



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**Sistema Domótico Inteligente**



Presentado por David Colmenero Guerra  
en Universidad de Burgos — 5 de enero  
de 2021

Tutor: Álvar Arnaiz González

Tutor: Alejandro Merino Gómez







UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



D. Álvar Arnaiz-González, profesor del departamento de Ingeniería Informática del área de Lenguajes y Sistemas Informáticos y D. Alejandro Merino Gómez, profesor del departamento de Ingeniería Electromecánica del área de Ingeniería de Sistemas y Automática,

Exponen:

Que el alumno D. David Colmenero Guerra, con DNI 02287122W, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección de quienes suscriben, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 5 de enero de 2021

Vº. Bº. del Tutor:

D. Arnaiz-González, Álvar

Vº. Bº. del co-tutor:

D. Merino Gómez, Alejandro





## Resumen

El proyecto pretende integrar diversas tecnologías para confeccionar una solución domótica generalista y de bajo costo. Es decir, se pretende publicar un producto final que pueda ser utilizado por quien lo desee para confeccionar su propio sistema domótico básico dentro de su domicilio y, para ello, se ha desarrollado un software así como vídeos explicativos para su correcta comprensión.

El sistema utiliza una placa Raspberry Pi con un Sistema Operativo propio con licencia GNU, para obtener información de diferentes APIS y páginas web, como pueden ser la hora a la que amanece, anocchece, la temperatura por horas, velocidad del viento, etc; permitiendo al sistema tomar decisiones conforme a estos datos y controlar diferentes dispositivos eléctricos mediante la activación de relés. Para añadir usabilidad y facilitar la capacidad de transmisión de información del sistema, se desarrollará un Bot en Telegram con el que poder interactuar.

En el proyecto se incluye la documentación que se debe consultar antes de realizar cualquier tipo de instalación de telecomunicaciones o eléctrica en el ámbito doméstico pese a que, finalmente, el proyecto se basa en el REBT<sup>1</sup>. Esta documentación es la necesaria para implementar este proyecto de forma profesional por una empresa.

## Descriptores

Android, Raspberry Pi, GPIO, REBT, ICT, Autónomo, Sistema Domótico, Linux, Bot, Telegram, energético, Python, relé, Bash scripting...

---

<sup>1</sup>Reglamento Electrotécnico de Baja Tensión

## Abstract

The project aims to integrate various technologies to make a generalist and low-cost home automation solution. That is to say, it is intended to publish a final product that can be used by whoever wishes to make their own basic home automation system within their home and, for this, a fully commented code has been developed as well as explanatory videos for their correct understanding.

The system uses a Raspberry Pi board with its own operating system with a GNU license, to obtain information from different APIs and web pages, such as the time of sunrise, sunset, hourly temperature, wind speed, etc; allowing the system to make decisions based on this data and control different electrical devices by activating relays. To add usability and facilitate the information transmission capacity of the system, a Telegram Bot will be developed with which to interact.

The project includes the documentation that must be consulted before carrying out any type of telecommunications or electrical installation in the domestic sphere, despite the fact that, finally, the project is based on the REBT<sup>2</sup>. This documentation is necessary to implement this project professionally by a company.

## Descriptors

Android, Raspberry Pi, GPIO, REBT, ICT, Standalone, Domotic System, Linux, Bot, Telegram, energetic, Python, relay, Bash scripting

...

---

<sup>2</sup>Low Voltage Electrotechnical Regulation

---

# Índice general

---

<b>Índice general</b>	<b>III</b>
<b>Índice de figuras</b>	<b>v</b>
<b>Índice de tablas</b>	<b>vi</b>
<b>Introducción</b>	<b>1</b>
<b>Objetivos del proyecto</b>	<b>3</b>
2.1. Objetivos generales . . . . .	3
2.2. Objetivos personales . . . . .	4
<b>Conceptos teóricos</b>	<b>5</b>
3.1. Domótica . . . . .	5
3.2. SOC . . . . .	6
3.3. GPIO . . . . .	6
3.4. API . . . . .	6
3.5. Aplicación de mensajería multiplataforma . . . . .	7
3.6. Json . . . . .	8
3.7. RETB . . . . .	9
3.8. Normativa de ICT . . . . .	10
3.9. Cableado estructurado . . . . .	10
3.10. WiFi . . . . .	10
3.11. Tipos de cableado de datos . . . . .	11
<b>Técnicas, herramientas y componentes</b>	<b>13</b>
4.1. Entorno Software . . . . .	13

4.2. Control de datos . . . . .	14
4.3. Técnicas manuales . . . . .	15
4.4. Metodologías . . . . .	16
4.5. Entorno de desarrollo del Proyecto . . . . .	16
4.6. Entorno físico . . . . .	18
4.7. Plataforma de interacción . . . . .	25
<b>Aspectos relevantes del desarrollo del proyecto</b>	<b>27</b>
5.1. Motivación del proyecto . . . . .	27
5.2. Formación necesaria . . . . .	27
5.3. Metodología . . . . .	28
5.4. Desarrollo del proyecto . . . . .	29
<b>Trabajos relacionados</b>	<b>51</b>
6.1. Comparativa con otros proyectos . . . . .	51
6.2. Fortalezas y debilidades este proyecto . . . . .	52
<b>Conclusiones y Líneas de trabajo futuras</b>	<b>55</b>
7.1. Conclusiones . . . . .	55
7.2. Líneas de trabajo futuras . . . . .	57
<b>Bibliografía</b>	<b>59</b>

---

# Índice de figuras

---

3.1.	Componentes Raspberry Pi . . . . .	7
3.2.	Salida comando gpio readall . . . . .	8
3.3.	Funcionamiento Bot . . . . .	9
4.4.	Dibujo de una placa Arduino. . . . .	21
4.5.	Estructura interna de un relé. . . . .	24
4.6.	Diagrama real de los relés utilizados en el proyecto. . . . .	24
4.7.	Imagen de una placa «protoboard». . . . .	25
5.8.	Proceso de obtención de datos, programación de tareas y ejecución. . . . .	30
5.9.	Diagrama físico. . . . .	35
5.10.	Disposición interna cables UTP, FTP y STP. . . . .	36
5.11.	Caja de derivación secundaria instalada. . . . .	37
5.12.	Caja de derivación principal instalada. . . . .	37
5.13.	Instalación domótica y caja de derivación principal. . . . .	38
5.14.	Protoboard conectada. . . . .	39
5.15.	Conectorizado final de las líneas GPIO. . . . .	40
5.16.	Raspberry Pi, relé y pulsador. . . . .	41
5.17.	Raspberry y relé conectados. . . . .	42
5.18.	Funcionamiento interno de un pulsador para persianas. . . . .	42
5.19.	Plano de tablero de pruebas. . . . .	43
5.20.	Comunicación del bot. . . . .	45

---

## **Índice de tablas**

---

4.1. Comparativa Modelos Raspberry Pi . . . . .	20
4.2. Comparativa Arduinos . . . . .	22
6.3. Comparativa de las características de los proyectos. . . . .	53

---

# Introducción

---

El concepto de domótica se acuñó para poder denominar a aquellos sistemas que disponen de la capacidad de automatizar elementos de una vivienda aportando confort, seguridad, mejoras energéticas, etc.

El término procede de la unión de dos palabras:

- Domo, procedente del griego «domus», que significa casa, vivienda.
- Por otra parte, «tica» procede de automática, cuyo significado es que dispone de la capacidad para realizar tareas por sí solo.

Formando una palabra cuyo significado aúna los términos de casa y automático.

Pese a conformarse el término de domótica en el año 1984, ésta aún es una gran desconocida, aunque se van introduciendo pequeños elementos automatizables como pueden ser las famosas bombillas que podemos encender o apagar desde diferentes plataformas.

Al carecer de movilidad desde la llegada de la pandemia de la COVID19 nos vemos en la necesidad de que nuestras viviendas dispongan de algún elemento de seguridad a un precio razonable, como puede ser un sistema que simule nuestra presencia en la vivienda para intentar evitar posibles percances, aumentando la sensación de confort.

Hay quien opta por opciones tradicionales de seguridad como el blindaje del domicilio para impedir el acceso o contratar a una empresa externa para que monitorice el domicilio. Los sistemas domóticos que desarrollaremos pretenden ser elementos complementarios.

Nuestro simulador de presencia funcionará de forma autónoma interactuando con persianas y luces desde una máquina Raspberry Pi mediante relés. De esta forma la vivienda parece estar ocupada de forma que ahuyentamos a potenciales delincuentes. También dispondremos de un estudio diario de la temperatura con la que podremos programar la calefacción. Además, este sistema domótico es fácilmente escalable con sistemas de acceso a la vivienda, telefonía IP, música, calefacción, telefonillo IP, etc.

Por otro lado, el que las persianas estén automatizadas generará un evidente ahorro energético, tanto en invierno como en verano, al hacer de pantalla térmica exterior.

En resumen, el proyecto se sitúa en un campo que cubre un conjunto de necesidades generales dentro de los domicilios, a un bajo coste y con relativa sencillez a la hora de implantarlo, lo cual hace que pueda llegar a un gran número de hogares.

---

# **Objetivos del proyecto**

---

Con este proyecto se pretende crear un sistema domótico automatizado que nos permita aumentar la seguridad y la sensación de confort y bienestar dentro de nuestros domicilios.

Para ello debemos alcanzar algunos objetivos funcionales mínimos.

## **2.1. Objetivos generales**

- El sistema domótico funcionará de forma autónoma para que no interfiera en la vida diaria del inquilino y consiga facilitarle el día a día.
- El sistema domótico debe ser capaz de extraer información de Internet ya sea vía API o web scrapping.
- El sistema domótico debe poder conectarse a distintas instalaciones para actuar sobre éstas de una forma parametrizada.
- El usuario podrá interactuar con la máquina cuando lo deseé.
- La instalación se podrá realizar con materiales de fácil acceso.
- Poder controlar elementos eléctricos desde la interfaz GPIO (General Purpose Input/Output, que significa Entrada/Salida de Propósito General).
- Debe ser un proyecto de bajo coste y asequible para que pueda llegar al mayor número de viviendas posible aumentando el beneficio social.

- Se pretende conseguir también un notable ahorro energético real que repercuta en el bolsillo de quien instale el sistema, además de ayudar a combatir el cambio climático consumiendo de una manera autónoma y responsable conforme a los parámetros del domicilio haciendo de éste un entorno más eficiente. Por ello, podremos controlar el encendido de la calefacción.

## 2.2. Objetivos personales

- Poder aportar un dispositivo útil a la sociedad.
- Comprender la composición de un sistema domótico y aplicarlo.
- Desarrollar un sistema domótico con cierta complejidad y autonomía más allá de subir o bajar persianas o encender y apagar luces de forma programada o manual.
- Obtener conocimientos sobre trabajo con json desde Python.
- Aprender a controlar elementos eléctricos desde una Raspberry Pi.
- Aprender a utilizar una Raspberry Pi para fines domóticos utilizando GPIO.
- Poner en práctica conocimientos de cableado estructurado y REBT.
- Profundizar mis conocimientos sobre Linux.
- Aprender a utilizar el procesador de textos L<sup>A</sup>T<sub>E</sub>X.

---

# Conceptos teóricos

---

Este punto nace ante la necesidad de enmarcar el proyecto dentro de las tecnologías y elementos que utilizaremos durante todo el proyecto y que no tienen por qué conocerse.

El término ‘domótica’ es el pilar principal del proyecto y, por ello, comenzaré explicando lo que es y cómo lo enfocaremos:

## 3.1. Domótica

La domótica podemos definirla como aquel conjunto de elementos capaces de automatizar una vivienda aportando un beneficio. En nuestro caso, el sistema domótico deberá controlar luces, persianas y calefacción permitiendo un aumento del confort y la seguridad, además de permitir un consumo eficiente de recursos a la hora de climatizar la vivienda.

*The home automation system can be applied to many areas including home security, lighting control, flame detection, smart heating, motion sensor and door control to provides its homeowner's comfort, security, energy efficiency (low operating costs) and convenience at all times. The Internet of Things (IoT) is anticipated to enable a variety of smart home services in which each service provides a set of home automation solutions. This proposed study consists of developing an automated home monitoring using Raspberry Pi that provides a customizable and cost-efficient platform for a smart home. [15]*

## 3.2. SOC

Un SOC, que proviene del inglés «system on a chip» y significa «sistema en chip», hace referencia a una tecnología concreta que facilita que en una misma placa se integren los módulos necesarios para componer un sistema informático completo. Como ejemplos, podemos nombrar las placas Raspberry Pi o Arduino, entre otras.

## 3.3. GPIO

Éstos son unos puertos de entrada y salida conformados en forma de pines que están albergados en las placas Raspberry Pi [11], al igual que en todos los microcontroladores [12]. Los puertos GPIO, de por sí, únicamente intercambian información entre dispositivos en forma de señales digitales, pero sin una funcionalidad específica. En el presente proyecto, gracias a ellos, enviaremos órdenes a otros dispositivos externos para que realicen las tareas que les designemos, como controlar unos relés para conseguir la acción final de subir o bajar una persiana.

Existen tres tipos de puertos GPIO en Raspberry Pi

Podemos ver los GPIO de la Raspberry Pi en la imagen 3.1<sup>3</sup>, y también, la imagen 3.2 del comando «gpio readall» dentro de RaspberryPi OS, donde podemos ver las numeraciones física, BCM y GPIO de los pines de la placa.

## 3.4. API

Es el acrónimo de ‘Application Programming Interfaces’ que, traducido al castellano significa ‘Interfaz de Programación de Aplicaciones’. Estas interfaces nos sirven para realizar desarrollos. En nuestro caso utilizamos la información obtenida de Latitud y Longitud así como las temperaturas del día siguiente. En el proyecto, accedemos a una URL como esta <http://ip-api.com/json/?fields=country,regionName,city,lat,lon,isp,query>, obteniendo los valores de país, región, ciudad, latitud, longitud, ISP y dirección IP. Estos valores, podemos recogerlos directamente desde Python en formato json [10], como podemos ver en el apartado 4.2.

Otro ejemplo de API podemos verlo en Telegram, ya que utilizaremos su API orientada a crear bots para poder interactuar con nuestro sistema domótico. Podemos ver como funciona la API de Telegram con un bot

---

<sup>3</sup>Imagen original de <https://raspberryparatorpes.net/>

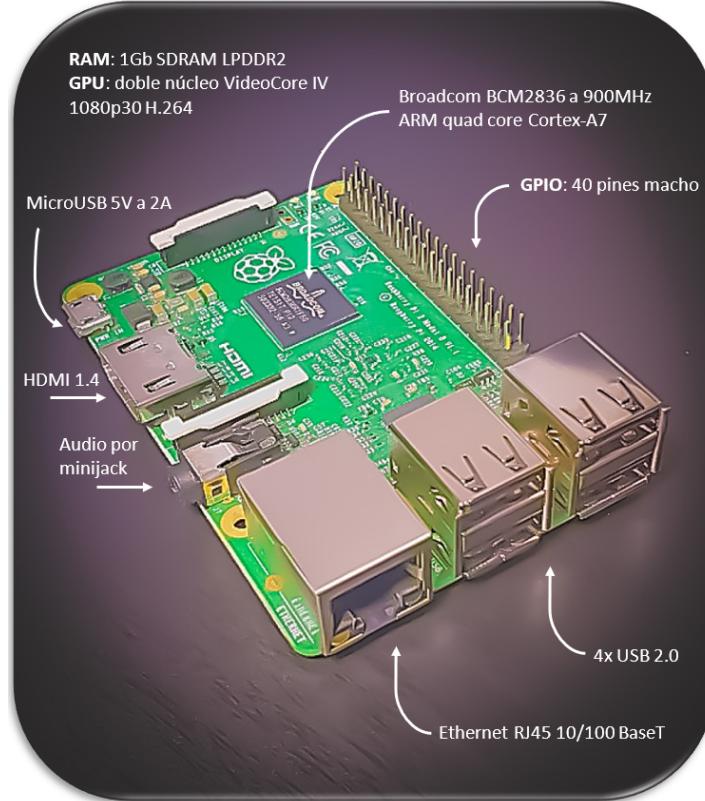


Figura 3.1: Componentes Raspberry Pi.

programado por nosotros dependiendo de una Raspberry Pi en la imagen 3.3, que es la misma arquitectura que se ha utilizado en el proyecto.

### 3.5. Aplicación de mensajería multiplataforma

Una aplicación de mensajería multiplataforma, es un software que nos permite enviar mensajes entre diferentes nodos o equipos, que utilizan este software, desde múltiples plataformas o sistemas. Telegram es una de éstas aplicaciones de mensajería multiplataforma. Esta plataforma dispone de APIs a disposición de los usuarios que éstos pueden programar y poner a su propia disposición. Podemos visitar la página web de la aplicación de mensajería en <https://telegram.org/>.

For further information, please refer to <a href="https://pinout.xyz/">https://pinout.xyz/</a>													
pi@raspberrypi: ~ \$ gpio readall													
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM			
		3.3v			1	2		5v					
2	8	SDA.1	IN	1	3	4		5v					
3	9	SCL.1	IN	1	5	6		0v					
4	7	GPIO. 7	IN	1	7	8	1	ALTO	TxD	15	14		
		0v			9	10	1	ALTO	RxD	16	15		
17	0	GPIO. 0	IN	1	11	12	0	IN	GPIO. 1	1	18		
27	2	GPIO. 2	IN	1	13	14		0v					
22	3	GPIO. 3	IN	1	15	16	1	IN	GPIO. 4	4	23		
		3.3v			17	18	1	IN	GPIO. 5	5	24		
10	12	MOSI	IN	0	19	20		0v					
9	13	MISO	IN	0	21	22	0	IN	GPIO. 6	6	25		
11	14	SCLK	IN	0	23	24	1	IN	CEO	10	8		
		0v			25	26	1	IN	CE1	11	7		
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31	1		
5	21	GPIO.21	IN	1	29	30		0v					
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26	12		
13	23	GPIO.23	IN	1	33	34		0v					
19	24	GPIO.24	IN	1	35	36	0	IN	GPIO.27	27	16		
26	25	GPIO.25	IN	0	37	38	0	IN	GPIO.28	28	20		
		0v			39	40	0	IN	GPIO.29	29	21		

Figura 3.2: Salida comando gpio readall.

## Bot de Telegram

Un bot, es un programa que permite simular una conversación en ciertos aspectos. Uno de estos aspectos puede ser enviarle órdenes para que las ejecute o nos brinde información de algún tipo. Como ejemplo, podemos ver plasmados ambos ejemplos en el bot que se ha desarrollado para el presente proyecto: podemos solicitarle al bot que ejecute acciones como el recopilado de datos o el envío de órdenes a los periféricos; y también que nos envíe información como puede ser una tabla informativa con las temperaturas por horas del día siguiente.

## 3.6. Json

Json [10], es el acrónimo en inglés de ‘JavaScript Object Notation’ y es un formato específico del que nos servimos para almacenar la información de forma estructurada mediante etiquetas «key» y etiquetas «value» del siguiente modo:

```

0 {
1   "country": "Spain",
2   "regionName": "Madrid",
3   "city": "Getafe"
4 }
```

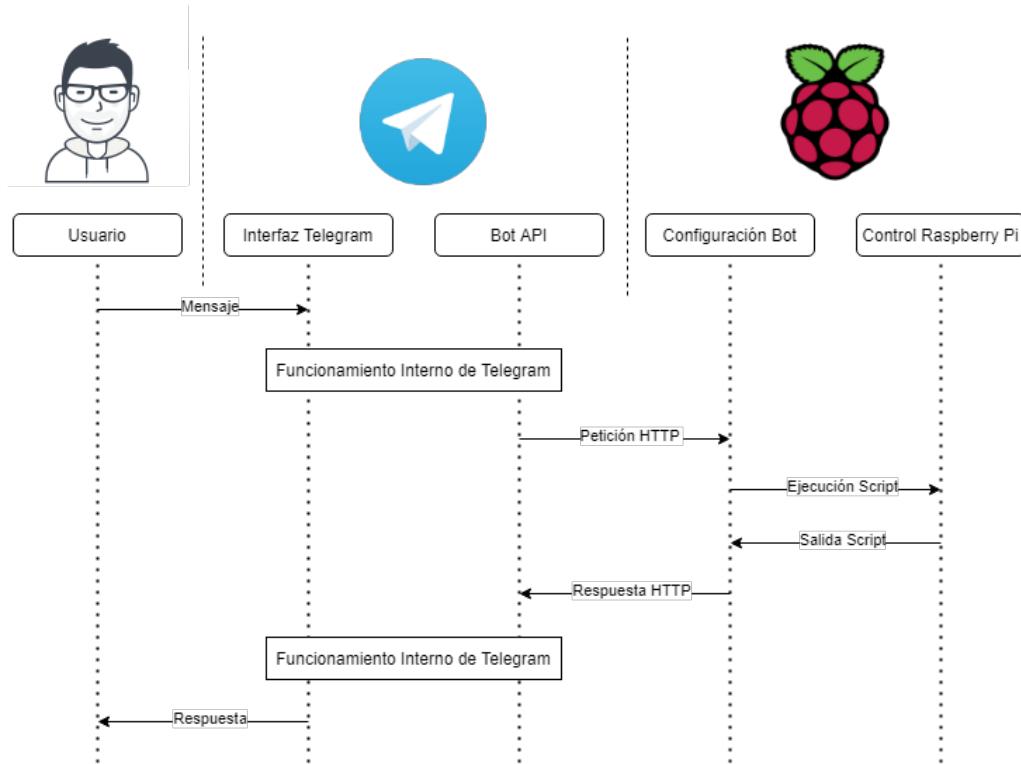


Figura 3.3: Funcionamiento Bot de Telegram con Raspberry Pi.

En ella, podemos ver que las etiquetas «key» son las que están a la izquierda de los dos puntos y las etiquetas de la derecha son las «value». Tenemos un ejemplo real en el punto [5.4](#) del presente proyecto.

## 3.7. RETB

Es el acrónimo de Reglamento electrotécnico para baja tensión y en él se recoge la normativa eléctrica aplicable en domicilios. Esta norma acaba de ser actualizada y podemos disponer de la información en páginas oficiales como puede ser el BOE [\[6\]](#). Del documento ICT-BT-21 [\[3\]](#) podemos extraer información para realizar las instalaciones eléctricas de nuestro sistema domótico como el número máximo de cables a introducir por un tubo eléctrico.

### 3.8. Normativa de ICT

Como figura en el *BOE 143, de 16 de junio de 2011, El Reglamento regulador de las infraestructuras comunes de telecomunicaciones para el acceso a los servicios de telecomunicación en el interior de las edificaciones, aprobado por el Real Decreto 346/2011, de 11 de marzo*: Debemos regirnos por esta normativa a la hora de hacer cualquier instalación de comunicaciones nueva dentro de domicilios. Podemos informarnos y ampliar información en la publicación en la sección de ICT en el BOE [4].

Por otro lado, disponemos de guías para instaladores con dibujos y tablas que facilitan la comprensión, como puede ser la documentación que publica Televés [7]. De este punto, obtendremos la norma para introducir cableado ICT [4] conforme a norma.

Tras hacer un estudio en mi domicilio, no necesitaré utilizar la normativa de ICT [4] porque toda la instalación se realizará mediante canales eléctricos, pero está bien conocer la norma para, en caso de necesitarla, poder hacer uso de ella correctamente.

### 3.9. Cableado estructurado

El establecimiento de un sistema de cableado estructurado consiste en la organización de los cables en un recinto conforme a una norma y constituye el nivel básico de cualquier red de comunicaciones. Al contar y cumplir con este estándar nos damos cuenta de que tendremos instalaciones limpias, uniformes, seguras y escalables, facilitando la supervisión, el mantenimiento y posibles migraciones de tecnologías. Un sistema de cableado genérico dispone de tres subsistemas, Troncal, de Edificio y Horizontal. En nuestro proyecto únicamente trataremos con el subsistema horizontal.

En este proyecto no contaremos con un gran número de cables, pero no está de más realizar una instalación lo más correctamente posible con unas normas de referencia

### 3.10. WiFi

Es una tecnología de comunicaciones de forma inalámbrica o «Wireless». WiFi es el acrónimo traducido de «Fidelidad Inalámbrica» que procede del inglés «Wireless Fidelity». Estas tecnologías inalámbricas se rigen por la nor-

ma [IEEE 802.11](#) [13]. En la web oficial del organismo podremos comprobar qué estándares dentro del 802.11 están vigentes y cuáles no.

### 3.11. Tipos de cableado de datos

Es un tipo de cableado de datos que se compone de 4 pares de cables sin apantallar que están albergados dentro de una camisa de PVC.

Existen diferentes tipos de cables de datos: UTP, STP, FTP:

- Los cables **UTP** (del inglés «Unshielded Twisted Pair» o «Par trenzado no apantallado») no disponen de protección ante interferencias electromagnéticas. Ver imagen [5.10](#).
- Los cables **FTP**(del inglés «Foiled Twisted Pair» o «Par trenzado con pantalla global») disponen de una pantalla global contra interferencias electromagnéticas dentro de la camisa de PVC que recoge los 4 pares destinados a transmisión de datos. Ver imagen [5.10](#).
- Los cables **STP**(del inglés «Shielded Twisted Pair» o «Par trenzado apantallado») disponen de una pantalla contra interferencias electromagnéticas por cada par de cables pero, además, también cuentan con una malla metálica exterior. Ver imagen [5.10](#).

En nuestro caso utilizaremos UTP puesto que no necesitamos un apantallamiento ya que no transmitiremos datos y tampoco tendremos un alto grado de interferencias.



---

## **Técnicas, herramientas y componentes**

---

Durante el proyecto se han utilizado diferentes tecnologías, herramientas y componentes que son imprescindibles y necesitan conocerse antes de continuar con el proyecto. Para optar por éstos y no por otros, se ha realizado una valoración que queda plasmada en este apartado a modo de justificación.

### **4.1. Entorno Software**

#### **Raspbian (Distribución Linux)**

Como he comentado anteriormente, pretendo correr una distribución Linux en nuestro microPC. Las placas Raspberry Pi disponen de unas distribuciones de Linux desarrolladas expresamente para su hardware desde la Raspberry Pi Foundation. De esta manera conseguimos que el entorno esté diseñado para el hardware donde será ejecutado incluyendo, además, utilidades preinstaladas para explotarlas más fácil y eficientemente. Una de estas distribuciones optimizadas y orientadas a estas placas es Raspbian [11] o Raspberry Pi OS, que incluye software orientado a la educación, programación y otras de uso general. Algunas de estas aplicaciones son Python [5] (Lenguaje de programación que pretende que se desarrolle código de una forma sencilla, rápida, poco costosa y legible), Scratch [9] (Simulador amigable para aprender programación) o Java[16] (Lenguaje de programación multiplataforma que utiliza una máquina virtual transparente para el usuario para ejecutarse), entre otros.

## Entorno de desarrollo Bash

- **Herramientas valoradas:** [Vi](#), [Vim](#), [Nano](#).
- **Herramienta elegida:** [Nano](#).

Nuestro Sistema Operativo Raspbian[11], al ser una distribución de Linux [14], dispone de intérpretes de líneas de comandos, procesadores de texto plano y editores de texto integrados. Nano es un editor de textos básico que facilita la interacción con el Sistema Operativo, pese a que Vi o Vim son mucho más potentes.

## Entorno de desarrollo Python

- **Herramientas valoradas:** [Jetbrains PyCharm](#) y [Jupyter Notebook](#).
- **Herramienta elegida:** [Jupyter Notebook](#).

Jupyter Notebook es un entorno de desarrollo interactivo y open source, basado en cuadernos que estructuran el código pudiendo ejecutarlo todo o parte. Dispone de una interfaz limpia, ligera e interactiva que nos permite programar en 40 lenguajes, incluyendo Python [5].

## 4.2. Control de datos

### Web Scraping

Es una técnica utilizada para extraer información de una página web utilizando las etiquetas de que dispone el propio lenguaje interpretado de HTML (del inglés «HyperText Markup Language» o lenguaje de marcas de hipertexto) para organizar elementos dentro de una página web, de forma que se introduce dentro de una etiqueta y subetiquetas hasta llegar al contenido del elemento requerido. Podemos entenderlo como si fueran contenedores lógicos configurables. En nuestro caso, podremos utilizarlo desde Python [5] sirviéndonos de la librería «beautifulsoup» siempre que necesitemos obtener información de una página web.

### APIS

- **API situación geográfica**

En primer lugar estuve haciendo pruebas con la API de [www.ifconfig.me/ip](http://www.ifconfig.me/ip) que devuelve la provincia en la que se encuentra tu IP pública pero quería una información más precisa ya que no tendremos la misma temperatura en El Escorial que en Aranjuez. Por ello, opté por <http://ip-api.com> que sí obtiene correctamente la ciudad desde la que nos conectamos.

#### ■ API Tiempo

Al principio probé la API de [www.weatherapi.com](http://www.weatherapi.com) pero me entregaba únicamente la hora de salida y puesta del sol, lo cual es correcto para el control básico de las persianas. El objetivo era llevar el proyecto más allá, obteniendo una previsión de las temperaturas para el día siguiente, pudiendo trabajar con ésta haciendo gráficos y poder decidir si encenderemos la calefacción o no.

Como solución, opté por probar con la API de [www.climacell.co](http://www.climacell.co) que además, aseguran que es un 60 % más fiable que otras APIS ya que obtiene información de teléfonos móviles, cámaras y otros servicios online.

#### json

La librería json [10] para Python [5] y Python v3 nos permite, entre otros, parsear el código json [10] de archivos mediante la estructura de `key:value`. En nuestro caso, tras obtener información de las APIs trataremos dicha información como json [10] gracias a su librería para Python [5].

## 4.3. Técnicas manuales

### Tirada de cable con guía pasacables

El procedimiento a seguir en la tirada de cable, con guía pasacables, es el siguiente:

1. Se abren las tapas de dos cajas de derivación próximas.
2. Se introduce una guía pasacables (herramienta plástica con la forma de cuerda para introducir cables por canalizaciones) por el extremo de uno de los tubos dentro de la caja hasta llegar al otro extremo.
3. Se asegura el cable a uno de los extremos de la guía pasacables.
4. Se tira del otro extremo de la guía pasacables hasta conseguir sacar el cable por éste.

## 4.4. Metodologías

### Scrum

Scrum [20] es un marco de trabajo para el desarrollo de software mediante la metodología ágil en el que se busca realizar un trabajo colaborativo de desarrollo incremental. Se aplica una metodología basada en milestones y sprints de forma iterativa.

### Modularidad

Utilizaremos la modularidad para subdividir la aplicación en pequeños subprogramas que tienen una pequeña funcionalidad. De esta manera es más fácil detectar errores y escalar el proyecto. Además, teniendo clara la entrada y salida de cada uno de los módulos se pueden lanzar pruebas a cada uno de los módulos para comprobar su correcto funcionamiento mejorando, también, el mantenimiento.

## 4.5. Entorno de desarrollo del Proyecto

### Control de versiones o CVS, Concurrent Versioning System

- **Herramientas valoradas:** [Git](#), [SVN](#).
- **Herramienta elegida:** [Git](#).

Git es un software destinado al control de versiones software en el que se registran los cambios producidos en el mismo, facilitando la integración de código por parte de cualquiera de los integrantes del proyecto. La diferencia más notable entre ellos es que Git es distribuido y SVN es un sistema centralizado. Significa que Git nos permite disponer de una copia en cada uno de los equipos desde los que se trabaje haciendo un clonado del repositorio, mientras que en SVN también permite el trabajo directamente en la nube.

### Hosting del Repositorio

- **Herramientas valoradas:** [Github](#) y [Bitbucket](#).
- **Herramienta elegida:** [Github](#).

Ambas opciones de hosting de repositorios funcionan de forma similar aunque GitHub incorpora opciones como la revisión de código, Kanban, Wikis o tableros entre otros que me han hecho decantarme por esta opción. Ésta, es la plataforma principal de trabajo, que a su vez es una red social de código donde cualquiera puede contribuir en proyectos públicos y Open Source.

## Gestión del proyecto

- **Herramientas valoradas:** [MS.Teams](#), [ZenHub](#), [GitHub Projects](#), [Trello](#), [Jira](#), [Board](#), [Monday](#), [Zube](#), [Clubhouse](#).
- **Herramienta elegida:** [ZenHub](#).

Es la única solución de colaboración en equipo integrada en GitHub y nos permite planificar hojas de ruta, generar informes, gestionar con Kanban, gestión ágil del proyecto mostrando la situación del proyecto para conseguir aumentar la productividad del equipo.

## Editor del proyecto

- **Herramientas valoradas:** [Atom](#), [Visual Studio Code](#), [Sublime](#).
- **Herramienta elegida:** [Atom](#).

Es un editor de código y texto creado por GitHub integrando las funciones de Git y GitHub, lo que nos facilita trabajar en nuestro equipo y replicar los cambios en Git, GitHub y ZenHub. Además, es un software multiplataforma, licencia open source, completamente personalizable, con temas y múltiples plugins, autocompletado y fácil navegación. En este proyecto se utilizará para publicar las actualizaciones del proyecto.

## Dibujos, diagramas y planos

- **Herramientas valoradas:** [Fritzing](#), [Paint](#), [Paint3D](#), [Photoshop](#), y [Draw.io](#).
- **Herramienta elegida:** [Fritzing](#) y [Draw.io](#).

Realmente no pude decidirme por entre Fritzing y draw.io puesto que cada uno está orientado a un tipo de tarea:

Fritzing es un software con licencia Open Source [17] que nos permite realizar material electrónico fácilmente, aunque desde hace algún tiempo debemos hacer un pequeño desembolso por la descarga. En mi caso con el fin de entregar diagramas de calidad los haré con este software.

Por otro lado, Draw.io está pensado para hacer planos y dibujos que no están orientados a la electrónica, por lo que es muy útil para hacer otro tipo de diagramas.

## Procesador de textos L<sup>A</sup>T<sub>E</sub>X

- **Herramientas valoradas:** L<sup>A</sup>T<sub>E</sub>X, MS Word, Sublime, Overleaf.
- **Herramienta elegida:** L<sup>A</sup>T<sub>E</sub>X y Overleaf.

L<sup>A</sup>T<sub>E</sub>X es un editor de textos open source [17] con alta calidad tipográfica que trabaja con etiquetas permitiéndonos separar nuestro contenido del estilo del mismo. Para escribir el proyecto en L<sup>A</sup>T<sub>E</sub>X utilizaré Overleaf por su capacidad de compilado instantáneo, de forma que se puede comprobar cómo modificas la redacción en cada compilado.

## 4.6. Entorno físico

En el mercado existen diversas opciones para poder generar un proyecto como este. Las opciones a considerar son los SBC y los MicroPc.

- **SBC:** Son pequeños equipos de placa única de bajo coste. Es el grupo de componentes en el que podemos encontrar Raspberry Pi y Arduino.
- **MicroPc:** Son pequeños equipos con mayor potencia y un precio más elevado que los anteriores. En este grupo podemos encontrar soluciones como los ShuttlePC.

Entre estas dos opciones, optaremos por la primera por varios motivos. El motivo principal es el coste: es innecesario hacer mayor desembolso para implementar este sistema domótico. Otros motivos son el consumo, que es mucho menor en las placas SBC y la necesidad de una plataforma intermedia para poder controlar los relés, que también incrementaría el precio.

## Comparativa SOC

Tras determinar la utilización de un SOC se hizo una comparativa entre las dos SBC principales, Arduino y Raspberry Pi teniendo en cuenta que son dos placas muy similares.

### RaspberryPi

Para dar un enfoque muy general, podemos decir que las placas RaspberryPi [11] son microordenadores que disponen de poca potencia si las comparamos con equipos usuales, pero disponen de suficiente potencia para llevar a cabo este tipo de proyectos.

Se diseñaron en su origen por la Raspberry Pi Foundation [11] en el Reino Unido para dotar de equipos informáticos a los centros de estudios a un bajo coste, pero el proyecto ha evolucionado para poder desarrollar, además, otras muchas tareas como puede ser nuestro caso, que la utilizaremos como ‘núcleo’ de toda nuestra instalación domótica y, será donde configuremos todo el entorno domótico de la vivienda. Estas placas pueden ejecutar con agilidad distribuciones Linux [14], y desde sus éstas, podemos interactuar con sus famosos «GPIO» [12] como explicamos en el punto 3.3, ver imagen 3.1.

Podemos ver en la tabla 4.1 una comparativa de los modelos B de Raspberry Pi. A modo de resumen, únicamente he representado los modelos B en al comparativa porque el proyecto se ha realizado con un modelo B.

Modelo Raspberry Pi	Zero y Zero W	B+	2B	3B y 3B+	4B
Precio(Amazon)	29,30€	34,30€	51,59€	37,44€	29,90€
SoC	BCM2835	BCM2835	BCM2836 BCM2837 en v1.2	BCM2837 BCM2837B0	BCM2711
CPU	ARM1176JZF-S	ARM1176JZF-S	ARM Cortex-A7 ARM Cortex-A53 en v1.2	ARM Cortex-A53	ARM Cortex-A72
Cores	Single-core	Single-core	Quad-core	Quad-core	Quad-core
Velocidad	1000MHz	700MHz	900MHz	1200MHz 1400MHz	1500MHz
RAM	512MB	512MB	1GB (1024MB)	1GB (1024MB)	1GB (1024MB) 2GB (2048MB) 4GB (4096MB)
GPU	250MHz Broadcom-VideoCore IV	250MHz Broadcom-VideoCore IV	250MHz Broadcom-VideoCore IV	400MHz Broadcom-VideoCore IV	500MHz Broadcom-VideoCore VI
Conexiones	miniHDMI 1 x micro USB2 40 GPIO pins MIPI camera connector Wi-Fi Bluetooth	4x USB2 ports 10/100 Ethernet 40 GPIO pins MIPI camera connector MIPI display DS1 Vídeo compuesto (PAL y NTSC) Minijack estéreo	HDMI 4x USB2 ports 10/100 Ethernet 40 GPIO pins MIPI camera connector MIPI display DS1 Vídeo compuesto (PAL y NTSC) Minijack estéreo	HDMI 4x USB2 ports 10/100 Ethernet 40 GPIO pins MIPI camera connector MIPI display DS1 Vídeo compuesto (PAL y NTSC) Minijack estéreo Wi-Fi 2,4GHz Bluetooth 4.1 Wi-Fi 2,4/5GHz Bluetooth 4.2	2 x micro HDMI 2x USB3 ports 2x USB2 ports 10/100/1000 Eth 40 GPIO pins MIPI camera connector MIPI display DS1 Vídeo compuesto (PAL y NTSC) Minijack estéreo Wi-Fi 2,4GHz Bluetooth 5.0

Tabla 4.1: Comparativa Modelos Raspberry Pi.

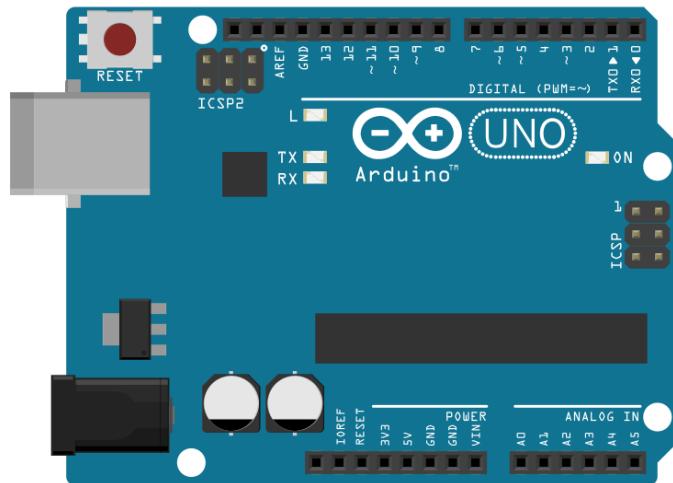


Figura 4.4: Dibujo de una placa Arduino.

## Arduino

Un SoC Arduino es una placa construida para poder detectar y controlar otros elementos, acercando y facilitando la electrónica a múltiples proyectos de diferentes disciplinas. Por ello, también se valoró realizar el proyecto con Arduino pero quizás sea una opción más electrónica que informática. Podemos ver una comparativa de algunos Arduino en la tabla 4.2.

Adjunto también la imagen 4.4 sobre un Arduino para que quede representado también en el proyecto.

Arduino	PROCESADOR	VOLTAJE OP/IN	MHz CPU	IN/OUT ANALÓGICAS	ENTRADAS/ SALIDAS DIGITALES	EEPROM	SRAM (KB)	FLASH (KB)	USB	UART
Uno	ATmega328P	5 V / 7-12 V	8MHz	6/0	14/6	1	2	32	Regular	1
Leonardo	ATmega32U4	5 V / 7-12 V	16MHz	12/0	20/7	1	2.5	32	Micro	1
101	Intel Curie	3.3 V / 7-12V	32MHz	6/0	14/4	-	24	196	Regular	-
Esploра	ATmega32U4	5 V / 7-12 V	16MHz	-	-	1	2.5	32	Micro	-
Arduino Zero	ATSAMD21G18	3.3 V / 7-12 V	48 MHz	6/1	14/10	-	32	256	2 Micro	2
Mega 2560	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	4	8	256	Regular	4

Tabla 4.2: Comparativa Arduinos

## Decisión sobre SOC

Finalmente he determinado que en nuestro proyecto tendremos el control de la instalación domótica desde una Raspberry Pi[11]. La justificación se basa en la naturaleza de ambas placas y la necesidad del proyecto: las placas Raspberry Pi están orientadas a utilizarse como un ordenador de poca potencia, mientras que un Arduino está orientado a utilizarse como un equipo electrónico.

Podemos ver una imagen ilustrativa de las placas Raspberry Pi en [3.1](#) y una de Arduino en la imagen [4.4](#).

Además, se pretende que el proyecto disponga de un Sistema Operativo sobre el que poder programar las funciones que queremos además de poder interactuar con éste de una forma natural además de facilitar el futuro mantenimiento.

Sobre las diferentes Raspberry Pi, podemos observar que prácticamente nos valdría cualquiera para el cometido que le vamos a encargar. En este caso optaré por la Raspberry Pi 2B frente a las demás porque dispongo de una placa en desuso; pero, en caso de tener que escoger una, elegiría una Raspberry Pi 3B+ por su escaso precio y capacidades superiores a la 2B. También deshecharía la 4B por su elevado precio con respecto a la seleccionada y sus problemas de sobrecalentamiento en la penúltima versión ya que al realizar la compra de la misma no dejan elegir la versión de la placa.

## Relé

Es un dispositivo electromagnético que permite, mediante un circuito primario, abrir o cerrar un circuito secundario. Dicho con otras palabras, desempeña la misma función de un interruptor, es decir, con nuestros relés, dejaremos pasar la energía, o no, a nuestros dispositivos. Podemos ver el funcionamiento interno en la imagen [4.5](#) <sup>4</sup>. También podemos ver un diagrama, en la imagen [4.6](#), de como son los relés utilizados en este proyecto con una breve aclaración sobre las conexiones que están a nuestra disposición.

---

<sup>4</sup>Imagen original de <https://commons.wikimedia.org/>

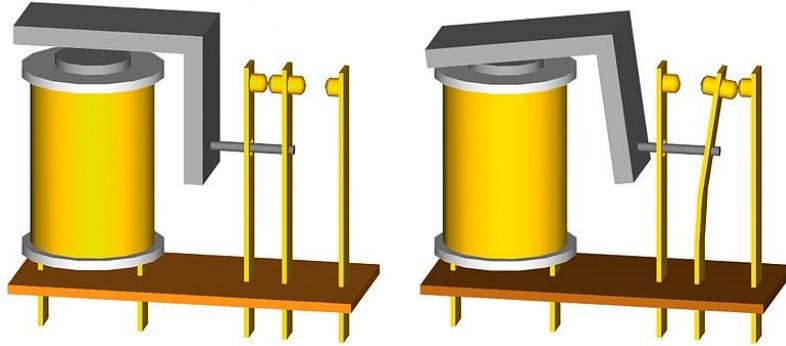


Figura 4.5: Estructura interna de un relé.

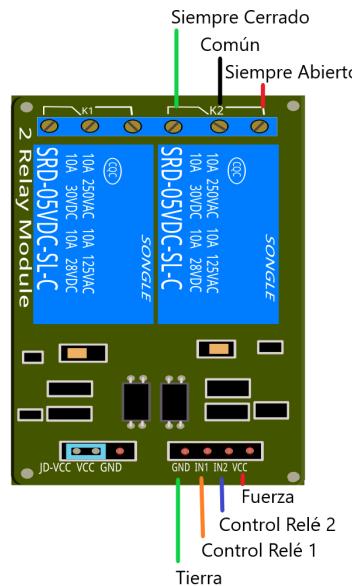


Figura 4.6: Diagrama real de los relés utilizados en el proyecto.

## Placa de Pruebas o ProtoBoard

Es un tablero electrónico para realizar pruebas. Protoboard es la agrupación de los términos ingleses “prototype board”. Esta protoboard la he configurado para poder hacer fácilmente el interconexiónado entre los cables que llegan de los relés y los que van a la Raspberry Pi, evitando posibles tirones y movimiento de cables a la hora de hacer alguna manipulación.

Éstas, disponen de tres zonas diferenciadas(Ver imagen [4.7](#)):

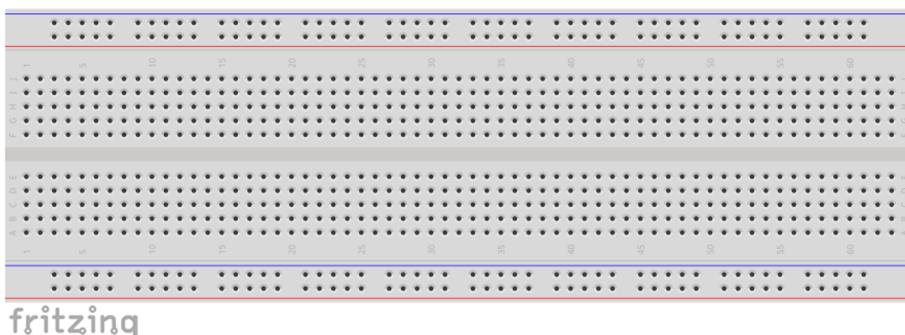


Figura 4.7: Imagen de una placa «protoboard».

- **Canal Central:** Está situada en el medio de la placa, es donde está la zona de las pistas y es donde se colocan los circuitos.
- **Buses:** Se sitúan en los extremos de la placa y disponen de dos líneas:
  - Línea roja:** Bus positivo o de voltaje.
  - Línea azul:** Bus negativo o de tierra.
- **Pistas:** Existen dos pistas en la zona central de la placa de la imagen y, conducen en sentido contrario de las líneas rojas y azul.

## Router

Es un dispositivo que nos permite interconectar diferentes redes de datos. En mi caso dispongo de un router con WiFi integrado para poder dotar a la Raspberry Pi de salida a Internet.

## 4.7. Plataforma de interacción

- **Herramientas valoradas:** [Telegram](#), [Flask](#).
- **Herramienta elegida:** [Telegram](#).

Para finalizar el proyecto con una interfaz de interacción se valoraron las opciones de desarrollar un página web ligera o una aplicación de mensajería. En primer lugar se propuso hacer una aplicación en Flask pero pareció más novedoso y potente hacer un Bot de Telegram de forma que podamos interactuar con él y enviarnos información bajo demanda. Además, se elimina

el buscar hosting y dominio para albergar los servicios propios del sistema domótico abaratando más los costes, además de incrementar la seguridad al utilizar mensajería cifrada.

---

# **Aspectos relevantes del desarrollo del proyecto**

---

En este punto se recogerán los aspectos más relevantes del desarrollo comentando en cada caso las decisiones tomadas para llegar a nuestros objetivos haciendo un resumen de la experiencia práctica del proyecto, de cómo se solucionaron los problemas encontrados en cada caso y la relevancia que tuvieron en el alcance total del proyecto.

## **5.1. Motivación del proyecto**

El proyecto se me ocurrió hace años cuando me emancipé a una vivienda con las persianas motorizadas que me parecieron que podrían hacer una función mayor si dispusieran de cierto equipamiento, y cogió fuerza con la llegada de la pandemia de la COVID19 cuando el tema surgió en distintas conversaciones en las que quien no podía acercarse a sus segundas viviendas y estaban preocupados por una posible ocupación. Tras darle vueltas a esta situación surgió la idea de crear un sistema domótico para poder ayudar a quien lo precise con este proyecto.

## **5.2. Formación necesaria**

El proyecto requirió muchas horas de búsqueda de ideas por la web, ya que no existe información fiable de cómo realizar una instalación de estas características sino que existen muchos pequeños proyectos amateur que se centran en cubrir una pequeña necesidad; siendo proyectos que no disponen de un respaldo documental detrás. Abundan las soluciones de profesionales

de otros campos que quieren probar a hacer sus propias soluciones de carácter amateur.

Por ello, para poder desarrollar el proyecto me vi en la necesidad de visitar numerosas páginas web de distinta índole para hacerme a la idea de cómo podría enfocar el proyecto. Aunque, el mayor hándicap a la hora de realizar este proyecto es la desinformación, por lo que me apoyé sobre todo en el REBT 3.7 [6] y en mis conocimientos básicos de motores fruto de formación pasada.

La información de cómo funcionan los GPIO la obtuve tras realizar el curso de: “Control de GPIO con Python en Raspberry Pi” de Programo Ergo Sum [19]. Aunque, en parte la información de la web de [bujarra.com](http://bujarra.com) [8] está desactualizada, esta publicación me ayudó a comprender cuál era el funcionamiento real de un relé y como conectarlo a los GPIO.

Otro punto importante a la hora de enfocar correctamente el proyecto fue el estudio del REBT [6], de su apartado BT-21 [3], del reglamento de ICT [4] y los estándares de comunicaciones como son el IEEE802.11 [13] y el TIA568 [2] [1]. Creo necesario confrontar estas normativas para poder realizar un proyecto a nivel profesional y documentalmente respaldado.

### 5.3. Metodología

Se eligió Scrum como la metodología global sobre la que realizar el proyecto de forma iterativa y ágil. Buscamos generar un proyecto lo más cercano a cómo sería un proyecto de este tipo en el ámbito laboral, pero salvando las distancias con un grupo de trabajo real con el que poder interactuar. Aunque he tenido reuniones semanales con los tutores no se ha realizado un seguimiento diario pero sí se cumplieron, en cualquier caso, unas reglas generales mínimas:

- Los trabajos se desarrollaron en forma de «sprints» semanales.
- Tras cada «sprint» semanal se realiza una entrega de trabajo de forma incremental.
- En cada revisión del «sprint» se disponen los trabajos a realizar durante la semana.
- Realizando una revisión semanal, el proyecto goza de flexibilidad al integrar cambios sobre trabajos pequeños de forma que el proyecto está en constante mejora.

- Sobre el Kanban se realiza la estimación de tiempos por tareas según dificultad.
- Se priorizan las tareas del proyecto según mayor valor de negocio.
- Se cambia el estado de los «issues», mediante un Kanban, según evoluciona el trabajo.
- Comprobamos como avanza el proyecto con los gráficos BurnDown de que dispone Zenhub.

Reseñar que en las fases de trabajos físicos no se ha aplicado Scrum pero se ha realizado con un criterio lo más cercano posible. Por ello, se predispuso realizar la tirada de cable y el conexionado la misma semana para que pudiera integrarse lo mejor posible en la metodología.

## 5.4. Desarrollo del proyecto

A medida que el proyecto ha ido tomando forma también se han ido implementando cambios: La idea original del proyecto era parametrizar los elementos según horas pero posteriormente se valoró e implantó la idea de tomar datos externos para poder controlar los elementos del sistema domótico. El proyecto se ha creado de forma cronológica siguiendo los pasos que se desarrollan a continuación.

El proyecto se comenzó por los scripts de extracción de datos para comprobar que se podía realizar el software que se había propuesto antes de continuar ya que es más fácil presentar cambios al principio que cuando el proyecto está avanzado. Aunque cabe destacar que se ha invertido una gran cantidad de tiempo en la búsqueda de información y normativas que desconocía, así como en la extracción y el tratamiento de datos.

Por tanto, se comenzó extrayendo datos desde Python [5] mediante web scraping 4.2 pero nos dimos cuenta de que una web es más susceptible de sufrir cambios. Por ello, recurrimos a APIS de terceros sin coste para poder realizar dicha extracción de datos, y tratándolos como json [10] extraer los datos que nos interesen.

### Extracción de datos

Los algoritmos de extracción y tratamiento de datos también han ido cambiando ya que se han ido modificando las APIS para obtener más datos y de carácter más fiable.

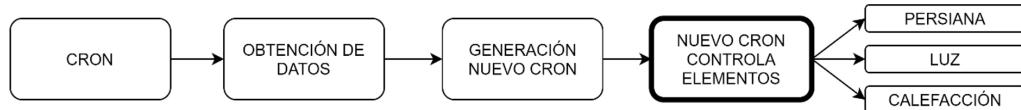


Figura 5.8: Proceso de obtención de datos, programación de tareas y ejecución.

Al comenzar a extraer datos de la API de geolocalización de [www.ifconfig.me/ip](http://www.ifconfig.me/ip) me obtenía la ubicación de la provincia, lo cual no es nada exacto ya que tendremos temperaturas diferentes según ubicación. Por ello, me vi obligado a hacer una búsqueda más a fondo para encontrar una API que se ajustase a las necesidades del proyecto. Tardé algunos días hasta que encontré <http://ip-api.com> que me entregaba más información y más fiable. Además, en el mismo intervalo, pasamos de utilizar web scraping a tratar los datos directamente como json [10], que también es un cambio importante.

En la segunda API, en el caso de [www.weatherapi.com](http://www.weatherapi.com) podía obtener únicamente los valores de salida y puesta de sol para realizar el control de las persianas pero también era insuficiente, por lo que tuve que volver a hacer una búsqueda más amplia hasta que encontré [www.climacell.co](http://www.climacell.co), que me entrega las temperaturas de las próximas horas además de otros muchos parámetros que pueden llegar a servirnos como, por ejemplo, la velocidad del viento o si será un día nublado.

Tras obtener la ubicación de la instalación y extraer los parámetros necesarios de las APIS pasamos a conformar el archivo de datos para generar la configuración de Cron (Cron, es el programador de tareas de Linux [14]). Esta fase dispone de varios pasos:

1. Desde «Cron» se llama al script de toma de datos.
2. Conformamos el nuevo archivo de «Cron».
3. El nuevo «Cron» lanza los scripts según las horas predispuestas.

De esta manera el control de los dispositivos se hará automáticamente desde «Cron» con los datos actualizados diariamente. Podemos ver un diagrama explicativo de este proceso en la imagen 5.8. Además, podemos ver a continuación un ejemplo de archivo de recopilación de datos:

Listing 5.1: Ejemplo archivo de recopilado de datos.

```

1 {
2 "Planeta" :
3     {
4         "Amanecer" : "08:37:00",
5         "Anochecer" : "17:57:00"
6     },
7
8 "temperaturas" :
9     {
10        "0" : 5.26,
11        "1" : 5.78,
12        "2" : 5.93,
13        "3" : 5.95,
14        "4" : 5.97,
15        "5" : 5.76,
16        "6" : 4.72,
17        "7" : 3.79,
18        "8" : 3.38,
19        "9" : 3.98,
20        "10" : 5.3,
21        "11" : 6.58,
22        "12" : 7.46,
23        "13" : 8.47,
24        "14" : 8.78,
25        "15" : 8.89,
26        "16" : 7.91,
27        "17" : 6.58,
28        "18" : 5.66,
29        "19" : 5.05,
30        "20" : 4.43,
31        "21" : 3.85,
32        "22" : 3.37,
33        "23" : 2.91
34    }
35 }
```

Este archivo de datos es leído siempre que queremos modificar alguno de los parámetros de automatización desde el bot, como son la temperatura o la hora más temprana a la que pueden bajar las persianas, además de cuando se regenera el Cron.

## Scripts

Utilizaremos dos tipos de scripts, Bash [14] y Python [5].

Una vez determinados los datos que vamos a necesitar en un notebook de Jupyter debemos realizar la extracción de estas líneas de código a un script Python. En este punto tuve un problema bastante sencillo de resolver: el script necesita importar librerías para hacerlo correr y necesitaba instalarlas. En principio, esto no debería ser un problema ya que sabemos que se puede instalar con Pip. El problema surge al tener instalado Python v2 y Python v3, ya que si instalas las librerías utilizando Pip no funcionarán con Python3. Así que, hay que instalarlas con Pip3 para poder utilizarlas desde Python3:

```
0
pip install paquete
```

```
1
pip3 install paquete
```

Controlaremos la Raspberry Pi y sus periféricos mediante comandos Bash y realizaremos el cocinado de datos con Python. Se ha tomado esta determinación para conseguir la mayor velocidad de procesamiento posible así como para evitar posibles problemas. Por ello, la automatización que haremos desde CRON que lanzará otros scripts será mediante bash.

Podemos ver el resultado final del Cron que genera la máquina diariamente:

```
1 # Edit this file to introduce tasks to be run by cron.
2 #
3 ##      Entry          Description   Equivalent To
4 ##      @yearly     Run once a year at midnight in the morning of January 1 0 0 1 1 *
5 ##      @monthly    Run once a month at midnight in the morning of the first of the month 0 0 1 * *
6 ##      @weekly     Run once a week at midnight in the morning of Sunday 0 0 * * 0
7 ##      @daily      Run once a day at midnight 0 0 * * *
8 ##      @hourly     Run once an hour at the beginning of the hour 0 * * * *
9 ##      @reboot     Run at startup  @reboot
10 ##
11 # For more information see the manual pages of crontab(5) and cron(8)
12 #
13 #minute (0-59),
14 #| hour (0-23),
15 #| | day of the month (1-31),
16 #| | | month of the year (1-12),
17 #| | | | day of the week (0-6 with 0=Sunday).
18 #| | | | commands
19
20 #-----
21 #Este CRON ha sido generado en el instante Sun Dec 27 13:13:27 2020
22 #-----
23 #Codigo de control Automatico de Persianas
24 #-----
```

```

25 #Subimos las persianas Modo ma ana de Lunes a Domingo
26 37 8 * * * sh /home/pi/source/TFG/scripts/control/GPIO_off.sh
27 38 8 * * * sh /home/pi/source/TFG/scripts/control/Subir.sh
28 44 8 * * * sh /home/pi/source/TFG/scripts/control/GPIO_off.sh
29
30 #Encendemos las luces
31 7 18 * * * sh /home/pi/source/TFG/scripts/LucesOn.sh
32
33 #Bajamos TODAS
34 12 18 * * * sh /home/pi/source/TFG/scripts/control/GPIO_off.sh
35 13 18 * * * sh /home/pi/source/TFG/scripts/control/Bajar.sh
36 19 18 * * * sh /home/pi/source/TFG/scripts/control/GPIO_off.sh
37
38 #Apagamos las luces
39 19 18 * * * sh /home/pi/source/TFG/scripts/LucesOn.sh
40
41 #Todos los d as se reinicia la maquina par que el demonio siempre
42 #este correcto por la ma ana
43 05 04 * * * sudo reboot
44
45 #Lanzamos el script de toma de horas
46 05 00 * * * cd /home/pi/source/TFG/scripts/auto/ && sudo sh LanzaTodoElProceso.sh
47
48 #-----
49 #Codigo de control Automatico de Calefaccion
50 #-----
51
52 0 0 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
53 0 1 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
54 0 2 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
55 0 3 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
56 0 4 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
57 0 5 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
58 0 6 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
59 0 7 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
60 0 8 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
61 0 9 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
62 0 10 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
63 0 11 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
64 0 12 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
65 0 13 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
66 0 14 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
67 0 15 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
68 0 16 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
69 0 17 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
70 0 18 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
71 0 19 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
72 0 20 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
73 0 21 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
74 0 22 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
75 0 23 * * 1-7 sh /home/pi/source/TFG/scripts/control/EncenderCaldera.sh
76
77 #Instante de grabado de datos: 2020-12-27 - 13:13:27

```

El script que lanza todo el código contiene las siguientes líneas:

Listing 5.2: Script que lanza el proceso automático completo.

```

1 #!/bin/bash
2 path=$(pwd)
3 echo $path
4
5 # Este script se ha desarrollado para lanzar todo el proceso desde CRON.
6 python3 ${path%}/1_recabaInfo.py
7 python3 ${path%}/3_cocinado.py
8 sh ${path%}/4_reescribeCron.sh

```

Este archivo es un archivo Bash porque desde Cron no he conseguido lanzar los scripts Python, sin embargo no he tenido problemas al lanzarlo desde este script Bash siempre que especifique nuevamente que se lance con python3.

Por otro lado, para controlar los relés desde los GPIO, debemos configurarlos como OUT y con valor 1 para activarlos y, con valor 0 y configuración IN para desactivarlos. Podemos ver un ejemplo a continuación:

Listing 5.3: Ejemplo activación y desactivación de relé.

```

1 gpio -g mode 17 out # Activa el rel conectado al pin 17
2 sleep 1               # Espera 1 segundo
3 gpio -g mode 17 in   # Desactiva el rel conectado al pin 17

```

Este código lo lanzaremos tanto desde el bot directamente para controlar las persianas «en vivo», como desde el Cron para el control de persianas y

## Instalación física

Quizás, la parte más compleja del proyecto ha sido la instalación física dado mi grado de desconocimiento así como el encontrar las piezas idóneas para el domicilio.

Al comienzo del proyecto, hubo una gran parte de investigación e instalación del sistema ya que quería realizar el proyecto de la forma más profesional posible. Por ello, me dispuse a buscar en el REBT [6] y en la normativa de ICT [3] para no saltarme ningún paso y que la instalación fuera totalmente legal.

Fue complejo leer tanto y comprender cómo realizar los cálculos de los cables dentro de los tubos y conseguir discernir entre la asignación real entre los diferentes canales de la vivienda. Una vez comprobado que utilizaría la

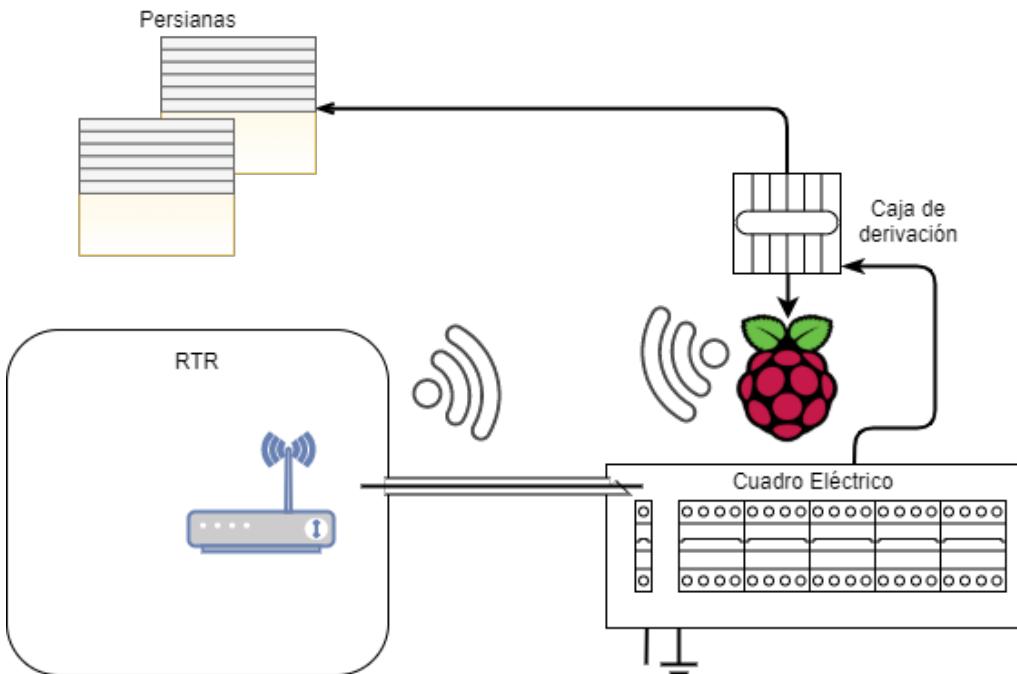


Figura 5.9: Diagrama físico.

norma actualizada de 2020 del REBT, realicé los cálculos para comprobar que podía ‘tirar’ el cable UTP por estos canales con seguridad.

Estuve valorando la opción de ‘tirar’ un cable de datos desde la primera caja de derivación después del Cuadro General de Mando y Protección o Cuadro eléctrico, hasta el RTR (Registro Terminación de Red) que es donde tengo la línea de datos más cercana, pero en único canal que comunica es de naturaleza eléctrica y no está bien mezclar tipos de instalaciones.

Buscando por Internet he podido ver que es más sencillo utilizar medios inalámbricos ya que no tendríamos que hacer tirada de cable, pero no me ofrecen la misma garantía ante un posible fallo de comunicación o interferencias, y con el cable eliminamos la posibilidad de sufrir este tipo de problemas. Sobre la categoría del cable, me he decantado por UTP cat5e (Ver imagen 5.10) porque es un cable suficientemente capaz de transmitir los 3.3V en pico de nuestra Raspberry Pi y el precio no es muy alto.

Me he documentado sobre las características eléctricas de cada uno de los hilos del cable UPT cat5e y he visto que, según la norma TIA/EIA 568, el cableado debe soportar 500VDC<sup>5</sup>, como podemos comprobar que figura

<sup>5</sup>VDC:Voltage Direct Current

en el punto A3 «Insulation resistance», conforme a los test de voltaje que deben pasar de acuerdo con IEC60512-2 que figuran en la página 55 del documento de la norma [1]. En cualquier caso, como las comunicaciones de nuestros gpio tienen un máximo de 3,3VDC y además dispone de dos salidas de 5VDC para alimentación de periféricos, necesitaremos soporte hasta 5V y nuestro cableado lo soporta sin problema.

El siguiente paso fue determinar la situación de la Raspberry Pi para que fuera lo más accesible posible ahorrando cable y que la instalación quedase lo más limpia posible, por lo que lo determiné para instalarla cerca de la primera caja de derivación eléctrica después del cuadro eléctrico general. Podemos ver un diagrama simplificado en la imagen 5.9. En ella podemos diferenciar varios elementos: Por un lado, tenemos el cuadro eléctrico y la primera caja de derivación, que ya vienen instaladas en el domicilio en la misma disposición que podemos ver en el diagrama (la caja de derivación encima del cuadro eléctrico), y en medio, me dispuesto la Raspberry Pi, para que puedan llegar los cables de datos fácilmente, ya que los hemos dispuesto por los canales eléctricos hasta los diferentes elementos a controlar (persianas, luz y calefacción).

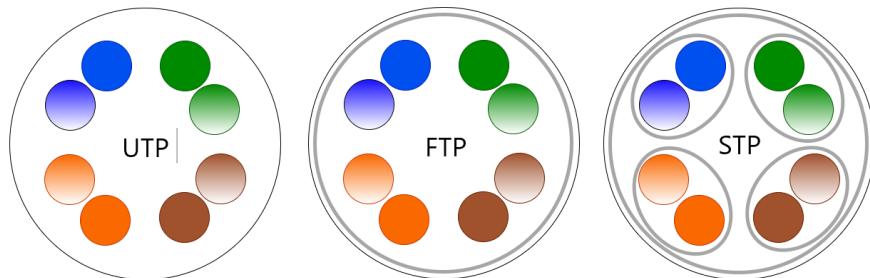


Figura 5.10: Disposición interna cables UTP, FTP y STP.

Sobre el medio de comunicación entre dispositivos electrónicos, ha sido complejo escoger entre:

- **Tecnologías valoradas:** UTP cat5e (Ver imagen 5.10), UTP cat6 (Ver imagen 5.10), Coaxial (ANSI/TIA/EIA 568.4D), ZigBee (IEEE 802.15.4), Bluetooth[13].
- **Tecnología elegida:** UTP cat5e (Ver imagen 5.10).

Cabe destacar una particularidad de mi instalación: el router está albergado dentro del RTR ya que sobra espacio dentro del mismo y mi router

está optimizado para no calentarse en espacios con poca ventilación. El canal que podemos ver entre el cuadro eléctrico general y el RTR es una toma eléctrica que, por normativa debe estar ahí y es la que alimenta algún dispositivo que deba estar dentro de esta situación.

Otro punto nuevo para mi fue el crimpado de los pines para poder trabajar con ellos. Al principio, no contaba con una crimpadora de estos pines JST-XH pero terminé haciéndome con una para que los cables quedaran correctamente fijados a los conectores.

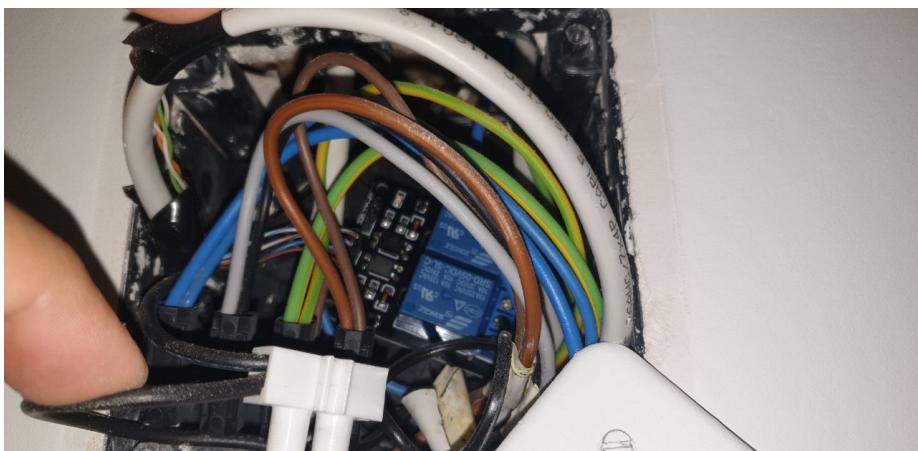


Figura 5.11: Caja de derivación secundaria instalada.



Figura 5.12: Caja de derivación principal instalada.

Tras valorar la información y la toma de decisiones hice la instalación física quedando del siguiente modo:

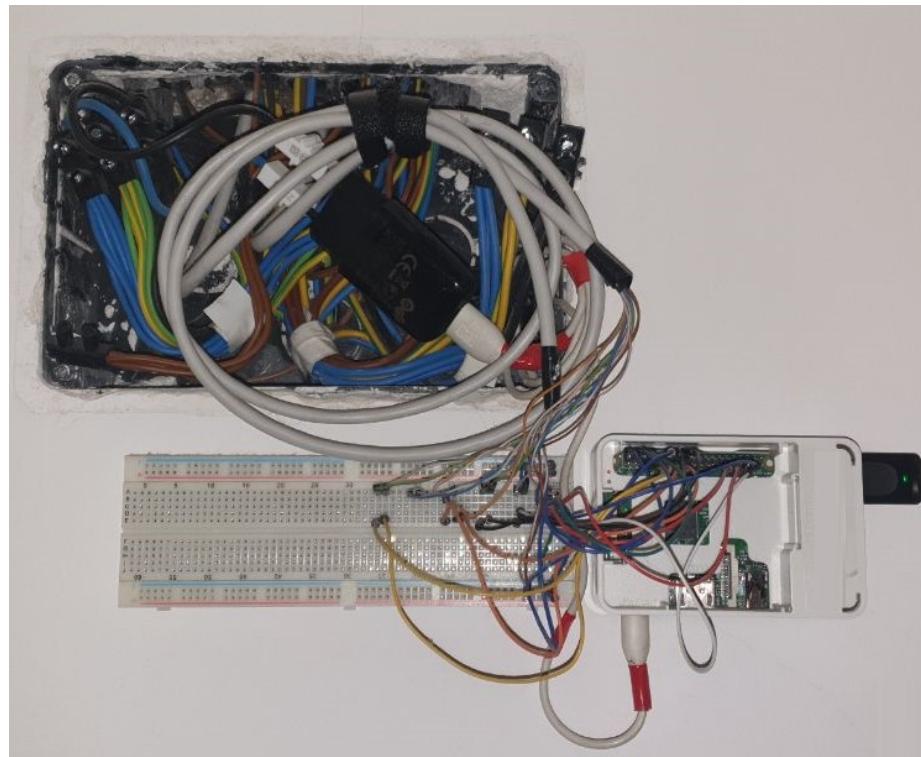


Figura 5.13: Instalación domótica y caja de derivación principal.

- En la imagen 5.11 podemos ver una caja de derivación donde ya tenemos albergada la placa de relés.
- En la imagen 5.12 vemos la caja de derivación principal con la instalación terminada indicando algunos elementos.
- En la imagen 5.13 vemos la situación final de la tirada de cable y conectorizado.
- En la imagen 5.14 podemos diferenciar que existen dos tipos de conectores. Los redondos ya vienen conectorizados de fábrica pero los cuadrados los he fabricado según las necesidades. Estos conectores cuadrados son del tipo [JST-XH](#).

El conectorizado final de las salidas de los GPIO es el que se puede ver en la imagen 5.15

Podemos interactuar con los puertos GPIO desde Bash y desde Python [5]. Se ha decidido utilizar lenguaje Bash para interactuar con los GPIO, dejando

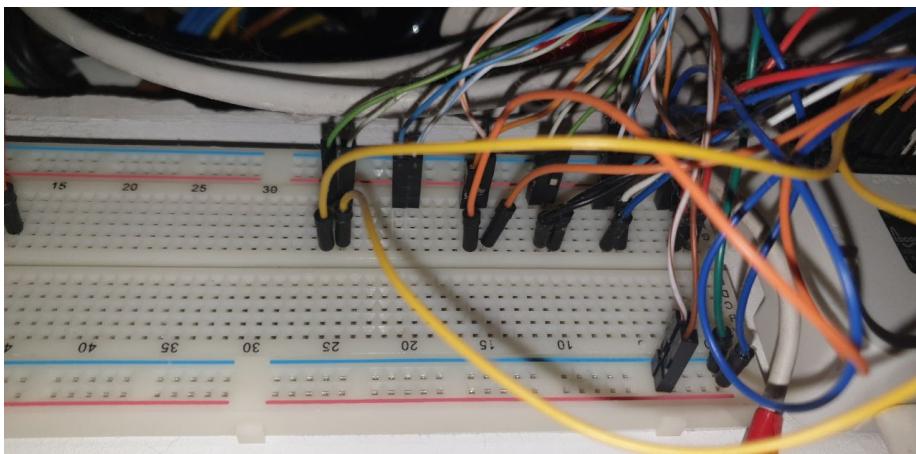


Figura 5.14: Protoboard conectada.

la potencia de Python para la obtención y procesado de datos. En ocasiones se producen errores al lanzar desde Cron scripts Python, por lo que se lanzará todo desde scripts bash.

## Diagramas eléctricos

Para mayor claridad he generado unos planos eléctricos para aclarar como se ha realizado la instalación.

- En la imagen 5.16 podemos ver que tenemos conectada una placa con dos relés y un pulsador en paralelo, con el motor de la persiana. La protoboard realmente no existe en este punto pero la he incluido para clarificar el circuito.
- En la imagen 5.17 podemos ver como se conecta el relé a una persiana identificando cada uno de los pines.

Para explicar el funcionamiento de un pulsador de 3 posiciones para persianas podemos ver la imagen 5.18. En ella podemos ver como en la posición central no tiene ninguna salida, pero en la salida 1 el orden de salida de cables es (+/-) y en la salida 2 el orden de salida de cables es (-/+). La lógica del cambio de orden en la salida de cables es porque según el sentido de la polaridad el motor girará en un sentido o en otro. Esto se respalda mediante la [segunda ley de Laplace](#) para cada espira del bobinado del motor quedando de la forma  $M = N * (I * B * l + \sin\theta)$ .

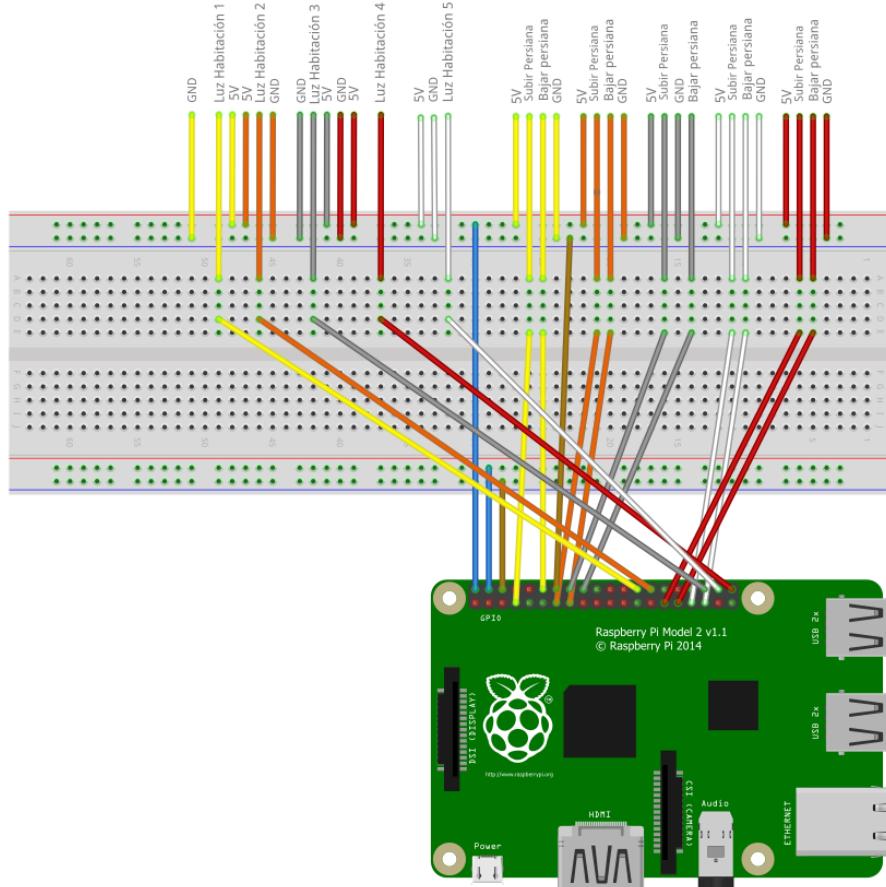


Figura 5.15: Conectorizado final de las líneas GPIO.

Siguiendo esta lógica, podemos ver que en la imagen 5.16 tenemos dos relés de forma que cada uno de ellos deja pasar la corriente en un sentido. En la realidad, el motor no tiene por qué tener dos entradas positivas y dos negativas pero en el dibujo queda mejor ilustrado que son entradas diferentes.

Resumiendo, la instalación mecánica y nuestra instalación mediante relés se instalarán en paralelo para poder activar una opción o la otra según la ocasión.

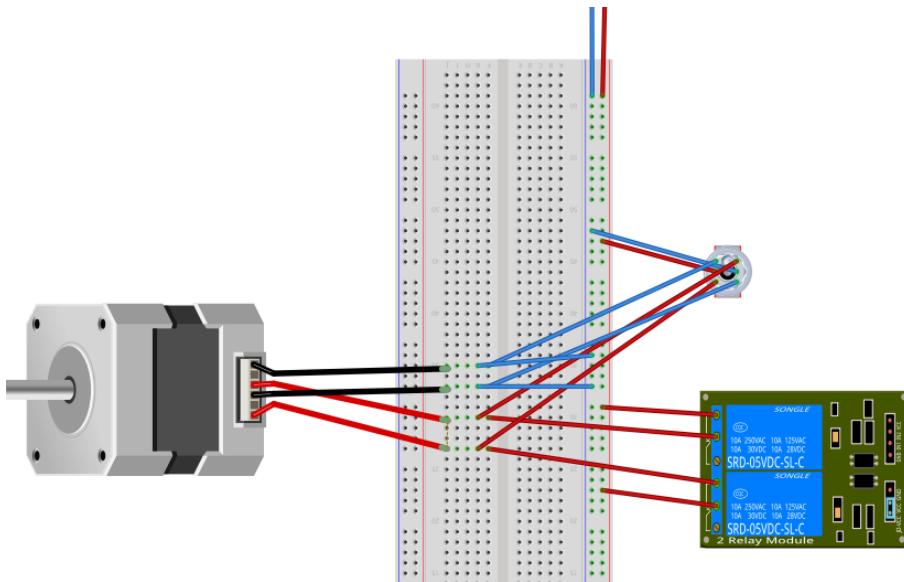


Figura 5.16: Raspberry Pi, relé y pulsador.

## Pruebas Físicas

La primera prueba, consistía en la instalación de la Raspberry Pi con un relé (ver imagen 4.5) y una bombilla conectada a 220VAC<sup>6</sup>, y se llevó a cabo con éxito. Podemos ver un diagrama de como se conectó en la imagen 5.19.

Llegados a este punto, era interesante el comprobar que se podía lanzar correctamente un script de este tipo desde CRON, encendiendo y apagando la bombilla, que también fue un éxito.

La tirada de cable y conectorizado en todo el domicilio, desde la primera caja de empalmes a las 5 persianas ya que es el punto del domicilio donde se recogen los cables eléctricos que llegan hasta las persianas, quedó como podemos ver en el diagrama 5.9.

## Automatizado y puesta a producción

Una vez se tienen las funcionalidades de los scripts corriendo correctamente, debemos asegurar las funciones y métodos de nuestra aplicación ante posibles fallos utilizando try/except. De esta manera siempre sabremos si se han actualizado correctamente los datos necesarios para que modifique nuestro Cron reportando un mensaje de error en caso contrario.

---

<sup>6</sup>VAC:Voltage Alternating Current

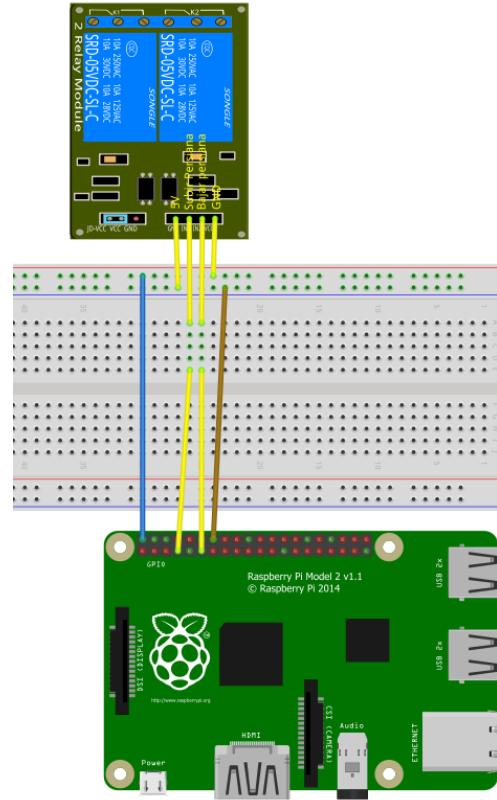


Figura 5.17: Raspberry y relé conectados.

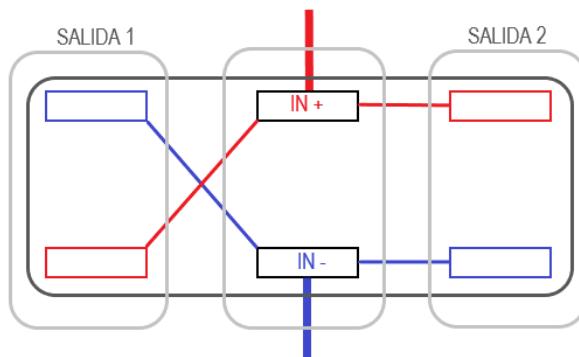


Figura 5.18: Funcionamiento interno de un pulsador para persianas.

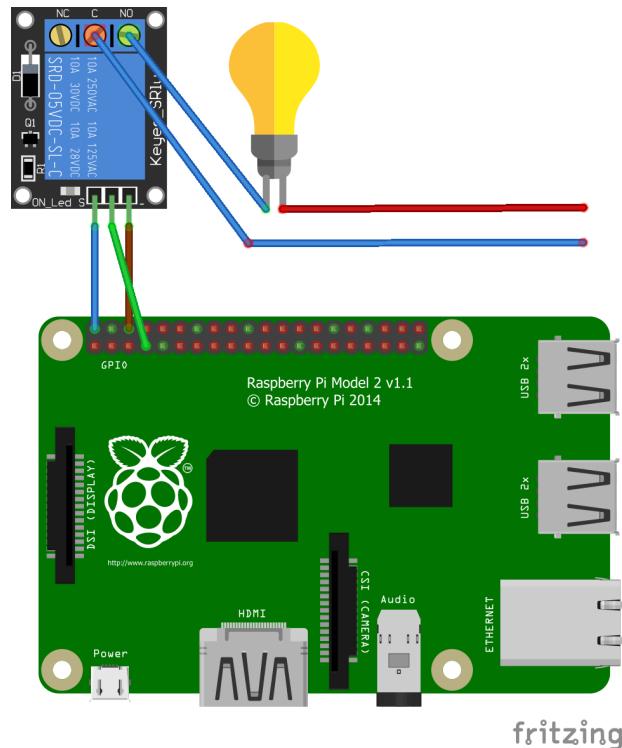


Figura 5.19: Plano de tablero de pruebas.

Para poder generar el proceso de que hablamos en la imagen 5.8 no es necesario salvar la cabecera del archivo de configuración de Cron, pero en nuestro caso, haremos una copia de la cabecera de forma que nuestra manipulación sea lo menos intrusiva posible. De esta manera, generaremos un Cron similar al que tenemos «de serie» pero con nuestras líneas de lanzado de scripts. En ese punto he tenido problemas al intentar ejecutar scripts Python desde Cron impidiendo que se obtuvieran los datos.

Uno de los puntos de la automatización es la generación de un demonio para nuestro bot de forma que se ejecutará con el inicio del sistema y podremos trabajar con una sencilla sentencia:

```
0 sudo systemctl bot stop
```

```
1 sudo systemctl bot start
```

Para conseguirlo, he realizado los siguientes pasos:

He generado el archivo el archivo de nuestro demonio en lib/systemd/system/ con la extensión «.service». Lo he escrito fácilmente con nano:

```
0 | sudo nano /lib/systemd/system/bot.service
```

El contenido del archivo es:

Listing 5.4: Modificaciones en el archivo /lib/systemd/system/bot.service.

```
0 | [Unit]
1 | Description=Lanza el bot de control domotico
2 | After=network.target
3 | StartLimitIntervalSec=0
4 |
5 | [Service]
6 | Type=simple
7 | Restart=always
8 | RestartSec=1
9 | User=pi
10 | WorkingDirectory=/home/pi/source/TFG/scripts/bot/
11 | ExecStart=/usr/bin/env python3 /home/pi/source/TFG/scripts/bot/bot.py
12 |
13 | [Install]
14 | WantedBy=multi-user.target
```

Después debemos actualizar los demonios con:

```
0 | systemctl daemon-reload
```

Iniciar el demonio:

```
0 | sudo systemctl start bot
```

Parar el demonio:

```
0 | sudo systemctl stop bot
```

Estado del demonio, con el que podremos conocer el pid <sup>7</sup>, que siempre es útil:

```
0 | sudo systemctl status bot
```

Para incluirlo en el inicio de la máquina habría que moverlo a /etc/init.d y luego ejecutar:

```
0 | sudo update-rc.d bot defaults
1 | sudo systemctl daemon-reload
2 | sudo systemctl enable bot
3 | sudo systemctl start bot
```

---

<sup>7</sup>Identificador del proceso

## Telegram Bot

En primer lugar, explicaré qué es un bot

El medio de comunicación con el sistema domótico es un bot de Telegram que he diseñado y programado para hacer completamente funcional el proyecto. Desde el Bot, podemos controlar las persianas de forma instantánea, modificar la configuración del sistema domótico y simulador de presencia, volver a generar recolección de datos y el grabado de los archivos. También podemos obtener información sobre la información recopilada, sobre nuestra Raspberry Pi y sobre el funcionamiento futuro del sistema domótico y simulador de presencia.

Afortunadamente estaba mejor documentado que el resto de la recursos que he consultado para el proyecto pero aún así no ha resultado tarea fácil entender la lógica del bot. Podemos ver la lógica de comunicaciones del bot en la imagen [5.20](#).

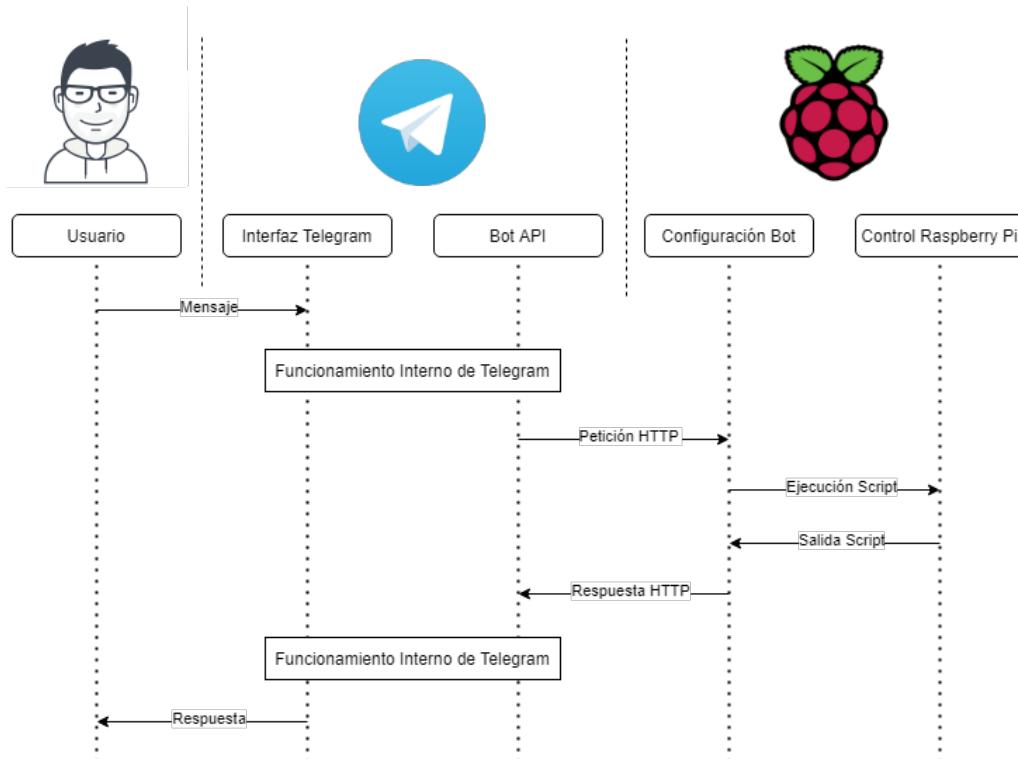


Figura 5.20: Comunicación del bot.

Todos los mensajes de Telegram que llegan a nuestro bot, tienen la siguiente estructura json:

Listing 5.5: Contenido de un mensaje de Telegram.

```

0  {
1      'content_type': 'text',
2      'id': 0000,
3      'message_id': 0000,
4      'from_user': {
5          'id': 0000000000,
6          'is_bot': False,
7          'first_name': 'David',
8          'username': 'David',
9          'last_name': 'David',
10         'language_code': 'es',
11         'can_join_groups': None,
12         'can_read_all_group_messages': None,
13         'supports_inline_queries': None
14     },
15     'date': 1609105539,
16     'chat': {
17         'id': 0000000000,
18         'type': 'private',
19         'title': None,
20         'username': 'David',
21         'first_name': 'David',
22         'last_name': 'David',
23         'all_members_are_administrators': None,
24         'photo': None,
25         'description': None,
26         'invite_link': None,
27         'pinned_message': None,
28         'permissions': None,
29         'slow_mode_delay': None,
30         'sticker_set_name': None,
31         'can_set_sticker_set': None
32     },
33     'forward_from': None,
34     'forward_from_chat': None,
35     'forward_from_message_id': None,
36     'forward_signature': None,
37     'forward_sender_name': None,
38     'forward_date': None,
39     'reply_to_message': None,
40     'edit_date': None,
41     'media_group_id': None,

```

```
42     'author_signature' :None,
43     'text' : 'a',
44     'entities' :None,
45     'caption_entities' :None,
46     'audio' :None,
47     'document' :None,
48     'photo' :None,
49     'sticker' :None,
50     'video' :None,
51     'video_note' :None,
52     'voice' :None,
53     'caption' :None,
54     'contact' :None,
55     'location' :None,
56     'venue' :None,
57     'animation' :None,
58     'dice' :None,
59     'new_chat_member' :None,
60     'new_chat_members' :None,
61     'left_chat_member' :None,
62     'new_chat_title' :None,
63     'new_chat_photo' :None,
64     'delete_chat_photo' :None,
65     'group_chat_created' :None,
66     'supergroup_chat_created' :None,
67     'channel_chat_created' :None,
68     'migrate_to_chat_id' :None,
69     'migrate_from_chat_id' :None,
70     'pinned_message' :None,
71     'invoice' :None,
72     'successful_payment' :None,
73     'connected_website' :None,
74     'reply_markup' :None,
75     'json' :{
76       'message_id' :0000,
77       'from' :{
78         'id' :0000000000,
79         'is_bot' :False,
80         'first_name' : 'David',
81         'last_name' : 'David',
82         'username' : 'David',
83         'language_code' : 'es'
84     },
```

```

85     'chat ':{  

86         'id ':000000000,  

87         'first_name ':'David ',  

88         'last_name ':'David ',  

89         'username ':'David ',  

90         'type ':'private '  

91     },  

92     'date ':1609105539,  

93     'text ':'a'  

94 }  

95 }
```

De esta estructura utilizamos únicamente unos pocos y podríamos resumir en los siguientes:

Listing 5.6: Resumen de los datos que utilizamos del mensaje de Telegram.

```

0 {  

1     'content_type ':'text ',  

2     'message_id ':0000,  

3     'text ':'a',  

4     'chat ':{  

5         'id ':000000000,  

6         'first_name ':'David ',  

7         'last_name ':'David ',  

8         'username ':'David '}  

9 }
```

Al comienzo del proyecto pensé en crear un grupo de usuarios para gestionar esta información pero he creído oportuno que cada usuario tenga su propia interfaz aunque realmente no habría problema en hacerlo de esta manera y hubiera facilitado el proyecto. Esto se ha solucionado incluyendo la lista de usuarios que pueden interactuar con el bot en el archivo de configuración. Para este paso, debemos recoger el id del usuario. Este id aparece varias veces en el json que recibimos en cada mensaje por lo que valdría cualquiera de ellos. Para el resumen he escogido el id que pertenece al objeto chat, junto al nombre y apellidos del propietario y el nombre de usuario. El número de usuario es necesario para saber a quién le enviamos el mensaje. Por ejemplo, al comienzo del bot del proyecto tenemos las siguientes líneas:

```

1 for usuario in usuarios:  

2     bot.send_message(usuario, "Iniciando Sistema a las "+str(hora))
```

Estas líneas envían un mensaje de inicio con la hora a la que se produce el evento a cada uno de los usuarios de la lista de admitidos.

Otro de los puntos necesarios es el de texto que envía el usuario. Gracias a éste obtendremos la orden que enviamos por Telegram y podremos actuar en consecuencia. Hay un punto a tener en cuenta para comprender el funcionamiento del bot, éste radica en las líneas que hay al principio de cada uno de los métodos de que dispone el bot:

```
1 @bot.message_handler(commands=['temp'])
2 @bot.message_handler(func=lambda message: message.text == "t")
3 @bot.message_handler(func=lambda message: message.text == "T")
4 def command_long_text(m):
5     usuario = m.chat.id
6     if (compruebaUsuario(m)):
7         temperaturasManana.temperaturas(m , bot)
```

En ellas podemos ver, por orden, el comando temp, al que podemos acceder enviando el comando /temp. Posteriormente vemos dos funciones lambda, éstas significan que si introducimos únicamente esa letra también lanzará el método. He introducido estos comandos rápidos para facilitar la introducción de órdenes. Observamos que si el usuario que escribe está en la lista de admitidos, lanzará el método temperaturas pasándole el mensaje enviado y el objeto bot, que contiene información sobre el usuario y el mensaje que envía. Por último, el método al que se llama, recoge esta información y opera en consecuencia según lo que se le pida, bien con una función u otro método.



---

## Trabajos relacionados

---

Desde la aparición de elementos electrónicos accesibles y asequibles se ha intentado, con mayor o menor fortuna, generar sistemas que nos ayuden en nuestro día a día. A continuación se exponen proyectos con objetivos similares.

### 6.1. Comparativa con otros proyectos

En este apartado mostraré otras opciones de acometer un proyecto similar y compararé los principales puntos de éstos con el presente proyecto para poder tener una visión global entre estas opciones.

#### Interacción remota para elementos de un hogar mediante una red de sensores actuadores

Se trata de un TFG/TFM desarrollado en la Universidad de Salamanca que trata sobre el control domótico basado en un conjunto de sensores instalados en el domicilio utilizando nodos intermedios para realizar el control de los dispositivos finales y la comunicación con los sensores.

Podemos ver su referencia en la bibliografía en la referencia [18].

En la columna ‘Domo\_1’ de la tabla 6.3 podemos ver algunas características de este proyecto.

## Diseño e implementación de un sistema domótico basado en Raspberry Pi

Éste es un TFG/TFM desarrollado en la Universidad Carlos III, en Madrid que trata sobre el control domótico de un domicilio basándose únicamente en un conjunto de sensores instalados en el domicilio.

- Url del trabajo: [https://e-archivo.uc3m.es/bitstream/handle/10016/26313/TFG\\_Hector\\_Santos\\_Senra.pdf?sequence=1&isAllowed=y](https://e-archivo.uc3m.es/bitstream/handle/10016/26313/TFG_Hector_Santos_Senra.pdf?sequence=1&isAllowed=y)

En la columna ‘Domo\_2’ de la tabla 6.3 podemos ver algunas características de este proyecto.

### 6.2. Fortalezas y debilidades este proyecto

Las principales fortalezas del proyecto que se presenta en esta memoria son:

- Éste es un proyecto que cualquiera puede implementar fácilmente en su domicilio haciendo una copia local del repositorio y haciendo unas sencillas configuraciones en el archivo de configuración.
- En este proyecto se ha tenido en cuenta la eficiencia de recursos y de consumo de la propia máquina Raspberry Pi [11], por lo que no se han incluido servicios extraordinarios como puede ser un servidor de bases de datos o web, como sucede en otros proyectos. Se ha optado por incluir un bot que ‘escucha’ las órdenes y actúa conforme a un pequeño archivo de configuración y actúa contra la máquina utilizando los lenguajes nativos de la máquina. De esta manera, el gasto en cómputo es menor y también el utilizado en mantener los servicios en memoria, quedando recursos suficientes para continuar haciendo un uso normal de la máquina para cualquier otra tarea. Además, al no tener servidores levantados y con puertos de comunicaciones abiertos, aumentamos la seguridad de nuestra instalación.
- Se ha procurado utilizar los lenguajes integrados de forma nativa en el Sistema Operativo para evitar posibles fallos de integración entre ellos y aumentando la fiabilidad del sistema. Estos son Python [5] y Bash [14].

Características	SDI	Domo_1	Domo_2
Proyecto libre	✓	✓	✗
No precisa montar servicios	✓	✗	✗
No requiere lenguajes no nativos en el SO	✓	✗	✗
Obtiene información externa contrastada	✓	✗	✗
Interacción multiplataforma	✓	✓	✓
No necesita nodos intermedios	✓	✗	✓
Cableado entre elementos	✓	✗	✓
WiFi entre elementos	✗	✓	✗

Tabla 6.3: Comparativa de las características de los proyectos.

- El sistema automatizado de nuestro proyecto se nutre de la información contrastable obtenida de API's reconocidas.
- Se puede interactuar con el sistema domótico de varias formas. La más cómoda es mediante el bot de Telegram creado en exclusiva para controlar las funcionalidades principales de nuestro sistema domótico, aunque también se puede conectar de forma remota desde otra máquina mediante VNC<sup>8</sup> siempre que se esté en la misma red. Se puede interactuar con el bot mediante aplicaciones Android o IOs, o desde navegadores de uso extendido como Chrome, Firefox o Edge, entre otros.
- No se necesitan elementos intermedios de ningún tipo, únicamente tenemos nuestra Raspberry Pi [11] y los actuadores o relés 4.5. De esta manera evitamos posibles fallos de los elementos intermedios. Podemos comprobarlo en la imagen 5.9.
- Se ha preferido cablear la instalación para evitar que los elementos receptores puedan quedar en un estado no deseado así como evitar reconfigurar elementos ante cualquier posible cambio de alguno de los elementos como pueden ser el *dongle* WiFi o el router, ver secciones 3.10 y 4.6 respectivamente. También, al dejar libres los canales WiFi, evitamos contribuir a la actual saturación de frecuencias en los espectros de 2.4GHz y 5GHz [13].

Podemos ver en la tabla 6.3 de comparativa de proyectos lo anteriormente expuesto.

<sup>8</sup>VNC: «Virtual Network Computing», programa de control remoto de equipos informáticos basado en una estructura «cliente-servidor»

Las principales debilidades del proyecto son:

- Hace falta cierta soltura a la hora de trabajar con el cableado UTP, ver sección [4.3](#).
- Es necesaria una línea de datos para que se actualicen los parámetros de los scripts.

---

# **Conclusiones y Líneas de trabajo futuras**

---

En este apartado se detallan el punto final con que concluye este proyecto, desglosando cada uno de los objetivos superados en el mismo. Posteriormente se sugerirán algunas posibles líneas de trabajo futuras que puedan dar continuidad al proyecto consiguiendo una mejora notable.

## **7.1. Conclusiones**

Una vez finalizado el proyecto, podemos concluir con las siguientes afirmaciones:

- Se ha conseguido crear un Sistema de Simulación de Presencia que funciona según las horas de luz del día. En cada jornada, el Sistema genera una consulta de forma automática obteniendo los parámetros de ubicación, y posteriormente, los parámetros de salida y puesta de sol. Seguidamente cocina los datos para controlar los diferentes elementos externos.
- Además, se ha ampliado el Sistema de Simulación de Presencia con el Sistema Domótico que controla la caldera, las persianas y las luces. De esta manera, también controlamos a qué hora se suben las persianas por las mañanas y se controla el encendido de la caldera. Esta parte es parcialmente automática ya que tenemos la opción de subir las persianas, tras el amanecer, a la hora que queramos.

- Se ha conseguido dotar al sistema de un control instantáneo de las persianas. Este apartado beneficia a la hora de querer controlar una persiana de forma individual y en un intervalo de tiempo corto.
- También, se han incluido órdenes informativas que funcionan a partir de la información recopilada porque no cuesta nada poder contar con la información recopilada de una forma limpia e instantánea siempre que la queramos. Además, se dibujan diariamente una gráfica con las temperaturas del día siguiente que también están a nuestra disposición siempre que solicitemos el envío de alguna de ellas
- El sistema es autónomo de la ubicación en la que está instalado ya que obtiene su ubicación en cada consulta lo cual es necesario para poder obtener las temperaturas del día siguiente.
- Se ha generado un bot para automatizar todo el proceso de comunicación con el Sistema Domótico Inteligente de forma que podemos interactuar con él desde cualquier lugar de una manera segura y minimizando el cómputo, los servicios activos y también los puertos abiertos.
- El sistema, finalmente, funciona con un mínimo de instrucciones y programas en background, lo cual contribuye a un mínimo consumo energético y pone a disposición del usuario los recursos de la máquina para realizar otras tareas que se precisen.
- Estoy satisfecho con la instalación física realizada en el domicilio, me parece profesional ya que sigue las líneas de las normativas legales vigentes por las que se rigen este tipo de instalaciones.
- Otro punto muy satisfactorio es el de la mínima inversión acometida en el proyecto junto a una funcionalidad completa hacen que sienta que el dinero ha sido muy bien invertido ya que ha aumentado notablemente la comodidad y confortabilidad del domicilio. Además se nota que, ahora en invierno, la temperatura de la vivienda desciende más despacio al bajar las persianas cuando anocchece.
- La única pega que le podría al proyecto es que al utilizar la versión 2B de la Raspberry Pi, algunas peticiones tardan unos segundos más de lo que me gustaría. Esto sucede cuando se le ordena que modifique la hora de subida de las persianas ya que debe abrir y modificar varios archivos, pero se soluciona con una versión más nueva de Raspberry Pi.

## 7.2. Líneas de trabajo futuras

El proyecto ha quedado bastante «redondo», es decir, se ha generado una funcionalidad completa y se ha dado solución a una necesidad dentro de los parámetros que se esperaba. No obstante, el proyecto es susceptible de crecer para aumentar el confort y mejorar el Sistema Simulador de Presencia y el Sistema Domótico.

- El proyecto se centra únicamente en valores externos como son el sol o la temperatura exterior. Se puede incluir un sensor de temperatura en la vivienda de forma que también se controle la calefacción siempre que la caldera esté encendida.
- Al Sistema Simulador de Presencia podemos incluirle sonido. Una buena opción puede ser conseguir emitir un programa en streaming<sup>9</sup>, ya sea solo audio o también con vídeo.
- Ya que hemos incluido Telegram y éste dispone de llamadas, podemos incluir un «portero IP» de forma que te llame a un usuario y puedas comunicarte con quien llame a la puerta. De esta manera no sólo simulas que estás en el domicilio sino que también sabes quién ha estado en tu puerta. Este portero puede ser sólo con voz o también con imagen.
- Al mismo sistema, se le pueden incorporar unos sensores PIR para saber en qué parte de la casa hay movimiento y disponer un control de luces o grabado de imágenes.
- También se puede optar por crecer en la rama de seguridad añadiendo sensores de gas, agua y humo para y notifique las alertas.
- Si se vive en un lugar con puerta de garaje individual, se puede programar para interactuar con ésta permitiendo el acceso con un mensaje.

---

<sup>9</sup>Retransmisión en directo.



---

# Bibliografía

---

- [1] ANSI/TIA/EIA. *ANSI/TIA/EIA568-Balanced Twisted-Pair Cabling Components*. [Página web Oficial, Noviembre de 2020].
- [2] ANSI/TIA/EIA568. *ANSI/TIA/EIA568-General Requirements*. [Página web Oficial, Noviembre de 2020].
- [3] BOE. *ICT-BT-21*. [Documentación técnica, Octubre de 2020].
- [4] BOE. *Normativa de ICT*. [Documentación técnica, Octubre de 2020].
- [5] BOE. Python software foundation. [Página web Oficial].
- [6] BOE. *Reglamento Electrotécnico de Baja Tensión*, 2020. [Documentación técnica, Octubre de 2020].
- [7] BOE+Televés. *Normativa de ICT Televés*. [Página web Televés, Octubre de 2020].
- [8] Bujarra.com. Rbp uso de relé y detector de movimiento. [Página web].
- [9] Grupo Lifelong Kindergarten del MIT Media Lab. Scratch. [Página web Oficial].
- [10] ECMA. Javascript object notation. [Página web Oficial, Noviembre de 2020].
- [11] Raspberry Pi Foundation. Raspberry pi foundation. [Página web Oficial].
- [12] <https://descubrearduino.com>. ¿qué es un microcontrolador? introducción para principiantes. [Página web].

- [13] IEEE. *Normativa IEEE802.11.* [Página web Oficial del estándar, Noviembre de 2020].
- [14] Linux. Linux. [Página web Oficial, Noviembre de 2020].
- [15] Arslan Musaddiq, Farhan Amin, Yousaf Zikria, Rojeena Bajracharya, and Sung Kim. Raspberry pi-based smart home automation system for internet of things. 05 2017.
- [16] Oracle Technology Network. Java. [Página web Oficial].
- [17] OpenSource. Opensource. [Página web Oficial, Noviembre de 2020].
- [18] Alberto Vaquero Pedruelo. Interacción remota para elementos de un hogar mediante una red de sensores actuadores. [TFG].
- [19] programoergosum.com. Control de gpio con python en raspberry pi. [Página web].
- [20] Wikipedia. *Metodología Scrum.* [Página web, Noviembre de 2020].