EECS 1021 – Lab F: Connecting your Grove / Arduino Hardware to Java

Dr. James Andrew Smith, PEng and Richard Robinson

Summary: In this lab, you will connect your Arduino or Grove board to your computer and communicate with it using Java.

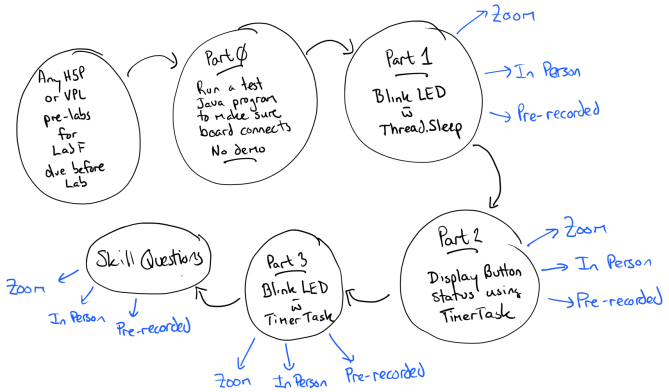


Figure 1 Lab F has four main parts after the pre-lab components. Make sure to do the pre-lab.

Intro

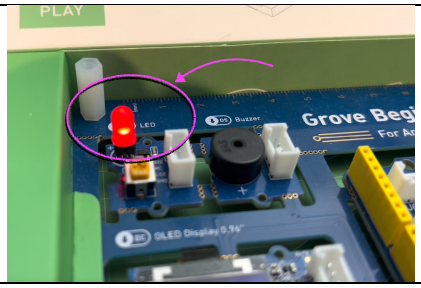
Arduino devices are commonplace in Engineering programs. Even if the profs don't use them, the students do. In Engineering programs, we often need to set up sensor monitoring or motor control systems. If students haven't worked with these in the first through third years of their programs, they are likely to encounter them in the "capstone" course in the final year of their program.

In my first year programming classes we alternate between using MATLAB and Java for linking a standard personal computer and an Arduino or Arduino-compatible board like the Seeed Studio Grove Beginner Kit for Arduino. The MATLAB exercises use vendor-supported hardware libraries. For Java, the solution is a little more complex. We could either have the students write custom C++ code for their Arduino and link to Java using a serial library like jSerialComm, or we could look for a simpler solution. That solution is Firmata. Two examples are shown on this page:

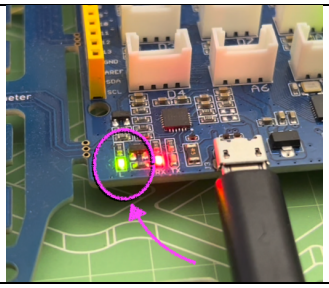
- https://www.yorku.ca/professor/drsmith/2022/02/25/easy-java-arduino-with-firmata/

To use Firmata we need two components:

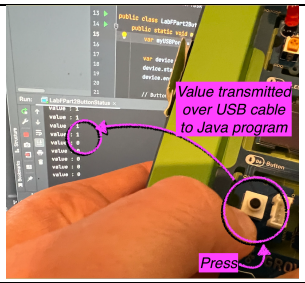
1. On the Arduino we use the Standard Firmata program, built into the Arduino IDE.
2. On the personal computer we use Kurbatov's Firmata4J library.



Part 1 & 2 : Flashing LED on Digital Pin 4 (https://youtu.be/XQVinyHokSw)



Alternative for Parts 1 & 2: Flashing LED on Digital Pin 13 (https://youtu.be/gakIoA1TfvE)



Part 3: Pressing the Button on Digital Pin 6 (https://youtu.be/Rcn8MNldnCc)

Figure 2 You will flash an LED and push a button in this lab.

Due dates.

- *Pre-Lab*: All of the interactive pre-lab activities are due on the Sunday night before the lab sessions.
- *Lab Demo*:
  - *Option 1*: live lab demo to the TA (Zoom or in-person)
  - *Option 2*: record a screen capture and submit a video to eClass.

Marking Guide:

- All interactive Pre-lab activities are graded out of 1 and count towards your "interactive" activity grade. Any other pre-lab activity is not graded.
- Part 0: No marks, just check that you can connect to the board.
- Part 1: 0.3 marks. 0.2 if partially successful. 0 if not attempted.
- Part 2: 0.3 marks. 0.2 if partially successful. 0 if not attempted.
- Part 3: 0.3 marks. 0.2 if partially successful. 0 if not attempted.
- Skill question: 0.1 marks. 0.05 if partially successful. 0 if wrong or not attempted.

Pre-lab

Check for pre-lab activities in Module 7. These could be either H5P activities or VPL activities or both. All pre-lab activities are due on the Sunday before the lab at 11:55pm.

Introduction

Set up your Arduino or Grove board

1. Download the Arduino IDE from https://www.arduino.cc/en/Main/Software
2. Connect the Grove board to your computer
3. Open the Arduino IDE
4. Go to Tools > Board and select the Arduino Uno.
5. Go to File > Examples > Firmata > StandardFirmata
6. Compile and run the sketch on your device to install the firmware.
7. Note your Arduino port. This is different for each device.
    a. Do this by going to Tools > Board > Port
    b. Note which ports are listed.
    c. Disconnect the USB cable from your board
    d. Go back to Tools > Board > Port
    e. Which one disappeared? Note it down.
    f. Reconnect the USB cable to your board.
    g. Go back to Tools > Board > Port
    h. Which one reappeared? Note it down.
    i. The one that disappeared, the reappeared is the one you will use in Java.

Create a Java Project that is ready for Firmata

1. Create a new project in Java, complete with a new class and a main method.
2. Import libraries via Maven
    a. Windows: JSSC, Firmata & SLF4j
    b. macOS: Firmata4J & SLF4J
3. Complete the class and method

	Windows	macOS
1st Maven Import	JSSC: io.github.java-native:jssc:2.9.4	Firmata4J: com.github.kurbatov:firmata4j:2.3.8
2nd Maven Import	Firmata4J: com.github.kurbatov:firmata4j:2.3.8	SLF4J: org.slf4j:slf4j-jcl:1.7.3
3rd Maven Import	SLF4J: org.slf4j:slf4j-jcl:1.7.3	Nothing.

Figure 3 Libraries that need to be imported into your IntelliJ java project. Import them in this order (JSSC first in Windows, Firmata4j first in macOS)

Part 0: Simple Connection to Arduino / Grove

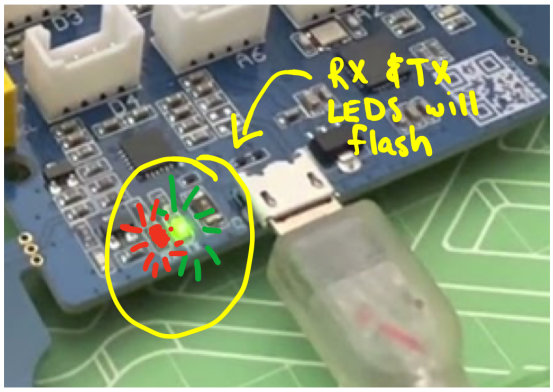


Figure 4 The RX and TX LEDs will flash briefly if your Java program successfully connects to the Arduino board. See it here: https://youtu.be/0dq-Mi-ddoU

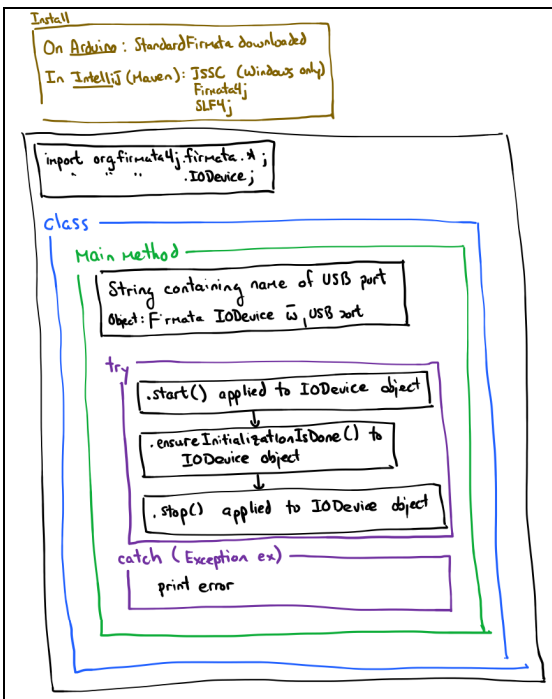
Follow the instructions above for setting up an IntelliJ project that is ready for Firmata and your Arduino-compatible hardware:
Prior to starting Intellij, download the StandardFirmata firmware to your Arduino or Grove board.

1. Create a class and main method in IntelliJ, as usual.
2. Import the JSSC (if you're using Windows), Firmata4j (everyone) and SLF4j (everyone) libraries using Maven

The class could be called LabFPart0 and you should have a main method.

Implement the source code found below.

There is **no** demonstration. Do this on your own to verify that you have a working setup. You may ask the TA for help if it's not working.




```
import org.firmata4j.firmata.*;          // Maven import Firmata4j &  SLF4j on macOS & Windows
import org.firmata4j.IODevice;           // You also need to import JSSC in using Windows.

public class SimpleExample {
   public static void main(String[] args) {
      String myPort = "/dev/cu.usbserial-0001";          // The USB port name varies.

      IODevice myGroveBoard = new FirmataDevice(myPort); // Board object, using the name of a port

      try {
         myGroveBoard.start();                           // start comms with board;
         System.out.println("Board started.");

         myGroveBoard.ensureInitializationIsDone();      // make sure connection is good to board.

         myGroveBoard.stop();                            // finish with the board.  Shut down the connection.
         System.out.println("Board stopped.");
      }
      catch (Exception ex){
         System.out.println("couldn't connect to board."); // message if the connection didn't happen.
      }
   }
}
```

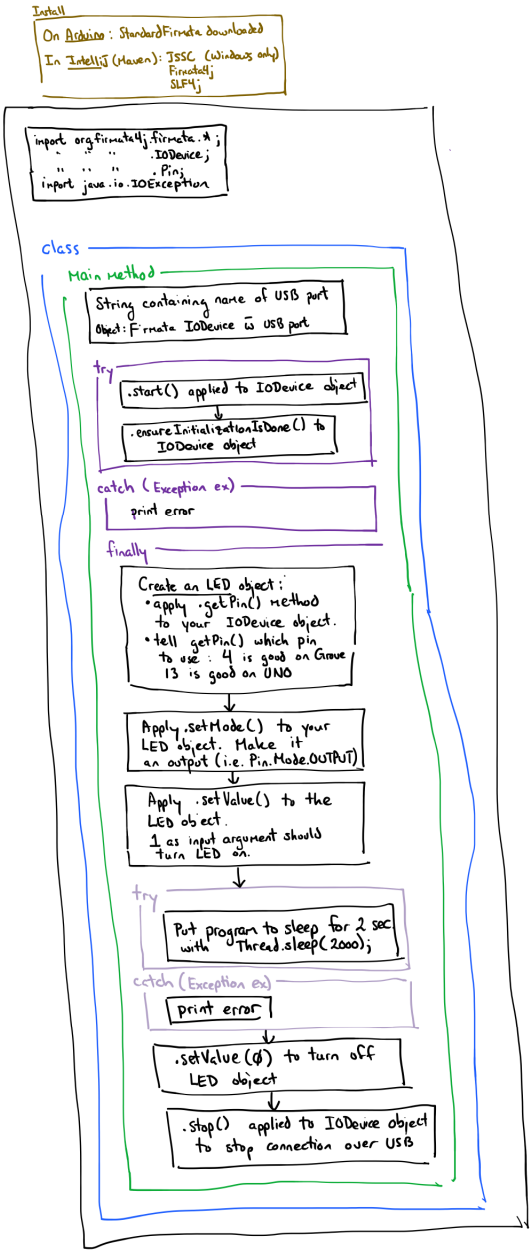
Flow chart for Part 0	Part 0 source code. Note the use of the try-catch.

Figure 5 Source code for Part 0. This is a minimal program to ensure that Firmata is running on your Arduino -compatible board.

Part 1: Blink LED Multiple times on Arduino / Grove

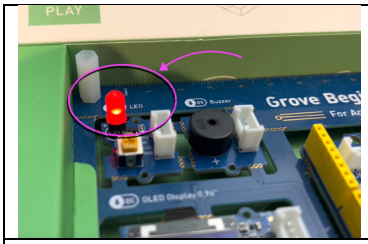
It's time to blink an LED on purpose. Which one? The best one to do this with is the D4 LED on the Grove board. However, you can also use the standard Arduino UNO one if you are using that one. The D4 LED requires you to use the number 4 when applying the getPin() method (e.g. myArduinoObject.getPin(4)). If you want to use the D13 LED that is standard on the UNO, then replace 4 with 13.

Keep the LED on for two seconds. Then turn it off. Use the Thread.sleep() method and pass the time value, in milliseconds into sleep() as its input argument.

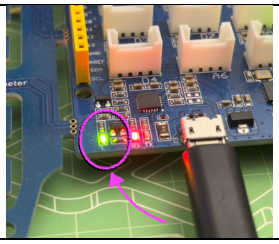


Flowchart for Part 1

```
// Maven: com.github.kurbatov:firmata4j:2.3.8
//        (https://mvnrepository.com/artifact/com.github.kurbatov/firmata4j/2.3.8)
// Also we need to manually re-install SLF4J : org.slf4j:slf4j-jcl:1.7.3
//        (https://mvnrepository.com/artifact/org.slf4j/slf4j-jcl/1.7.3)

import org.firmata4j.firmata.*;
import org.firmata4j.IODevice;
import org.firmata4j.Pin;
import java.io.IOException;

public class SimpleExampleRead {
  public static void main(String[] args)
  throws IOException
  {
    String myPort = "/dev/cu.usbserial-0001";

    IODevice myGroveBoard = new FirmataDevice(myPort); // using the name of a port

    try {
      myGroveBoard.start();     // start comms with board;
      System.out.println("Board started.");
      myGroveBoard.ensureInitializationIsDone();
    }

    catch (Exception ex) {
      System.out.println("couldn't connect to board.");
    }

    finally {
      var myLED = myGroveBoard.getPin(/* put value here */);
      myLED.setMode(Pin.Mode.OUTPUT);

      // LED D4 on.
      myLED.setValue(/* put value here */);

      // Pause for two seconds.
      try {
        Thread.sleep(/* put value here */);
      }
      catch(Exception ex){
        System.out.println("sleep error.");
      }
      // LED D4 off.
      myLED.setValue(/* put value here */);

      myGroveBoard.stop();      // finish with the board.
      System.out.println("Board stopped.");

    }
  }
}
```



Flashing LED on Digital Pin **4** (https://youtu.be/XQVinyHokSw)



Flashing LED on Digital Pin **13** (https://youtu.be/gakIoA1TfvE)

Source code for Part 1

***(modify** to match the number of blinks required)*

Figure 6 Source code and flowchart for Part 1

Now, modify the source code above to make it flash the LED multiple times. You need to manipulate the setvalue() method and the Thread.sleep() method to make this happen. What procedural program structure should you use to make something happen a fixed number of times?

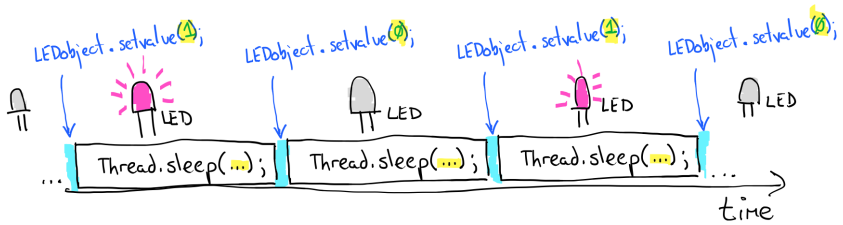


Figure 7 You need to turn the LED on and off multiple times using the .setValue() and Thread.sleep() methods.

Demonstrate to the TA that you can make your board blink as described in the table below.

Monday	Last Name A-F	LED flashes: twice (on for 2 sec; off for 2 sec; on for 2 sec; ;off for 2 sec)
	Last Name G-M	LED flashes: three times
	Last Name N-S	LED flashes: four times
	Last Name T-Z	LED flashes: five times
Tuesday	Last Name A-F	LED flashes: six times
	Last Name G-M	LED flashes: seven times
	Last Name N-S	LED flashes: eight times
	Last Name T-Z	LED flashes: nine times
Wednesday	Last Name A-F	LED flashes: twice (on for 2 sec; off for 2 sec; on for 2 sec; ;off for 2 sec)
	Last Name G-M	LED flashes: three times
	Last Name N-S	LED flashes: four times
	Last Name T-Z	LED flashes: five times
Friday	Last Name A-F	LED flashes: twice (on for 2 sec; off for 2 sec; on for 2 sec; ;off for 2 sec)
	Last Name G-M	LED flashes: three times
	Last Name N-S	LED flashes: four times
	Last Name T-Z	LED flashes: five times
Pre-recorded submission	Last Name A-F	LED flashes: six times (on for 2 sec; off for 2; on for two; off for two… four more times)
	Last Name G-M	LED flashes: seven times
	Last Name N-S	LED flashes: eight times
	Last Name T-Z	LED flashes: nine times

Part 2: Display Button Status in Console on Arduino / Grove

In Lab C you created a Timer task and modified the run method. We're going to leverage that here.

There are three files in this project

1. LabFPart2.java (contains main method)
2. Pins.java (contains pin defintions for the board)
3. ButtonTask.java (defines what happens each time the Timer goes "bing!")

Recall from Lab C that you were to create a Timer task. It looked like this:

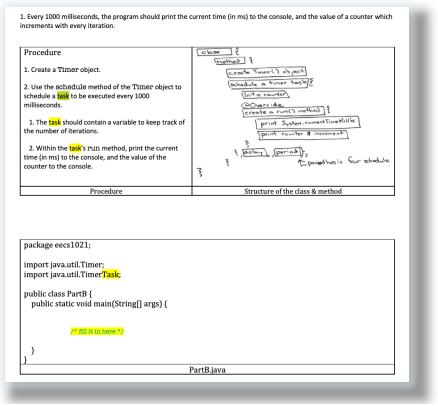
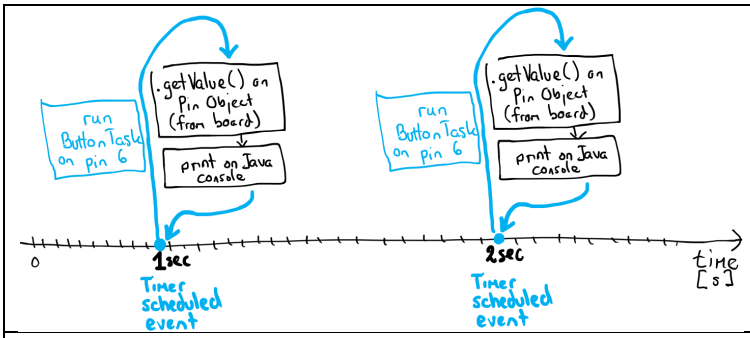
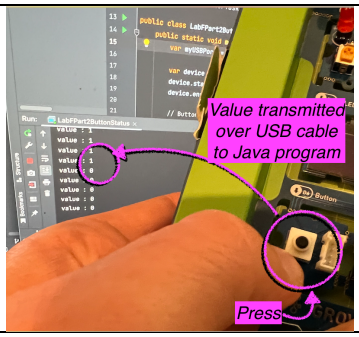


Figure 8 The Lab C task for creating a Timer task. This is the basis for this part of the Lab F exercise.

Now, create a new project. You need to include two files, one is called ButtonTask.java and the other is whatever you feel like calling it.



Events run on a timed schedule



Button status is examined on schedule.
(https://youtu.be/Rcn8MNldnCc)

Develop the main method so that it can examine the status of the Button on Digital Input 6 using the task schedule system from Lab C. You will override the run() method in the TimerTask so that it can print the value of the button. The examination of the button should happen every 1000 milliseconds (1 sec).

```
import org.firmata4j.Pin;                       // Firmata
import org.firmata4j.firmata.FirmataDevice;

import java.io.IOException;                     // Exceptions
import java.util.Timer;                         // Timer
import java.util.TimerTask;                     // Timer tasks.

public class LabFPart2 {
   public static void main(String[] args)
   throws IOException, InterruptedException
   {
      var myUSBPort = "COM4";   // TO-DO : modify this!

      // Create a FirmataDevice, start it and ensure Init is done.
       ...

      // Use getPin() to tell us which pin the button is connected to.
      // Set the mode of this pin to Input.
      // Button is Digital Input 6 on the Grove board (D6)
      // This is exactly like with the LED in Part 1, except that
      //       1. The Button is on digital input 6 (not 4)
      //       2. It's not Pin.Mode.OUTPUT ... it's INPUT.

      ...

      // make a new task for the button
      ...

      var task = new ButtonTask(button);
      // Provide this task to the Timer.schedule() method
      // and set the time to "no delay" and 1000 ms. periods.
      // this is like in Lab C!
      ...

   }
}
```

LabFPart2.java

```
import org.firmata4j.Pin;                       // Firmata
import java.util.TimerTask;                     // Timer tasks.

public class ButtonTask extends TimerTask {
   private final Pin myPin;

   // The Constructor for ButtonTask
   ButtonTask(Pin pin) {
      // Assign the externally-set "pin" to internal variable myPin
      this.myPin = pin;
   }

   @Override
   public void run() {
      // Print this each time TimerTask runs (e.g. once per 1000 ms.)
      System.out.println("value : " + myPin.getValue() );

   }
}
```

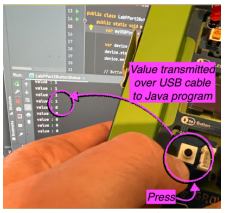


Figure 9 Pressing the button to return a value to the Java program. Video: https://youtu.be/Rcn8MNldnCc

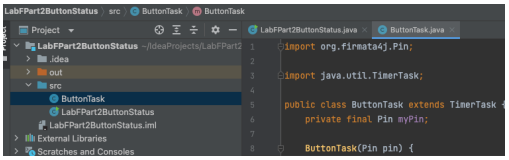


Figure 10 Note that we have two files in the project.

ButtonTask.java

Run the program. You should see 0's on the IntelliJ terminal when the button *isn't* pressed. You should see 1's in the terminal when the button *is* pressed.

Demonstrate to the TA that you can press the button and have a change appear in the terminal.

Part 3: Timed Tasks & Arduino / Grove

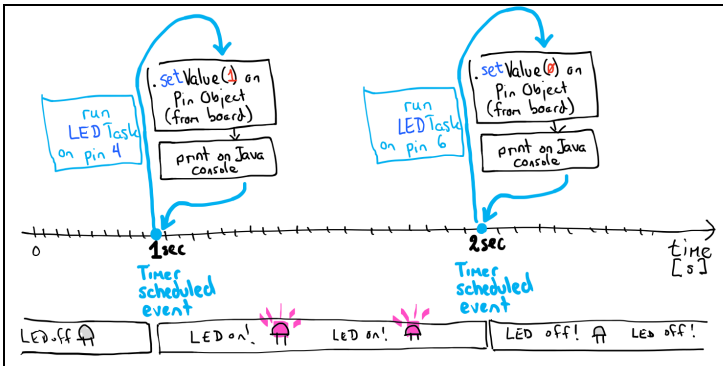
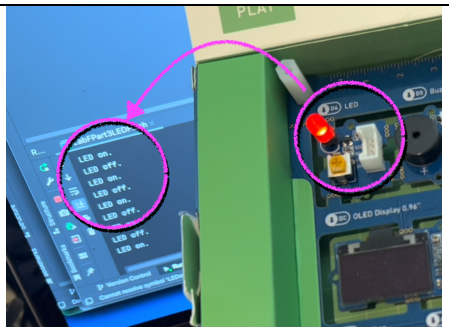
We will now modify the Timer task in Part 2 to make an LED blink with a period as described below. It is your job to extend the solution from Part 2 with what you know about the LED in Part 1 to make this happen.

Hints:

1. ButtonTask is a good place to start. Modify it to be an "LEDTask"
2. Within LEDTask you should keep track of a state (i.e. an integer that alternates between 0 and 1). This should be a variable that is private to the LEDTask class.
3. Consider using a conditional statement inside of the overridden run() method.

If you follow those hints you will effectively be creating an LED **state machine** that is driven by the TimerTask. Remember state machines from EECS 1011? Yeah. They're important. We need them here for these labs, but also for your minor and major projects.

Any solution that contains a loop (with or without sleep()) will be graded with half marks. Full marks only if you use the Timer task approach.

	
Run the LED flashing on a TimerTask.	Use the TimerTask approach to flash an LED at a desired rate. Video: https://youtu.be/z2Jq4HuAJj0

Demo to the TA that you can make the LED blink as specified using the Timer task.

Monday	Last Name A-F	LED Period: 1 second
	Last Name G-M	LED Period: 2 seconds
	Last Name N-S	LED Period: 3 seconds
	Last Name T-Z	LED Period: 4 seconds
Tuesday	Last Name A-F	LED Period: 4 second
	Last Name G-M	LED Period: 3 seconds
	Last Name N-S	LED Period: 2 seconds
	Last Name T-Z	LED Period: 1 seconds
Wednesday	Last Name A-F	LED Period: 5 second
	Last Name G-M	LED Period: 10 seconds
	Last Name N-S	LED Period: 2 seconds
	Last Name T-Z	LED Period: 4 seconds
Friday	Last Name A-F	LED Period: 2 second
	Last Name G-M	LED Period: 4 seconds
	Last Name N-S	LED Period: 1 seconds
	Last Name T-Z	LED Period: 3 seconds
Pre-recorded submission	Last Name A-F	LED Period: 5 second
	Last Name G-M	LED Period: 7 seconds
	Last Name N-S	LED Period: 10 seconds
	Last Name T-Z	LED Period: 3 seconds

Part 4: Skill Question

We've noticed that a number of students are simply copying and pasting each others' code. This is a recurring problem in programming classes, not just this one. To encourage you to try to understand the lab exercise content a bit more we're having TAs ask you a question during the lab demonstrations (Zoom or in-person).

If you are submitting a pre-recorded video, then you will have to answer one of following questions as a text submission to Turn-it-in on eClass. If the answer appears to be plagiarized, based on the Turn-it-in score, you will be given a grade of 0 for the answer.

- Last Name A – J: When making a Firmata4j project what is the library that needs to be imported separately via Maven if you are running Windows rather than macOS?
- Last Name K -- R: In the context of this lab, what method do you need to use if you are changing the value of the LED on the Arduino or Grove board. Compare and contrast with a similar method that does the opposite job.
- Last name S – Z : Explain Thread.sleep().