

UNIVERSIDAD DE MÁLAGA

**ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN**



TRABAJO FIN DE MASTER

*Objeto inteligente IoT: Monitorización y riego
automático para plantas*

**MASTER EN SISTEMAS ELECTRÓNICOS
PARA ENTORNOS INTELIGENTES**

MÁLAGA, 2019

SERGIO GASQUEZ ARCOS

UNIVERSIDAD DE MÁLAGA

**ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN**

**Objeto inteligente IoT: Monitorización y
riego automático para plantas**

REALIZADO POR:

Sergio Gasquez Arcos

DIRIGIDO POR:

Eva González Parada

Ignacio Herrero Reder

DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA

TITULACIÓN: Máster en Sistemas Electrónicos para Entornos Inteligentes

Resumen

Palabras clave: Sistema de monitorización, sistema de riego, sensores, firmware, MQTT, interfaz gráfica, CC3200, internet de las cosas.

En este trabajo fin de máster se ha desarrollado un sistema de monitorización y riego para plantas que permite ser usado tanto a escala domestica como a gran escala. El sistema se basa en el microcontrolador CC3200, para el cual se ha desarrollado la interfaz de 3 sensores (humedad en suelo, temperatura y humedad de ambiente y nivel de agua) y se ha diseñado la actuación del regadío mediante una bomba alimentada de forma externa y controlada por un relé. También se ha desarrollado una aplicación para PC que incorpora una interfaz gráfica en la que se pueden visualizar los datos de los sensores, tanto el más actual como un histórico, y desde la que se puede regar o programar el riego de la planta mediante unos umbrales de humedad en suelo y temperatura ambiente, los cuales harán que se riegue la planta si se sobrepasan. Los datos también son almacenados en un fichero del PC en formato CSV, por si fuese necesario su posterior tratamiento. La comunicación entre el microcontrolador y el PC se realiza de forma inalámbrica, mediante Wi-Fi, y usando el protocolo MQTT, el cual hace muy sencillo el escalado del proyecto, debido a su funcionamiento basado en la publicación/subscripción. Además, se ha desarrollado un sitio web, en el que también se puede acceder a todos los datos e interactuar con el sistema aun estando fuera de la red Wi-Fi.

Málaga, Septiembre de 2019

Abstract

Keywords: Monitoring system, watering system, sensors, firmware, MQTT, graphic user interface, CC3200, internet of things.

In this final master project a monitoring and watering system, that can be used for domestic use as for large scale, has been developed. The system is based on the CC3200, 3 sensors interfaces have been developed (soil moisture, temperature and humidity ratio and water level) as well as a way to water the plant via a water pump powered by an external battery and controlled by a relay. A PC application has also been developed consisting of a graphic user interface in where the user can visualize the sensors data, both the latest measure as well as a graphic containing the history of the variable. The user interface also allows to water the plan in two ways, normal watering or fixing a temperature and soil moisture threshold, if those thresholds are surpassed the plant will water by itself. Data is also stored in a file at the computer in CSV format, just in case, any further processing is needed. The communication between PC and microcontroller is wireless and done via Wi-Fi, using MQTT, which makes scaling the project very easy thanks to the publish-subscribe method. Besides, a websyte has been developed in order to be able to access the data and interact with the system even when not in the same Wi-Fi network.

Málaga, September 2019

*Dedicado a aquellas personas que hacen del mundo un lugar
mejor.*

Índice general

1. Introducción	15
1.1. Introducción	15
1.2. Motivación	17
1.3. Objetivos	18
2. Directivas del Proyecto	19
2.1. Oportunidades de negocio	19
2.2. Descripción del problema	20
2.3. Descripción del producto	21
3. Descripción de participantes y usuarios	23
3.1. Resumen de los participantes	24
3.2. Resumen y entorno de los usuarios	25
3.2.1. Entorno de los usuarios	25
3.3. Perfiles de los participantes	26

3.3.1. Jefe de la empresa	26
3.3.2. Jefe del proyecto	26
3.3.3. Usuario final	27
3.4. Alternativas y competencia	27
3.4.1. Qampo	27
3.4.2. Agrae	27
4. Requisitos	29
4.1. Diagrama general	29
4.2. Precedencia y prioridad	30
4.3. Requisitos Funcionales	31
4.3.1. Medición	31
4.3.2. Mostrar	31
4.3.3. Regar	31
4.3.4. Procesado de datos	32
4.3.5. Arquitectura de la red	32
4.4. Requisitos No Funcionales	32
4.4.1. Fecha de entrega	32
5. Casos de uso	33
5.1. Actores del sistema	33

5.2. Diagrama general	34
5.3. Descripción textual de los casos de uso	34
5.3.1. Medir los valores de la planta	34
5.3.2. Mostrar los valores de la planta	35
5.3.3. Regar	36
5.3.4. Almacenar valores	37
5.3.5. Guardar Histórico	38
5.3.6. Fijar parámetros de riego	39
6. Arquitectura	41
6.1. Arquitectura Lógica	41
6.2. Arquitectura Física	43
7. Descripción del sistema	45
7.1. Alimentación del sistema	47
7.2. CC3200 Launchpad	48
7.3. Sensor de humedad en suelo SEN0193	49
7.4. Sensor de temperatura y humedad relativa en ambiente SHT31	51
7.5. Sensor de ultrasonidos HC SR04	52
7.6. Sistema de riego	55
8. Desarrollo Firmware/Software	57

8.1.	Aplicación de microcontrolador	57
8.1.1.	Tarea de sensado	60
8.1.2.	Tarea de riego	61
8.1.3.	Arquitectura de red	61
8.1.4.	Interfaces	62
8.2.	Interfaz Gráfica	66
8.3.	Arquitectura de red MQTT	70
8.4.	Losant, Plataforma IoT	75
8.5.	Prototipos desarrollados	79
8.5.1.	Prototipo V0	79
8.5.2.	Prototipo V1	81
9.	Plan de aceptación	85
9.1.	Pruebas	86
9.1.1.	Mostrar valores	87
9.1.2.	Regar	90
9.1.3.	Comprobar umbrales	91
9.1.4.	Comprobar histórico	93
10.	Conclusiones	95
10.1.	Conclusiones	95

10.2.Trabajo futuro	96
11.Apéndice	99
11.1.Presupuesto	99

Índice de figuras

1.1. Diagrama de Gantt del proyecto	17
4.1. Diagrama general de los requisitos del proyecto	29
5.1. Diagrama general de casos de uso	34
6.1. Arquitectura lógica general del proyecto	42
6.2. Arquitectura física general del proyecto	43
7.1. Diagrama general del proyecto	46
7.2. CC3200 Launchpad	48
7.3. Sensor de humedad en suelo SEN0193	49
7.4. Divisor de tensión diseñado para el sensor de humedad en suelo SEN0193	50
7.5. Disposición de los pines del SHT31	52
7.6. Esquemático del SHT31	52
7.7. Sensor de ultrasonidos HC SR04	53

7.8.	Funcionamiento del sensor de ultrasonidos HCSR04 [17]	54
7.9.	Divisor de tensión para adaptar el pulso de Echo	55
7.10.	Circuito usado para el sistema de riego	56
8.1.	Diagrama del código principal	58
8.2.	Envío y lectura del comando de lectura en alta repetibilidad	63
8.3.	Pestaña "General"de la interfaz gráfica	67
8.4.	Pestaña " <i>Watering</i> "de la interfaz gráfica	68
8.5.	Métodos de la clase GUIPanel	69
8.6.	Ejemplo de archivo almacenado	70
8.7.	Ejemplo de un posible escenario con 2 estaciones y un teléfono móvil conectados	71
8.8.	Tramas recibidas en la aplicación MyMQTT	72
8.9.	Ejemplo de comunicación MQTT del sistema	73
8.10.	Arquitectura del sistema con varias estaciones	75
8.11.	Atributos del dispositivo de Losant	76
8.12.	<i>Dashboard</i> desarrollado en Losant	77
8.13.	Página de bienvenida de la web desarrollada	78
8.14.	Prototipo V0	80
8.15.	Prototipo V1	82
9.1.	Representación en la interfaz de los datos enviados	88

9.2.	Representación en las gráficas de los datos enviados . . .	89
9.3.	Comprobación del riego	91
9.4.	Comprobación de riego fijando umbrales	92
9.5.	Comprobación de los datos guardados en el histórico . . .	94

Índice de tablas

2.1. Descripción del problema	20
2.2. Descripción del producto	21
3.1. Resumen de los participantes	24
3.2. Resumen y entorno de los usuarios	25
3.3. Perfil del jefe de la empresa	26
3.4. Perfil del jefe del proyecto	26
3.5. Perfil del usuario final	27
4.1. Precedencia y prioridad	30
5.1. Actores del sistema	33
5.2. Diagrama general de casos de uso	35
7.1. Conexiones del sensor de humedad en suelo SEN0193 . .	50
7.2. Conexiones del SHT31	52
7.3. Conexiones del sensor de distancia HC SR04	54

9.1.	Prueba de lectura de sensores	86
9.2.	Prueba de mostrar valores en la interfaz	87
9.3.	Prueba de riego	90
9.4.	Prueba de comprobar umbrales	91
9.5.	Prueba de comprobar el histórico	93
11.1.	Presupuesto del proyecto	99

Capítulo 1

Introducción

1.1. Introducción

Hoy en día, la agricultura es uno de los sectores que más se está beneficiando de las nuevas tecnologías. En los últimos años, un gran número de compañías apuestan por el IoT (*Internet of Things*) para facilitar y mejorar el rendimiento de sus cultivos, pero, la gran mayoría de las veces se trata de proyectos enfocados a grandes terrenos de cultivo en lugar de agricultura doméstica para un número más reducido de plantas. En este proyecto se analizará, definirá y desarrollará un sistema monitorización y riego enfocado para el uso doméstico pero con facilidades para su escalado.

El sistema será capaz de:

- Monitorizar: Recoger diferentes medidas (humedad del suelo, temperatura, humedad relativa ...), de gran importancia para el bienestar de la planta, mostrarlas por una interfaz gráfica y almacenarlas en el ordenador.
- Actuar: El sistema podrá regar bajo la demanda del usuario o mediante

unos umbrales que se podrán variar a través de la interfaz.

Para conseguir este propósito se ha usado la placa CC3200 Lauchpad de Texas Instrument [1] como microprocesador principal, se han usado 3 sensores (humedad en suelo, ambiente y nivel de agua) y un actuador que controla el circuito de regadío. También se ha desarrollado una interfaz gráfica en la que se puede observar el valor de las diferentes variables, su progresión a lo largo del tiempo, así como activar el riego de diferentes formas. Tanto la monitorización como la actuación se pueden realizar de forma remota, incluso a kilómetros de distancia, ya que la comunicación se realiza a través de Wi-Fi usando el protocolo MQTT (*MQ Telemetry Transport*) [2], esto le dará un gran atractivo y una gran utilidad al proyecto.

Además, se ha desarrollado una web, usando la plataforma Losant [3], la cual esta conectada con el sistema y ofrece una interfaz web en la que se pueden realizar las mismas acciones que en la interfaz local, pero tiene el extra de que no necesariamente se debe estar conectado a la misma red Wi-Fi.

El proyecto se ha desarrollado usando, en medida de lo posible, *software* gratuito por lo que se ha usado el entorno de programación de Texas Instrument, Code Composer Studio [4], para el desarrollo del código en C del microcontrolador y para el desarrollo de la interfaz se ha usado QtCreator [5]. Además, se ha usado control de versiones con GitHub [6] en el que se pueden encontrar todos los cambios realizados en el proyecto así como esquemas, códigos y diseños.

Para mejorar la organización de objetivos y tareas se desarrolló un diagrama de Gantt [7] en el que se detalla el tiempo asignado a cada tarea:

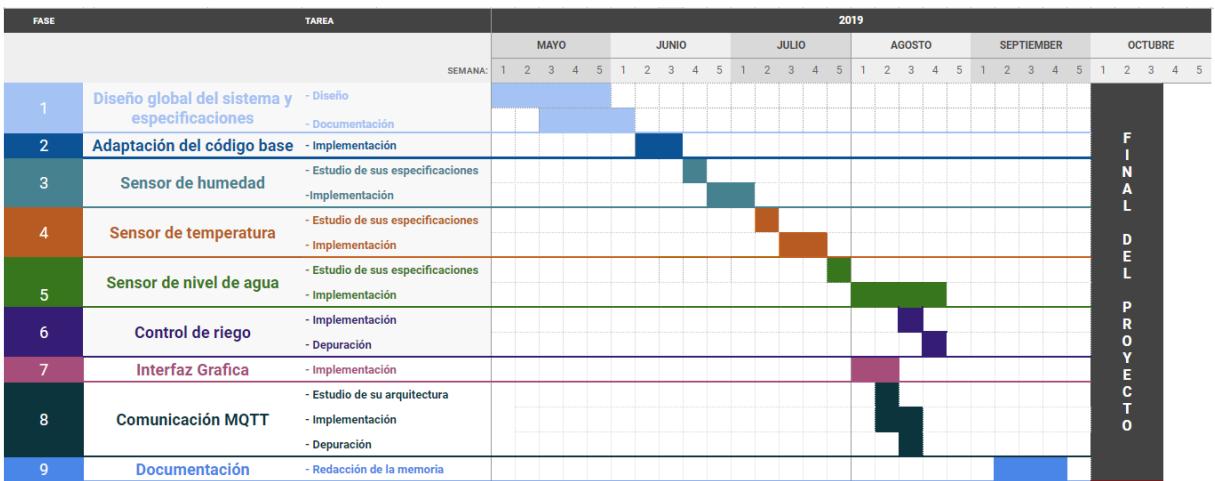


Figura 1.1: Diagrama de Gantt del proyecto

1.2. Motivación

El proyecto ofrece una gran variedad de desafíos y oportunidades para el desarrollo de un ingeniero de telecomunicaciones que se encuentra realizando el Máster en Sistemas Electrónicos para Entornos Inteligentes, ya que abarca diferentes ámbitos y los une. Desde el desarrollo de una interfaz gráfica en C++ hasta la programación en C de un microcontrolador pasando por la selección y implementación de sensores, comunicaciones inalámbricas a través de MQTT y muchos más retos.

En los últimos años se han comenzado a valorar cada vez más y más el valor de los datos ya que con ellos se pueden optimizar procesos, evitar situaciones no deseadas o simplemente visualizar estados. Por lo que el sistema de monitorización y riego que se desarrollará será una manera fácil de adquirir datos para su posterior envío para su visualización y procesamiento.

1.3. Objetivos

El principal objetivo será desarrollar un sistema de monitorización y riego económico, funcional y con capacidad de enviar los datos (a través de MQTT) a una interfaz y almacenarlos.

El sistema será fácilmente escalable, ya que el uso del protocolo MQTT, permite añadir nuevos terminales relativamente fácil y el usuario podrá fijar parámetros de activación de riego o incluso regar en el instante que desee. Conseguir una arquitectura que facilite esta tarea es primordial para el proyecto.

Como se comentó anteriormente, el sistema usará 3 sensores, los cuales deben ser integrados en el sistema, es decir, se deberá crear una interfaz para que el microcontrolador pueda comunicarse con el de forma correcta. Cada sensor funciona con un protocolo diferente, por lo que la complejidad de esta tarea puede llegar a ser muy alta.

El sistema de riego deberá diseñarse con una alimentación externa ya que el sistema se alimentará principalmente a través de una conexión USB (*Universal Serial Bus*) a un ordenador, la cual no es capaz de alimentar una bomba. Este es otro de los grandes problemas con los que se deberá lidiar en el sistema.

Un gran objetivo que se ha conseguido, a pesar de no estar planeado en un principio, es la conexión del sistema con una plataforma IoT, la cual permite visualizar y almacenar los datos, además de interactuar con el sistema. De forma que se puede regar y monitorizar desde cualquier lugar del mundo.

Capítulo 2

Directivas del Proyecto

2.1. Oportunidades de negocio

El proyecto presenta gran variedad de oportunidades de negocio debido a su flexibilidad y bajo coste, por lo cual se adapta a las necesidades de usuarios domésticos, que deseen monitorizar y automatizar el riego de una o varias plantas en su domicilio pero también permite su uso a gran escala en campos de agricultura o invernaderos, ya que gracias a su arquitectura la escalabilidad es muy sencilla.

Se trata de un producto único, ya que la mayoría de productos similares tienen un solo enfoque, que en la mayoría de casos suele ser el enfoque para usuarios a gran escala, dejando de lado el apartado doméstico.

2.2. Descripción del problema

El problema de	Mostrar, almacenar, procesar cuando sea necesario y actuar sobre el estado de la planta
Afecta a	Usuarios con planta, ya sean aficionados o trabajadores del sector.
Lo cual tiene como impacto	Es necesario un trabajo constante por parte del usuario para que la planta se mantenga saludable. Se podría mejorar mucho la salud de la planta si se riega de forma eficiente.
Una solución satisfactoria sería	Poder monitorizar los parámetros más importantes que afectan a la salud de la planta y que esta pueda regarse de forma autónoma e inteligente gracias a un sistema.

Tabla 2.1: Descripción del problema

2.3. Descripción del producto

Para	Cualquier usuario preocupado por la salud de una planta.
Los cuales	Necesitan un sistema para poder monitorizar en tiempo real el estado de la planta y actuar sobre ella.
El	Sistema de monitorización y riego.
Que	Monitoriza y riega una planta de forma autónoma.
Frente a	Productos que se centran solo en la monitorización de campos agrícolas a gran escala y a un precio muy elevado.
Nuestro producto	Permite no solo monitorizar, sino también regar, por un precio menor y siendo también posible su uso a pequeña escala.

Tabla 2.2: Descripción del producto

Capítulo 3

Descripción de participantes y usuarios

3.1. Resumen de los participantes

Nombre	Representa	Rol
Jefes de la empresa	Persona que encarga el proyecto.	Encargados de definir los requerimientos del sistema, aceptar los presupuestos y validar el proyecto...
Jefe del proyecto	Persona encargada de dirigir y desarrollar el proyecto.	Usando los parámetros establecidos por los jefes de la empresa, debe diseñar, gestionar y desarrollar el proyecto.
Usuario Final	Persona que usará el proyecto desarrollado.	Dar retroalimentación del proyecto durante la fase de desarrollo y usar este de forma funcional cuando este terminado.

Tabla 3.1: Resumen de los participantes

3.2. Resumen y entorno de los usuarios

Nombre	Descripción	Participante
Jefes de la empresa	Verificará que el proyecto esta completado y cumple todas los requisitos que expuso.	Representados por: Jefe de la empresa.
Usuario Final	Usuario general que usará el sistema	Representados por: Usuario Final

Tabla 3.2: Resumen y entorno de los usuarios

3.2.1. Entorno de los usuarios

Los usuarios pueden tener más o menos conocimientos en el área, ya que puede ser un aficionado a la agricultura o dedicarse de forma profesional a ello, por lo que esto debe de tenerse en cuenta al desarrollar el proyecto, sobretodo a la hora de la interfaz. En cuanto a plataformas, la interfaz se ejecutará en un PC, por lo que este será necesario para el funcionamiento del sistema.

3.3. Perfiles de los participantes

3.3.1. Jefe de la empresa

Representante	Eva González Parada / Ignacio Herrero Reder
Responsabilidades	Revisar y dar realimentación al jefe de proyecto. Validar que los requisitos se cumplen.
Criterio de Éxito	Poder monitorizar una planta y que esta se riegue de forma autónoma. Cumplir las fechas de entrega establecidas.
Entregables	Documentación del proyecto así como el proyecto en si (código)
Comentarios	Será el encargado de aprobar la finalización del proyecto.

Tabla 3.3: Perfil del jefe de la empresa

3.3.2. Jefe del proyecto

Representante	Sergio Gasquez Arcos
Responsabilidades	<i>Background</i> tecnológico.
Criterio de Éxito	Diseñar, gestionar y desarrollar el proyecto.
Entregables	Conseguir alcanzar los requisitos establecidos y validarlos.
Comentarios	Presupuestos, documentación de planificación y de los requisitos bien detallada.

Tabla 3.4: Perfil del jefe del proyecto

3.3.3. Usuario final

Representante	Usuario general
Responsabilidades	No tiene responsabilidad como tal, pero será quien le dé uso al resultado del proyecto.
Criterio de Éxito	Poder monitorizar una planta y que esta se riegue de forma autónoma.
Entregables	
Comentarios	

Tabla 3.5: Perfil del usuario final

3.4. Alternativas y competencia

3.4.1. Qampo

Qampo [8] es una empresa que se dedica a la venta de sistemas de monitorización y análisis de parámetros agronómicos para optimizar la producción. Es un producto enfocado para grandes cultivos, con sensores de gran precisión , por lo que su precio es mucho más elevado, y sin posibilidad de actuar sobre las plantas, tan solo monitoriza.

3.4.2. Agrae

De nuevo, aGrae [9] es una compañía que se centra en la monitorización de grandes cultivos, aunque esta incorpora un mecanismo para estimar la evolución de los estados fenológicos (para predecir posibles plagas), sin embargo sigue sin cubrir el cultivo de plantas a baja escala y el factor de que se pueda regar de forma autónoma.

Capítulo 4

Requisitos

4.1. Diagrama general

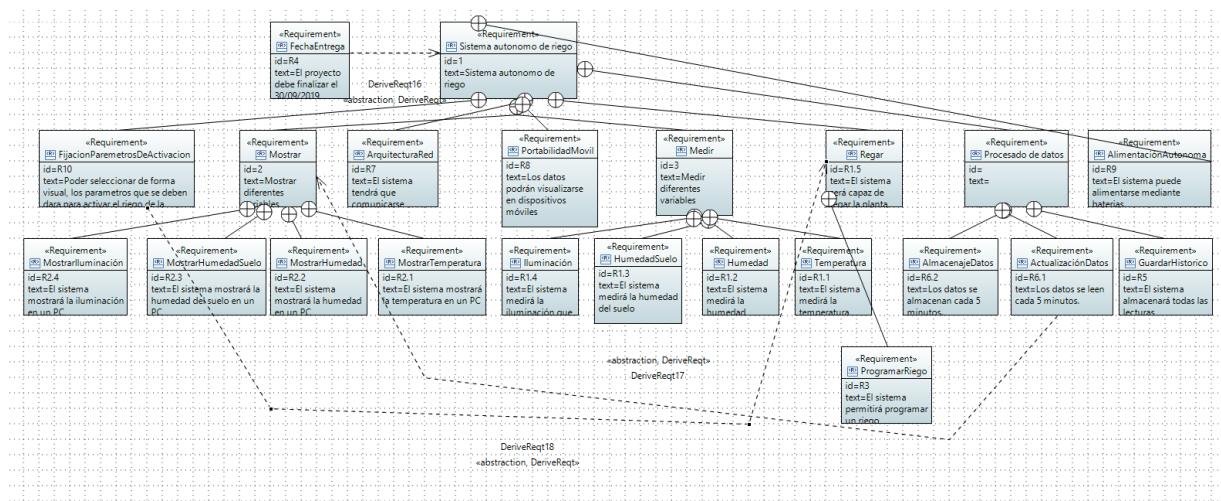


Figura 4.1: Diagrama general de los requisitos del proyecto

4.2. Precedencia y prioridad

Id	Nombre	Prioridad	Precedencia
R1	Medición	F	
R1.1	Temperatura	F	
R1.2	Humedad	F	
R1.3	HumedadSuelo	F	
R1.4	NivelAgua	O	
R2	Mostrar	F	R1
R2.1	MostrarTemperatura	F	R1.1
R2.2	MostrarHumedad	F	R1.2
R2.3	MostrarHumedadSuelo	F	R1.3
R2.4	MostrarNivelAgua	O	R1.4
R3	Regar	F	
R3.1	ProgramarRiego	O	R3
R4	FijacionParametrosActivacion	O	R3
R5	FechaEntrega	F	
R6	ProcesadoDatos	F	R1.*
R6.1	ActualizacionDatos	F	R1.* R2.*
R6.2	AlmacenajeDatos	F	R1.*
R6.3	GuardarHistorico	F	R1.*
R7	ArquitecturaRed	F	
R8	PortabilidadMovil	O	

Tabla 4.1: Precedencia y prioridad

***Nota:** F = Fundamental O = Optativo

4.3. Requisitos Funcionales

4.3.1. Medición

- **Temperatura:** El sistema estará habilitado para sensar la temperatura ambiente.
- **Humedad relativa:** El sistema medirá la humedad relativa del ambiente.
- **Humedad del suelo:** El sistema usará un sensor para obtener la humedad del suelo en el lugar en el que se encuentra la planta.
- **Nivel de agua:** Un extra para el proyecto sería poder saber el nivel de agua en el tanque de riego.

4.3.2. Mostrar

- **Temperatura:** El sistema estará habilitado para mostrar la temperatura ambiente.
- **Humedad relativa:** El sistema mostrará la humedad ambiente.
- **Humedad del suelo:** El sistema mostrará la humedad del suelo en el lugar en el que se encuentra la planta.
- **Nivel de agua:** En caso de desarrollar este extra, el sistema enseñará por la interfaz el nivel de agua del tanque.

4.3.3. Regar

- **Regar:** Un requisito fundamental y primordial del sistema es tener la habilidad de poder regar la planta.

- **Fijación de parámetros de activación:** Una de las funcionalidades que sería muy interesante añadirle es la poder fijar umbrales que hagan saltar el riego de la planta.

4.3.4. Procesado de datos

- **Actualización de datos:** El sistema actualizará de forma periódica la interfaz gráfica donde se muestran los datos.
- **Almacenaje de datos:** Como funcionalidad añadida sería interesante que el sistema tuviese la capacidad de almacenar los datos.

4.3.5. Arquitectura de la red

Se usará una red con dos nodos, el PC y el microcontrolador, los cuales se comunicarán usando el protocolo MQTT a través de Wi-Fi. Hay que destacar, que el proyecto permite implementar, de forma sencilla, más nodos a la red, pero por limitación de material en el sistema desarrollado solo se ha usado un microcontrolador.

4.4. Requisitos No Funcionales

4.4.1. Fecha de entrega

El proyecto debe finalizarse antes del 30/09/2019.

Capítulo 5

Casos de uso

5.1. Actores del sistema

Nombre	Descripción
Usuario	Grupo de personas que usaran el sistema final
Planta	Planta en la que se instalará el sistema
Tiempo	Reloj interno que invoca a una frecuencia dada ciertos casos de uso.

Tabla 5.1: Actores del sistema

5.2. Diagrama general

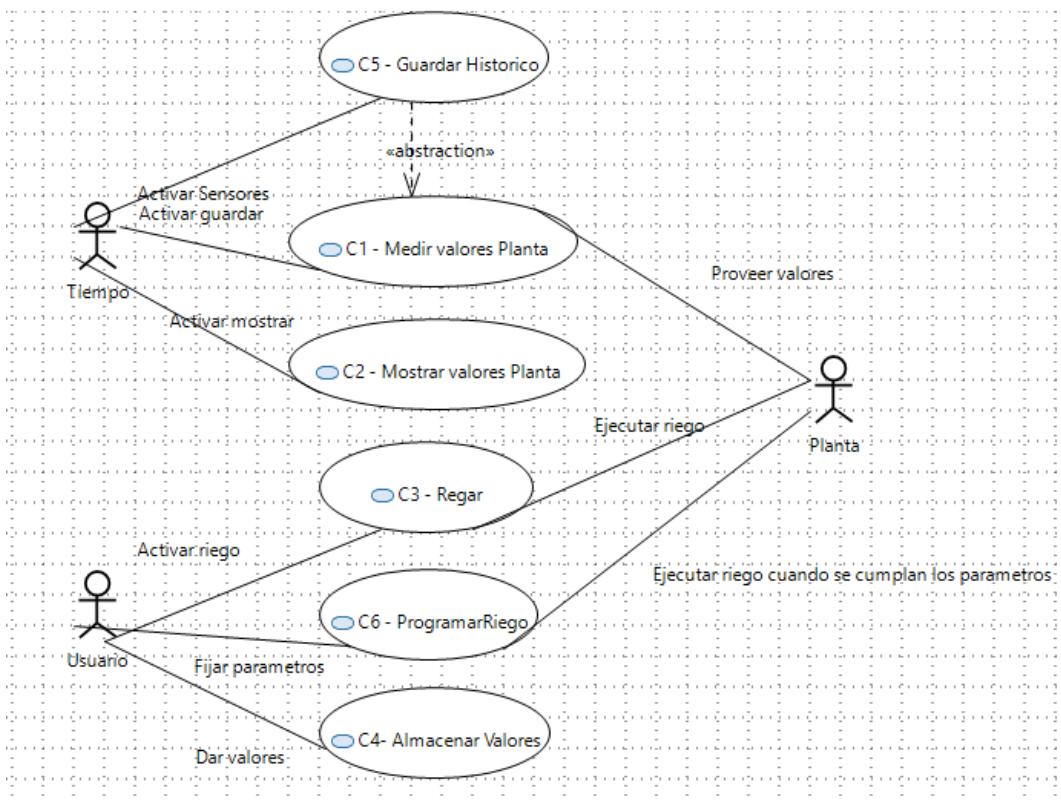


Figura 5.1: Diagrama general de casos de uso

5.3. Descripción textual de los casos de uso

5.3.1. Medir los valores de la planta

- **Contexto de uso:** Se fijará un periodo de medición de los datos usando los sensores instalados en la planta.
- **Actor principal:** Tiempo
- **Participantes y Objetivos:**

Participante	Objetivo
Tiempo	Activará la lectura de datos de la planta cuando sea necesario
Planta	Usando los sensores, proveerá los datos.

Tabla 5.2: Diagrama general de casos de uso

- **Pre Condiciones:** Sensores correctamente instalados y periodo de medición fijado.
- **Garantías mínimas:** Se almacenan los valores proporcionados por los sensores de la planta
- **Escenario de éxito principal:**
 - El tiempo solicita la medida de valores de los sensores de una planta.
 - El sistema de motorización se lo comunica a los sensores.
 - Los sensores realizan su medición y devuelven sus resultados.
 - El sistema actualiza su valor interno del sensor.
- **Escenario secundario 1:** El sensor, por alguna razón, no devuelve resultados.
 - El sistema actualiza mal o no actualiza su valor interno.
- **Escenario secundario 2** El sensor, por alguna razón, devuelve resultados que no tienen sentido.
 - El sistema actualiza mal actualiza su valor interno.

5.3.2. Mostrar los valores de la planta

- **Contexto de uso:** Se fijará un periodo de actualización de los datos mostrados
- **Actor principal:** Tiempo

■ **Participantes y Objetivos:**

Participante	Objetivo
Tiempo	Activará la actualización de datos de la planta cuando sea necesario

- **Pre Condiciones:** Tener datos en las variables internas del sistema
- **Garantías mínimas:** El sistema muestra los valores de sus variables internas en la interfaz.
- **Escenario de éxito principal:**
El tiempo solicita la actualización de valores en la interfaz.
El sistema de motorización actualiza la interfaz con los valores más actuales.
- **Escenario secundario 1:**
El tiempo solicita la actualización de valores en la interfaz.
El sistema actualiza mal o no actualiza los valores de la interfaz

5.3.3. Regar

- **Contexto de uso:** El usuario desea regar la planta
- **Actor principal:** Usuario
- **Participantes y Objetivos:**

Participante	Objetivo
Usuario	Activará el riego de la planta
Planta	Regarse

- **Pre Condiciones:** Sistema de riego funcional y operativo.
- **Garantías mínimas:** El usuario podrá activar el riego de la planta.

- **Escenario de éxito principal:**

El usuario solicita el riego de la planta.

El sistema activa el riego y la planta se riega.

- **Escenario secundario 1:**

El usuario solicita el riego de la planta.

El sistema no recibe esta llamada o la recibe mal y la ejecuta mal o no la ejecuta.

5.3.4. Almacenar valores

- **Contexto de uso:** El sistema almacena los valores obtenidos de los sensores tras comprobar que son válidos.

- **Actor principal:** Usuario

- **Participantes y Objetivos:**

Participante	Objetivo
Usuario	Almacenar los valores cuando son correctos.

- **Pre Condiciones:** Sistema programado correctamente.

- **Garantías mínimas:** Los datos se almacenan si son coherentes.

- **Escenario de éxito principal:**

El sistema verifica los rangos de valores esperados para cada sensor.

El PC los almacena.

- **Escenario secundario 1:**

El sistema recibe datos fuera del rango que los sensores pueden medir.

La medida queda descartada.

5.3.5. Guardar Histórico

- **Contexto de uso:** El usuario podrá almacenar los valores de las variables medidas por los sensores.
- **Actor principal:** Tiempo
- **Participantes y Objetivos:**

Participante	Objetivo
Tiempo	Activar el almacenaje con la frecuencia fijada.

- **Pre Condiciones:** Sistema con espacio disponible y el mecanismo de guardado de datos funcionando correctamente.
- **Garantías mínimas:** Datos almacenados
- **Escenario de éxito principal:**
 - El tiempo activa la lectura de datos cada vez que toque.
 - El sistema comunica la medición a los sensores.
 - Los sensores devuelven su resultado.
 - El sistema los transmite al PC.
 - El PC los almacena.
- **Escenario secundario 1:**
 - El tiempo no activa correctamente la medición de datos y esta llamada es ignorada.
- **Escenario secundario 2:**
 - El tiempo activa la lectura de datos cada vez que toque.
 - El sistema comunica la medición a los sensores.
 - Los sensores devuelven su resultado.
 - El sistema los transmite al PC.
 - El PC no consigue escribir el archivo

5.3.6. Fijar parámetros de riego

- **Contexto de uso:** El usuario podrá fijar umbrales que activen el riego de la planta
- **Actor principal:** Usuario
- **Participantes y Objetivos:**

Participante	Objetivo
Usuario	Que el riego se active en determinados casos
Planta	Regar la planta

- **Pre Condiciones:** Sistema de riego correctamente conectado y en funcionamiento, así como la funcionalidad de controlar los umbrales correctamente programada.
- **Garantías mínimas:** La planta se riega cuando se dan las condiciones programadas.
- **Escenario de éxito principal:**

El usuario fija unos umbrales.
El sistema realiza medidas periódicamente, si alguna variable supera este umbral, se activa el riego.
La planta se riega.
- **Escenario secundario 1:** El usuario fija unos umbrales.

El sistema realiza medidas periódicamente, pero obtiene una respuesta errónea que no se detecta como tal.
La planta se riega de más o no se riega.

Capítulo 6

Arquitectura

6.1. Arquitectura Lógica

El sistema se dividirá en dos aplicaciones principales, una se ejecutará en un PC mientras que la otra se ejecutará en el microcontrolador, en este caso el CC3200.

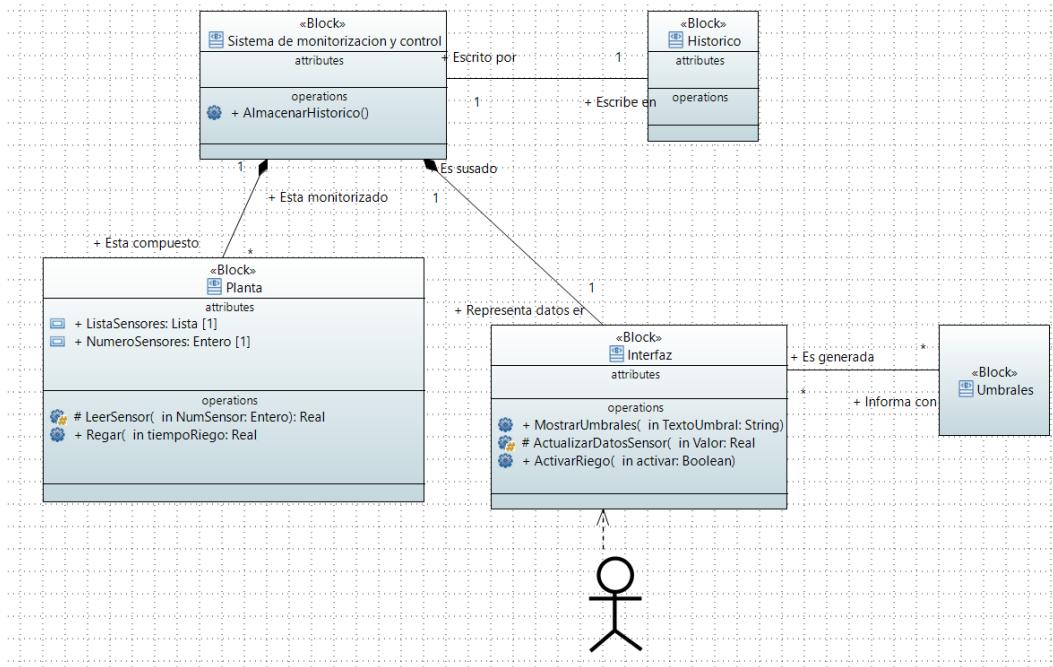


Figura 6.1: Arquitectura lógica general del proyecto

- **Sistema de monitorización y control:** Se trata del conjunto del software, teniendo en cuenta sus dos partes, la que se ejecuta en el PC y la que se ejecuta en el Microcontrolador. El sistema será capaz de monitorizar y regar la planta.
- **Planta:** Modela la planta en la que se instalará el sistema, es decir, donde se colocarán los sensores.
- **Interfaz:** Representa la interfaz gráfica en la que se podrá observar el valor de las variables así como dar ordenes de regado.
- **Umbrales:** Simboliza los umbrales que el usuario podrá fijar para activar el riego.
- **Histórico:** Modela el archivo en el que se almacenarán los valores de los sensores.

6.2. Arquitectura Física

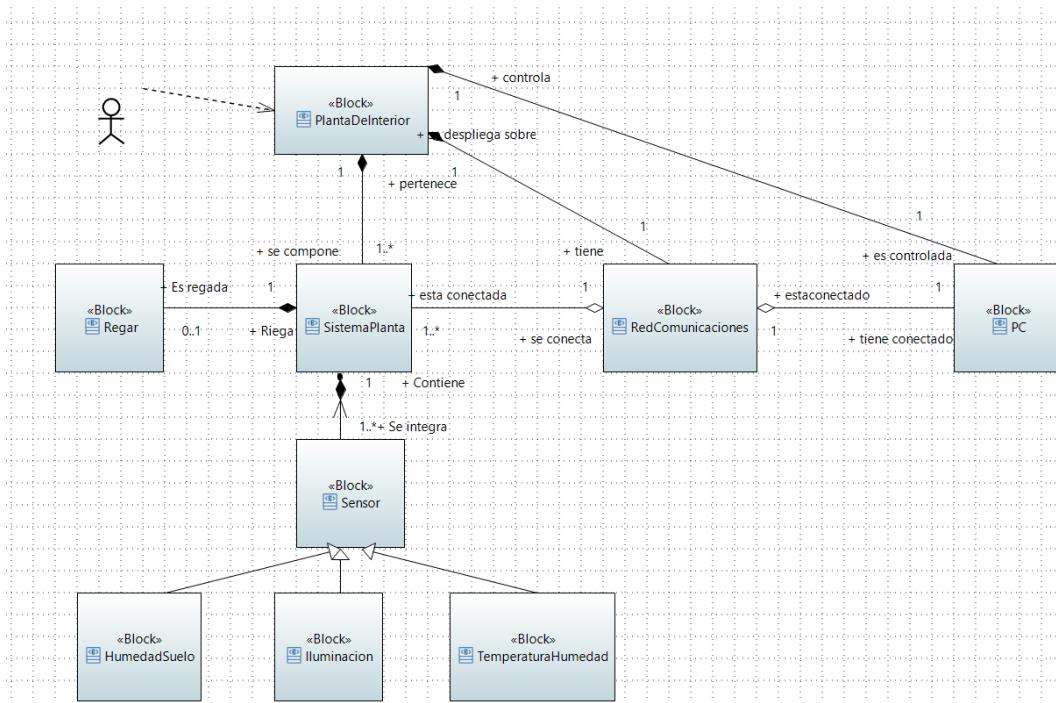


Figura 6.2: Arquitectura física general del proyecto

- **Planta de interior:** Hace referencia al entorno que deseamos monitorizar y controlar, es el conjunto de la planta con el sistema diseñado para su riego autónomo.
- **Planta:** Representa el sistema instalado en la planta, es decir el microcontrolador con los sensores.
- **Red de Comunicaciones:** Se trata de la red de comunicación que usarán el PC y el microcontrolador para comunicarse.
- **PC:** Como se comentó con anterioridad, la interfaz del sistema será ejecutada en un PC, desde el que se podrán visualizar el valor de las diferentes variables así como actuar sobre el sistema.
- **Regar:** Representa el sistema que controlara el riego de la planta.

■ **Sensor:** Se trata del conjunto de todos los sensores que se usarán en el sistema.

- *Sensor de humedad y temperatura:* Se usará un sensor de temperatura y humedad el cual estará conectado al microcontrolador, el sensor seleccionado para tal tarea es el SHT31.
- *Sensor de humedad de suelo:* Para sensar la humedad del suelo se usará el modelo SEN0193, el cual es un sensor capacitivo que da una salida analógica de entre 1.2 V y 2.5 V en función de la húmeda del suelo.
- *Sensor nivel de agua:* Para saber el nivel de agua del tanque donde se almacenará el agua que se use para regar la planta se usará el sensor HC-SR04, de forma que gracias a los ultrasonidos, se pueda saber el nivel de agua en el tanque.

Capítulo 7

Descripción del sistema

En este apartado se describirán todos los componentes usados, citando y describiendo sus principales características así como el motivo por el cual se han elegido dichos componentes en lugar de otros.

A continuación se podrá ver un diagrama general del proyecto, en el que solo se representará una estación, aunque se podrían manejar más de una, con cada uno de sus módulos principales:

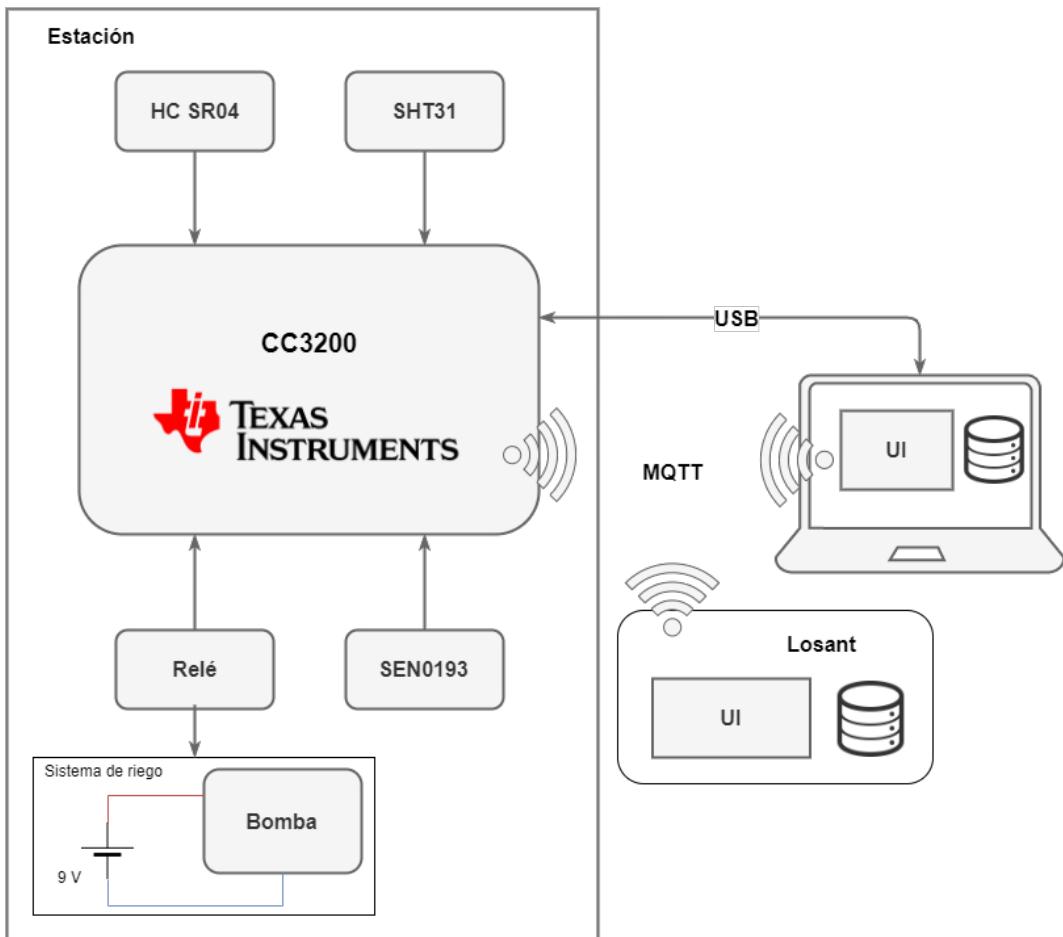


Figura 7.1: Diagrama general del proyecto

El sistema está formado por 5 módulos interconectados y con una unidad de procesamiento, el CC3200 Launchpad. Se dispondrán de 3 sensores con los cuales se recolectarán las variables más importantes y un módulo de riego.

En la Fig.7.1, se puede observar como quedaría conectada una estación con el PC en el que ejecuta la interfaz gráfica, así como los diferentes módulos de la estación. Es importante destacar, que podría haber más clientes, ya sean en forma de estaciones o cualquier dispositivo que pueda recibir tramas MQTT.

El ultimo bloque, representa la plataforma Losant, la cual aporta una interfaz web y una base de datos *online*, este se conecta mediante MQTT, gracias a la configuracion realizada en la arquitectura MQTT que posteriormente se comentará.

7.1. Alimentación del sistema

El sistema se puede alimentar de varias formas, algunas de ellas incluso podrían hacer que el sistema sea portable y autónomo, aunque no se han llegado a utilizar en este proyecto.

La alimentación del sistema se ha obtenido vía USB, conectando un PC con el terminal de microUSB del CC3200. El cable USB esta formado por cuatro hilos, dos de alimentación (positivo y negativo) y 2 de datos, por los que se enviará el programa tras compilarse desde el PC a la placa (CC3200 no almacena el código cuando se reinicia debido a su arquitectura de memoria). El USB [10] proporciona 5V y una corriente máxima de 500mA si es USB 2.0 y 900 mA en caso de ser USB 3.0. Ambos serán suficientes para alimentar el sistema.

Otra de las maneras de alimentar el sistema sería usando una batería cuya salida fuese un USB y cargarle el programa a la placa de forma alternativa, de forma que conseguiríamos que el sistema fuese portable.

El módulo de riego se alimentará de forma independiente mediante una batería de 9V debido a que su consumo es demasiado elevado para poder suplirlo con el USB.

7.2. CC3200 Launchpad

La placa CC3200 de Texas Instrument es un kit de desarrollo con conexión inalámbrica a través de Wi-Fi. Integra 2 sensores (un acelerómetro y un termómetro) además de tener entornos de desarrollo adaptados a la placa como *Code Composer Studio*, el cual se ha usado para el desarrollo del sistema.

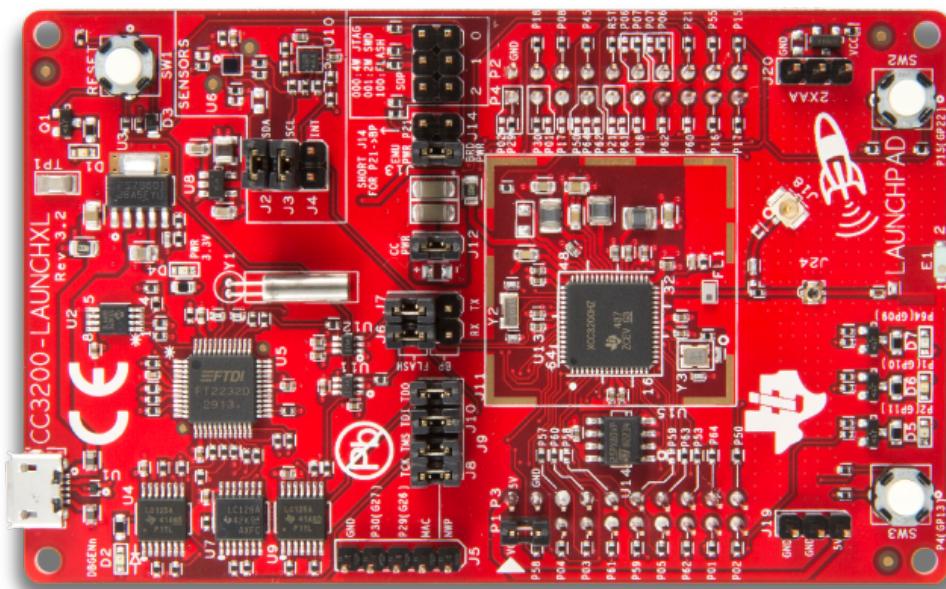


Figura 7.2: CC3200 Launchpad

Como principales características del CC3200 cabe destacar:

- Procesador ARM Cortex-M4 Core a 80 MHz
 - 27 pines de entrada/salida de propósito general.
 - 4 entradas analógicas con resolución de 12 bits.
 - Wi-Fi.
 - Memoria RAM (*Random Access Memory*) de hasta 256KB.

- Interfaz I2C (*Inter-Integrated Circuit*) [11].
- Interfaz SPI (*Serial Peripheral Interface*) [12].
- Conector tipo mini-USB.
- Botón de reinicio.

7.3. Sensor de humedad en suelo SEN0193

Se trata de un sensor capacitivo el cual se usará para medir la humedad en la tierra, esta fabricado con un material resistente a la corrosión y opera entre 3.3 V y 5V. Ofrece una salida analógica que puede variar entre 1.2 V y 2.5 V [13].



Figura 7.3: Sensor de humedad en suelo SEN0193

Las conexiones que se realizaran serán las siguientes:

SEN0193	CC3200
VDD	5 V
GND	GND
AOUT	Divisor de tensión

Tabla 7.1: Conexiones del sensor de humedad en suelo SEN0193

Para adaptar la salida del sensor a una señal viable para el convertidor analógico digital del CC3200 (el cual solo lee señales entre 0 V y 1.4 V) [14] se usará el siguiente divisor de tensión:

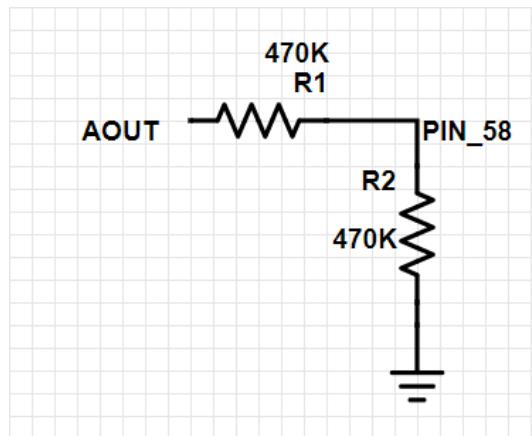


Figura 7.4: Divisor de tensión diseñado para el sensor de humedad en suelo SEN0193

Con esto se convertiría la señal de salida del SEN0193 al rango 0.6 V a 1.25 V, aunque se este perdiendo bastante precisión, este rango será más que suficiente para el propósito del sistema que solo busca fijar rangos de humedad. Si se hubiese tenido acceso a componentes electrónicos como amplificadores operacionales, la opción implementada habría sido más óptima.

7.4. Sensor de temperatura y humedad relativa en ambiente SHT31

La serie de sensores SHT3X es una gran conocida entre los sensores de temperatura y humedad en ambiente debido a su módico precio y su alto rendimiento. Incorporan una interfaz I2C y tienen un consumo bastante bajo, $5\mu\text{W}$ durante una medición. Opera con unas temperaturas entre los -40 °C y 90 °C y humedades entre el 0 % y el 100 %. Su precisión típica es de $\pm 1.5\%$ para la humedad relativa y de $\pm 2\text{ }^{\circ}\text{C}$ en la temperatura [15].

A continuación se presentara la disposición de pines y el diagrama del SHT31:

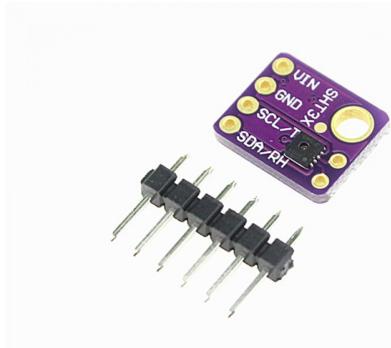


Figura 7.5: Disposición de los pines del SHT31

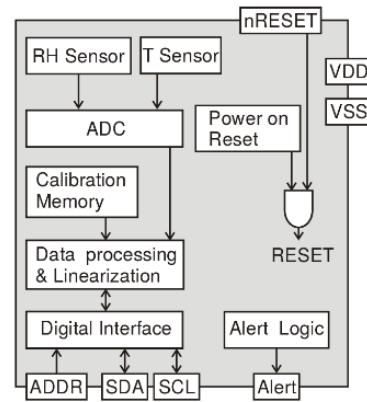


Figura 7.6: Esquemático del SHT31

Las conexiones realizadas con el CC3200 son las siguientes:

SHT31	CC3200
VIN	5 V
GND	GND
SCL	PIN_01
SDA	PIN_02

Tabla 7.2: Conexiones del SHT31

7.5. Sensor de ultrasonidos HC SR04

Para obtener el nivel de agua del tanque se usará medirá la distancia entre la cima del tanque y el agua, usando este sensor de ultrasonido [16].

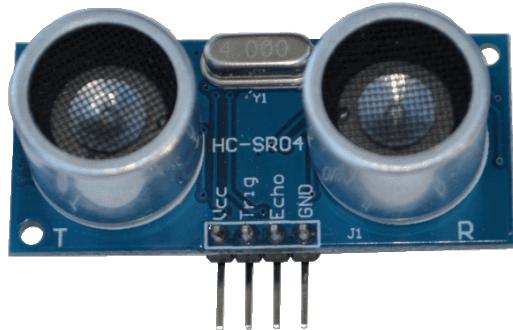


Figura 7.7: Sensor de ultrasonidos HC SR04

El cual funciona de la siguiente manera:

1. Se genera un pulso, de al menos $10\mu S$, en el pin del *Trigger*
2. El modulo genera y envía automáticamente 8 pulsos a 40 kHz y detecta si hay un pulso de vuelta
3. Si la señal ha vuelto, se pone en alto el pin de Echo durante un tiempo proporcional a la distancia medida.

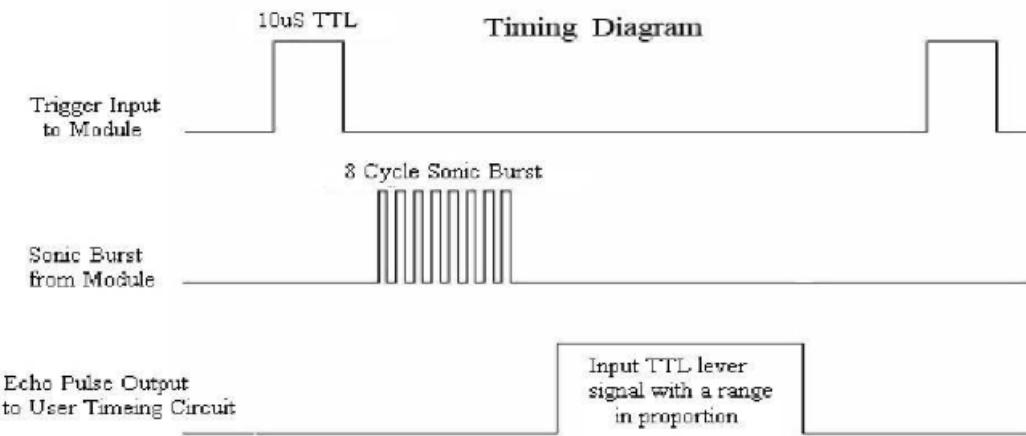


Figura 7.8: Funcionamiento del sensor de ultrasonidos HCSR04 [17]

HC SR04	CC3200
VCC	5 V
Trig	PIN_64
Echo	Divisor de tensión
GND	GND

Tabla 7.3: Conexiones del sensor de distancia HC SR04

Se debe usar un divisor de tensión, ya que el pulso generado por el pin de Echo es de 5V, lo cual no es apto para la CC3200.

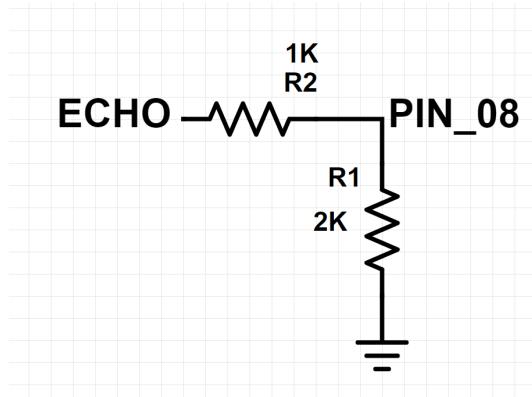


Figura 7.9: Divisor de tensión para adaptar el pulso de Echo

Al realizar este acondicionamiento, el sensor pasa a ser bastante menos preciso [18], esto no es critico para el sistema ya que no es especialmente importante dicha medida.

7.6. Sistema de riego

El sistema de riego, será alimentado de forma externa, ya que la bomba requiere más potencia de la que la placa CC3200 puede suministrar, para ello se usará una pila de 9 V y un relé controlado por un pin del microcontrolador. El circuito resultante es el siguiente:

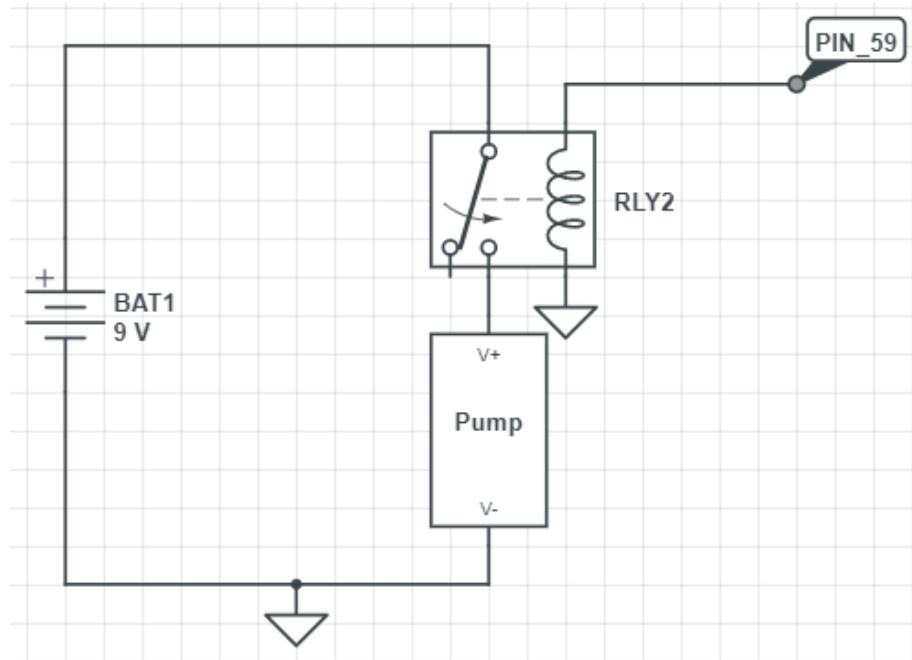


Figura 7.10: Circuito usado para el sistema de riego

El relé se encuentra en la posición NO (*Normally Open*) de normal, es decir, mientras que no activemos el relé el circuito estará abierto.

Capítulo 8

Desarrollo Firmware/Software

8.1. Aplicación de microcontrolador

Para el desarrollo del código que se ejecutará en la CC3200 Launchpad se ha usado el entorno de desarrollo Code Composer Studio y el lenguaje de programación C [19].

Se usará como base el código proporcionado en la asignatura Intensificación de microcontroladores el cual se basa en FreeRTOS [20], un sistema operativo de código libre.

El código del proyecto, así como el resto de partes ha sido desarrollado usando control de versiones git mediante *GitHub*.

Para tener de manera mas organizada el código, se han realizado varias interfaces, una para cada sensor, que se han almacenado en archivos diferentes, de forma que se pueden exportar fácilmente. Como estilo se ha usado *camelCase* para la definición de variables y funciones .

A continuación se describirá el código principal del sistema, y, posteriormente se detallará el funcionamiento de los módulos por separado, de

forma muy simplificada, el diagrama de flujo del código es el siguiente:

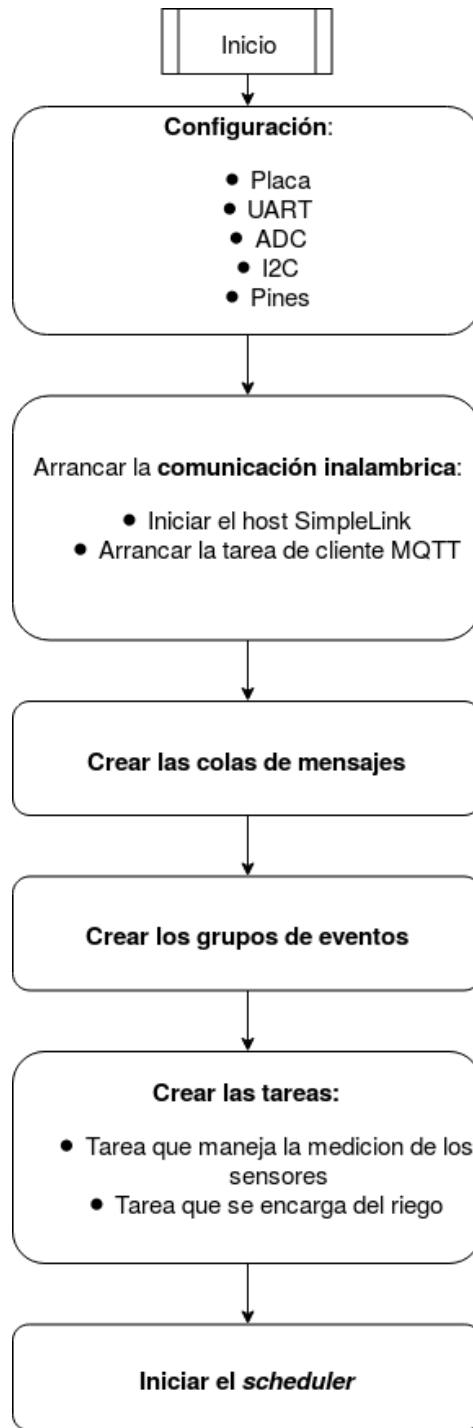


Figura 8.1: Diagrama del código principal

Se han usado los siguientes mecanismos IPC (*Inter-Process Communication*):

- **Timers:** Para capturar el tiempo que dura un pulso.
- **Colas de mensajes:** Para compartir información entre diferentes tareas y para almacenar los datos que se deben enviar se han usado colas de mensajes.
- **Grupos de *flags* de eventos:** Que bloquearan y desbloquearan el código de forma que las esperas sean no-bloqueantes.

Tras arrancar el microprocesador y cargarle el código, se comienza la configuración del sistema:

- Placa: Se habilitan las interrupciones y se configura la MCU (*Microcontroller Unit*).
- UART (Universal Asynchronous Receiver-Transmitter): Configura los pines necesarios para la transmisión serie UART [21].
- ADC (Analog Digital Converter): Configura el pin que se usara para la lectura analógica.
- I2C: Configura los pines SDA (*Serial Data*) y SCL (*Serial Clock*) y habilitamos la biblioteca que gestiona el I2C.
- Pines: Se configuran el resto de pines que se usarán para el sistema (el pin del relé, los pines del sensor de nivel de agua...)

Tal y como se comentó anteriormente, se han usado colas de mensajes para compartir información entre las diferentes tareas, esto se ha realizado usando colas unitarias cuyo tamaño es la longitud del dato que deseamos almacenar, se han realizado tres colas con este propósito:

- Frecuencia: Para almacenar la frecuencia con la que se sensan las variables y se envían los datos.
- Umbrales: Cuando un usuario fija unos umbrales a los que desea que se riegue.
- Tiempo de riego: Se almacena el intervalo de riego (en milisegundos) que el usuario ha decidido.

Estas colas se escriben y leen usando las funciones `xQueueOverwrite()`[22] y `xQueuePeek()`[23] de forma que siempre se mantiene en la cola el dato más actual. También se usará una cola en la que se almacenan los mensajes que se desean enviar, esta será de tamaño 20, este tamaño será más que suficiente ya que como mucho mediremos cada minuto, ya que no es de interés medir a más frecuencia para monitorizar una planta.

El sistema usa también 2 grupos de eventos:

- Medidas: Este grupo de eventos de eventos dicta cuando se medirán los sensores y cuando no, esto dependerá de si el usuario decide iniciar las medidas o decide detenerlas.
- Riego: Existe un grupo de eventos que controla cuando se debe activar el riego, al activar esta bandera desbloqueará una tarea encargada del riego.

Finalmente el sistema cuenta con 2 tareas, las cuales serán explicadas de forma detallada.

8.1.1. Tarea de sensado

Esta tarea realiza una espera, de forma pasiva, a que se activen las mediciones de los sensores, una vez activadas, pone en la cola de mensajes a enviar el mensaje `SENSORS_REPORT`.

La función que maneja la cola de mensajes MQTT a enviar, se desbloqueará y procederá a recolectar los datos de todos los sensores, los almacenará en un *struct* creado para el almacenamiento de todos los datos en una sola variable y, finalmente, enviará esta variable a través de la red.

Tras, el envío del mensaje, se comprueba cual es el ultimo dato recibido en la cola que almacena las frecuencias y duerme la tarea durante el tiempo que dicha variable dicte.

8.1.2. Tarea de riego

Al igual que la tarea de sensado, espera de forma no bloqueante a que el usuario, active desde la interfaz el riego, o esta se active debido a que se han superado los umbrales fijados por el usuario. Tras esto, comprueba el tiempo de riego de la cola destinada a este propósito y activa el relé durante el tiempo de riego leído de la cola. La tarea se vuelve a bloquear hasta que un *flag* vuelva a activarse.

8.1.3. Arquitectura de red

Como se ha comentado anteriormente, la comunicación se realiza a través de Wi-Fi, gracias al modulo Wi-Fi de la CC3200, usando el protocolo MQTT. En proyecto realizado solo se ha monitorizado una planta, por lo que se usa un solo *topic* en el que se publicarán los resultados de los sensores, "Station1". Si se desease agregar alguna otra planta, tan solo haría falta agregar otro *topic* y definir cuando se debe publicar en cada uno de los dos *topics*, esto hace que escalar el sistema sea muy sencillo.

8.1.4. Interfaces

A continuación se comentará las diferentes interfaces desarrolladas para el uso de los sensores elegidos para el proyecto. Se ha de destacar que todas las interfaces contienen una directiva, que activa las trazas para facilitar su depuración, pero para en el sistema final estarán deshabilitadas.

Sensor de temperatura y humedad relativa en ambiente

La comunicación se realiza a través de I2C, tal y como se vio en la Sec.7.4, su dirección I2C por defecto es 0x44, y incorpora diferentes comandos I2C de los cuales caben destacar [15]:

- *Soft Reset* : permite reiniciar el sensor a través de un comando.
- Medición: El sensor permite hasta 6 modos de medición, dependiendo de su repetibilidad (baja, media y alta), que dictan las medidas por segundo que hace, y una opción llamada *Clock Stretching*, la cual pone a nivel bajo la linea de SCL hasta que la medida se termina. En el caso del sistema se usara alta repetibilidad con la opción de *Clock Stretching* activada.

La respuesta de las mediciones vienen con un CRC (Cyclic Redundancy Check) [24] de 8 bits de tipo CRC-8, con polinomio 0x31 y cuya inicialización es 0xFF.

El flujo para el envío y lectura de un comando es el siguiente:

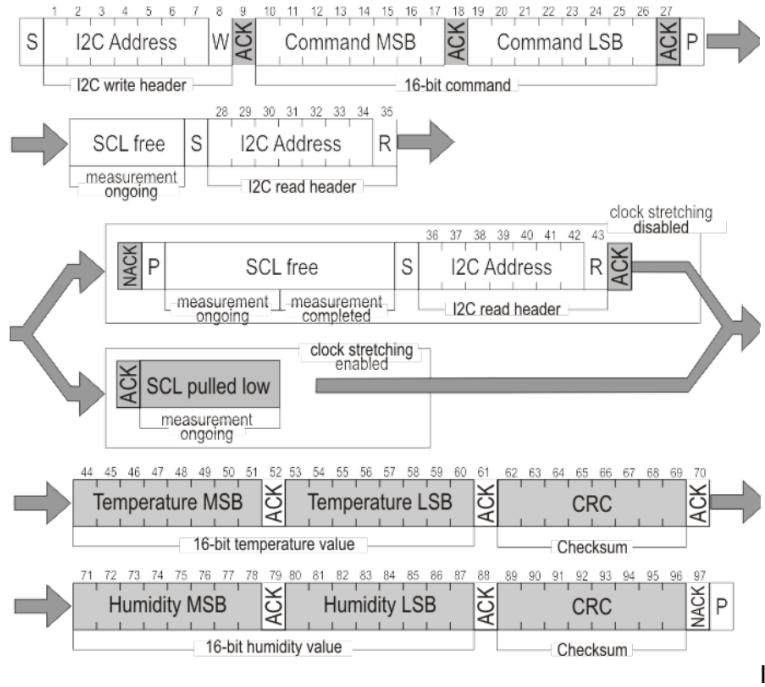


Figura 8.2: Envío y lectura del comando de lectura en alta repetibilidad

Los bloques blancos de la Fig.8.2 son controlados por el microcontrolador, mientras aquellos con el fondo gris son controlados por el sensor. En la figura se puede ver como es el flujo:

1. Se manda la cabecera de inicio junto a la dirección del dispositivo y el bit de escritura
2. Se envía el comando deseado
3. El sensor realizará la medida y mantendrá la linea SCL a nivel lógico 0, hasta que complete la medida (suponiendo que *Clock Stretching* esta activado).
4. Escribimos la dirección del dispositivo con el bit de lectura.
5. Recibiremos la respuesta que se compone de 48 bits (16 bits para el valor de temperatura, 8 para su CRC, 16 bits para el valor de humedad y 8 para su CRC)

La interfaz desarrollada para el sensor se compone de 3 funciones:

- **setupTemperatureSensor()**: Habilita y configura el I2C en la placa.
- **getTemperature()**: Envía el comando de medición, tal y como se explicó anteriormente e interpreta la respuesta. Se apoya en la ultima función para verificar los datos.
- **crcCheck()**: Funcion booleana que calcula y verifica el CRC-8 de los datos recibidos.

Sensor de humedad en suelo

Como se comentó en la Sec.7.3, el sensor de humedad ofrece una salida analógica por lo que para obtener su valor, deberemos usar un convertidor analógico-digital y posteriormente realizaremos la transformación para pasarlo de voltaje a porcentaje.

La tensión se obtendrá del divisor de tensión comentado en la Sec.7.3 (Fig.7.3) por la restricción del ADC del CC3200 de aceptar solo señales de [0, 1.4] V. De nuevo, recalcar que no se necesita mayor precisión en este sensor ya que no es una medida critica.

La interfaz se compone de dos funciones:

- **setupAdc()**: Configura el PIN_58 como entrada analógica y prepara el canal 1 del convertidor analógico (el asociado a este pin).
- **getTemperature()**: Lee el valor del pin, obtiene el valor del voltaje (el valor del voltaje viene en los bits [2, 13]) por lo que se aplica una mascara y posteriormente se convierte este valor en voltaje, sabiendo que el ADC, tiene como fondo de escala 1.4 V y 12 bits de resolución. Finalmente, se convierte este valor a porcentaje para su mejor interpretación.

Sensor de ultrasonidos HC SR04

La interfaz del sensor de ultrasonidos se basa en su funcionamiento comentado en la Sec.7.5. Se ha tratado del mayor reto del proyecto, ya que hay muy poca documentación al respecto y el sensor es bastante delicado. La interfaz cuenta con tres funciones:

- **setupWaterLevel()** : En dicha función se crea una grupo de eventos, el cual se usará para tener constancia de cuando esta calculándose la medida y cuando ha terminado, se configuran los pines que se usan 7.3, el PIN4, será configurado como pin de salida y será el encargado de generar el pulso de *Trigger*, mientras que el PIN_08 será configurado como entrada y con interrupción habilitada para ambos flancos, estas interrupciones activarán la función *echoInt()*, la cual se describe posteriormente.
- **readWaterLevel()** : Es el método encargado de calcular el nivel de agua del tanque, por lo que, tras comprobar que el sistema no se encuentra realizando otra medida en ese momento, genera el pulso necesario en el pin de *Trigger* y tras esto queda en una espera no bloqueante, a que la función *echoInt()* desbloquee dicha espera para procesar el valor final.
- **echoInt()** : Esta función es llamada cada vez que se genera el PIN_08 es interrumpido, ya sea por flanco de subida o por flanco de bajada, por lo que lo primero que debemos hacer, es comprobar que tipo de interrupción se trata. Primero, se recibe una interrupción por flanco de subida, que indica el inicio del pulso que genera el pin *Echo* del sensor, el cual dicta la distancia medida, por lo que se pone el *timer* a 0 y se habilita. Posteriormente, se debe entrar de nuevo en la interrupción, pero esta vez, por flanco de bajada, esto indica que el pulso ya se ha terminado, se obtiene el valor del *timer* y se levantan los *flags* de medida terminada, para desbloquear la espera de la función *readWaterLevel()*.

Como no es una medida precisa debido a la adaptación de los 5 V a los 3.3 V, la medida se toma 5 veces y se calcula su valor medio.

La medida se toma cada vez que la bomba se activa o al iniciar el sistema.

Relé

El relé que activa la bomba de riego se controla a través del PIN_59, el cual está configurado como salida. Para regar, solo se necesita poner en alto el pin comentado en alto durante el tiempo deseado, esto hará que se cierre el circuito de la bomba comentado en la Sec.7.6.

8.2. Interfaz Gráfica

Se ha usado el entorno de desarrollo QtCreator [5] para su desarrollo, debido a las facilidades que ofrece para el desarrollo de interfaces (permite crearlas de forma visual). La interfaz se desarrolló con el propósito de facilitar la comunicación del usuario con el sistema y ofrece las siguientes funcionalidades:

- Opción de elegir el *broker* al que conectarse.
- Botón de *ping* para comprobar la conexión.
- *Switch* para activar/desactivar las medidas de los sensores.
- Un texto que muestra el estado del sistema y los temas MQTT a los que se subscribe.
- En la parte principal de la interfaz, hay 4 pestañas:

- **General:** Esta pestaña permite ver el ultimo valor de los sensores, así como elegir la frecuencia con la que se desea medir.

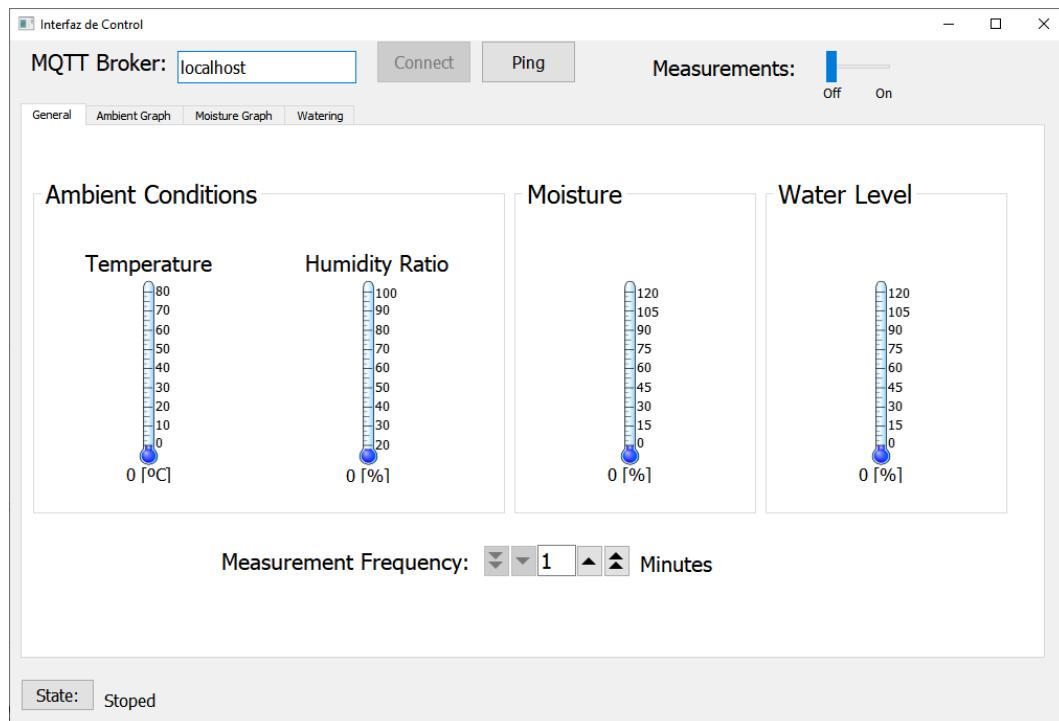


Figura 8.3: Pestaña "General"de la interfaz gráfica

- **Ambient Graph:** La pestaña muestra una gráfica en la que se representan las 300 ultimas muestras de la temperatura y de la humedad relativa en ambiente.
- **Moisture Graph:** Similar a la pestaña anterior, solo que esta vez se representa la humedad en suelo.
- **Watering:** Aquí se podrá fijar el intervalo de riego (en segundos) que se desee así como fijar los umbrales (temperatura y humedad en suelo), los cuales podrán activar el riego de forma automática.

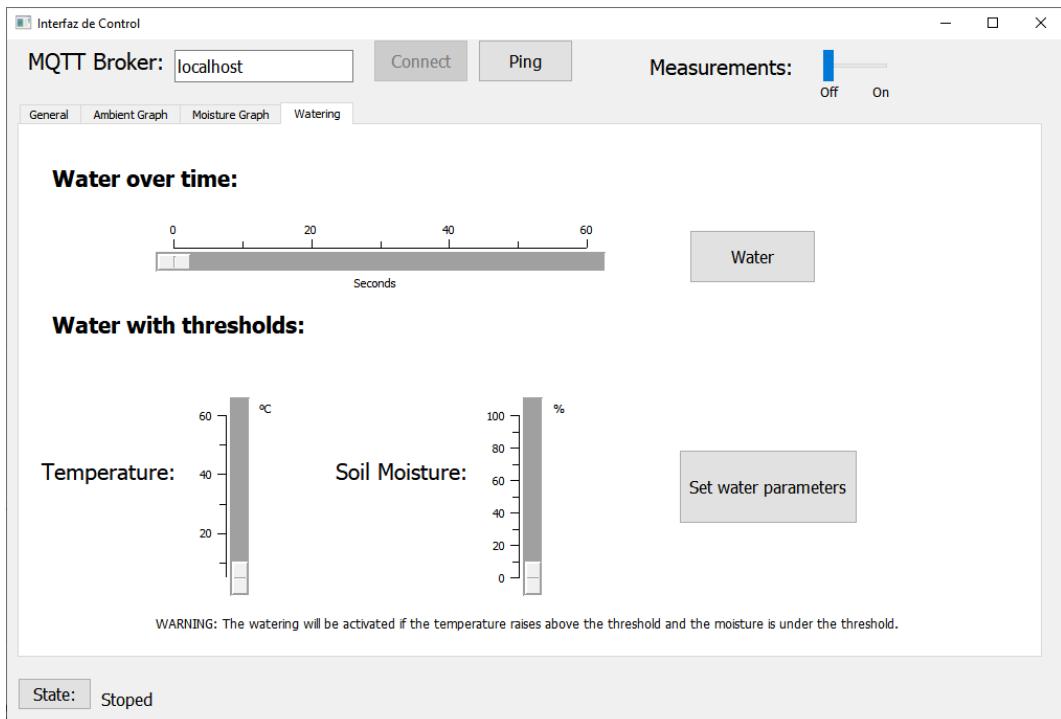


Figura 8.4: Pestaña "Watering"de la interfaz gráfica

La programación se ha realizado usando C++ [25] usando el paradigma de programación orientada a objetos, se ha realizado una clase, *GUIPanel*, la cual creara toda la interfaz y tendrá sus métodos privados los cuales se llamen cuando se interacciona con la interfaz. De forma que en el *main.cpp*, tan solo se creara una instancia de la clase y se llamará a su método publico *show* el cual muestra la interfaz. A continuación se muestran los métodos públicos y privados (Los públicos son accesibles desde cualquier parte del código, mientras que los privados solo se acceden desde dentro de la clase) de la clase:

```

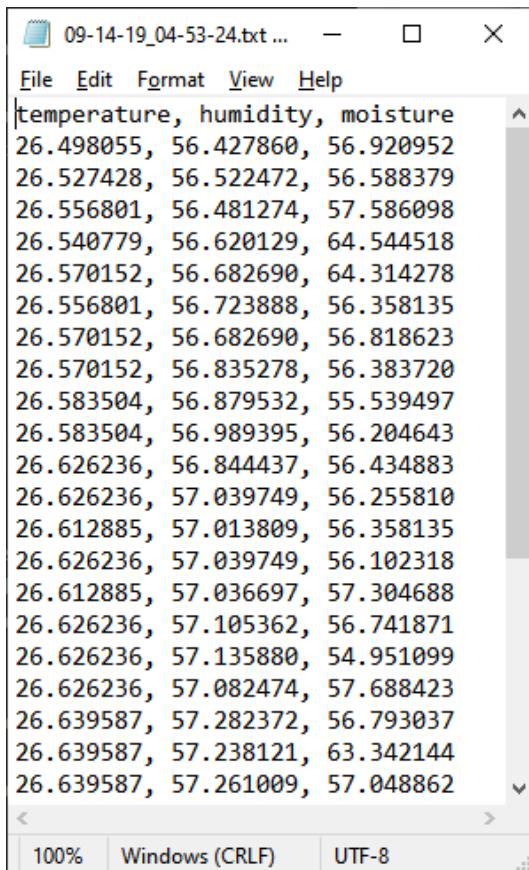
17 class GUIPanel : public QWidget
18 {
19     Q_OBJECT
20
21 public:
22     explicit GUIPanel(QWidget *parent = 0); // Constructor
23     ~GUIPanel(); // Destructor
24
25 private slots:
26     void onMQTT_Received(const QMQTT::Message &message); // Funcion que se ejecuta cuando recibimos un mensaje
27     void onMQTT_Connected(); // Funcion que se ejecuta cuando se conecta al broker
28     void onMQTT_Connacked(quint8 ack); // Funcion que se ejecuta al recibir el ack del Conect
29     void onMQTT_subscribed(const QString &topic); // Funcion que se ejecuta al suscribirse a algun topic
30     void onMQTT_subacked(quint16 msgid, quint8 qos); // Funcion que se ejecuta al recibir el ack del Subscribe
31
32     //Slots botones (creados automaticamente)
33     void on_connectButton_clicked(); // Funcion que se ejecuta al pulsar el boton "Connect"
34     void on_stateButton_clicked(); // Funcion que se ejecuta al pulsar el boton "State"
35     void on_pingButton_clicked(); // Funcion que se ejecuta al pulsar el boton "Ping"
36     void on_measurementSwitch_valueChanged(int value); // Funcion que se ejecuta al cambiar el valor del switch de iniciar medidas
37     void on_waterTimeSlider_valueChanged(double value); // Funcion que se ejecuta al cambiar el valor del slider de tiempo de riego
38     void on_waterButton_clicked(); // Funcion que se ejecuta al pulsar el boton "Water"
39     void on_waterMoistureSlider_valueChanged(double value); // Funcion que se ejecuta al cambiar el valor del slider de threshold de humedad
40     void on_waterTemperatureSlider_valueChanged(double value); // Funcion que se ejecuta al cambiar el valor del slider de threshold de temperatura
41     void on_waterParamsSet_clicked(); // Funcion que se ejecuta al pulsar el boton "Set Thresholds"
42     void on_freq_valueChanged(double value); // Funcion que se ejecuta al cambiar la frecuencia con la que se miden los sensores
43
44 private:
45     // Funciones privadas
46     void pingReceived(bool response = false); // Comprueba la respuesta del ping
47     void startClient(); // Funcion que se conecta con la direccion que introducimos
48     void processError(const QString &s); // Procesa los errores, en caso de que ocurran
49     void activateRunButton(); // Habilita el boton de "Connect" y deshabilita el resto
50     void SendMessage(QJsonDocument mensaje); // Envia un mensaje a la red MQTT
51     void saveData(double temperature, double humidity, double moisture); // Almacena los datos en un archivo .txt
52

```

Figura 8.5: Métodos de la clase GUIPanel

En la Fig.8.5 se pueden observar las declaraciones de los métodos de la clase. La clase hereda de QWidget y tiene como métodos públicos el constructor y el destructor. Como métodos privados tiene diferentes funciones que se activan al interactuar con la interfaz y funciones de ayuda. En los comentarios del código se explica de forma breve la utilidad de cada función.

Cabe destacar, la función *saveData()* la cual tiene como propósito almacenar en el disco duro del ordenador donde se esta ejecutando la interfaz, los datos recibidos en formato CSV (Comma Separated Value). El nombre del archivo es autogenerado usando la fecha y hora del momento en el que se crea, de forma que cada sesión esta asociada a un archivo diferente.



The screenshot shows a Windows Notepad window titled "09-14-19_04-53-24.txt ...". The menu bar includes File, Edit, Format, View, and Help. The main content area displays a series of comma-separated values representing sensor data. The data consists of three columns: temperature, humidity, and moisture. The values are floating-point numbers ranging from approximately 26.498055 to 57.261009. The Notepad window also features standard scroll bars and a status bar at the bottom indicating 100% zoom, Windows (CRLF) encoding, and UTF-8 encoding.

temperature	humidity	moisture
26.498055	56.427860	56.920952
26.527428	56.522472	56.588379
26.556801	56.481274	57.586098
26.540779	56.620129	64.544518
26.570152	56.682690	64.314278
26.556801	56.723888	56.358135
26.570152	56.682690	56.818623
26.570152	56.835278	56.383720
26.583504	56.879532	55.539497
26.583504	56.989395	56.204643
26.626236	56.844437	56.434883
26.626236	57.039749	56.255810
26.612885	57.013809	56.358135
26.626236	57.039749	56.102318
26.612885	57.036697	57.304688
26.626236	57.105362	56.741871
26.626236	57.135880	54.951099
26.626236	57.082474	57.688423
26.639587	57.282372	56.793037
26.639587	57.238121	63.342144
26.639587	57.261009	57.048862

Figura 8.6: Ejemplo de archivo almacenado

8.3. Arquitectura de red MQTT

El protocolo MQTT es un estándar de tipo publicación-suscripción que funciona en una capa superior a TCP/IP (Transmission Control Protocol/Internet Protocol) [26]. Es un protocolo ideal para el sistema, ya que hace que la transmisión de datos sea muy sencilla e inalámbrica. El *broker* principal usado en el proyecto se llama c [27] y es un *broker* MQTT de código libre desarrollado por Eclipse, se ha elegido dicho *broker* debido a su sencillez y ligereza.

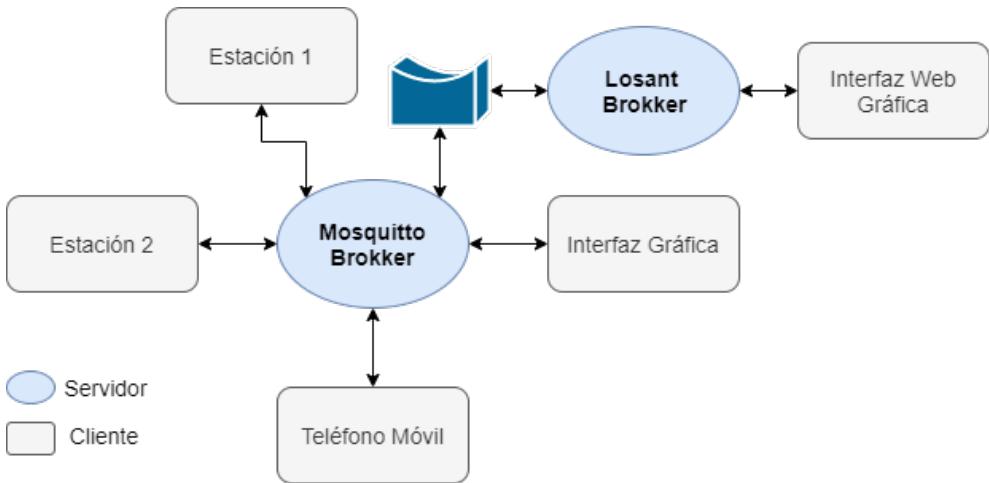


Figura 8.7: Ejemplo de un posible escenario con 2 estaciones y un teléfono móvil conectados

En la Fig.8.7 se puede observar un posible caso en el que hay conectados 4 clientes al servidor MQTT, en este caso hay 2 estaciones conectadas, la interfaz gráfica desde la que el usuario monitoriza y actúa y por ultimo hay un terminal móvil el cual se ha suscrito a determinados temas y también los recibirá.

También se ha configurado un puente desde el *broker* de Mosquitto, de forma que reenvíe todos los mensajes relacionados con los sensores que reciba al *broker* de Losant, la plataforma IoT que se ha usado. Dicho puente tiene en cuenta las limitaciones del *broker* de Losant (se debe autenticar con unos credenciales previamente generados, no permite retención de datos y la calidad de servicio debe ser 0) y permite que el sistema se pueda acceder de forma completamente remota, sin incluso estar conectado a la red Wi-Fi.

Es importante destacar, que se puede conectar cualquier dispositivo que este en la red Wi-Fi como cliente MQTT siempre que se subscriba a los temas deseados, como ejemplo, se ha usado la aplicación *MyMQTT* [28] para recibir los mensajes de la estación desarrollada :

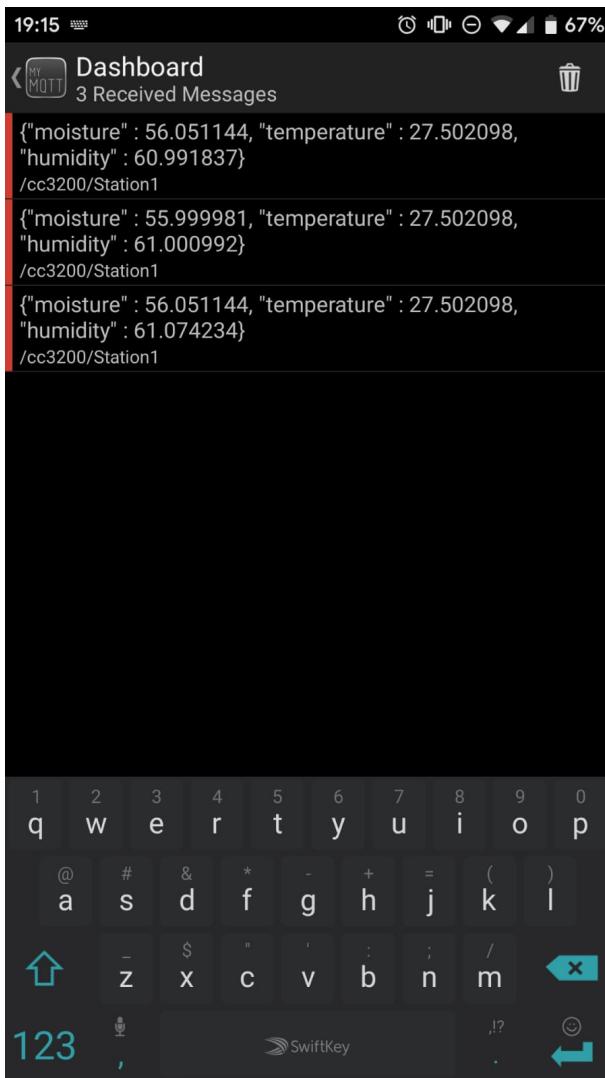


Figura 8.8: Tramas recibidas en la aplicación MyMQTT

A continuación se hará una explicación de los diferentes *topics* que se usan:

- **PC:** *Topic* en el que publica el ordenador, a través de la interfaz, y al que se subscribe la estación (o estaciones en caso de tener varias). A través de él se envían todos los cambios de configuración y todas las interacciones del usuario a través de la interfaz.

- **CC3200/Station1:** Sobre este tema publicará la estación del sistema, y enviará los datos recolectados de los sensores. La aplicación de PC se subscribe de forma que reciba la información.
- **CC3200/StationX:** En caso de tener más estaciones, solo tendríamos que añadir un *topic* con el número de la estación.

Es importante destacar, que la aplicación de PC se subscribe al *topic* *CC3200/#* de forma que podemos añadir estaciones sin tener que cambiar nada en la aplicación de PC. El carácter #, hará que reciba cualquier mensaje cuya *topic* comience por /CC3200/ [29].

Un ejemplo sencillo de comunicación del sistema con una sola estación, como es el caso desarrollado, y en el que la web de Losant no actúa podría ser:

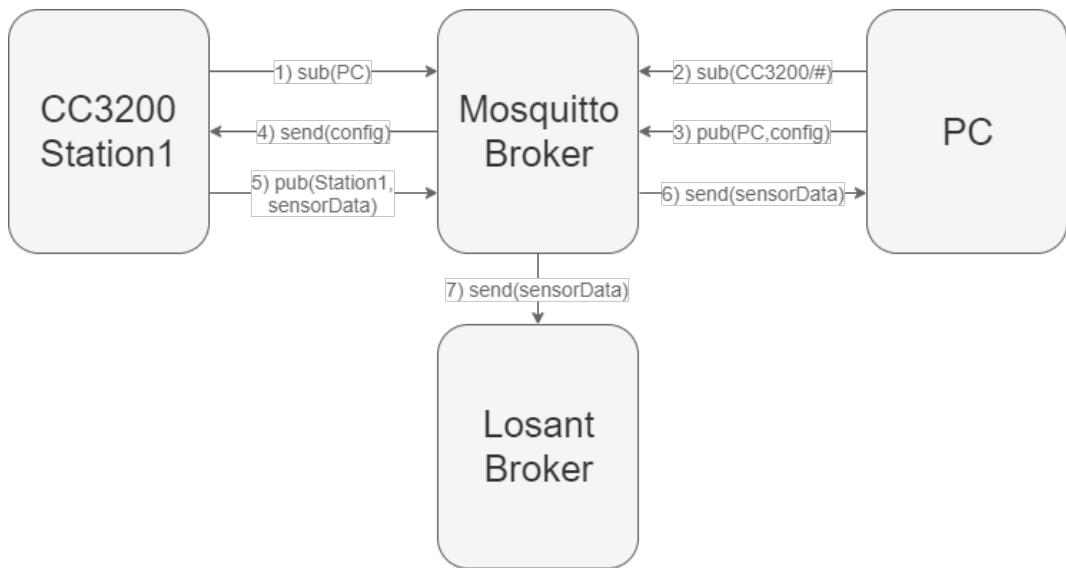


Figura 8.9: Ejemplo de comunicación MQTT del sistema

1. Al arrancar la aplicación del microcontrolador, esta se subscribe al *topic* *PC*

2. Cuando se inicia la interfaz, la aplicación de PC, se subscribe a todos los *subtopics* de */CC3200/* , mediante */CC3200/*
3. El usuario, debe conectar y activar las mediciones, fijando si desea la frecuencia de medida y otros parámetros, estos datos son publicados por la aplicación de ordenador en el *topic PC*.
4. El *broker* de mosquitto enviará los datos al microcontrolador, ya que se suscribió a dicho tema y los procesará.
5. El código estará realizando medidas periódicas, con la frecuencia recibida, y reportara estos datos publicándolos en el tema *Station1*.
6. El *broker* principal reenvía los datos a la aplicación de PC, donde son mostrados y almacenados.
7. Como se ha configurado un puente, el *broker* de mosquitto envía los datos al *broker* de Losant,

Como se ha comentado anteriormente, una de las grandes ventajas del sistema es su fácil escalabilidad, si deseásemos añadir más estaciones tan solo necesitaríamos replicar el hardware y subscribir las aplicaciones al *topic PC*. Por parte de Losant, se deberían replicar tanto el dispositivo como el *dashboard*. Un ejemplo simplificado de la conexión sería:

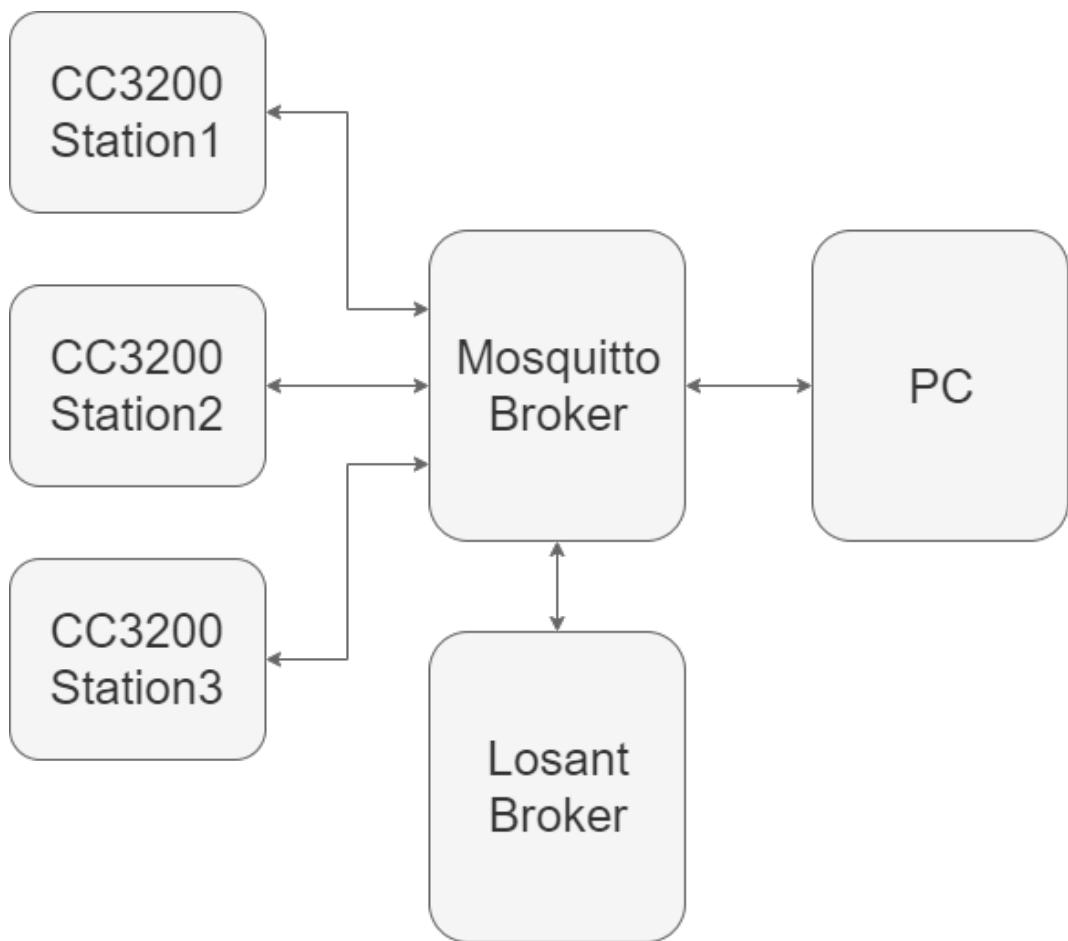


Figura 8.10: Arquitectura del sistema con varias estaciones

8.4. Losant, Plataforma IoT

Para sobrepasar la limitación de solo poder acceder a los datos e interactuar al sistema de forma local (estando conectado a la misma red Wi-Fi que el sistema) y facilitar la visualización de los datos se ha desarrollado un portal: tfm-sergiogasquez.onlosant.com [30] en un dominio facilitado por Losant, una plataforma que permite el almacenamiento, tratamiento y visualización de datos.

Como se ha comentado en la sección anterior, los datos llegan al *broker* de Losant, a través del puente configurado en el *broker* de mosquitto. En la plataforma, se han creado dos dispositivos:

- **Gateway:** Este dispositivo estará conectado directamente al *broker* de Losant y reencaminará los mensajes al otro dispositivo.
- **Station1:** Este dispositivo recibirá todos los mensajes, gracias a la *gateway*, y tiene los siguientes campos:

DEVICE ATTRIBUTES ⓘ		
Attributes define the fields and data types of the device's state. These are used to properly visualize and validate the various sensor or other data this device reports. For example a device with a temperature sensor might have an attribute with the data type "Number" and the name "temperature".		
Data Type	Name	
Number	moisture	–
Data Type	Name	
Number	temperature	–
Data Type	Name	
Number	humidity	–
Data Type	Name	
Number	waterLevel	–
Data Type	Name	
Number		

Figura 8.11: Atributos del dispositivo de Losant

De forma, que todas las tramas que reciba el dispositivo que tengan dichos atributos en formato JSON, serán almacenados por el dispositivo.

Para la visualización de datos, se han usado Dashboards de Losant, los cuales permiten mostrar de forma fácil y variada, las variables almacenadas en los diferentes dispositivos.

Antes de comentar el *dashboard*, se explicarán los tipos de bloques usados para mostrar los datos:

- **Time Series Graph:** Este bloque permite mostrar el valor de una variable a lo largo de un tiempo personalizable.
- **Gauge:** Permite mostrar el valor actual de una variable de forma numérica o visual.
- **Indicator:** Comprueba el valor de una variable y muestra un mensaje en función del valor de la variable.
- **Input Controls:** Permite al usuario fijar la configuración e interactuar con el sistema.

Por lo que el tablero final realizado para la estación desarrollada queda así:

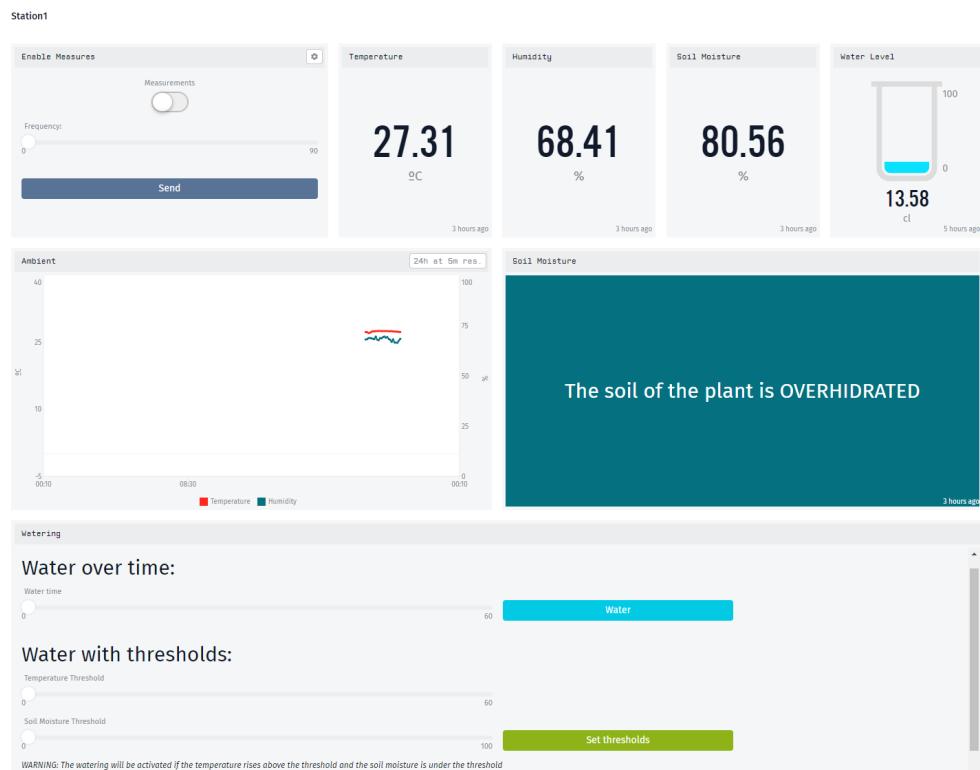


Figura 8.12: *Dashboard* desarrollado en Losant

En la Fig.8.12 se observa el tablón desde el que podemos interactuar con el sistema así como ver la progresión o valor actual de las diferentes variables.

Por ultimo, para que la visualización no se tuviese que hacer a través de la cuenta del desarrollador de Losant, se ha usado un dominio, *tfm-sergiogasquez.onlosant.com* [30], proporcionado por la propia plataforma para desarrollar una web. Dicho dominio necesitará de credenciales para acceder a el:

- **Email address:** test.user@example.com
- **Password:** 12345678

De forma, que solo los usuarios deseados puedan acceder a los datos y actuar en el sistema. Una vez autenticado el usuario, nos redirige a la pagina de bienvenida:

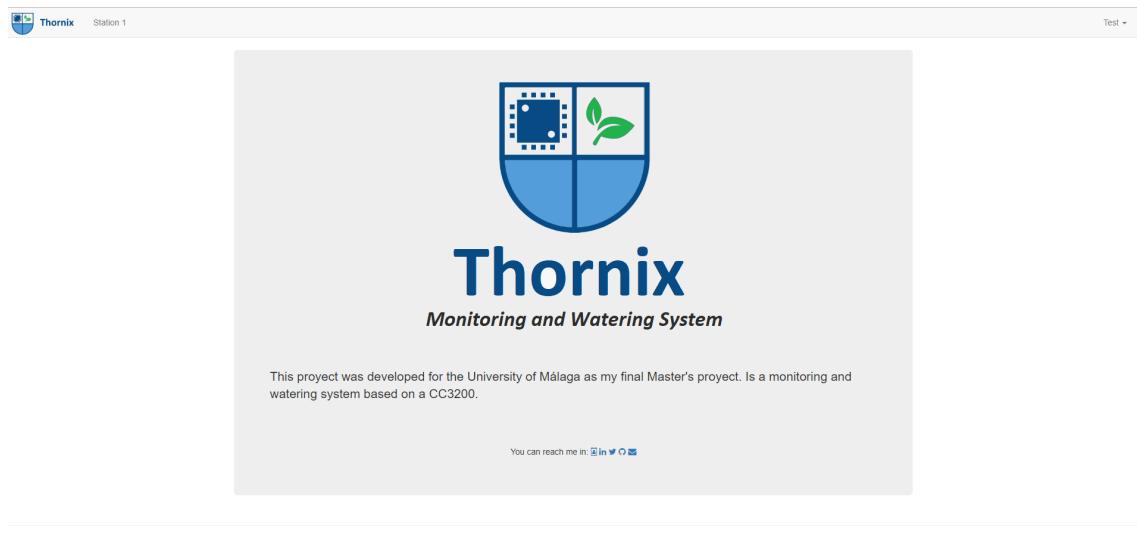


Figura 8.13: Pagina de bienvenida de la web desarrollada

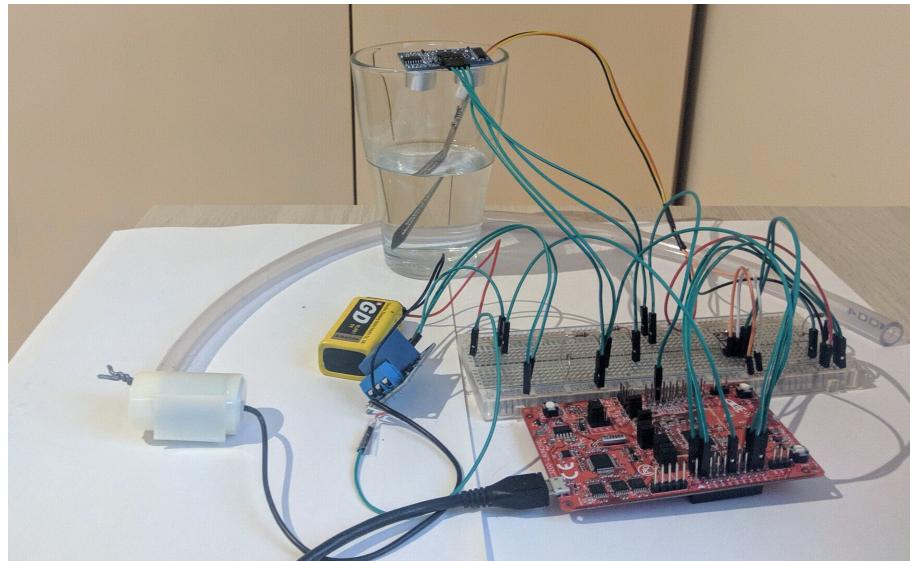
Desde esta vista de la web, se puede acceder al *dashboard* de la Fig8.12, pulsando en la pestaña *Station1*. En caso de que hubiese más

estaciones para el mismo usuario, estas serían añadidas a la web y presentadas con su propio panel.

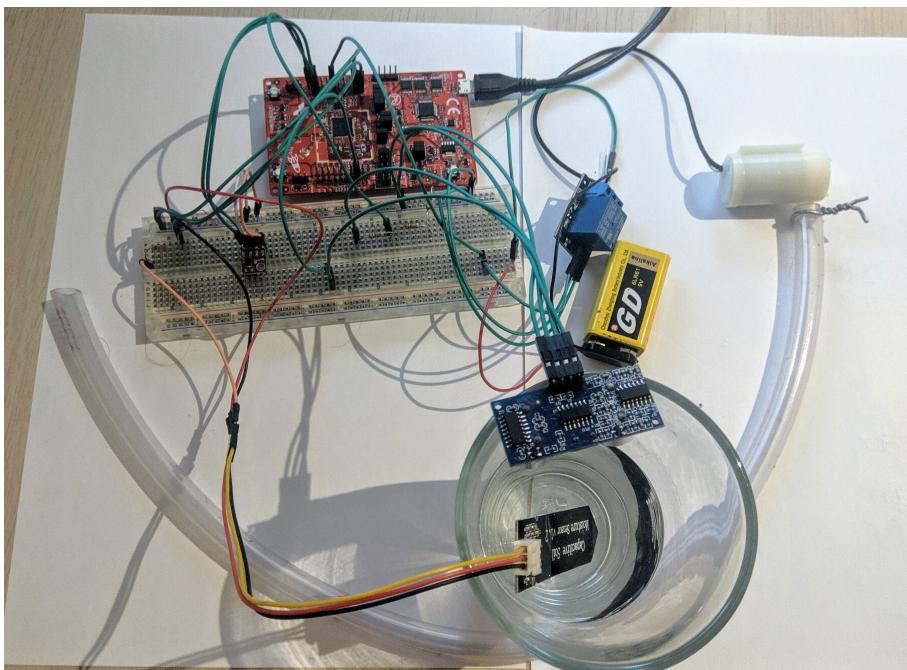
8.5. Prototipos desarrollados

8.5.1. Prototipo V0

En primer lugar, para desarrollar el sistema se uso una *protoboard* en la que se iban montando y conectando los diferentes módulos para introducirlos de forma escalonada y su fácil modificación.



(a) Vista lateral



(b) Vista superior

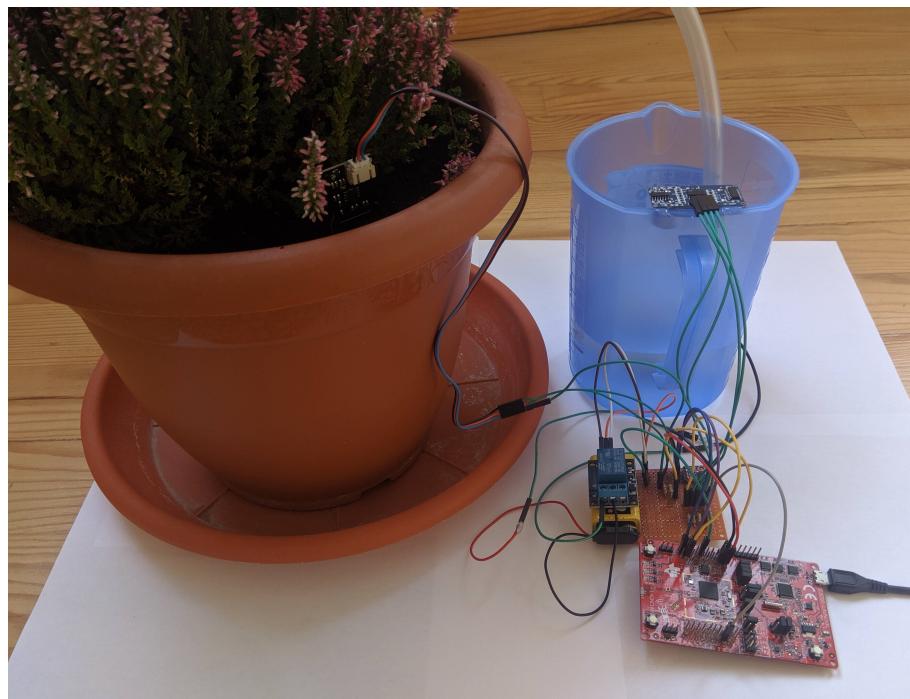
Figura 8.14: Prototipo V0

Este prototipo ha sido de gran utilidad debido a la facilidad que aporta-

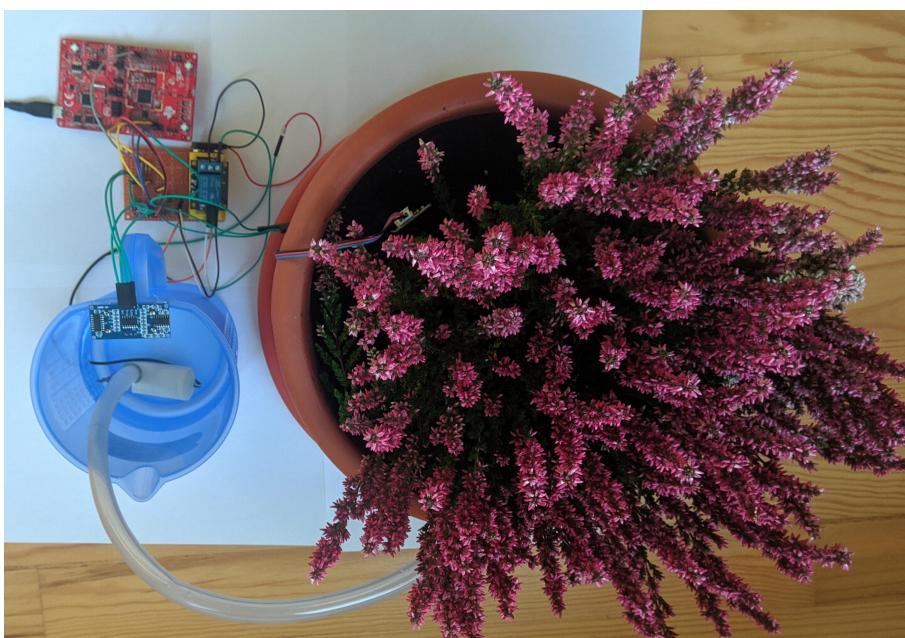
ba para realizar cambios, así como depurar errores de conexiones. Pero también se puede observar que la organización no era la más adecuada, los cables de elevada longitud resultaron, como era de esperar, muy poco fiables, algunos cables no realizaban bien las conexiones y algunos otros problemas destacables fueron encontrados.

8.5.2. Prototipo V1

Para mejorar la consistencia del sistema, se han desarrollado las conexiones en una placa de baquelita de forma que el sistema quedé muchos más compacto y eficiente.



(a) Vista lateral



(b) Vista superior

Figura 8.15: Prototipo V1

A pesar de que las conexiones se hayan realizado mediante soldadura, todos los componentes del sistema se pueden desconectar de manera sencilla, ya que se han usado cables y pines para ello, de forma que reemplazar cualquier sensor sea fácil. La mayoría de cables usados, podrían tener una longitud bastante menor, de forma

Capítulo 9

Plan de aceptación

En este capítulo se describirán las pruebas realizadas para la validación del proyecto, así como se discutirán los resultados obtenidos. De forma que se verificará el trabajo realizado.

9.1. Pruebas

Pgr1 - Lectura de sensores	
<i>Descripción:</i>	Se deberán realizar una cantidad razonable de lecturas con diferentes frecuencias de medida para ver que los se mide correctamente. Sería recomendable comparar el resultado de los sensores con el valor medido con otro instrumento que obtenga la misma variable.
<i>Prerrequisitos:</i>	Sistema instalado y funcional.
<i>Pasos:</i>	1 – Se debe poner en marcha la aplicación 2 – Se deben ver las lecturas desde los <i>logs</i> creados por la comunicación PC-Microcontrolador, para evitar introducir errores que podemos causar al mostrarlo en la interfaz.
<i>Resultado esperado:</i>	Los valores obtenidos deben ser coherentes y en medida de lo posible contrastados con el valor obtenido medido con otro instrumento.
<i>Resultado obtenido:</i>	Los valores obtenidos han sido verificados diferentes fuentes (la agencia estatal de meteorología para el caso del sensor de ambiente y una regla para el sensor de distancia).

Tabla 9.1: Prueba de lectura de sensores

9.1.1. Mostrar valores

Pgr2 - Mostrar valores	
<i>Descripción:</i>	Una vez recibido el valor del sensor en la aplicación de PC, se debe comprobar que este valor se muestra de la forma en la que deseamos y que no se altera ni deforma.
<i>Prerrequisitos:</i>	Ambas aplicaciones funcionando
<i>Pasos:</i>	1 – Se debe poner en marcha la aplicación y la interfaz 2 – Se deben ver las lecturas en la interfaz visual
<i>Resultado esperado:</i>	Los valores obtenidos se representan correctamente en la interfaz.
<i>Resultado obtenido:</i>	Los resultados se representan correctamente tanto en la interfaz general, así como en las gráficas.

Tabla 9.2: Prueba de mostrar valores en la interfaz

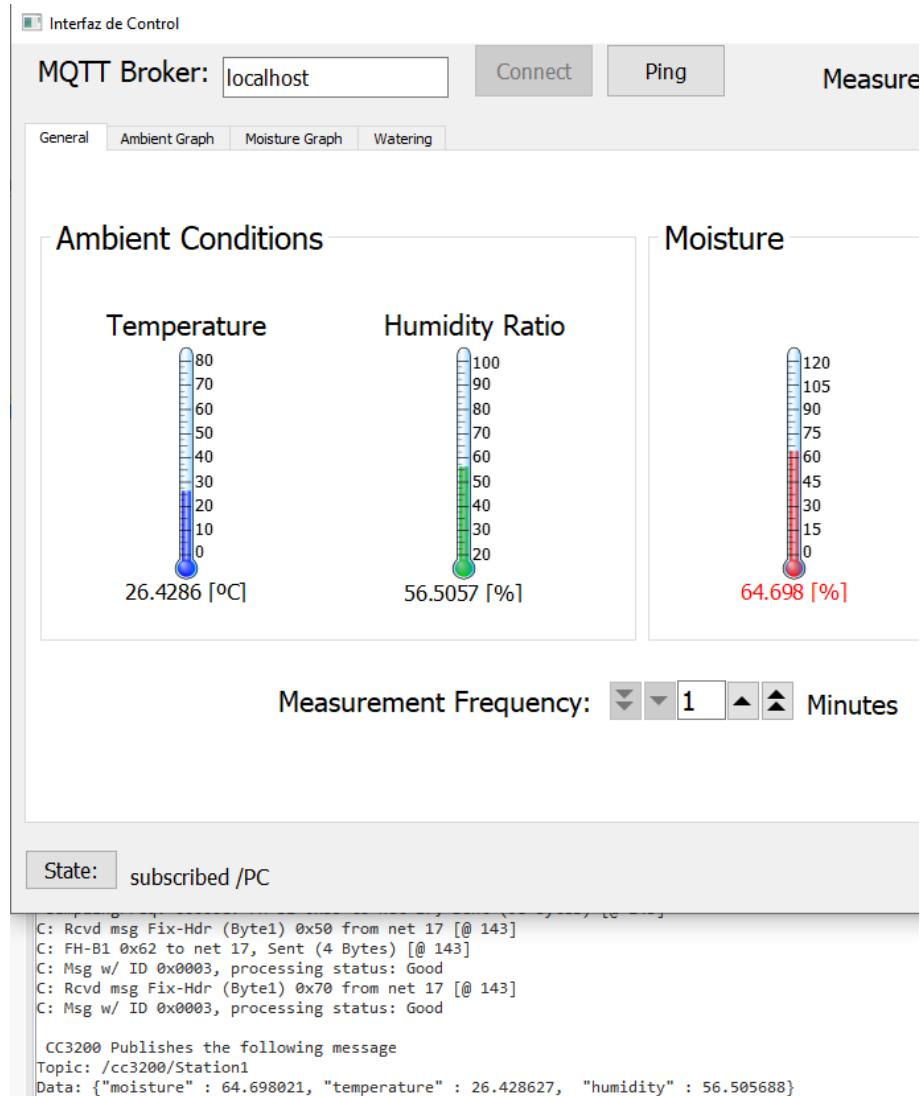


Figura 9.1: Representación en la interfaz de los datos enviados

En la Fig.9.1 se puede observar como se representan en la interfaz visual los datos enviados por el microcontrolador (se puede observar los datos en la ventana secundaria, cuya ultima linea contiene dichos datos)

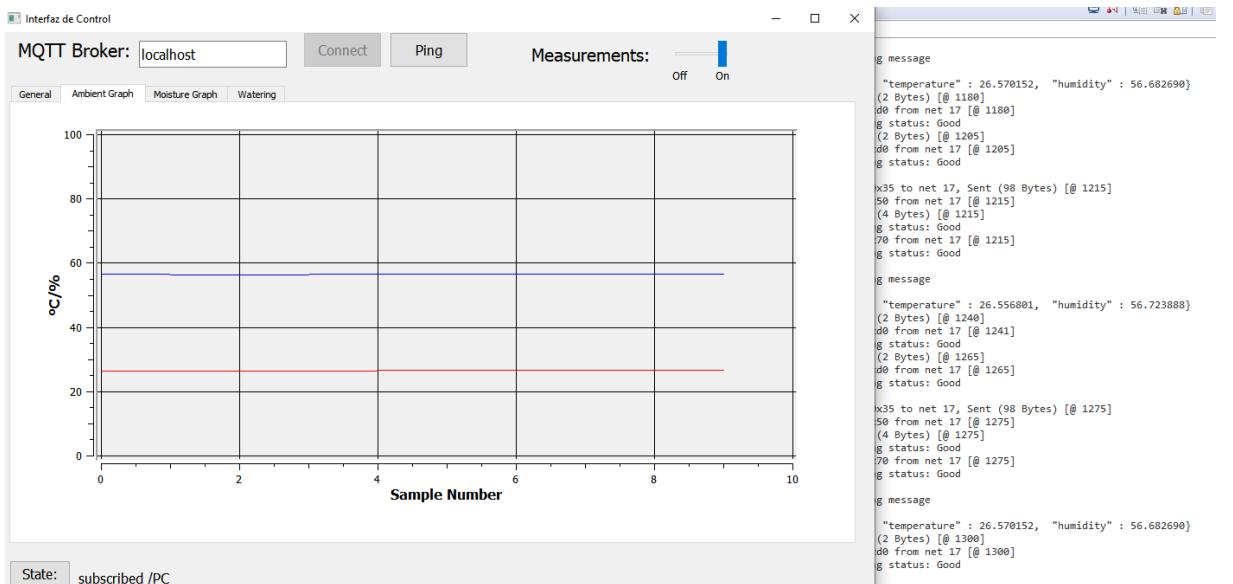


Figura 9.2: Representación en las gráficas de los datos enviados

La Fig.9.2 muestra la representación de 10 muestras (correspondiente con 10 minutos) de la temperatura y la humedad relativa en el ambiente. La linea roja representa la temperatura, mientras que la azul representa la humedad relativa. A la derecha se puede observar la consola del microcontrolador con los datos representados de forma numérica por puerto serie de las ultimas 3 medidas. No se observa mucha diferencia entre las medidas, ya que se han realizado en el interior de una casa y durante un periodo corto.

9.1.2. Regar

Pgr3 - Regar	
<i>Descripción:</i>	A través de la interfaz, el usuario podrá mandar el comando de riego y este debe ejecutarse
<i>Prerrequisitos:</i>	Sistema completamente funcional.
<i>Pasos:</i>	1 – Se debe activar el riego desde la interfaz 2 – El sistema debe regar la planta
<i>Resultado esperado:</i>	La planta recibe agua.
<i>Resultado obtenido:</i>	La planta se riega correctamente.

Tabla 9.3: Prueba de riego

Se han realizado diferentes pruebas de funcionalidad para verificar que la planta se regaría correctamente, cuando se manda el comando de regar durante un tiempo vemos, que la bomba comienza a regar y esto se puede comprobar en el siguiente vídeo [31]. También se puede observar por consola que se ejecuta la tarea que deseamos:

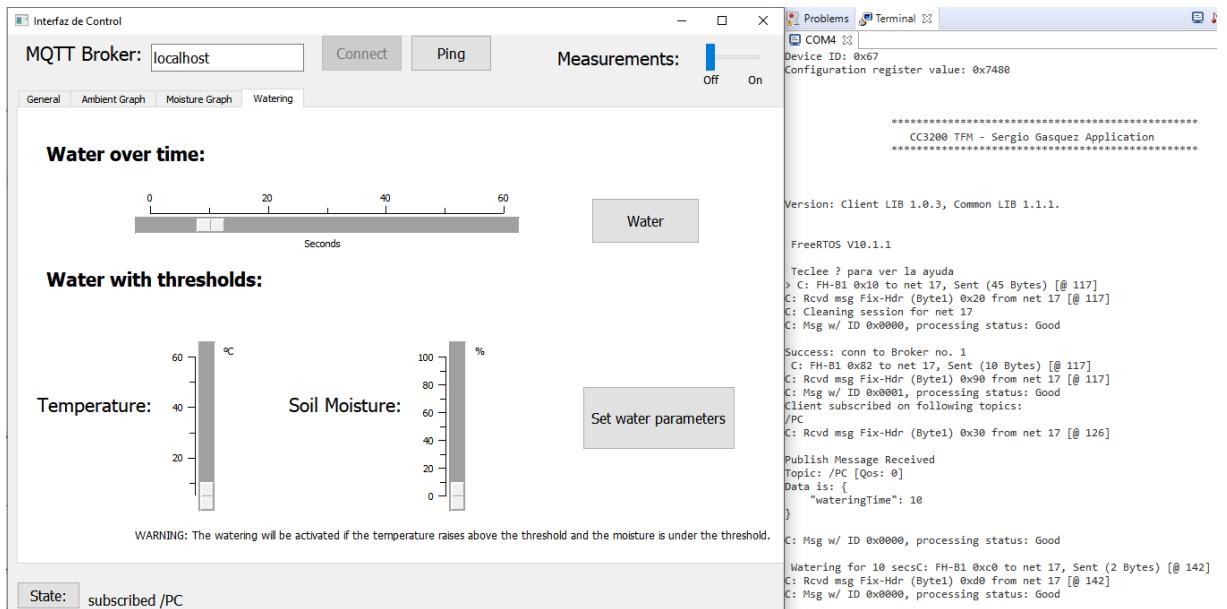


Figura 9.3: Comprobación del riego

9.1.3. Comprobar umbrales

Pgr4 - Comprobar umbrales	
<i>Descripción:</i>	A través de la interfaz, el usuario podrá fijar umbrales, los cuales si son sobrepasados el riego debe activarse.
<i>Prerrequisitos:</i>	Sistema completamente funcional.
<i>Pasos:</i>	1 – A través de la interfaz, se fijará los umbrales 2 – En caso de que el umbral se sobreponga, el sistema debe activar el riego
<i>Resultado esperado:</i>	La planta se riega en caso de sobreponerse el umbral.
<i>Resultado obtenido:</i>	Se han fijado unos umbrales que activen la bomba a propósito y esta se ha activado.

Tabla 9.4: Prueba de comprobar umbrales

De nuevo, se ha comprobado, en prototipo, que cuando se fijan los umbrales la planta es regada [32] y esto se puede observar en el siguiente vídeo, cuando los umbrales se sobrepasan, el relé activa la bomba, posteriormente se cambian los umbrales y en la siguiente medición la bomba no se activa ya que los umbrales se han cambiado entre las dos medidas. También se ha podido observar por consola:

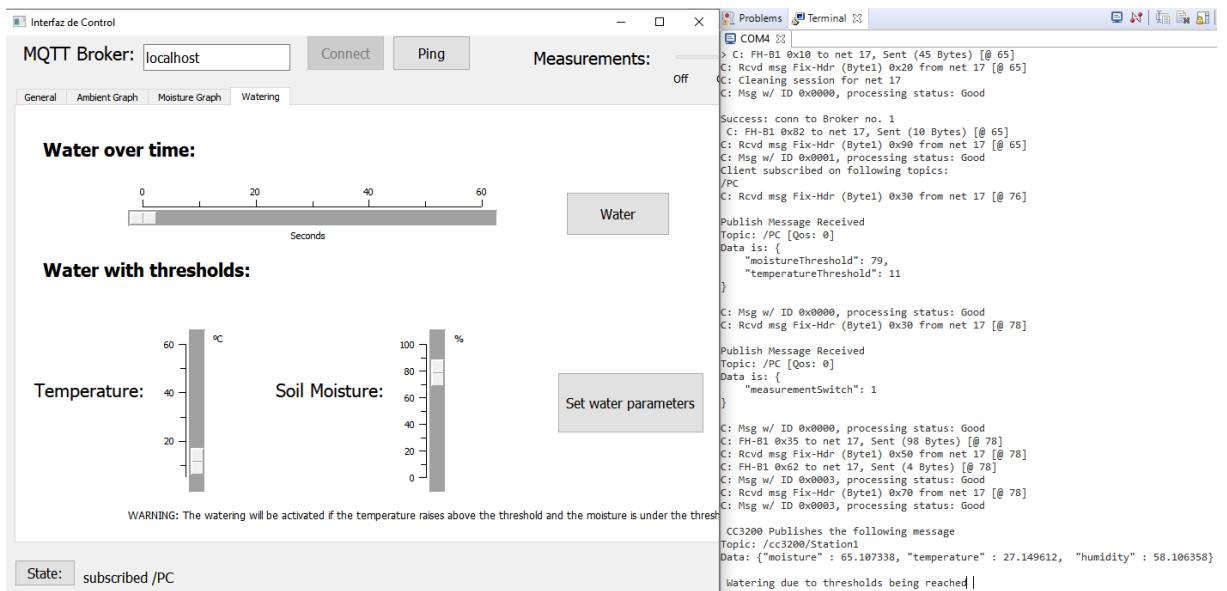


Figura 9.4: Comprobación de riego fijando umbrales

9.1.4. Comprobar histórico

Pgr5 - Comprobar histórico	
<i>Descripción:</i>	Tras un tiempo midiendo, se puede observar el histórico que ha guardado para observar que todo lo almacenado corresponde con lo medido y mostrado
<i>Prerrequisitos:</i>	Sistema completamente funcional.
<i>Pasos:</i>	1 – El sistema se debe dejar midiendo, mientras se van tomando nota de los valores que se van recibiendo de alguna otra forma alternativa 2 – Tras un tiempo razonable, se deben comprobar que las medidas coinciden
<i>Resultado esperado:</i>	Las medidas coinciden.
<i>Resultado obtenido:</i>	Se ha comprobado, tanto de forma visual como observando las gráficas que los datos coinciden.

Tabla 9.5: Prueba de comprobar el histórico

The screenshot shows a terminal window with two panes. The left pane displays a text file named '09-14-19_04-53-24.txt' containing a series of temperature, humidity, and moisture readings. The right pane shows a serial port named 'COM4' with messages related to CC3200 publishing data. The data in the file and the serial port output are identical, confirming the correctness of the historical data.

```

09-14-19_04-53-24.txt ...
File Edit Format View Help
temperature, humidity, moisture
[26.498055, 56.427860, 56.920952]
[26.527428, 56.522472, 56.588379]
[26.556801, 56.481274, 57.586098]
[26.540779, 56.620129, 64.544518]
[26.570152, 56.682690, 64.314278]
[26.556801, 56.723888, 56.358135]
[26.570152, 56.682690, 56.818623]
[26.570152, 56.835278, 56.383720]
[26.583504, 56.879532, 55.539497]
[26.583504, 56.989395, 56.204643]
[26.626236, 56.844437, 56.434883]
[26.626236, 57.039749, 56.255810]
[26.612885, 57.013809, 56.358135]
[26.626236, 57.039749, 56.102318]
[26.612885, 57.036697, 57.304688]
[26.626236, 57.105362, 56.741871]
[26.626236, 57.135880, 54.951099]
[26.626236, 57.082474, 57.688423]
[26.639587, 57.282372, 56.793037]
[26.639587, 57.238121, 63.342144]
[26.639587, 57.261009, 57.048862]
[26.682312, 57.320518, 57.560513]
[26.668961, 57.286949, 56.486050]
[26.668961, 57.373924, 56.434883]
[26.682312, 57.320518, 57.279106]
[26.668961, 57.503624, 57.151196]
[26.652939, 57.434959, 55.948818]
[26.682312, 57.506676, 57.714005]
[26.695663, 57.563133, 56.102318]
[26.695663, 57.694363, 55.411583]
[26.682312, 57.506676, 53.211494]
[26.682312, 57.639431, 55.999981]

CC3200 Publishes the following message
Topic: /cc3200/Station1
Data: {"moisture" : 55.411583, "temperature" : 26.695663, "humidity" : 57.694363}
C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [@ 2656]
C: Msg w/ ID 0x0047, processing status: Good

C: FH-B1 0xc0 to net 17, Sent (2 Bytes) [@ 2681]
C: Rcvd msg Fix-Hdr (Byte1) 0xd0 from net 17 [@ 2681]
C: Msg w/ ID 0x0000, processing status: Good
C: FH-B1 0xc0 to net 17, Sent (2 Bytes) [@ 2706]
C: Rcvd msg Fix-Hdr (Byte1) 0xd0 from net 17 [@ 2707]
C: Msg w/ ID 0x0000, processing status: Good

samplingFreq: 60000C: FH-B1 0x35 to net 17, Sent (98 Bytes) [@ 2716]
C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [@ 2716]
C: FH-B1 0x62 to net 17, Sent (4 Bytes) [@ 2716]
C: Msg w/ ID 0x0049, processing status: Good
C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [@ 2716]
C: Msg w/ ID 0x0049, processing status: Good

CC3200 Publishes the following message
Topic: /cc3200/Station1
Data: {"moisture" : 53.211494, "temperature" : 26.682312, "humidity" : 57.506676}
C: FH-B1 0xc0 to net 17, Sent (2 Bytes) [@ 2741]
C: Rcvd msg Fix-Hdr (Byte1) 0xd0 from net 17 [@ 2742]
C: Msg w/ ID 0x0000, processing status: Good
C: FH-B1 0xc0 to net 17, Sent (2 Bytes) [@ 2766]
C: Rcvd msg Fix-Hdr (Byte1) 0xd0 from net 17 [@ 2766]
C: Msg w/ ID 0x0000, processing status: Good

samplingFreq: 60000C: FH-B1 0x35 to net 17, Sent (98 Bytes) [@ 2776]
C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [@ 2776]
C: FH-B1 0x62 to net 17, Sent (4 Bytes) [@ 2776]
C: Msg w/ ID 0x004b, processing status: Good
C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [@ 2776]
C: Msg w/ ID 0x004b, processing status: Good

CC3200 Publishes the following message
Topic: /cc3200/Station1
Data: {"moisture" : 55.999981, "temperature" : 26.682312, "humidity" : 57.639431}

```

Figura 9.5: Comprobación de los datos guardados en el histórico

En la Fig.9.5 se observa en el archivo de texto con el histórico de los últimos datos en el que se puede comprobar como coinciden los últimos 3 datos con los mostrados en la consola del microcontrolador.

Capítulo 10

Conclusiones

10.1. Conclusiones

La realización del proyecto ha supuesto, sin duda, un gran reto debido a su amplitud, que abarca desde la selección de sensores hasta el diseño de una interfaz, por lo que se han tenido que superar grandes retos.

Cabe destacar que los objetivos fundamentales se han cumplido:

- Crear la interfaz de los diferentes sensores de forma que se pueda realizar medidas con ellos.
- Crear una interfaz en la que se puedan mostrar los datos y actuar sobre el riego.
- Capacidad de regar la planta.
- Crear una arquitectura de red estable y escalable.

Además de los objetivos principales, también se han introducido algunas de las mejoras planteadas:

- Se ha introducido un sensor para poder obtener el nivel del tanque de agua.
- La interfaz permite al usuario fijar unos umbrales (de temperatura y humedad de suelo) a los que se riega si se alcanzan.
- Procesar y almacenar los datos en el ordenador.

Por lo que el trabajo final es un trabajo con gran atractivo debido a su bajo coste y su versatilidad que permite ser usado tanto para uso doméstico como para gran escala, debido a su facilidad de escalado. Otra de sus cualidades que destaca en su comunicación inalámbrica la cual da libertad a la hora de posicionar la planta.

El hecho de tener los datos almacenados en un fichero, permite que posteriormente se puedan analizar para obtener más información sobre la planta e incluso, podrían servir para predecir futuros comportamientos.

La interfaz es una manera fácil y eficaz para interactuar con el sistema y que permite ver las medidas, tanto actuales, como un histórico. Es posible desde ella regar la planta o fijar umbrales así como también pausar o reanudar las medidas.

La realización de las diferentes interfaces con los sensores, ha sido sin duda el mayor reto del proyecto ya que había muy poca documentación para alguno de los sensores y integrarlos en la placa CC3200, no fue tarea fácil. Se disponía de herramientas muy limitadas (solo se tenía acceso a un multímetro) por lo cual las tareas de depuración eran bastante complejas.

10.2. Trabajo futuro

Se ha de tener en cuenta que uno de las principales restricciones para desarrollar el proyecto era su bajo presupuesto, lo cual a su vez supone un

atractivo para el proyecto, pero los sensores son baratos y su rendimiento a lo largo del tiempo es desconocido. En casos en los que se use el sistema en exterior, bajo condiciones muy variables, los sensores podrían degradarse al estar expuestos a cambios climáticos.

En cuanto a líneas de trabajo futuro, existen varias posibilidades que le darían al proyecto más atractivo:

- **Conseguir que el sistema sea autónomo** de forma que se pueda alimentar por baterías. Esto supondría que hubiese mayor libertad para usar el sistema en lugares más remotos. El principal reto de esta mejora consiste en cargarle el código al microcontrolador, el cual solo tiene memoria RAM. La acción de regar consume bastante energía por lo que unos paneles solares podrían ayudar mucho a la batería que alimentase al sistema, en caso de que estuviese en un lugar con luz solar directa.
- **Precisión de los sensores**, principalmente del sensor de humedad en suelo. La precisión del sistema no es de máxima relevancia, pero si es un sector en el que se podría mejorar mucho. Tan solo mejorando la adaptación comentada en la Fig.7.4 del SEN0193 mediante el uso de amplificadores operacionales, doblaríamos su resolución.
- **Introducir un sensor de luminosidad** que permita obtener otro dato interesante, para ver como afecta la luminosidad a la planta, así como poder prevenir la falta o el exceso de luz solar sobre ella.
- **Seguir trabajando con la plataforma Losant**, la cual da un inmenso abanico de opciones que podrían ser de gran utilidad al proyecto, cabe destacar la implementación de alertas vía e-mail o SMS.
- **Mejorar el prototipo** desarrollando una PCB (*Printed Circuit Boards*).

Capítulo 11

Apéndice

11.1. Presupuesto

Artículo	Cantidad	Precio/unidad [€]
CC3200-LAUNCHXL	1	26.88
SEN0193	1	5.29
SHT31	1	12.50
HC-SR04	1	3.54
DC 5V Relé	1	1.08
DC 5V Bomba	1	2.69
9V Batería	1	3
Conektor de batería	1	1.30
Resistencias	5	0.05
Cables	25	0.02
Placa de baquelita	1	0.8
Mano de obra	200 horas	15€/hora
Total		3057.83 €

Tabla 11.1: Presupuesto del proyecto

Bibliografía

- [1] *Texas Instruments SimpleLink Wi-Fi CC3200 LaunchPad.* <http://www.ti.com/tool/CC3200-LAUNCHXL>. Acceso: 2019-20-06.
- [2] Urs Hunkeler, Hong Linh Truong y Andy Stanford-Clark. “MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks”. En: *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*. IEEE. 2008, págs. 791-798.
- [3] *Losant.* <https://www.losant.com/>. Acceso: 2019-22-09.
- [4] Texas Instruments Incorporated. *Code Composer Studio: Getting Started Guide*. Texas Instruments Incorporated, 2001.
- [5] Ray Rischpater. *Application development with Qt creator*. Packt Publishing Ltd, 2014.
- [6] *GitHub.* <https://github.com>. Acceso: 2019-21-09.
- [7] *Gantt.com.* <https://www.gantt.com>. Acceso: 2019-11-09.
- [8] *Qampo.* <https://qampo.es>. Acceso: 2019-21-06.
- [9] *aGrae.* <https://www.agrae.es>. Acceso: 2019-21-06.
- [10] *ExtremeTech How USB Charging Works, or How to Avoid Blowing Up Your Smartphone.* <https://www.extremetech.com/computing/115251-how-usb-charging-works-or-how-to-avoid-blowing-up-your-smartphone>. Acceso: 2019-21-06.

- [11] NXP Semiconductors. “I2C-bus specification and user manual”. En: *Rev 3* (2007), pág. 19.
- [12] Frédéric Leens. “An introduction to I 2 C and SPI protocols”. En: *IEEE Instrumentation & Measurement Magazine* 12.1 (2009), págs. 8-13.
- [13] *DfRobot SEN0193*. https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193. Acceso: 2019-22-07.
- [14] *Texas Instrument Wiki CC3200 ADC Appnote*. http://processors.wiki.ti.com/index.php/CC3200_ADC_Appnote. Acceso: 2019-01-07.
- [15] *SENSIRION Datasheet SHT3x-DIS*. http://www.mouser.com/ds/2/682/Sensirion_Humidity_Sensors_SHT3x_Datasheet_digital-971521.pdf. Acceso: 2019-01-07.
- [16] Manpreet Kaur y Jai Pal. “Distance measurement of object by ultrasonic sensor HC-SR04”. En: *Int'l Journal for Scientific Research & Development* 3.05 (2015).
- [17] *ElecFreaks Ultrasonic Ranging Module HC - SR04*. <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>. Acceso: 2019-01-07.
- [18] *Raspberrypi Forums Ultra cheap ultrasonics, and a bit of a challenge*. <https://www.raspberrypi.org/forums/viewtopic.php?p=183386#p183386>. Acceso: 2019-01-07.
- [19] Brian W Kernighan y Dennis M Ritchie. *The C programming language*. 2006.
- [20] Richard Barry y col. “FreeRTOS”. En: *Internet, Oct* (2008).
- [21] Martin S Michael. *Universal asynchronous receiver/transmitter*. US Patent 5,140,679. Ago. de 1992.
- [22] *FreeRTOS xQueueOverwrite*. <https://www.freertos.org/xQueueOverwrite.html>. Acceso: 2019-22-08.
- [23] *FreeRTOS xQueuePeek*. <https://www.freertos.org/xQueuePeek.html>. Acceso: 2019-22-08.

- [24] *CardinalPeak Understanding the Cyclic Redundancy Check.* <https://cardinalpeak.com/blog/understanding-the-cyclic-redundancy-check/>. Acceso: 2019-22-08.
- [25] Bjarne Stroustrup. *A Tour of C++*. Addison-Wesley Professional, 2018.
- [26] W Richard Stevens. *TCP/IP illustrated vol. I: the protocols*. Pearson Education India, 1994.
- [27] Roger A Light y col. “Mosquitto: server and client implementation of the MQTT protocol.” En: *J. Open Source Software* 2.13 (2017), pág. 265.
- [28] *Google PlayStore MyMQTT*. <https://play.google.com/store/apps/details?id=at.tripwire.mqtt.client&hl=en>. Acceso: 2019-22-08.
- [29] *HiveMq MQTT Topics Best Practices - MQTT Essentials: Part 5*. <https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/>. Acceso: 2019-22-08.
- [30] *Thornix Losant*. tfm-sergiogasquez.onlosant.com. Acceso: 2019-23-09.
- [31] *YouTube TFM - Prueba Riego*. <https://www.youtube.com/watch?v=QUx5NQMJMQg&feature=youtu.be>. Acceso: 2019-22-08.
- [32] *YouTube TFM - Prueba umbrales riego*. <https://www.youtube.com/watch?v=eHevX1F8j-U&feature=youtu.be>. Acceso: 2019-22-08.