



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería Informática

**Diseño de un sistema
económico IoT de
monitorización de
invernaderos de cannabis
medicinal.
Documentación Técnica**



Presentado por José Luis Caballero
Martínez-Quintanilla.
en Universidad de Burgos — 16 de febrero de
2024

Tutor: Alejandro Merino Gómez
Tutor: Carlos Cambra Baseca

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	6
Apéndice B Especificación de Requisitos	13
B.1. Introducción	13
B.2. Objetivos generales	13
B.3. Catálogo de requisitos	14
Apéndice C Documentación técnica de programación	17
C.1. Introducción	17
C.2. Estructura de directorios	17
C.3. Manual del programador	24
C.4. Compilación, instalación y ejecución del proyecto	47
C.5. Calidad del código	48
Apéndice D Documentación de usuario	49
D.1. Introducción	49
D.2. Instalación Física	49
D.3. Instalación del software	54

D.4. Manual del usuario	57
Bibliografía	65

Índice de figuras

A.1. Compatibilidad de licencias.	12
C.1. Rama de directorios de hardware.	18
C.2. Rama de directorios de InverIoT.	20
C.3. Rama de directorios de dashboard.	21
C.4. descripcion	23
C.5. Conexión ssh permitida.	29
C.6. Mosquitto operativo	30
C.7. Descargar el firmware más reciente.	31
C.8. Conectar usb manteniendo presionado.	31
C.9. conexiones en el hardware.	32
C.10.DHT22: Sensor de humedad y temperatura del ambiente.	33
C.11.BH1750: Sensor de intensida de luz.	33
C.12.Sensor de humedad del suelo.	34
C.13.Pantalla oled de 128x64 píxeles.	34
C.14.Módulo led RGB.	36
C.15.Abrimos el configurador del interpreter.	36
C.16.Habilitamos Thonny para Micropython.	37
C.17.Ejecución y carga a un solo clic.	38
C.18.boot.py se ejecuta en el arranque.	38
C.19.Datos almacenados en tabla.	39
C.20.Dashboard en ejecución.	45
C.21.Data de sensores en Jupyter notebook	46
D.1. Descargar el firmware más reciente.	49
D.2. Conectar usb manteniendo presionado.	50
D.3. Abrimos el configurador del interpreter.	51
D.4. Habilitamos Thonny para Micropython.	51

D.5. Archivos necesarios para ejecutar en el hardware.	52
D.6. Archivo de configuración en Micropython.	52
D.7. Conexiones del hardware para el cliente.	54
D.8. Foto real de las conexiones.	57
D.9. Sensor de humedad de suelo operativo.	58
D.10.Pantalla oled mostrando los datos.	58
D.11.Interfaz de la aplicación de escritorio.	59
D.12.Activación de mecanismo con un clic.	59
D.13.Acesso al histórico de datos.	60
D.14.Capacidad de mostrar gráficas en un rango de fechas.	60
D.15.El color rojo indica se ha sobrepasado el umbral.	61
D.16.El color azul indica está por debajo del umbral.	61
D.17.Alertas recibidas por el bot de Telegram.	62
D.18.Capacidad de enviar comandos desde Telegram.	63

Índice de tablas

A.1. Coste de personal mensual	7
A.2. Coste total en personal durante el proyecto	8
A.3. Coste total en hardware	8
A.4. Tabla de costes totales	9
A.5. Dependencias en el servidor LAMP.	9
A.6. Dependencias en la Raspberry Pi Pico W.	10
A.7. Dependencias en nodeMqtt y dashboard web.	10
A.8. Dependencias en Aplicación de escritorio.	10
A.9. Licencias específicas.	11
A.10. Dependencias en la documentación.	11
A.11. Licencias aplicadas en el proyecto.	11
C.1. Directorios que contienen partes del proyecto.	17
C.2. Directorio: Firmware y reset.	18
C.3. Directorio: Fritzing.	19
C.4. Directorio: micropython.	19
C.5. Directorio: lib.	19
C.6. Directorio: nodeMqtt.	20
C.7. Directorio: InverIoT.	21
C.8. Directorio: dashboard.	22
C.9. Directorio: AnalisisDatos.	22
C.10. Directorio: TFG.	23
C.11. Directorio: img.	24
C.12. Directorio: tex.	24
C.13. Directorio: Totalidad del proyecto.	25
C.14. Características del servidor HP ProLiant.	25
C.15. Base de datos TFG_UBU.	27
C.16. Configuración de red interna.	28

C.17.Umbrales ideales para un invernadero de cannabis medicinal.	35
C.18.Indicadores por color en los leds RGB.	35
C.19.Topics MQTT usados en nodeMqtt	37
C.20.Topics MQTT usados en InverIoT	41
C.21.Topics MQTT usados en dashboard	42
C.22.Compilación y ejecución.	47

Apéndice A

Plan de Proyecto Software

A.1. Introducción

El objetivo fundamental de este documento es establecer una guía clara y estructurada que sirva como marco de referencia para el equipo de desarrollo, proporcionando lineamientos detallados que aseguren la ejecución exitosa del proyecto. A través de una cuidadosa planificación y coordinación, se busca garantizar la entrega oportuna y efectiva de un sistema robusto y funcional que cumpla con los requisitos establecidos y supere las expectativas del cliente.

Este proyecto ha documentado la mayoría de sus cambios y modificaciones en el historial de commits de GitHub, donde cada commit está registrado con fecha y hora. En algunas instancias, varios cambios fundamentales se agruparon en un solo commit, lo que podría dificultar la identificación de modificaciones individuales. Sin embargo, en la gran mayoría de los casos, los commits en GitHub representan cambios específicos con descripciones claras y precisas.

Aunque los commits de GitHub no detallan el tiempo real invertido en la realización del proyecto, el lapso de tiempo entre el primer y el último commit proporciona una estimación aproximada del tiempo total invertido.

A.2. Planificación temporal

Inicialmente hay 3 fechas clave:

- **03/10/2023:** Hice la propuesta a D. Carlos Cambra Baseca.

- **06/10/2023** Comentamos el proyecto por email entre D. Carlos y D. Alejandro Merino Gómez.
- **03/11/2023** Se realiza la primera reunión online por microsoft Teams.

Se optó por utilizar el **Modelo de Prototipos** [10] como enfoque para el desarrollo del proyecto. A continuación, se detallan los commits de GitHub que contribuyeron a cada fase del Modelo de Prototipos.

Requisitos de desarrollo:

- **20/01/2024 Initial commit:** La definición de requisitos era importante antes de iniciar el primer commit.
- **20/01/2024 Descripción inicial.**
- **22/01/2024 main.py:** Se empieza a programar alternativas para cumplir requisitos específicos.
- **22/01/2024 Actualización descripción general.**

Desarrollo del Prototipo:

- **22/01/2024 Librerías para el hardware:** Después de un descarte nos quedamos con las librerías que usaremos en el proyecto para la programación del hardware.
- **22/01/2024 actualización descripción.**
- **23/01/2024 actualizando diagrama de conexiones:** Vamos definiendo los gpio específicos para cada componente.
- **23/01/2024 fritzing: conexiones:** Mediante fritzing se va documentando el diagrama de conexiones en el hardware.
- **23/01/2024 imágenes de componentes y conexiones individuales.**

Evaluación del prototipo:

- **23/01/2024 corrección imagen de Raspberry Pi Pico W.**

- 23/01/2024 Cambio DHT11 por DHT22 para mayor precisión: Se cambia de sensor para tener mayor precisión en la toma de datos de temperatura y humedad del ambiente.
- 25/01/2024 Agregando MQ-135 Sensor de Calidad del aire: Se pretendo agregar un nuevo sensor para una nueva variable.
- 25/01/2024 activar actualizacion valor de sensor de calidad del aire MQ-135.
- 26/01/2024 libreria mq135.py.
- 26/01/2024 Agregando batería li-ion.

Modificación:

- 27/01/2024 modificación clase de sensor de humedad de suelo.
- 27/01/2024 conexión wifi.
- 28/01/2024 utelegram.
- 29/01/2024 Create Readme.md.
- 29/01/2024 creando carpeta micropython para ubicar el código correspondiente.

Documentación:

- 29/01/2024 TFG.
- 29/01/2024 actualizando datos.
- 29/01/2024 Ubicación de carpeta micropython.
- 29/01/2024 borrando carpeta: AppWinDesktop.
- 30/01/2024 corrección nombre: diagramas.
- 30/01/2024 agregando fotos.
- 31/01/2024 TFG: Introducción, Objetivos del proyecto, Conceptos Teóricos.
- 31/01/2024 actualización de esquema de conexiones.

- 31/01/2024 eliminando concepto: Domótica.
- 31/01/2024 corrección de nombre identificador de los leds.
- 31/01/2024 acualización código micropython.
- 01/02/2024 eliminando archivos innecesarios.
- 01/02/2024 coloco librería umqtt en carpeta lib para mayor orden y corrección en main.py.
- 02/02/2024 Herramientas: IDE Thonny.
- 03/02/2024 Subido archivo de audio creado con IA.
- 03/02/2024 Presentación del TFG por Iker Jiménez (IA).
- 03/02/2024 Update README.md.
- 04/02/2024 Update README.md.
- 04/02/2024 Update README.md.
- 03/02/2024 TFG: Técnicas y Herramientas-Entorno físico.
- 05/02/2024 Análisis de datos.
- 05/02/2024 Update README.md.
- 06/02/2024 AnalisisDatos: Regresión lineal.
- 06/02/2024 explicación del coeficiente de determinación.
- 06/02/2024 TFG: 4-Técnicas y herramientas : jupyter notebook.
- 07/02/2024 TFG: 5 Aspectos relevantes del desarrollo del proyecto.
- 07/02/2024 Update README.md.
- 07/02/2024 AnalisisDatos: Resumen estadístico.
- 08/02/2024 Update dht.py.
- 08/02/2024 Update sh1106.py.
- 08/02/2024 Update utelegram.py.
- 08/02/2024 Update umqtt.py.

- 08/02/2024 Update umqtt.py.
- 08/02/2024 Update utelegram.py.
- 08/02/2024 Update main.py.
- 08/02/2024 Update main.py.
- 08/02/2024 Update sh1106.py.
- 08/02/2024 Update README.md.
- 08/02/2024 TFG: 5 Aspectos relevantes del desarrollo del proyecto : Desarrollo del proyecto.
- 08/02/2024 TFG: 5 Aspectos relevantes del desarrollo del proyecto : nodeMqtt.
- 09/02/2024 TFG: img/diagramas/mqtt_{dashboard}.
- 10/02/2024 TFG: memoria versión 1.0.
- 10/02/2024 TFG: memoria: corrección en imágenes.
- 10/02/2024 Update README.md.
- 10/02/2024 Update README.md.
- 10/02/2024 Updates from Overleaf.
- 10/02/2024 Merge overleaf-2024-02-10-1155 into main.
- 10/02/2024 Arreglado algunos signos de puntuación..
- 11/02/2024 Reorganizada alguna palabra..
- 10/02/2024 TFG: agregando imágenes de conexiones con panel solar y power bank.
- 11/02/2024 TFG: esquema InverIoT.
- 11/02/2024 Agregando tabla README.
- 11/02/2024 Update README.md Añadido video presentación.
- 12/02/2024 Create Readme.md Firmware y reset.
- 12/02/2024 Add files via upload.

- 12/02/2024 Update Readme.md.
- 12/02/2024 Update Readme.md.
- 12/02/2024 Update README.md Añadido vídeo demo funcional.
- 12/02/2024 Hardware: Modularización de código.

Pruebas:

- 12/02/2024 Hardware: corrección en uso de variable `next_send`.
- 12/02/2024 Hardware: Corrección libreria umqtt.
- 12/02/2024 Hardware: Corrección de librería utelegram.py.
- 12/02/2024 Hardware: Corrección librería dht.
- 12/02/2024 TFG: archivos innecesarios.
- 12/02/2024 Hardware: agregando `wlan_config`.
- 12/02/2024 Hardware: cambio de key en diccionario.
- 12/02/2024 Hardware: eliminando comentarios innecesarios.
- 13/02/2024 Hardware: ordenando main.py.
- 13/02/2024 TFG: diagramas de InverIoT y dashboard.
- 13/02/2024 TFG: Actualización de diagramas.
- 14/02/2024 Update README.md Actualizada descripción.
- 14/02/2024 memoria versión 1.1.

A.3. Estudio de viabilidad

El estudio de viabilidad es una fase esencial en cualquier proyecto, ya que tiene como objetivo principal evaluar la aplicabilidad del mismo. Además, desempeña un papel crucial al proporcionar claridad sobre aspectos fundamentales que facilitan la toma de decisiones.

Viabilidad económica

En esta etapa, abordaremos los aspectos económicos del proyecto, que desempeñan un papel fundamental para determinar la viabilidad del mismo en un contexto empresarial y profesional.

En esta sección, se detallan los costos asociados al proyecto, destacando el esfuerzo por minimizarlos en todas las áreas, desde la instalación hasta el material utilizado.

Coste de personal

La sección de costos de personal destaca por presentar el mayor gasto en comparación con otros aspectos económicos del proyecto.

Se ha considerado un salario de 22.700€ brutos anuales, distribuidos en 12 pagas, para el período comprendido entre el 20 de enero de 2024 y el 14 de febrero de 2024. Al calcular los 25 días de duración del proyecto, se observa que equivale a 0,83 meses de trabajo, resultando en un gasto total en personal de 2459,16€.

El desglose mensual se presenta en la tabla A.1, y el costo total se detalla en la tabla A.2.

Concepto	Porcentaje	Coste
Sueldo Base (Neto)		1891,66 €
Contingencias Comunes	23,60 %	446,43 €
Tipo general	5,50 %	104,04 €
Fogasa	0,20 %	3,78 €
FP	0,70 %	13,24 €
Total Gasto Mensual (Suma)		2.459,16 €

Tabla A.1: Coste de personal mensual

Concepto	Coste
Número de meses	0.8 meses
Gasto mensual	2.459,16 €
Total Gasto Proyecto	2.041,10 €

Tabla A.2: Coste total en personal durante el proyecto

Para realizar el cálculo me he apoyado en la documentación oficial que tiene a disposición del ciudadano la Seguridad Social en su página web [18]. En ella podemos ver que hay que aportar un 23,60 % en concepto de Contingencias comunes, un 5,50 % en concepto de desempleo de Tipo general, un 0,20 % a FOGASA y un 0,70 % para FP.

Coste hardware

Los esfuerzos para optimizar los costos asociados con el hardware han arrojado resultados positivos.

Se ha seleccionado la Raspberry Pi Pico W debido a su asequibilidad, conectividad Wi-Fi y bajo consumo de energía.

Concepto	Coste
Raspberry Pi Pico W	16,99 €
Módulo LED RGB	3,29 €
Pantalla OLED I2C de 128x64 píxeles	8,49 €
Protoboard y cables	13,99 €
DHT22 Sensor de Temperatura y Humedad	9,49 €
Sensor de humedad del suelo V1.2	4,99 €
Sensor de luz GY-302 BH1750	5,99 €
Módem WiFi portatil 4G	21,26 €
Panel Solar de 8W IP65	17,95 €
Batería externa 20000 mAh	29,28 €
Total	131,72 €

Tabla A.3: Coste total en hardware

Coste software

La única inversión adicional es la licencia del programa Fritzing, con un costo total de 8 € [7].

Coste Total

La suma asciende a **2180,82 €** que han quedado descritos en la tabla A.4.

Concepto	Coste
Personal	2.041,10 €
Hardware	131,72 €
Software	8 €
Total	2180,82 €

Tabla A.4: Tabla de costes totales

Viabilidad legal

En este apartado, se proporciona un análisis del marco legal del proyecto. Para comprenderlo completamente, es esencial definir qué es una licencia:

Servidor LAMP

Nombre	Versión	Licencia	Descripción
Linux Ubuntu	23.10.1	GPL	Sistema Operativo del servidor
Apache2	2.4.57	Apache license 2.0	HTTP Server
Mysql	8.0.35	GPL	Servidor de base de datos
PHP	8.2.18	PHP Licence 3.01	Lenguaje para crear sitios web
ssh	9.2	BSD	Protocolo para administración remota
vsftpd	3.0.3	GPL2	Servidor FTP
ufw	0.36.2	GPL2	administrador de firewalls
node.js	20.10.0	MIT	Entorno de ejecución de Javascript
pm2	5.3.1	MIT	Administrador de procesos para Node.js
mosquitto	2.0.11	EPL 1.0	Broker MQTT

Tabla A.5: Dependencias en el servidor LAMP.

Hardware

Principalmente es el firmware que permite programar con Micropython [5] en la Raspberry Pi Pico W y las librerías usadas.

Fritzing [7] fue usado para hacer el diseño de las conexiones.

Nombre	Versión	Licencia	Descripción
Firmware Micropython	1.22.1	MIT	Python para microcontroladores
IDE Thonny	4.1.4	GPL3	IDE para micropython
urllib3	1.26.9	Apache2.0	Solicitudes HTTP

umqtt	1.4.6	MIT	MQTT para micropython
urequests	0.8.0	MIT	Solicitudes HTTP
machine	0.0.1	MIT	Control de hardware
dht	0.1.0	MIT	Librería para sensor DHT22 y DHT11
utelegram	2.1.1	MIT	Wrapper para la API de Telegram
sh1106	1.5.0	MIT	Librería para pantalla oled
network	0.1	MIT	Networking library
time	0.1.0	MIT	Agregar delay
utime	1.1.7	MIT	Hora actual e intervalos
Fritzing	1.0.2	GPL3	Diseño electrónico

Tabla A.6: Dependencias en la Raspberry Pi Pico W.

NodeMqtt y Dashboard

Nombre	Versión	Licencia	Descripción
MQTT.js	4.10.0	Apache 2.0	Protocolo MQTT para Node.js
Mysql2	3.9.1	MIT	Mysql para Node.js
Express	4.18.1	MIT	Gestiona rutas y peticiones HTTP
Socket.IO	4.5.4	MIT	Comunicación bidireccinal
CORS	2.8.5	MIT	Configuración de CORS

Tabla A.7: Dependencias en nodeMqtt y dashboard web.

Aplicación de Escritorio

Nombre	Versión	Licencia	Descripción
C#	11	Ms-PL	Lenguaje de programación
.NET	7	MS-PL	Creación de aplicaciones con C#

Tabla A.8: Dependencias en Aplicación de escritorio.

Bot de Telegram

En el proyecto se ha utilizado un bot de Telegram para el envío de alertas y poder realizar acciones mediante comandos. Podemos ver las licencias del sistema que utilizamos en la tabla A.9.

Nombre	Licencia	Descripción
Telegram Code	MIT	Código interno de Telegram

Telegram App [21]	GPLv2 y post	Aplicaciones móviles
Telegram Api [20]	GPLv3 excepto OpenSSL	API pública de Telegram

Tabla A.9: Licencias específicas.

Documentación

Para la generación de la documentación, se han aplicado dos licencias: una para L^AT_EX y otra para la aplicación Fritzing. Los diagramas electrónicos del proyecto se han creado bajo la licencia Creative Commons by-sa [3].

La documentación del proyecto comparte la misma cobertura CC by-sa, lo que permite el uso comercial y la distribución tanto de la obra original como de sus derivados, siempre y cuando se mantenga la misma licencia que rige la obra original.

Nombre	Licencia	Descripción
L ^A T _E X [13]	LPPL	Procesador de textos
GIMP	GPL3	Editor de imágenes
Fritzing	CC by-sa	Diagramas electrónicos

Tabla A.10: Dependencias en la documentación.

Resumen del licenciamiento del presente proyecto

Tras analizar la compatibilidad de las licencias de las dependencias de nuestro proyecto, hemos determinado que la licencia GPL3 es la más adecuada. Por lo tanto, el software de nuestro proyecto se distribuirá bajo la licencia GPL3 [8].

En resumen estamos empleando la licencia GPL3 [8] y CC-BY-SA-3.0 [3].

Recurso	Licencia
Código Fuente	GPL3
Documentación	CC-BY-SA-3.0
Imágenes	CC-BY-SA-3.0

Tabla A.11: Licencias aplicadas en el proyecto.

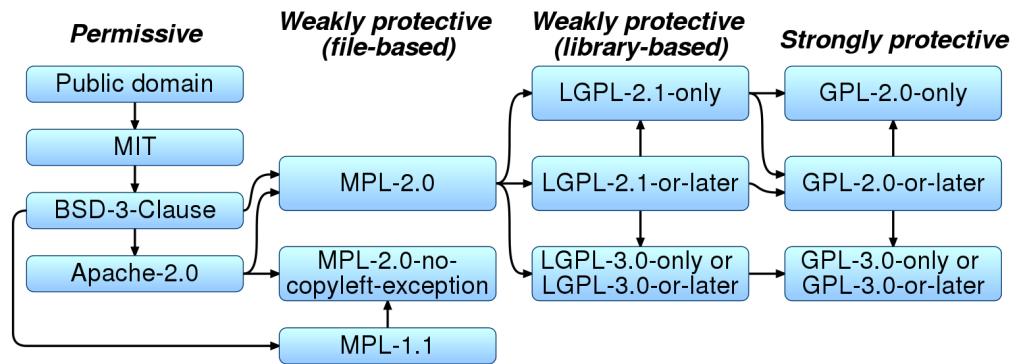


Figura A.1: Compatibilidad de licencias.



Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apartado se definen los requisitos que debe cumplir el proyecto. Estos requisitos determinarán las restricciones, funcionalidades y comportamiento del mismo.

B.2. Objetivos generales

Los objetivos generales del proyecto definen las funcionalidades que este debe ofrecer, las cuales se detallan a continuación.

- Crear un sistema de monitorio para un invernadero de cannabis medicinal.
- El sistema debe recopilar datos de la humedad y temperatura del ambiente, intensidad de luz y humedad del suelo.
- Se debe implementar opciones de visualización de la información in situ, a través de un teléfono móvil y a través de un navegador web.
- El hardware utilizado en el proyecto debe ser eficiente respecto a consumo de energía y económico.
- Los datos deben ser almacenados para consultas del histórico de datos.

B.3. Catálogo de requisitos

A continuación se detallan los requisitos específicos del proyecto, a partir de los objetivos generales mencionados anteriormente.

Requisitos funcionales

- **RF-1 Obtención de datos:** El Sistema debe ser capaz de captar los datos respecto a características específicas en el invernadero.
 - **RF-1.1 Obtención de la humedad del ambiente:** Se debe captar la humedad relativa del ambiente, y su valor está representado mediante un porcentaje.
 - **RF-1.2 Obtención de la temperatura del ambiente:** Se debe captar la temperatura del ambiente en grados Celsius.
 - **RF-1.3 Obtención de la intensidad de la luz:** La intensidad de la luz debe medirse en lux.
 - **RF-1.3 Obtención de la humedad del suelo:** La humedad del suelo debe medirse en porcentaje de humedad gravimétrica, está representado por un porcentaje.
- **RF-2 Indicadores de condiciones críticas:** Diversas maneras de observar si los valores están fuera del umbral establecido.
 - **RF-2.1 Alertas:** Se deben recibir alertas al teléfono celular.
 - **RF-2.2 Indicador visual in situ:** Dentro del invernadero deben haber indicadores visuales que indiquen que los valores medidos están fuera del umbral.
- **RF-3 Visualización de datos:** Los datos deben estar representados por números con sus respectivas unidades o gráficas.
 - **RF-3.1 In situ:** Los valores de los sensores deben verse reflejados en números y sus unidades.
 - **RF-3.2 Fuera del invernadero:** A través de acceso a internet.
- **RF-4 Almacenaje de la data:** Se debe tener la capacidad de guardar los datos de los sensores y los umbrales establecidos.
 - **RF-4.1 Robustez:** El usuario no debe de preocuparse por la integridad de los datos si alguna parte del sistema en el invernadero falla.

- **RF-4.1 Modificable:** El usuario debe poder modificar los valores de los umbrales.
- **RF-5 Capacidad de enviar comandos:** Se debe tener la capacidad de enviar comandos para acciones específicas.
 - **RF-5.1 Activación de un mecanismo:** Mediante un comando el Sistema debe poder activar y desactivar un mecanismo asociado a uno de los sensores conectados.
 - **RF-5.2 Consulta de la data en tiempo real:** El usuario debe poder consultar los valores de los sensores en tiempo real usando comandos.

Requisitos no funcionales

- **RNF-1 Escalabilidad:** Debe poder ampliarse fácilmente.
- **RNF-2 Eficiencia:** Debe minimizar el consumo energético del Sistema.
- **RNF-3 Rendimiento:** El sistema debe ser fluido y evitar cargas innecesarias.
- **RNF-4 Usabilidad:** Debe ser fácil de utilizar e intuitivo, y adaptado a las necesidades que pretende cubrir.
- **RNF-5 Disponibilidad:** El Sistema debe estar funcionamiento correctamente.
- **RNF-6 Durabilidad:** El software y hardware deben poder funcionar correctamente durante un tiempo relativamente largo.
- **RNF-7 Capacidad:** Debe poder captar la información necesaria y actuar conforme a lo que se espera de él.
- **RNF-8 Documentación:** Debe existir la documentación suficiente para poder implementar e interactuar con el Sistema.
- **RNF-9 Operabilidad:** Debe permitir controlar y manejar el Sistema según los requisitos funcionales.
- **RNF-10 Mantenibilidad:** Debe desarrollarse de tal manera que el mantenimiento sea lo más fácil y rápido posible.

- **RNF-11 Seguridad:** Todas las operaciones deben ser seguras y estar cifradas.
- **RNF-12 Legibilidad:** El software debe ser fácilmente legible.
- **RNF-13 Extensibilidad:** El código debe ser fácilmente adaptable y reutilizable.
- **RNF-14 Respaldo documental:** Toda la instalación debe realizarse conforme a los estándares legales vigentes.

Apéndice C

Documentación técnica de programación

C.1. Introducción

En esta sección encontrarás información detallada sobre la estructura del código. Esta documentación está diseñada para brindar claridad y orientación a los desarrolladores, facilitando el mantenimiento, la expansión y la comprensión del código fuente del proyecto.

C.2. Estructura de directorios

Cada parte del proyecto está contenido en una carpeta.

Tabla C.1: Directorios que contienen partes del proyecto.

Carpeta	Descripción
Hardware	Implementación del hardware
nodeMqtt	Servicio para almacenar datos en una base de datos
InverIoT	Aplicación de escritorio
dashboard	Dashboard web accesible desde internet
AnalisisDatos	Limpieza de datos y resumen estadístico
TFG	Documentación del proyecto

Hardware

Contiene detalles de las conexiones y código de programación en MicroPython [5].

El directorio **fritzing** C.3 contiene los esquemas de conexiones entre la Raspberry Pi Pico W, los módulos, sensores y leds RGB.

El directorio **micropython** C.4 contiene el código de programación para controlar el hardware.

Respecto al directorio **lib** C.5 los archivos que empiezan con "mi_" son implementaciones más que hizo posible reducir la cantidad de líneas de código de programación en el archivo **main.py**.

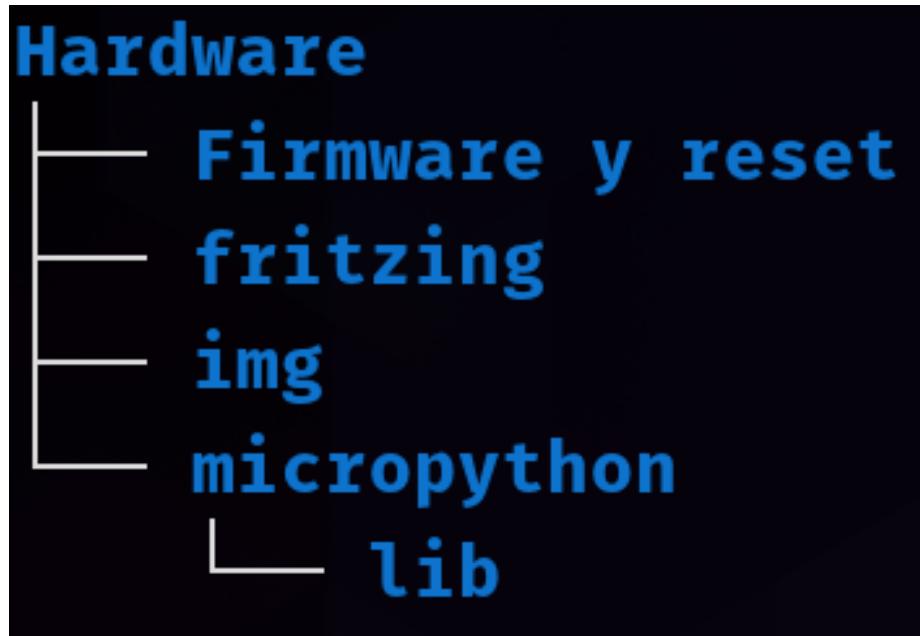


Figura C.1: Rama de directorios de hardware.

Tabla C.2: Directorio: Firmware y reset.

Nombre de Archivo	Descripción
flash_nuke.uf2	Firmware para configuración de fábrica
RPI_PICO_W-20240105-v1.22.1.uf2	Firmware oficial de la RPI PICO W

Tabla C.3: Directorio: Fritzing.

Nombre de Archivo
BH1750.fzz
conexiones.fzz
DHT22_humedad_temperatura.fzz
leds_rgb.fzz
oled_i2c_168x64.fzz
Sensor_humedad_suelo.fzz

Tabla C.4: Directorio: micropython.

Nombre de Archivo	Descripción
boot.py	Se ejecuta al prender la RPI PICO W.
config.py	Configuración wifi, mqtt, etc.
lib	librerías y funciones implementadas
main.py	Programa principal

Tabla C.5: Directorio: lib.

Nombre de Archivo	Descripción
bh1750.py	librería para sensor BH1750 [9]
dht.py	librería para sensor DHT22 [19]
mi_modulos.py	Mi implementación para pantalla oled y leds RGB
mi_mqtt.py	Mi implementación para MQTT
mi_sensores.py	Mi implementación para sensores
mi_telegram.py	Mi implementación para bot de Telegram
mi_wifi.py	Mi implementación para conectarse al wifi
sh1106.py	librería para pantalla oled [26]
umqtt.py	librería para MQTT [25]
utelegram.py	librería para bot de Telegram [22]

nodeMqtt

NodeMtt recopila los datos mediante MQTT y los almacena en una base de datos Mysql [16].

El directorio **nodeMqtt** no contiene subdirectorios dentro pero contiene los archivos indicados en la tabla C.6.

Tabla C.6: Directorio: nodeMqtt.

Nombre de Archivo	Descripción
index.js	Archivo javascript principal
package.json	Información de dependencias

InverIoT

Es una Aplicación de escritorio para el sistema operativo Windows.

Fue desarrollada usando el lenguaje de programación C#, permite ver los datos en tiempo real, el histórico de datos y activar mecanismos (representado por la activación de un led verde).

La obtención de datos en tiempo real y las órdenes para activar mecanismos se hace por medio de MQTT [12].

La obtención del histórico de datos se hace por medio del protocolo Mysql.

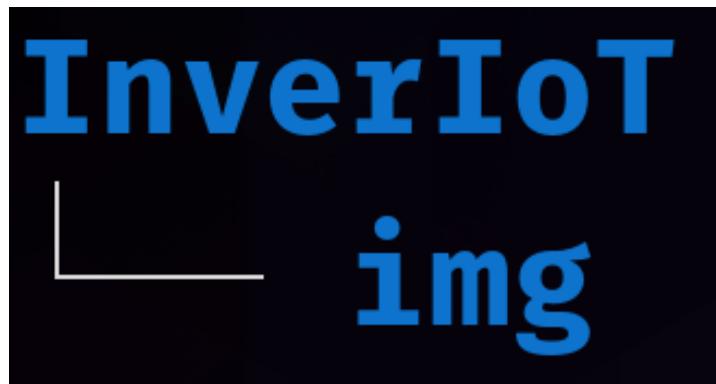


Figura C.2: Rama de directorios de InverIoT.

Tabla C.7: Directorio: InverIoT.

Nombre de Archivo
favicon.ico
frmGraficas.cs
frmGraficas.Designer.cs
frmGraficas.resx
frmHistorico.cs
frmHistorico.Designer.cs
frmHistorico.resx
frmMain.cs
frmMain.Designer.cs
frmMain.es-ES.resx
frmMain.resx
Funciones.cs
img
'Instalador - InverIoT - v2.2.exe'
InverIoT.csproj
InverIoT.sln
Program.cs
README.md

dashboard

Este dashboard web permite acceder a los datos en tiempo real y también ver el histórico de datos. No envía órdenes, solo recibe datos mediante MQTT [12] y MySQL [16].

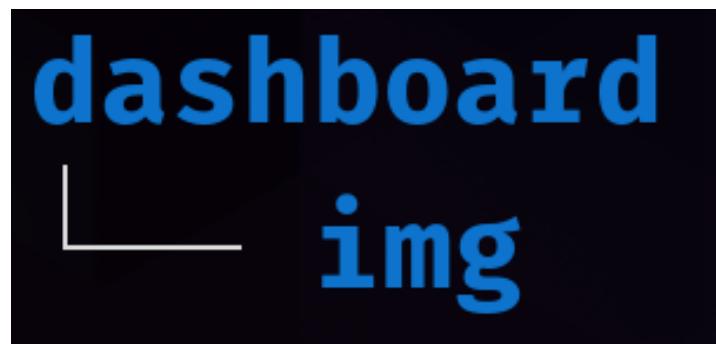


Figura C.3: Rama de directorios de dashboard.

Tabla C.8: Directorio: dashboard.

Nombre de Archivo	Descripción
android-chrome-192x192.png	Ícono
android-chrome-512x512.png	Ícono
apple-touch-icon.png	Ícono
favicon-16x16.png	Ícono
favicon-32x32.png	Ícono
favicon.ico	Ícono
historico.html	Estructura visual del historial
img	Imágenes del dashboard funcionando
index.html	Estructura visual por defecto
package.json	Dependencias y configuraciones
servers.js	Hace funcionar el dashboard
site.webmanifest	Manifiesto web

AnalisisDatos

Limpia los datos y da un resumen estadístico usando Jupyter Notebook [17] y el lenguaje de programación Python.

El archivo **analisis.ipynb** contiene todo el código de programación utilizado en el proceso.

El archivo **data.csv** es solo una muestra representativa de los datos, los cuales siguen creciendo conforme siga en funcionamiento el sistema.

La librería de python que principalmente fue usada es **pandas**.

La recopilación de datos abrirá otras posibilidades de investigación para futuros análisis usando herramientas como machine learning. Por ahora solo nos centramos en almacenar los datos y ver las anomalías que puedan surgir antes de hacer un análisis, es decir, respecto a la limpieza de datos.

Tabla C.9: Directorio: AnalisisDatos.

Nombre de Archivo	Descripción
analisis.ipynb	Jupyter notebook para analizar datos
data.csv	Data de los sensores
img	Directorio de imágenes

TFG

Documentación del proyecto usando L^AT_EX.

Los archivos principales son **memoria.tex** y **anexos.tex** como se puede ver en la tabla C.10.

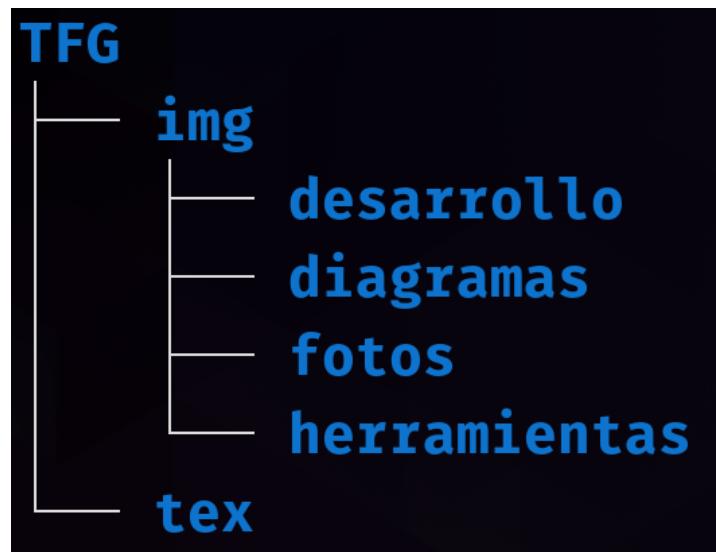


Figura C.4: descripcion

Tabla C.10: Directorio: TFG.

Nombre de Archivo	Descripción
anexos.pdf	PDF de anexos
anexos.tex	documentación técnica
bibliografiaAnexos.bib	Referencia bibliográficas de anexos.tex
bibliografia.bib	Referencia bibliográficas de memoria.tex
img	directorio de imágenes
memoria.pdf	Resultado de memoria.tex
memoria.tex	Descripción de conceptos y del proyecot en sí
tex	Archivos usados por memoria.tex y anexos.tex

Tabla C.11: Directorio: img.

Nombre de Archivo	Descripción
cabecera.pdf	Cabecera con el nombre de la universidad
desarrollo	Imágenes que representan el proceso
diagramas	Esquemas del hardware y diagramas
escudoInfor.pdf	Escudo de la universidad
fotos	Fotos relacionadas al proyecto
herramientas	Imágenes de herramientas usadas

Tabla C.12: Directorio: tex.

Nombre de Archivo
1_Introduccion.tex
2_Objetivos_del_proyecto.tex
3_Conceptos_teoricos.tex
4_Tecnicas_y_herramientas.tex
5_Aspectos_relevantes_del_desarrollo_del_proyecto.tex
6_Trabajos_relacionados.tex
7_Conclusiones_Lineas_de_trabajo_futuras.tex
A_Plan_proyecto.tex
B_Requisitos.tex
C_Diseno.tex
D_Manual_programador.tex
E_Manual_usuario.tex
F_ODS.tex

C.3. Manual del programador

Este apartado describe los recursos y las estrategias para el desarrollo del código, incluyendo una explicación detallada de las diferentes partes del proyecto.

Veremos las instalaciones y configuraciones necesarias para todo funcione correctamente, tanto referente a software y hardware.

El usuario final no tiene porque hacer todas estas configuraciones, él solo tendrá que ejecutar ciertas cosas específicas y muy puntuales.

Se va explicar respecto a las partes que conforman la totalidad de este proyecto:

Tabla C.13: Directorio: Totalidad del proyecto.

Parte del proyecto	Descripción
Servidor LAMP	Gestiona la base de datos y MQTT
Hardware	Implementación del hardware
nodeMqtt	Servicio para almacenar datos en una base de datos
InverIoT	Aplicación de escritorio
dashboard	Dashboard web accesible desde internet
AnalisisDatos	Limpieza de datos y resumen estadístico
Bot de Telegram	Recibe alertas y envía comandos

Servidor LAMP

Se está usando un servidor físico HP Proliant [11], mediante Hyper-V se ha virtualizado el sistema operativo Ubuntu para el servidor. Se ha hecho esto porque el servidor físico tiene como sistema operativo principal a Windows Server.

El servidor LAMP consta principalmente de un sistema operativo Linux, que en este caso es Ubuntu [24], Apache [1], Mysql [16] y PHP.

Adicionalmente se instalaron otros servicios que se vieron necesarios. La instalación en Ubuntu se hace mediante comandos en la terminal. La descripción de servicio instalado se menciona en la tabla A.5.

Tabla C.14: Características del servidor HP ProLiant.

Característica	Descripción
Sistema operativo	Windows Server 2019 Standard
Fabricante	Hewlett Packard Enterprise
Modelo	Hewlett Packard Enterprise x64 Class PC
Procesador	Intel(R) Xeon(R) Silver 4210R CPU @ 2.4GHz 2.39GHz
Memoria instalada (RAM)	128 GB (128 GB utilizable)
Tipo de sistema	Sistema operativo de 64 bits, procesador x64

Código C.1: Comando para instalar apache2 .

```
o sudo apt install apache2
```

Código C.2: Comando para instalar Mysql.

```
o sudo apt install mysql-server
```

Código C.3: Comando para instalar PHP.

```
o sudo apt install php
```

Código C.4: Comando para instalar ssh.

```
o sudo apt install ssh
```

Código C.5: Comando para instalar phpmyadmin.

```
o sudo apt install phpmyadmin
```

Código C.6: Comando para instalar vsftpd.

```
o sudo apt install vsftpd
```

Código C.7: Comando para instalar ufw.

```
o sudo apt install ufw
```

Código C.8: Comando para instalar node.js.

```
o sudo apt install nodejs
```

Código C.9: Comando para instalar pm2.

```
o sudo apt install pm2
```

Código C.10: Comando para instalar mosquitto.

```
o sudo apt install mosquitto
```

Código C.11: Comando para instalar npm.

```
o sudo apt install npm
```

Código C.12: Comando para instalar express.

```
o sudo apt install express
```

Ahora usando mysql crearemos la base de datos y tablas correspondientes.

Tabla C.15: Base de datos TFG_UBU.

Nombre de tabla	Descripción
sensores	Captura la fecha, hora y valores de sensores
umbrales	Captura los umbrales

Código C.13: Comando para acceder a Mysql.

```
0 sudo mysql
```

Código C.14: Comandos Mysql para crear Database **TFG_UBU**.

```
0 CREATE DATABASE TFG_UBU;
1 CREATE USER 'joseluis'@'%' IDENTIFIED BY 'Mipassword';
2 GRANT ALL ON TFG_UBU.* TO 'joseluis'@'%';
3 use TFG_UBU;
```

Código C.15: Comandos Mysql para crear tabla **umbrales**.

```
0 CREATE TABLE 'umbrales' (
1   'temperatura_minima' float NOT NULL,
2   'temperatura_maxima' float NOT NULL,
3   'humedad_ambiente_minima' float NOT NULL,
4   'humedad_ambiente_maxima' float NOT NULL,
5   'luminosidad_minima' float NOT NULL,
6   'luminosidad_maxima' float NOT NULL,
7   'humedad_suelo_minima' float NOT NULL,
8   'humedad_suelo_maxima' float NOT NULL,
9   'fecha_actualizacion' timestamp
10  NULL DEFAULT CURRENT_TIMESTAMP
11  ON UPDATE CURRENT_TIMESTAMP
12 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
13 INSERT INTO 'umbrales' (
14   'temperatura_minima', 'temperatura_maxima',
15   'humedad_ambiente_minima', 'humedad_ambiente_maxima',
16   'luminosidad_minima', 'luminosidad_maxima',
17   'humedad_suelo_minima', 'humedad_suelo_maxima') VALUES
18   (30, 35, 30, 70, 30, 150, 20, 80);
```

Código C.16: Comandos Mysql para crear tabla **sensores**.

```
0 CREATE TABLE 'sensores' (
```

```

1   'id' int NOT NULL AUTO_INCREMENT,
2   'fecha' date DEFAULT NULL,
3   'hora' time DEFAULT NULL,
4   'temperatura' decimal(4,2) DEFAULT NULL,
5   'humedad' decimal(4,2) DEFAULT NULL,
6   'intensidad_luz' decimal(4,2) DEFAULT NULL,
7   'humedad_suelo' decimal(4,2) DEFAULT NULL,
8   PRIMARY KEY ('id')
9 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

Teniendo en cuenta una configuración de red interna específica, se tiene que reaperturar y redirigir los puertos desde el router para tener acceso al servidor desde el exterior (ip pública), ya que como existen otros servidores sirviendo alguno de los mismos datos, necesitamos redirigir el servicio por puerto.

Tabla C.16: Configuración de red interna.

Característica	Valor
inet	192.0.0.6
netmask	255.255.255.0
broadcast	192.0.0.255

nat server protocol tcp global interface LoopBack 0 3001 inside 192.0.0.6 ftp	0
nat server protocol tcp global interface LoopBack 0 3002 inside 192.0.0.6 22	1
nat server protocol udp global interface LoopBack 0 3002 inside 192.0.0.6 22	2
nat server protocol tcp global interface LoopBack 0 3003 inside 192.0.0.6 www	3
nat server protocol udp global interface LoopBack 0 3003 inside 192.0.0.6 80	4
nat server protocol udp global interface LoopBack 0 3001 inside 192.0.0.6 21	5
nat server protocol tcp global interface LoopBack 0 8080 inside 192.0.0.6 8080	6
nat server protocol udp global interface LoopBack 0 8080 inside 192.0.0.6 8080	7
nat server protocol tcp global interface LoopBack 0 135 inside 192.0.0.6 135	8

```

nat server protocol udp global interface LoopBack 0      9
  135 inside 192.0.0.6 135
nat server protocol tcp global interface LoopBack 0     10
  3307 inside 192.0.0.6 3307
nat server protocol udp global interface LoopBack 0     11
  3307 inside 192.0.0.6 3307
nat server protocol tcp global interface LoopBack 0     12
  1883 inside 192.0.0.6 1883
nat server protocol udp global interface LoopBack 0     13
  1883 inside 192.0.0.6 1883
nat server protocol udp global interface LoopBack 0     14
  3000 inside 192.0.0.6 3000
nat server protocol tcp global interface LoopBack 0     15
  3000 inside 192.0.0.6 3000
nat server protocol tcp global interface LoopBack 0     16
  30308 inside 192.0.0.6 3308
nat server protocol udp global interface LoopBack 0     17
  30308 inside 192.0.0.6 3308

```

Código C.17: Comandos para reaperturar y redirigir puertos.

```
sudo ufw enable
```

Código C.18: Comando para activar el firewall.

```
sudo ufw allow ssh
```

Código C.19: Comando para permitir conexión ssh.

```
joseluis@TFG-UBU: ~
joseluis@TFG-UBU: ~$ sudo ufw show added
Reglas añadidas del usuario (vea «ufw status» para ejecutar el cortafuegos):
ufw allow 20/tcp
ufw allow 21/tcp
ufw allow 3002/tcp
ufw allow 3002/udp
ufw allow 3000/tcp
ufw allow 3000/udp
ufw allow 3001/tcp
ufw allow 3001/udp
ufw allow 3307/tcp
ufw allow 3307/udp
ufw allow 1883/tcp
ufw allow 1883/udp
ufw allow 22/udp
ufw allow 22/tcp
ufw allow 80/tcp
ufw allow 80/udp
ufw allow 8080/udp
ufw allow 3007/tcp
ufw allow 3007/udp
ufw allow 3003
```

Figura C.5: Conexión ssh permitida.

Ahora necesitamos instalar **mosquitto** [15] para la conectividad por MQTT [12].

MQTT nos servirá para la transmisión de datos en tiempo real, mediante los topics correspondientes.

El servidor LAMP será el Broker MQTT, se encarga de gestionar las peticiones.

```
sudo apt install mosquitto
```

Código C.20: Comando para instalar mosquitto.

```
sudo apt install mosquitto-clients
```

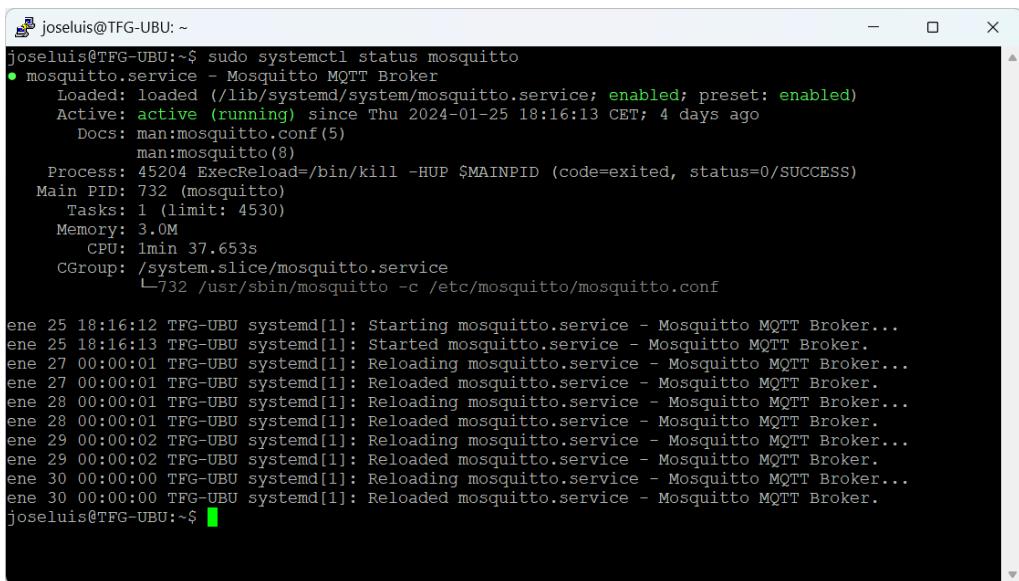
Código C.21: Comando para instalar mosquitto-clients.

```
sudo apt install mosquitto-clients
```

Código C.22: Comando para instalar mosquitto-clients.

```
sudo systemctl status mosquitto
```

Código C.23: Comando para comprobar el estado de mosquitto.



```
joseluis@TFG-UBU:~$ sudo systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT Broker
    Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; preset: enabled)
    Active: active (running) since Thu 2024-01-25 18:16:13 CET; 4 days ago
      Docs: man:mosquitto.conf(5)
             man:mosquitto(8)
   Process: 45204 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCESS)
     Main PID: 732 (mosquitto)
        Tasks: 1 (limit: 4530)
       Memory: 3.0M
          CPU: 1min 37.653s
        CGrou...
```

The terminal window shows the output of the command 'sudo systemctl status mosquitto'. It displays the status of the 'mosquitto' service, which is active and running. It provides detailed information about the service, including its process ID (732), memory usage (3.0M), and CPU usage (1min 37.653s). The log output shows several reload events and successful exits.

Figura C.6: Mosquitto operativo

Hardware

Se tiene que instalar el firmware en la Raspberry Pi Pico W para que podamos programarlo mediante Micropython. Primero descargamos el firmware de Micropython desde su página oficial [14].

Para fines prácticos nombraremos al firmware descargado: **rp2-pico.uf2**.

Firmware

Releases

- [v1.22.1 \(2024-01-05\) .uf2 / \[Release notes\] \(latest\)](#) 
- [v1.22.0 \(2023-12-27\) .uf2 / \[Release notes\]](#)
- [v1.21.0 \(2023-10-05\) .uf2 / \[Release notes\]](#)
- [v1.20.0 \(2023-04-26\) .uf2 / \[Release notes\]](#)

Figura C.7: Descargar el firmware más reciente.

Conectar el puerto usb de la Raspberry Pi Pico W manteniendo presionado el botón **BOOTSEL**.

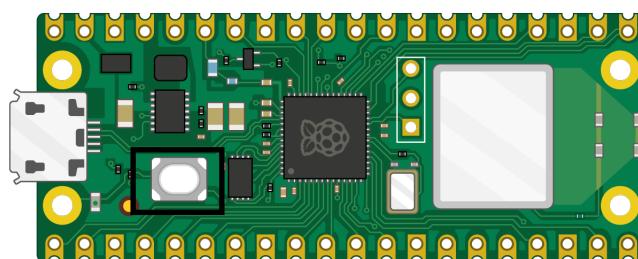


Figura C.8: Conectar usb manteniendo presionado.

Se ha creado un script en bash al que llamamos **instaladorFirmware.sh**. La ejecución de este permitirá instalar el firmware de MicroPython en la Raspberry Pi Pico W.

Código C.24: Script en bash para instalar el firmware MicroPython.

```

0 puerto=$(sudo dmesg | tail | grep -o 'sd[b-z]1')
1 sudo mkdir /mnt/pico
2 sudo mount /dev/$puerto /mnt/pico
3 sudo cp rp2-pico.uf2 /mnt/pico
4 sudo sync

```

Tener presente que el archivo **instaladorFirmware.sh** y el firmware descargado, deben estar en el mismo directorio.

Código C.25: Comando para ejecutar el instalador de Firmware.

```
o sudo ./instaladorFirmware.sh
```

Ahora que el firmware ya está cargado en la Raspberry Pi Pico W, se le conectan los sensores, módulos y leds RGB tal como se muestra.

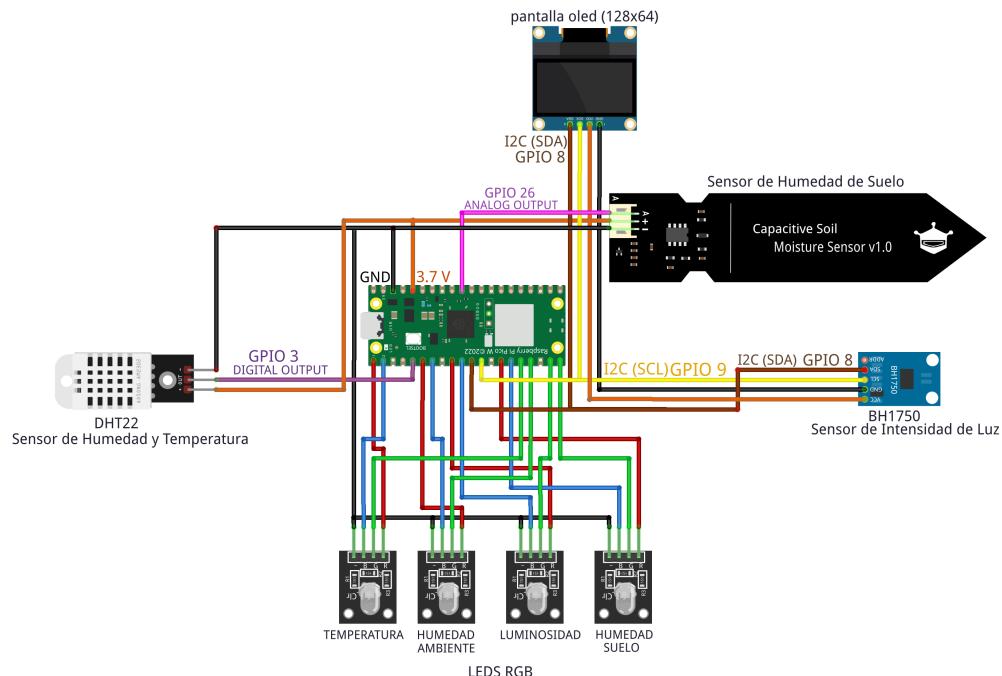


Figura C.9: conexiones en el hardware.

Para llevar a cabo estas conexiones, se utiliza un protoboard junto con cables diseñados específicamente para conexiones en este tipo de placas experimentales.

Todos los componentes comparten la misma fuente de alimentación de energía proporcionada por la Raspberry Pi Pico W.

Por ahora ninguno de los componentes necesita alimentación de energía adicional.

Procederemos a mostrar las conexiones específicas de cada componente para más detalle.

Los diagramas de conexiones son muy claros, no hay gran dificultad para ello.

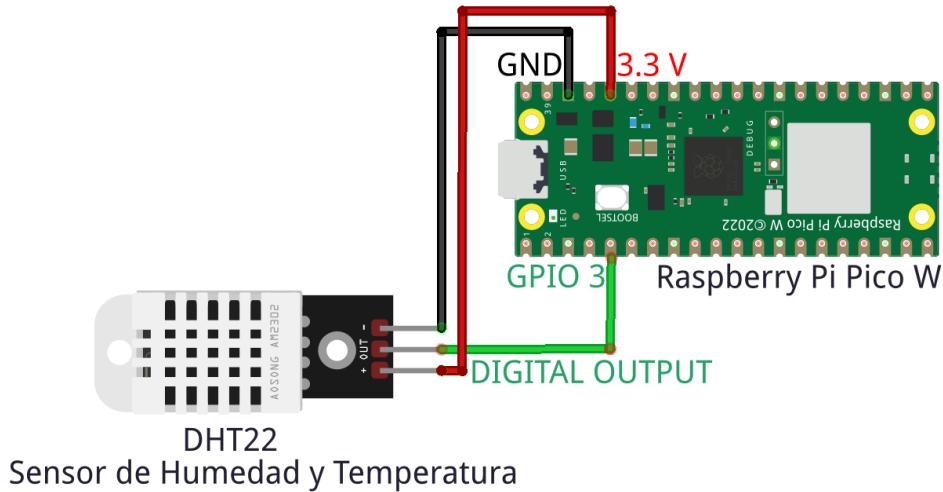


Figura C.10: DHT22: Sensor de humedad y temperatura del ambiente.

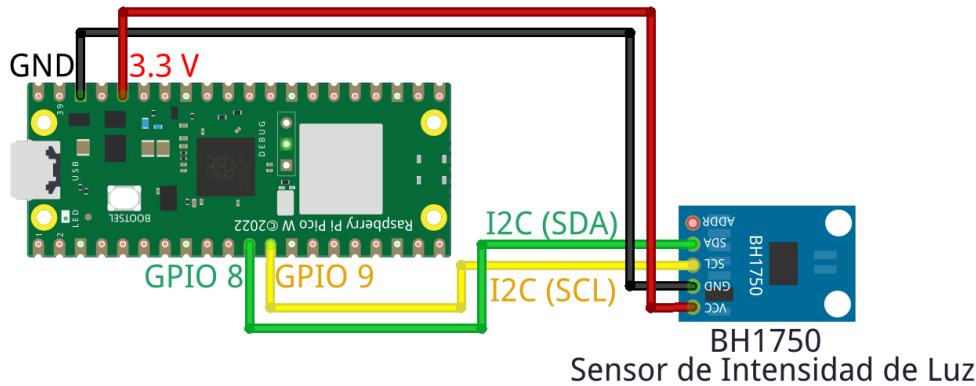


Figura C.11: BH1750: Sensor de intensidad de luz.

Respecto a todo el hardware usado, todos los componentes son de fácil adquisición y hay gran documentación en internet.

Si bien por ahora las conexiones se hacen usando un protoboard, en un futuro se puede diseñar una placa para conectar todos los componentes sin necesidad de cables.

El sensor de humedad de suelo [6] debe introducirse en el suelo solo hasta cierto límite.

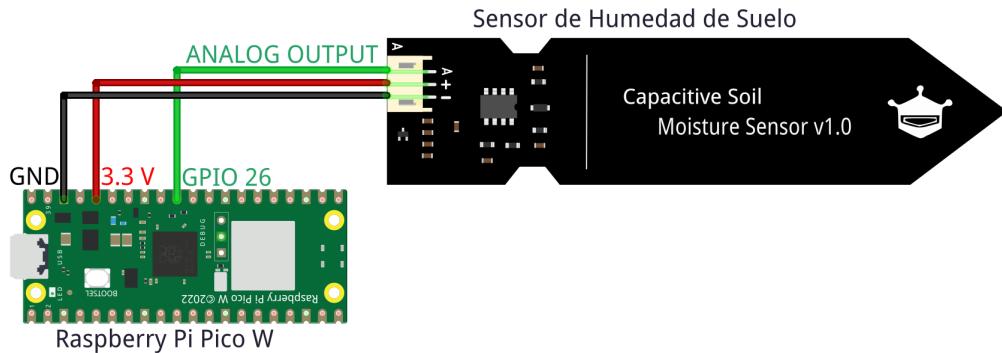


Figura C.12: Sensor de humedad del suelo.

La pantalla oled [C.13](#) permitirá observar los valores de los sensores en tiempo real en el mismo invernadero. Se ha escogido este modelo de pantalla por su bajo consumo de energía y el protocolo I2C nos permite hacer menos conexiones a comparación de este mismo modelo de pantalla pero con protocolo SPI.

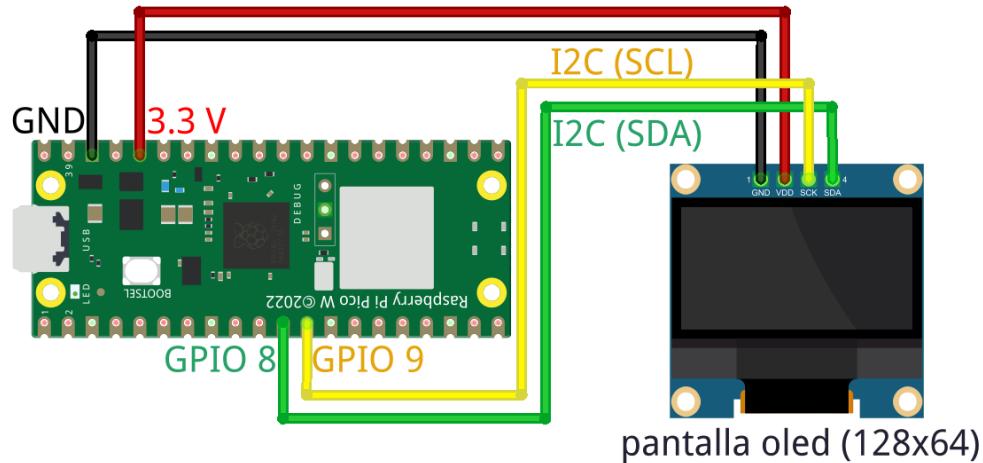


Figura C.13: Pantalla oled de 128x64 píxeles.

Los leds RGB [4] se usarán como alertas visuales, cada color indicará en qué posición respecto al umbral está un valor. Los umbrales representan los valores en el que tienen que estar cada una de las variables medidas.

Se están usando 4 módulos led RGB, uno por cada variable medida por los sensores.

Tabla C.17: Umbrales ideales para un invernadero de cannabis medicinal.

Característica	Mínimo	Máximo	Unidades
TEMPERATURA	30	35	°C
HUMEDAD	30	70	%
LUMINOSIDAD	30	150	lux
HUMEDAD DEL SUELO	20	80	%

Tabla C.18: Indicadores por color en los leds RGB.

Color	Descripción
Rojo	Por encima del umbral
Azul	Por debajo del umbral
Apagado	Valor dentro del umbral
Verde	Mecanismo activado

Cada módulo RGB tiene un cable para GND y tres respecto a los colores rojo, verde y azul.

Usando el IDE Thonny [23] y con Micropython vamos a poder programar el hardware.

Tenemos que hacer unas configuraciones simples antes de poder usar el IDE Thonny con Micropython en nuestra Raspberry Pi Pico W.

Hacemos clic en **Run/Configure interpreter**.

Seleccionamos que usaremos **Micropython** para Raspberry Pi Pico y seleccionamos el puerto que corresponde su conexión por usb.

Respecto a este proyecto solo tienes que seleccionar el archivo **main.py** y le das clic en el botón verde para ejecutar y cargar el código a la placa Raspberry Pi Pico W.

El archivo **boot.py** se ejecuta al prender la Raspberry Pi Pico W, y al importar el **main.py** hace que éste se ejecute. Entonces una vez hecha la configuración y programación solo tendremos que suministrar energía a la placa y empezará a funcionar para lo que se le ha programado.

nodeMqtt

Esta parte del proyecto se encarga de capturar datos mediante MQTT y los almacena en una base de datos usando Mysql.

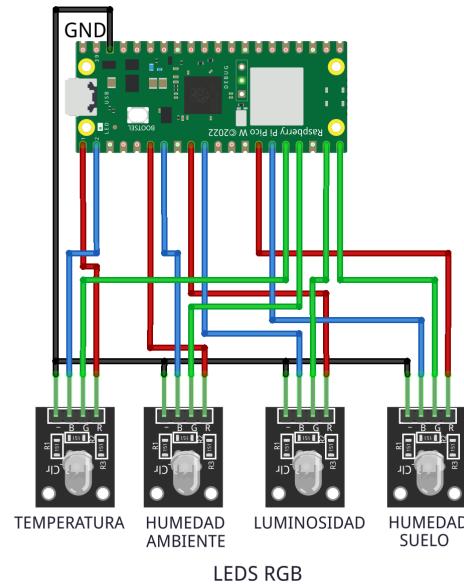


Figura C.14: Módulo led RGB.

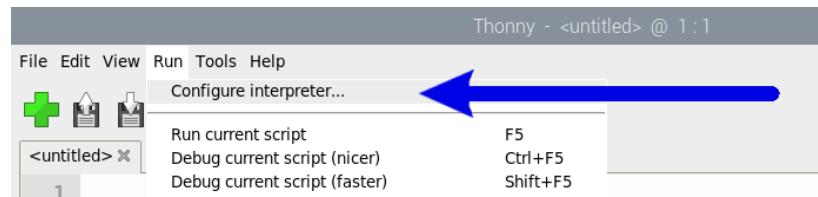


Figura C.15: Abrimos el configurador del interpreter.

La instalación y ejecución es muy simple.

Código C.26: Comando para instalar MQTT.js.

```
o npm install mqtt
```

Código C.27: Comando para instalar mysql para node.js.

```
o npm install mysql2
```

Código C.28: Ejecución.

```
o mp2 start index.js --name "nodeMqtt"
```

Respecto a MQTT **index.js** utiliza los siguientes topics.

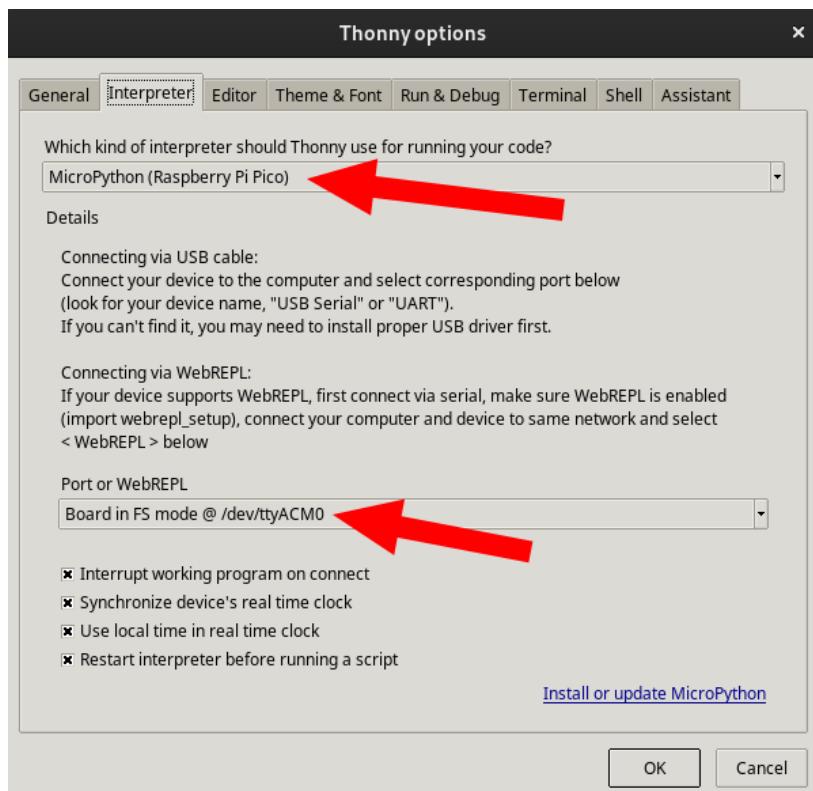


Figura C.16: Habilitamos Thonny para Micropython.

Tabla C.19: Topics MQTT usados en **nodeMqtt**.

Característica	Descripción
invernadero/sensores	Captura la fecha, hora y valores de sensores
invernadero/umbrales	Captura los umbrales

Almacena los datos en la base de datos **TFG_UBU C.15**.

Veamos el contenido de **index.js**.

```
var mqtt = require( "mqtt" );
const mysql = require( 'mysql2' );

// Opciones para la conexión MQTT
const mqttOptions = {
  keepAlive: 60,
  reconnectPeriod: 1000,
```

0
1
2
3
4
5
6

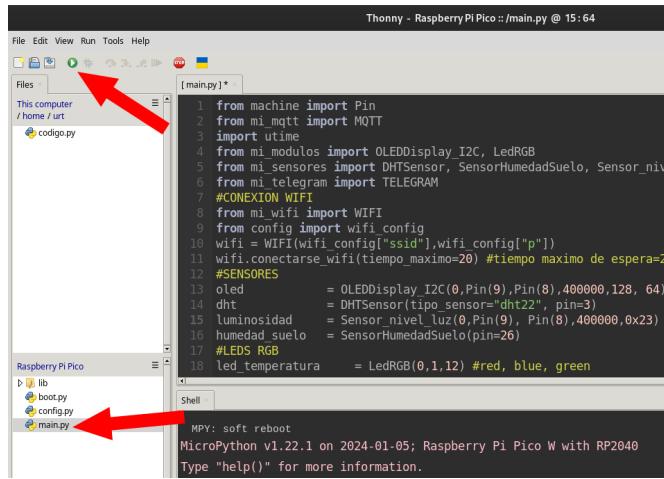


Figura C.17: Ejecución y carga a un solo clic.

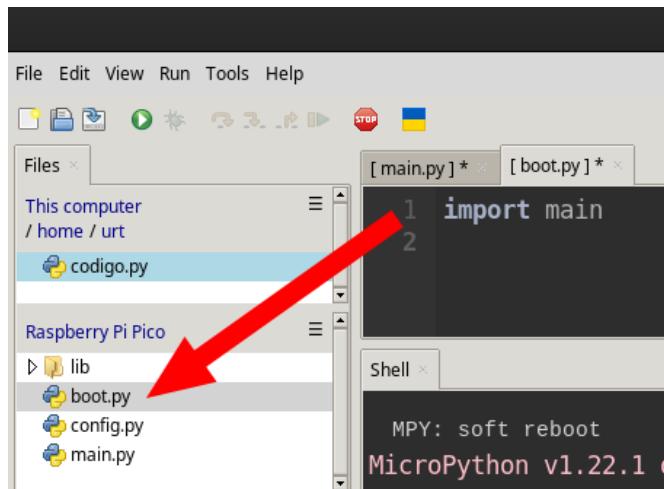


Figura C.18: boot.py se ejecuta en el arranque.

```

    connectTimeout: 4000
};

const client = mqtt.connect("mqtt://localhost:1883",
    mqttOptions);

const dbConfig = {
    host: 'localhost',
    user: 'joseluis',
}

```

7
8
9
10
11
12
13
14

	<input type="checkbox"/>											
					id	1	fecha	hora	temperatura	humedad	intensidad_luz	humedad_suelo
					4118		2024-02-03	22:45:13	20.00	64.50	50.83	-1.03
					4117		2024-02-03	22:45:10	20.00	64.40	50.00	-0.67
					4116		2024-02-03	22:45:06	20.00	64.50	50.00	-0.67
					4115		2024-02-03	22:45:03	19.90	64.80	50.00	-0.55
					4114		2024-02-03	22:45:00	19.90	65.00	50.00	-0.97
					4113		2024-02-03	22:44:57	19.90	65.10	50.00	-0.97

Figura C.19: Datos almacenados en tabla.

```

password: 'TuPassword',
database: 'TFG_UBU',
port: 3307
};
const pool = mysql.createPool(dbConfig);
function EventoConectar() {
    client.subscribe("invernadero/sensores");
    client.subscribe("invernadero/umbrales");
}
function EventoMensaje(topic, message) {
    if (topic === "invernadero/sensores") {
        const [fecha, hora, temperatura, humedad,
            intensidad_luz, humedad_suelo] = message.
            toString().split(',');
        const query = 'INSERT INTO sensores (fecha, hora,
            temperatura, humedad, intensidad_luz,
            humedad_suelo) VALUES (?, ?, ?, ?, ?, ?)';
        pool.query(query, [fecha, hora, temperatura,
            humedad, intensidad_luz, humedad_suelo], (error
        ) => {
            if (error) {
                console.error('Error al realizar la inserci n
                    en la base de datos:', error);
            } else {
                console.log('Inserci n en base de datos
                    exitosa');
            }
        });
    } else if (topic === "invernadero/umbrales") {
        const [temperatura_minima, temperatura_maxima,
            humedad_minima, humedad_maxima,

```

```

luminosidad_minima, luminosidad_maxima,
humedad_suelo_minima, humedad_suelo_maxima] =
message.toString().split(',') ;

37
const query = 'UPDATE umbrales SET
    temperatura_minima = ?, temperatura_maxima = ?,
    humedad_ambiente_minima = ?,
    humedad_ambiente_maxima = ?, luminosidad_minima
    = ?, luminosidad_maxima = ?,
    humedad_suelo_minima = ?, humedad_suelo_maxima
    = ?';

38
39
pool.query(query, [temperatura_minima,
    temperatura_maxima, humedad_minima,
    humedad_maxima, luminosidad_minima,
    luminosidad_maxima, humedad_suelo_minima,
    humedad_suelo_maxima], (error) => {
40
41
    if (error) {
        console.error('Error al actualizar umbrales en
            la base de datos:', error);
    } else {
42
        console.log('Actualización de umbrales en
            base de datos exitosa');
    }
43
44
});
45
46
47
48
49
client.on("connect", EventoConectar);
client.on("message", EventoMensaje);

50
51
52
pool.on('error', (err) => {
    console.error('Error inesperado en la conexión del
        pool:', err);
53
54
55
});

```

Código C.29: Contenido de index.js.

InverIoT

Aplicación de escritorio para Windows usando C#.

Los archivos relacionados a este proyecto están indicados en la tabla C.7.

Tabla C.20: Topics MQTT usados en InverIoT.

Característica	Descripción
invernadero/ordenes	Captura las ordenes
invernadero/sensores	Captura la fecha, hora y valores de sensores

MQTT se usa para mostrar los datos en tiempo real y Mysql para el histórico de datos.

Las ordenes que se envían son para activar mecanismos (representados por la activación de un led verde).

La ejecución del programa lo podemos hacer en Visual Studio Code [2] de la siguiente forma:

Código C.30: Ejecutar proyecto InverIoT.

```
o dotnet build
```

```
namespace InverIoT
{
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            // see https://aka.ms/
            // applicationconfiguration.
            ApplicationConfiguration.Initialize();
            Application.Run(new frmMain());
        }
    }
}
```

Código C.31: Contenido de Program.cs.

dashboard

Dashboard web para mostrar datos, no envía ordenes. Es accesible desde internet. El contenido del directorio **dashboard** lo encontramos en la tabla C.8.

Tabla C.21: Topics MQTT usados en **dashboard**.

Característica	Descripción
invernadero/sensores	Captura la fecha, hora y valores de sensores

Código C.32: Instalaciones necesarias.

```
0 npm install express
1 npm install socket.io
```

Respecto al archivo **servers.js** debemos configurar respecto a la base de datos creada en el servidor LAMP:

```
const dbConfig = {
    host: 'localhost',
    port: 3307, // Puerto especificado aquí
    user: 'joseluis',
    password: 'TuPassword',
    database: 'TFG_UBU'
};
```

Código C.33: Configuración de base de datos.

Código C.34: Ejecución.

```
0 mp2 start servers.js --name "dashboard"
```

```
1
2
3
4
5
6
7
8
9
```

```
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');
const mqtt = require('mqtt');
const path = require('path');
const mysql = require('mysql2');
const cors = require('cors');
const app = express();
const server = http.createServer(app);
const io = socketIo(server);
```

```

const mqttClient = mqtt.connect('mqtt://localhost:1883'
  , {
    keepAlive: 60,           // Enviar un mensaje de
    ping: cada 60 segundos para mantener la conexión
    reconnectPeriod: 1000,   // Intentar reconectar
    cada 1000 milisegundos (1 segundo) en caso de
    desconexión
    connectTimeout: 4000     // Tiempo de espera de
    4000 milisegundos (4 segundos) para la conexión
    inicial
  });
// Configuración de la base de datos
const dbConfig = {
  host: 'localhost',
  port: 3307, // Puerto especificado aquí
  user: 'joseluis',
  password: 'TuPassword',
  database: 'TFG_UBU'
};

const pool = mysql.createPool(dbConfig);
// Intentar obtener una conexión para probar si la
// conexión a la base de datos es exitosa
pool.getConnection((err, connection) => {
  if (err) {
    console.error('Error al conectar a la base de
      datos:', err);
    return;
  }
  console.log('Conectado exitosamente a la base de
    datos MySQL');
  connection.release(); // No olvides liberar la
    conexión
});

// Conexión MQTT y manejo de mensajes
mqttClient.on('connect', () => {
  console.log('Conectado al broker MQTT');
  mqttClient.subscribe('invernadero/sensores');
});

```

```

mqttClient.on('message', (topic, message) => {
    console.log(`Mensaje recibido en el topic ${topic}
${message}`);
    if (topic === 'invernadero/sensores') {
        io.emit('mqtt_data', message.toString());
    }
});

// Middleware
app.use(express.static(path.join(__dirname, './')));
app.use(cors());

// Rutas para servir las páginas
app.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, './', 'index.
html'));
});
app.get('/historico', (req, res) => {
    res.sendFile(path.join(__dirname, './', 'historico
.html'));
});

// Ruta para obtener datos históricos en un rango de
fechas
app.get('/datos-historicos', (req, res) => {
    const { fechaInicio, fechaFin } = req.query;
    let query = "SELECT fecha, hora, temperatura,
    humedad, intensidad_luz, humedad_suelo FROM
    sensores";
    let params = [];
    if (fechaInicio && fechaFin) {
        query += ' WHERE fecha BETWEEN ? AND ?';
        params.push(fechaInicio, fechaFin);
    }
    pool.query(query, params, (error, results, fields)
=> {
        if (error) {
            console.error('Error al realizar la
            consulta a la base de datos:', error);
            res.status(500).send('Error al obtener los
            datos');
        }
    });
});

```

```

        return ;
    }
    res.json( results );
})
);

app.get( '/umbrales' , (req , res) => {
    pool.query( 'SELECT * FROM umbrales ORDER BY
        fecha_actualizacion DESC LIMIT 1' , (error ,
    results , fields) => {
        if (error) {
            console.error('Error al realizar la
                consulta a la base de datos:' , error);
            res.status(500).send('Error al obtener los
                datos de los umbrales');
            return ;
        }
        res.json( results [0] );
    });
}
);
const PORT = 3000;
server.listen(PORT, () => console.log(`Servidor
    corriendo en el puerto ${PORT}`));

```

Código C.35: Contenido de **servers.js**.

Figura C.20: Dashboard en ejecución.

AnalisisDatos

Mediante Jupyter Notebook [17] se hace la limpieza de datos y un resumen estadístico.

En el directorio **AnalisisDatos** C.9 vamos a encontrar los archivos **analisis.ipynb** y **data.csv**.

```
sudo apt install jupyter-notebook
```

Código C.36: Instalación de Jupyter Notebook.

```
jupyter-notebook analisis.ipynb
```

Código C.37: Ejecución de **analisis.ipynb**.

Luego automáticamente se ejecuta el notebook y se visualiza en el navegador con una ruta muy similar a esta:

<http://localhost:8888/notebooks/analisis.ipynb>

Ahora ya puedes ver el código de análisis de datos con python en Jupyter Notebook.

The screenshot shows a Jupyter Notebook interface with the title "jupyter analysis (autosaved)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and various cell type icons. Below the toolbar, the main area has two code cells and their outputs.

In [144]: `data = pd.read_csv("data.csv")
data.head(3)`

Out[144]:

	id	fecha	hora	temperatura	humedad	intensidad_luz	humedad_suelo
0	1	2024-01-28	22:46:29	17.6	65.2	76.67	4.67
1	2	2024-01-28	22:46:33	17.6	65.2	76.67	5.28
2	3	2024-01-28	22:46:37	17.6	65.3	76.67	4.06

In [145]: `# últimas filas
data.tail(3)`

Out[145]:

	id	fecha	hora	temperatura	humedad	intensidad_luz	humedad_suelo
4115	4116	2024-02-03	22:45:06	20.0	64.5	50.00	-0.67
4116	4117	2024-02-03	22:45:10	20.0	64.4	50.00	-0.67
4117	4118	2024-02-03	22:45:13	20.0	64.5	50.83	-1.03

Figura C.21: Data de sensores en Jupyter notebook

C.4. Compilación, instalación y ejecución del proyecto

Luego de hacer las configuraciones ya mencionadas, se va a ejecutar respecto a estas partes específicas del proyecto:

Tabla C.22: Compilación y ejecución.

Parte del proyecto	Descripción
Servidor LAMP	Automático
Hardware	Automático al suministrar energía
nodeMqtt	Automático
InverIoT	Solo ejecutar su instalador en Windows
dashboard	Automático
AnalisisDatos	Necesita abrir mediante Jupyter Notebook
Bot de Telegram	Automático

Hardware

Simplemente conectar la Raspberry Pi Pico a una toma de energía y todo el hardware va a empezar a funcionar, ya que se ha configurado para que el archivo **main.py** se está ejecutando en el arranque.

InverIoT

Se puede usar simplemente su instalador llamado **Instalador - InverIoT - v2.2.exe**.

O se puede ejecutar el código fuente usando Visual Studio Code [2].

Código C.38: Ejecutar proyecto InverIoT.

```
o dotnet build
```

AnalisisDatos

Se ejecuta usando Jupyter Notebook [17].

```
jupyter-notebook analisis.ipynb
```

0

Código C.39: Ejecución de **analisis.ipynb**.

C.5. Calidad del código

Para evaluar la calidad del código, se ha empleado un servicio externo denominado SonarCloud, integrado con GitHub. Este servicio realiza un análisis exhaustivo del código, identificando posibles errores y sugiriendo mejoras para optimizar la calidad del código fuente.

Puedes acceder a las estadísticas de SonarCloud mediante el siguiente enlace: https://sonarcloud.io/project/overview?id=JLCaballeroMQ_Proyecto_TFG_UBU_23_24

Apéndice D

Documentación de usuario

D.1. Introducción

Este documento está diseñado para facilitar la comprensión y el uso del sistema, contribuyendo así a una experiencia de usuario satisfactoria y a la consecución de los objetivos de monitorización de cultivos de cannabis medicinal de manera eficaz y económica.

D.2. Instalación Física

Se tiene que instalar el firmware en la Raspberry Pi Pico W para que podamos programarlo mediante Micropython. Primero descargamos el firmware de Micropython desde su página oficial [14].

Para fines prácticos nombraremos al firmware descargado: **rp2-pico.uf2**.

Firmware

Releases

- [v1.22.1 \(2024-01-05\) .uf2 / \[Release notes\] \(latest\)](#) 
- [v1.22.0 \(2023-12-27\) .uf2 / \[Release notes\]](#)
- [v1.21.0 \(2023-10-05\) .uf2 / \[Release notes\]](#)
- [v1.20.0 \(2023-04-26\) .uf2 / \[Release notes\]](#)

Figura D.1: Descargar el firmware más reciente.

Conectar el puerto usb de la Raspberry Pi Pico W manteniendo presionado el botón **BOOTSEL**.

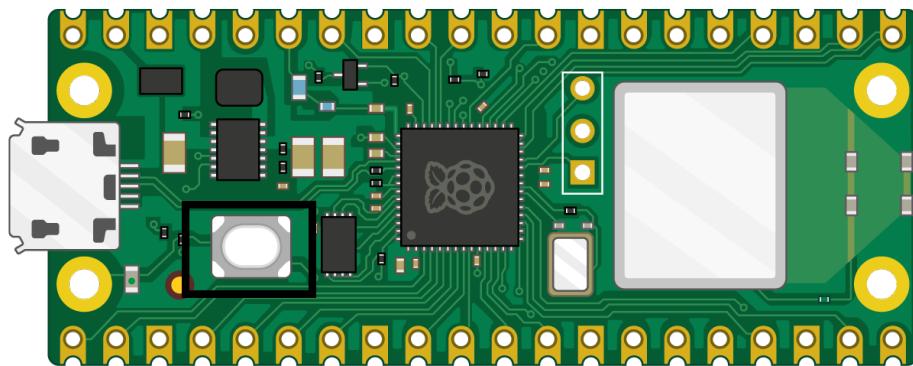


Figura D.2: Conectar usb manteniendo presionado.

Se ha creado un script en bash al que llamamos **instaladorFirmware.sh**. La ejecución de este permitirá instalar el firmware de Micropython en la Raspberry Pi Pico W.

Código D.1: Script en bash para instalar el firmware Micropython.

```
0 puerto=$(sudo dmesg | tail | grep -o 'sd[b-z]1')
1 sudo mkdir /mnt/pico
2 sudo mount /dev/$puerto /mnt/pico
3 sudo cp rp2-pico.uf2 /mnt/pico
4 sudo sync
```

Tener presente que el archivo **instaladorFirmware.sh** y el firmware descargado, deben estar en el mismo directorio.

Código D.2: Comando para dar permisos de ejecución.

```
0 sudo chmod +x instaladorFirmware.sh
```

Código D.3: Comando para ejecutar el instalador de Firmware.

```
0 sudo ./instaladorFirmware.sh
```

Usando el IDE Thonny [23] y con Micropython vamos a poder programar el hardware. Lo podemos descargar e instalar desde su pagina oficial [23].

Tenemos que hacer unas configuraciones simples antes de poder usar el IDE Thonny con Micropython en nuestra Raspberry Pi Pico W.

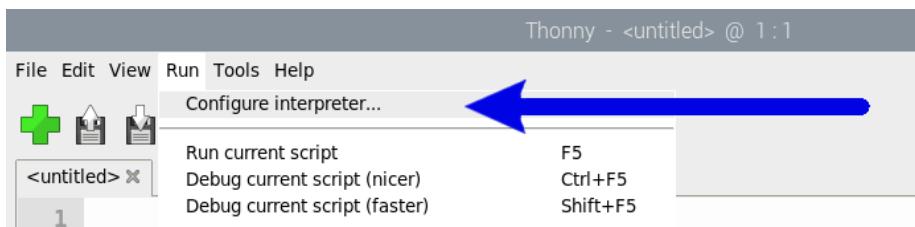


Figura D.3: Abrimos el configurador del interpreter.

Hacemos clic en **Run/Configure interpreter**.

Seleccionamos que usaremos **Micropython** para Raspberry Pi Pico y seleccionamos el puerto que corresponde su conexión por usb.

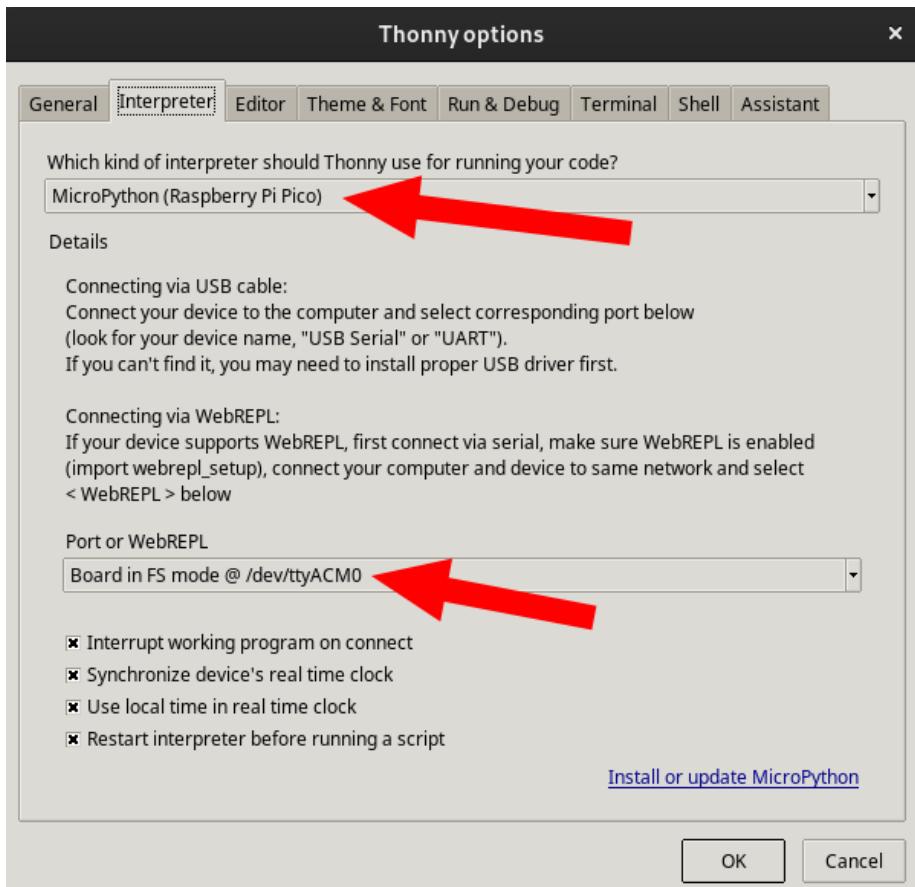


Figura D.4: Habilitamos Thonny para Micropython.

Usar los archivos del directorio **Hardware** para guardarlos en la Raspberry Pi Pico W usando el IDE Thonny.

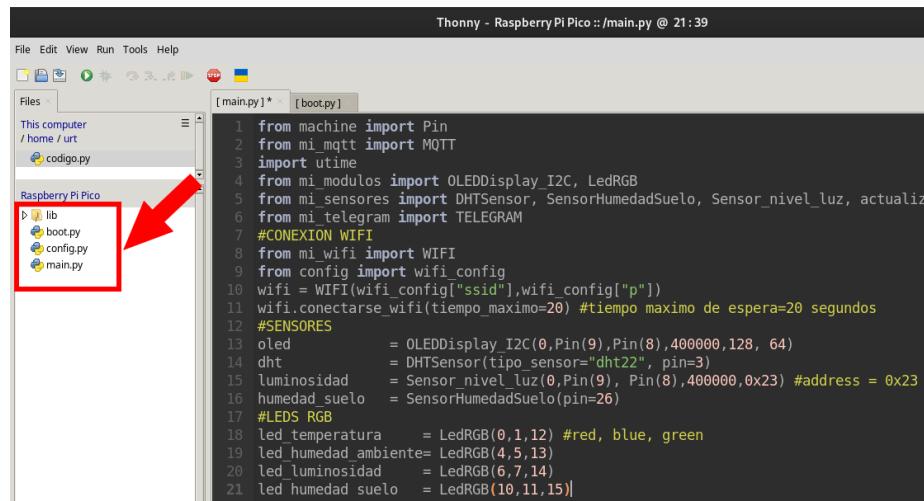


Figura D.5: Archivos necesarios para ejecutar en el hardware.

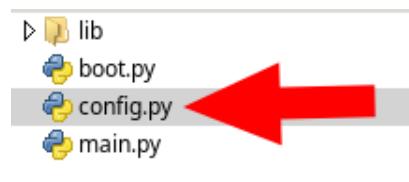


Figura D.6: Archivo de configuración en Micropython.

Modificamos el archivo **config.py** con nuestras credenciales.

Código D.4: Configurar SSID y clave wifi.

```

0 wifi_config = {
1     'ssid': 'SSID_WIFI',
2     'p': 'TU_PASSWORD_WIFI'
3 }

```

Código D.5: Configución de bot de Telegram.

```

0 utelegram_config = {
1     'token': 'BOT_TOKEN_TELEGRAM',
2     'chat_id': 'BOT_CHAT_ID'
3 }

```

Código D.6: Configuración de umbrales para los sensores.

```

0 umbrales_sensores = {
1     'TEMPERATURA_MINIMA': 30,
2     'HUMEDAD_MINIMA': 30,
3     'LUMINOSIDAD_MINIMA': 30,
4     'HUMEDAD_SUELO_MINIMA': 20,
5     'TEMPERATURA_MAXIMA': 35,
6     'HUMEDAD_MAXIMA': 70,
7     'LUMINOSIDAD_MAXIMA': 150,
8     'HUMEDAD_SUELO_MAXIMA': 80
9 }
```

Código D.7: Configuración MQTT.

```

0 mqtt_config = {
1     'server': '1.39.4.38', #IP DE TU SERVIDOR LAMP
2     'port': 1883, #PUERTO USADO POR MQTT
3     'topic_pub': 'invernadero/sensores',
4     'topic_sub': 'invernadero/ordenes',
5     'topic_umb': 'invernadero/umbrales',
6     'client_id_pub': 'TFG_UBU_mqtt_id_pub',
7     'client_id_sub': 'TFG_UBU_mqtt_id_sub',
8     'client_id_umb': 'TFG_UBU_mqtt_id_umb'
9 }
```

Código D.8: Configuración Wlan para usar ip fija.

```

0 wlan_config = {
1     'ip': '192.168.1.232',
2     'mask': '255.255.255.0',
3     'gateway': '192.168.1.1',
4     'dns': '8.8.8.8'
5 }
```

Una vez que has completado las credenciales ya puedes desconectar y volver a conectar la Raspberry Pi Pico W, al arrancar automáticamente se ejecutará la programación ya establecida.

Solo volverás a usar el IDE Thonny si quieras cambiar algo en el código o en el archivo de configuración.

Ahora procedemos a hacer las conexiones en el hardware.

Solo comprobar que todas las conexiones sean coherentes a la imagen.

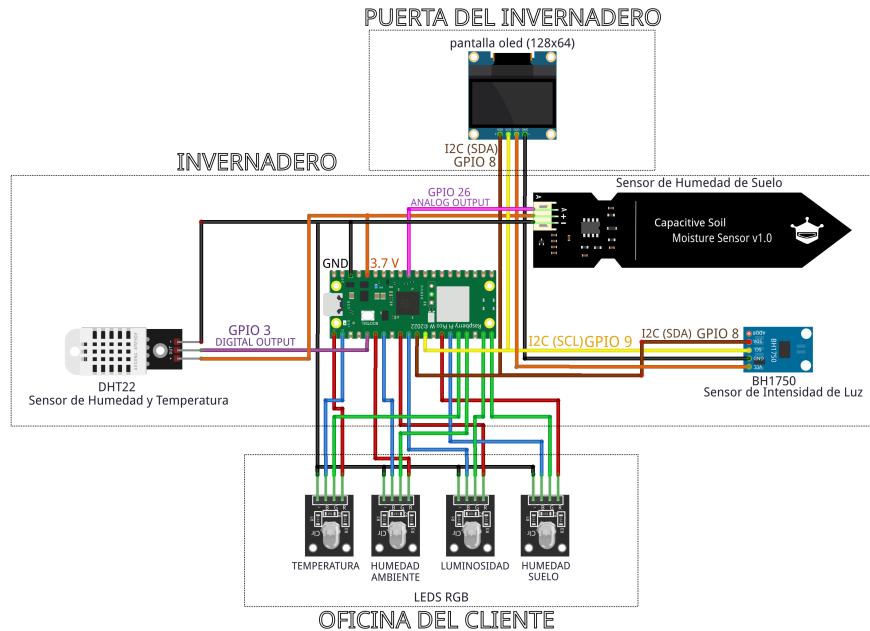


Figura D.7: Conexiones del hardware para el cliente.

D.3. Instalación del software

Servidor LAMP

Se ha creado un script en bash para hacer las instalaciones automáticamente al que se ha nombrado **InstaladorServidor.sh**.

Código D.9: Damos permisos de ejecución a **InstaladorServidor.sh**.

```
0 sudo chmod +x InstaladorServidor.sh
```

Código D.10: Contenido de **InstaladorServidor.sh**.

```
0 sudo apt install apache2
1 sudo apt install mysql-server
2 sudo apt install php
3 sudo apt install ssh
4 sudo apt install phpmyadmin
5 sudo apt install vsftpd
6 sudo apt install ufw
7 sudo apt install nodejs
8 sudo apt install pm2
```

```

9 sudo apt install npm
10 sudo apt install express
11 sudo apt install mosquitto

```

Código D.11: Ejecutamos **InstaladorServidor.sh**.

```
0 sudo ./InstaladorServidor.sh
```

Creación de base de datos

Código D.12: Comando para acceder a Mysql.

```
0 sudo mysql
```

Código D.13: Comandos Mysql para crear Database **TFG_UBU**.

```

0 CREATE DATABASE TFG_UBU;
1 CREATE USER 'joseluis'@'%' IDENTIFIED BY 'Mipassword';
2 GRANT ALL ON TFG_UBU.* TO 'joseluis'@'%';
3 use TFG_UBU;

```

Código D.14: Comandos Mysql para crear tabla **umbrales**.

```

0 CREATE TABLE 'umbrales' (
1   'temperatura_minima' float NOT NULL,
2   'temperatura_maxima' float NOT NULL,
3   'humedad_ambiente_minima' float NOT NULL,
4   'humedad_ambiente_maxima' float NOT NULL,
5   'luminosidad_minima' float NOT NULL,
6   'luminosidad_maxima' float NOT NULL,
7   'humedad_suelo_minima' float NOT NULL,
8   'humedad_suelo_maxima' float NOT NULL,
9   'fecha_actualizacion' timestamp
10  NULL DEFAULT CURRENT_TIMESTAMP
11  ON UPDATE CURRENT_TIMESTAMP
12 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
13 INSERT INTO 'umbrales' (
14   'temperatura_minima', 'temperatura_maxima',
15   'humedad_ambiente_minima', 'humedad_ambiente_maxima',
16   'luminosidad_minima', 'luminosidad_maxima',
17   'humedad_suelo_minima', 'humedad_suelo_maxima') VALUES
18   (30, 35, 30, 70, 30, 150, 20, 80);

```

Código D.15: Comandos Mysql para crear tabla **sensores**.

```

0 CREATE TABLE `sensores` (
1   `id` int NOT NULL AUTO_INCREMENT,
2   `fecha` date DEFAULT NULL,
3   `hora` time DEFAULT NULL,
4   `temperatura` decimal(4,2) DEFAULT NULL,
5   `humedad` decimal(4,2) DEFAULT NULL,
6   `intensidad_luz` decimal(4,2) DEFAULT NULL,
7   `humedad_suelo` decimal(4,2) DEFAULT NULL,
8   PRIMARY KEY (`id`)
9 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

dashboard

Código D.16: Instalaciones necesarias.

```

0 npm install express
1 npm install socket.io

```

Respecto al archivo **servers.js** debemos configurar respecto a la base de datos creada en el servidor LAMP:

```

const dbConfig = {
  host: 'localhost',
  port: 3307, // Puerto especificado aquí
  user: 'joseluis',
  password: 'TuPassword',
  database: 'TFG_UBU'
};

```

Código D.17: Configuración de base de datos.

Solo se ejecuta una vez y siempre va a estar ejecutándose, incluso si se reinicia el servidor.

Código D.18: Ejecución.

```

0 mp2 start servers.js --name "dashboard"

```

InverIoT

Solo ejecutar el instalador **Instalador - InverIoT - v2.2.exe**.

D.4. Manual del usuario

Una vez todo está instalado y configurado ya puedes interactuar con las diferentes partes del proyecto.

Hardware

Solo conectar la Raspberry Pi Pico a una fuente de energía y está operativo.

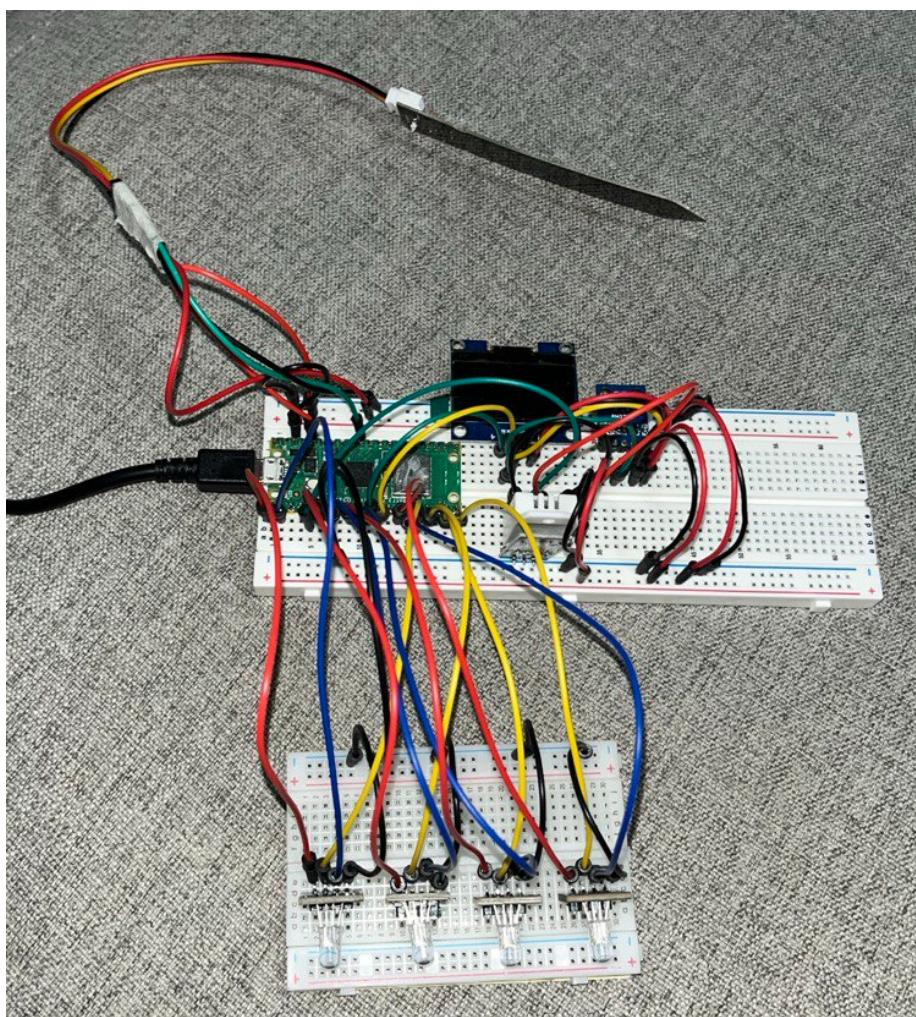


Figura D.8: Foto real de las conexiones.

Clavar el sensor de humedad de suelo en el punto que se desea monitorear.



Figura D.9: Sensor de humedad de suelo operativo.

Los valores en tiempo real se pueden ver en la pantalla oled con sus respectivas unidades.

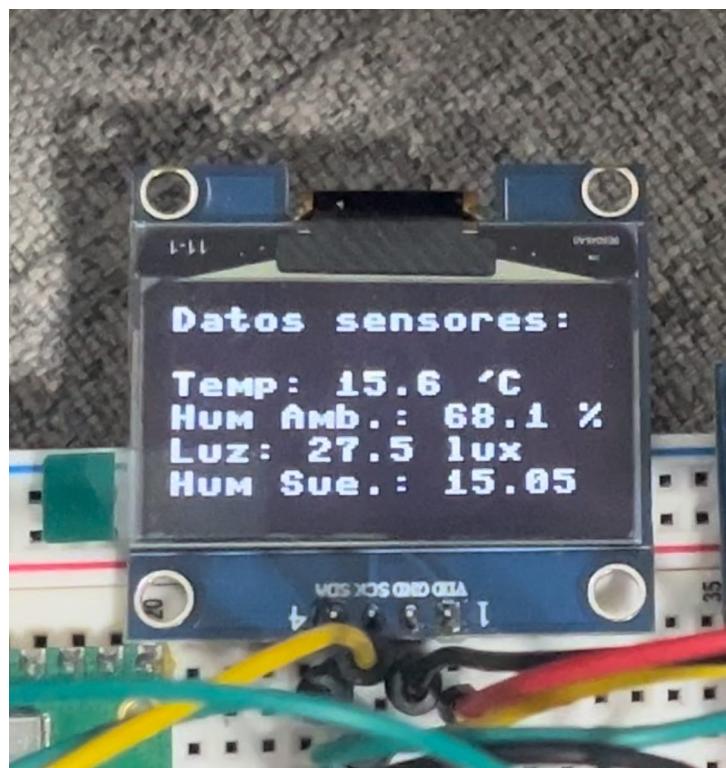


Figura D.10: Pantalla oled mostrando los datos.

InverIoT

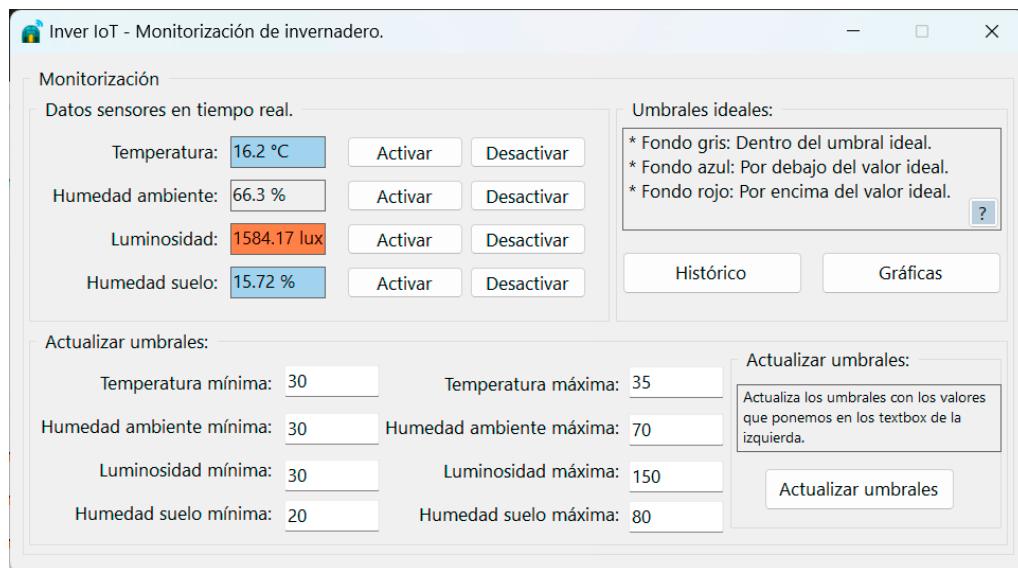


Figura D.11: Interfaz de la aplicación de escritorio.

La interfaz permite modificar los umbrales, tiene una comunicación bidireccional con la base de datos.

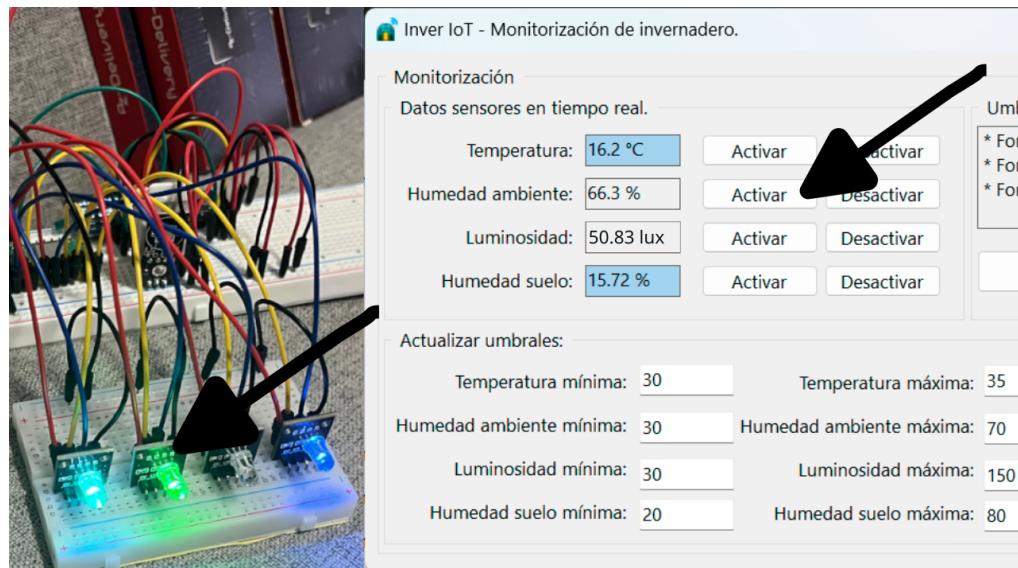


Figura D.12: Activación de mecanismo con un clic.

La activación del mecanismo está representado por el led verde prendido, pero si se desea activar otro mecanismo, solo desconectar del módulo led RGB el cable asociado al color verde, y conectarlo al mecanismo que se desea. Hacer esto solo si el nuevo mecanismo a activar es compatible en cuanto al voltaje necesario.

	ID	Fecha	Hora	Temperatura	Humedad Ambiente	Luminosidad	Humedad Suelo
▶	2592	31/01/2024	12:26:28	15,60	68,70	20,83	3,64
	2593	31/01/2024	12:26:32	15,50	68,50	20,83	3,82
	2594	31/01/2024	12:26:36	15,50	68,40	20,83	5,22
	2595	31/01/2024	12:26:40	15,50	68,40	20,83	4,79
	2596	31/01/2024	12:26:44	15,50	68,30	21,67	4,85
	2597	31/01/2024	12:26:48	15,50	68,30	21,67	4,13
	2598	31/01/2024	12:26:52	15,50	68,30	20,83	5,22
	2599	31/01/2024	12:26:57	15,50	68,30	21,67	5,04
	2600	31/01/2024	12:27:01	15,50	68,30	21,67	4,49
	2601	31/01/2024	12:27:05	15,50	68,30	21,67	5,16
	2602	31/01/2024	12:27:09	15,50	68,30	21,67	5,46
	2603	31/01/2024	12:27:14	15,50	68,30	21,67	4,43
	2604	31/01/2024	12:27:18	15,50	68,30	21,67	5,16
	2605	31/01/2024	12:27:22	15,50	68,30	21,67	4,37
	2606	31/01/2024	12:27:26	15,50	68,30	21,67	4,13
	2607	31/01/2024	12:27:31	15,50	68,30	23,33	4,06
	2608	31/01/2024	12:27:35	15,50	68,30	27,50	4,73
	2609	31/01/2024	12:27:40	15,50	68,20	33,33	4,43
	2610	31/01/2024	12:27:45	15,50	68,20	33,33	4,55
	2611	31/01/2024	12:27:49	15,50	68,20	32,50	5,22
	2612	31/01/2024	12:27:54	15,50	68,20	33,33	4,55
	2613	31/01/2024	12:27:58	15,50	68,20	33,33	4,43

Figura D.13: Acceso al histórico de datos.



Figura D.14: Capacidad de mostrar gráficas en un rango de fechas.

dashboard

Solo permite visualizar los datos, no envía comandos.



Figura D.15: El color rojo indica se ha sobrepasado el umbral.



Figura D.16: El color azul indica está por debajo del umbral.

Bot de telegram

Permite enviar comandos y recibir alertas.

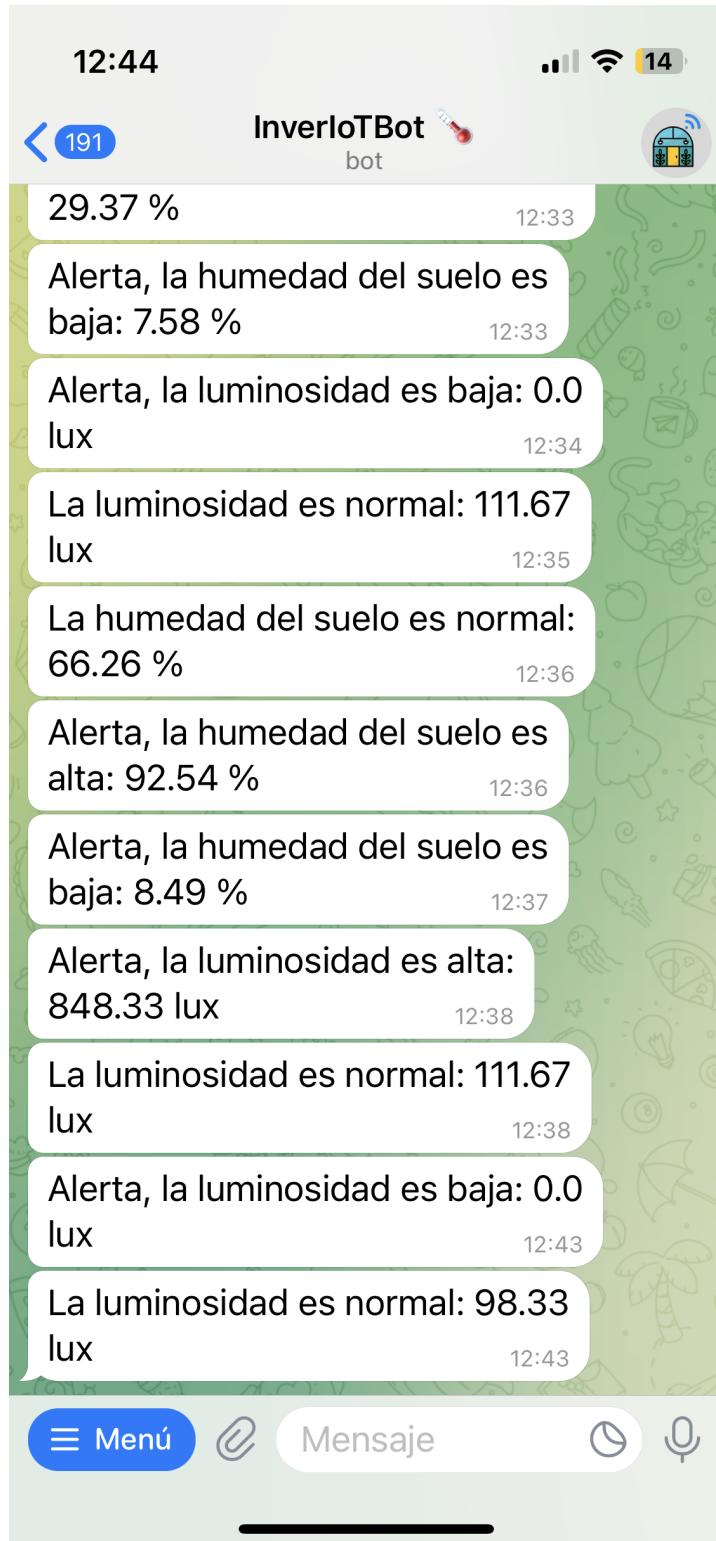


Figura D.17: Alertas recibidas por el bot de Telegram.

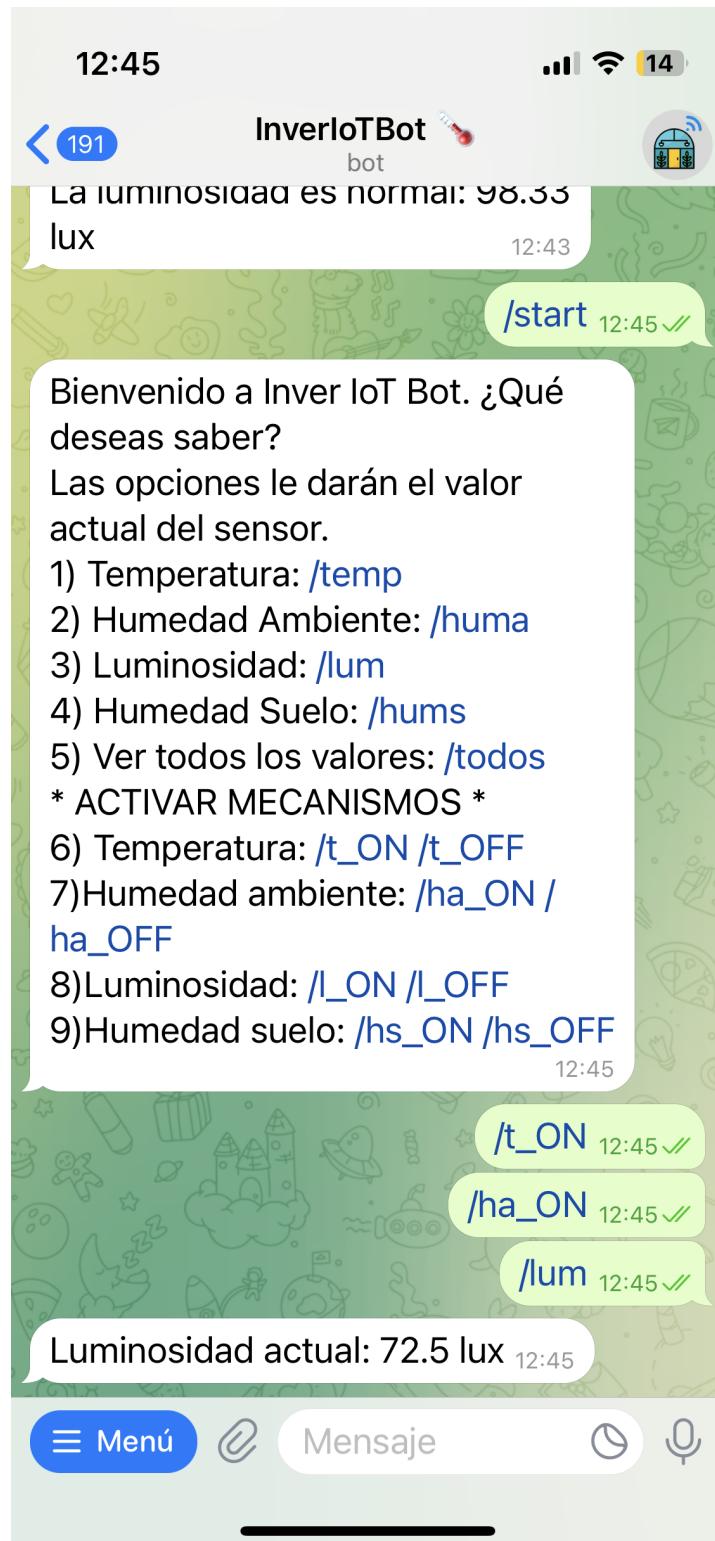


Figura D.18: Capacidad de enviar comandos desde Telegram.

Bibliografía

- [1] Apache. Http server project. <https://httpd.apache.org/>.
- [2] Visual Studio Code. Code editing redefined. <https://code.visualstudio.com/>.
- [3] Creative Commons. Norma creative commons. https://creativecommons.org/licenses/by-sa/3.0/deed.es_ES.
- [4] datasheetspdf. *KY 016 RGB 5mm LED module.* <https://datasheetspdf.com/mobile/1402027/Joy-IT/KY-016/1>.
- [5] Equipo de Desarrollo de MicroPython. Micropython documentation, 2024. <https://github.com/micropython/micropython/wiki>.
- [6] DFROBOT. Capacitive soil moisture sensor, 1 2024. https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193.
- [7] Fritzing. Electronics made easy. <https://fritzing.org/>.
- [8] GNU. *Licencia GPL3.* <https://www.gnu.org/licenses/gpl-3.0.html>.
- [9] Handsontec. *BH1750 Ambient Light Sensor Module.* <https://www.handsontec.com/dataspecs/sensor/BH1750%20Light%20Sensor.pdf>.
- [10] Hostingplus. Modelo de prototipos: ¿qué es y cuáles son sus etapas? <https://www.hostingplus.com.es/blog/modelo-de-prototipos-que-es-y-cuales-son-sus-etapas/>.

- [11] HPE. Hp proliant servers. https://www.hpe.com/emea_europe/en/hpe-proliant-servers.html.
- [12] IEEEEXPLORE. *MQTT Protocol: Fundamentals, Tools and Future Directions*. <https://ieeexplore.ieee.org/document/8931137>.
- [13] LaTeX. The latex project. <https://www.latex-project.org/>.
- [14] Micropython. Micropython firmware. https://micropython.org/download/RPI_PICO_W/.
- [15] Mosquitto. An open source mqtt broker. <https://mosquitto.org/>.
- [16] MySQL. The world's most popular open source database. <https://www.mysql.com/>.
- [17] Jupyter Notebook. Free software, open standards, and web services for interactive computing across all programming languages. <https://jupyter.org/>.
- [18] Seguridad Social. *Información General sobre cotización*, 2024. <https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/10721/10957/583>.
- [19] Sparkfun. *DHT22 Digital-output relative humidity & temperature sensor/module*. <https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf>.
- [20] Telegram. Telegram api. <https://core.telegram.org/api>.
- [21] Telegram. Telegram apps. <https://telegram.org/apps>.
- [22] Telegram. Telegram bots. <https://core.telegram.org/bots>.
- [23] Thonny. Python ide for beginners. <https://thonny.org/>.
- [24] Ubuntu. Ubuntu. <https://ubuntu.com/>.
- [25] umqtt. Micropython. <https://github.com/micropython/micropython-lib/tree/master/micropython/umqtt.simple/umqtt>.
- [26] waveshare. *Pantalla oled sh1106 128x64 1.3 pulgadas datasheet*. <https://www.waveshare.com/w/upload/e/e3/1.3inch-SH1106-OLED.pdf>.