Developing an Android application for monitoring and controlling a greenhouse environment

Peter Ridolfi
Mentored by Christopher Rogers and Charlie Pendorf

Introduction

Recently, hydroponic farms have increased in popularity as they are able to grow plants very efficiently while requiring much less land and fewer resources than traditional farming techniques. However, due to the nature of this efficiency, hydroponic farms are also not as easy to maintain. Factors such as water temperature, nutrient levels, ambient temperature, and humidity must all constantly be monitored and manually controlled so that the hydroponics work correctly. As a result, a system to reduce the amount of labor and resources spent on maintaining hydroponics is necessary to increase its cost-effectiveness and worldwide use.

According to Barik (2019), the use of sensors connected to an Arduino for gathering data and motors for controlling temperature and water level was both cheaper and more environmentally sustainable than the normal process of manually checking sensors and adjusting the environment. While the use of an Arduino microcontroller to control such a system has been shown to work, Barik (2019) states that more research is needed to implement an "automated agricultural system." The creation of such a system would be very influential, as it would reduce cost and labor requirements as well as increase crop yield. A study by Jose et. Al (2018) showed that an automated system like this costs around $58.00 to create, which is a much cheaper than a standard hydroponics system. So, the application created by this project will greatly improve the practicality of the hydroponics.

The purpose of this project was to create an Android application that allowed a user to interact with the hydroponic systems remotely, thus eliminating the need for manual maintenance. In order to accomplish this, the app needed to communicate with a server using a small computer called a Raspberry Pi. This allowed the app to both gather data from sensors as well as send commands that were passed to an Arduino microcontroller, and ultimately controlled the hardware inside.

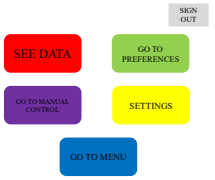
Materials and Methods

This application was created using an interactive development environment (IDE) called Android Studio in the language of Java. The first step to creating the app was to design the physical components that would be seen in the user-interface (UI). These components, such as buttons and text boxes, were designed in .xml files within Android Studio. Before the components were created, the layout of the UI was designed. In order to ensure the best user-experience possible, buttons were first created to allow the user to navigate through the app. These buttons are located both in a central menu activity, as shown in Figure 1, as well as at the top of all other activities, shown in Figure 2.

Materials and Methods (cont.)

Figure 1 (right): The central menu activity with navigation buttons, which is launched when the user logs into the app.

Figure 2 (below): The navigation buttons at the top of the preferences activity in the app.



After creating the main UI elements in the app, the requests to the server were written. When certain events occur in the app, such as the click of a button or the opening of an activity, the app sends an HTTP request to the server, which either returns data to the app or sends new data to the server to be processed. Three main activities were created to accomplish this, which are shown in Figures 3, 4, and 5, respectively. First, a data activity was written that gets live data from the server and displays it to the user. There was also a preferences activity created to allow the user to update parameters for the Arduino to control the hardware automatically. Finally, a manual control option was created whereby the user can directly control the hardware.

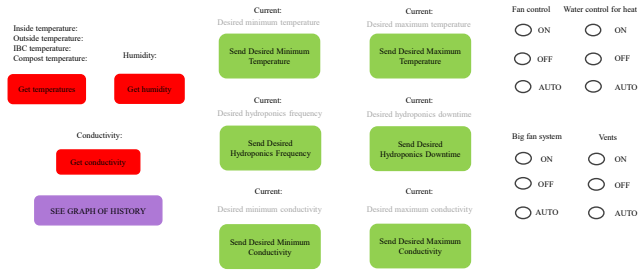


Figure 3 (above): (A) The data activity, where live data can be fetched and graphed by the user. (B) The preferences activity, where the user can send new parameters to the server. (C) The manual control activity, where the user can turn each piece of hardware on and off or allow it to be automatically controlled.

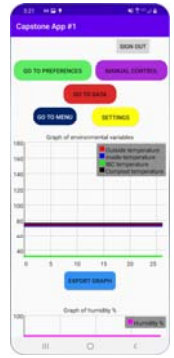
After the requests were written and handled, a notification system was written to tell the user when there are problems with the environment or hardware. To implement notifications, a request for data is automatically sent every minute, and the user is notified as necessary. Finally, an activity was created for the user to login with an account to ensure security and privacy.

Results

After the application was completed and the physical system was ready to test, the app was taken to the greenhouse dome to assess its functionality. The app was built on a virtual emulator in the IDE, and each request was tested. The app was able to successfully get and display environmental data (Figure 4), send parameters to the server, which directly caused changes in the hardware, and manually control the hardware. All functions of the app including user login, server requests, and graph collection (Figure 5) were fully functional.

Figure 4 (left): The raw data that is collected by the app and displayed to the user when the data activity is launched, a "get" button is pressed, or 1 minute of time passes. A notification about the water temperature in the IBC tote is being sent

Figure 5 (right): A graph of the outside temperature, inside temperature, IBC temperature, and compost temperature taken at the dome over a short period of time.



Conclusions

The purpose of this project was to create an Android application that remotely monitors and controls hydroponics in a cost-effective manner. The app was successfully created and refined to ensure a good user-experience, and it is fully functional with regards to controlling the greenhouse. Despite the success of this project, there are additions and improvements for future studies to make, such as including a live video feed to the app, handling data for pH levels, and using server requests rather than local storage to save the state of activities in the app.

References

Barik, L. (2019). IoT based Temperature and Humidity Controlling using Arduino and Raspberry Pi. *International Journal of Advanced Computer Science and Applications, 10*(9), 494–502.

Jabbar, Z & Kawitkar, R.S. (2016). Implementation of Smart Home Control by Using Low-Cost Arduino & Android Design. *International Journal of Advanced Research in Computer and Communication Engineering* 5(2), 248–256.

José de Oliveira Júnior, A., Lucas de Souza, S., Fitas da Cruz, V., Vicentin, T., & Glavina, A. (2018). Development of an android APP to calculate thermal comfort indexes on animals and people. *Computer and Electronic in Agriculture, 151*, 183.