UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

**TFG del Grado en Ingeniería Informática**

# Sistema Domótico Inteligente
# Documentación Técnica


Presentado por
en Universidad de Burgos — 30 de diciembre de 2020
Tutor: Álvar Arnaiz-González
Tutor: Alejandro Merino Gómez

# Índice general

# Índice de figuras

# Índice de tablas

*Apéndice A*

# Plan de Proyecto Software

## A.1. Introducción

En cualquier proyecto de este tipo es relevante incluir cierta información sobre la planificación, viabilidad y análisis de costes en el desarrollo del mismo con la finalidad de conseguir una visión de cómo ha ido evolucionando el proyecto y los ciclos de trabajo que se han llevado a cabo.

He de señalar que en la última tercera parte de los sprints, no siempre he contado con una línea de datos con la que poder replicar datos en GitHub, por lo que se han generado menos commits de los que se hubiera hecho si hubiera contado con medios a mi alcance. En cualquier caso, se ha intentado maximizar el número de puntos de salvado para que quedase constancia de la evolución del proyecto. Además, al redactar el proyecto en LaTeX desde la plataforma de Overleaf no refleja el tiempo real invertido puesto que no se realizan cambios en el repositorio de forma automática, hasta que compilo, descargo y actualizo el repositorio manualmente.

En el primer apartado trataremos la **planificación temporal**, donde podremos ver como han evolucionado los tiempos de trabajo durante las semanas en las que se ha trabajado en el proyecto. En el segundo apartado se reflejará un **estudio de viabilidad** sobre el proyecto, que a su vez incluye dos partes, la parte económica y la parte legal.

## A.2. Planificación temporal

Al comenzar el proyecto se determinó que se utilizaría una metodología ágil para hacer el desarrollo del proyecto pero esta decisión ha terminado

siendo un hándicap producido por la falta de documentación y la dudosa credibilidad de muchas páginas web y documentación que he encontrado, y se ha traducido en una gran inversión en tiempo.

Ante todo, se ha intentado seguir un mínimo de pautas **Scrum** [**?**] pese a no existir un grupo de trabajo real con unas tareas diarias y roles definidos:

- Las tareas fueron siempre semanales en forma de «sprints».
- Al finalizar el sprint se hace la entrega del trabajo elaborado en la semana y se determinan las próximas tareas.
- Tras determinar las tareas se definen los «milestones» y los «issues».
- Para hacer el seguimiento de las tareas se ha utilizado en tablero **Kanban** de **ZenHub**.
- Tras finalizar el sprint se puede comprobar el trabajo mediante los *gráficos burndown*.

Las reuniones de trabajo han sido consensuadas y planificadas para realizarlas los jueves de cada semana, con el siguiente resultado:

## Sprint 00 - 01/10/2020 - 08/10/2020

En la primera reunión nos reunimos Álvar Arnaiz González y yo. En ella hice la propuesta de temática del TFG y, además expuse diferentes ideas y enfoques sobre el proyecto en la que se determinó a grandes rasgos la posible viabilidad del proyecto. Los objetivos de este «sprint» introductorio fueron la búsqueda de repositorios y otros proyectos para tomar ideas o si era posible hacer algún fork. También, el estudio la estructura del proyecto y herramientas software e interfaces a utilizar.

## Sprint 01 - 09/10/2020 - 15/10/2020

Tras determinar el enfoque final de proyecto, se consensuó que los objetivos de este sprint fueron la búsqueda de información relevante, otros proyectos y recursos que puedan servir de utilidad como pueden ser APIS. Se buscaron algunas páginas web para realizar pruebas de extracción de datos con «web scraping» [1] y se descompusieron las tareas en unidades pequeñas de trabajo para poder afrontarlas durante sprints.

[1] Se trata brevemente en el punto 4 de la memoria

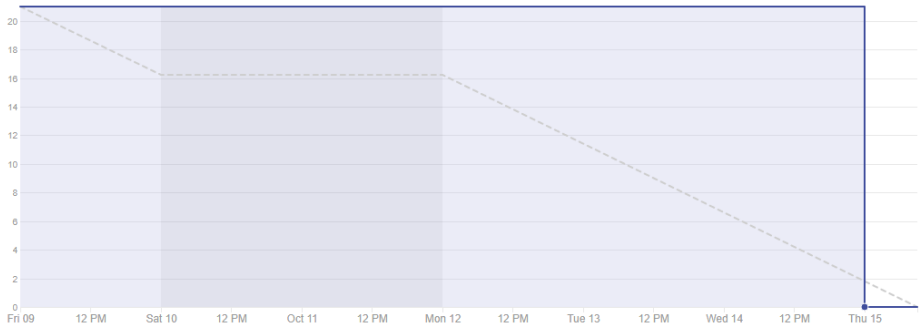


Figura A.1: Gráfico Burndown sprint1.

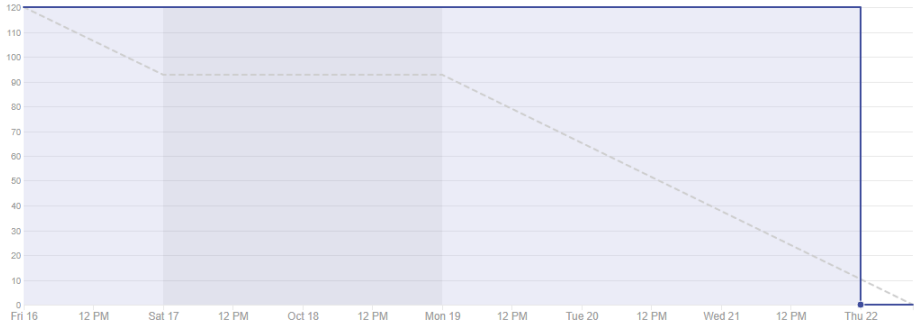


Figura A.2: Gráfico Burndown sprint2.

En este sprint también generé el repositorio en GitHub para poder trabajar contra él y podemos ver las líneas de código utilizadas en el primer *issue*.

## Sprint 02 - 16/10/2020 - 22/10/2020

Los objetivos de este sprint fueron el buscar repositorios, APIS, servicios y tecnologías con las que darle forma al proyecto. Se decidió generar algún tipo de aplicación desde la que poder controlar la instalación haciéndola más amigable y con mejor capacidad de interacción. Se investigan opciones.

En el issue 2 se hizo una búsqueda por la web para buscar repositorios y proyectos sobre domótica

En el issue 3 buscamos algunas APIS para orientar el proyecto hacia el uso de APIS destinadas a ofrecer información que pueda servirnos y se realizaron algunas pruebas.


Figura A.3: Gráfico Burndown sprint3.

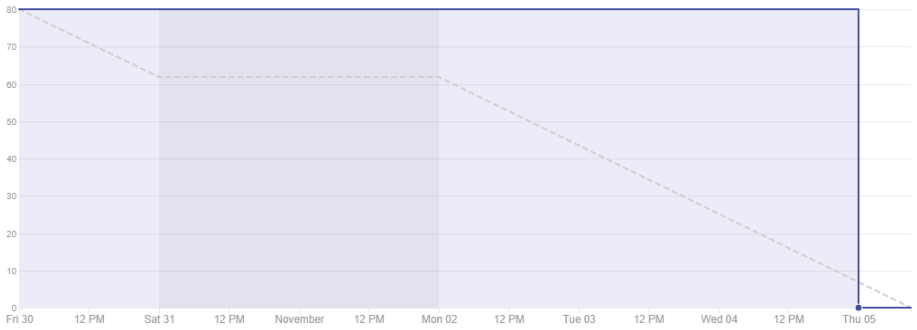


Figura A.4: Gráfico Burndown sprint4.

En el issue 4 se realizó un pequeño estudio de las tecnologías que podríamos utilizar, desde el procesador de textos, que finalmente se utilizó LaTeX, como la revisión de las normativas electrotécnicas que debía seguir en el proyecto y también se determinó la utilización de Telegram para interactuar con el Sistema Domótico.

## Sprint 03 - 23/10/2020 - 29/10/2020

Los objetivos de este sprint fueron determinar los componentes hardware necesarios para completar el proyecto y se realiza la compra de material. En todos los «issues» del «milestone 3» podemos ver múltiples enlaces, comparativas, justificaciones e imágenes en las que se ha basado toda la instalación física posterior.

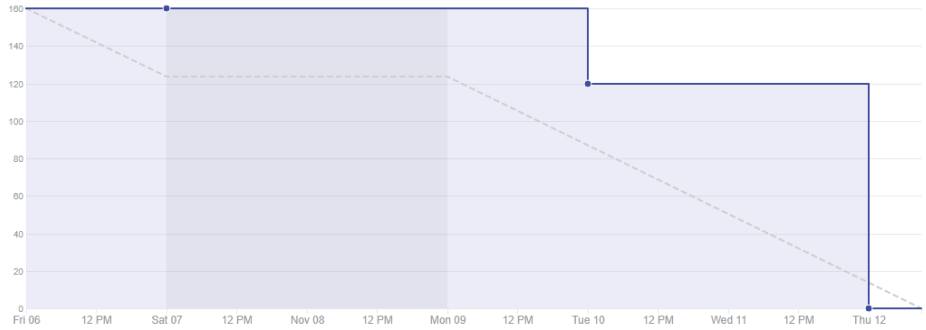


Figura A.5: Gráfico Burndown sprint5.

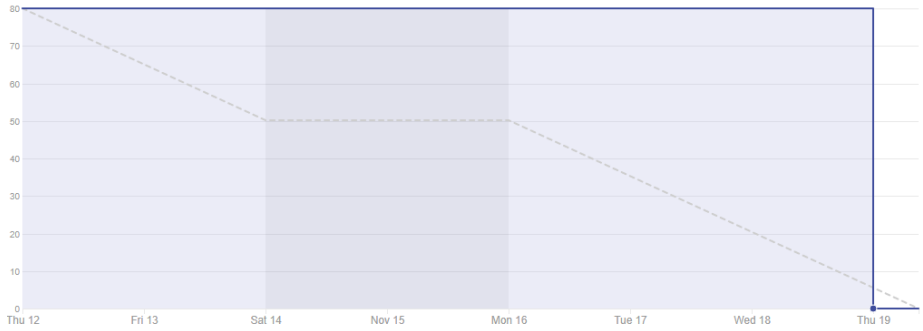


Figura A.6: Gráfico Burndown sprint6.

## Sprint 04 - 30/10/2020 - 05/10/2020

Se incorpora D.Alejandro Merino Gómez al que se le presenta el proyecto y se le concede acceso a los repositorios para que pueda co-tutorizar el proyecto.

Los objetivos de este «milestone» fueron todos de la parte física de la instalación:

Podemos ver en el issue 15 que finalicé la tirada de cable junto a un diagrama explicativo del funcionamiento de un pulsador de 3 posiciones (Posición 1, Reposo y Posición 2. Teniendo en cuenta que las dos posiciones que dan continuidad al circuito tienen invertida la polaridad).

También vemos en el issue 16 la presentación de la Raspberry Pi que utilizaremos, los conectores del tipo JST-XH que se han utilizado en el crimpado de los cables electrónicos, los cables finalizados con sus conectores y la disposición final de la Raspberry Pi en su ubicación final de la instalación.

## Sprint 05 - 06/11/2020 - 12/11/2020

Se repasan los cambios propuestos y se incluyen otros nuevos sobre la parte física de la instalación. Se hace el seguimiento de la instalación y se comentan las fotos y el estado de la instalación. Los objetivos de este sprint son todos aquellos que sean necesarios para la adecuación del software básico para comenzar el proyecto como la actualización del sistema operativo o instalación de software adicional. Además se deben valorar opciones para controlar los GPIO.

Vemos en los issues 21 y 19 las configuraciones básicas realizadas a nuestro Raspbian. Para explicar este proceso grabé dos vídeos, titulados Primeros Pasos Raspberry Pi y Actualización de Raspberry Pi y configuración básica, respectivamente.

## Sprint 06 - 13/11/2020 - 19/11/2020

Se revisan los puntos anteriores y se compone un tablero de pruebas de forma que se puede controlar el encendido de una bombilla desde nuestra Raspberry Pi. También se graba un vídeo explicativo y se diseñan unos diagramas explicativos.

Podemos ver en el issue 22 que se presenta un diagrama con la lista de los componentes que vamos a utilizar en el tablero de pruebas y un diagrama de como se compondrá el tablero. En este issue también grabé un vídeo para explicar el tablero de pruebas y decidí utilizar un efecto más conservador de cara al final del proyecto.

En el issue 23 he explicado como se controlan los GPIO de nuestra Raspberry Pi desde su distribución Linux [2] desde Bash y desde Python.

## Sprint 07 - 20/11/2020 - 26/11/2020

Se realiza una investigación sobre como se puede implantar el bot en nuestro proyecto, se realiza el primer código de pruebas y se presentan las primeras pruebas con el bot. Podemos ver en el issue 13 un resumen de las pruebas que estuve realizando con los diferentes teclados de que dispone Telegram.

también, se proponen correcciones en la redacción.

[2]Raspbery pi (Raspbian OS).

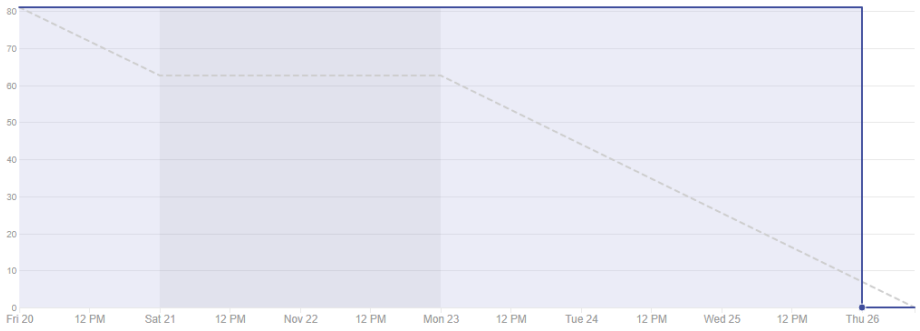


Figura A.7: Gráfico Burndown sprint7.

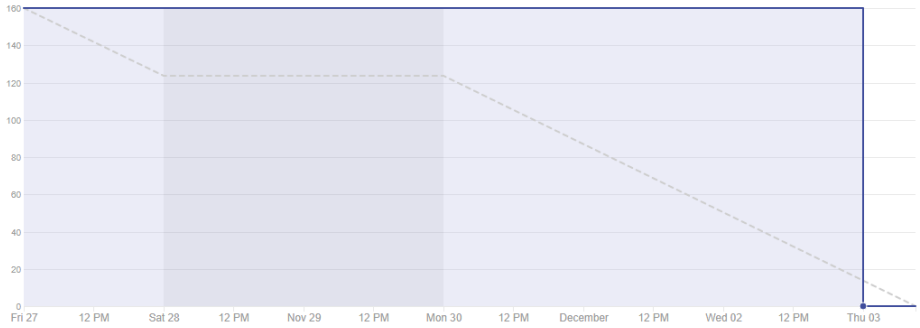


Figura A.8: Gráfico Burndown sprint8.

## Sprint 08 - 27/11/2020 - 03/12/2020

En el milestone 8, podemos ver que se incluyen varios issues:

- Creación de scripts Bash para controlar el conjunto del Sistema Domótico inteligente.
- Obtención de los datos de geolocalización y cálculo de temperaturas.
- Generación del nuevo Cron tras integrar los scripts existentes.

De esta manera, se genera la primera automatización completa de la parte automática del proyecto utilizando CRON y lo pasamos del entorno de pruebas a un entorno de producción en fase experimental. En este momento únicamente tenemos control sobre la máquina de forma física o por VNC. Por último, se presentan las primeras funcionalidades del bot, se proponen cambios y mejoras.

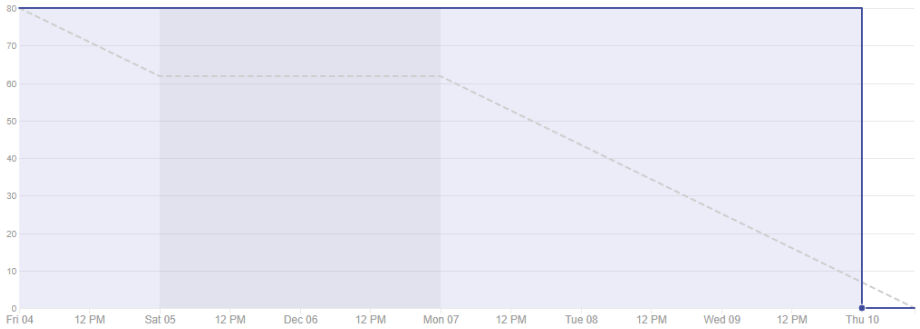


Figura A.9: Gráfico Burndown sprint9.

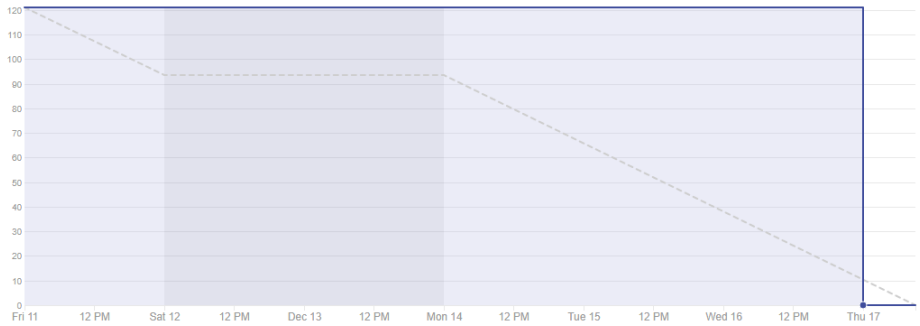


Figura A.10: Gráfico Burndown sprint10.

## Sprint 09 - 04/12/2020 - 10/12/2020

En el milestone 9 se proponen revisan las funcionalidades del Bot [**?**] de Telegram [**?**] que utilizaremos, se proponen cambios y mejoras en el código preexistente en fase de pruebas.

Finalmente se integra un bot de Telegram completamente funcional desde el que podremos lanzar órdenes a nuestro sistema domótico con los scripts existentes.

## Sprint 10 - 11/12/2020 - 17/12/2020

En el milestone 10 se ha aprovechado para optimizar las rutas de los archivos a los que se invoca desde el código en ejecución y se han modificado algunas salidas para que tengan un aspecto visual más atractivo, incluyendo tablas, emoticonos desde Unicode [**?**].

Se comprueban las funcionalidades proponiendo cambios funcionales y de estilos de forma que se pueda interactuar contra el bot y se muestren

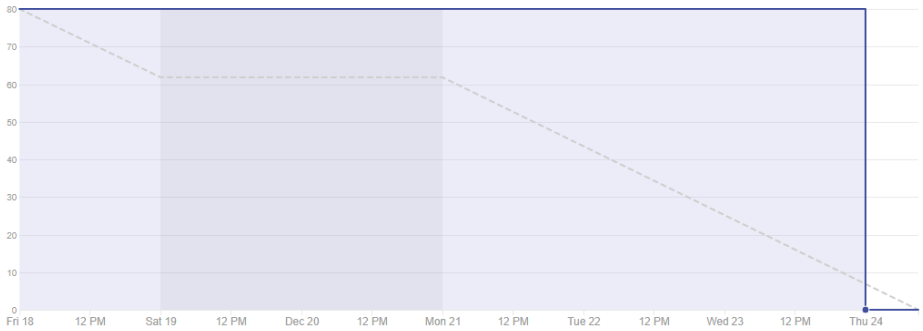


Figura A.11: Gráfico Burndown sprint11.

los mensajes en un formato más atractivo. También se propone unificar las rutas de trabajo de los diferentes scripts.

## Sprint 11 - 18/12/2020 - 24/12/2020

Se propone dividir el código por funcionalidades (obtención de información de la web, y por otro lado, modificación de archivos del sistema y se proponen cambios en la redacción a implementar en el sprint 12.

Finalmente, se ha conseguido generadr la división del código en 3 archivos principales:

1. Toma de datos de la web y guardado en json.
2. Lectura de parámetros personalizados.
3. Grabado de CRON conforme a esas preferencias.

De esta manera conseguimos minimizar las comunicaciones con el exterior y poder relanzar el código separado por funcionalidad lógica de forma que podamos relanzarlo de forma aislada en caso de necesitarlo. Un ejemplo puede ser cuando se modifica la hora de subida de las persianas, el módulo de generación de CRON leería el archivo de parámetros personalizados y generaría la configuración a partir de éstos.

## Sprint 12 - 30/12/2020 - 06/01/2021

Se finaliza la redacción de la memoria, antes de la revisión final del texto, y se comienza con los anexos. Además se produce depurado del código y se utiliza SonarCloud.

### Sprint 13 - 07/01/2020 - 13/01/2020

Redacción del proyecto del TFG y grabado de vídeos.

### Sprint 14 - 14/01/2020 - 20/01/2020

Edición de vídeos y entrega.

## A.3. Estudio de viabilidad

### Viabilidad económica

### Viabilidad legal

Apéndice $B$

# Especificación de Requisitos

## B.1. Introducción

## B.2. Objetivos generales

## B.3. Funcionamiento autónomo

Se pretende que tras pocas configuraciones el sistema sea capaz de funcionar extrayendo información de Internet y decidiendo que acción realizar en cada caso a partir de ese momento.

## B.4. Escalabilidad

Como cualquier proyecto de calidad debe ser escalable, esto es, debe existir la posibilidad de que sea ampliado fácilmente.

## B.5. Acceso multiplataforma

En este caso dispondremos de un acceso GUI(del inglés «Graphical User Interface» o «Interfaz Gráfica de Usuario») o CLI[1] (Línea de comandos) a nuestro equipo.

[1]Traducción del inglés: 'Command Line Interface'

## B.6. Ahorro energético

El ahorro energético se produce al permitir que la temperatura exterior incida, o no, en la vivienda para conseguir las condiciones deseadas optimizando el consumo de recursos.

COMPROBAR UBICACIÓN

## B.7. Objetivos técnicos

- Debe disponer de un Sistema Operativo GNU como puede ser una distribución de Linux. En nuestro caso utilizaremos Raspbian.
- Posibilidad de interacción multiplataforma para aumentar la flexibilidad a la hora de interactuar con los equipos.
- También, debe ser fácilmente escalable para permitir la posibilidad de continuar con el desarrollo.
- Utilizar Git y Github como sistema de control de versiones, de Zenhub como herramienta de gestión de proyectos y Atom para trabajar con el control de versiones.
- Utilización de Frizking para realizar los diseños electrónicos de la instalación.
- Utilización de www.draw.io para realizar los diseños de la instalación.
- Tratamiento de datos utilizando lenguaje Python.
- Utilización de diferentes APIs para obtener datos.
- Correcto conectorizado electrónico de todos los elementos según RETB.

## B.8. Catálogo de requisitos

## B.9. Especificación de requisitos

*Apéndice* $C$

# Especificación de diseño

## C.1. Introducción

## C.2. Diseño de datos

## C.3. Diseño procedimental

## C.4. Diseño arquitectónico

Apéndice $D$

# Documentación técnica de programación

## D.1. Introducción

## D.2. Estructura de directorios

Hay que conocer las diferentes partes del proyecto y tener en cuenta que son completamente diferentes aunque en este proyecto se han integrado para poder hacer uso del sistema domótico. Podemos ver una primera distribución general en la imagen D.1. En ella podemos diferenciar tres partes principales:

Tuve un problema generalizado en el código puesto que obtenía las rutas a los archivos de forma unitaria desde el archivo en ejecución. Este es el código utilizado para obtener la ruta en Python: [language=json,firstnumber=0] !/usr/bin/env python3 Calculamos ruta ruta=os.getcwd().split('/')

Éste es el código utilizado para obtener la ruta en Bash: [language=json,firstnumber=0] !/bin/bash path=(*pwd*)

- La parte de toma de datos.
- La parte de mensajería.
- La parte física de nuestra instalación domótica, incluyendo la Raspberrry Pi.

Por ello, el código se ha estructurado en tres partes:

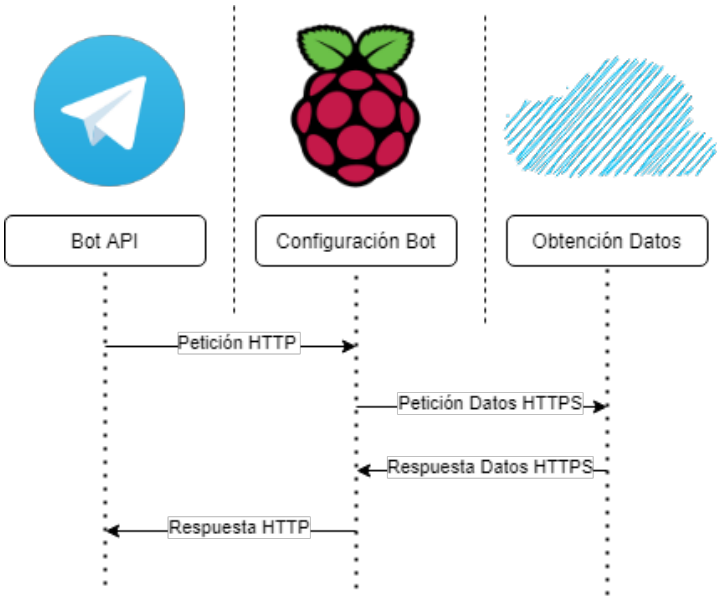


Figura D.1: Partes del proyecto.

- La carpeta **auto**, que permite obtener la información de las APIs seleccionadas de Internet.
- La carpeta **bot**, donde almacenamos la configuración de nuestro bot.
- La carpeta **control**, donde almacenamos los scripts de control de la instalación domótica.

En todas ellas, los scripts, tanto de Bash como de Python llevan la cabecera específica para el tipo de código contenido en el archivo:

[language=json,firstnumber=0] !/usr/bin/env python3

Éste es el código utilizado para obtener la ruta en Bash: [language=json,firstnumber=0] !/bin/bash

Para los archivos Python, la sentencia «!/usr/bin/env python3» y para los archivos Bash, la sentencia «!/bin/bash». He introducido esta cabecera porque he tenido problemas al lanzar las instrucciones desde otro script.

## Carpeta auto

Al comienzo del proyecto esperé encontrar alguna web desde la que poder recoger la información haciendo webscraping pero tras valorarlo

detenidamente en las tutorías prefería a buscar una API (ver concepto **??**). Uno de los motivos es que una web es más susceptible de sufrir cambios que una API que se ha predispuesto para una función. Finalmente, dispuse el proyecto para poder funcionar con dos APIs públicas y gratuitas para dotar a nuestro sistema domótico de autonomía. Las siguientes fases de la parte de toma de datos, se albergan en la carpeta auto ya que es un proceso automatizado que permite al sistema obtener los datos de forma desatendida y continua. El proceso de automatización comprende varias fases:

1. La fase de recopilación de datos.
2. La fase de lectura local de parámetros.
3. El cocinado de los datos.
4. El grabado de datos en el sistema local.

La fase de recopilación de datos llama a la API de geolocalización y nos devuelve, entre otros, los parámetros de nuestra ubicación en formato de latitud y longitud. Posteriormente, estos datos de ubicación son necesarios para incorporarlos en la consulta de la siguiente API **??** y obtener la información de la salida y puesta del sol así como la información de las temperaturas del día siguiente en dicha ubicación. Con esta recopilación de datos tenemos la base para poder determinar el comportamiento autónomo que necesitamos por lo que almacenaremos esta información en un archivo de datos.

El siguiente paso es leer las preferencias del usuario, que se almacenan en el archivo de condicionantes. Y, una vez tengamos estos datos, podemos pasar a generar el Cron para controlar de forma automática toda la instalación.

Estas fases se han confeccionado al final del proyecto puesto que al principio se desarrolló de forma unitaria pero surgió el problema de que, cada vez que queríamos hacer una modificación en el sistema domótico de cualquier índole, teníamos que hacer una petición nueva a las APIs y generar de nuevo un archivo Cron con los parámetros del usuario «al vuelo» para conseguir regenerar la automatización con los parámetros nuevos. Este sistema funcionaba correctamente pero era poco eficiente puesto que que se multiplican las operaciones en función de las veces que se quiera modificar algún parámetro. Únicamente he permitido realizar dos salidas .ªl vuelo"tras obtener los datos, estos son: el grabado de los datos recién obtenidos y la gráfica de las temperaturas conforme a dichos datos.

He tenido algunos problemas al realizar varias peticiones a las APIS ya que, por falta de permisos, no podía sobreescribir los archivos ni las imágenes.

Para subsanarlo, incluí una restricción para modificar los permisos de los archivos de forma que únicamente pueden acceder el propietario y el grupo.

## Carpeta control

La carpeta control alberga aquellos scripts bash que controlan los elementos domóticos. Este punto únicamente me dio problemas a la hora de implantarlo desde el bot, que se subsanó con la librería os.

## Carpeta bot

La carpeta bot contiene varios archivos, desde el archivo de control del bot hasta otros que contienen la mayoría de funcionalidades de éste. En primera instancia, se dispuso todo el código dentro del archivo de control del bot, pero por legibilidad y facilidad de mantenimiento se extrajeron las funciones. En este punto, lo que más problemas me ha dado han sido los teclados. Como comportamiento de los teclados es diferente entre ellos tendría que dividir las opciones del bot entre dos tipos de teclados diferentes. Finalmente, opté por no incluir estos teclados ya que no incluye funcionalidad alguna y disminuirían la legibilidad y facilidad de uso del bot. Además, para aumentar la usabilidad del bot, se ha dispuesto un menú que podemos ver al introducir el carácter /.ªdemás de la posibilidad de utilizar la primera letra del comando siempre que no tengamos que acompañarlo de parámetros.

## Configuración del bot como servicio

El bot se puede lanzar como si fuera un script más pero en este caso es necesario incorporarlo como servicio. Esta decisión tiene beneficios como la posibilidad de que se inicie con el sistema o que podamos lanzarlo o pararlo desde cualquier ubicación del SO sin tener que recordar la ubicación con la sentencia al efecto de «**sudo service bot start/stop**». Pero, una vez que el demonio está corriendo, tenemos que tener en cuenta de que ya está levantada la instancia del bot, por lo que tendremos que detenerla a la hora de hacer algún tipo de modificación así como recordar que debemos levantarlo una vez terminemos.

## D.3. Manual del programador

## D.4. Compilación, instalación y ejecución del proyecto

## D.5. Pruebas del sistema

*Apéndice* $E$

# Documentación de usuario

## E.1. Introducción

## E.2. Requisitos de usuarios

## E.3. Instalación

## E.4. Manual del usuario

# Bibliografía