más simple

---

<sup>5</sup>*CoolTerm* es una aplicación que contiene una terminal para comunicarse vía USB-Serie con el microcontrolador.

controlar la terminal desde el microcontrolador, el que va haciendo *echo* de los caracteres recibidos cuando se determina necesario.

#### **4.2.2 Actuador**

La lógica del actuador se verificó de a partes. Las interrupciones de los pulsadores rojo y amarillo se dejó para el final. Primero se hicieron pruebas de la lógica base estando en funcionamiento activo el sistema. cuando el sensor (emulado primero por un pulsador, luego conectando el módulo HC-SR501) no detectaba movimiento por un determinado tiempo (fijo) el relay se desactivaba. El relay primero se emuló con un LED de la placa. Luego a circuito abierto se verificó que no se veía afectada la funcionalidad. Posteriormente se conectó un LED a 5 V a través del relay, luego una lámpara a 220 Vac y finalmente se probó la funcionalidad del sistema con una estufa de 2 kW a 220 Vac.

Luego de verificar la lógica base. Se verificó que variando el estado de funcionamiento activo, el sistema no desactivaba el relay sin importar el estado del sensor ni el tiempo que pasaba.

Finalmente se verificó que las interrupciones funcionaban correctamente. Sin la presencia del buffer BC547, el ruido producido por el switching del relay era tan grande que se accionaba la interrupción del pulsador rojo. Con la etapa buffer conectada, no se percibieron anomalías.

#### **4.3 Pruebas de integración**

Para completar la validación del proceso, se realizaron pruebas de integración. Principalmente verificando el acople de los dos grandes subsistemas: *ble\_com* y *actuator*. Se verificaron los distintos casos de uso, en particular cómo afectan al flujo principal del actuador las configuraciones introducidas por el usuario en vivo a través de la comunicación BLE.

Al realizar estas pruebas, se desconectó de vuelta el relay de la red y luego se volvió a conectar de a pasos: primero conectando un LED a 5 V, luego una lámpara a 220Vac y finalmente la estufa de 2 kW.

#### **4.4 Cumplimiento de requisitos**

Los 26 requisitos se cumplieron satisfactoriamente, se puede observar el detalle en la tabla 4.1.

**Tabla 4.1:** Requisitos cumplidos del trabajo.

Grupo	ID	Descripción	¿Alcanzado?
9. Sensor	1.1	El sistema podrá detectar la presencia de una persona en el ambiente.	✓
	1.2	El sistema podrá detectar cuando no haya nadie presente en el ambiente.	✓
	1.3	El sistema podrá llevar cuenta de cuánto tiempo continuo no detecta presencia en el ambiente.	✓
	1.4	El sistema indicará mediante un LED amarillo si hay alguien en el ambiente.	✓
10. Interfaz	2.1	El sistema tendrá un pulsador que al ser presionado desconecta el equipo de consumo.	✓
	2.2	El sistema tendrá un pulsador que al ser presionado conecta el equipo de consumo.	✓
	2.3	El sistema permite al usuario conectarse vía Bluetooth mediante una aplicación al mismo.	✓
	2.4	El sistema permite enviar mensajes de ayuda al usuario vía Bluetooth mediante la aplicación utilizada.	✓
	2.5	El sistema permite recibir mensajes (instrucciones) vía Bluetooth mediante la aplicación utilizada.	✓
	2.6	El sistema permite configurar el timer de acción en función de la información recibida vía Bluetooth.	✓
	2.7	El sistema indicará mediante un LED verde que el equipo de consumo se encuentra en conectado a su alimentación	✓
	2.8	El sistema indicará mediante un LED rojo que el equipo de consumo se encuentra desconectado de su alimentación	✓
	2.9	El sistema indicará mediante un LED azul si hay una conexión Bluetooth activa.	✓
11. Reloj	3.1	El sistema tendrá un reloj interno de tiempo real.	✓
	3.2	Mediante la comunicación Bluetooth se podrá configurar el horario del reloj interno del dispositivo.	✓

	3.3	El usuario podrá definir vía Bluetooth intervalos de horarios para los cuales el dispositivo está en funcionamiento activo (sensando y accionando).	✓
	3.4	Dentro del rango horario definido para el funcionamiento activo, el microcontrolador encenderá un LED rojo a fin de indicar el estado activo.	✓
	3.5	Fuera del rango horario definido para el funcionamiento activo, el microcontrolador no desconectará el relay sin importar el estado del sensor.	✓
12. Control	4.1	El sistema accionará un relay para conectar/desconectar la alimentación del aparato de consumo.	✓
	4.2	El sistema desconectará la alimentación del aparato de consumo si no detecta presencia en el ambiente luego de X tiempo.	✓
13. Alimentación	5.1	El sistema se alimentará mediante el uso de un transformador 220 V a 5 V.	✓
	5.2	La placa NUCLEO F429ZI se alimentará con 5 V	✓
14. Costo	6.1	Implementar el proyecto tendrá un costo de materiales menor a 50 USD.	✓
	6.2	Implementar el proyecto tendrá un costo de horas de trabajo menor a 240 USD, a un valor de 6 USD/hora.	✓
15. Tiempo	7.1	El proyecto será finalizado antes del 28 de junio de 2023.	✓
16. Documentación	8.1	Junto al prototipo, se proveerá un listado de partes, el repositorio de código documentado y un manual de uso.	✓

Los casos de uso también fueron validados y verificados. Su detalle se puede encontrar en la tabla 4.2.

**Tabla 4.2:** Casos de uso cumplidos.

Caso de uso	Descripción	¿Cumplido?
#1	El sistema registra que no hubo nadie en el ambiente por 20 minutos.	✓
#2	El usuario presiona el pulsador para desconectar el equipo.	✓
#3	El usuario envía un comando a través de la aplicación para configurar el tiempo de espera a 15 minutos.	✓

## 4.5 Comparación con otros sistemas similares

Teniendo en cuenta lo implementado y los resultados del trabajo, se analiza nuevamente el rol del producto en el mercado actual. La comparación actualizada se puede observar en la tabla 4.3. El costo de USD 40 fue estimado correctamente. Esto posiciona al producto en un punto ventajoso ya que trae una funcionalidad útil y novedosa mientras se coloca en un alcance económico en comparación al termostato inteligente.

**Tabla 4.3:** Comparación de los productos ofrecidos por el mercado y el E-Save Bluetooth implementado.

Característica	Enchufe inteligente. <a href="#">Link</a>	Termostato Inteligente <a href="#">Link</a>	E-Save Bluetooth <a href="#">Link</a>
Conectividad a internet / Wi-Fi	Sí	Sí	No
Conectividad Bluetooth	No	No	Sí
Control manual	Sí	Sí	Sí
Lógica de control	Encendido/Apagado accionado por el usuario vía Wi-Fi	Encendido/Apagado accionado por el usuario vía aplicación móvil y Wi-Fi. También permite configurar timers para encender y apagar.	Encendido/Apagado accionado por el usuario o automáticamente cuando no se detecta presencia por cierto tiempo.

<b>Sensor de movimiento</b>	No	No	Sí
<b>Configuración</b>	Encendido/Apagado accionado por el usuario. No posee configuración de timer.	Vía Wi-Fi.	Vía Bluetooth.
<b>Precio estimado (USD)</b>	\$12	\$360	\$40

## 4.6 Documentación del desarrollo realizado

Se presenta a continuación, en la tabla 4.4, un resumen para organizar la documentación del trabajo. En la tabla se encuentran los principales puntos a tener en cuenta acerca del trabajo y su desarrollo.

**Tabla 4.4:** Elementos más importantes del trabajo y su desarrollo.

<b>Elemento</b>	<b>Referencia</b>
Origen del producto	1.1 Origen del proyecto
Requerimientos	2.1 Requisitos
Casos de uso	2.2 Casos de uso
Hardware del E-Save Bluetooth	3.1 Hardware del E-Save Bluetooth
Conexiones de los módulos	3.1.2 Sistema completo
Listado de componentes	3.1.3 Listado de componentes
Firmware del E-Save Bluetooth	3.2 Firmware del E-Save Bluetooth
Repositorio	<a href="#">Link a Github</a>
Requerimientos y casos de uso alcanzados	4.4 Cumplimiento de requisitos
Conclusiones	Capítulo 5: Conclusiones
Próximos pasos	5.2 Próximos pasos

## CAPÍTULO 5

# Conclusiones

## 5.1 Resultados obtenidos

El resultado principal del trabajo es el prototipo funcionando, cumpliendo los 26 requerimientos listados en el Capítulo 2: Introducción específica, específicamente en la tabla 2.1 de la subsección 2.1 Requisitos. No solo se tiene el prototipo sino el código base documentado. Se pueden resumir los resultados en la siguiente lista:

- Determinar si hay personas en un ambiente y controlar el consumo de un aparato de consumo eléctrico en base a eso.
- Definir rangos horarios de funcionamiento activo para el dispositivo, el cual mantiene registro del horario en tiempo real y actualiza su estado.
- Implementar código no bloqueante para la actualización de los subsistemas, de manera que no interfieran en su performance.
- Desarrollar un sistema completo con módulos funcionando en simultáneo, de manera progresiva y modular y con etapas aisladas de desarrollo y testeo.

Más allá del producto terminado, a lo largo del desarrollo se ganó experiencia con conceptos de máquinas de estado, control de interrupciones, reloj interno de tiempo real, utilización de un relay, comunicación serie y Bluetooth. También experiencia con la placa STM32 Nucleo-144 y su microcontrolador ARM Cortex-M4 de 32 bits.

## 5.2 Próximos pasos

A futuro se propone escalar el trabajo para funcionar en un hogar completo, con varios ambientes y distintos sensores y equipos de consumo en cada ambiente. La lógica principal ya estando implementada, sería cuestión de ajustar el manejo de los drivers por un lado y por otro lado la interfaz con el usuario.

Respecto a la interfaz con el usuario, teniendo distintos equipos y ambientes, sería de gran utilidad el desarrollo de una aplicación para llevar mejor organizada la configuración de cada ambiente.

A corto plazo se pueden implementar también mejoras, especialmente agregando validaciones de los inputs del usuario. A fines de proteger el funcionamiento del sistema se podría verificar que las fechas y horas ingresadas son válidas. También se propone agregar la opción de configurar horarios de funcionamiento activos no solo dentro de un día calendario sino a través de la semana.

Partiendo de la base ya implementada, se puede dirigir el trabajo en las direcciones propuestas aprovechando el desarrollo realizado.

## Bibliografía

- [1] Agustín de Vedia (Junio 2023). Repositorio del trabajo *E-Save Bluetooth* [En línea]. Disponible en: [https://github.com/deve023/E-Save\\_Bluetooth](https://github.com/deve023/E-Save_Bluetooth)
- [2] Ariel Lutenberg, Pablo Gomez y Eric Pernia (2022). A Beginner's Guide to Designing Embedded System Applications on Arm Cortex-M Microcontrollers.
- [3] Ariel Lutenberg, Pablo Gomez y Eric Pernia (2022). Repositorio con ejemplos [En línea]. Disponible en: <https://github.com/armBookCodeExamples/Directory>
- [4] Mbed. Página web de Mbed. [En línea]. Disponible en: <https://os.mbed.com>
- [5] Mbed. Especificaciones de la placa NUCLEO-F429ZI. [En línea]. Disponible en: <https://os.mbed.com/platforms/ST-NUCLEO-F429ZI/>
- [6] ST. Hoja de datos del microcontrolador Arm STM32F429ZI. [En línea]. Disponible en: <https://www.st.com/en/microcontrollers-microprocessors/stm32f429zi.html>
- [7] Agustín de Vedia. Demo del E-Save Bluetooth (Junio 2023). [En línea]. Disponible en: [https://www.youtube.com/watch?v=fHSL6gb9g30&ab\\_channel=Agust%C3%ADnMarianoDeVedia](https://www.youtube.com/watch?v=fHSL6gb9g30&ab_channel=Agust%C3%ADnMarianoDeVedia)
- [8] Agustín de Vedia. Definición de Requisitos y Casos de Uso (Junio, 2023). [En línea]. Disponible en: [https://docs.google.com/document/d/1u4NkyVK6ZjpZ\\_S7S0y2ql65oQEIMmXFwW9pk4DBa8Yg/edit#heading=h.moaftktcy3q9](https://docs.google.com/document/d/1u4NkyVK6ZjpZ_S7S0y2ql65oQEIMmXFwW9pk4DBa8Yg/edit#heading=h.moaftktcy3q9)
- [9] Handson Tecnology. 1 Channel 5 V Relay Module User Guide. [En línea]. Disponible en: <https://handsontec.com/dataspecs/relay/1Ch-relay.pdf>
- [10] AllDatasheet. HM-10 Bluetooth 4.0 BLE Module Datasheet V303. [En línea]. Disponible en: <https://pdf1.alldatasheet.com/datasheet-pdf/view/1179058/ETC1/HM-10.html>
- [11] AllDatasheet. HC-SR501 PIR Motion Detector Datasheet. [En línea]. Disponible en: <https://pdf1.alldatasheet.es/datasheet-pdf/view/1131987/ETC2/HC-SR501.html>