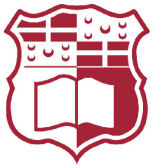

L-Università ta' Malta
Faculty of Information & Communication Technology
Department of Artificial Intelligence

Developing a Remote Plant Monitoring System using IoT

Oleg Grech 0012101(H)

M.Sc. Artificial Intelligence
May 2024

Study-unit: **Internet of Things**
Code: **ICT5101**
Lecturer: **Prof. Ing. Carl James Debono**

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as “the unacknowledged use, as one’s own work, of work of another person, whether or not such work has been published” (Regulations Governing Conduct at Examinations, 1997, Regulation 1 (viii), University of Malta).

I, the undersigned, declare that the assignment submitted is my work, except where acknowledged and referenced.

I understand that the penalties for making a false declaration may include, but are not limited to, loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected, and will be given zero marks.

(N.B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

Oleg Grech
Student Name


Signature

Student Name

Signature

Student Name

Signature

Student Name

Signature

ICT5101
Course Code

Developing a Remote Plant Monitoring System using IoT
Title of work submitted

18/05/2024
Date

Developing a Remote Plant Monitoring System using IoT

Oleg Grech
oleg.grech.19@um.edu.mt

***Abstract*—In a time of rapidly advancing technology, the incorporation of Internet of Things (IoT) applications has transformed several sectors in the world we live in, especially in the agriculture sector. This report explores building a prototype for a remote plant monitoring system using IoT technology. The system uses an Arduino UNO R4 WiFi board to collect data and transmit the data wirelessly, where the obtained readings are displayed on a web interface. The report dives into the system's components, including sensors and communication modules, explaining how each part contributes to the system's reliability and efficiency. Moreover, the testing procedures and results of a week-long data collection period are discussed along with the system's dependability. Improvements like automatic watering, webhook integration, and machine learning are suggested to expand the system's capabilities.**

***Index Terms*—Internet of Things (IoT), Arduino, Remote Plant Monitoring, Soil Moisture Sensor, Web Interface, Agriculture**

Abbreviations and Acronyms

API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IoT	Internet of Things
PoC	Proof of Concept
SSID	Service Set Identifier

I. Introduction

In a time of rapidly advancing technology, the incorporation of Internet of Things (IoT) applications has transformed several sectors in the world we live in [1]. One of the sectors in which IoT has thrived in is the agriculture sector. The convergence of sensor technologies and IoT devices is driving a paradigm change in agriculture as it allows for the adaptation of precision agricultural practices. The applications in which IoT can be utilised within this sector can be divided into four main categories; management systems, including water and agricultural machinery management systems, monitoring systems, including livestock and soil moisture monitoring systems, control systems, including irrigation and water quality control systems and unmanned machinery including automating certain machinery [2]. The main objective of this project is to develop a Proof of Concept (PoC) of a soil moisture monitoring system. Moreover, within this report, the selection of hardware, implementation, testing of PoC and results obtained will be discussed in the following sections.

II. Selection of Hardware

The selection of hardware components is critical to the successful installation of the remote plant monitoring system PoC. Every piece of hardware, from the chosen board to the sensors capturing the readings and also the communication modules that enable data transfer, is essential to maintaining the dependability and efficiency of the system. This section explains the role of the selected hardware components within the PoC, justifying why such hardware was selected. The respective datasheets of the main components used in this PoC can be found at the end of this document.

A. Arduino UNO R4 WiFi Board

The pervasiveness of computers in modern systems has revolutionised efficiency and transformed how we interact with them. The same applies for this PoC where a decision needed to be made on which embedded hardware will be part of the system. The **Arduino UNO R4 WiFi** board was selected as the embedded hardware for this PoC.

The Arduino UNO R4 WiFi[1] is one of the latest boards released by Arduino (as of May 2024) and it integrates the **ESP32-S3** wireless networking module from Espressif[2] with the processing power of the Renesas[3] **RA4M1** microcontroller. This board was a well-suited choice for this PoC due to its ease of use, built-in WiFi capabilities, and robust functionality. The familiar Arduino platform, its large community and also a vast choice of libraries allowed for quick development. The integrated WiFi module eliminated the need for external modules, simplifying the overall design. This board's ability to handle various tasks efficiently made it an ideal solution for achieving the PoC's goals.

B. LM393 Comparator & FC-28

For data collection, two components were utilised in the PoC, the **LM393 Comparator** and the **FC-28 Soil Moisture Sensor**. The LM393 is an integrated circuit comparator, which compares two analog voltage inputs and outputs a digital signal (high or low) based on which voltage is greater. In the context of this PoC, the LM393 was used to convert the analogue signal from the FC-28 soil moisture sensor, into a digital signal that the microcontroller (Arduino Board) can easily understand.

[1] https://store.arduino.cc/products/uno-r4-wifi
[2] https://www.espressif.com/
[3] https://www.renesas.com/us/en

The FC-28 is a type of soil moisture sensor that measures the electrical conductivity of the soil. The FC-28 typically outputs an analogue voltage that reflects the conductivity the sensor has with the soil. As soil moisture decreases, conductivity increases i.e. a higher voltage is produced. This sensor is a great choice for a remote plant monitoring system because it's relatively inexpensive, easy to use, and offers a good balance between sensitivity and durability for long-term monitoring of soil moisture levels.

C. Other Components

In addition to the primary hardware specified above, additional parts were employed to finish the system. The Arduino board is powered by a 10W (5.1V/2.1A) plug. Multiple male-to-male and female-to-female cables allowed us to create the necessary connections between the Arduino board, breadboard and sensor in the required ways. The two connected LEDs—one red and one green—were powered up via the Arduino board through the breadboard. The red LED served as an alert indicator for low soil moisture levels, while the green LED showed that the Arduino board was connected to the wireless network. Every LED had a 220 Ω resistor to prevent the respective LED being fried. These additional components allowed for a more comprehensive and integrated system.

Figure 1 depicts the setup of the PoC system showcasing the aforementioned hardware components for the remote plant monitoring system.

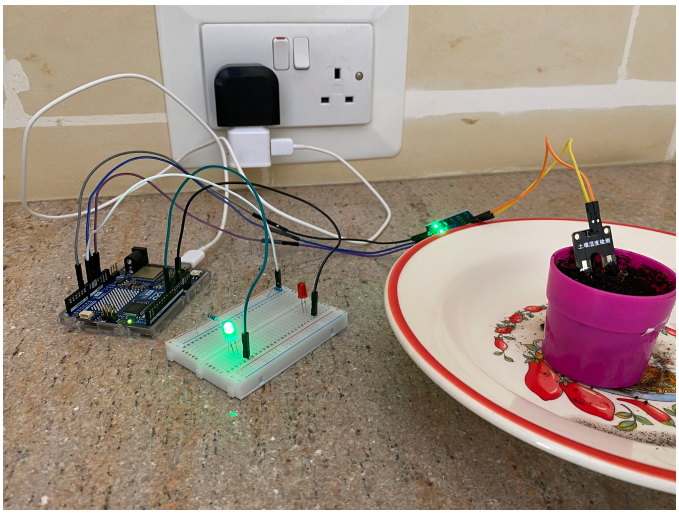

Fig. 1. Remote Plant Monitoring System Setup

III. Implementation

This section outlines the implementation of the proposed remote plant monitoring system. The PoC system comprises two main components, the embedded system code development for the Arduino UNO R4 WiFi board, responsible for data acquisition from sensors and wireless communication, and the development of a web interface for remote data visualisation and user interaction. A basic representation of the PoC implementation is depicted in Figure 2. A deeper understanding of the embedded system and web application will be detailed within this section.

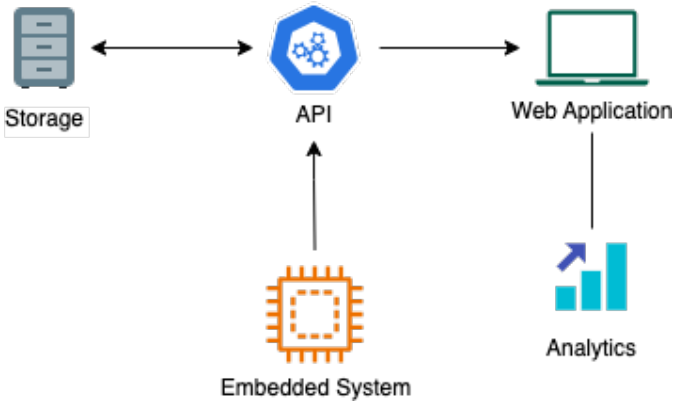

Fig. 2. Basic representation of Proof of Concept (PoC)

A. Embedded System

The embedded system is responsible for the collection of sensor data and remote communication. The Arduino UNO R4 WiFi board serves as the core of the system, interfacing with sensors and facilitating wireless data transmission.

A variant of the C/C++ programming language is the development language compatible with the Arduino board. To make the development experience a better one, the Arduino company, along with other software products, provides developers with their own Integrated Development Environment (IDE) called **Arduino IDE**[4]. This IDE is designed specifically for Arduino boards, offering a streamlined development environment with pre-built libraries for common functionalities like interacting with sensors and controlling hardware components. Moreover, the Arduino IDE boasts a large and active user community, providing a wealth of online tutorials, libraries, and forums for troubleshooting and project inspiration. This extensive support system makes it easier to overcome challenges and accelerate the development process.

For the remote plant monitor system implementation, we use three libraries, **WiFiS3** (following a similar Application Programming Interface (API) to that of the WiFiNINA library [3]), **WiFiSSLClient** [4] and **ArduinoHTTPClient** [5]. All three libraries played a pivotal part in the remote data communication aspect of the implementation. The setup part starts with initialising several variables including the WiFi's SSID and password, hostname, HTTP port and WiFi client. The values for the SSID, password and hostname are being retrieved from a *secrets* file to not expose sensitive information in the main code base. Furthermore, in the setup function, we attempt to connect to the passed SSID and also initialise the pins on the board to which an output signal will be given in the loop function.

The loop function starts off with checking whether or not the WiFi module is still connected to the SSID. If it is connected, the green LED is turned on to indicate that there is a connection and vice-versa, it is turned off if we do not have a connection. Once a connection is established, we initialise the endpoint to which we will be sending data and also the

[4]https://www.arduino.cc/en/software

content type. The sensor reading is read in an analogue manner to get the moisture level and we add this value to the body request of the endpoint that will be triggered. This reading is also checked to see if it is greater than a value of 625 (value inspired by the code developed in [6]), and if so, the red LED is turned on indicating that the moisture is low and the plant needs re-watering. Then, the HTTP client is initialised and we attempt to hit the previously initialised endpoint. If the response status code received is not equal to 201 and we have not reached the number of retries, we attempt to hit the endpoint again. When a successful response status code is received we blink the green LED to indicate that the endpoint was hit successfully and the data was added to the database. Finally, the loop function is delayed for one hour before going through this flow again.

B. Web Application

The web application for this PoC was developed to provide an interface for a user to keep track of the moisture levels remotely. The application was built using Next.js[5], which is a React framework built on top of React with other features such as routing, data fetching and caching. This choice of framework was a no-brainer as it allowed for the development of both the user application and the necessary endpoints to store and read data obtained from the Arduino board. A more in-depth look at the development of both the endpoints and the user application will be discussed within this section. A database was also required for the completeness of the whole web application and thus a PostgreSQL database was set up. The database only contains one table used to read and write sensor readings from and to it. Both the web application and the database were hosted on Vercel[6], which is a cloud platform that allows for easy deployment of your web applications and also offers serverless SQL.

1) Endpoints: Two endpoints were required for this PoC. The first endpoint, `/api/add-sensor-reading`, is a `POST` request which adds the passed-in value within the body request to be stored within the database table containing the sensor readings. This endpoint is triggered from the Arduino board whenever a new sensor reading is read from the sensor. The second and final endpoint, `/api/get-sensor-readings`, is a `GET` request that retrieves all the sensor readings collected from the database table and returns the list in an array of objects. This endpoint is used in the user application to retrieve and display the obtained results in a user-friendly manner.

2) User Application: The user application is a simple single-page application which contains all the necessary functionality and display of data. The page has four main components, a date picker, a warning alert, the average reading and a line chart. The date picker, average reading and line chart are all connected components. This means that when a date is selected from the date picker, the average reading and line chart reflect the data from that user-selected date. By default, the line chart shows all the sensor readings and the average of all the sensor readings. The warning alert component is displayed on the screen whenever the latest reading indicates that the moisture level is low. This allows the user who is using the web application to be aware that the plant needs re-watering without the need to be next to the embedded system. Figure 3 shows a snippet of how the web application looks.

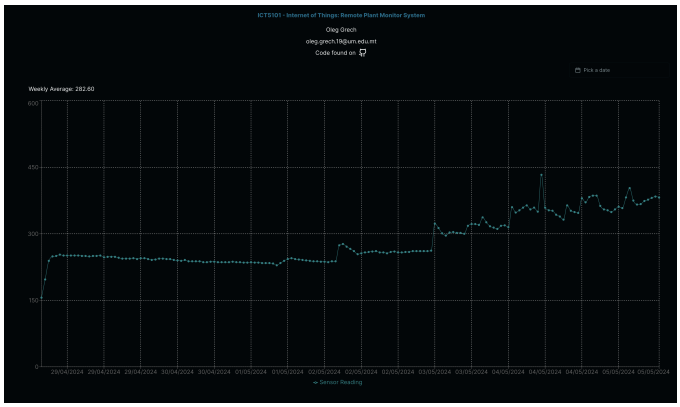

Fig. 3. Web Application

The web application can be found on https://ict-5101-iot-remote-plant-monitor-system.vercel.app/.
The embedded system and web application implementations can be found on GitHub using this link: https://github.com/oleggrech7/ICT5101-IOT-Remote-Plant-Monitor-System.

IV. Testing & Results

For any form of system, a testing process is done to evaluate the systems' functionality and performance. The same applies for this PoC, in which the system's efficacy is assessed in order to guarantee that the data it collects is accurate and useful. This section details the testing procedures and results obtained after a one-week period of data collection with the developed PoC.

The system setup is already depicted in Figure 1. Some of the decisions and/or actions that were taken prior the testing phase include:

- *The use of a very small pot, having a similar height to the FC-28 moisture sensor*
  Since the sensor relies on conductivity, ideally the sensor is fully submerged into the main zone for accurate readings. Using a small pot ensures that the sensor is completely surrounded by the soil. This would also help in strategic placement of the sensor if it were to be scaled up for future experiments.
- *The soil was watered*
  Watering the soil prior to testing allowed us to have a baseline measurement which allows for better tracking of the sensor's reading as the soil dries over time. If the soil was left in its original state, it would have been more difficult to streamline how the sensor would react over time.
- *The pot was placed inside without having direct sunlight*
  By doing this, a more controlled environment for testing is created. This consistency minimises external factors

[5]https://nextjs.org/
[6]https://vercel.com/

which could hinder sensor readings and data interpretation. This would also benefit future testing which can involve direct sunlight having this baseline condition.

With respect to the results obtained over one week (seen in Figure 3 and can be accessed on the web application itself), one can observe that as the days went by the value increased, meaning that there was a decrease in moisture. This is so because as moisture decreases, the volatility increases thus the slope of the graph is always increasing. One can notice that there was a sudden increase in volatility, in the first few hours into the experiment on the 28th of April, 2024. This means that there was a sudden decrease in moisture. Up until the 2nd of May, 2024, the readings obtained were relatively equal to each other, meaning that the soil kept a constant moisture level. On the 2nd of May itself, the chart indicates that there was a decrease in moisture as the volatility went up. As the days progressed until the end of the experiment, the moisture levels continue to decrease as we continue to see an increase in volatility. These obtained results were expected as typically as time goes by moisture tends to decrease and the PoC performed as expected.

V. Conclusion & Future Improvements

The remote plant monitoring Proof of Concept (PoC) was successfully designed and implemented using an embedded system via an Arduino Uno R4 WiFi board and a suitable web application built with Next.js. The testing of this PoC confirmed that the embedded system was able to acquire data via the FC-28 sensor and transmit such data via the ESP32-S3 module. The PoC also showed a high level of reliability with no downtime at all during a week of testing.

Like any other form of system, there is always room for improvement. Some of these improvements could include:

- *Auto-water dispensing*
  Rather on depending on people to water the plant itself, when low moisture is detected, we can update the embedded system implementation to dispense water accordingly.
- *Webhook Integration*
  Developing a webhook would allow us to get a reading from the embedded system in real-time rather than having to wait for an hour to pass before we can get a new reading.
- *Machine Learning Integration*
  Integrating machine learning into the system would have been useful to predict when the plant needs watering based on the sensor data and historical trends.

Overall, this PoC was a good starting point for one who would like to scale up and attempt to have a remote plant monitoring system on a larger scale.

References

[1] L. D. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6714496/

[2] W.-S. Kim, W.-S. Lee, and Y.-J. Kim, "A Review of the Applications of the Internet of Things (IoT) for Agricultural Automation," *Journal of Biosystems Engineering*, vol. 45, no. 4, pp. 385–400, Dec. 2020. [Online]. Available: https://doi.org/10.1007/s42853-020-00078-3

[3] "WiFiNINA - Arduino Reference." [Online]. Available: https://www.arduino.cc/reference/en/libraries/wifinina/

[4] "WiFiNINA - WiFiSSLClient - Arduino Reference." [Online]. Available: https://www.arduino.cc/reference/en/libraries/wifinina/wifisslclient/

[5] "ArduinoHttpClient - Arduino Reference." [Online]. Available: https://www.arduino.cc/reference/en/libraries/arduinohttpclient/

[6] "Hello-tech/Soil_moisture_sensor_with_relay___lcd.ino at master · passion-tech/Hello-tech." [Online]. Available: https://github.com/passion-tech/Hello-tech/blob/master/Soil_moisture_Sensor_with_Relay___Lcd.ino