Ryan Bentz
Ram Bhattarai
ECE 558
Project 3
11/17/18

ECE 558 Project 3 Final Report

I. Project Architecture

The project includes a “software app” that runs on a mobile device and a “hardware app” that runs on a Raspberry Pi. The software app allows the user to interact with the hardware connected to the hardware app through Google Firebase. The hardware app is connected to an RGB LED, temperature sensor, and DC motor. The Raspberry Pi uses I2C to communicate with a PIC microcontroller that does the heavy lifting for the direct control over the hardware components.

The software app uses SeekBars to control the values of the PWM for each LED in the RGB LED. It uses buttons to increment and decrement the value of the DAC. And, it uses a ProgressBar to visually indicate the duty cycle of the PWM controlling the DC motor. When either of the SeekBars or the DAC buttons are pressed and the values are updated, the app writes the new value to entries in Google Firebase. The app also reads the temperature and ADC values written to the database by the hardware app.

The hardware app runs on Android Things flashed to a Raspberry Pi. The app listens for changes in the Google Firebase and gets the new values if anything changes. It reads the PWM values for the LED, motor, and DAC values from the database and writes them to the PIC to control the hardware. It also polls the PIC microcontroller periodically to get new temperature and ADC values. When it gets new values, it writes them to the database for the software app to read.

Technical Details

The project makes use of Google Firebase as a centralized location to read/write data. The software app makes use Seekbars and a ProgressBar as new widgets used. The hardware app makes use of the Peripheral Manager to interact with the hardware connected to the Raspberry Pi.

Handling Exceptions

When exceptions occur, they are caught and a stack trace is printed. In the case of an I2C exception, the Raspberry Pi will toggle a GPIO pin connected to an LED.

Test Strategy

Testing was straightforward. The temperature sensor was tested by opening a window to cool it and pinching it between two fingers to warm it up. Verifying interactivity with the database was done using the adb debugging tools and manually altering database values through the Firebase console.

Work Division

Ryan wrote the software app. Ram created and setup the database. Both Ryan and Ram worked on the AndroidThings/HW portion of the app independently and both created working versions of the project.

Challenges Faced

Ryan

There was a problem trying to get AndroidThings to run on the Raspberry Pi Model 3 B+. It appears that AndroidThings does not yet currently support the model of the Raspberry Pi.

I had an issue with the permissions on the HW app because I didn't put in the permissions for the peripherals in the manifest.

I had an issue with the HW app not running because I had used "new empty activity" in Android Studio and it was calling setContentView when there was no content to inflate.

I had an issue with getting the app to persist on the Raspberry PI after the device had been power cycled because I didn't put in the lines in the manifest to have the app auto-launch on boot.

I had an issue with trying to install the app using the adb tool because there was a conflict with the AndroidThings API level and the API level the app was being compiled for. I never did get this fixed but ended up finding a work-around by flashing the app with AndroidStudio and going from there.

I had an issue setting up the database because I didn't understand how to set up the real time database.

I had an issue with AndroidThings and the default empty activity setup using AppCompatActivity instead of just the normal Activity.

Ram

It took me a while to setup the hardware because the hardware was not functioning as i planned. Initially, I could not even flash the Android things on my Raspberry PI because it always complained about not having enough permission for the script even though i provided admin credentials. I had difficulty getting my wifi working properly and after a while researching online, I found out that once the image is flashed to the Raspberry PI, we also need to run the Setup wifi utility from the Android things script.

I also had difficulty getting the temperature sensor to function correctly. It was always giving me random temperature. I had to ask professor Roy about that and he recommended me to look at the section where it says temperature vs voltage ratio.

I also did not get the correct Resistors to start with from EPL store. They were all 300 ohms resistors. I had to get another resistors from the EPL store.

Initially, the gradle file was giving me trouble because the google app services was not up to date. It kept complaining on start the FirebaseApp. Once I updated the google app services in the gradle file, it started working fine.

During the demo, we had difficulty in connecting to the database. Due to slow connection in the room, the database was lagging.

Feedback

This was a fun project and really great to learn how easy and fun it is to make an IoT device with AndroidThings. Aside from the mixup with the Model B+, We think the whole project went really smoothly. There seemed to be plenty of information online in terms of Android Documentation and StackOverflow answers to solve all my troubleshooting problems.