Project 3
IoT Home Automation System

Course: ECE 558 Embedded System Programming

Instructor: Prof. Roy Kravitz

Submitted by: Mrunal Hirve, Shubham Vasaikar

Date of Submission: 11/17/2018

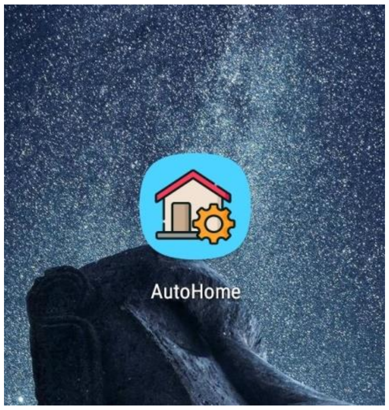
Video: https://youtu.be/3VxIDZAHDf4

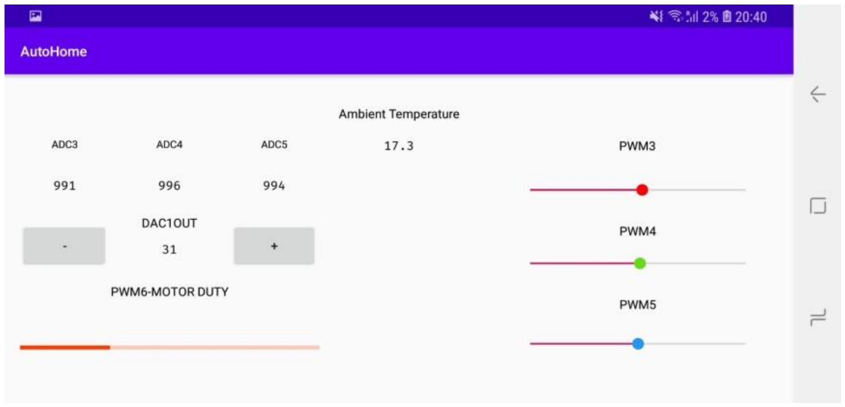
Code: https://github.com/ECE558-fall2018/ece558proj3-shubhamv_mrunalh

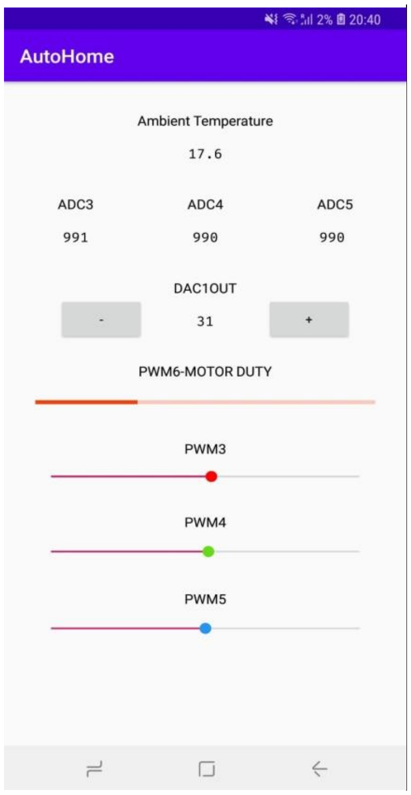
Software Details

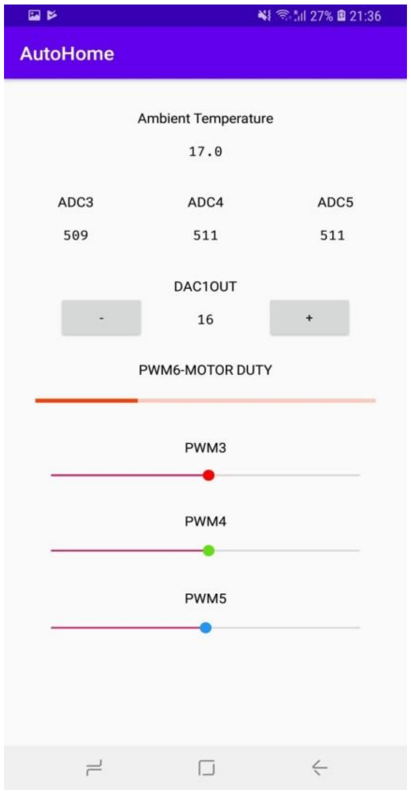
Android App

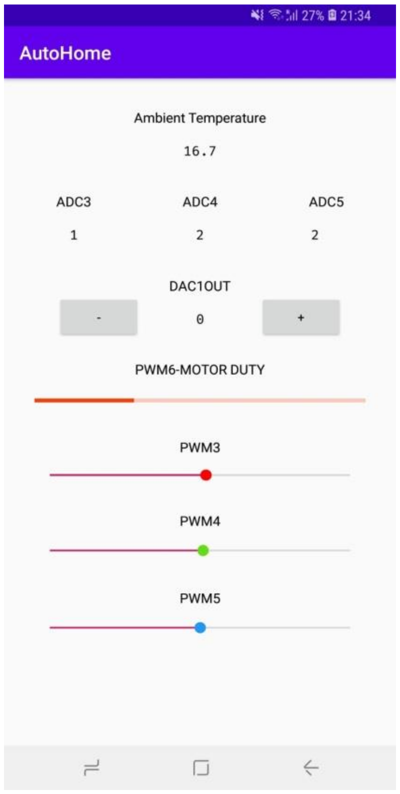
The layout of the Android app is based on the default layout given in the project spec. The layout is implemented using ConstraintLayout and Android components such as SeekBar, ProgressBar etc. The app also has a custom icon.











New Java Features Used

Threads

On the Raspberry Pi, threads are used to get values from the peripherals simultaneously. This allowed the main() thread to not be blocked when the raspberry pi is trying to access the peripherals. There are 3 threads in the activity:

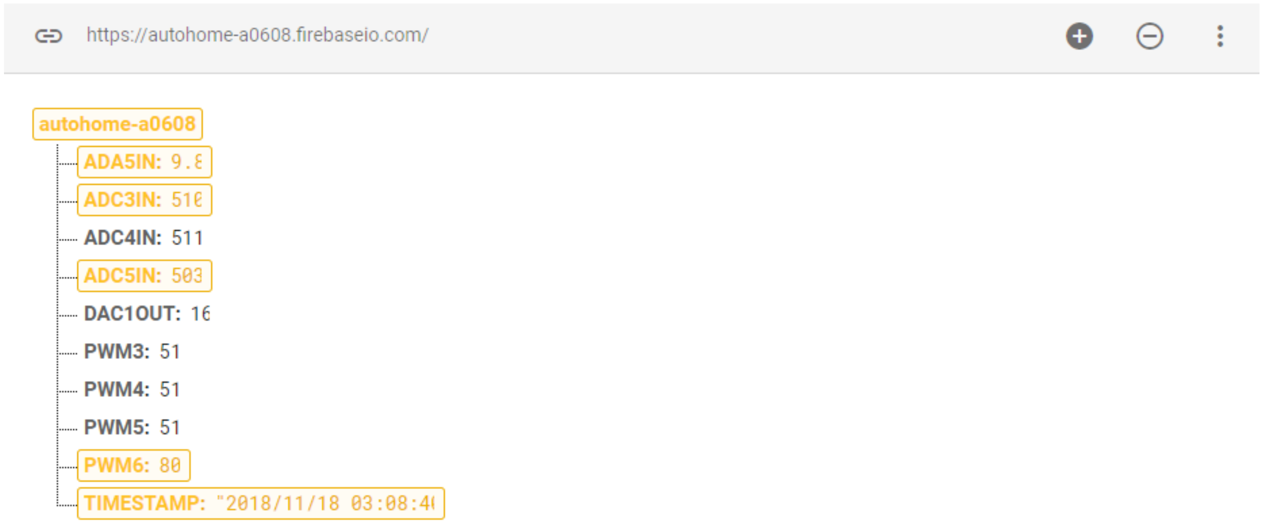
1. Thread to get temp values and set motor duty cycle.
2. Thread to get values from ADC Channels 3, 4, and 5.
3. Thread to blink the LED.

Below is a code snippet for the thread to get ADC Channels:

```java
new Thread(new Runnable() {
    @Override
    public void run() {
        try {
            while (true) {
                mAdc3In.setValue(readADC("ADC3IN"));
                mAdc4In.setValue(readADC("ADC4IN"));
                mAdc5In.setValue(readADC("ADC5IN"));
                updateTimestamp();
                Thread.sleep(2000);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}).start();
```

Firebase

The firebase realtime database is implemented as follows:



```
https://autohome-a0608.firebaseio.com/

autohome-a0608
    ADA5IN: 9.
    ADC3IN: 51
    ADC4IN: 511
    ADC5IN: 503
    DAC1OUT: 16
    PWM3: 51
    PWM4: 51
    PWM5: 51
    PWM6: 80
    TIMESTAMP: "2018/11/18 03:08:4
```

In the Android app, there are ValueEventListeners set on the ADA5IN, ADC[3,4,5]IN, PWM6 to listen for changes and update the changes as soon as the RPi updates these values. Similarly, on the RPi, there are listeners for PWM[3,4,5] and DAC1OUT to update the LED duty cycles and the DAC1OUT value on the PIC.

To ensure that the Android app always has the latest values, there is also a SingleValueEventListener on the DatabaseReference. This ensures that anytime the onCreate() method of the activity is called, the values are retrieved from the Firebase Realtime DB instead of using savedInstanceState. This also takes care of the rotation scenario where we don't have to save values between configuration changes. Below is the code snippet:

```
myRef.addListenerForSingleValueEvent(new ValueEventListener() {
  @Override
  public void onDataChange(DataSnapshot dataSnapshot) {
    //sets value onetime from firebase on start of application and on device rotation
    mDAC1OutTextView.setText(dataSnapshot.child("DAC1OUT").getValue(Integer.class).toString());
    mPWM3bar.setProgress(dataSnapshot.child("PWM3").getValue(Integer.class), true);
    mPWM4bar.setProgress(dataSnapshot.child("PWM4").getValue(Integer.class), true);
    mPWM5bar.setProgress(dataSnapshot.child("PWM5").getValue(Integer.class), true);
    mTempTextView.setText(dataSnapshot.child("ADA5IN").getValue(Double.class).toString());
    mChannelADC3TextView.setText(dataSnapshot.child("ADC3IN").getValue(Integer.class).toString());
    mChannelADC4TextView.setText(dataSnapshot.child("ADC4IN").getValue(Integer.class).toString());
    mChannelADC5TextView.setText(dataSnapshot.child("ADC5IN").getValue(Integer.class).toString());
    mPWM6ProgressBar.setProgress(dataSnapshot.child("PWM6").getValue(Integer.class), true);
  }
}
```

Hardware Details

Raspberry pi functionality

I2C is used to communicate with the PIC microcontroller. Android Things provides a PeripheralManager that can be used to open a communication channel with an I2C device. To blink the LED, GPIO is used instead of I2C. The same PeripheralManager can be used to open a connection to the GPIO device. Below is a snippet to open I2C and GPIO devices.

```
try {
    PeripheralManager manager = PeripheralManager.getInstance();
    mLedGpio = manager.openGpio("BCM17"); // Open GPIO device
    mDevice = manager.openI2cDevice("I2C1", 0x08); // Open I2C device
} catch (IOException e) {
    Log.d(TAG, "Unable to open I2C Device.");
    e.printStackTrace();
}
```

Below is a snippet to set register values using I2C:

```
try {
    mDevice.writeRegByte(0x00, (byte) pwm3);  // Write to reg 0x03
} catch (IOException e) {
    Log.d(TAG, "Failed to write to PWM3");
    e.printStackTrace();
    setLed();
}
```

Work division

Mrunal Hirve: Android app and Firebase

Shubham Vasaikar: Firebase, RPi app, circuit

Challenges faced

Raspberry Pi 3B+ compatibility issues

1 day was spent on trying to install Android Things on a Raspberry Pi 3B+ after which we realized that there is a compatibility issue.

Measuring temperature with temp sensor

At first, we couldn't figure out what value was being returned from the temperature sensor. Then we consulted Nandita, and she told us that it was the output value in mV and, it could be converted to degC by dividing it by 10.

Motor voltage issue

Due to a power issue the voltage on the RPi was dropping below 4.65V (as indicated by the Red LED going off when the motor was connected) to the RPi. Despite all the connections being correct and running on a 2Amp power supply, we were running into this issue. When the motor is physically disconnected, there was no power issue, and everything ran fine. When the motor was connected, we ran into ServiceSpecificExceptions seemingly arbitrarily. There was no pattern as to when or how many times the exception would be thrown. Regardless, the exception is being gracefully handled and the blinking LED is set.

Testing Strategies

Temperature sensor

To test the temp sensor, we used a hair dryer to increase the ambient temperature around the sensor. This also confirmed the correct operation of the code for changing the motor duty cycle.

RGB LED

To test the RGB LEDs we simply changed the values for the 3 color channels and compared the resulting LED color with an online color picker. For e.g. R100, G0, B100 turned the LED to a bright purple color.

JavaDoc

The JavaDoc for both the projects is placed in the Javadoc/ directory of the 2 projects.

References

- https://firebase.google.com/docs/database/android/read-and-write
- https://stackoverflow.com/questions/44040416/convert-two-unsigned-bytes-to-an-int-in-java
- https://developer.android.com/things/sdk/pio/i2c
- https://developer.android.com/reference/com/google/android/things/pio/I2cDevice
- https://developer.android.com/reference/android/widget/SeekBar
- https://developer.android.com/reference/android/widget/ProgressBar

Attributions

App icon made by Freepik from http://www.flaticon.com/.