UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA POLITECNICA SUPERIOR DE GANDIA

GRADO EN ING. SIST. DE TELECOM., SONIDO E IMAGEN

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

ESCOLA POLITÈCNICA
SUPERIOR DE GANDIA

"Sistema de control de parámetros de cultivo mediante tecnología inalámbrica de bajo consumo"

TRABAJO FINAL DE GRADO

Autor/a:
David Hostalrich Jato

Tutor/a:
José Pelegrí Sebastiá
Tomás Sogorb Devesa

GANDIA, 2019

Resumen

El proyecto se basa en el desarrollo de un sistema hidropónico, haciendo uso de una placa Arduino para la sensorización de los parámetros necesarios, y una Orange Pi Zero como unidad central de procesamiento y almacenamiento

La información obtenida por los sensores se encapsulará y se enviará a la unidad central mediante Bluetooth, por lo que será necesario establecer un protocolo de comunicación entre ambos sistemas.

Una vez la unidad central reciba la información, la procesará y la almacenará en una base de datos.

Palabras clave: Sistema de medida, temperatura, inalámbrico, base datos, bluetooth.

Abstract

The project is based on an hydroponic system development using an Arduino board for the sensorization of the necessary parameters, and an Orange Pi Zero as a central processing and storage unit.

The data obtained from the sensors will be encapsulated in a single data package that will be sent to the central unit via Bluetooth, so we will need to establish a communication protocol between both systems.

The received data will be processed and stored in a database.

Keywords: Measurement system, temperature, wireless, database, bluetooth.

Índice

Contenido

Figuras

1. Introducción

Hoy en día disponemos de una gran facilidad para utilizar la tecnología en nuestro día a día. Desarrollar e implementar sistemas utilizando dichas tecnologías es relativamente sencillo incluso para las personas que no están muy adentradas en el mundo de electrónica y la informática. Además, en los últimos años, esto se ha visto acentuado por la revolución del internet de las cosas (IoT - Internet of Things).

En este caso, se ha pensado en como esta tecnología se podría implementar en los cultivos, un sector que ha sido clave en el pasado de este país y que ahora está en declive.

A la hora de controlar un cultivo, debemos medir y almacenar todos los datos importantes del entorno, con el objetivo de llevar un control sobre este, mejorar su mantenimiento y producción.

Por lo tanto, se ha planteado diseñar y programar un sistema de sensorización que monitorice una plantación y que envíe los datos obtenidos a una unidad central mediante bluetooth, donde se procesarán y almacenarán en una base de datos.

De esta forma, solo con acceder a la unidad central, podremos saber el estado de la plantación sin necesidad de tener un sistema fijo o aparatoso, además de hacer de este diseño un sistema escalable y adaptable.

Motivación

El mundo de la electrónica y de la programación es apasionante y con un gran potencial. Este conocimiento permite a las personas desarrollar y llevar a la práctica sus ideas más creativas en cualquier ámbito.

En este caso, la razón para elegir este proyecto en concreto ha sido la de poder aportar algo que ayude o facilite el trabajo a los demás con la ayuda de las nuevas tecnologías.

Se ha querido utilizar el bluetooth como método de conexión debido a su bajo consumo y a que actualmente esta tecnología está logrando grandes avances, permitiendo con su última versión, la 5.0, conseguir una gran mejora en alcance, velocidad de transmisión y consumo. Por ello, hemos pensado que aprender a usar esta tecnología será beneficioso en el futuro.

Objetivos

El propósito de este trabajo es el de diseñar y programar un sistema de control de cultivo de bajo consumo, eficiente, fácil de implementar, escalable y basado en la comunicación bluetooth.

Por lo tanto, el objetivo principal será desarrollar un sistema basado en Arduino con los sensores para medir todos los parámetros necesarios. Una vez alcanzado dicho objetivo, pasaremos a los siguientes dos objetivos. El primero será diseñar y programar la unidad central, la cual utilizará una placa Orange Pi Zero, la cual recibirá los datos y los procesará para almacenarlos en la base de datos interna. El segundo objetivo será el de establecer el protocolo de comunicación bluetooth que se usará para comunicar el sistema de sensorización con la unidad central.

Estructura

La estructura que se va a seguir es la siguiente:

I. Estado del arte. En este apartado se analiza las aplicaciones actuales de la tecnología de la que disponemos en el sector del cultivo y las distintas opciones que existen.
II. Análisis del problema. Se estudiará adecuadamente el problema a resolver, dividiéndolo en distintos objetivos más simples con tal de poder trabajar de una forma más sistemática y organizada.
III. Diseño del conjunto de sensores controlado por Arduino. Se tratará con más profundidad como se ha diseñado el apartado de toma de datos de los sensores.
IV. Diseño de la unidad central. Se explica la función y el diseño que se ha tomado para el funcionamiento de dicha unidad.
V. Conexión y comunicación mediante bluetooth. Se explica cómo se ha configurado los dispositivos de comunicación y como se envía la información a través de estos.
VI. Puesta en funcionamiento. Una vez configurado el sistema, se pasa a realizar pruebas experimentales para ver su comportamiento.
VII. Consumo energético. Se hará un estudio de la energía consumida por el sistema, valorando el efecto de los métodos utilizados para conseguir un mayor ahorro.
VIII. Conclusión. Una vez tratado todo el problema y su solución, se repasará la utilidad del proyecto y su impacto real, así como valorar posibles mejoras.

2. Estado del arte

El uso de la sensorización y control mediante sistemas electrónicos no es algo nuevo en el mundo de la agricultura. Con el progreso en el desarrollo de microcontroladores y la eficiencia en el consumo de estos, las aplicaciones y la sencillez para utilizar dichos sistemas ha ido creciendo.

Grandes empresas usan sistemas de control mediante sensores y microcontroladores en la agricultura desde hace bastante tiempo, pero con la llegada del IoT la posibilidad de poder crear un cultivo inteligente se ha vuelto más accesible, ya que el coste que supone adquirir este tipo de sensores y microcontroladores es muy bajo, además de la facilidad para implementarlos.

Actualmente, cualquier persona puede comprar una placa Arduino, los sensores necesarios y montar un pequeño sistema para controlar el cultivo de alguna planta que tenga en casa.

A esto cabe añadir los avances comentados anteriormente en la tecnología bluetooth 5.0[1]. Como es de esperar, ha sido diseñado teniendo muy en cuenta su aplicación en tecnologías IoT, ya que cuenta con el modo de bajo consumo (BLE – Bluetooth Low Energy), así como un aumento del doble de velocidad y hasta cuatro veces más alcance respecto a su versión anterior. Además, incorpora la reciente tecnología Mesh (redes de malla). Estas redes se basan en la funcionalidad de bajo consumo de energía y permite conectar los módulos bluetooth vecinos entre sí formando una red. De esta forma se puede enviar información de un punto a otro haciendo uso de los módulos bluetooth intermedios, que actúan como puente, consiguiendo expandir mucho el alcance del bluetooth manteniendo un nivel de consumo reducido. El problema que presenta esta tecnología es que aún está en una fase temprana de desarrollo, lo que hace que trabajar con ella sea más complicado.

En este caso, se va a utilizar el bluetooth 4.0, ya que para el propósito de este proyecto es suficiente y los módulos que lo implementan son más baratos y sencillos de usar. No obstante, el 5.0 se está normalizando rápidamente y se puede contemplar como una posible mejora para este proyecto.

Otra posibilidad que se había estudiado como método de comunicación para este proyecto era el de usar Wifi [2]. Ciertamente, el alcance de este método de comunicación es mayor, así como su ancho de banda y su seguridad. En este caso, la distancia que alcanza nuestro modulo bluetooth es suficiente, la seguridad para el uso que se le va a dar no es un problema y no es necesario un mayor ancho de banda. En contrapartida, el Wifi tiene un problema que sí afecta precisamente al objetivo de este proyecto: el consumo. La tecnología Wifi tiene un consumo mayor que el Bluetooth, y dado que lo que se busca es un sistema de bajo consumo, esto sí que es un factor a tener en cuenta y que inclina la balanza a favor del Bluetooth. Además, como hemos comentado anteriormente, con la versión 5.0 del bluetooth, el ancho de banda y sobre todo el alcance no será un problema. De hecho, con la tecnología de redes Mesh que implementa el bluetooth 5.0 podrá incluso llegar a ser mejor que el Wifi si se dispone de

una buena cantidad de dispositivos que sirvan como nodos, como se explicará más adelante.

Propuesta

Dado el bajo coste que supone el uso de este tipo de tecnologías y su gran efectividad, unido a su bajo consumo y fácil accesibilidad, se propone hacer un sistema de control de cultivos con esta tecnología.

El sistema se basará en la comunicación de los módulos de sensorización basados en Arduino (en este caso para el desarrollo se ha utilizado un único módulo) con la placa Orange Pi Zero, que hará de sistema central.

La placa de Arduino será la encargada de alimentar los sensores y leer los datos que obtengan, para posteriormente enviarlos al sistema central.

La placa Orange Pi Zero hará de sistema central, encargándose de procesar los datos recibidos y de almacenarlos en una base de datos interna.

Los parámetros que se van a medir para el control del cultivo serán:

- Humedad del suelo
- Humedad en el ambiente
- Temperatura ambiente

Como ya hemos comentado, la comunicación entre ambos dispositivos se establecerá mediante bluetooth.

3. Análisis del problema

Para abordar el objetivo de monitorizar un cultivo con dicha propuesta, vamos a analizar los pasos que deberemos de seguir.

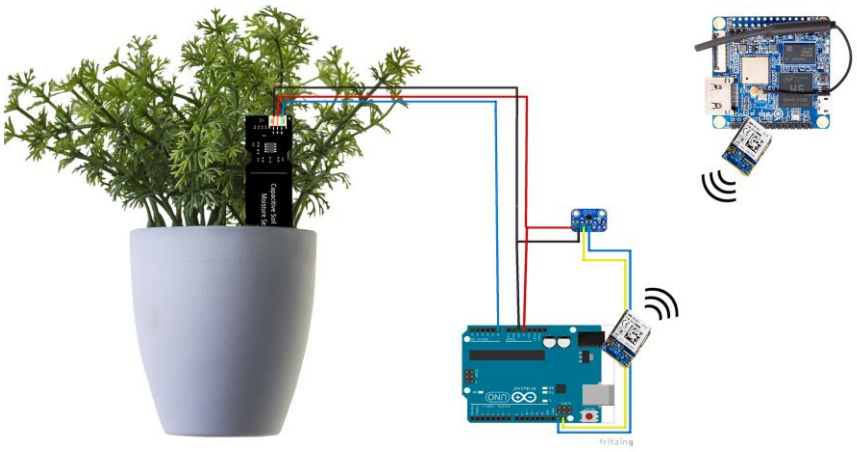

Figura 1: Aproximación al diseño real

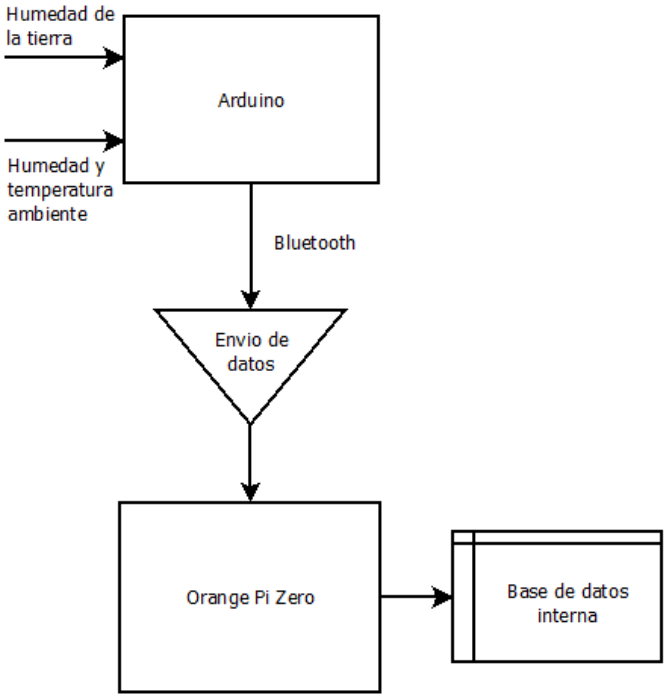

Figura 2: Procesamiento de datos

Como vemos en el diagrama de flujo, el sistema está pensado para dividir el proceso en dos partes:

Bloque de sensorización: Este bloque está formado por la placa Arduino junto a los sensores SKU SEN 0193 (humedad del suelo), SHT31-D (temperatura y humedad ambiente) y el módulo HC-05 (bluetooth). Los sensores estarán alimentados por la Arduino, la cual leerá los valores obtenidos de dichos sensores, para periódicamente enviar a la Orange Pi los datos mediante el módulo bluetooth.

Bloque de procesamiento: Formado únicamente por la Orange Pi Zero y el módulo HC-05. Esta recibirá la información enviada por la Arduino y la almacenará en una base de datos, la cual guardará el nombre de los sensores, los datos enviados, y la hora a la que fueron registrados.

De este modo, accediendo a la Orange Pi podremos saber el estado del entorno de las plantas, y así actuar en consecuencia.

Para hacer más fácil la lectura de los datos, se visualizará la información obtenida de la base de datos en una gráfica que nos muestre como ha variado la temperatura y la humedad ambiente y la humedad del suelo a lo largo del tiempo.

Identificación y análisis de soluciones posibles

El proyecto está orientado a un uso sencillo de la tecnología para controlar los cultivos, pero es de esperar que se pueda llevar más allá.

Una solución posible sería la de omitir el bloque regido por la Orange Pi, usando la misma Arduino para almacenar la base de datos local (añadiendo una microSD como sistema de almacenamiento). El problema de esta solución es que en el caso de querer escalar el proyecto añadiendo más bloques de sensorización con Arduino tendríamos que revisar las bases de datos una a una por cada Arduino, mientras que con el sistema actual tendríamos toda esa información centralizada.

Otra opción similar habría sido la de conectar directamente la Arduino a una base de datos en un servidor web y hacer que esta enviase los datos allí directamente, pero el problema sería que se necesitaría acceso a internet, lo que limitaría su uso.

Por ello, se ha optado por la solución actual, la cual no depende de una conexión a internet y nos permitiría consultar las bases de datos de varios módulos (de haberlos) de manera sencilla y cómoda. Además, si se quisiera expandir el proyecto para controlar más sensores o válvulas, por ejemplo, bastaría con adaptar el código del sistema actual para así poder llevar un control centralizado de todos los módulos Arduino mediante el uso de única Orange Pi.

4. Diseño del conjunto de sensores controlados por Arduino

Vamos a analizar en profundidad cada una de las partes que conforman el sistema.

Arduino Uno R3

Figura 3: Placa Arduino Uno R3

La Arduino Uno [3][4] utiliza el microcontrolador Atmega328, bastante utilizado en este tipo de proyectos, dado que es bastante barato, tiene un bajo consumo y es bastante fácil de usar.

Se utiliza C++ como lenguaje de programación. Además, la placa cuenta con varios pines, tanto digitales (entrada y salida) como analógicos (entrada), además de pines de alimentación (5V, 3,3V, GND) y pines especializados para controlar otros parámetros (AREF y RESET).

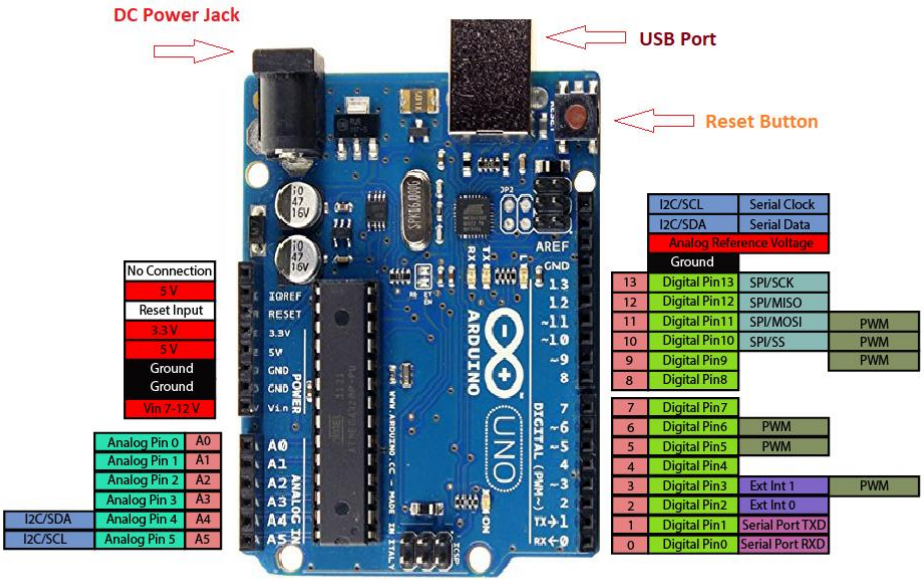

Figura 4: Arduino pinout (www.theengineeringprojects.com)

Como vemos, la placa Arduino bien equipada para ser usada como controlador de un sistema sencillo.

En nuestro caso, vamos a controlar tres dispositivos:

- SHT31-D, temperatura y humedad ambiente (digital)
- SKU SEN 0193, humedad de la tierra (analógico)
- HC-05, módulo bluetooth (digital)

Para comunicarnos con ellos y alimentarlos vamos a hacer uso de casi todos los tipos de pines que incorpora la Arduino.

Además, cabe destacar la gran comunidad y apoyo de la plataforma. Gracias a esto, se puede encontrar mucha documentación de todo tipo, desde tutoriales extensos donde se explica paso a paso como funciona exactamente la placa hasta librerías de código para una gran cantidad de sensores, las cuales hacen que usarlos sea tan fácil como añadirlas al código y utilizar sencillos comandos.

SHT31-D. Sensor de temperatura y humedad digital

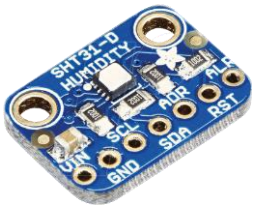

Figura 5: Sensor SHT-31D

El SHT31-D [5] es un sensor digital capaz de medir la temperatura y humedad ambiente.

Utiliza como método de comunicación el protocolo i2c [6] (Inter-Integrated Circuit), un protocolo síncrono de comunicación serie que utiliza dos vías de comunicación, uno para los datos, el SDA, y otro para el reloj, el SCL.

Normalmente, se hacen usos de resistencias Pull-Up, las cuales son unas resistencias que se ponen entre SDA/SCL y la línea de alimentación con el objetivo de definir un estado por defecto a estos pines, y evitar así errores a la hora de interpretar la lectura. En este caso Arduino tiene implementados una serie de resistencias Pull-Up internas que podemos activar mediante software, y que serán suficientes para cumplir con su propósito.

El uso de dispositivos i2c en Arduino está muy bien implementado, ya que, con incluir la librería adecuada, podemos comunicarnos con él de forma sencilla. Además, el protocolo i2c nos da la posibilidad de conectar varios dispositivos al mismo puerto serie, asignando a cada uno una dirección para comunicarnos con el dispositivo que queramos.

Este sensor cuenta con una librería propia que podemos usar para obtener fácilmente la temperatura y humedad mediante el uso de comandos sencillos.

SKU SEN 0193. Sensor de humedad de la tierra

Figura 6: Sensor SKU SEN 0193

En este caso, el sensor utilizado ha sido el SKU SEN 0193 [7].

A diferencia de muchos de los sensores que se suelen utilizar para medir la humedad de la tierra, los cuales suelen ser resistivos, hemos optado por uno capacitivo. La diferencia principal es que los sensores capacitivos son resistentes a la corrosión, dándole una vida útil mucho mayor.

Su funcionamiento es bastante simple, basta con alimentarlo y conectar su salida a una entrada analógica del Arduino. No hará falta uso de ninguna librería específica, solo tendremos que leer lo recibido por el pin analógico.

Como el valor que recibiremos mediante el puerto analógico estará entre 0 y 1023 (entre 0 y 5V respectivamente), tendremos que calibrar el sistema para interpretar correctamente las lecturas. Para ello, según el fabricante tendremos que anotar que valor leemos cuando el sensor está en su máximo de humedad (calibrar según su uso, en este caso sería con la tierra recién regada) y cuando está totalmente seco. De esta forma podemos calcular el porcentaje de humedad relativa.

La ecuación que hemos usado para calibrar el sistema es la siguiente:

$$x = (lecturaSensor - hum.Min) * 100$$
$$C = hum.Max - hum.Min$$
$$Resultado = x/C$$

Ecuación 1: Calibrado del sensor DFRobot

Siendo hum.Min y hum.Max los valores obtenidos con el sensor totalmente seco y saturado de humedad respectivamente, y siendo lecturaSensor los datos recibidos por el sensor.

HC-05. Módulo transmisor y receptor bluetooth

Figura 7: Módulo bluetooth HC-05

Para comunicarnos con la Orange Pi Zero contamos con dos HC-05, módulos bluetooth que funcionan mediante comunicación UART [8] (Transmisor-Receptor Asíncrono Universal). Este protocolo de comunicación se basa en la transformación de los datos recibidos en un formato serie, para posteriormente enviarlos, y viceversa. La velocidad de comunicación se puede aumentar o disminuir (baudrate). En nuestro caso, no es necesaria una comunicación de alta velocidad, por lo que hemos optado por una opción estándar (9600 baudios).

Para la comunicación UART se han utilizado los dos pines básicos, el TX y el RX (transmisión y recepción), aunque otros dispositivos también incorporan otra pareja de pines, CTS y RTS, para el control de flujo.

Este módulo en concreto puede ser configurado como maestro o esclavo. La diferencia entre estos dos modos radica en que es el maestro quien inicia el proceso de emparejamiento, es decir, es el maestro quien se conecta a los esclavos. Por lo tanto, la configuración más lógica es configurar el módulo conectado a la Arduino como esclavo, para que sea la Orange Pi la que tenga el módulo maestro, ya que será esta la que se conectará a todos los demás dispositivos según se vayan agregando.

Montaje y configuración

Como hemos comentado anteriormente, el sensor de SHT31-D utiliza i2c como protocolo de comunicación y dispone de librerías propias, por lo que haremos uso de ellas.

Una vez incluidas, mediante los comandos “readTemperature()” y “readHumidity()” obtenemos el valor de la temperatura en grados centígrados y el porcentaje de humedad.

Para conectar el sensor a la placa alimentamos este conectando los pines VIN y GND a los pines 5V y GND de la Arduino respectivamente. Conectamos el pin SDA del sensor al pin A4 de la Arduino (dicho pin esta designado para esta función además de funcionar como entrada analógica normal) para la transmisión de información. Además, también será necesario conectar el pin SCL, que transmite los datos del reloj, con el pin A5 de la Arduino (al igual que el pin A4, también funciona como entrada analógica).

El sensor de humedad de la tierra solo se tiene que alimentar (conectar a 5V y a GND) y conectar el pin de transmisión de datos al pin A0 del Arduino, ya que es un sensor analógico, y como tal, la respuesta recibida será analógica. Para leer la respuesta del sensor utilizaremos el comando “analogRead()”, uno comando básico de Arduino que, como hemos comentado anteriormente, lee el voltaje recibido (en este caso entre 0 y 5 voltios) y lo transforma a un valor entero entre 0 y 1023. Con el valor obtenido, solo tenemos que implementar la ecuación que hemos explicado anteriormente (Ecuación 1) para obtener el porcentaje de humedad relativa.

Por último, solo necesitamos conectar el módulo bluetooth, el HC-05. La placa Arduino tiene asignados unos pines específicos para la comunicación UART (los pines TX y RX), así que utilizaremos estos, aunque se pueden usar otros pines mediante el uso de una librería específica (softwareserial.h), lo cual nos dejaría programar la Arduino sin bloquear el puerto serie, pero en contrapartida la velocidad máxima de transmisión bajaría. En este caso, la velocidad de transmisión no es algo importante, pero tampoco tenemos la necesidad de mantener comunicación serie con la Arduino y el PC, por lo que se ha optado por la opción de utilizar simplemente los pines designados. Por lo tanto, alimentaremos el módulo bluetooth de la misma forma que el resto de los sensores, y conectaremos los pines TX y RX del bluetooth con los pines RX y TX del Arduino respectivamente. Nótese que la conexión de los pines TX y RX es cruzada. Esto es debido a que la información que se quiera enviar desde la Arduino, saldrá por TX, y será el módulo bluetooth la que reciba esa información, por lo que entrará por el pin RX, y viceversa.

Cabe destacar que es necesario configurar previamente los módulos bluetooth para que se comuniquen entre ellos. Este proceso se explica más adelante.

Una vez conectado el módulo bluetooth, se envía la información como texto mediante la función “Serial.print()”. Primero enviaremos una palabra clave para que la Orange Pi detecte que le están enviando los datos de los sensores. En este caso se ha utilizado la palabra “Start”. La información de los sensores se encapsulará en una cadena de texto específica, de modo que al recibirla la Orange Pi pueda descomponerla, aislando y almacenando los datos.

Al final de cada transmisión se utiliza el comando “Serial.println()”, ya que fuerza el envío de la información. De esta forma nos aseguramos de que la información se envía de una sola vez para que no se tenga errores a la hora de insertar la información en la base de datos debido a que alguna variable sea nula.

A continuación, tenemos una imagen representativa de cómo queda conectado todo este sistema:

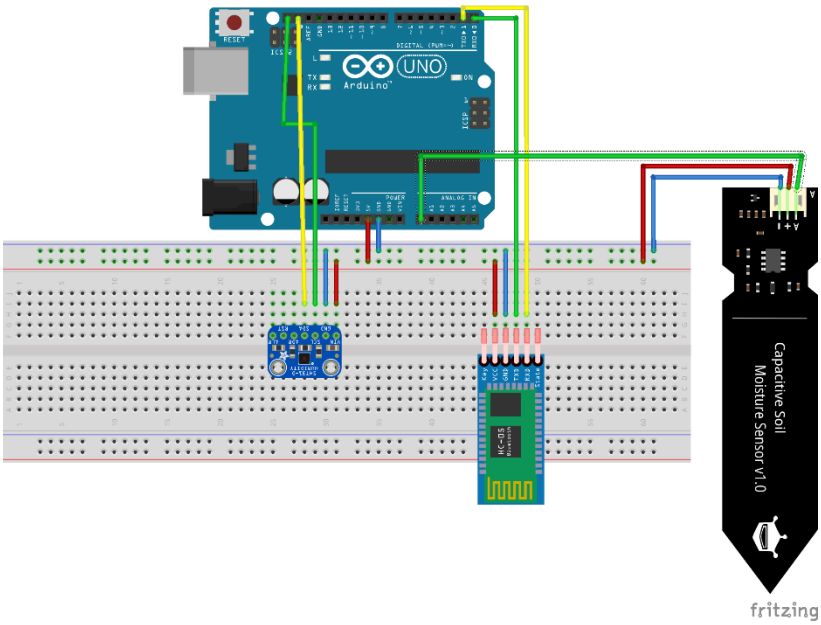

Figura 8: Conexión bloque Arduino

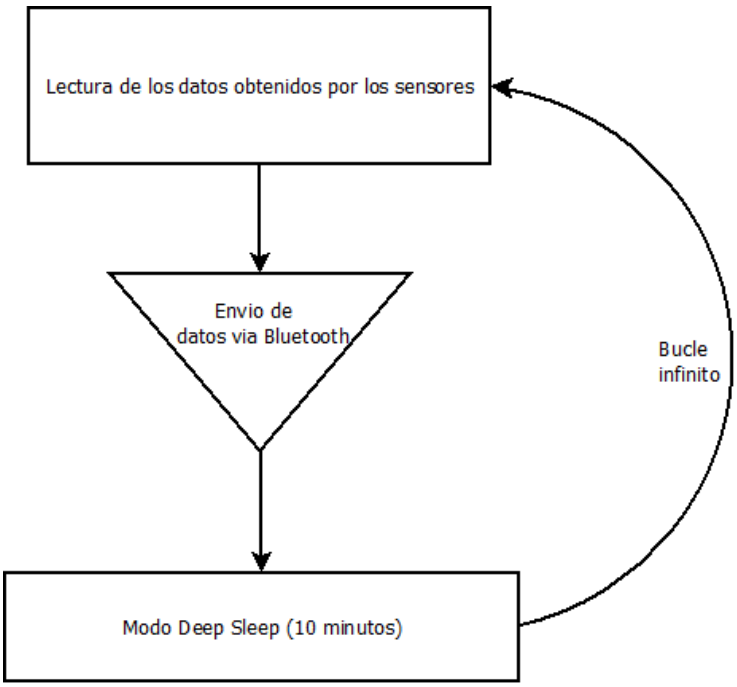

Figura 9: Diagrama de flujo Arduino

El sistema actúa de la siguiente manera:

Al iniciarse, la Arduino entra en un bucle infinito donde leemos y almacenamos de forma temporal los datos de los sensores. Como hemos explicado anteriormente, el sensor SHT31-D utilizamos los comandos definidos en su propia librería, mientras que en el caso del SKU SEN 0193 leemos su respuesta en el pin analógico y aplicaremos la ecuación que nos proporcionará la humedad relativa (Ecuación 1).

Una vez hecho esto, se envían los datos obtenidos mediante bluetooth a la unidad central, y seguidamente hacemos uso del modo Deep Sleep, el cual consiste en el uso de una librería que nos permite en este caso apagar los convertidores analógicos y el circuito Brown Out Detection (BOD) [9], el cual se encarga de detectar las bajadas de tensión para proteger el circuito. Está configurado para estar en este modo durante diez minutos, de esta forma la Arduino solo enviará la información cuando acabe dicho periodo, manteniéndose el resto del tiempo en un modo de bajo consumo.

En principio, la forma más fácil y segura de “dormir” es haciendo uso del Watch Dog Timer para despertarlo, pero este solo nos permite mantener el sistema en este estado durante ocho segundos. Aun así, esto no es un problema, ya que para solucionar este inconveniente es tan sencillo como utilizar un bucle que repita el comando para entrar de nuevo en modo Deep Sleep. Por lo tanto, manteniendo el comando para entrar en el modo de bajo consumo dentro de un bucle que se repita setenta y cinco veces conseguiríamos los diez minutos deseados.

5. Diseño de la unidad central

Orange Pi Zero

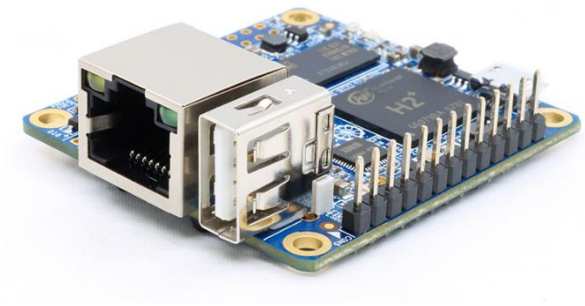

Figura 10: Orange Pi Zero

Esta pequeña placa es lo que se llama una “single-board computer” [10], la cual es de código abierto y es capaz de ejecutar un sistema operativo, normalmente basado en Linux.

Este tipo de placas suelen implementar también una seria de pines GPIO, por lo que combinando esto con su gran potencia de procesamiento y la posibilidad de poder instalar un sistema operativo como Linux, hace que su uso para el desarrollo y la investigación sea habitual.

En el caso de la Orange Pi Zero contamos con 13 pines de interfaz y 26 pines GPIO, un USB, Ethernet, un micro USB OTG y una antena wifi. Además, tiene un puerto para microSD, donde se instalará el sistema operativo y se almacenaran los datos.

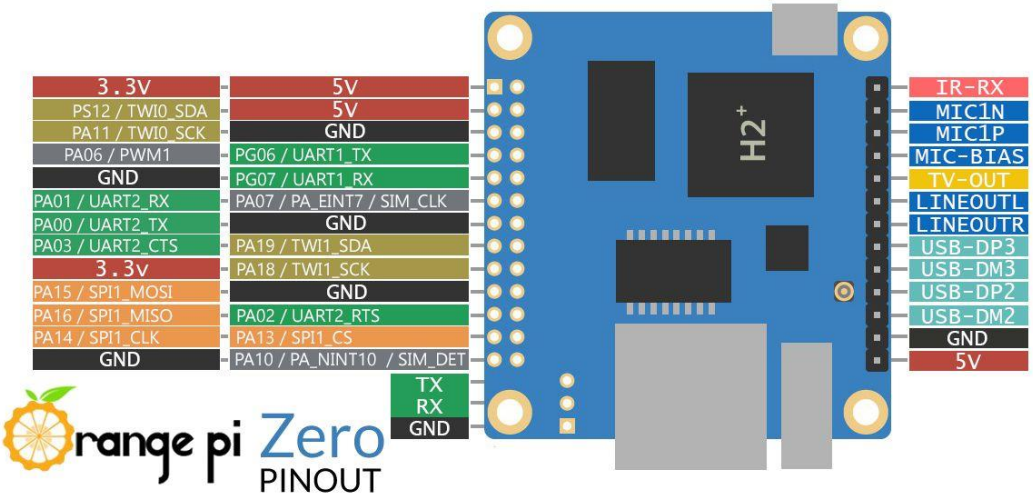

Figura 11: Orange Pi Zero pinout (www.oshlab.com)

Para este proyecto se ha instalado Debian GNU/Linux. La placa no cuenta con salida de video implementada, así que nos hemos comunicado con ella a través del protocolo SSH, el cual nos permite acceder a la Orange Pi a través de la consola. Al tratarse de un sistema Linux, no disponer de un entorno gráfico no es un gran problema, ya que es muy habitual trabajar con estos sistemas utilizando en gran medida la consola de comandos. Mediante esta, podemos instalar todos los programas y paquetes necesarios para el funcionamiento de este proyecto.

Para facilitar el trabajo, se ha hecho uso del programa WinSCP[1], el cual se conecta al dispositivo mediante SSH y nos proporciona de un entorno gráfico para poder ver los directorios sus directorios, pudiendo tanto leer como editar. Además, integra otro programa, Putty[2], un cliente SSH que nos permite acceder al dispositivo mediante una consola de comandos.

[1] https://winscp.net/eng/index.php
[2] https://www.chiark.greenend.org.uk/~sgtatham/putty/

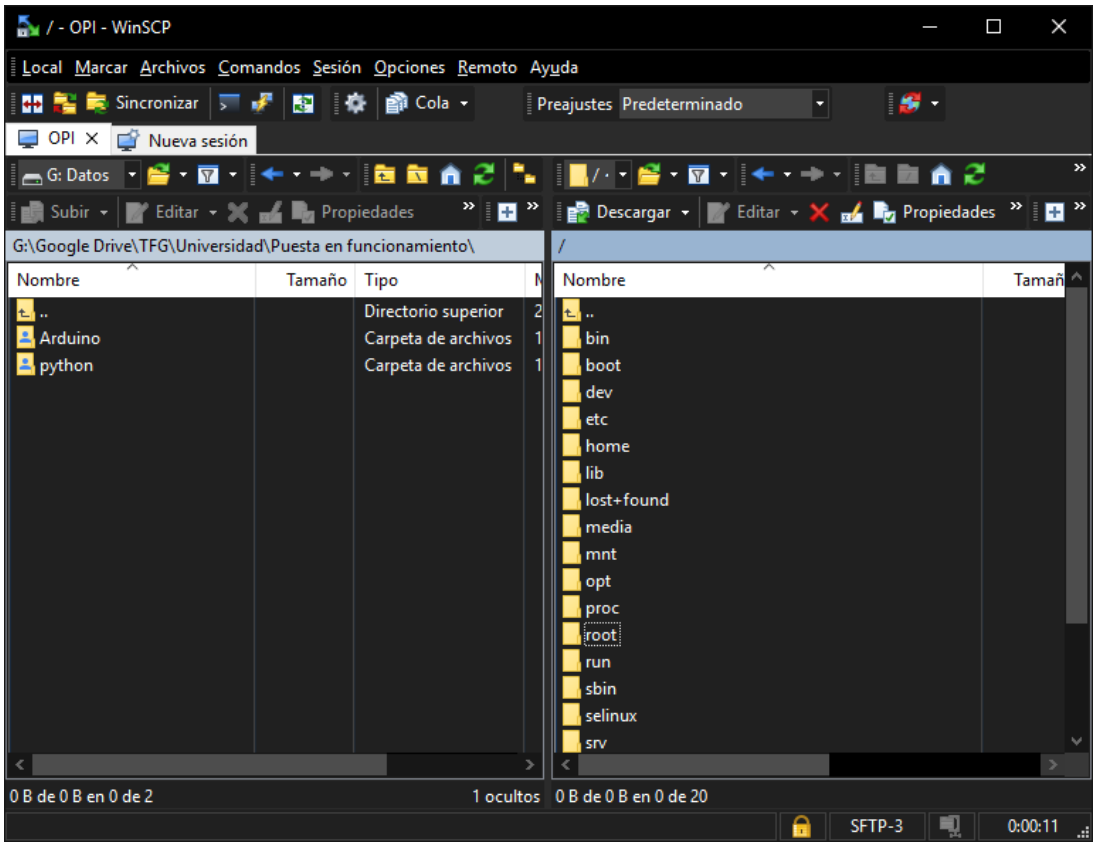

Figura 12: Entorno gráfico WinSCP

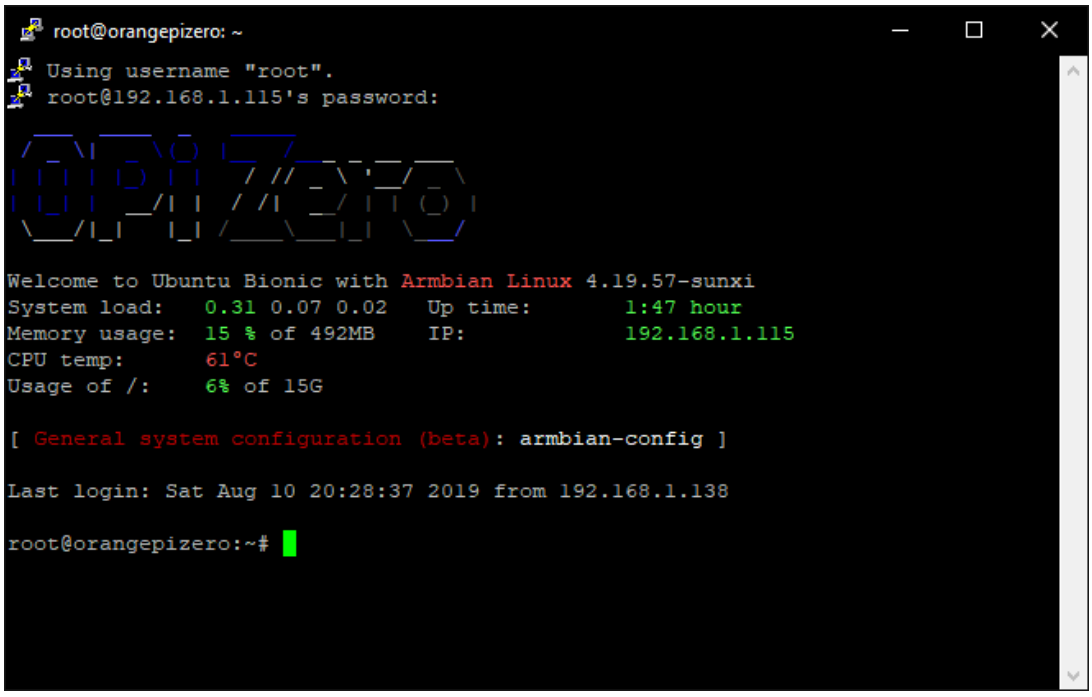

Figura 13: Acceso mediante consola (Putty) a la Orange Pi Zero

En nuestro caso, vamos a usar Python como lenguaje de programación, y SQLite como sistema de gestión de base de datos.

La elección de usar Python como lenguaje de programación es debido a que es muy fácil programar con él. Es sencillo tanto usarlo como entenderlo, y es por ello por lo que mucha gente opta por esta opción para programar con este tipo de placas (la más habitual es la Raspberry pi), y por eso mismo es fácil encontrar documentación o solución a varios problemas en internet. Además, es un lenguaje orientado a objetos, al igual que el lenguaje aprendido en la carrera, Java, lo que hace que sea más fácil de usar.

En cuanto a la base de datos, se ha elegido SQLite porque no necesita de un servidor, además de ser una base de datos muy ligera, eficiente y simple. En esta base de datos se guardará la información recibida de la Arduino.

Montaje y configuración

Al igual que la Arduino, la Orange Pi Zero tendrá conectado, haciendo uso de los pines GPIO, el módulo bluetooth mediante UART.

Utilizando un script en Python, la placa leerá los datos obtenidos por el módulo bluetooth y se encargará de almacenarlos en las tablas de la base de datos que almacenará de forma local en la microSD.

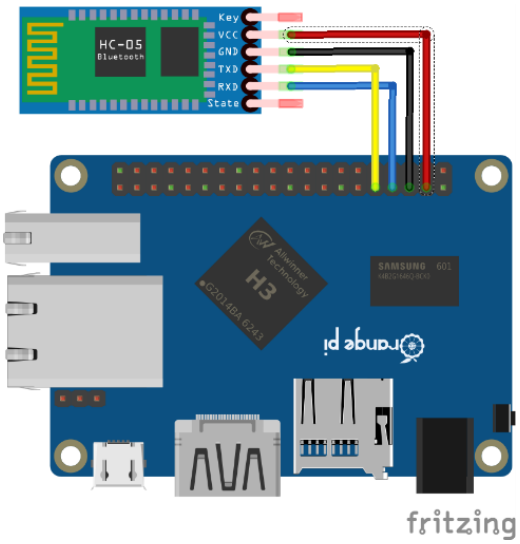

Figura 14: Conexión bloque Orange Pi Zero

(Dado que no se ha encontrado el recurso de la placa Orange Pi Zero para Fritzing[3], el programa utilizado para hacer las figuras de los montajes, se ha utilizado el de una placa Orange Pi One).

[3] https://fritzing.org/home/

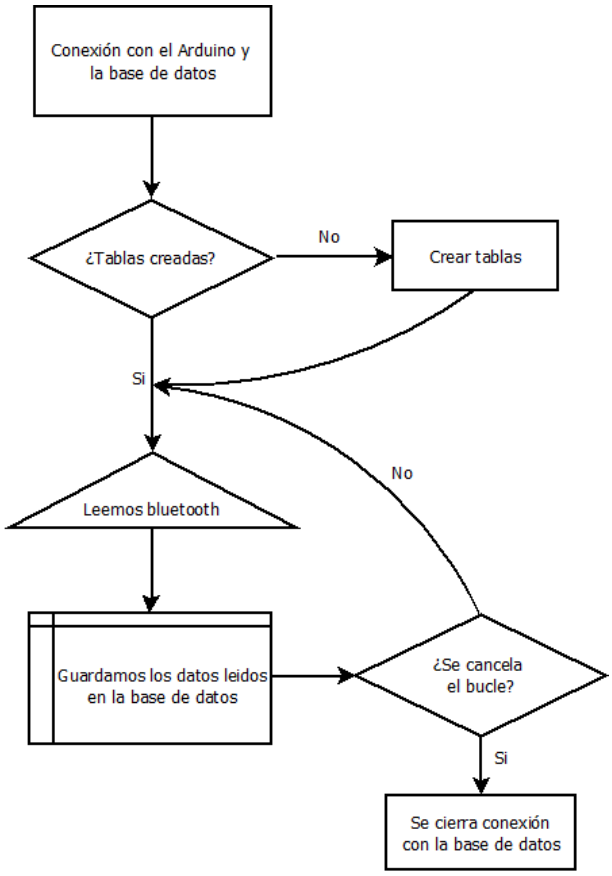

Figura 15: Diagrama de flujo del funcionamiento de la Orange Pi Zero

Como vemos explicado en el diagrama, el script empieza estableciendo comunicación con el módulo bluetooth. Seguidamente abre comunicación con la base de datos interna.

Comprueba si la base de datos está vacía (no hay tablas). En caso de estarlo, creará las tablas.

Una vez hecho esto, se pasa al bucle infinito, donde se lee la información recibida a través del bluetooth. Como hemos visto, la Arduino envía los datos en forma de texto, usando un “Serial.println()” al final de cada bloque de datos para forzar el envío, además de como texto la palabra “Start” al principio, evitando de esta forma cualquier tipo de error de lectura o almacenamiento.

Una vez los datos han sido recibidos, se extrae de la trama de texto los datos de los sensores, se almacenan en sus respectivas variables y se envían a la base de datos mediante comandos SQL. Además, se adjunta la fecha y la hora en la que se han tomado las muestras, para que sea más fácil hacer un seguimiento útil en la práctica y poder visualizar mejor los resultados usando gráficas.

6. Conexión y comunicación

Para utilizar los módulos bluetooth primero debemos configurarlos. Esto lo conseguimos mediante comandos AT.

Los comandos AT [11] son instrucciones que se utilizan para configurar los dispositivos a través del puerto serie. En nuestro caso, los utilizaremos para configurar los módulos bluetooth, para así establecer una comunicación entre ellos.

Para este caso, se han utilizado dichos comandos para cambiar el nombre del dispositivo, la contraseña/pin que van a utilizar para el emparejamiento, el rol que van a desempeñar (maestro o esclavo) y especificar si se va a conectar a un dispositivo en específico o va a ser una comunicación más transparente.

Para enviar los comandos AT a los módulos, se ha configurado la Arduino para establecer comunicación serial con el bluetooth de forma que se pueda tanto enviar comandos como recibir la respuesta acorde. Además, debemos tener en cuenta que hay que cambiar los pines que utiliza la Arduino para comunicarse con el módulo bluetooth, ya que como hemos comentado antes, la comunicación con el pc queda bloqueada con la conexión actual. Por lo tanto, se ha pasado a utilizar los pines digitales 10 y 11 para comunicarse con los pines RX y TX del módulo bluetooth respectivamente, haciendo uso de la librería “softwareserial” para utilizarlos como puerto UART.

Antes de conectar el módulo a la Arduino, en este módulo en concreto es necesario mantener presionado el botón que incorpora, y conectar la alimentación. Una vez hecho esto, el módulo estará en modo configuración AT.

El baudrate que se ha utilizado es de 38400. Esto es debido a que para entrar al modo de configuración mediante comandos AT de este módulo se debe de utilizar dicho baudrate.

Los comandos AT utilizados para la configuración han sido los siguientes [12]:

NAME	AT+NAME=x	Modifica el nombre del dispositivo
	AT+NAME?	Averigua el nombre del dispositivo
PASSWORD	AT+PSWD=”x”	Modifica la contraseña que utiliza para la vinculación
	AT+PSWD?	Muestra la contraseña de vinculación
ROLE	AT+ROLE=x	Modifica el rol del módulo. O para esclavo y 1 para maestro
	AT+ROLE?	Averigua el rol del módulo
MODE	AT+CMODE=x	Modifica el modo en el que actúa el módulo con el rol de maestro. Es

		0 cuando se configura para conectarse a una dirección en particular y 1 para cualquier esclavo disponible
	AT+CMODE?	Averigua el modo
UART	AT+UART=x,y,z	Modifica la configuración UART. X=baudrate Y=bit de parada Z=paridad
	AT+UART?	Averiguael baudrate, bit de parada y la paridad de la configuración UART
ADDRESS	AT+ADDR=x:x:x:x	Modifica la dirección del módulo
	AT+ADDR?	Averigua la dirección del modulo
BIND	AT+BIND=x:x:x:x	Añade/modifica el módulo con el que se va a conectar
	AT+BIND?	Averigua, en caso de tenerlo, el módulo con el que se va a conectar

Tabla 1: Comandos AT utilizados

Configuración del bluetooth esclavo

```
+AT+NAME=BtArduino
+AT+PSWD="1598753"
+AT+ROLE=0
```

Una vez hecho esto, comprobamos que todos los cambios se han realizado correctamente preguntándole al módulo los campos modificados

```
+AT+NAME?
+BtArduino
+OK
+AT+PSWD?
+PSWD: 1598753
+OK
+AT+ROLE?
```

```
+ROLE:0
+OK
+AT+UART?
+UART:9600,0,0
+OK
+AT+ADDR?
+ADDR: 18:e4:354dad
+OK
```

Configuración del bluetooth maestro

La configuración va a ser parecida, pero además añadiremos los pasos necesarios para establecer la conexión solamente con el bluetooth del Arduino.

```
+AT+NAME=BtOpi
+AT+PSWD=”1598753”
+AT+ROLE=1
+AT+CMODE=0
```

Le especificamos la dirección que hemos obtenido anteriormente del bluetooth del Arduino.

```
+ AT+BIND=0018,e4,354dad
```

Realizamos la comprobación para asegurarnos del resultado. Deberíamos obtener unas respuestas similares a las que hemos recibido con el anterior módulo, cambiando eso si la respuesta en función de lo que le hemos especificado (otro nombre, otro rol…). Las respuestas de los comandos adicionales que hemos usado son las siguientes:

```
+AT+CMODE?
+CMODE=0
+OK
+AT+BIND?
+BIND: 18:e4:354dad
+OK
```

Una vez configurados, se emparejarán automáticamente cuando se enciendan.

7. Prueba experimental

Ya explicado cómo funciona el sistema y como está planteado, pasamos a las pruebas reales.

Figura 16: Implementación real del sistema

Como vemos, tenemos el sensor de humedad del suelo introducido en una maceta con bolas de arcilla, las cuales se usan porque drenan mejor los sustratos, por lo que es idóneo para la fase de crecimiento y alimentación de las plantas. A su lado, tenemos la Arduino con el sensor de humedad y temperatura ambiente, así como el módulo bluetooth.

Una vez montado el sistema, lo único que hay que hacer es cargar el programa en la Arduino e iniciar el script en Python que tenemos en la Orange Pi Zero. Como hemos visto, la función de la Orange Pi Zero será la de recibir los datos enviados desde la Arduino y almacenarlos en una base de datos interna, por lo que al ejecutar el script nos generará un archivo que contendrá las tablas con toda la información que ha ido recibiendo.

Visualización de la información

Para visualizar de forma sencilla el archivo de la base de datos, utilizaremos el programa “DB browser for SQLite[4]”.

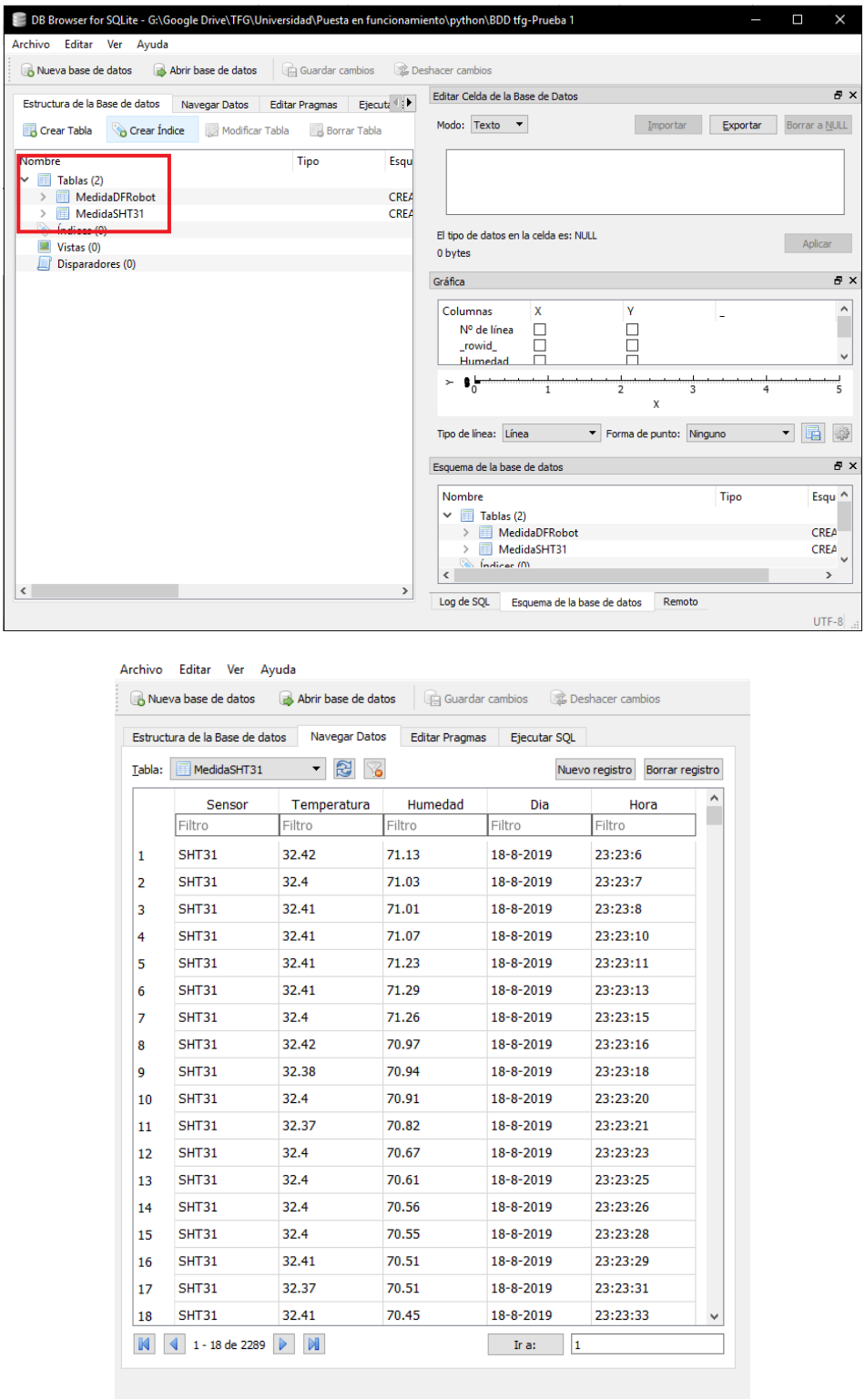

Figura 17: Interfaz del "DB browser for SQLite"

[4] https://sqlitebrowser.org/

Con este programa podemos ver fácilmente las tablas y la información que contienen.

Además, para visualizar mejor la evolución de los datos obtenidos se ha hecho un pequeño script en Python para visualizar en graficas los datos obtenidos [13].

Lo que conseguimos es que use el archivo generado de la base de datos y lo represente en tres gráficas diferentes: Una gráfica para la temperatura ambiente, otra para la humedad ambiente y una última para la humedad del suelo. El resultado es el siguiente:

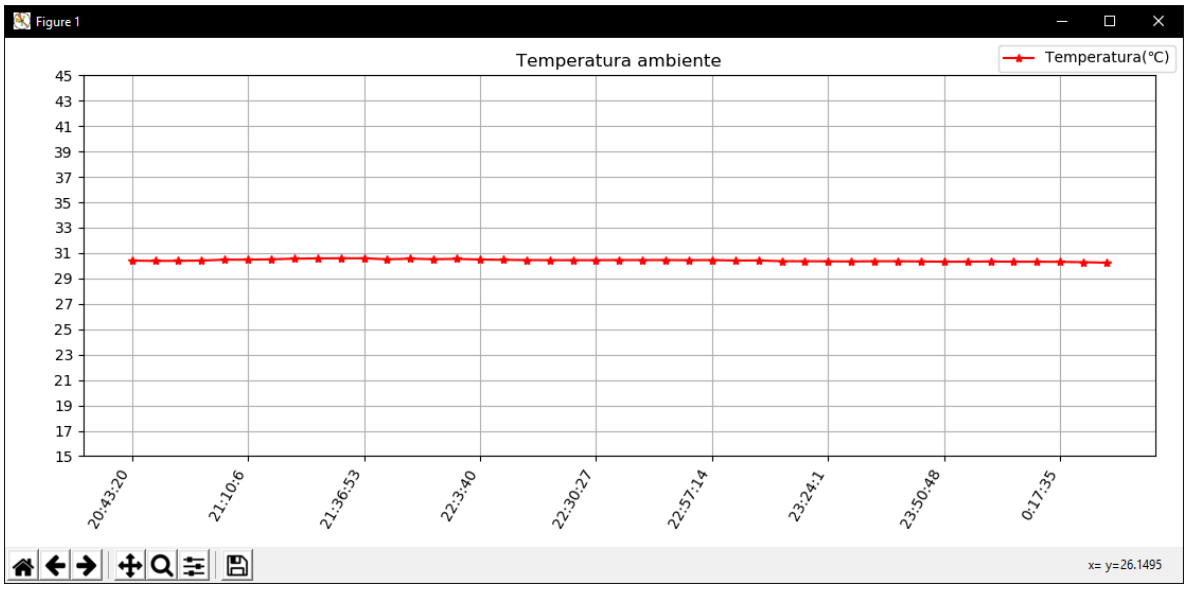

Figura 18: Gráfica de temperatura ambiente

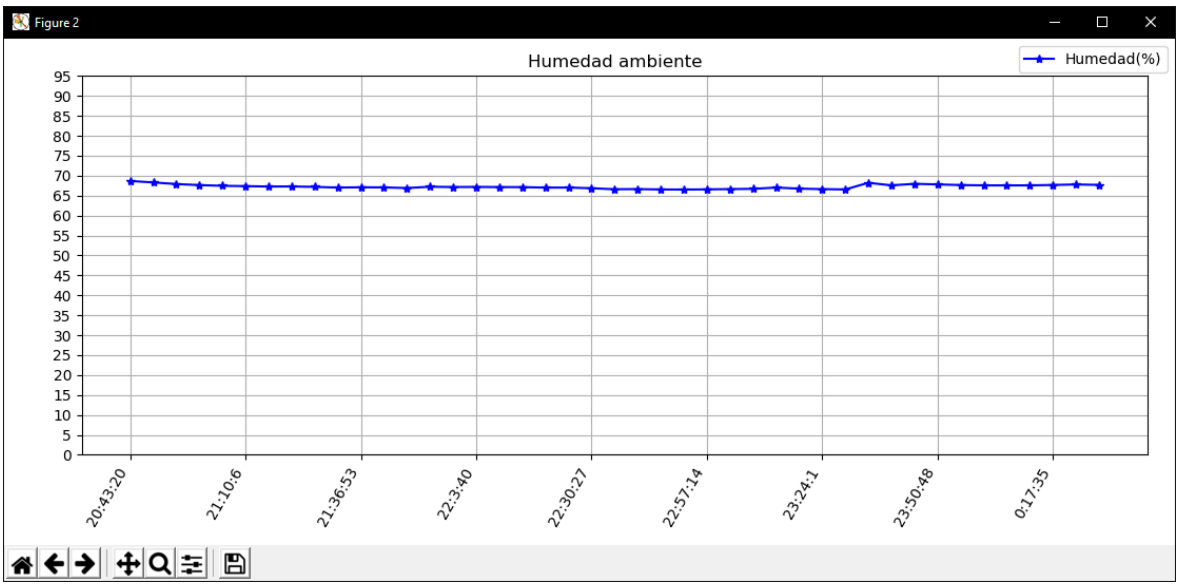

Figura 19: Gráfica de humedad ambiente

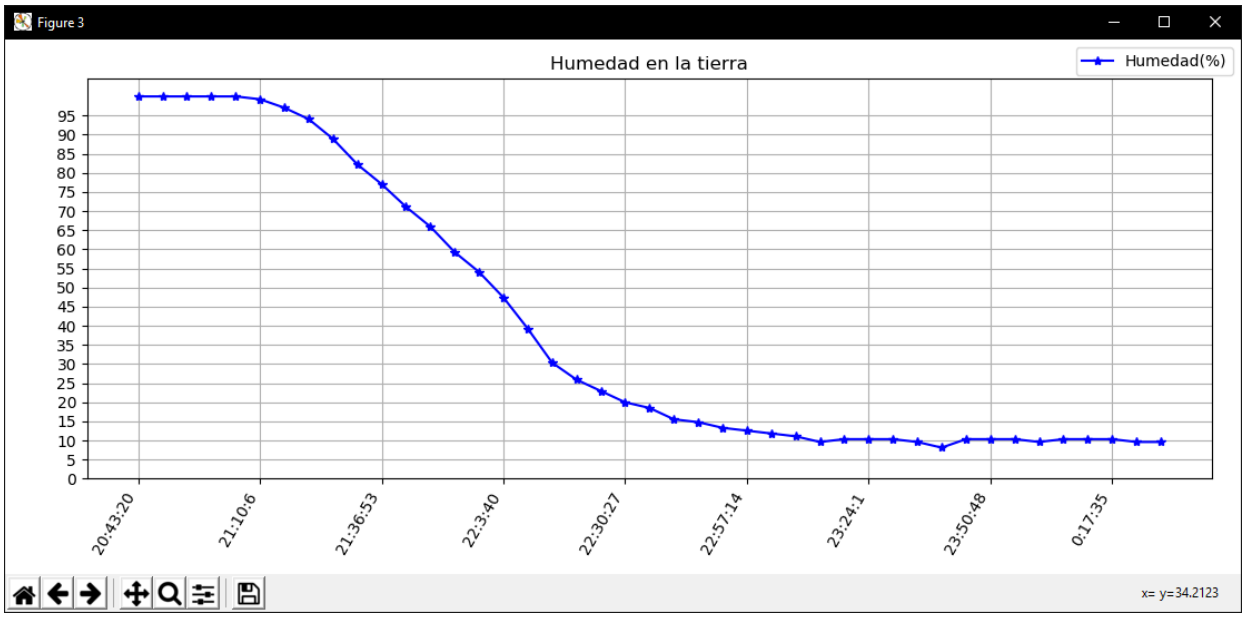

Figura 20: Gráfica de humedad del suelo

Las gráficas para representar la humedad se han representado en un intervalo de 0 a 100%, y la temperatura ambiente entre los 16 y 48 grados centígrados (ya que actualmente es imposible que llegue a acercarse siquiera a dichos valores).

Ya que la humedad (tanto ambiente como del suelo) y la temperatura no es algo que varíe rápidamente, para esta prueba hemos tomado una muestra cada 6 minutos aproximadamente durante 2 horas y 5 minutos, punto donde la humedad de la tierra deja de disminuir.

Con esta prueba hemos podido comprobar la estabilidad del sistema tanto por la parte de la Arduino como la de la Orange Pi. Esta última, al tratarse de un script, en caso de fallo puede cancelar su proceso y no volver a ponerse en funcionamiento sin nuestra intervención, lo que ha sido un problema para resolver en los primeros intentos.

Problemas encontrados

Los problemas que encontramos a la hora de ejecutar el script eran debidos a dos factores.

El primero era el formato de la información recibida. Si no se especifica el formato, este lo decodifica en formato ASCII, lo que provocaba errores al no poder decodificar bien la información. Esto se debe a que este tipo de microcontroladores usan como formato el utf-8. Este tipo de codificación es un estándar ampliamente utilizado por su capacidad de representar cualquier carácter en Unicode y de utilizar símbolos de longitud variable, entre otros.

El segundo problema era debido a la generación de gráficas. En un primer diseño se había pensado en hacer que las gráficas se generasen y actualizasen de forma dinámica a medida que la Orange Pi recibía los datos. Como al principio para hacer las pruebas se tomaban muestras cada cinco o diez segundos, el sistema terminaba

sobrecargándose hasta fallar, debido a que la representación gráfica utilizaba demasiados recursos.

Al final se terminó eliminando del código principal esta función, ya que el objetivo de la Orange Pi no es el de mostrar las gráficas a tiempo real, sino de almacenar los datos. No obstante, el código para generar las gráficas se ha podido adaptar para formar el script usado actualmente para representar todos los valores ya almenados en los archivos generados.

Interpretación de los datos obtenidos

Los datos obtenidos han sido los esperados. Como vemos, la temperatura y la humedad ambiente son los que menos han variado. Esto es debido a que se ha hecho la medición en un interior y durante un periodo de tiempo no muy prolongado.

Donde sí que vemos un cambio significativo es en la medición del sensor de humedad del suelo. Como podemos apreciar, al principio empieza con un 100% de humedad, momento en el que se regó la maceta. Una vez regada, conforme pasa el tiempo vemos como la humedad empieza a decrecer rápidamente. Esto es debido a que hemos usado bolas de arcilla, que como hemos comentado anteriormente, absorbe muy rápido los líquidos. El descenso de la humedad continua hasta el punto donde la maceta está casi seca, alrededor del 10% de humedad, donde parece estabilizarse (ya que al estar en interior no recibirá luz solar, por lo que tardará bastante en secarse del todo).

Para conseguir una buena lectura de la humedad del suelo se ha tenido especial cuidado en intentar que el sensor estuviera en contacto con las bolas de arcilla lo máximo posible, ya que las zonas donde toque con estas pueden generar un error de lectura, mostrando picos muy altos o muy bajos en la gráfica. En la siguiente figura tenemos una gráfica de una de las primeras puestas en funcionamiento donde sucedió lo mencionado:

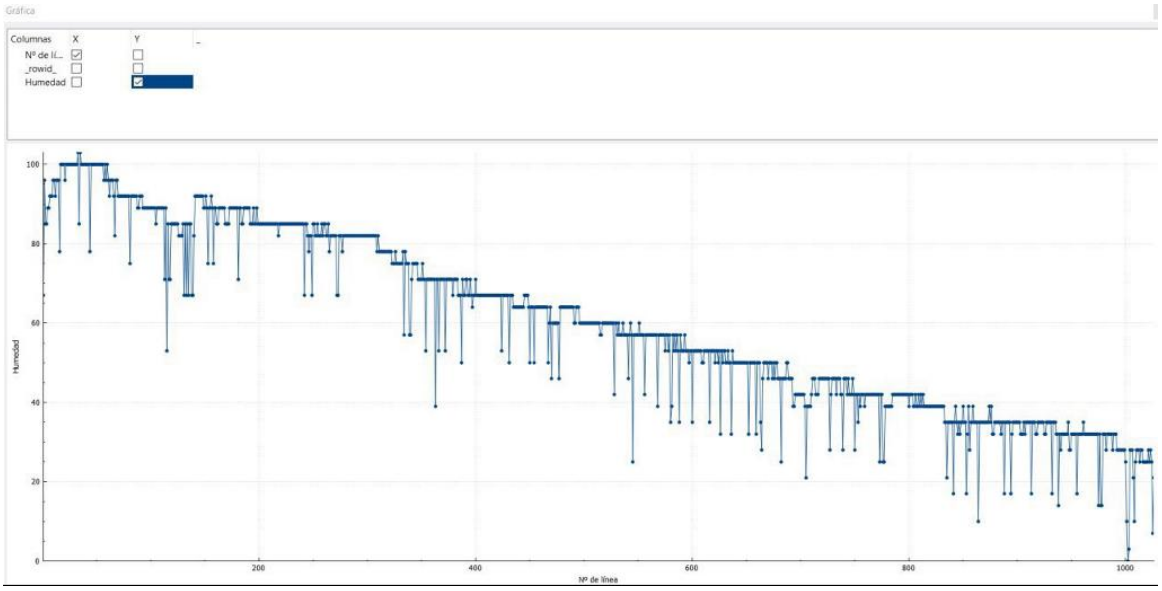

Figura 21: Resultados de las primeras pruebas

8. Análisis energético

Dado que se quiere diseñar un sistema de bajo coste energético, es necesario estudiar su consumo.

Consumo Arduino

Para medir la corriente consumida por la Arduino utilizaremos un multímetro como amperímetro conectándolo en serie entre la placa y la fuente de alimentación.

Como hemos comentado anteriormente, haremos uso del modo Deep Sleep para conseguir un consumo de energía lo menor posible. Por lo tanto, para estudiar el impacto en el consumo que causa este modo, hemos hecho las siguientes medidas:

I. Medida sin modo Deep Sleep
II. Medida con modo Deep Sleep

De esta forma, podremos comparar el ahorro energético que conseguimos con el modo Deep Sleep.

Para la medida sin Deep Sleep, se ha añadido un delay después de la lectura y envío de datos, ya que es lo que se habría usado en el caso de no haber usado dicho modo de ahorro de energía.

Resultados

	Sin Deep Sleep (mA)	Con Deep Sleep (mA)
Consumo en lectura y envío	78.9-76	78.8-76
Consumo en reposo	61.5-56.2	42-40.3

Tabla 2: Resultados de consumo

Como podemos observar, el consumo cuando el sistema lee los datos y los envía es el mismo. No obstante, en lo respectivo al consumo en reposo, es decir, cuando el sistema no está haciendo nada, sí que cambia en las distintas configuraciones.

Al realizar la medida sin modo Deep Sleep, cuando llega a la fase de reposo, es decir, entra en el delay, notamos un descenso del consumo. Tras hacer varias pruebas, se ha visto que la placa entra en un modo de bajo consumo cuando el delay es bastante grande (al transcurrir alrededor de quince segundos). Antes de que entre en dicha fase de bajo consumo los valores rondan los 61.5-58.2 mA, y después entre 57.6-56.2.

Al hacer la medida utilizando el modo Deep Sleep obtenemos el resultado esperado. Utilizando este método el consumo ronda los 42-40.3 mA. Por lo tanto, dicho método sí que es de utilidad para conseguir un menor consumo cuando la Arduino está en reposo.

Consumo Orange Pi Zero

Para medir el consumo de la Orange Pi Zero conectaremos, como hemos hecho con la Arduino, el multímetro en modo amperímetro en serie entre esta y la fuente de alimentación.

La Orange Pi Zero por defecto varia la frecuencia de la CPU según la demanda que reciba, y sus frecuencias preestablecidas se encuentran entre los 480 MHz y los 1.4 GHz. Para conseguir el mayor ahorro posible se ha forzado a la CPU a trabajar siempre a 120MHz, la mínima frecuencia que admite.

Para conseguir primero se ha cambiado la configuración de la placa de "ondemand"(según la demanda) a "powersave"(ahorro de energía). De esta forma forzamos al sistema a usar la frecuencia mínima que tiene configurada. El segundo paso ha sido cambiar dicha frecuencia mínima, es decir, hacer un underclock, llegando a bajarla hasta los 120MHz, su frecuencia mínima límite.

Para llevar a cabo todos estos cambios se ha hecho uso de varios comandos. El primero de ellos es el de **sudo cpufreq -set info,** donde se especifica los valores de frecuencia que admite la placa. De esta forma se ha obtenido el valor mínimo de frecuencia, 120 MHz.

Sabiendo la frecuencia mínima, mediante el comando **sudo cpufreq -set -g powersave** se ha cambiado del modo "ondemand" al modo "powersave". Una vez hemos conseguido que el sistema funcione siempre con la mínima frecuencia, bajamos esta al mínimo permitido con el comando **sudo cpufreq -set –min 120**, estableciendo el mínimo en los 120 MHz.

Para asegurarnos de que los cambios se llevaban a cabo se ha utilizado el comando **armbianmonitor -m** en una segunda consola para poder monitorizar la frecuencia a la cual se encuentra funcionando la CPU.

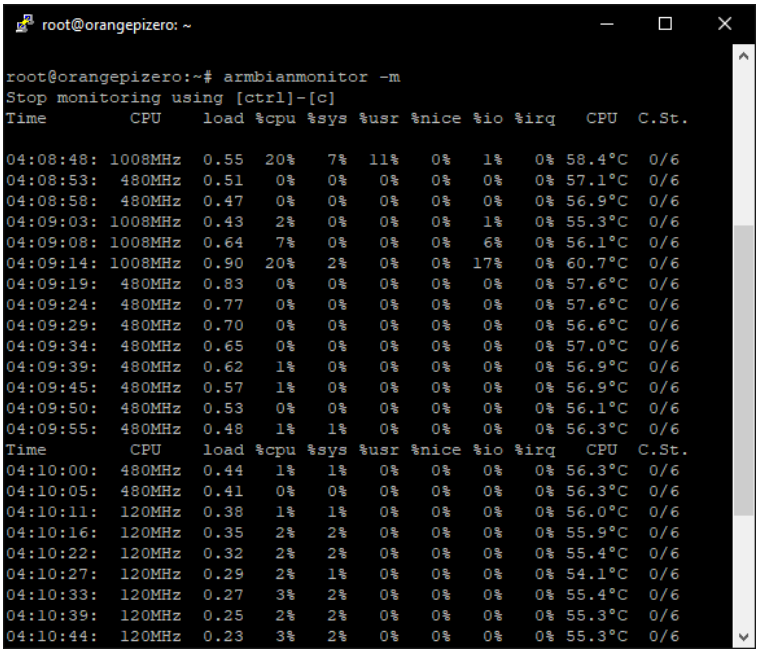

Figura 22: Consumo CPU Orange Pi Zero

Para realizar las medidas también se ha tenido en cuenta si se ha conectado la placa mediante cable Ethernet o Wifi, ya que el consumo varia en consecuencia.

Por lo tanto, se ha realizado las mediciones de la siguiente forma

- Medición con cable Ethernet y configuración por defecto
- Medición con cable Ethernet y modo powersave+underclock
- Medición con Wifi y configuración por defecto
- Medición con Wifi y modo powersave+underclock

(Para referirnos al modo powersave+underclock lo haremos llamándolo simplemente modo ahorro).

Resultados

	Ethernet+defecto(mA)	Ethernet+ahorro(mA)	Wi-Fi+defecto(mA)	Wifi+ahorro(mA)
Recibiendo información	240-210	180-160	270-250	250-230
Esperando información	200-150	160-140	260-200	240-200

Tabla 3: Consumo Orange Pi Zero

Como vemos, al hacer uso de un cable Ethernet el consumo es considerablemente menor. También vemos como el modo ahorro de energía que se ha configurado consigue un mayor efecto cuando se usa también el cable Ethernet, por lo que, a la hora de implementar el sistema, para tener acceso a la Orange Pi Zero lo recomendable es conectar la placa a la red mediante cable Ethernet.

A la hora de realizar las mediciones, cabe destacar que el consumo obtenido durante la espera de la información se mantenía bastante estable alrededor del valor mínimo que podemos ver en la tabla, subiendo de vez en cuando hasta el valor máximo indicado.

9. Conclusión

Se han logrado cumplir con todos los objetivos propuestos. El objetivo principal, diseñar el módulo de sensorización basado en Arduino, junto con el objetivo de configurar un protocolo de comunicación mediante bluetooth han sido relativamente fáciles de abordar, dejando como el más complicado el diseño del módulo central.

La toma de contacto con la Orange Pi Zero ha sido un poco difícil al principio, dado que hacen falta cierto conocimiento sobre este tipo de dispositivos y sobre el software que lo ejecuta (Linux). Además, este dispositivo en concreto no tiene conexión HDMI al que poder conectar una pantalla para acceder a un gráfico, lo que hace que el usarla por primera vez sea más costoso. Por lo que aparte de aprender las bases de cómo funciona, decidí hacer uso del WinSCP, programa mencionado anteriormente, el cual ha hecho más amigable la forma de interactuar con la Orange Pi.

Además de los objetivos marcados, también se ha programado un script en Python para poder ver los datos obtenidos y almacenados en la base de datos en una gráfica. Esto es un añadido interesante ya que hace muy fácil acceder e interpretar las medidas, en vez de mirar directamente la tabla almacenada.

Cabe mencionar que a la hora de llevar el sistema a la práctica hubo contratiempos. En un principio se usó una maceta con una planta en tierra normal y sin luz directa del sol. Esto provocó que la humedad de la tierra apenas bajase, llegando a pasar días sin notar un descenso notable en la humedad. Es por ello por lo que se decidió pasar a usar bolas de arcilla en vez de tierra. De esta forma, para hacer las pruebas experimentales no era necesario pasar largos periodos de tiempo recabando muestras para conseguir ver el descenso de la humedad del suelo.

También se rehízo el estudio de consumo energético en relación a la placa Arduino. En un principio se había usado una fuente de alimentación externa que proporcionaba una salida estable de 5V, la cual se conectaba al pin Vin de la Arduino. El problema es que para alimentar de esta forma la placa el voltaje que se recomienda es entorno a los 7-12V, ya que al pasar por el regulador de voltaje sufre una caída de 2V aproximadamente. Esto puede provocar inestabilidad en la Arduino, y aunque no se ha detectado ningún problema de funcionamiento al alimentarla de esta forma, no podemos asegurar que siga funcionando adecuadamente a largo plazo, ya que las lecturas del consumo que se obtenían eran exageradamente bajas, casi un 50%, de lo que podemos deducir que podría provocar problemas en un futuro en la Arduino o alguno de los módulos conectados por no haber tenido una correcta alimentación.

El desarrollo de este proyecto a supuesto enfrentarme a varios problemas que no había tenido en cuenta, me ha permitido aprender más sobre Arduino e introducirme en el uso de ordenadores de placa reducida como la Orange Pi, aprendiendo además un poco de Linux básico.

Futuras mejoras

En la realización del proyecto, se han pensado varias mejoras posibles para el sistema que se ha desarrollado.

Bluetooth V5.0

Como hemos comentado anteriormente, existe una versión de bluetooth más actual, la versión 5.0.

En un futuro, se ha pensado utilizar el módulo BL652 de Nordic, el cual es muy compacto y ciertos modelos tienen la posibilidad de integrar una antena externa, lo que mejoraría su rango.

Figura 23: BL652 Nordic

Para integrar dicho módulo en nuestros sistemas, es necesario diseñar una PCB para poder conectarla adecuadamente.

Actualmente ya se ha desarrollado una PCB para conectar el módulo junto a otros sensores a la placa Arduino tanto mediante UART como I2c:

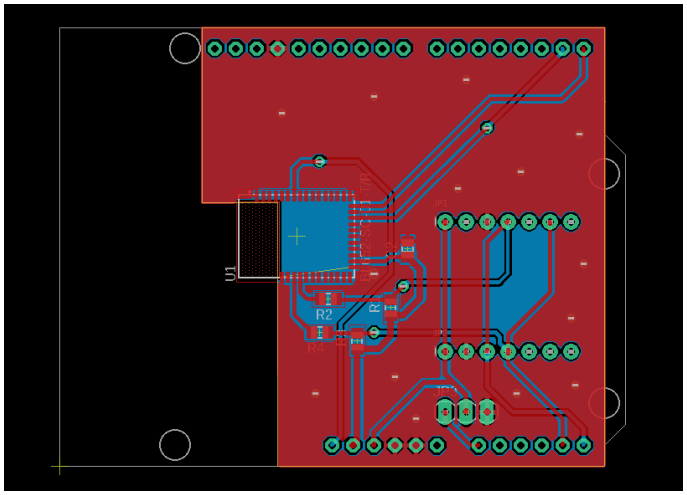

Figura 24:PCB BL652 + Sensores

La función más interesante del bluetooth 5.0 es, como hemos comentado anteriormente, la tecnología Mesh. Esta funcionalidad es especialmente útil en proyectos como este, ya que se podría aplicar el sistema en una gran zona de cultivo, emplazando varios módulos de sensorización que a su vez contribuirían con la creación

de una red Mesh, haciendo posible que la Orange Pi pueda monitorizar todos los módulos independientemente de su lejanía.

Uso de placas Arduino de menor consumo

Para realizar el desarrollo de este proyecto se ha optado por la versión de Arduino Uno R3, ya que es la más común para empezar a experimentar con este tipo de placas. No obstante, el software que se ha desarrollado puede ser usado en otras placas Arduino siempre y cuando cuente con los pines de los que se ha hecho uso (además de hacer los arreglos necesarios a nivel de asignación de pines en el software).

En este tipo de proyectos se suele utilizar por ejemplo la Arduino Nano, una versión más reducida y con menos pines, pero con los suficientes para conectar todos los módulos que hemos usado, además de tener un consumo bastante menor. Además, existe una nueva versión muy reciente la cual está enfocada para las tecnologías IoT que incorpora un módulo bluetooth 5.0 que utiliza el mismo procesador que el módulo de Nordic comentado en el apartado anterior, el nRF52840, lo cual la convierte en una de las mejores opciones, aunque su precio es aún algo elevado.

Otra opción también habría sido utilizar la ESP32, una placa compatible con el compilador de Arduino que se ha estado usando mucho actualmente para este tipo de proyectos. Incorpora un módulo BLE y cuenta con una gran cantidad de pines, aunque no incorpora una salida de alimentación de 5V, lo cual tampoco llega a ser un problema importante debido a que por ejemplo el SHT31-D puede funcionar a 3.3V, y existen en el mercado sensores de humedad del suelo que operan también con dicho voltaje. Además, su característica más destacable es su extremadamente bajo consumo, sobre todo cuando se encuentra en modo reposo (inferior a los 5 μA), lo cual también lo convierte en un gran candidato.

Control de riego

El sistema actual puede escalarse a un proyecto mucho más grande, añadiendo control de riego y de abonado.

Para ello necesitaríamos añadir un sensor de nivel de CO2 [14] del aire además de sensores para el pH, temperatura y conductividad en el sustrato para las plantas.

Un nivel inadecuado de CO2 en el aire puede afectar al crecimiento de las plantas, ya que sin la cantidad adecuada de este puede provocar que estas no se alimenten bien, o que su exceso influya en el proceso normal de fotosíntesis.

El control de pH [15] sirve para llevar un control de los nutrientes en la tierra, midiendo su acidez o alcalinidad (ácido cuando el pH es bajo y alcalino cuando es alto). Si el nivel de acidez del suelo no es el adecuado, esto provocará que las plantas no puedan absorber bien los nutrientes.

La conductividad eléctrica [15] nos permite saber la salinidad del sustrato, es decir, la cantidad de nutrientes que contiene. El añadir componentes al sustrato eleva al EC, por eso es importante llevar un control de este parámetro dado que afecta al proceso de fertilización y a la fitotoxicidad en el cultivo (efecto tóxico producido por un compuesto).

Tanto el pH como la conductividad son determinantes a la hora de proveer a las plantas de los nutrientes necesarios, teniendo en cuenta que los rangos adecuados de estos parámetros cambian según la fase de desarrollo de la planta.

Con todos estos parámetros podemos controlar el cultivo de forma más eficiente. Podemos controlar las propiedades del sustrato mediante los sensores de temperatura, pH y EC, modificando dichos parámetros añadiendo los componentes necesarios mediante el uso de bombas peristálticas. Además, con el sensor de CO2 en el aire se puede determinar si es necesario limpiar el aire para bajar su nivel, encendiendo o apagando una turbina de aire desde la Arduino. De forma similar podemos controlar el riego. Dependiendo del nivel de humedad de la tierra, se podrían encender o apagar un sistema de electroválvulas que dejarían pasar (o no) el agua a las distintas zonas del cultivo.

Energía solar como fuente de alimentación

El módulo de sensorización regido por la Arduino consume muy poco, por lo que sería factible conectarlo a una batería la cual se recarga gracias a un panel solar. Así conseguimos que la batería aguante más, y que la placa Arduino tenga una fuente estable de alimentación.

De esta forma podríamos distribuir los módulos de sensorización usando piquetas, clavándolas junto a la zona de cultivo a monitorizar.

Aplicación Android

Durante la realización del TFG se ha trabajado de forma paralela en el desarrollo de un sistema hidropónico para una empresa. En dicho proyecto se ha utilizado una base de datos alojada en un servidor, a la cual se propuso acceder a ella mediante una aplicación Android.

Figura 25: Pantalla de inicio de sesión

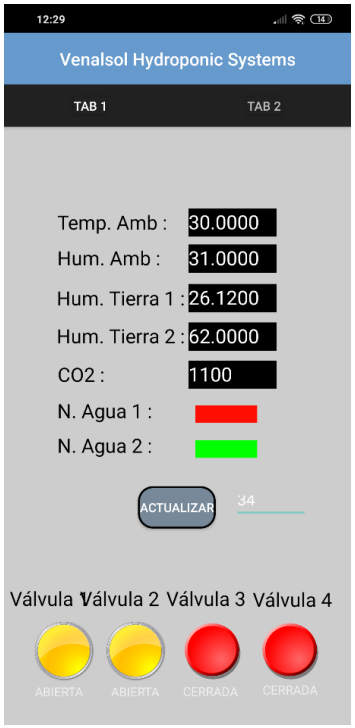

Figura 26: Pantalla módulo 1

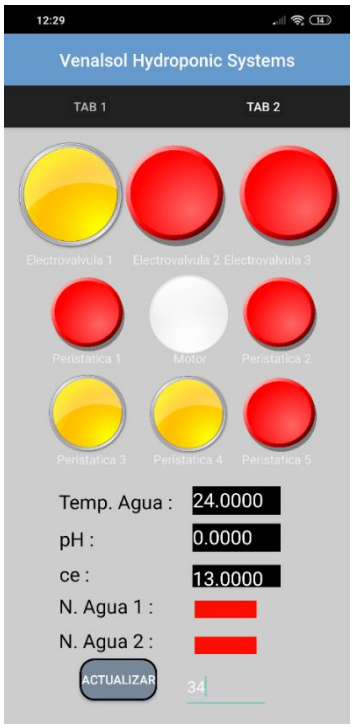

Figura 27: Pantalla módulo 2

De esta forma, se puede acceder de forma sencilla a la base de datos, pudiendo tanto visualizarla como editarla.

Esto se podría añadir al sistema que se ha diseñado en este trabajo, adaptando la aplicación para que se pueda comunicar con la base de datos almacenada dentro de la Orange Pi Zero, y de esta forma poder ver los datos desde un dispositivo móvil.

Código

Arduino

```
#include <Arduino.h>
#include <Wire.h>
#include "Adafruit_SHT31.h"
#include <Adafruit_Sensor.h>
#include "LowPower.h"

Adafruit_SHT31 sht31 = Adafruit_SHT31();

int seco=357;
int humedo=261;

void setup() {

    Serial.begin(9600);

    pinMode(13,OUTPUT);

    if (! sht31.begin(0x44)) {   // Set to 0x45 for alternate i2c addr
        Serial.println("Couldn't find SHT31");
        while (1) delay(1);
    }//if

}//setup()

void loop() {
    LeerEnviar();
    delay(2000);
    Dormir();
}//loop()

float calibracion(int x){
    float obtenidoSoil=(x - seco)*100;
    float constanteSoil=humedo - seco;
    float resultadoSoil = obtenidoSoil / constanteSoil;

    if (resultadoSoil > 100){
        resultadoSoil=100;
    }//if
    if (resultadoSoil < 0){
        resultadoSoil=0;
    }//if

    return resultadoSoil;
}

void Dormir(){
    for (int i = 0 ;  i  <  74 ; i++){
        LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
   }//for
}//Dormir()

void LeerEnviar(){

    float t2 = sht31.readTemperature();
    float h2 = sht31.readHumidity();

    int soilMoistureValue = analogRead(A0);  //put Sensor insert into soil
```

```
58.     float h=calibracion(soilMoistureValue);
59.
60.     Serial.print("Start");
61.     Serial.print("       ");
62.     Serial.print("Hum.ROBOT = "); Serial.print(h); Serial.print("/a");
63.
64.     if (! isnan(t2)) {  // check if 'is not a number'
65.         Serial.print("       ");
66.         Serial.print("Temp SHT31 = "); Serial.print(t2);Serial.print("/b");
67.     } else {
68.         Serial.print("       ");
69.         Serial.println("Failed to read temperature");
70.     }//else
71.
72.     if (! isnan(h2)) {  // check if 'is not a number'
73.         Serial.print("       ");
74.         Serial.print("Hum. SHT31 = "); Serial.print(h2);Serial.print("/c");
75.     } else {
76.         Serial.print("       ");
77.         Serial.println("Failed to read humidity");
78.     }//else
79.     Serial.print("       ");
80.     Serial.print(soilMoistureValue);
81.     Serial.println();
82. }//LeerEnviar()
```

Configuración bluetooth

```
1.  #include <SoftwareSerial.h>   // Incluimos la librería  SoftwareSerial
2.  SoftwareSerial BT(10,11);    // Definimos los pines RX y TX del Arduino conect
    ados al Bluetooth
3.
4.  void setup()
5.  {
6.  Serial.begin(38400);
7.  BT.begin(38400);
8.  }
9.
10. void loop()  {
11.   if (BT.available())  {
12.     Serial.write(BT.read());
13.   }
14.   if (Serial.available()) {
15.     BT.write(Serial.read());
16.  }
17.
18.  }
```

Python

```
1.   #importamos la libreria sql
2.  import sqlite3, serial, time, re
3.  import datetime
4.
5.  arduino = serial.Serial('/dev/ttyS1', 9600)
6.  time.sleep(2)
7.  #Creamos la base de datos
8.  miConexion=sqlite3.connect("BDD tfg")
9.  miCursor=miConexion.cursor()
10.
11. #Creamos las tablas para cada sensor
```

```
12.
13. #Tabla del DFRobot
14. miCursor.execute('''''
15.     CREATE TABLE IF NOT EXISTS MedidaDFRobot (
16.     Sensor varchar(10),
17.     Humedad integer,
18.     Dia varchar(14),
19.     Hora varchar(20))
20. ''')
21. #Tabla del SHT31
22. miCursor.execute('''''
23.     CREATE TABLE IF NOT EXISTS MedidaSHT31 (
24.     Sensor varchar(10),
25.     Temperatura integer,
26.     Humedad integer,
27.     Dia varchar(14),
28.     Hora varchar(20))
29. ''')
30.
31. while True:
32.     x=arduino.readline().strip().decode('ascii')
33.     print(x)
34.     if 'Start' in x:
35.         humRobot = float(re.search('Hum.ROBOT = (.*)/a', x).group(1))
36.         tempSHT31 = float(re.search('Temp SHT31 = (.*)/b', x).group(1))
37.         humSHT31 = float(re.search('Hum. SHT31 = (.*)/c', x).group(1))
38.
39.         fecha=datetime.datetime.now()
40.         hora=str(fecha.hour)+':'+str(fecha.minute)+':'+str(fecha.second)
41.         dia=str(fecha.day)+'-'+str(fecha.month)+'-'+str(fecha.year)
42.
43.         PruebaDFRobot=[
44.             ('DFRobot',humRobot,dia,hora)
45.         ]
46.         PruebaSHT31=[
47.             ('SHT31',tempSHT31,humSHT31,dia,hora)
48.         ]
49.
50. #Insertamos los datos de prueba en la tabla
51.         miCursor.executemany("INSERT INTO MedidaDFRobot VALUES (?,?,?,?)",Prue
    baDFRobot)
52.         miCursor.executemany("INSERT INTO MedidaSHT31 VALUES(?,?,?,?,?)",Prueb
    aSHT31)
53.
54. #Se guardan cambios y se cierra la conexiÃ³n
55.         miConexion.commit()
56. miConexion.close()
```

Gráficas

```
1.  #importamos la libreria sql
2.  import sqlite3, serial, time, re
3.  import matplotlib.pyplot as plt
4.  import matplotlib.dates as mdates
5.  import datetime
6.  import numpy as np
7.
8.  #Creamos la base de datos
9.  miConexion=sqlite3.connect("BDD tfg")
10. miCursor=miConexion.cursor()
11.
```

```
12.
13. fig=plt.figure(1)
14. plt.grid()
15.
16. ax = fig.add_subplot(111)
17. ax.set_title('Temperatura ambiente')
18.
19. fig2=plt.figure(2)
20. plt.grid()
21. ax2 = fig2.add_subplot(111)
22. ax2.set_title('Humedad ambiente')
23.
24. fig3=plt.figure(3)
25. plt.grid()
26. ax3 = fig3.add_subplot(111)
27. ax3.set_title('Humedad en la tierra')
28.
29. miCursor.execute('SELECT * FROM MedidaSHT31')
30. tsht=[]
31. hsht=[]
32. horaSHT=[]
33.
34. for row in miCursor.fetchall():
35.     tsht.append(row[1])
36.     hsht.append(row[2])
37.     horaSHT.append(row[4])
38.
39. miCursor.execute('SELECT * FROM MedidaDFRobot')
40. hdf=[]
41. horadf=[]
42.
43. for row in miCursor.fetchall():
44.     hdf.append(row[1])
45.     horadf.append(row[3])
46.
47. plt.figure(1)
48. plt.plot_date(horaSHT,tsht,'r-*',label='Temperatura(ºC)')
49. fig.autofmt_xdate(rotation=60)
50. ax.set_xticks(np.arange(0,len(horaSHT),5))
51. ax.set_yticks(np.arange(int(tsht[0]/2),(tsht[0]+tsht[0]/2),2))
52. fig.tight_layout()
53. fig.legend()
54.
55.
56. plt.figure(2)
57. plt.plot_date(horaSHT,hsht,'b-*',label='Humedad(%)')
58. fig2.autofmt_xdate(rotation=60)
59. ax2.set_xticks(np.arange(0,len(horaSHT),5))
60. ax2.set_yticks(np.arange(0,100,5))
61. fig2.tight_layout()
62. fig2.legend()
63.
64. plt.figure(3)
65. plt.plot_date(horadf,hdf,'b-*',label='Humedad(%)')
66. fig3.autofmt_xdate(rotation=60)
67. ax3.set_xticks(np.arange(0,len(horadf),5))
68. ax3.set_yticks(np.arange(0,100,5))
69. fig3.tight_layout()
70. fig3.legend()
71.
72. plt.show()
73. miConexion.close()
```

Referencias

[1] Bluetooth. Disponible Web. : https://www.bluetooth.com/

[2] R. Prasad, Pratiksha. :"BLE vs. Wi-Fi for IoT Product Development", IoT Zone, 2018

[3] Arduino. Disponible Web. : https://store.arduino.cc/

[4] Pelegrí Sebastiá,José y Lajara Vizcaino, José Rafael. : "Sistemas Integrados con Arduino", Marcombo, 2013.

[5] Sensor SHT31-D. : https://learn.adafruit.com/adafruit-sht31-d-temperature-and-humidity-sensor-breakout/downloads

[6] Protocolo I2C. : https://es.wikipedia.org/wiki/I%C2%B2C

[7]Sensor SKY SEN0193 de DFRobot : https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193

[8] Protocolo UART. : https://es.wikipedia.org/wiki/Universal_Asynchronous_Receiver-Transmitter

[9] Circuito BOD. : https://www.quora.com/What-is-a-brownout-detector-in-an-AVR-microcontroller

[10] Orange Pi. : http://www.orangepi.org

[11] Comandos Hayes. :https://es.wikipedia.org/wiki/Conjunto_de_comandos_Hayes

[12] HC-05 comandos AT. : https://www.teachmemicro.com/hc-05-bluetooth-command-list/

[13] MatplotLib. Disponible Web. : https://matplotlib.org/

[14] Agricultural IoT: Great importance of CO2 Sensors in Greenhouse. : http://news.isweek.com/105/

[15] A. Barbaro,Lorena, A. Karlanian, Monica y A. Mata, Diego. :"Importancia del pH y la Conductividad Eléctrica (CE) en los sustratos para plantas", Infoagro, 2018

[14] Pallás Areny, R. :"Sensores y Acondicionadores de señal. ", Marcombo, 1994.

[15] Jacob Fraden. : AIP Handbook of Modern Sensors, AIP Press, 1993.

[16] Microchip Technology Inc, PIC19F87XA Datasheet [en líina] [Consulta 12/11/2006] Disponible a:http://www.microchip.com

[17] James Bryant Walt Kester, Linear desing seminar. IEEE Computer Society Press, 1991.

[18] Webster: The Measurement, Instrumentation and sensors Handbook. CRC Pres, 1999.

[14] H. Andaluz, Victor, Y. Tovar, Andrea. :"Automatic control of drip irrigation on Hydroponic agriculture: Daniela tomato production".