Final Project for 21VT- 4ME307- Internets Architecture

Rema Salman
School of Computer Science, Physics and Mathematics, Linnaeus University, Sweden
rs223bc@student.lnu.se

Abstract. In this report, I designed and developed an IoT backyard prototype, which concerns and explores the sector of agriculture, specifically, smart plant monitoring and automatic irrigation. Different technologies and methods were investigated from the domains of Internet Architecture (IA) and the Internet of Things (IoT). The solution consisted of virtually collecting the readings of soil moisture sensors for two fields, a weather station, and two water pumps (actuator for each field) but the system can still be applicable for scaling.

Keywords: Internet of Things, system architecture, plant monitoring, smart irrigation.

1. INTRODUCTION AND BACKGROUND

Many studies have documented the positive effects that plants have on humans, apart from being materially and economically complementary to the needs of humans [1]. Some of these benefits contribute to improving the mental and physical health of humans. According to the United Nations, sustainable development, even in households, improves human lives and protects the environment. Currently, individuals lean towards sustainable farming at their house gardens to be self-sufficient and not rely on produce from the agriculture or food industry.

The rapid growth of technology in the world is more available than ever in many respects, including embedded controllers, database structure, data cloud collection, big data, data mining, machine learning, back-end data processing, HTML standard, front-end technology, data visualization, etc. This growth led everything to become controlled and operated automatically. More and more information technology applications are deployed for monitoring and controlling purposes in different sectors. Agriculture is one of the sectors that use new technologies to automate plant monitoring and weather conditions to directly or indirectly govern plant growth and optimize the captivation procedure. Automation refers to the control process of "industrial machinery and processes, thereby replacing human operators" [2]. Table 1 summarizes some of the previous works that use IoT in agriculture.

Table 1. Related IoT-work for plant monitoring and irrigation controlling.

Paper Title & Reference	Physical Devices	IoT Gateway	Communication	User Interface	Findings
An IoT-Based Smart Plant Monitoring System. [2]	-Moisture sensors -Ultrasonic distance sensors to monitor the water level in water tank	ESP8266, Arduino	Wi-Fi (MQTT)	Android App	Trying to overcome drawbacks of monitoring technologies, e.g., GSM, Zigbee and Bluetooth.
Enhancement of plant monitoring using IoT. [3]	-Soil monitoring sensor (moisture sensor). -Light sensor -Temperature sensor	Arduino, GSM	Not specified	Not specified	The system provided better results to monitor plants, increase plant growth, and better use of water in comparison with traditional methods.
IOT based smart irrigation system. [4]	-Moisture sensors. - Motor for water tank	Arduino, GSM	Traditional Cellular (3G)	Things Speak & web page	The system is a potential solution to the problems faced by the manual process of irrigation, which enables the efficient utilization of water resources.

The authors [5] have proposed an IoT based assessment system for automating water quality. Their proposed system includes several sensors for collecting physical and chemical water properties, being turbidity, pH, temperature, and TDS sensors. An Arduino board was used to read the sensor data and transmit them to GSM via wireless sensor network technology. The remote monitoring can be viewed either in Things Speak or accessed on any Android platform. The authors found that they can collect the data from any location, which depends on where the nodes and sensors are installed at. As a future recommendation, the study has proposed providing more options to users for monitoring remotely the water quality data in real-time.

In [6], the authors developed a system that consists of four parts: back-end Google Firebase real-time database, front-end both SPA web application and mobile monitoring application, controller for both software-hardware, and intelligence server that support MQTT connection and condition control. The authors claim that the system can receive data and allow users to set conditional controls for remote controlling and monitoring.

2. PROJECT DESCRIPTION

2.1. Project Summary

As described in the introduction, there is recently an increased attention towards sustainable farming, which can be used to produce food to be self-sufficient or independent from the agriculture or food industry (see Figure 1).

Figure 1. Backyard sustainable garden with different crop types.

Such sustainable gardening, as any farming, may come with several challenges, which were extracted from[1-4]:

- Individuals may lack knowledge/experience in plant cultivation.
- Absence of regular plant monitoring, or human error while observing the plants.
- Manual irrigation may be an exhausting process.
- Weather conditions are generally unpredictable if there are no technologies capturing weather forecasts.
- Water wastage, because of the inefficient use of water resources.

In this project, I explore the use of new technologies with the combination of standard IoT architecture to build a new prototyped system. This prototype is expected to be used for monitoring backyard fields and automate the irrigation process of these fields based on pre-specified thresholds. Meanwhile, the proposed system could be more secure and reliable, while easier to develop and maintain.

2.2. Project Objectives

I will gain and expand my knowledge and practical experience regarding new technologies by researching and implementing an IA and IoT solution for monitoring and automatically irrigating the plants. Compiling my solution to overcoming the listed challenges above, while focusing on sustainable gardening.

2.3. Expected Results

IoT-based solutions is a rapidly growing development, especially in the agriculture sector, research has been conducted regarding plants monitoring [1, 2] and smart irrigation systems [3, 4]. However, these papers cover the topic from a wider and generic perspective of farming rather than focusing on household gardens. Thus, I will explore the topic of backyard plant monitoring to answer the following research questions (RQ).

RQ1: How can I design and develop an IoT based solution for plant monitoring?
RQ2: How can the data obtained from RQ1 be effectively visualized in a Web application?
RQ3: How can the irrigation process be automated based on the outcome from the proposed solution in RQ1?

2.4 Use Cases

As presented in the use case diagram, Figure 2, the web application will allow users to be able to select which plant field they would like to monitor. The user can monitor the plants' state in real-time in their backyard gardens, including the soil moisture level, while viewing historical soil moisture levels. Also, the user can view additional weather conditions (as explained in subsection 3.1). The user can check the condition of the automated irrigation system, which will also be updated in real-time (when the irrigation process starts and finishes).

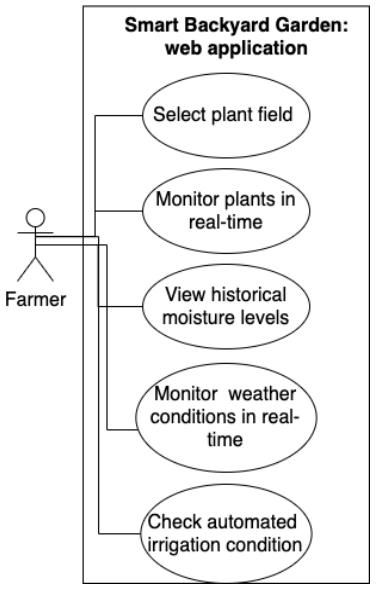


Figure 2. Use case diagram of the proposed web app

3. PROPOSED SOLUTION AND PROTOTYPE

3.1. Solution Summary

The prototype is expected to be used for monitoring two backyard fields and automate their irrigation process based on pre-specified thresholds. Beside answering the RQs, the IoT backyard system is designed to achieve the following requirements:
1) Cloud computing architecture: the system is flexible to load demand.
2) New front-end web technologies (React.js Framework), since web technologies have a big impact on IoT development.
3) Real-time monitoring: any web browser on any device could be used for real-time monitoring.
4) System security: SSL encryption of data while transmission.
5) Control rules: provides a conditional control service.

3.2. Methodology

3.2.1 Tools and Components

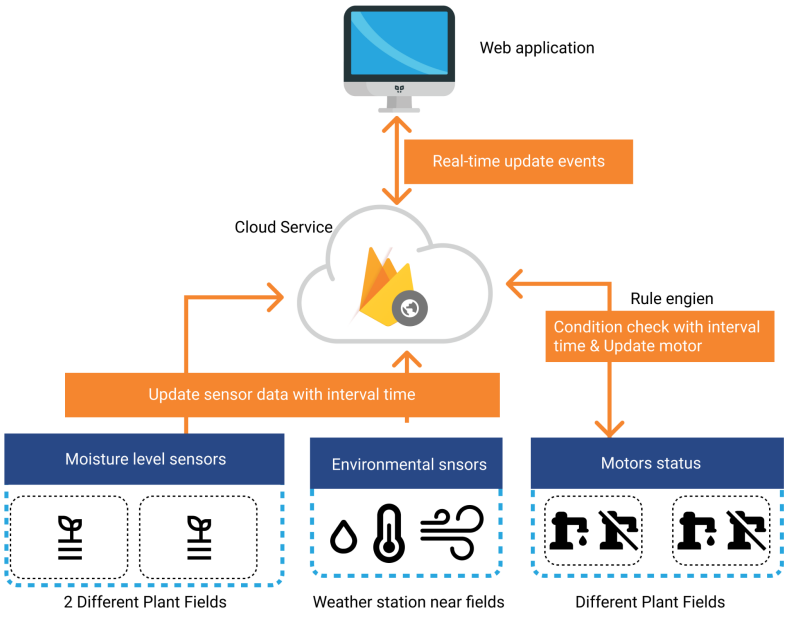


Figure 3. Abstract architecture of the proposed system.

To achieve the project objectives, I designed the system structure based on the three layers of standard IoT systems architecture, as displayed in Figure 3, focusing on the components as follows:

1. **Web application/Presentation layer:**
   An essential component of any IoT architecture is data visualization and management. In the proposed system, they are mainly through the browser and accessible via either the Firebase dashboard or the developed serverless web application. The application was built using web technology (HTML, CSS, JavaScript), developed on React.js[1] framework (javascript framework for front-end development). Figure 4 illustrates the components used to develop the application, all of them were installed using Node Package Manager (NPM[2]). React-bootstrap provides a better user interface, while chartjs[3] was used for visualizing the data through interactive line graphs. Momentjs[4] was used to provide more user-friendly dates. Most importantly, Firebase SDK was used to access the cloud from the web application environment and request the data in real-time.

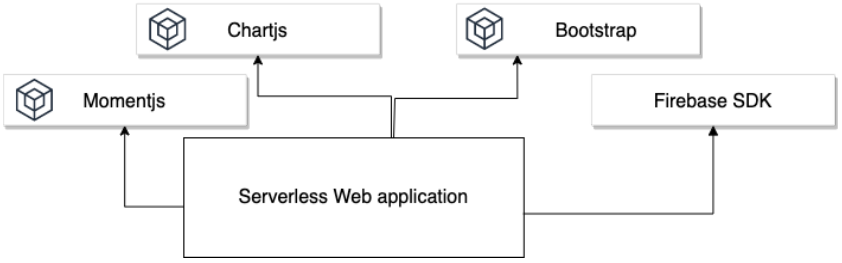


Figure 4. Components of the web application.

2. **Firebase cloud system/ Service layer:**
   Google Firebase provides many services in the cloud, including instant messaging, user authentication, real-time database, storage, hosting, etc. In this project, Firebase real-time database is used to store changing data (with a data precision) and to provide the necessary cloud computation. Firebase cloud system also has SSL encryption for data transmission, which ensures the system's security.

3. **IoT application/ Perception, Network, and Application layers:**
   Node.js application was built to simulate the IoT devices (data generating/transferring), the Firebase connection, and a condition control Rule Engine, see Figure 5. Each of these components is briefly described in the following sections.

   a. **Virtual sensors and data collection:** Assuming the farm includes 2 fields with their required irrigation for growing the crops. Different sensors (soil moisture and environmental) were virtually created to simulate the data collection from these fields, which were randomly generated data. The soil moisture sensor readings are random percentage (%) for both fields with a date (data collection timestamp). The environmental sensors were assumed to be connected to a node (weather station) and installed near the fields. This station collected: Air temperature (C), Air humidity (%), Wind intensity (Km/h), Wind direction (Deg), Rain height (Mm/h), Date (representing a timestamp of their creation/collection). These virtual IoT devices transmit the data in five-minute intervals.

   b. **Conditions control Rule Engine:** The proposed system provides a condition control [6] to enhance the system intelligence, where certain conditions are set and if they are met, actions are accordingly executed. This feature was built to be executed with a six-minute interval in the Node application. It communicates with Firebase real-time database, reads the soil moisture, checks whether the latest values meet the set thresholds, and updates accordingly the water pump/motor status (ON/OFF) on the database to be distributed on the web application. The defined thresholds and type of fildes are shown in Table 2.

[1] React – A JavaScript library for building user interfaces
[2] npm
[3] Chart.js | Open source HTML5 Charts for your website
[4] Moment.js | Home

Table2. Assumptions of the field types and their set soil-moisture thresholds.

Filed	***Type***	***Threshold for soil-moisture level***
Field 1	Maize, less water irrigation	30%
Field 2	Peanuts, normal irrigation	50%

c. **Firebase:** One of the main functions of the IoT is data collection. Considering Firebase Cloud, virtual nodes/devices generated the corresponding data and were directly connected to the database to read and write data. Firebase SDK is used in the Node environment and represents the connection layer of the applications, where data is exchanged between the layers through REST APIs (general network services) via HTTP protocols. All of these connections are encrypted by environmental files, following the best practices.

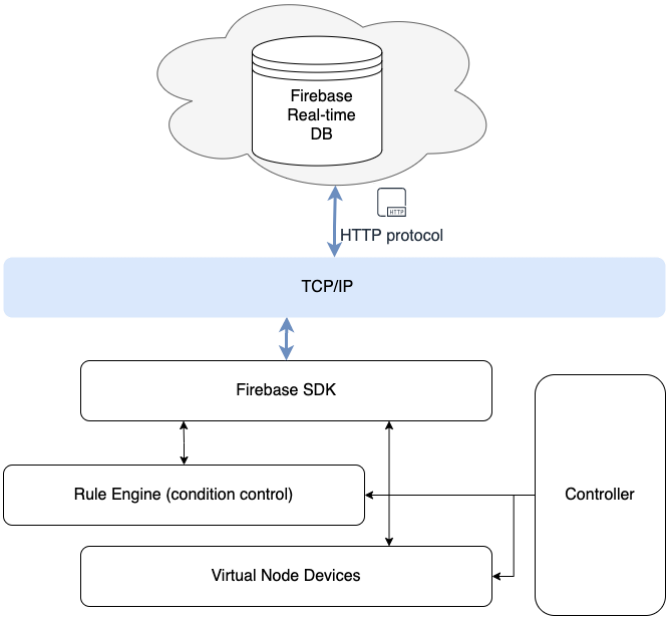


Figure 5. Components of the IoT application.

3.2.2 Prototype Design Details

1. **Data Visualization:** Traditional web service consists of a three-tier architecture, front-end web pages (HTML, CSS, JavaScript), an intermediate web server, and a database. However, the proposed system is a serverless web application, being a two-tier architecture. As represented in Figure 6, the application establishes a connection with the Firebase, reads the data and displays it, using the different libraries mentioned earlier in 3.2.1.

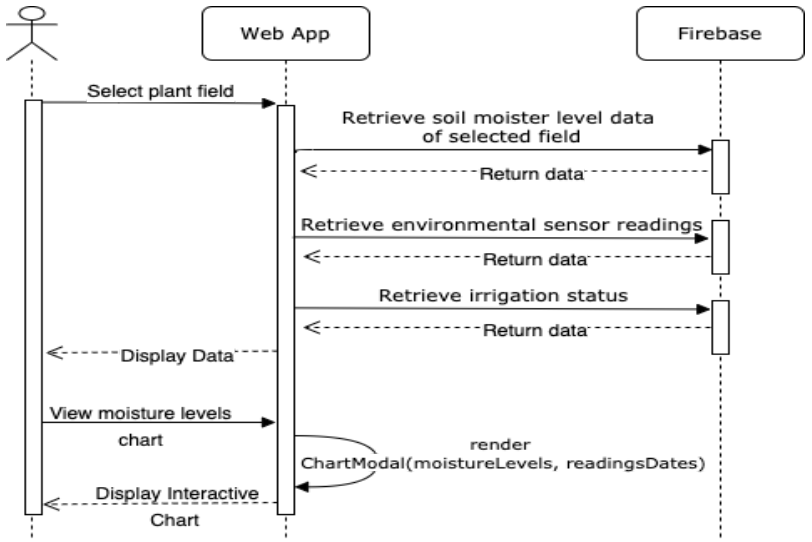


Figure 6. Behaviour diagram of the web application.

2. **IoT application:** Figure 7 shows the logical execution of the IoT application, while further implementation details are presented in the upcoming next sections.

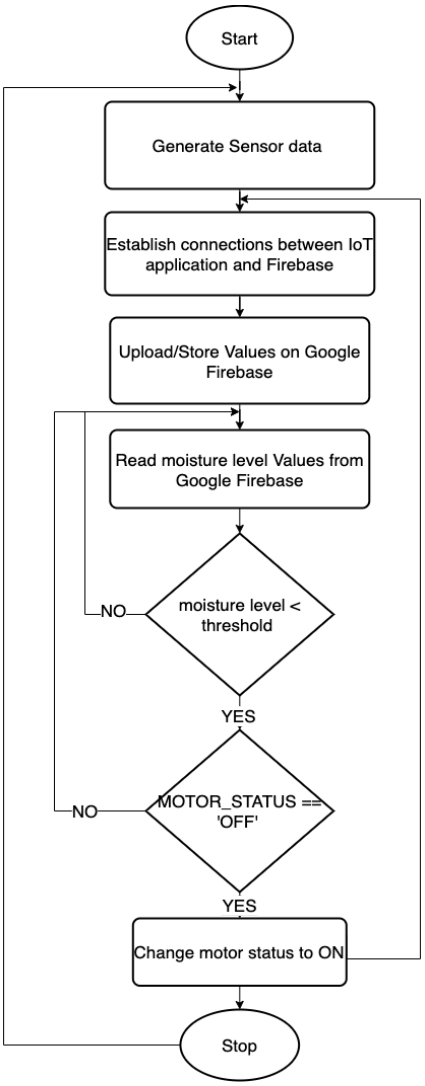


Figure 7. Flowchart for the IoT application of the system

a. Main Controller: As presented in Figure 8, the main controller evokes the weather and field stations to generate, establish the connection and post the values to Firebase. Meanwhile, it executes the *checkMoistureAndWater* function with an interval of 6 minutes.

```
const weather_station =
require("./node_devices/weather_station");
const fields_station =
require("./node_devices/fields_station");
const rule_engine = require("./rule_engine");
weather_station.fetchWeatherData();
fields_station.fetchfieldsData();
setInterval(rule_engine.checkMoistureAndWater,
360000); //360000 milliseconds == 6 minutes
```

Figure 8. Code snippet of the main controller.

b. Data Generation: For generating the virtual sensor readings, the function *getRndInteger* is used where it takes two arguments and returns a random integer between the passed argument numbers, see Figure 9. After, a JSON object is created to be stored in Firebase DB.

```
setInterval(function () {
// Function to generate random values, then sends
them to the cloud platform
    function getRndInteger(min, max) {
     return Math.floor(Math.random() * (max -
min)) + min;}
   const fieldsData = {
     field_1: {
       soil_moisture: getRndInteger(0, 100),
     },
     field_2: {
       soil_moisture: getRndInteger(0, 100),
     },
     date: new Date().toISOString(),
   };
   //  Publish data to cloud the weather_station
   FirebaseDB.ref("fields_station")
     .push()
     .set(fieldsData)
     .catch(function (error) {.....
     });
 }, 300000);
```

Figure 9. Code snippet of the virtual sensor readings (soil moisture sensors) .

c. Rule engine: it is responsible for the conditional control logic (in Figure 7) and explained earlier in this subsection. The function (*checkMoistureAndWater*) implementation is shown in Figure 10. It reads the soil moisture levels and stores the readings in a temporary list (A), reset both motors (B), checks the last reading with the thresholds for each field and accordingly changes the motor status of the field (C).

```
function checkMoistureAndWater() {
FirebaseDB.ref("fields_station").on("value", function (snapshot) {
  const soil_moisture_list = [];
  const levels = snapshot.val();
  for (i in snapshot.val()) {
    soil_moisture_list.push(levels[i]);
    console.log(soil_moisture_list);
  }                                                              (A)
  // set both motors to off before the check
  resetIrrigationStatus();                                       (B)
  //--------------- Low moisture of field One ---------------
  if (
    soil_moisture_list[soil_moisture_list.length - 1].field_1.soil_moisture <=
    30
  ) {
    changeIrrigationStatus(FirebaseDB.ref("irrigation_status/motor_one"));

  //--------------- Low moisture of field two ---------------
  if (
    soil_moisture_list[soil_moisture_list.length - 1].field_2.soil_moisture <=
    50
  ) {
    fields_station.changeIrrigationStatus(
      FirebaseDB.ref("irrigation_status/motor_two")
    );                                                           (C)
     }
});
```

Figure 10. Code snippet of the condition control for updating the irrigation status.

3. Firebase Configuration & Connection:

In order to read/write data from/to firebase, the Firebase SDK needed to be installed in both system applications front-end (react) and IoT (node). This was done by using the NPM package following this command: `npm install firebase`. As presented in Figure 11, to configure and connect to the created database the following code was written, where the database instance was used anytime the connection needed to be established. Following the best programming practices, Firebase credentials were stored in environment files to ensure the system's security and prevent any damage to the credentials, which may affect the connection.

```
// configure firebase
const firebaseConfig = firebase.initializeApp({
 apiKey: process.env.API_KEY,
 authDomain: process.env.AUTH_DOMAIN,
 databaseURL: process.env.DATABASEURL,
 projectId: process.env.PROJECTID,
 storageBucket: process.env.STORAGEBUCKET,
 messagingSenderId: process.env.MESSAGINGSENDERID,
 appId: process.env.APPID,
});

// initialise a create an instance
const database = firebaseConfig.database();
```

Figure 11. Code snippet of the Firebase configuration.

3.2.3 Experiments Execution

The prototyped system for the IoT backyard plants monitoring and control actuators (virtual water pump) included a real-time Firebase database, IoT application, and a web application. In order to ensure that they work properly as intended. Firstly, the applications were separately executed, while the cloud was observed. After that, the applications were run simultaneously to ensure conflict absence and to test the real-time monitoring and the cloud computing features. The results from the experiments are presented in the next section.

3.2.4 Results and Analysis

1. **IoT applications:**

   a. **Virtual sensors and data collection:** Considering that Firebase is a NoSQL database, the data (in the form of objects) were pushed to their corresponding directions in Firebase. The data for both stations were handled as lists, where Firebase generates a unique key for each pushed data entry by default, as shown in Figure 12.

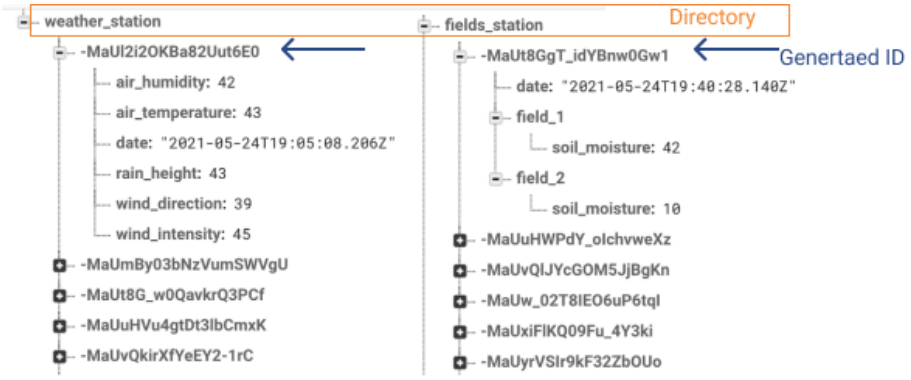
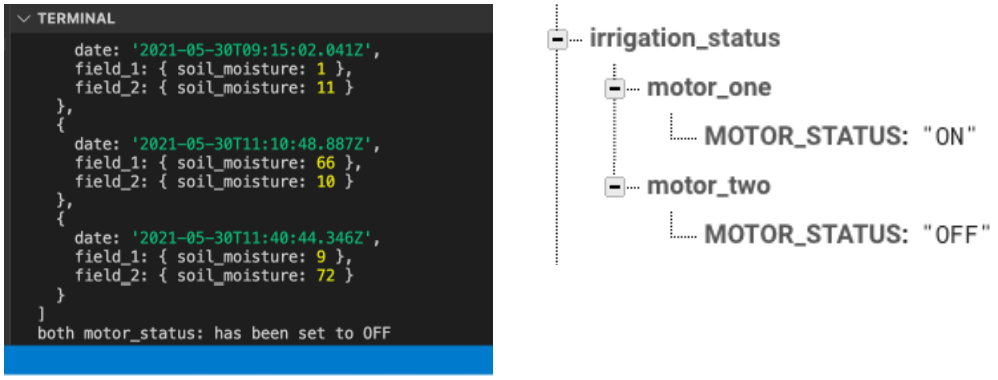


Figure 12. Screenshot of the Firebase console.

   b. **Conditions control/Rule Engine:** The moisture levels are stored in a list and both motors are reset (turned off) when executing this function, see Figure 13 (A). After that, the motor status is automatically changed according to the last level in comparison with the thresholds. In the case in Figure 13, the level of the first field was below the threshold (30% for field one); therefore, the motor status has changed, see (B).



(A) Latest readings and reset (B) Changed irrigation status of motor_one

Figure 13. Results of the Rule Engine execution.

2. **Web applications:** the field selection is the default page when running the web application, where the user can select which field they want to monitor, see Figure 14.

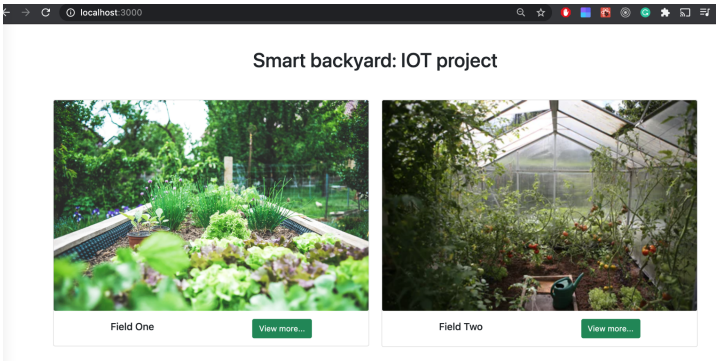
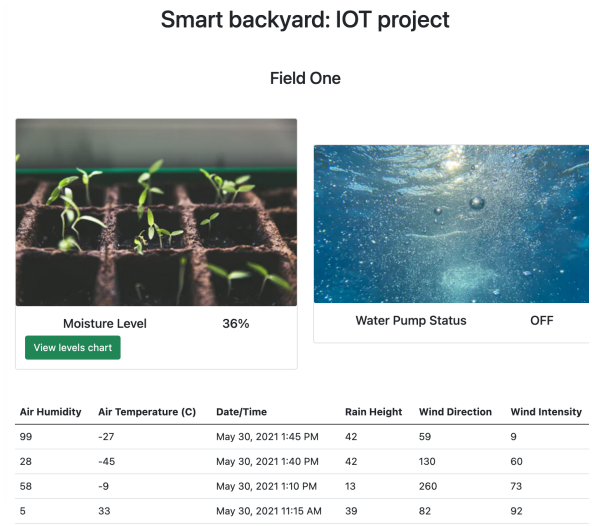
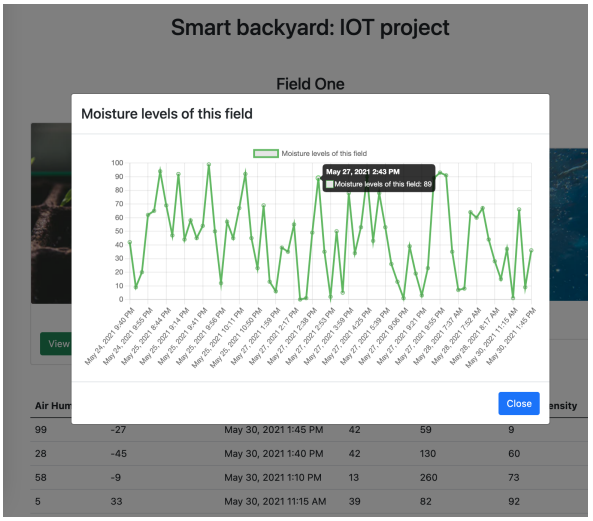


Figure 14. Results of Launching the web application.

a. **Data Visualization:** When selecting to view Field One, the data was dynamically requested from Firebase and displayed in different ways. The environmental readings were displayed in a table and sorted from the latest readings to the earliest, see Figure 15 (A). Meanwhile, the last moisture level and the irrigation status were displayed in cards. An overlying modal with an interactive chart of all of the historical moisture levels was displayed as soon as the button (view levels chart) Figure 15 (B).

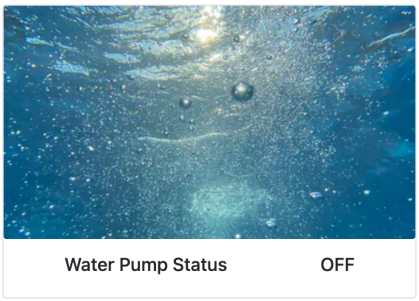




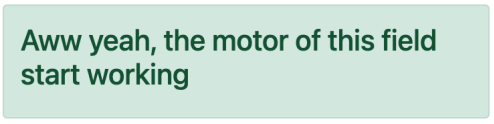
(A) Data visualization. *(B) Chart of the historical moisture levels readings.*

Figure 15. Results of data visualization for Field One.

b. **Irrigation status change:** As shown in Figure 16, whenever the irrigation status changes on Firebase it is dynamically displayed on the web application.



(A)



(B)

Figure 16. Different states of the irrigation status. (A) turned OFF and (B) turned ON.

3.3. Detailed Discussions

RQ1: How can I design and develop an IoT based solution for plant monitoring?
Considering the time frame of this project, it prevented the use of hands-on hardware. Therefore, the design consisted of the standard three-layered IoT systems architecture. Firstly, a presentation layer included a web monitoring application. Secondly, a cloud service to store the data, provide cloud-computing services and allow real-time data transmission. Thirdly, an IoT application which simulates the perception layer (hardware/sensors) and their connection to publish their readings to a cloud service. This IoT application uses a Node environment to substitute the sensors with virtual ones and the data collection with generated datasets based on real-world examples.
The proposed prototype considered the following readings: 1) soil moisture sensors for two fields, 2) a weather station that was assumed to be placed near the fields, and 3) two water pumps (actuator for each field) but the system can still be applicable for scaling, e.g. increasing the number of fields or adding more things (sensors and their readings). The datasets were randomly generated, where the soil moisture levels presented a percentage (%) reading for both fields with a date (data collection timestamp). The environmental sensors were assumed to be connected to a node (weather station), which collected: Air temperature (C), Air humidity (%), Wind intensity (Km/h), Wind direction (Deg), Rain height (Mm/h), Date (representing a timestamp of their creation/collection). These virtual IoT devices transmit the data in five-minute intervals.

Another pitfall that came with not having hands-on hardware is the absence of exploring the usual MQTT connection or any power-efficient and low-range wireless communicatively. Instead, WIFI with TCP/IP was the main connectivity method used during this project, where HTTP protocol was used for transmitting the data to/from the cloud service whether applications (IoT or web). However, the simulation of the connectivity and data consumption can be replaced by an Arduino controller, a NodeMCU based on ESP8266 Arduino (with a build-in WIFI), and real sensors and accelerator (water pumps/motors) but this will not affect the logical implementation of the IoT application neither the results.
Firebase SDK was integrated with both applications to ensure a secure connection and configuration to the correct cloud service as it also has SSL encryption for the transmitted data. Considering that Firebase real-time database is a NoSQL database, data precision is considered through the directories that the readings were published/pushed to and the formation of the objects. To ensure better performance, object nesting was avoided as much as possible.

RQ2: How can the data obtained from RQ1 be effectively visualized in a Web application?
In order to visualize the data obtained from RQ1, a web application was developed using a new front-end technology (React framework). The application was designed to provide a user-friendly GUI with real-time monitoring of the data obtained from the sensor nodes. In the first step, the user needs to choose which field is desired to be monitored. The second step is to view the different collected data, where the environmental readings are presented in a table and sorted from the latest to the earliest. The water pump status and the current soil moisture level are virtualized within cards, while the historical moisture levels are presented in an interactive chart with the time-stamp and level for each reading.

RQ3: How can the irrigation process be automated based on the outcome from the proposed solution in RQ1?
As explained in RQ1 the generated data through the IoT application is published to the cloud service (Firebase), which are used within the *Rule Engine* (conditions controls), as explained in [6]. This *Rule Engine* enhances the system intelligence, where certain conditions are set and if they are met, actions are accordingly executed. This feature was built to be executed with a six-minute interval in the IoT application. It communicates with Firebase real-time database, reads the soil moisture, stores the readings in a temporary list, reset both motors to ensure the irrigation process, checks whether the latest values meet the set thresholds (30% for field one and 50% for field two), and updates accordingly the water pump/motor status on the database to be distributed on the web application.

4. LIMITATIONS AND CHALLENGES

- The time frame of this project affected the use of a simulator or real sensors. To mitigate this, the IoT part of the system was substituted with an IoT application that generates and connects with a cloud service instead. However, the logical implementation will not need any changes even if the virtual sensors were replaced with real sensors.
- Concerning the data and the field, their design was based on assumptions rather than a real-world example. For example, the threshold needed for the plant and the number of fields.
- All outcomes of the web application relied on the dataset generated by the IoT application; therefore, there is no assurance of the data source and accuracy apart from them being an example of real-world sensor readings.

- Another threat of validity is the absence of any user testing and evaluation. To mitigate that, I performed system testing, focusing on usability and system performance aspects rather than the user experience.
- The developed applications are not deployed but the links (a video demo and source code) as part of the project deliverables, see Sec 8 (appendices).

5. RECOMMENDATIONS

In order to improve the efficiency and effectiveness of the proposed system, the following recommendations can be considered:

- **Remote controlling:** The option of controlling the water pump can be given to the farmer i.e. the farmer switches on/off the water pumps to start/stop the irrigation process without being present at the backyard farm.
- **Water quality monitoring:** Since the quality of water may affect the growth of crops and the irrigation system's watering pipes, implementing additional monitoring for the water may prevent these problems from happening.
- **Fire detection and climate control:** Although the currently proposed system monitors the weather conditions, it does not contribute to detecting any natural fires or climate control. Therefore, I would recommend investigating these aspects by integrating the system with NASA data.
- **Improve the web application:** The current system can be improved to deliver alerts to the farmer, using email or SMS notifications. An example of this, when the irrigation process starts and ends. It can be also improved by adding general descriptions about the plants. Finally, the system could have the possibility for customization and change by allowing the farmer to input the plant fields and the thresholds for the moisture levels.

6. CONCLUSION

This paper presented the design and implementation of the prototyped IoT Backyard system. As the results obtained, the system indeed solved the problem it was supposed to solve, being automatically water backyard fields based on virtual sensor readings of soil-moisture levels and providing additional environmental readings. The IoT Backyard web application enables farmers to monitor the plants through the current and historical values of soil moisture, as well as the weather conditions with the state of the irrigation process. I believe that the developed prototype can be applied to any plant and its irrigation. Meanwhile, the collected data can automate the regular monitoring of the plant, contribute to water saving, and effectively replace manual irrigation with smarter and more efficient process execution.

7. REFERENCES

[1] Lohr, V. I. (2009, June). What are the benefits of plants indoors and why do we respond positively to them?. In *II International Conference on Landscape and Urban Horticulture 881* (pp. 675-682).
[2] Athawale, S. V., Solanki, M., Sapkal, A., Gawande, A., & Chaudhari, S. (2020). An IoT-Based Smart Plant Monitoring System. In *Smart Computing Paradigms: New Progresses and Challenges* (pp. 303-310). Springer, Singapore.
[3] Pravin, A., Jacob, T. P., & Asha, P. (2018). Enhancement of plant monitoring using IoT. *Int. J. Eng. Technol.(UAE)*, *7*(3), 53-55.
[4] Rawal, S. (2017). IOT based smart irrigation system. *International Journal of Computer Applications*, *159*(8), 7-11.
[5] Pant, D., Bhatt, A., Khan, M., Nautiyal, O. P., & Adhikari, P. (2019). Automated IoT based Smart Water Quality Assessment System. In *2019 8th International Conference System Modeling and Advancement in Research Trends (SMART)* (pp. 98-104). IEEE.
[6] Li, W. J., Yen, C., Lin, Y. S., Tung, S. C., & Huang, S. (2018). JustIoT Internet of Things based on the Firebase real-time database. In *2018 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE)* (pp. 43-47). IEEE.

8. APPENDICES

- The source code for this project was hosted on GitHub and can be found here.
- The video demo is available here.