SANTA CLARA UNIVERSITY

Department of Computer Science and Engineering and Web Design and Engineering

I HEREBY RECOMMEND THAT THE THESIS BE PREPARED

UNDER MY SUPERVISION BY

Ryan Le, Erik Mitchell, Mark Castillo

Entitled

DOMINICAN REPUBLIC GREENHOUSE AUTOMATION

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

BACHELOR OF SCIENCE

IN

COMPUTER SCIENCE AND ENGINEERING

and

WEB DESIGN AND ENGINEERING

Sean Choi
Sean Choi (Jun 15, 2023 16:43 PDT) Jun 15, 2023

Thesis Advisor(s) date

N. Ling
N. Ling (Jun 15, 2023 16:45 PDT) Jun 15, 2023

Department Chair(s) date

DOMINICAN REPUBLIC GREENHOUSE AUTOMATION

By

Ryan Le, Erik Mitchell, Mark Castillo

Submitted to

Department of Computer Science and Engineering

And

Web Design and Engineering

Of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements

for the degree of

Bachelor of Science in Computer Science and Engineering

and

Web Design and Engineering

Santa Clara, California

2022-23

ABSTRACT

The Dominican Republic's Instituto Politécnico Loyola (IPL) needs a wirelessly connected network of sensors for their greenhouses. Our project uses long-range (LoRa) communication to receive data from sensors within the greenhouse on the recorded temperature, humidity, soil moisture, and light values; stores and organizes these measurements in a database and backup text file; and displays appropriate recordings on an interactive application. We collaborated with a team of two General Engineering majors, with our focus on data retrieval and display functionality. Aside from basic data senders used solely for testing purposes, our final design serves as a central hub remotely accessible from outside the greenhouse: a Raspberry Pi serving as processing functionality receives information sent over LoRa, logs it to an internal database, and imports it into an HTML application displayed on an attached touchscreen tablet. While designed with specifically IPL’s greenhouses in mind, our system serves as a framework for expansion outside of IPL and eventually to remote farmlands in the global south. Our architecture was designed to use frugal materials for our project to be practically implemented.

ACKNOWLEDGEMENTS

We would like to thank Allan Baez Morales and the Frugal Innovation Hub team for connecting us and facilitating our communication with IPL and for their consistent support of our project team.

We would also like to thank Claudia Diaz and the IPL for giving us the opportunity to work on this project and for providing us with valuable feedback and encouragement.

We would also like to thank Dr. Sean Choi, our project advisor, who guided us through our research and development throughout the year.

We would also like to thank Henry Kayser and Pedro Alvarenga and their advisor Dr. JAK who worked in collaboration with us to deliver a comprehensive solution.

Lastly, we would like to thank Jean Tsai and the School of Engineering for providing us with the necessary resources and funding for our project.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Problem

The Dominican Republic is among the largest agricultural sectors in Latin America, growing to be one of the greatest exporters of organic, quality, and fair trade products in the world. Previously, agriculture supported most of the country's economy, contributing to about 11% of the GDP with about 15-17% employment in 2001 [1]. However, studies from 2020 show that these numbers are declining, with its contribution to the country's GDP dropping to about 6.04%, and employment to 8.824% [2]. This downturn of the country's agricultural production can be greatly attributed to natural disasters; cyclones, storms, and droughts add further strain to farmers' existing dilemmas of poor infrastructure and a lack of resources for improvement [3-4]. As a result, the country is currently focusing efforts towards the innovation of existing agricultural practices, and in particular is encouraging utilizing greenhouse for greater control over crops' growing conditions.

1.2 Objective

Our objective is to create an automated sensor system connected to a remote central unit displaying information on an interactable application. Due to the cost of implementing and maintaining Wi-Fi and lithium ion batteries, more cost-efficient and eco-friendly solutions are crucial components of this project. Additionally, we designed our solution with a greenhouse setting in mind. While the overall problem statement focuses on agricultural practices in general, we decided to appropriately scale down our testing environment to further promote the popularity of greenhouse farming in the Dominican Republic.

1.3 Solution

Our solution is a wirelessly connected network of soil quality sensors that store soil temperature and humidity levels. This network uses LoRa communication to ensure long-distance, reliable data transmission and solar panels to power the system efficiently. As a part of this network, an interactable application would aim to streamline the soil data and organize it in a way that can be displayed remotely. We aim to accomplish this through the creation of a dashboard that would include the corresponding temperatures and soil data for both individual green houses.

Chapter 2: Background

2.1 Research

Our client, Instituto Politécnico Loyola (IPL), contacted the Frugal Innovation Hub at SCU with their project: they have multiple greenhouses on their campus in the Dominican Republic, which are prime testing grounds for new technologies that can later be shared with the local farmers who are often required to use low-budget, manual solutions. If IPL could create and incorporate advanced yet cheaply reproducible systems in their greenhouses, they could refine and share them with the local farmers, significantly increasing their crop yield and decreasing the amount of physical labor needed.

Greenhouses are currently one of the more reliable practices because they allow farmers to set and cultivate desired environments for their crops. However, without the proper technology, maintaining a greenhouse is a difficult undertaking requiring long-term commitment and physical capability. Providing adequate support to these farmers is crucial in the recovery of agricultural practices in the Dominican Republic.

2.2 Hardware Used

Raspberry Pi 4 (8gb ram)

The Raspberry Pi 4 is a miniature computer capable of intense processing and projection to an external graphical interface [6].

Raspberry Pi 4 7 inch Touchscreen

This tablet, which comes with an adapter board and touchscreen display, connects to the Raspberry Pi 4 [6]. Its portability and intuitive control system provides user access to the application.

Arduino Uno

The Arduino Uno is a microcontroller board connected to the soil sensors. The board sends sensor data to the Raspberry Pi when prompted [7]-[12].

SX1278 LoRa Module 433MHz Ra-02

The LoRa module allows for LoRa communication over 433MHz between the Raspberry Pi and the Arduino Uno boards [7], [9]-[11].

433MHz Antenna

Each LoRa module connects to a 433MHz compatible antenna for long-ranged communication [7], [9]-[11].

2.3 Software Technologies Used

Python

Python is a high-level programming language. The code for the data transport system, storage of sensor information into local storage, and parsing of sensor data files is all written in Python [10], [13].

SQL

SQL is a programming language for storing and processing information in a database and thus was used for our sensor data tables [13]-[15].

Javascript

JavaScript is a programming language used primarily for adding client side functionality. These scripts are instrumental in the retrieval of data from the database for display on the HTML page [16].

HTML

HTML is a markup language used primarily. for constructing web pages [16], used in this project for the structure of the interactive application.

CSS

CSS is a stylesheet language used to alter the appearance of a webpage [16]. The visual presentation of the application is completely controlled by the external CSS file.

Chart.js

Chart.js is a JavaScript library that can be locally installed (rather than requiring an internet connection for access by a CDN) that generates graphs for HTML pages [17]. It creates the charts for each of the sensor values' subpages.

Chapter 3: Use Cases

Scenarios for individual use and commercial use of our product are crucial to planning our experience design. We considered a wide variety of potential users to account for numerous applications of our solution. This also allowed us to cultivate extra ideas that we would not have otherwise considered, which enhanced our final product with vital features to fulfill the requests of IPL and countless future clients.

3.1 User Stories

The first conceptual user we considered was our original clientbase: the students, faculty, and other residents of IPL. With the originally provided design constraints, we hypothesized that the primary use of our system would be monitoring seasonal greenhouse crops. They access the information through a single system located a fair distance from the sensors. Thus, they need a communication method that maintains a strong connection from a long distance and a device to act as a central hub for data collection and analysis.

3.2 Client Scenarios

Our client proposed the project with the intent of being able to help the greater farming community within the Dominican Republic in order to improve their agricultural practices and increase their production output. The capabilities of the system would improve the accuracy of crop monitoring, allowing all workers on the farm to have a better understanding of the different factors that affect growth and output. In addition to farmers, this would also greatly benefit ecologists, as it would allow them to record and analyze data for various types of produce, providing greater understanding for fields of study such as ideal environment and growth conditions, optimal harvesting time, ecological effects, and countless more.

Chapter 4: Requirements

In order for our solution to fulfill the use cases of our users, there are certain requirements (both functional and non-functional) that the system must meet.

4.1 Functional Requirements

The functional requirements of our project describe the static goals that are key to the entire system's operation.

- The sensor data must be accessible through interactive application.
- The sensor data should be accessible at a long range (15-20 km) from the greenhouse location.
- The sensor data must be saved to an external hardware storage.
- The interactive application must be accessible through a mobile device.

4.2 Non-Functional Requirements

Non-functional requirements are dynamic goals that describe how our system should perform.

- The data transport of sensor data must be in real time.

- The system must have data transport that is resistant to interruption with no packet loss.
- The user interface of the application must be intuitive.
- The application must be responsive.
- The application must support Spanish and English.

4.3 Design Constraints

Design constraints limit the design of the solution in order to fit the desires of the client and work in its intended environment.

- The system must be able to operate without an internet connection.
- The system must be able to operate without wifi.
- The system must be energy efficient and run on a local battery.
- The system hardware must collectively be inexpensive.
- The system must have a compact design.

Chapter 5: Conceptual Model

5.1 Overall Architecture

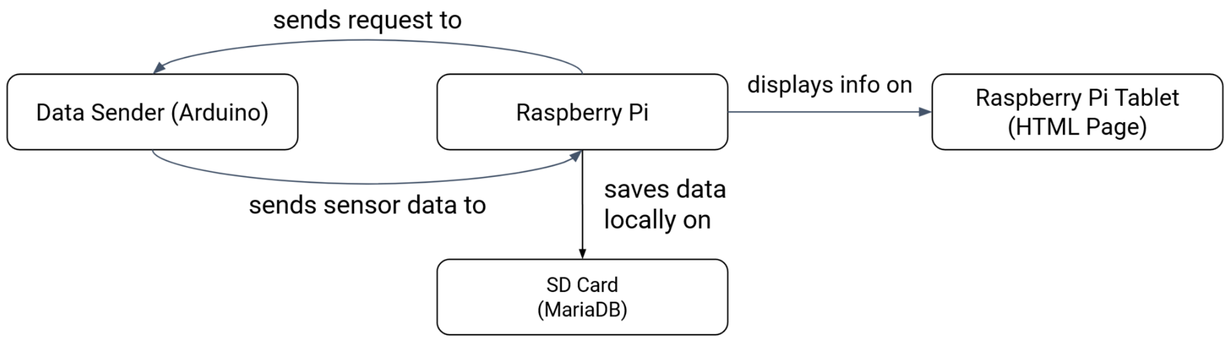


Fig. 1
Overall architecture of the proposed solution.

The complete solution between our group and the group we worked alongside with can tentatively be split into two sections: data sending and data reception. Our group is focused on the aspects regarding reception of data observed and sent by the sensors. The sensors “wake up” to a signal from the Raspberry Pi and send the current readings. The Pi unpackages the data and stores it locally in MariaDB with each row corresponding to a specific reading and each column as a separate measurement. Then, the data is converted into a JSON object to send to the application to be displayed in an intuitive, interactive format, with overall readings and trend charts for the past 24 hours.

5.2 LoRa Protocol

In order to ensure secure and stable data transfer in the network, we designed a synchronous, client-server communication protocol. The gateway module, which in our testing we used a Raspberry Pi, acts as the client by requesting sensor data from one or more nodes. The node(s), represented by Arduinos in our testing, are placed within the greenhouse(s) and remain in sleep mode until prompted by a signal received from the gateway. Figs. 2a and 2b explain these protocols on the different types of modules.

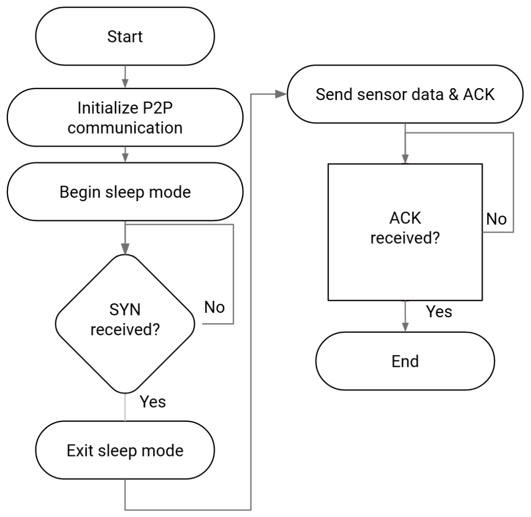


Fig. 2a
LoRa node protocol, using SYNs and ACKs to exit sleep mode and send data.

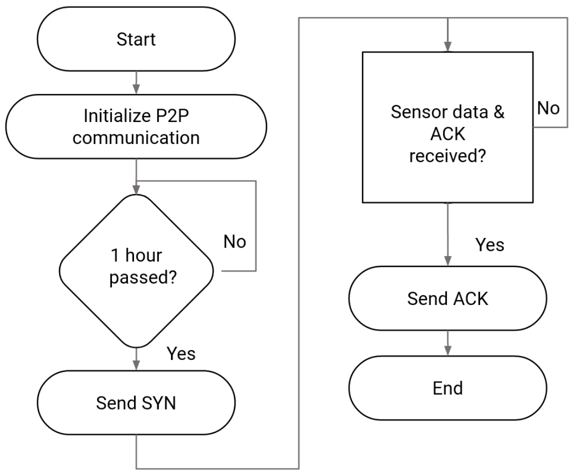


Fig. 2b
LoRa gateway protocol, using SYNs and ACKs to retrieve data every hour.

Fig. 2
LoRa Protocol Flowcharts

The gateway module engages with synchronous communication with each node after each hour has passed. This ensures enough data is collected to make analysis while conserving energy and processing costs. The communication protocol is designed to enable communication between multiple nodes such that many sensors can be placed within a single greenhouse or sensors can be placed within many greenhouses. In order to differentiate themselves and ensure proper data is being sent, nodes add checksums and their ids for the gateway to process.

5.3 Raspberry Pi Framework

The Raspberry Pi itself uses an SX1278 LoRa Module, which is directly wired to the raspberry pi, along with a 7 inch display. The casing for the system safely stores the hardware in an efficient way, and allows the user to place the system in an office desk or shelves.

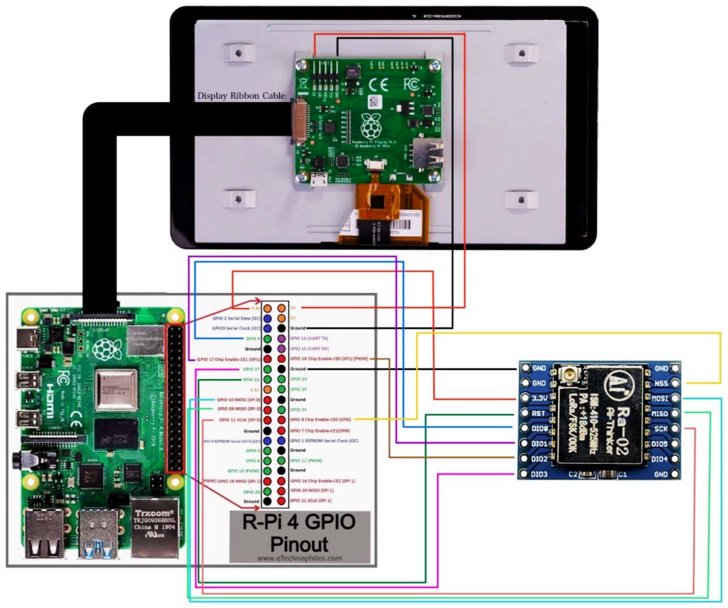
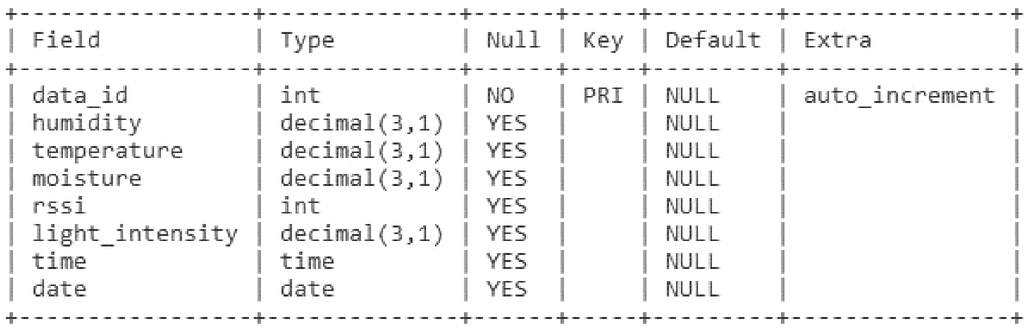


Fig. 3
Wiring guide for the Raspberry Pi, tablet, and LoRa chip.

The system is designed to operate on a Raspberry Pi 4. This allows data to be stored locally and avoid relying on WiFI to communicate with an external server. The database is created using MariaDB, which would hold all the collected data, in addition to information on LoRa signal strength and timestamps of when data is being received. Our database would be stored on the raspberry pi itself, along with all the necessary process and handling protocols for LoRa and the data.

5.4 Database Management



```
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| data_id         | int          | NO   | PRI | NULL    | auto_increment |
| humidity        | decimal(3,1) | YES  |     | NULL    |                |
| temperature     | decimal(3,1) | YES  |     | NULL    |                |
| moisture        | decimal(3,1) | YES  |     | NULL    |                |
| rssi            | int          | YES  |     | NULL    |                |
| light_intensity | decimal(3,1) | YES  |     | NULL    |                |
| time            | time         | YES  |     | NULL    |                |
| date            | date         | YES  |     | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
```

Fig. 4a

The Raspberry Pi's internal MySQL database structure, with each sensor value stored along with the automatically generated Data ID, RSSI, time, and date.

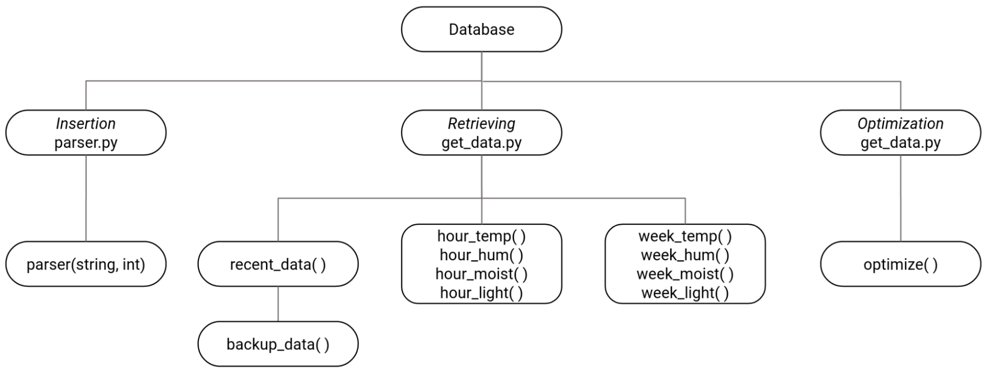


Fig. 4b

Chart detailing the flow of database management, with three different branches for inserting data into the database, retrieving data from the database, and optimizing search times.

The database was constructed internally using MariaDB, to allow data to be stored locally on the system. Entries are inserted through our parser function which takes in two values: the data received through LoRa in a string format and the signal strength as an integer. The string is then split and stored into their respective variables within the function, along with the time and date as measured by the Raspberry Pi system itself. The variables are then entered into their corresponding columns (humidity, temperature, soil moisture, light intensity, date, and time) within the database. Each entry is assigned a unique data ID, which is used for internal organization.

Data is retrieved from the database using a series of functions for different capabilities within the application. The get_data() function selects the most recent entry based on the time and date, the program also has several functions hourly and weekly for each of the monitored data.
The data being packaged into a json object and being written into a json file. At the same time, these entries are also being written into a text file which is being saved on the SD card in the Pi.

Lastly, the program includes optimization methods to avoid the possibility of long search times due to the growing size of the database; any data prior to the last seven days gets removed from the database and written into the aforementioned text file.

5.5 Application Structure

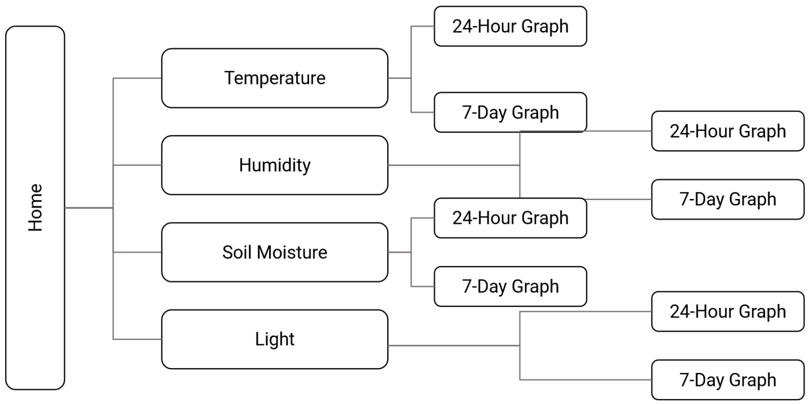


Fig. 5

The general application structure, allowing users to read each most recent recorded value on the homepage and viewing in-depth graphs of trends over the past day and week.

The application is a simple website coded in HTML, CSS, and JavaScript. It is designed for visibility on a small screen, which in our case is 7 inches. An overall view of each sensor's most recent chosen measurements (temperature, humidity, soil moisture, and light) along with the timestamp displays on the homepage, and each measurement is clickable to view graphs plotted with recordings over two different time periods: the last 24 hours and the last 7 days.

The JavaScript code imports the provided JSON files as modules and inserts the proper values into their respective attribute locations within the HTML. The locally-installed Chart.js library also imports the JSON files, using them to generate the graphs for each value.

Chapter 6: Design Rationale

6.1 LoRa vs. WiFi

Specification	**LoRa**	**WiFi**
Range	*15-20 km*	139 m^2
Reliability	Some packet loss	*Minimal packet loss*
Efficiency	*Battery-powered*	100-240V per router
Affordability	*$10 per module*	$89+ per router

Fig. 6
Comparison of LoRa vs. WiFi communication, showing that LoRa satisfies the project requirements far better than WiFi.

The planned deployment environment greatly affected our design, including the decision of which communication method to use for data sending. The greenhouse layout at the IPL campus includes a greenhouse that is 38 m away from the main offices and that has no power or wifi access. The infrastructure required to bring a large power supply along with wifi access to this remote area violates our frugal requirement, so LoRa communication was the best option for the solution. We needed to account for some packet loss through a checksum and client-server network architecture.

6.2 User Interface

Application

A GUI is much more accessible than requiring syntax knowledge and program installation on a personal device from each user. Instead of remembering the specific terminal commands to view the values in the database, anyone can power on the tablet and view the relevant data visually. Furthermore, HTML, CSS, and JavaScript do not need to be installed on a new device, and thus the code repository can be copied and replicated on other systems rather than being constrained to systems able to install any necessary dependencies.

The application design centers around the knowledge of the tablet's size and controls. The information is large so as to be completely legible on a 7-inch tablet, and the navigation works perfectly with input solely from a touchscreen rather than requiring a mouse or keyboard. Furthermore, CSS styling ensures responsiveness on any screen size, though it is optimized for the smaller screen size.

Data Format

Data extracted from the database for use in the application is in a JSON (JavaScript Object Notation) format for intuitive use in JS code. However, in order to import data from other files, the JS files must be defined as modules within the HTML.

Chapter 7: Testing & Evaluation

7.1 Unit Test: Individual Systems

LoRa Transmission

Each device communicating over LoRa needs its own module. Various libraries exist for LoRa functionality in many different languages; for our purposes, we used the Arduino variant of C to send test data and Python for the Raspberry Pi's reception from the Arduino modules.

As a part of our testing, we sent a Received Signal Strength Indicator (RSSI) value along with the sensor packet data. This value tells how strong the connection is between the node(s) and the gateway. During our testing, we collected 50 different RSSI values that averaged to -97. Although RSSI values are relative to specific manufacturers, -97 generally indicates poor communication connectivity. This is expected considering the hardware and communication protocols implemented. The gateway protocol accepts multiple packets during the time of communication while rejecting any garbage data. While running communication tests, we recorded that garbage data makes up 25% of input from the sensor node.

Data Path Verification

The data was processed through brute-force building of various functions with the goal of optimizing fast data retrieval. The system first establishes a connection to the internal database. Once the connection is made, the parser( ) function splits the transmitted data (formatted in a string), converts each value into floats, then inserts them into their corresponding columns within the table using SQL instructions. When testing on MacOS, the runtime for the insertion using the parser function is about 0.0007. In contrast on the Raspberry Pi, the time to insert into the table is 0.00508 seconds.

For data retrieval, we placed a heavy emphasis on ensuring selection of the most recent entry from the database. The code applies the MAX( ) function to both the date and time and verifies the SQL requests are functioning correctly. The received data is then formatted and assigned to a Python dictionary for each corresponding field. Finally, the code transforms the dictionary into JSON as a .js file, which can be read by the application and used to extract any necessary data. The most time-consuming portion of this process was workshopping the file’s syntax, requiring multiple iterations to finalize a format readable by the application. On MacOS this function takes about 0.00106 seconds to retrieve the most recent data. On the raspberry pi, run time was 0.00981 sec.

After receiving and formatting a single entry using the most optimal method, the next step was to do the same for multiple entries so the application could create graphs out of points from over the last 24 hours and the past 7 days. Our data would then request data within a 24 hour time as well as 7 week time frame. These programs use the AVG( ) function to calculate the average of a given variable (temp, humidity, etc) within a given hour or day. From there, we again format the received data into JS file using the same process as before. On MacOS, the various hourly functions took anywhere from 0.00035-0.00044 seconds to retrieve their specific data. The weekly data, however, took 0.00033-0.000037 seconds. On pi the hourly data was produced between 0.00418 - 0.00505 seconds and the weekly data was retrieved between 0.00294 - 0.00369 seconds.

The optimizer function's main goal is to keep search times as short as possible. Thus, the first step was beginning with small amounts of data, and letting the table grow. Since the data being displayed would be within a 7 day period, the design of the optimization algorithm is to check for data marked 7 days prior to the current date, and then remove it.

7.2 System Test: Component Integration

Data Accuracy & Range Performance

Based on the previous analysis of communication methods between LoRa and Wifi, we acknowledged that packet loss and garbage data are more likely to occur using LoRa. As discussed earlier, we account for these through sending multiple messages and adding checksums.

Information Display

Displaying the sensor values in the HTML application relied on data retrieval from the database and thus could not be tested until this component was implemented. With the JSON files output properly, testing consisted of ensuring the data's proper importation from the external sources into the JavaScript modules by checking for error messages on the browser's console window, then observing any visual changes on the application opened in the browser with a live server. Should all connections be sound, the console window logs no errors, and the application displays number values rather than the coded-in placeholder text.

To analyze the performance of the application, we reloaded the page 10 times in a row on both the Raspberry Pi's Chromium browser and a fresh install of Google Chrome on Windows and compared the load times given by the browser's console. On Windows, the browser finished requesting content in 34.3ms on average; fully loaded the requested HTML and CSS content in 37.7ms on average; and loaded all of the DOM content in 38.2ms on average. The ranges of these values were extremely small: 15ms, 12ms, and 13ms respectively. In contrast, the Raspberry Pi finished requesting content in 380.1ms on average; fully loaded the requested HTML and CSS content in 46.9ms on average, and loaded all of the DOM content in 473.9ms on average, with ranges of 239ms, 29ms, and 293ms. It is clear that the performance of our application on the Windows machine is about 10 times faster than when it is loaded onto the Raspberry Pi. The variation in load times is also significantly larger on the Raspberry Pi than on the Windows machine. Although it is clear that the hardware of the Pi is a harsh limiter to the performance of the software, the load times are short enough to not interfere significantly with the user experience.

7.3 System Test: End-to-End

After incorporating each component into a full system, we did various tests to simulate real-world behavior. While working on the project, we designed a generic system to suit any kind of hardware for the sake of a more general deliverable. In order to deliver a complete solution to our client, however, we made slight adjustments for increased compatibility with the sensors developed by the general engineers. This involved some data conversions and different communication methods due to different electronic modules. For example, we tested our central hub sending a series of floats in a string over C-to-Python communication, but the finished sensor was coded in Python and was only able to communicate with integers. Thus, we edited our code to convert the integers into floats with a tenths-place digit of 0.

7.4 Ethical Considerations

While designing our solution, we kept in mind the major issues of food insecurity and economic development. As of 2019, 9% of the economically active population in the Dominican Republic works under the agricultural sector [1]. These farmers have been struggling over the past 5 years as climate change and fluctuations of the global economy limit their land availability and their returns on labor. We hope that our solution will allow these farmers to gain valuable insight on their crops, improve yields, and increase productivity.

In order to best suit the needs of these farmers, we designed our solution to be frugal and reliable, while also consuming low amounts of energy. Large power consumption has accelerated the effects of climate change, which has impacted the livelihoods of everyone worldwide and particularly workers in underdeveloped countries. Our solution attempts to reduce our contribution to this issue by sending the sensor data with low-power radio communication and powering our system with solar powered batteries.

Chapter 8: Conclusion

8.1 Summary

Our final deliverable consists of a Raspberry Pi tablet display capable of cross-device LoRa communication, MySQL database storage, and interactive data display. The overall cost of the hardware components is significantly cheaper than current commercially available systems, the latter of which also usually require expensive pre-existing infrastructure typically unavailable in the global south. In order to best support IPL's use of the device, aspects such as the values read and retrieval frequency currently optimized for their university greenhouses. Even so, the general framework of both the hardware and software provides the groundwork to potentially iterate, develop, and implement a future revision for widespread usage in agricultural work beyond the scope of a single university.

8.2 Obstacles Encountered

One of the biggest obstacles we faced in the process was designing and building the hardware for our testbed. We were not fully prepared for the level of hardware knowledge required to set up a Raspberry Pi with a tablet and LoRa chip. This lack of experience ended up being an obstacle in the path to reaching our greater area of expertise: software programming. Thus, we needed to spend time studying, researching, and experimenting with techniques such as wiring and soldering before we could dive deeply into coding. This barrier, however, was not a setback, and was in fact useful in rounding out our overall skill and knowledge of computer science as a field. This experience ultimately promoted our learning for not only this project but also for our futures.

8.3 Experience Gained

When we decided to work on this project, we understood that we took on a challenge that would test the limits of our previously held knowledge. We worked with new technologies, such as LoRa communication, and learned new ways to design databases and parse different data types. Along with new software skills, we also gained valuable experience working with hardware modules such as the Raspberry Pi and Arduino Uno.

Working on this project also involved creating academic presentations and formal write-ups, which provided irreplaceable industry experience. Thanks to this, we gained the ability to effectively communicate engineering solutions to customers and experts alike. We also have proper documentation of our process easily accessible to all.

8.4 Future Work

Our goal for this project was to provide a viable solution to the client, IPL, as well as to farmers in the Dominican Republic and other areas. We must build an instruction and data manual for the students at IPL that both give more specific details of the solution. After these are made, we will send them along with the hardware and software of the system to the Dominican Republic where the IPL students can implement the solution in their environment.

Chapter 9: References

[1] Advameg, Inc. Writers. (2023). Dominican Republic - Agriculture. *Nations Encyclopedia* [Online]. Available: https://www.nationsencyclopedia.com/Americas/Dominican-Republic-AGRICULTURE.html

[2] W. Skinner et. al. (2017, Sept. 13-14). Dominican Republic Country strategic opportunities programme. *IFAD Conf.* [Online]. 121. Available: https://www.ifad.org/en/-/document/dominican-republic-country-strategic-opportunities-programme

[3] The World Bank. (2012). Agricultural risk management in the Caribbean: lessons and experiences (2009-2012) (English). *World Bank Group* [Online]. Available: https://documents.worldbank.org/en/publication/documents-reports/documentdetail/842071468010837254/agricultural-risk-management-in-the-caribbean-lessons-and-experiences-2009-2012

[4] A. N. Zamudio and M. Keller. (2013, Feb. 6). Climate Risk Management for Water and Agriculture in the Dominican Republic: Focus on the Yaque Del Sur Basin. *IISD* [Online]. Available: https://www.iisd.org/publications/report/climate-risk-management-water-and-agriculture-dominican-republic-focus-yaque

[5] IPL Staff. (2018). El Instituto Especializado de Estudios Loyola. *IPL* [Online]. Available: https://superior.ipl.edu.do

[6] J. Tyler. (2021, Feb. 22). Official Raspberry Pi 7" Touchscreen Setup Guide. *Howchoo* [Online]. Available: https://howchoo.com/pi/raspberry-pi-touchscreen-setup

[7] IoTDesignPro Authors. (2020, Mar. 26). Wireless Communication between Arduino & Raspberry Pi using LoRa Module SX1278. *IoTDesignPro* [Online]. https://iotdesignpro.com/projects/wireless-communication-between-arduino-and-raspberry-pi-using-lora-module-sx1278

[8] A. Raj. (2019, May 7). LoRa with Raspberry Pi – Peer to Peer Communication with Arduino. *Circuit Digest* [Online]. Available: https://circuitdigest.com/microcontroller-projects/raspberry-pi-with-lora-peer-to-peer-communication-with-arduino

[9] M. S. Hidayat, A. P. Nugroho, L. Sutiarso, and T. Okayasu (2019, Nov.) Development of environmental monitoring systems based on LoRa with cloud integration for rural area. *IIOP Conf. Ser.: Earth Environ. Sci.* [Online]. 355(1). Available: https://iopscience.iop.org/article/10.1088/1755-1315/355/1/012010

[10] L. F. Cerfeda. (2017, Jun. 6). LoRa nodes real-time data plotting using Python for microcontrollers. *Zerynth Tutorials* [Online]. Available: https://zerynth.com/blog/lora-nodes-real-time-data-plotting-using-python-for-microcontrollers

[11] E. Odunlade. (2020, Jan.). Introduction to LoRa - Send Data between Two Arduino Using LoRa. *Electronics-Lab* [Online]. Available: https://www.electronics-lab.com/project/introduction-lora-send-data-two-arduino-using-lora

[12] P. Khatri. (2019, Sept. 12). Arduino Sleep Modes and How to Use Them to Save the Power. *Circuit Digest* [Online]. Available: https://circuitdigest.com/microcontroller-projects/arduino-sleep-modes-and-how-to-use-them-to-reduce-power-consumption

[13] C. M. Agbenyenu. (2021, Feb. 26). How To Store and Retrieve Data in MariaDB Using Python on Ubuntu. *DigitalOcean* [Online]. Available: https://www.digitalocean.com/

community/tutorials/how-to-store-and-retrieve-data-in-mariadb-using-python-on-ubuntu-18-04

[14] A. Khan. (2023, Jan.). How to Install and Setup MariaDB Server on Raspberry Pi. *LinuxHint* [Online]. Available: https://linuxhint.com/install-setup-mariadb-server-raspberry-pi

[15] P. Fromaget. (2023). How to Install MariaDB on Raspberry Pi? (MySQL Server). *RaspberryTips* [Online]. Available: https://raspberrytips.com/install-mariadb-raspberry-pi

[16] W3Schools Authors. (2023). W3Schools Online Web Tutorials. *W3.CSS* [Online]. Available: https://www.w3schools.com

[17] Chart.js Developers. (2023, Apr. 28). Chart.js Documentation. *Chart.js* [Online]. Available: https://www.chartjs.org/docs/latest