First Prototype Test Report

Team: 31
Members: Michelle Thevenin, Jiawei Liao, Keven DeOliveira, Samarah Uriarte, Michael Harkess
Test Date: 11/16/21

Equipment Summary

For the hardware aspect of our first prototype testing, we used an ESP32-WROOM-32 microcontroller, a computer that acted as the system's power source, and a DHT11 temperature and humidity sensor. For the software side, a React Native program acted as the front end, and temporary back end, of the mobile application, XAMPP was used as the interface to the MySQL database with Apache 2.4.51, and the microcontroller was programmed in the Arduino IDE. In order to demonstrate the mobile app's functionality, Expo Go was integrated into the testing process. Using this app, a team member connected to the EcoSwitch mobile application through a smartphone and interacted with the simple interface that was available for testing purposes. All of the equipment used during the first prototype testing was consistent with the expectations outlined in the test plan.

Setup Summary

Before testing, we set up the central server, mobile app, and ESP32 program. For the server side, we initially made sure the server was not running so the database would start collecting values once testing had begun. Then, we ran XAMPP with admin privileges and activated Apache and MySQL to run the server. For the mobile app side, we started the temporary backend service and Expo project in addition to checking that the IP address was properly configured to minimize technical errors. Following this step, we opened Expo Go on a team member's smartphone and scanned the QR code for the EcoSwitch mobile app to access it. In order to initialize the temperature and humidity values with the most recent entry from the central server database, we pressed the "Update" button that was displayed through the mobile app's UI. Finally, for the ESP32 side, we made certain that the device was correctly connected to a teammate's computer. Then, we used the Arduino IDE to run the appropriate program in the corresponding directory.

Testing Summary

During testing, we first flashed and monitored the ESP32 program through the Arduino IDE and observed the terminal output consisting of the room temperature and humidity values. This data did not change until we manually refreshed the page but it was successfully queried by the mobile app client and displayed in its interface. We then compared the values in the database and the values in the app interface, ensuring that there were no discrepancies, which would have been the result of errors. Finally, we observed the time it took for the values to automatically update throughout the system, with new data values ultimately being displayed on the mobile app. As intended, it took half a minute for this to successfully occur. However, we will change this time interval to every 5 minutes in the final prototype. Figure 1 shows the workflow of our testing procedure.

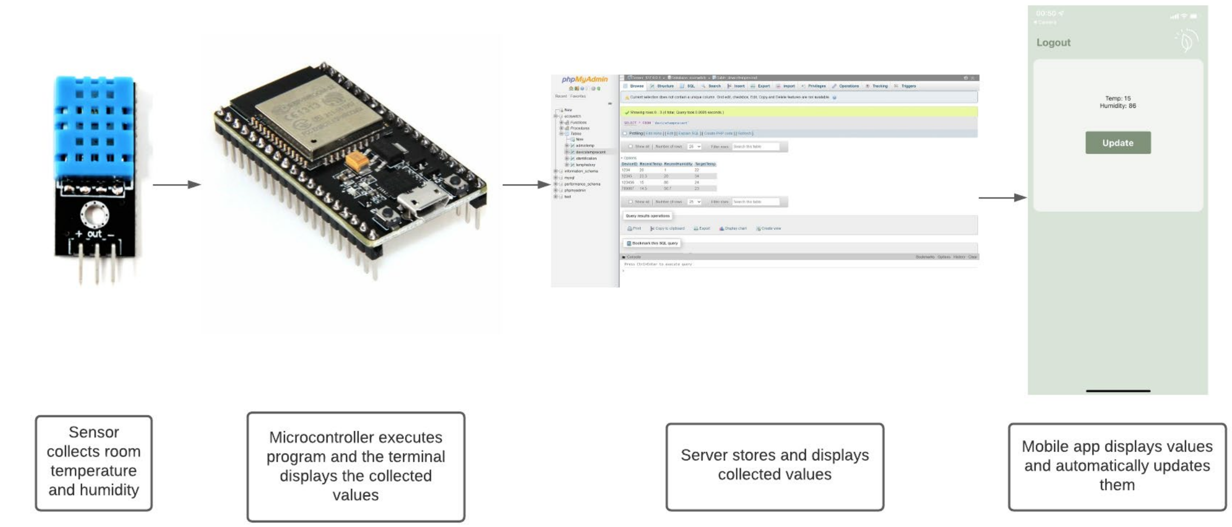
Logout
Temp: 15
Humidity: 86
Update
Sensor collects room temperature and humidity
Microcontroller executes program and the terminal displays the collected values
Server stores and displays collected values
Mobile app displays values and automatically updates them

Figure 1: Illustration of Setup and Process flow

Detailed Measurements

The first set of measurements taken were those of room temperature and humidity levels using the DHT11 sensor. The values themselves were generally accurate representations of the conditions of the testing room. The next "measurement" was to ensure that the values displayed on the application were the correct ones from the sensor and the database. Following this first demonstration, the sensor was then covered using fingers to skew the measurements to show that the sensor was working correctly.

Conclusion

Due to the success of our first prototype testing, we feel confident that we will be able to complete our current goals for the rest of our semester. This includes setting up a simple version of the administrator website, while ensuring it is able to query for information from the central server, and adding the actuator to the circuit, along with code to program it. Additionally, we plan to add a functionality to the app that allows it to send information back to the server, namely, temperature adjustments.

We will be making changes to the environment we were originally programming the ESP32 in, as a result of the conclusions we have arrived at based on the process of setting up the system for the first prototype test. Previously, we were working with the ESP-IDF, but upon encountering several difficulties, we switched to the Arduino IDE and this allowed us to finish all of our initial prototype goals on time. The stability and easy setup that the Arduino environment offers makes it a more optimal choice, given our time constraints.

Finally, we have noticed that the DHT11 sensor readings are more inaccurate than expected. We plan to reconsider our choice and possibly upgrade to the DHT22, though the cost is slightly higher than the DHT11.