DESIGN AND IMPLEMENTATION OF AN IOT SMART FARMING SYSTEM

Julius Kakooza Kawemba*
Student, M.S./Computer Science Program, Rivier University

Abstract

In all human history, agriculture is one of the most important industries and a backbone for prevalence of life. The industry has provided and still provides of the essence imperative resources like food, energy, and raw materials. Adoption of the Internet of Things (IoT) is on the raise in all sectors of the economy and life, from health, home systems, transportation, and education among others. Every sector that incorporates IoT has proven tremendous benefit from this innovation that gears automation and connection of devices "things". Employing the IoT technology into the agriculture industry will foster a drastic growth of the industry. In this project, I describe how a simple IoT smart farming system can be designed and implemented. The project is designed towards helping farmers to monitor, predict and improve efficiency on their farms through real-time data collection and enhanced decision making.

Introduction

The application of modern Information Communications Technology (ICT) into different industries and sectors like agriculture, health, education, transport, etc., boosts functionality, efficiency and high productivity. The United Nations Food and Agriculture Organization[1] predicts that in order to keep pace with population growth, food production must increase by 70% by 2050. Biomedical analyst and author Dr. Therese Cory from Beecham research explains this growth: "…The demand for more food has to be set against the challenges of climate change, more extreme weather conditions and the environmental impact of intensive farming practices." [2] Employment of technologies like the Internet of Things (IoT) to create smart farms and smart agricultural system could be the key solution to the farming and agriculture industry to increase production, efficiency and functionality by 70% that can meet the goal of feeding the 9.7 billion population estimated growth by 2050.

Primarily, the internet was used to connect computers anytime and anyplace with the help of human monitoring. However, the evolution of IoT technologies brings the connection of anything at any place and time by linking objects of the real world with the virtual world. The International Telecommunication Union (ITU) defines IoT as "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies"[3]. Smart farming conceptualizes farm management using ICT and IoT to foster decision making for improving and increasing quality and quantity of farm products.

Agriculture in general involves the cultivation of crops and taking care of domestic livestock. Both crops and livestock require different sets of care management and preparation order to yield a product

[1] The Food and Agriculture Organization (FAO) of the United Nations, September 23, 2009. http://www.fao.org/news/story/en/item/35571/icode/. Accessed October10, 2019.
[2] Beechamresearch, Smart Farming Sales Brochure 2017. http://www.beechamresearch.com/. Accessed September 20, 2019.
[3] International Telecommunication Union Series Y.2060. Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks. – An Overview of the Internet of Things, ITU-T, June 2012.

ISSN 1559-9388 (online version), ISSN 1559-9396 (CD-ROM version).

ready for human consumption, surplus food and commercial markets. However, in this project we are particularly drawing our focus on crop growing. A process that involves cultivation of crops/plants and caring for them till they yield either on a small or large scale. Some of the main factors that facilitate and influence the proper growth of crops and plants include water levels, moisture, soil nutrients/fertility, light and temperature. Successful harvest for a farmer will highly depend on the farmer's knowledge about the status of the factors mentioned above, which is the critical part of farming, monitoring and making decisions based on what has been observed.

Project plan

The purpose of the project plan is to give the direction of the project while keeping focus on achieving its objectives on time. This is demonstrated through showing the main activities, outcomes, milestones and resources required for the project. For the implementation of this project, here is the activity project plan (see Table 1).

Table 1: The IoT Smart Farm project plan

Week 1		Week 2	4 Weeks [Week 3,4,5,6]	Week 8
Create project plan				
Functional Specification Plan	Create Functional Specifications			
	Create Class diagram			
	Create Use Case Diagrams			
	Create Object Diagrams			
	Create Squence Diagrams			
System Design Specications		Create systems architecture		
		Entity relational diagrams		
		Windows navigation diagrams		
		Draft user interface		
		Database tables		
Integration Test Plan			Programmng environment & tools	
			Coding	
			User interfaces	
			Functional tests	
				Unit test plan
				Integration test
				Results test
				Document test results
Final Documentation				

System Functional Specifications

A broad concept that smart farming and smart agriculture derive from IoT technology is based on the instant data collection, transmission and processing. The idea of creating a smart farming system is to enhance and foster instant decision making by the farmer basing on the data collected in real time. The health development of crops and animals on a farm depends on the decisions made by the farmer through observation and monitoring, which can cause setbacks because the farmer cannot be at the farm 24/7 making all these observations. Our goal here is to deploy the IoT technology to help the farmer to remotely make the observation and monitoring 24/7 and in real time. Factors like water level, soil moisture, temperatures, soil fertility, soil pH, humidity, and wind speed that affect crop growth and production can be monitored using sensors directly installed at the farm.

In this project the system is expected to perform the following functions:

- Physically monitor farm-soil conditions like soil moisture, soil temperature, humidity, soil pH, etc.
- Control farm-soil conditions
- Collect real-time data on farm-soil conditions
- Transmit the collected data
- Generate reports.

Use case diagram

To model functionality of the system, we use different actors, external entities ("roles"), and the associated use cases represented on the use case diagram. The use case diagrams are also used to show the functions, actions and services that the systems will perform. The use case diagram for the IoT Smart Farm project is shown in Fig. 1.

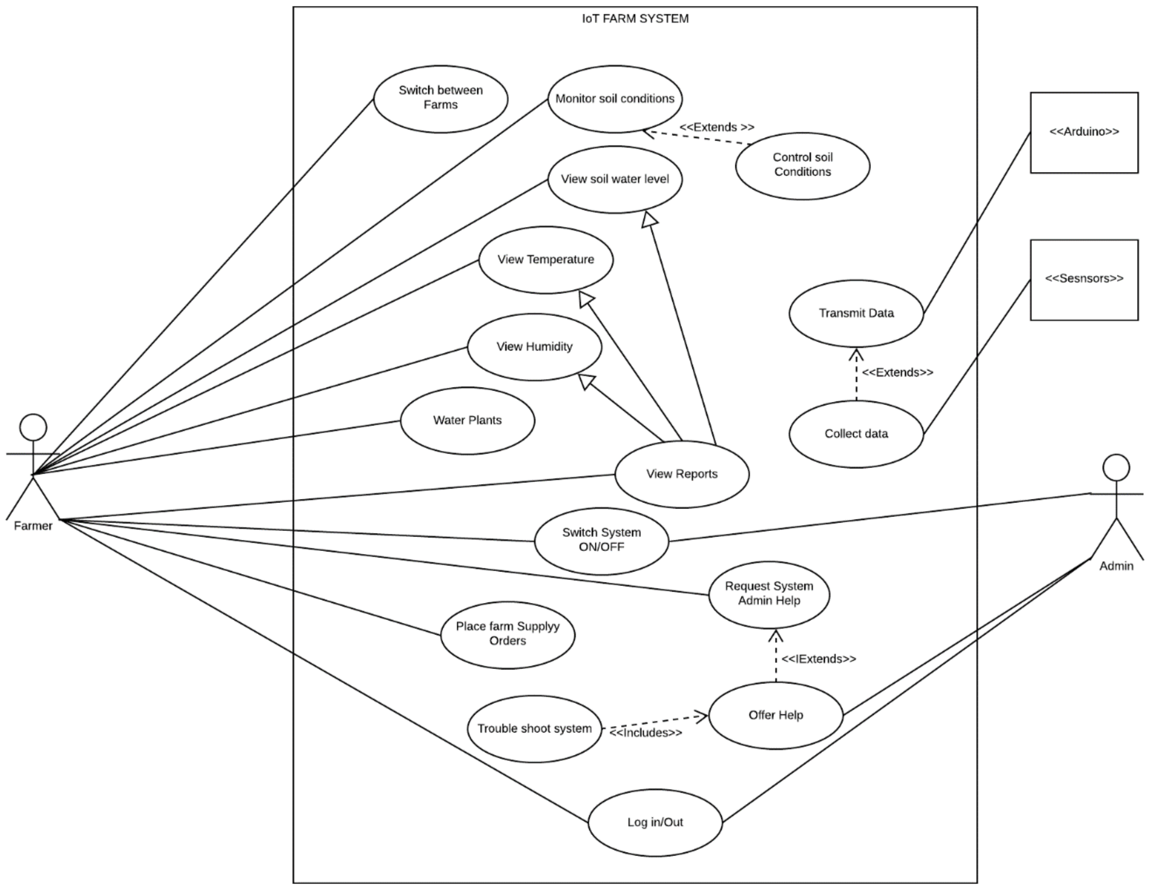


Figure 1: The use case diagram for the IoT Smart Farm System

Class diagrams

Class diagrams are structural Unified Modeling Language (UML) diagrams that are used to demonstrate the static view of the system. They are used to document, visualize and describe different aspects of the object-oriented system as well as for constructing the executable functions and developing the programming code for the system. Class diagrams are widely used for describing the attributes and operations of the class and constraints imposed on the system. Samples of the class diagrams are shown in Figs. 2 and 3.

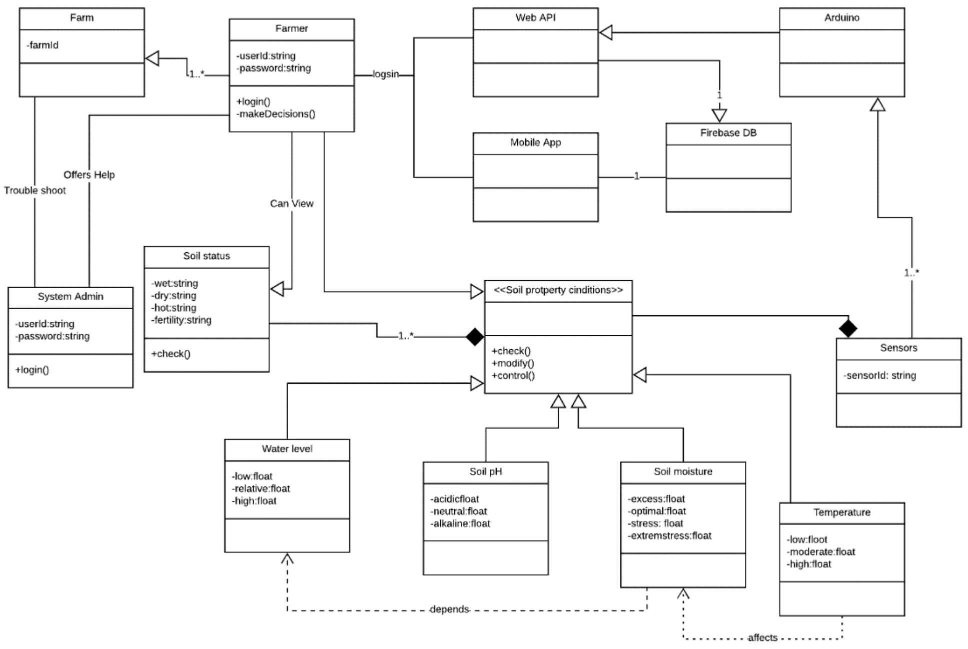


Figure 2: The class diagram for the Smart IoT Farm System

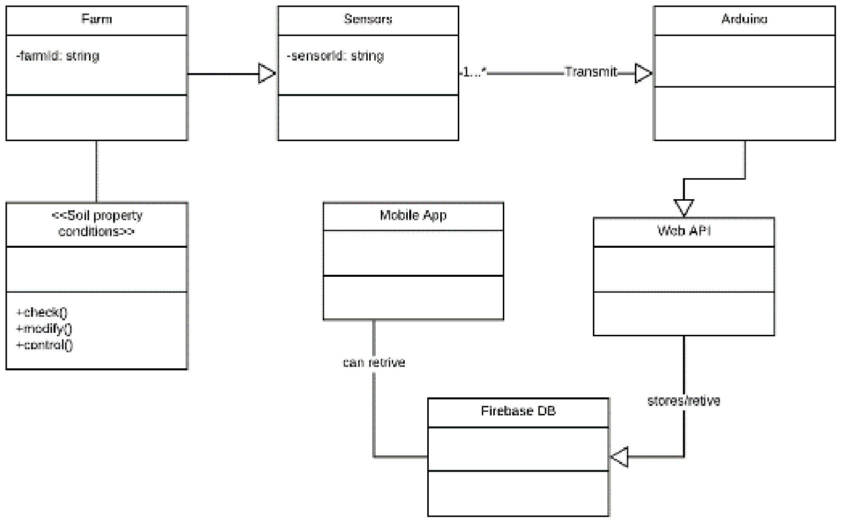


Figure 3: The class diagram for the data storage of the IoT Farm System

Object diagrams

Object diagrams can be used to represent specific instances of classes and relationships between them at a point of time. Object diagrams are similar to class diagrams, however, object diagrams show the instances of classes within a system. They show the complete or partial view of the structure of a modeled system at a specific time. This can help us study the behavior of the system at a particular instant. The sample of the object diagrams is shown in Fig. 4.

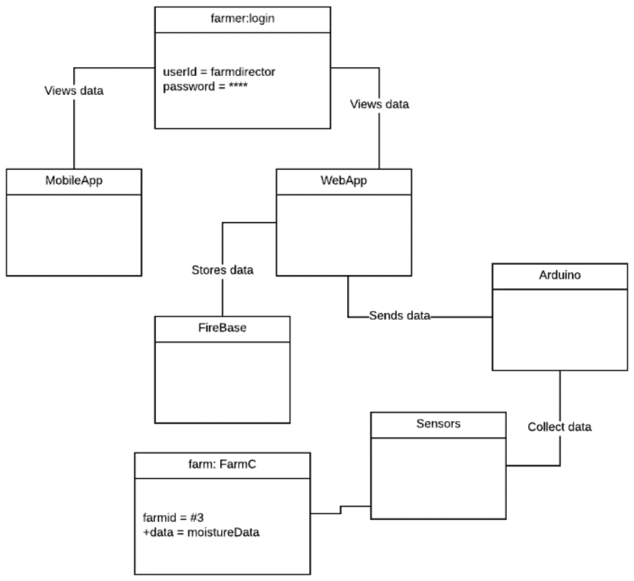


Figure 4: The IoT Smart Farm object diagram for observing data on soil moisture from Farm C

Sequence diagrams

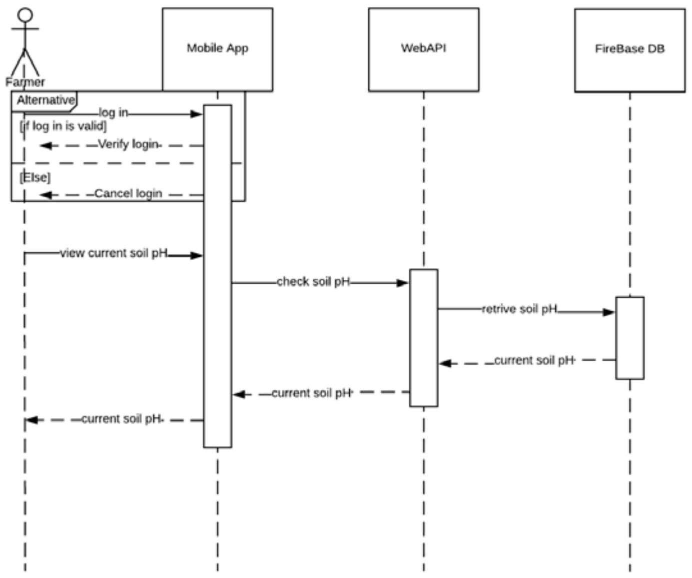


Figure 5: The IoT Smart Farm System sequence diagram for observing data on the current soil pH level

In order to demonstrate the interaction among classes in terms of message exchange over time/events we use “sequence diagrams” also called “event diagrams”. Sequence diagrams help us validate and visualize several system events for predicting and analyzing how the system will behave. Here are some of the sequences overtime. Samples of the sequence diagrams are shown in Figs. 5-7.

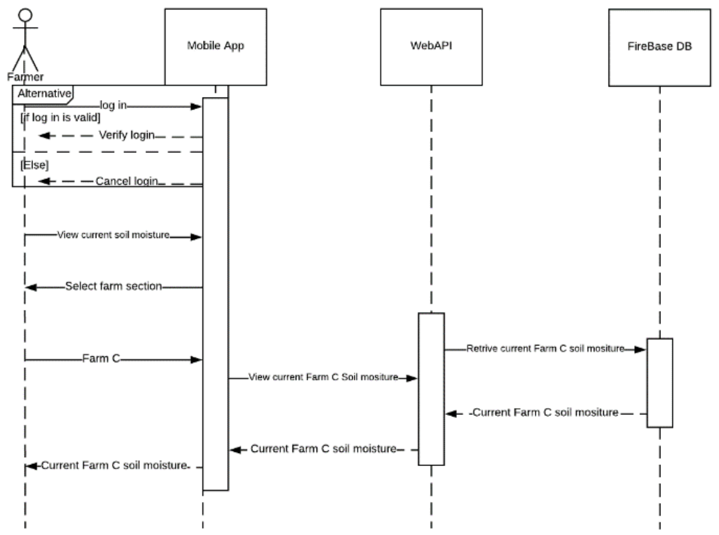

Farmer
Mobile App
WebAPI
FireBase DB
Alternative
log in
[if log in is valid]
Verify login
[Else]
Cancel login
View current soil moisture
Select farm section
Farm C
View current Farm C Soil mositure
Retrive current Farm C soil mositure
Current Farm C soil mositure
Current Farm C soil moisture
Current Farm C soil moisture

Figure 6: The Smart IoT Farm sequence diagram for observing data on soil moisture from a farm

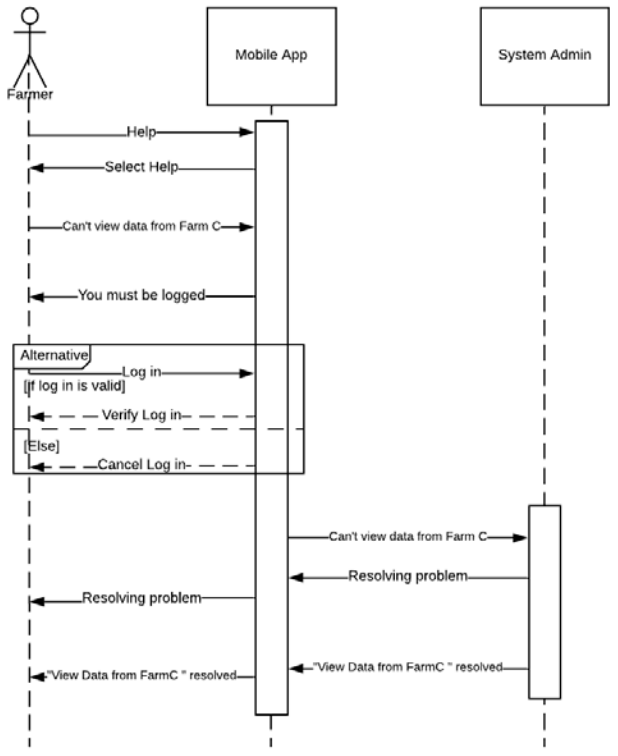


Figure 7: The Smart IoT Farm System sequence diagram for sending a Help Request

Functional Test Plan

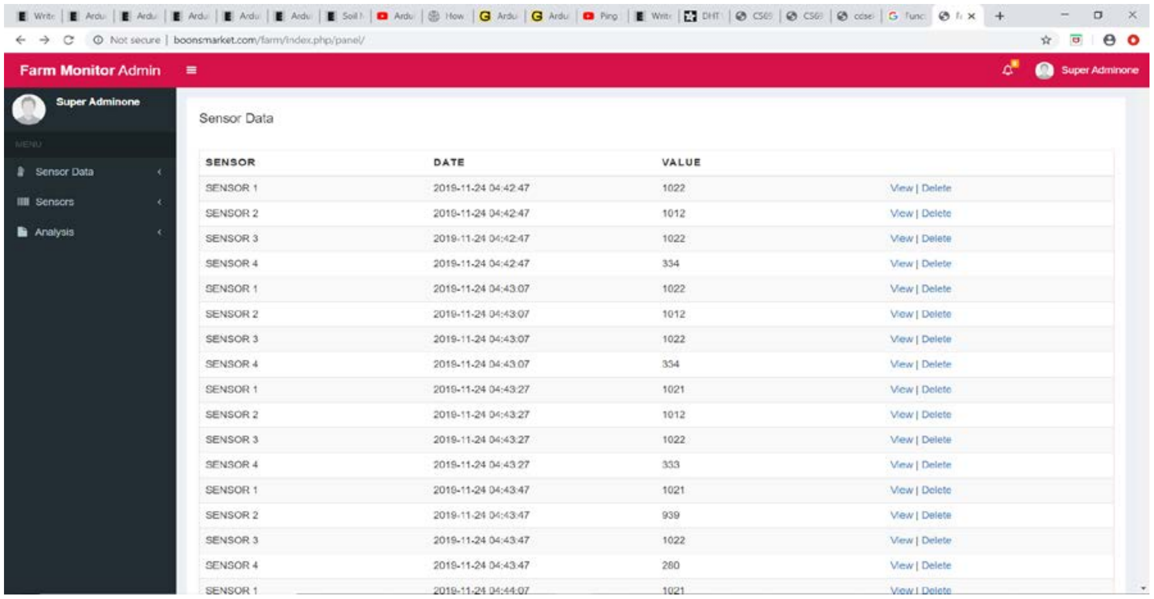
The main purpose of designing and implementing ICT systems is to solve problems through performing specific functions. In this project the purpose is to create an interactive IoT smart farm system that can enable the farmer to monitor farm soil conditions and also be able to implement and make certain decisions in real-time.

A function test plan helps us to check and validate each system's function from the user's point of view before implementation. In this project the following functionalities were tested (see Table 2).

Table 2: Functional tests performed

Functions tested	**Test result**
Log in	Tested both Admin & User Log instructions
Collect data	The sensors collected data and sent it to the Arduino™
Transmit data	Data collected was transmitted to the Web App
Retrieve data	After capturing data, we were able to view it and view it in different forms like graphs
Turn water pump on	Water pumped was activated
Send requests	Requests were sent through the user mode and received via admin mode
Place order	While in user/famer's mode, we were able to place an order to a supplier.
Units tested	
Moisture sensors	Different moisture sensors were tested out to determine their default/stable readings
Water level sensor	These are very hard to work with; however, ranges of performance were established after multiple tests
Temperature & Humidity sensor	This is highly sensitive area; during testing it was discovered that it is hard to determine the actual temperature of a farm using sensors that were purchased for the project
Arduino™ UNO	Three versions of Arduino™ were tested; after the testing, the decision was made on the UNO Wi-Fi Rev.2 for the project
Breadboards	Five Breadboards were tested
LEDs	The LEDs were used for the regime indications
Water pump	The water pumps were tested with and without a relay
Mobile App	Tested the functionality and its connectivity to the database and web app
Web App	Web App was able to receive data from Arduino™
Firebase Database	Data was able to be stored and retrieved as required

Several user interfaces (UI) were designed and used for conducting these tests (see Figs. 8-9).



Sensor Data

SENSOR	DATE	VALUE	
SENSOR 1	2019-11-24 04:42:47	1022	View | Delete
SENSOR 2	2019-11-24 04:42:47	1012	View | Delete
SENSOR 3	2019-11-24 04:42:47	1022	View | Delete
SENSOR 4	2019-11-24 04:42:47	334	View | Delete
SENSOR 1	2019-11-24 04:43:07	1022	View | Delete
SENSOR 2	2019-11-24 04:43:07	1012	View | Delete
SENSOR 3	2019-11-24 04:43:07	1022	View | Delete
SENSOR 4	2019-11-24 04:43:07	334	View | Delete
SENSOR 1	2019-11-24 04:43:27	1021	View | Delete
SENSOR 2	2019-11-24 04:43:27	1012	View | Delete
SENSOR 3	2019-11-24 04:43:27	1022	View | Delete
SENSOR 4	2019-11-24 04:43:27	333	View | Delete
SENSOR 1	2019-11-24 04:43:47	1021	View | Delete
SENSOR 2	2019-11-24 04:43:47	939	View | Delete
SENSOR 3	2019-11-24 04:43:47	1022	View | Delete
SENSOR 4	2019-11-24 04:43:47	280	View | Delete
SENSOR 1	2019-11-24 04:44:07	1021	View | Delete

Figure 8: A screen shot of raw data collected from 4 sensors and transmitted during some testing phases

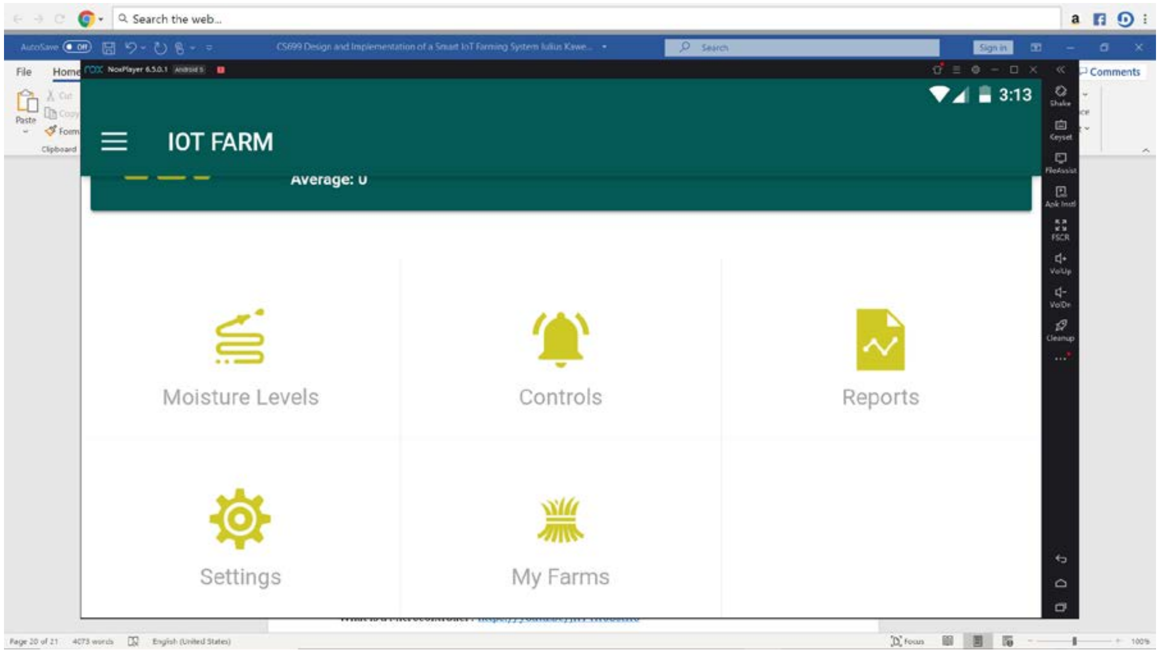


Figure 9: The IoT Smart Farm App as tested in the NoxPlayer™ mobile demo App

Systems architecture

To visualize the IoT Smart Farm system from the perspective of different viewers and users, a simple organization of project components was developed in order to have a simple design (see Fig. 10). The components are put together in a way of defining a structured solution that meets the technical and operational requirements while optimizing common quality attributes such as performance in real-time.

There is the list of project-development components:

Physical components:

- Arduino™ UNO Wi-Fi Rev.2
- LEDs
- Breadboard
- Jumper wires
- Moisture sensors
- Water level sensors
- Water pump
- DHT11 temperature & humidity sensors
- Garden sets (soil samples)
- Farmer/User

Virtual components:

- Firebase™ Database
- Mobile Application
- Web Application

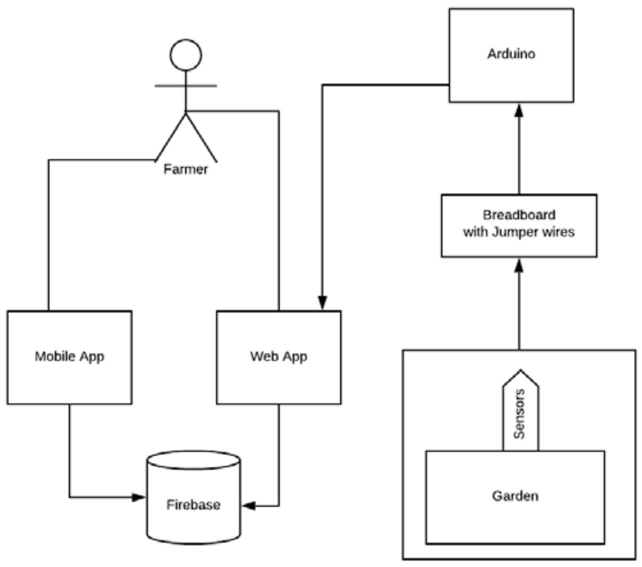


Figure 10: The IoT Smart Farm system architectural design diagram

Programming Environments and Tools

The development of this Smart IoT Farm project has been made possible by utilization of some programming tools and environments in order to test the theoretical concepts and system design implementations explored by the researcher. Here are some of the tools and environments.

Microcontrollers

Microcontrollers are integrated circuits designed to control/govern a specific task or operation in an embedded system. A microcontroller is a computer dedicated to a single task and contains a CPU, memory, peripherals and a system clock (oscillator). Most programmable microcontrollers used today are embedded in consumer products, health devices, cars, phones, etc.; therefore, these microcontrollers are named as "embedded controllers".

The CPU contained in the microcontroller acts as the brain of the device and is used to process and respond to various instructions that direct the microcontroller's function. It also performs basic arithmetic logic operations, I/O commands, and data transfer operations that communicate commands to other components in larger embedded systems.

Memory on the microcontroller is used to store data that the processor receives and uses it to respond to the instruction that the processor has been programmed to carry out. A microcontroller consists of two kinds of memory: 1) program memory (non-volatile memory) which stores long term information about the instructions that the CPU carries out, and 2) data memory (a volatile memory) for temporary data storage while instructions are being executed.

The I/O peripherals on the microcontroller connect the chip to the physical world. The micro-controller contains input ports that are used to receive information sent to the processor for instructions to execute external tasks. Microcontrollers are designed with pins that are used to perform the I/O peripheral functionality.

The system clock on a microcontroller is based on the oscillator performance and acts as the engine of the controller. It controls the speed at which it receives and processes information that affects the controller's resolution, response, speed, and power consumption.

It is noted that microcontrollers are often confused with microprocessors. Both have been designed for real time application and share common features; however, they have significant differences and cannot be distinguished by looking at them. Microprocessors only have a CPU inside them and cannot function as a stand-alone system. They require additional external connections of peripherals, I/O, RAM and ROM to the chip for functioning. On the other hand, microcontrollers have a CPU, a fixed amount of RAM, ROM and other peripherals that are all embedded on a single chip.

Arduino™

What is Arduino™?

When someone mentions the word "Arduino" to you, a lot goes on into your mind to what they could be referring to.

One, Arduino™ is an open-source electronics company that designs and manufactures integrated circuit boards that make microcontrollers easy to user. It has a community of users that design single board microcontrollers and microcontroller kits for building digital devices. The Arduino™ project was started in 2003 at the Interactive Design Institute Ivera (IDII) in Italy and came much into use in 2005. The project and user community has since grown widely, as estimated to have millions of designers, developers and manufacturers.

Arduino™ can also refer to the integrated circuit board controller designed and manufactured by the Arduino Company. The circuit board designed by the Arduino Company makes microcontrollers easy to use. Today, there are many types of Arduino™ circuit boards because the company open sources their software and designs. The most popular Arduino™ circuit designs include: Arduino UNO, Arduino Mega, Arduino Yun, and Arduino UNO Wi-Fi Rev2 among others. These circuit boards contain microcontrollers that can be used to control motors, sensors, robots, cameras or just anything. For demonstration of this project, we use the Arduino™ UNO Wi-Fi Rev2, and Arduino™ circuit board that comes integrated with Wi-Fi.

Arduino™ UNO Wi-Fi Rev2

Open source electronics circuit board built with an on-board MEGA4809 microcontroller and a crystal resonator that controls how fast the microcontroller runs.

Before the development of Arduino™ circuit boards came into play, microcontrollers were complex to program. One had to write a lot of binary code, work with registers, and then use special programming hardware with custom-made cables to upload your program to a microcontroller. When Arduino™ came into play, programming a microcontroller is just as simple as uploading your code via a type-B USB cable through your computer. This is one of the factors that make Arduino easy to use and program popularly. When the USB is plugged into your computer, it also powers the Arduino, so you do not need an external power supply. However, the Arduino™ circuit board can also be powered through an adapter via its power jack. The board only requires 5v to turn on, but can support external power of up to 12v which can be regulated. The Arduino board also has an on-board voltage regulator and can supply 5v and 3.3v to power up low power components.

At its glance, the UNO board has 14 I/O digital pins (of which 6 can be used as PWM outputs), 6 analog input pins, a 16MHz ceramic resonator, a USB connection, power jack, an ICSP header, and a reset button at basic observation.

In detail, the 14 I/O digital and 6 analog pins incorporated on the circuit board allow external connection with any circuits and devices with the board. The analog pins are marked A0 to A5 and come with a resolution of 10 bits, meaning they can represent analog voltage by 1024 digital levels. The reset button on the board can be used to reset the whole board and takes the running program in the initial stage. Pushing this button clears everything up in the program and starts the programming from the beginning, - this is useful when the board hangs up the middle of a running program. The on-board MEGA4809 microcontroller comes with 2Kbs RAM, 48K flash memory, 256 bytes of EEP-ROM and 16MHz-frequency crystal oscillator. The programming of the MEGA4809 happens through an ATmega32U4 unit programmed with the mDBG code and creates a virtual COM port when connected to a PC. The I/O digital and analog pins on the board operate at 5v and come with standard operating rating between 20mA to 40mA. The board also consist of a VIN pin that can be used to power the Arduino™ using an external power source. Any current exceeding (20mA to 40mA) is controlled by an internal resistor on the board. The Arduino™ UNO board also comes with a built-in LED connect pin number 13, providing the board with HIGH power value to turn the LED on and with LOW power value to turn the LED off. The GND ground pins on the board are all interconnected and are used to close the electro-circuit and provide a common logic reference level throughout the entire circuit.

The board also contains IOREF pin for I/O reference that provide the voltage reference with which the microcontroller operates. The AREF pin is the analog reference, and it is used to provide voltage to analog inputs.

Arduino™ uses serial communication to exchange data between other devices like sensors, displays, computers, etc. The board uses the 0(Rx) digital pins to receive and the 1(Tx) pins to transfer data as well as the data communication via the board’s type-B USB port.

Arduino™ supports serial communication through digital pins with software serial libraries, allowing users to connect to multiple serial-enabled devices and leave serial port available for USB. Digital 10-13 pins are dedicated to Serial Peripheral Interface (SPI) communication. SPI is a serial data protocol used by microcontrollers to communicate with one or more external devices in the bus-like connection. The SPI can also be used to connect 2 microcontrollers. ATMega16U2 is incorporated the Arduino™ board to enable this serial communication using the USB com drivers.

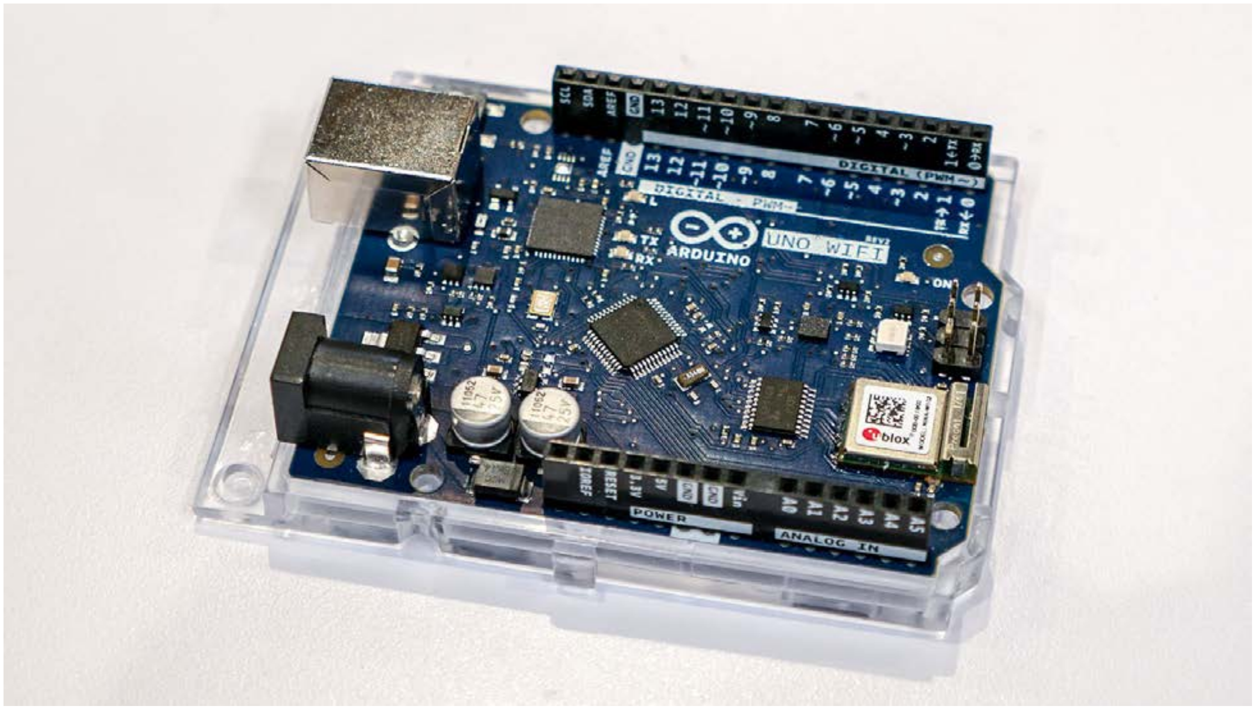


Figure 11: Arduino™ UNO Wi-Fi Rev2[4] (www.arduino.cc)

Arduino IDE

The Arduino Company also created an integrated development environment cross platform software that helps you configure your Arduino™ hardware products to UNO via USB. The software application is written in Java and is used to write the instructions which perform your tasks. The code contains specific definitions:

- **setup( )**: This function is present in every Arduino sketch ant its run once before the loop ( ) function. Often used to set pinMode to input or output.
  The setup( ) function has the following structure:
  void setup( ) { //code goes here }
- **loop( )**: The loop function is also present in every Arduino sketch and runs over and over again. It is in this loop that all the program functionality happens. The one exception to this is setup() and variable declaration.
  The loop( ) function looks like:
  void loop( ) { //code goes here }
- **input:** A pin mode that intakes information.
- **output:** A pin mode that sends information.

[4] Arduino™ UNO Wi-Fi Rev2, n.d. www.arduino.cc. Accessed October 29, 2019

- **HIGH:** Electrical signal present (5V). Also used for the ON or TRUE in Boolean logic.
- **LOW:** No electrical signal present (0V). Also used for the OFF or FALSE in Boolean logic.
- **digitalRead**: This is used to get HIGH or LOW reading from a pin already declared as an input.
- **digitalWrite**: Assign a HIGH or LOW value to a pin already declared as an output.
- **analogRead**: Get a value between or including 0 (LOW) and 1023 (HIGH). This allows you to get readings from analog sensors or interfaces that have more than two states.
- **analogWrite**: Assign a value between or including 0 (LOW) and 255 (HIGH). This allows you to set output to a PWM value instead of just HIGH or LOW.
- **PWM**: Stands for Pulse-Width Modulation, a method of emulating an analog signal through a digital pin. A value between or including 0 and 255. Used with analogWrite.

Firebase™ Database

"Firebase™ Real-time Database[5] is a cloud-hosted NoSQL database that lets you store and sync data between users in real-time." – firebase.google.com/products.

NoSQL database also commonly referred to as "not only SQL" and or "non SQL" database is a database that uses a mechanism of storage and retrieval of data that is modeled in means other than tabular relations.

A real-time database management system is a system that employs a real-time processing of data to work with data that is constantly being manipulated or changed. Like traditional (relational) databases, real-time databases also offer data repository retrieval, manipulation of information, efficient storage but with a uniqueness timing aspect. For example, if you are monitoring traffic conditions, the information (which you expect to provide) is the current live status of traffic in a specific location. Such information happens in real-time, collected in real-time, stored and retrieved instantly in real-time. Real-time databases can be employed in stock markets, traffic control, accounting, banking, process control, multimedia, etc.

The concept of NoSQL databases was introduced in the early 1960's, however it was not prominent. It became popular after the introduction of Database as a Service (DaaS) as a result of the high level of scalability of NoSQL databases.

NoSQL database model does not guarantee the ACID (Atomicity Consistency Isolation Durability) properties of database management systems, but guarantees the BASE (Basically available Soft state Eventual consistency) database management properties. The ACID properties of a database system are the system's concept characteristics that guarantee the four transactional properties: atomicity, consistency, isolation, and durability - to ensure reliability, accuracy, completeness, and data integrity of the database. In this case, a transaction is a single logic operation that may consist of one or many steps. The ACID properties will ensure that, in the case of any failures or errors, the data contained in the database remains valid and accurate.

On the other hand, BASE properties accommodate the flexibility of NoSQL databases by relaxing consistency to allow the system to process requests even in an inconsistent state, which gives it scalability and resiliency. The BASE data storage model values data availability for scalability purpose but they won't be guaranteed consistency of replicated date at write time.

[5] Google, Inc. Firebase™ database, n.d. firebase.google.com/products. Accessed October 25, 2019

Scalability is the ability of a database to handle changes by adding/removing resources and increasing data without sacrificing its performances. So, rather than requiring consistency after a transaction, it is desirable for the database to “eventually” be in a consistent state.

It is important to note that NoSQL databases have no the standard structures-query-language (SQL) instructions, and this fact sometimes makes it difficult for user to switch from one NoSQL database to another. However, a new concept called UnQL – Unstructured Query Language (pronounced as “uncle”) is being developed to provide a familiar syntax, standardized data definitions, and manipulation to NoSQL. UnQL is considered as the superset of SQL and provides the SQL-like syntax that is appropriate for the Unstructured self-defining data format.

The other concept is a hybrid model in which the familiar SQL is implemented at the front-end of the system and the NoSQL at the back-end. However, such system models do not perform as fast as the purely NoSQL databases.

The Firebase™ real-time database has been used in this project in order to be able to collect data and monitor real-time crop situations on the farm.

Solderless breadboards

The solderless boards have been used for prototyping electrics and testing circuit designs. Breadboards have strips of metal underneath the board and connect that holes on the board.[6] Breadboards come in a variety of designs and basically have the same working mechanism. All boards come with a set of holes that are just the right size for accepting the leads of electronical devices (see Fig. 12). The holes on the board are electronically connected in a symmetric manner. The top and bottom usually consist of a bus strip. The holes are connected horizontally, and the board is usually connect to the power supply and ground. The middle of the board has holes with terminal strips and are vertically connected. These holes hold most all the electronic components.

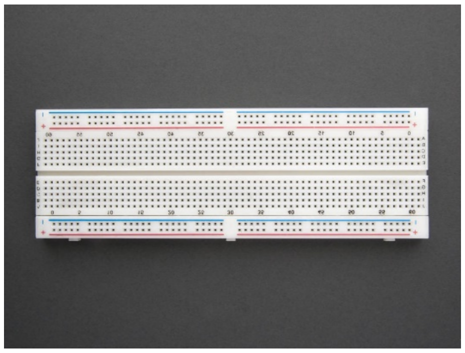


Figure12: Solderless breadboard

Soil moisture sensor

Soil moisture is the water that is held up in the spaces between soil particles.[7] Soil moisture is an important factor for crop growth; therefore, to a crop farmer, it is important to monitor and acquire information about the amount of moisture in the soil. Soil moisture can be measured and determined using various methods and instruments. In this project, we employ soil moisture sensors that measure the volumetric water content directly by electric resistance. The sensors are calibrated for a resistance

[6] Wiring, what is a breadboard? N.d. http://wiring.org.co/learning/tutorials/breadboard/. Accessed November 12, 2019.

[7] GHCC, Soil Moisture. December 30, 1999. https://weather.msfc.nasa.gov/landprocess/. Accessed September 20, 2019.

between 0 to 1023 levels, when the current flows through the two probes of the sensor. The more resistance is observed between the two probes, - the soil is drier; and the less resistance reflects the fact that the more moisture is in the soil. The moisture sensor components are shown in Fig. 13.

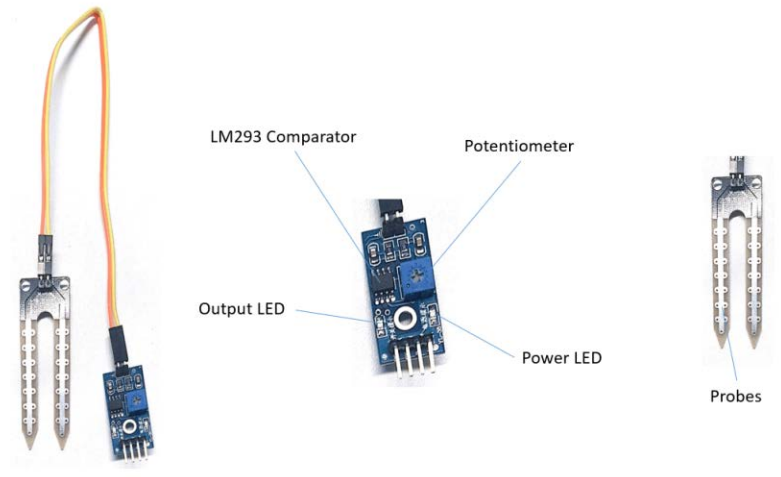


Figure 13: The moisture sensor components[8]

Other programming tools (that were used in this project but have not been described above) include: JavaScript to write part of the mobile application, Android studio for the Mobile App development, PHP for the Web Application, and HTML.

Project schematic layout and connections

The schematic layout and connections are specified in Table 3 below.

Table 3: IoT Smart Farm project schematic layout and connections

Arduino	**Connections**		
GND	All GND connections are connected to the GND of Arduino, the +ve terminal of the LEDs, the +ve Terminal of the H2Op		
5v	VCC of Moisture sensors, Comm of R, VCC of R, + terminal of H2OLv		
A0	H2OLv S terminal		
A1	MS1		
A2	MS2		
A3	MS3		
A4	MS4		
A5			
Digital Pin2	RLD1(-ve)		
Digital Pin3	GLD1(-ve)		
Digital Pin4	RLD2(-ve)		

[8] UWE PDT Soil Moisture sensors, April 03, 2018. https://uwearduino.wordpress.com/. Accessed September 15, 2019

Digital Pin5	GLD2(-ve)		
Digital Pin6	RLD3(-ve)		
Digital Pin7	GLD3(-ve)		
Digital Pin8	RLD4(-ve)		
Digital Pin9	GLD4(-ve)		
Digital Pin10			
Digital Pin11			
Digital Pin12			
Digital Pin13	IN pin of R		
Relay	*NO pin is connected to the +ve terminal of the H2Op*		
ID	**Key**	**ID**	**Key**
Moisture sensor1	MS1	Green LED for MS1	GLD1
Moisture sensor2	MS2	Green LED for MS2	GLD2
Moisture sensor3	MS3	Green LED for MS3	GLD3
Moisture sensor4	MS4	Green LED for MS4	GLD4
		Red LED for MS1	RLD1
		Red LED for MS2	RLD2
		Red LED for MS3	RLD3
Water pump	H2Op	Red LED for MS4	RLD4
Relay	R		
Water level sensor	H2OLv		

System Integration Test Plan

Each component of the system have been tested individually following the unit test plans. After this the components have been assembled and put together employing the designed system architecture. The description of the system integration tests and the corresponding design setups are shown in Table 4. The focus is to ensure that the flow of data among the objects meets the requirements and that the connections conform to the system design and objectives.

Table 4: Integration tests performed

The Conducted Test	Description Notes & Results
Sensors	All sensors were able to pick up signals
Arduino & sensors	Successful, sensors have different connections as shown in the schematics (see Fig. 10). All entities sent data to Arduino
Breadboard connections	The breadboard helps organize the connections
Arduino & Web App	Successful, data collected was transmitted to the web via Internet
Web App & Firebase™ DB	Successful
Mobile App & Firebase™ DB	Successful

Samples of User Interfaces with Demos

After the tests were performed on the individual entities and on the system with the assembled components, some raw data has been collected and the data communication channels have been analyzed. The samples of the user interfaces and demos are shown on Figs. 14-21 below.

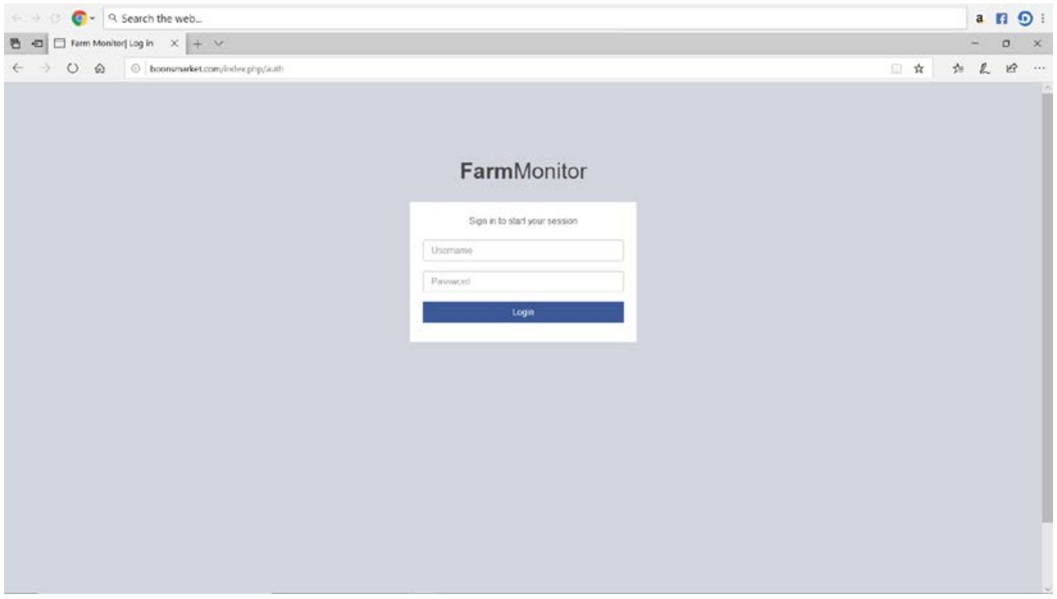


Figure 14: The Login home page for the IoT Smart Farm user

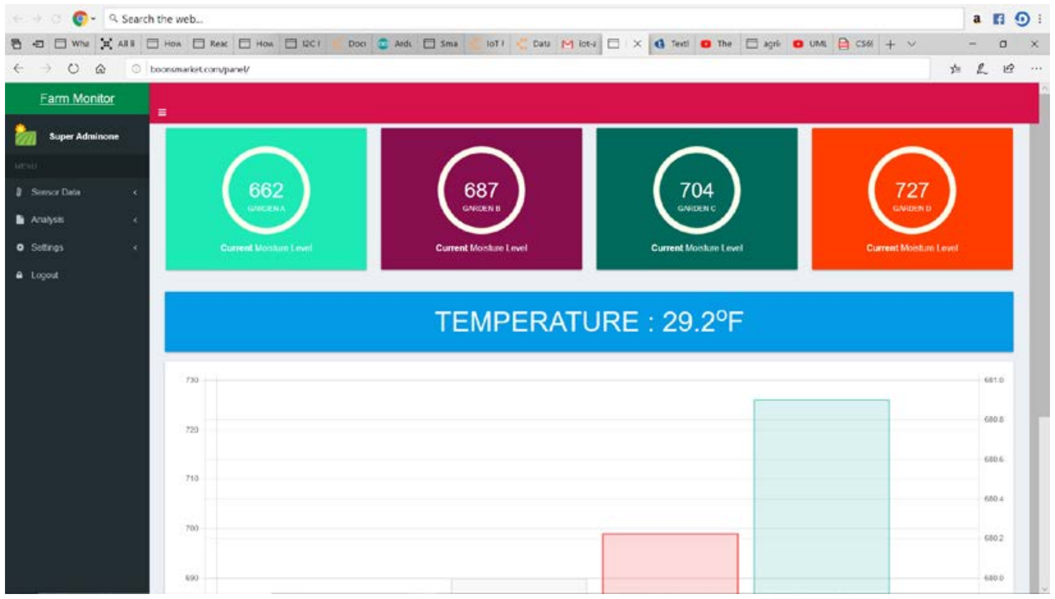
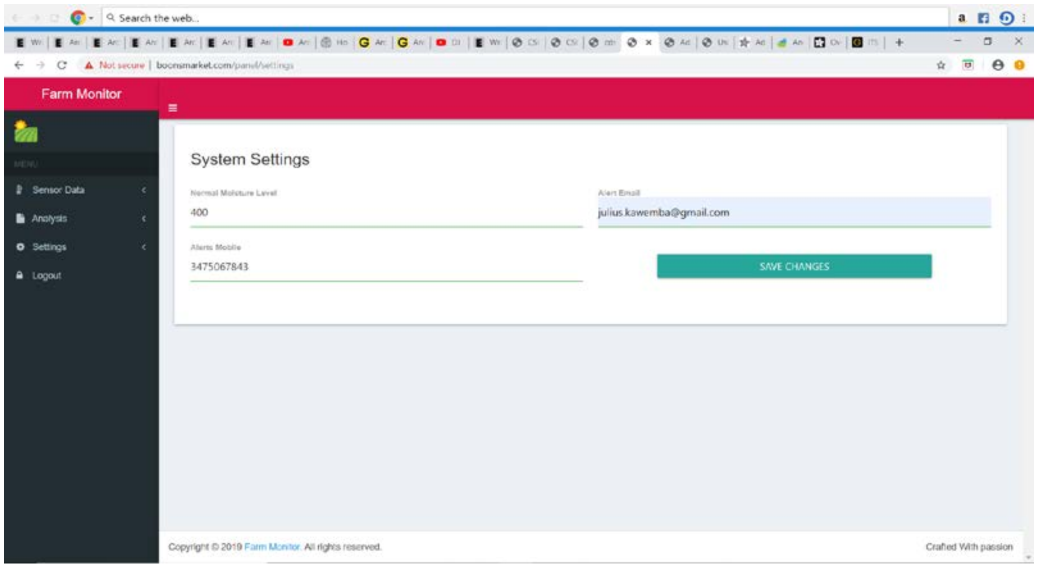


Figure 15: The IoT Smart Farm home page with monitors for soil moisture and current outside temperature with graphs of the moisture levels



System Settings

Normal Moisture Level: 400

Alert Email: julius.kawemba@gmail.com

Alerts Mobile: 3475067843

SAVE CHANGES

Copyright © 2019 Farm Monitor. All rights reserved.

Crafted With passion

Figure 16: Web App settings page to change systems alerts for soil moisture

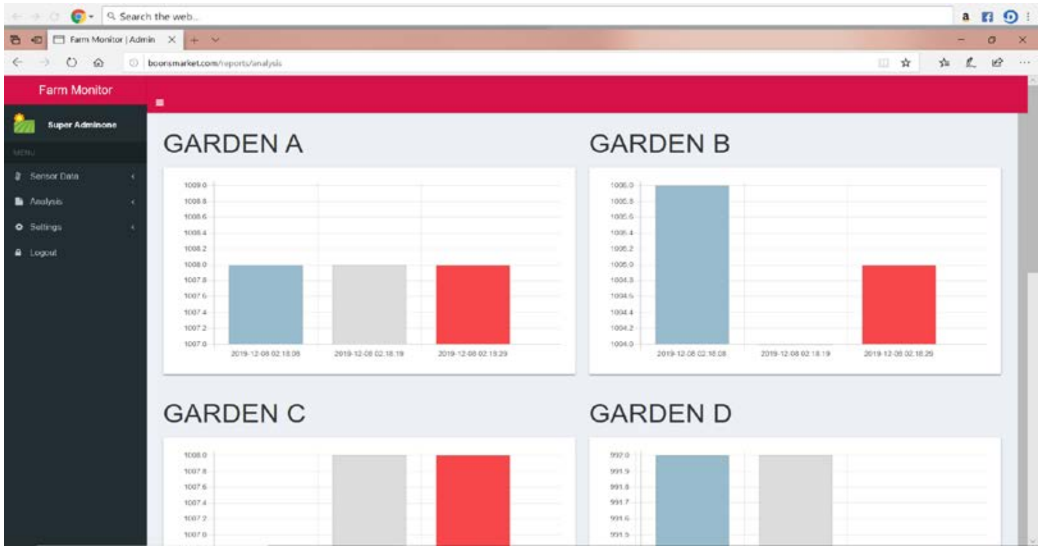
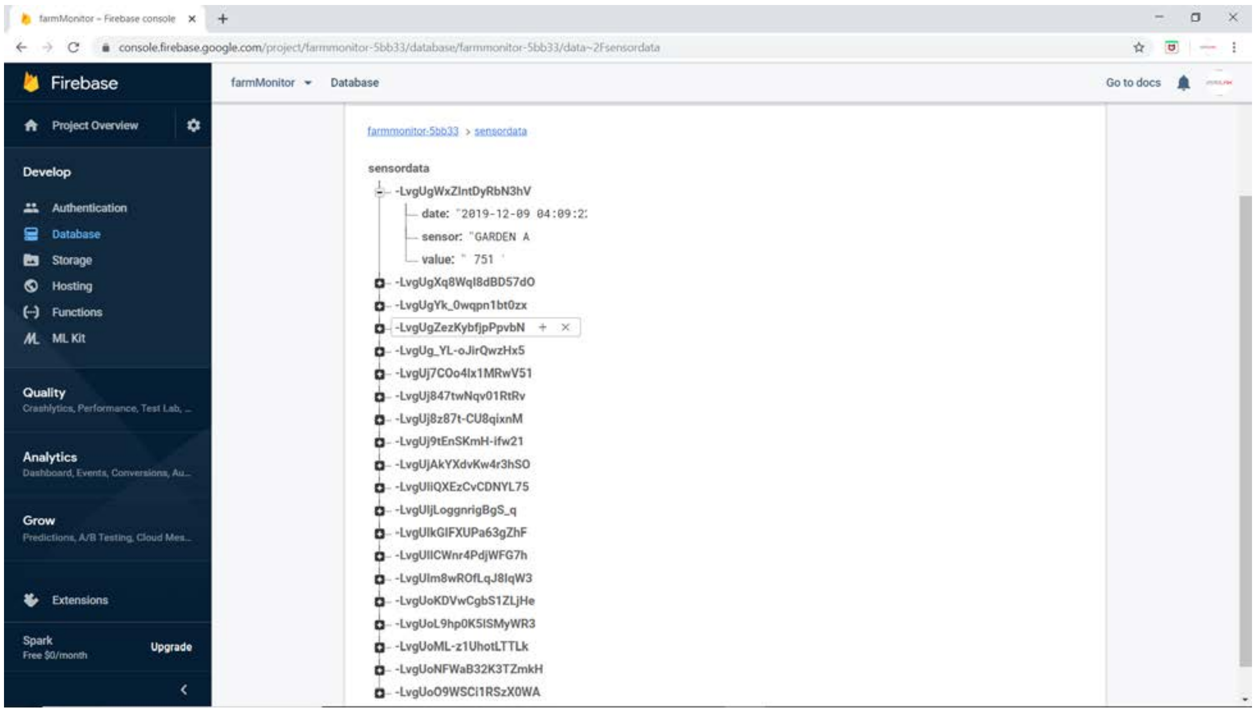
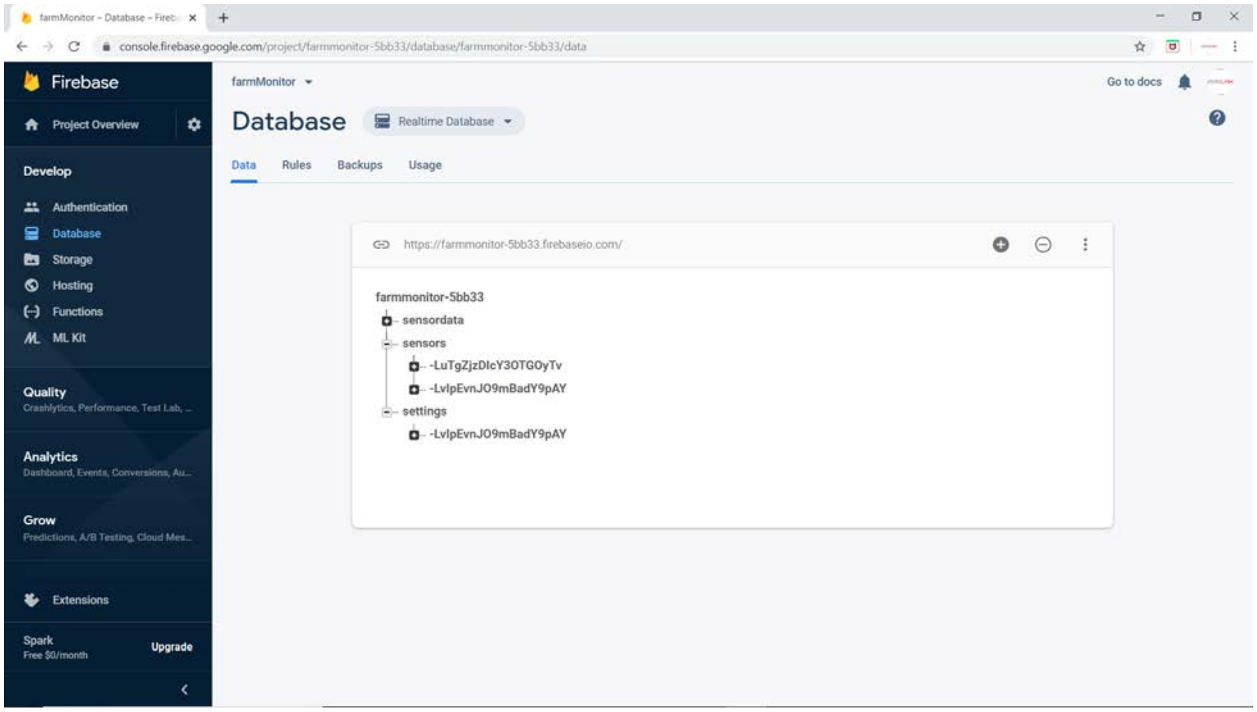


Figure 17: Individual garden reports over time through the Web API


farmMonitor – Firebase console
console.firebase.google.com/project/farmmonitor-5bb33/database/farmmonitor-5bb33/data~2Fsensordata
Firebase
Project Overview
Develop
Authentication
Database
Storage
Hosting
Functions
ML Kit
Quality
Crashlytics, Performance, Test Lab, ...
Analytics
Dashboard, Events, Conversions, Au...
Grow
Predictions, A/B Testing, Cloud Mes...
Extensions
Spark
Free $0/month
Upgrade
farmMonitor
Database
Go to docs
farmmonitor-5bb33 > sensordata

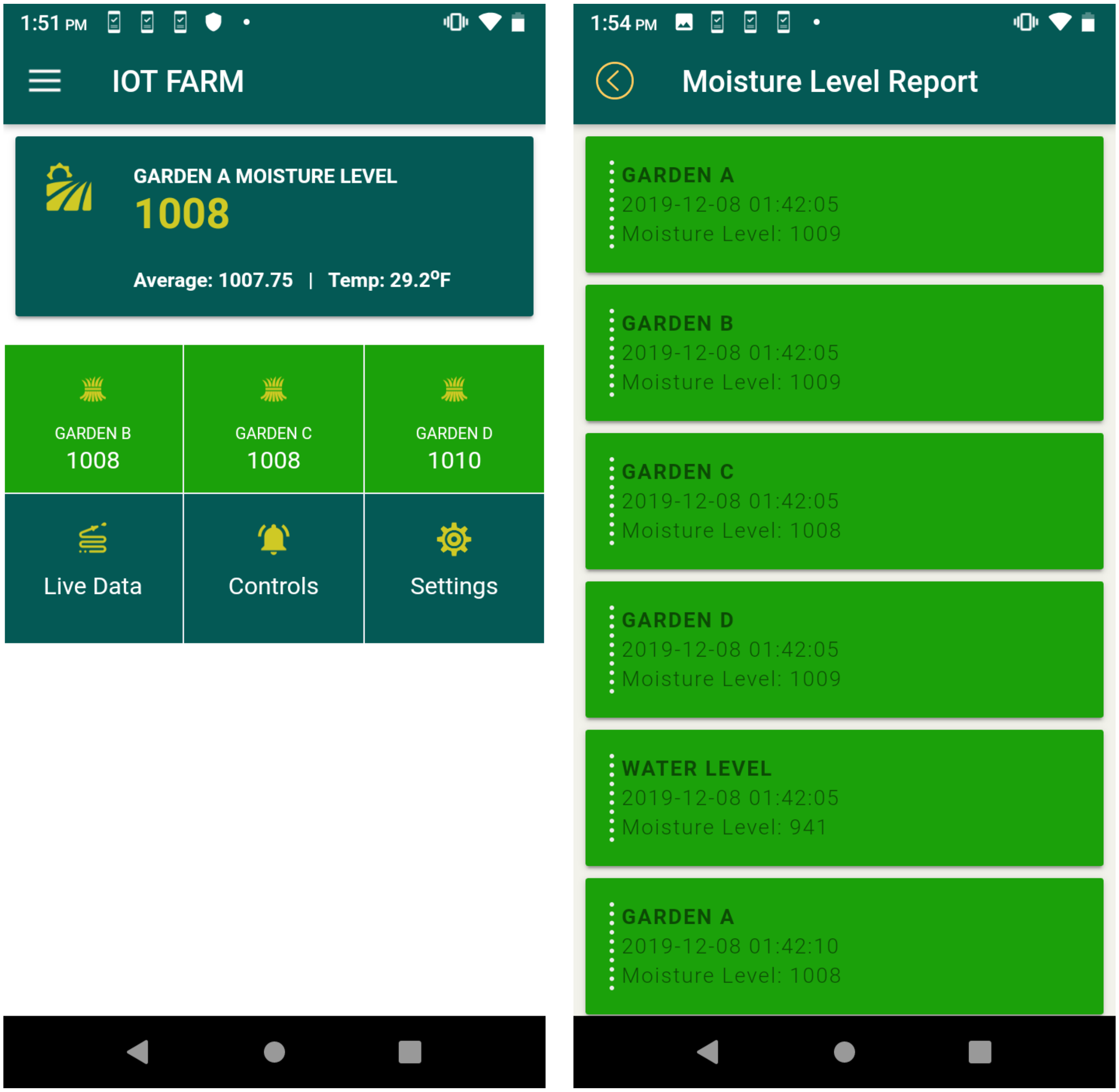
```
sensordata
  -LvgUgWxZIntDyRbN3hV
     date: "2019-12-09 04:09:2:
     sensor: "GARDEN A
     value: " 751 '
  -LvgUgXq8WqI8dBD57dO
  -LvgUgYk_0wqpn1bt0zx
  -LvgUgZezKybfjpPpvbN
  -LvgUg_YL-oJirQwzHx5
  -LvgUj7COo4Ix1MRwV51
  -LvgUj847twNqv01RtRv
  -LvgUj8z87t-CU8qixnM
  -LvgUj9tEnSKmH-Ifw21
  -LvgUjAkYXdvKw4r3hSO
  -LvgUIiQXEzCvCDNYL75
  -LvgUIjLoggnrigBgS_q
  -LvgUIkGIFXUPa63gZhF
  -LvgUIICWnr4PdjWFG7h
  -LvgUIm8wROfLqJ8IqW3
  -LvgUoKDVwCgbS1ZLjHe
  -LvgUoL9hp0K5ISMyWR3
  -LvgUoML-z1UhotLTTLk
  -LvgUoNFWaB32K3TZmkH
  -LvgUoO9WSCi1RSzX0WA
```

Figure 18: How the data is stored in the Firebase™ database


farmMonitor – Database – Fireb
console.firebase.google.com/project/farmmonitor-5bb33/database/farmmonitor-5bb33/data
Firebase
Project Overview
Develop
Authentication
Database
Storage
Hosting
Functions
ML Kit
Quality
Crashlytics, Performance, Test Lab, ...
Analytics
Dashboard, Events, Conversions, Au...
Grow
Predictions, A/B Testing, Cloud Mes...
Extensions
Spark
Free $0/month
Upgrade
farmMonitor
Go to docs
Database
Realtime Database
Data
Rules
Backups
Usage
https://farmmonitor-5bb33.firebaseio.com/

```
farmmonitor-5bb33
  sensordata
  sensors
     -LuTgZjzDIcY3OTGOyTv
     -LvIpEvnJO9mBadY9pAY
  settings
     -LvIpEvnJO9mBadY9pAY
```

Figure 19: Nodes of data storage in the Firebase™ database



1:51 PM

IOT FARM

GARDEN A MOISTURE LEVEL
1008
Average: 1007.75 | Temp: 29.2°F

GARDEN B 1008
GARDEN C 1008
GARDEN D 1010

1:54 PM

Moisture Level Report

GARDEN A
2019-12-08 01:42:05
Moisture Level: 1009

GARDEN B
2019-12-08 01:42:05
Moisture Level: 1009

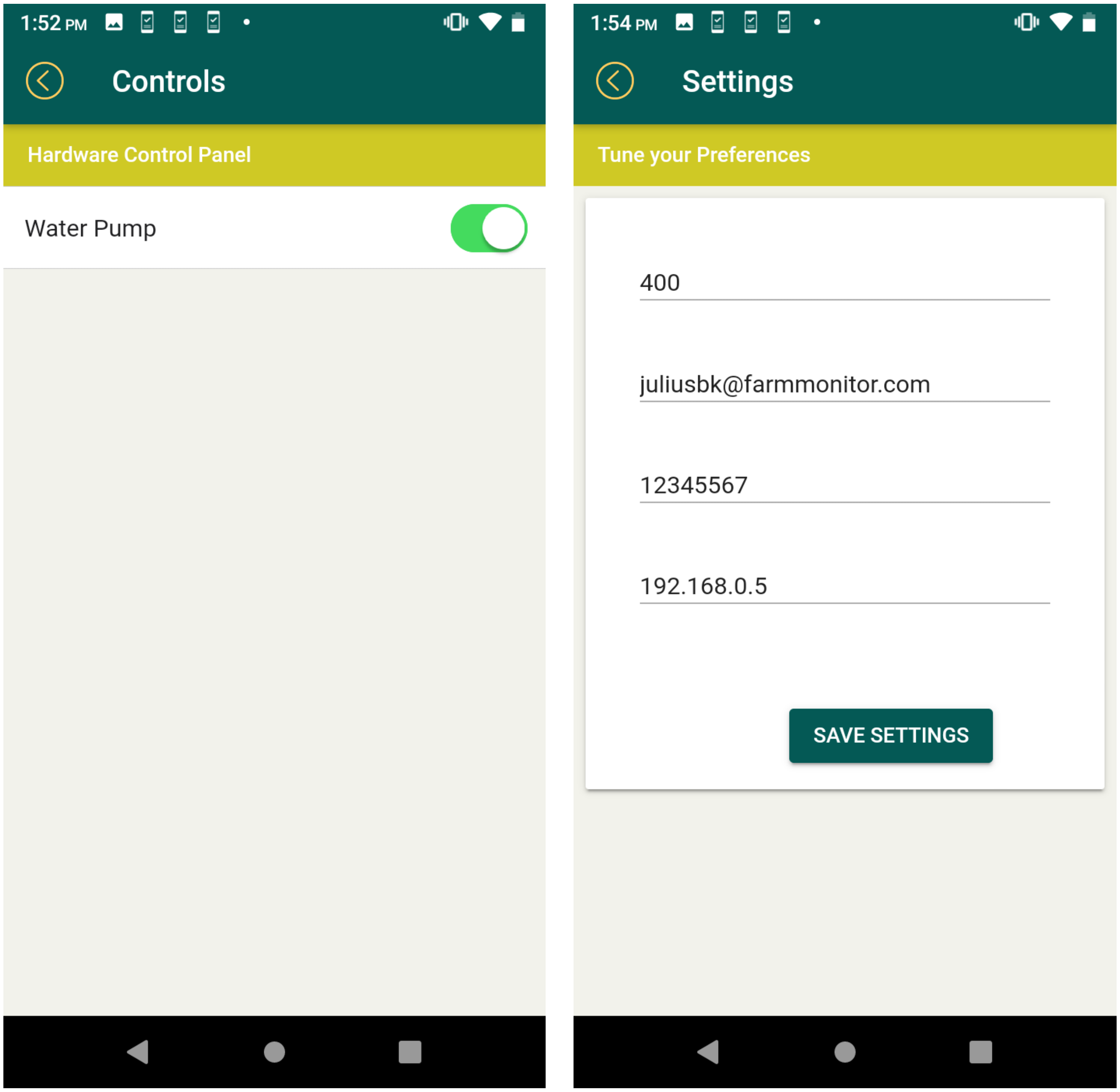
GARDEN C
2019-12-08 01:42:05
Moisture Level: 1008

GARDEN D
2019-12-08 01:42:05
Moisture Level: 1009

WATER LEVEL
2019-12-08 01:42:05
Moisture Level: 941

GARDEN A
2019-12-08 01:42:10
Moisture Level: 1008

Figure 20: *(L-R)* Mobile App Home and Report pages of the IoT Smart Farm system



Water Pump

400

juliusbk@farmmonitor.com

12345567

192.168.0.5

Figure 21: *(L-R)* Implemented Controls on the IoT Smart Farm to turn on water pump, and Settings to change the desired water moisture levels plus the email address, phone number, and IP address for notifications

Conclusion

There are no limitations on how many connections and sensors can be set up in an IoT agricultural smart system.

The discussed model and IoT system design explored in this project can be implemented in a small setting like a backyard garden, in-door garden and even on a large-scale farm. All the components and design can be scaled to meet the user's needs. However, it should be noted that the smart farming (as an application of IoT technology) has not been given much attention and sufficient research. This is basically happened because many of the farming operations usually occur in the remote areas. But given more attention especially from countries with much agriculture exports, the smart farming has the potential to revolutionize the way farms operate and hence to boost the food market around the world.

Issues for feature studies in IOT smarting farming

With the constant change in global climate, farmers and the agricultural industries are in need to upscale their production methods in order to meet market demand. Employing the IoT technology is one of the key guaranteeing the developers in achieving this goal. For such a small-scale smart farm discussed and demonstrated in this project, we discovered how a simple IoT Smart Farm model can be designed and implemented. For further research and study, we can develop and integrate an automatic mode for the system. That is a farmer who can set his farms in an "Auto-Mode" that allows him not only monitor the farm and collect data, but also take automatically some preset actions basing on the collected data. Furthermore, the system can be incorporated with USSD functionality to implement an offline action control mechanism for farm management. This is relevant to farmers in remote areas where the internet connection is weak, or smart phones are not in use or not preferred.

References

John Nussey. Arduino for Dummies, 2nd edition. John Wiley & Sons, 2018.

Massimo Banzi (*co-founder of the Arduino Project*) & Michal Shiloh. Getting Started with Arduino: The Open Source Electronics Prototyping Platform, 3rd edition. Maker Media, Inc., 2013.

John Boxall. Arduino Workshop: A hands-on introduction with 65 projects. William Pollock No Starch Press, 2013.

Neil Smyth. Android Studio Essentials 2nd edition. eBook Frenzy, 2015.

International Telecommunication Union Series Y.2060: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks. – An Overview of the Internet of Things. ITU-T, June 2012.

Sites

Engineers garage, n.d. https://www.engineersgarage.com/. Accessed September 15, 2019.

UWE PDT Soil Moisture sensors, April 03, 2018. https://uwearduino.wordpress.com/. Accessed September 15, 2019.

IoT Agenda, n.d. https://internetofthingsagenda.techtarget.com/. Accessed September 15, 2019.

Arduino. Arduino UNO Wi-Fi Rev2, n.d. www.arduino.cc. Accessed October 29, 2019.

Wiring, what is a breadboard? N.d. http://wiring.org.co/learning/tutorials/breadboard/. Accessed November 12, 2019.

GHCC, Soil Moisture. December 30, 1999. https://weather.msfc.nasa.gov/landprocess/. Accessed September 20, 2019.

ESP8266 AT Instruction Set Version 1.5.4. Espressif Systems IOT Team, 2016. Available online: https://www.itead.cc/wiki/images/5/53/Esp8266_at_instruction_set_en_v1.5.4_0.pdf

HACKADAY, n.d. https://hackaday.com/. Accessed September 20, 2019.

Beechamresearch, Smart Farming Sales Brochure 2017. http://www.beechamresearch.com/. Accessed September 20.2019.
Food and Agriculture Organization (FAO) of the United Nations, September 23, 2009. http://www.fao.org/news/story/en/item/35571/icode/. Accessed October 10, 2019.
Google Inc. Firebase™ database, n.d. http://firebase.google.com/products. Accessed October 25, 2019.
IoT Analytics, November 2019. https://iot-analytics.com/. Accessed November 22, 2019.

[*] **JULIUS KAKOOZA KAWEMBA** is an international graduate student at Rivier University. He has successfully completed his Master's degree in Computer Science in fall 2019. Prior to that he had obtained a B.S. in Computer Science & Mathematics from Uganda Martyrs University. This work (based on his final capstone project in the CS699A Professional Seminar in fall 2019) has been inspired by the desire to improve agricultural management methods and IT systems of farmers not only in Uganda, but also in the entire world.

Appendix

Arduino code: //Arduino UNO Wi-Fi Rev2

```
#include <SPI.h>
#include <WiFiNINA.h>

char ssid[] = "BK"; //your network ssid
char pass[] = "";  //your Network ssaid password. For this project I used an open Network
int keyIndex = 0;

int status = WL_IDLE_STATUS;
char farm[] = "boonsmarket.com";

String postData;
String alertData;
String postVariable = "moisture= ";
String readString;

WiFiClient client;
//WiFiClient client2 = server.available();
WiFiClient client3;
WiFiServer server(80);

/*
  ----------------------------------------------
*/

      //Delays
#define WATERDELAY 750
#define WATERPOSTDELAY 50000

      //Moisture sensors
```

```
#define soilSensor1 A1
#define soilSensor2 A2
#define soilSensor3 A3
#define soilSensor4 A4
#define soilSensor5 A5

       //Water level sesnor
#define H2Olevel A0

        //Indicator LEDs
#define greenLED4 9
#define redLED4 8
#define greenLED3 7
#define redLED3 6
#define greenLED2 5
#define redLED2 4
#define greenLED1 3
#define redLED1 2

        //Water pump
#define H2OPump  13
#define LEDH2OPump  12
//int val;

        //Level of presence of Moisture in the soil set.
        //This can vary depending on the sensor
int goodMoisture = 400;

void setup(){

  pinMode(greenLED1, OUTPUT);
  pinMode(redLED1, OUTPUT);
  pinMode(greenLED2, OUTPUT);
  pinMode(redLED2, OUTPUT);

  pinMode (H2OPump, OUTPUT);
  pinMode (LEDH2OPump, OUTPUT);

  Serial.begin(9600);

  //wifi setup now

    while (status != WL_CONNECTED) {
      Serial.print("Attempting to connect to Network named: ");
      Serial.println(ssid);
      status = WiFi.begin(ssid, pass);
      delay(2000);
    }
server.begin();

  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
  IPAddress ip = WiFi.localIP();
  //IPAddress gateway = WiFi.gatewayIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  //end wifi

}
void loop(){

  int sensorValue1 = analogRead(soilSensor1);
  int sensorValue2 = analogRead(soilSensor2);
  int sensorValue3 = analogRead(soilSensor3);
  int sensorValue4 = analogRead(soilSensor4);
  //int sensorValue5 = analogRead(soilSensor5);

  int H2OVal = analogRead(H2Olevel);


  //DATA TO SEND TO SERVER
   postData = "&GARDEN_A= " + String(sensorValue1) + " &GARDEN_B= " +
String(sensorValue2)+
   " &GARDEN_C= " + String(sensorValue3)+ " &GARDEN_D= " + String(sensorValue4)+ "
&WATER_LEVEL= " + String(H2OVal);

   alertData = "&GARDEN_A= " + String(sensorValue1);

  Serial.print("GARDEN A = " );
  Serial.println(sensorValue1);

  Serial.print("GARDEN B = ");
  Serial.println(sensorValue2);

  Serial.print("GARDEN C = ");
  Serial.println(sensorValue3);

  Serial.print("GARDEN D = ");
  Serial.println(sensorValue4);
```

```
 // Serial.print("Sensor 5 = ");
  //Serial.println(sensorValue5);

   /***
          *
          * Water level output
          */
if (H2OVal <= 90){
  Serial.println("Water Lever: Empty");
}
  else

    if (H2OVal > 100 && H2OVal <= 250){
      Serial.println("Water Level: Medium");
    }
    else
      if (H2OVal > 500 ){
        Serial.println ("Water Level: High");

      }
      delay(100);
/***
 *
 *
 * FARM...... A ......
 */
  if (sensorValue1 <= goodMoisture){
    digitalWrite(greenLED1, HIGH);
    digitalWrite(redLED1, LOW);

  }
  else{
    digitalWrite(greenLED1, LOW);
    digitalWrite(redLED1, HIGH);

  }
  delay(250);
  /**
   * FARM B
   */
  if (sensorValue2 <= goodMoisture){
    digitalWrite(greenLED2, HIGH);
    digitalWrite (redLED2, LOW);
  }
  else {
    digitalWrite(greenLED2, LOW);
    digitalWrite (redLED2, HIGH);
  }
  delay(100);

/***
 * FARM ......C.....
 */
  if (sensorValue3 <= goodMoisture){
    digitalWrite(greenLED3, HIGH);
    digitalWrite (redLED3, LOW);
  }
  else {
    digitalWrite(greenLED3, LOW);
    digitalWrite (redLED3, HIGH);
  }
delay(100);

/**
 * FARM D
 */
if (sensorValue4 <= goodMoisture){
    digitalWrite(greenLED4, HIGH);
    digitalWrite (redLED4, LOW);
  }
  else {
    digitalWrite(greenLED4, LOW);
    digitalWrite (redLED4, HIGH);
  }

    delay(1);

     /// Water Pump SERVER CONTROL LOGIC

   WiFiClient client2 = server.available();
  if (client2)
  {
    //Serial.println("new client");

    while (client2.connected())
    {
      if (client2.available())
      {
        char c = client2.read();
     if (readString.length() < 100)
     {
      readString += c;
      Serial.write(c);
      if (c == '\n') {
        client2.println("<a href=\"/H\"\">Turn Water Pump ON</a>");
        client2.println("<br />");
        client2.println("<br />");
        client2.println("<a href=\"/L\"\">Turn Water PumP OFF</a><br />");

        delay(1);

        if(readString.indexOf("?H") < 0)
        {
          digitalWrite(H2OPump, HIGH);
           digitalWrite (LEDH2OPump, HIGH);
          delay(1);
        }
        else{
          if(readString.indexOf("?L") < 0)
          {
            digitalWrite(H2OPump, LOW);
             digitalWrite(LEDH2OPump, LOW);
            delay(100);
          }
        }
        readString="";

        delay(100);
        client2.stop();
        //Serial.println("client disonnected");
     }
    }
   }
  }
  }
    delay(1);

//CHECK CONNECTION TO SERVER AND SEND
 if (client.connect(farm, 80)) {

    Serial.print(postData);
    //Serial.print(alertData);

    client.println("POST /record HTTP/1.1");
```

```
      client.println("Host: boonsmarket.com");
      client.println("Content-Type: application/x-www-form-urlencoded");
      client.print("Content-Length: ");
      client.println(postData.length());
      client.println();
      client.print(postData);
  }

  if (sensorValue1 < goodMoisture && client3.connect(farm, 80)){
    Serial.print(alertData);
    client3.println("POST /alert HTTP/1.1");
      client3.println("Host: boonsmarket.com");
      client3.println("Content-Type: application/x-www-form-urlencoded");
      client3.print("Content-Length: ");
      client3.println(alertData.length());
      client3.println();
      client3.print(alertData);
  }

   if (client.connected()) {
    client.stop();
  }
delay(WATERPOSTDELAY);
}
```