Final Prototype Test Report

Team: 31
Members: Michelle Thevenin, Jiawei Liao, Keven DeOliveira, Samarah Uriarte, Michael Harkess
Test Date: 4/21/22

Equipment Summary

Hardware:

- Microcontroller: ESP32-WROOM-32
- A computer (to show server data)
- DHT22 Temperature and Humidity Sensor
- 12V 1.5A Stepper Motor
- 12V Battery Pack
- L7805 5V Voltage Regulator
- L298N Motor Driver
- 3D printed casing
- 3D printed attachment
- Lutron FSQ-2FH-DK Electronics Rotary Fan-Speed Control
- A blow dryer

Software:

- Mobile application: React Native, Node.js
- Website: ReactJS, Node.js
- Server: Amazon EC2 running Linux (Debian) and Apache 2.4.51
- Database: Amazon RDS (MySQL, MariaDB)
- Arduino IDE

Setup Summary

The setup is divided into hardware and software: the hardware consists of the ESP32 microcontroller, the DHT22 temperature and humidity sensor, the L298N Motor driver, a 12V stepper motor and a rotary Fan Speed Control. The software consists of the mobile app, website, and server and database.

The DHT22 sensor detects the room temperature and humidity levels and sends signals to the ESP32 via pin D13 (GPIO 13). The ESP32 program then sends this data to the server that is running on a local computer. Upon reaching the server, the data is recorded in Amazon RDS (in a MySQL database) and the mobile app then queries for this data on a local phone to display the values through the interface. The website also does this, along with displaying additional test devices and their corresponding data, and it offers a registration functionality that adds and removes student information to and from a device user database when entered by the user as well. Additionally, the website provides an interface for turning off all the devices in one location. When the

temperature read from the sensor is out of range with the temperature input from the mobile app, the motor will slightly turn, which would allow it to switch a Fan Speed Control dial in practice. The Fan Speed Control dial goes from off, high, medium and low in clockwise direction.

Testing Summary

We first set up the testing environment which included our hardware circuit connected to a motor, attachment, and ceiling fan switch. We then explained that the DHT sensor would read in temperature and humidity information, send it to the AWS server instance we have constantly running, and store it in the database that is accessible to the mobile app and website via API calls.

Following this high-level overview of the system, we demonstrated the mobile app interface and functionalities of retrieving data from the EcoSwitch device and sending back a user-inputted temperature to the database, which then stored this value. The website was also shown, as well as the various pages that were currently available with this version. All available EcoSwitch devices and their corresponding information were queried and displayed in a clean chart. This demonstrated the website's ability to connect to the server with the aforementioned API interface. The website was able to update the student user database by registering a student to a specific test device, as well as lock all the devices in a specific location.

To demonstrate our complete system, we input a low temperature into the app, and waited for the motor to turn our switch on. At first, our battery was not fully connected to the circuit, so the switch was not turning. Once we fixed the ground wire, the motor moved successfully with a few undesirable patterns. At the end of the testing session, all aspects were able to connect, allowing us to demonstrate the device's functionality. While the testing process did not go completely smoothly, we were successfully able to demonstrate our device's logic.

The logic of the device is an algorithm that makes a linear approximation of the rate the temperature is changing in the room given two temperatures a set time apart. With that approximation, the algorithm predicts the future temperature of the room. If that predicted temperature is higher than the desired temperature and the current temperature is lower than the desired temperature, the algorithm will move the dial to a higher setting. This was demonstrated by heating the temperature sensor in the device and setting a high desired temperature. The algorithm moved the dial 3 places to the left (from off to low) and then moved to the right twice (from low to medium and medium to high) to get the highest setting to make the current temperature the same as the desired temperature.

Detailed Measurements

Most of the measurements made for testing were to ensure that the different components of the project were working properly in tandem. The first measurement made was confirming that the information polled by the DHT22 sensor and the ESP32 was being correctly sent to the database. Then, that the data was correctly fetched and displayed by the mobile application and administration website. These were all accurate.

Afterward, changes in temperature were made using a blowdryer. The new data was viewed in the database and then confirmed on the mobile application once the periodic updates were made. A desired temperature was set on the application to demonstrate the state logic that drives the switch. Using said logic, the dial was shown switching dial positions appropriately. The desired temperature was set to 30°C while the room temperature read 25°C, and it was observed that the dial was turned from the "OFF" position, to the "HIGH" position.

The administration website's functionalities were successfully demonstrated during the first attempt of testing; students were successfully added/removed, the secure login worked properly, and locking individual devices through the website worked as well. The administration website was not revisited during the final round of testing, so no further measurements were made with it at that time.

Conclusion

Since we had a total of 3 testing sessions for the final prototype testing period, we were able to improve and finalize our system after each session by using our testing results and the professors' feedback.

The conclusion we arrived at after the first testing attempt was that, since the device was unable to turn the actual FCU switch, we should focus on finding a switch with less torque but with similar qualities to those of the FCUs. Specifically, the professors suggested that we find a switch with 4 states ("OFF", "HIGH", "MEDIUM", "LOW") and a comparable shape to the FCU switch. In addition, we were told to focus on developing the logic behind how the device interacts with the dial based on user input. Both the website and mobile applications were working as intended but were missing their login features, so this was also completed during the preparation period for the next testing session.

After the second testing session, the website and mobile applications were completed and the logic behind the device's behavior was further advanced. There were still some inconsistencies in the system, however, due to the circuit's individual components seemingly failing, despite the ESP32 script producing outputs that were in line with what was expected. As a result, we agreed that we should switch out the motor for a newer motor of the same type, as well as the batteries for the battery pack. Additionally, the dial we purchased was still too difficult

to turn, so we seeked out another dial with significantly less torque. We also ordered a gear ratio component that we mounted on the motor.

Finally, we observed several technical issues during the third testing session that we quickly resolved upon working on the system more afterwards. Prior to this attempt, we ran through our test plan and were able to complete it and demonstrate the device's functionality amongst ourselves. However, the wire configuration for the battery pack was damaged, leading to a lack of power in the circuit that prevented the motor from turning the dial. After testing, we concluded that we had to increase the circuit's integrity by securing the connections between all components and replacing the batteries. Upon doing so, the device was able to turn the dial according to the input entered through the mobile app and the temperature of the surrounding air. While we were unable to test the motor with the gear ratio component during the session itself, we also examined its effectiveness at turning the more resistant fan dial and discovered that it was able to do so. As a result, we can now proceed testing on an actual FCU dial and observe the full functionality of our system in its real world application.