EECS 1021 – Lab G: Using the Display on your Arduino-Compatible Board

Dr. James Andrew Smith, PEng and Richard Robinson

Summary: In this lab, you will use the OLED, button and LED on your Arduino or Grove board to your computer and communicate with it using Java.

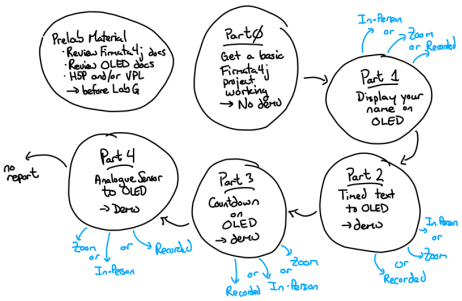


Figure 1 Lab G has four main parts after the pre-lab components. Make sure to do the pre-lab.

Intro

The OLED display on the Grove Beginner Kit is one of a family of similar displays found in many products. Learning to use this display will permit you to use this or similar displays in your engineering projects.

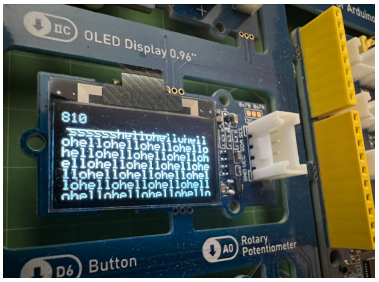


Figure 2 The OLED display on the Grove Beginner Kit for Arduino (https://bit.ly/3HY0Oyh)

There are three relevant main sites for data on the OLED display found on the Grove Beginner Kit for Arduino (SSD1315):

1. Seeed Studio's SSD1515 page: https://bit.ly/3HY0Oyh
2. The Firmata4j SSD1306 class: https://bit.ly/3t2Ez65
3. Adafruit's page on these types of displays: https://bit.ly/3vV7Rp5

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, verify that you can still connect to the Arduino board.
- Part 1: Simple text. 0.3 marks. 0.2 if partially successful. 0 if not attempted.
- Part 2: Timed text. 0.2 marks. 0.1 if partially successful. 0 if not attempted.
- Part 3: Countdown text 0.2 marks. 0.1 if partially successful. 0 if not attempted.
- Part 4: Potentiometer to text and bar: 0.2 marks. 0.1 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

You'll be repeating the same steps for project setup and library importing as you did for Lab F. Refer to the Lab F instructions.

Make sure that you

1. Start a new project from scratch.
    a. Don't include any files from a previous lab or project.
2. Import, via Maven, each of the libraries listed below.
    a. Three libraries (Win10/11 or macOS M1) or two libraries (macOS Intel)
    b. Follow the order
    c. Press the "Apply" button during import, after each library is brought it.
    d. Verify that all libraries are imported before proceeding
    e. Write up the simplest class and main method. Test.
    f. If it doesn't work, start again with a new project.

	Windows 10/11 or macOS (M1)	macOS (Intel)
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) Hit "Apply" after each import.

Important methods in the file:
(https://github.com/kurbatov/firmata4j/blob/master/src/main/java/org/firmata4j/ssd1306/MonochromeCanvas.java)

1. Clear()
2. setTextsize(int textsize)
3. setWordWrap(boolean wrap)
4. setCursor(int x, int y)
5. getCursorX() and getCursorY()
6. getWidth() and getHeight()
7. setPixel(int x, int y, Color color)
8. drawLine(int fromX, int fromY, int toX, int toY)
9. drawVerticalLine(int x, int y, int h, Color color)
10. drawHorizontalLine(int x, int y, int w, Color color)
11. drawRect(int x, int y, int h, Color color)
12. fillRect((int x, int y, int h, Color color)
13. fillScreen(Color color)
14. drawCircle(int centerX, intCenterY, int r, Color color)
15. more Triangles… RoundRect…
16. write(char c)
17. write(String s)
18. Color enums : DARK, BRIGHT and INVERSE

Important class: MonochromeCanvas. Color defaults to BRIGHT. Bgcolor defaults to DARK. cursor and cursory default to 0; wrap defaults to TRUE. Rotation defaults to 0;

Part 0: Simple Connection to Arduino / Grove

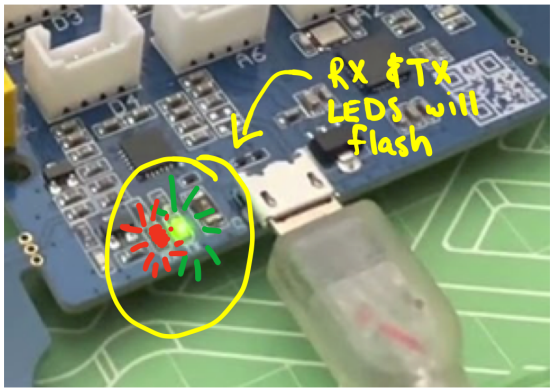


Figure 4 The RX and TX LEDs will flash briefly if your Java program successfully connects to the Arduino board. See it here: https://youtu.be/0dq-Mi-ddoU

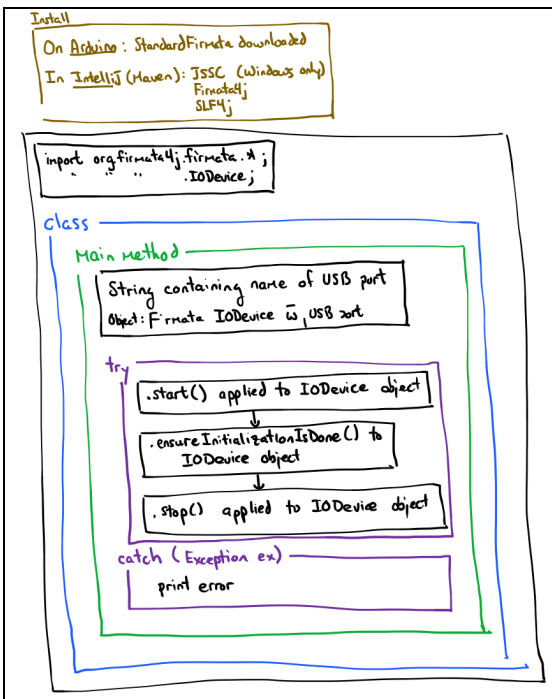
Follow the instructions above for setting up an IntelliJ project that is ready for Firmata and your Arduino-compatible hardware:
Prior to starting Intellij, download the StandardFirmata firmware to your Arduino or Grove board.

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: Print your Name on the OLED

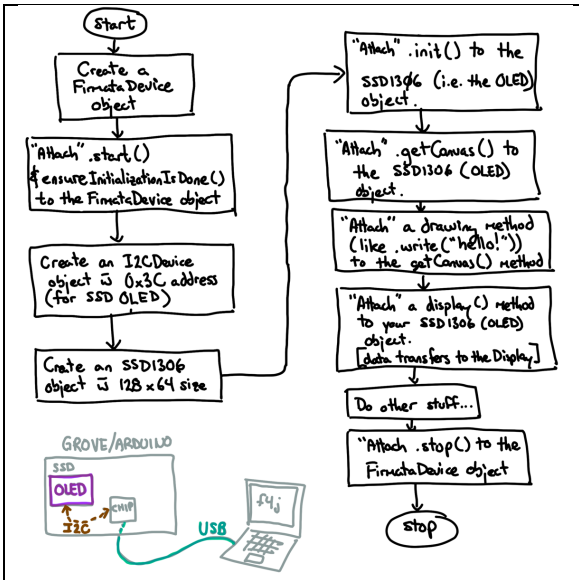
After you've instantiated an OLED object, the standard way to send a command to the OLED is to add another method to the object's getCanvas() method:

```
myDisplayObject.getCanvas().yourDrawMethod(arguments);
```

where yourDrawingMethod() can be

1. .drawHorizontalLine(0,0,3,MonochromeCanvas.Color.BRIGHT) // 3 pixel long white line, starts at (0,0)
2. .write("hello!")
3. .drawString(4,10,"bonjour!")

Implement the following program and **demonstrate** to the TA that you can display your name (first or last… up to you) on the OLED.



Flowchart

```
import org.firmata4j.firmata.FirmataDevice;
import org.firmata4j.IODevice;
import org.firmata4j.ssd1306.SSD1306;
import org.firmata4j.I2CDevice;
import org.firmata4j.ssd1306.MonochromeCanvas;

import java.io.IOException;

public class oledMain {
   public static void main(String[] args)
         throws InterruptedException, IOException
   {

      // Define your USB Connection
      String myUSB = "/dev/cu.usbserial-0001";

      // Create a FirmataDevice object with a USB connection.
      IODevice theArduinoObject = new FirmataDevice(myUSB);

      // Start up the FirmataDevice object.
      theArduinoObject.start();
      theArduinoObject.ensureInitializationIsDone();

      // Create an I2C communication object b/w the Arduino chip and the OLED over I2C wires
      // 0x3C is the standard address over these wires for the SSD1306 (or SSD1515)
      // Then create an SSD1306 object using the I2C object with the right pixel size for the OLED
      I2CDevice i2cObject = theArduinoObject.getI2CDevice((byte) 0x3C); // Use 0x3C for the Grove OLED
      SSD1306 theOledObject = new SSD1306(i2cObject, SSD1306.Size.SSD1306_128_64); // 128x64 OLED SSD1515

      // Initialize the OLED (SSD1306) object
      theOledObject.init();

      // Fill the OLED memory with something to draw.
      theOledObject.getCanvas().write("hello");

      // Update the OLED display (move data from memory onto the screen itself)
      theOledObject.display();

      // Wait.
      Thread.sleep(2000);

      // Stop the Grove or Arduino board.
      theArduinoObject.stop();

   }
}
```

Program

Figure 6 Starting up and using the OLED (based on example here https://bit.ly/3i1pTxL)

Part 2: Display three different pieces of text on the OLED

Modify your program from Part 1 so that you display three pieces of information to the screen, one at a time:

1. Your first name
2. Your last name
3. Your student ID

Each needs to be placed at location 0,0 on the screen (the upper right corner). The previous piece of information needs to be erased, too. Rather than use the .write(String) method, use the two following methods:

1. .clear()
2. .drawString(xlocation,ylocation,String)

The timing for the displaying of information is as follows:

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

Demonstrate to the TA that all three pieces of information appear on the OLED.

Part 3: Display a Countdown on the OLED

Next, combine the components from Parts 1 and 2 with the Timer Task from Lab F and Lab C.

Remember that to convert an integer into a String you can use the following method:
String.valueOf(theInteger);

Count down from 10 to 0 and then start again. Each count should take about 1 second.

Demonstrate to the TA that the system counts down on the OLED.

```
import org.firmata4j.Pin;
import org.firmata4j.firmata.FirmataDevice;
import org.firmata4j.ssd1306.SSD1306;

import java.io.IOException;
import java.util.Timer;

public class MainPart {

   static final byte I2C0 = 0x3C; // OLED Display



   public static void main(String[] args) throws IOException, InterruptedException {
      var myUSBPort = "/dev/cu.usbserial-0001";   // TO-DO : modify based on your computer setup.

      var device = new FirmataDevice(myUSBPort);
      device.start();
      device.ensureInitializationIsDone();

      // Set up the display (type, size ...)

      var task = new CountTask(myOledDisplay);
      new Timer().schedule(task, 0, 1000);
   }
}
```

Main java file

```
import org.firmata4j.Pin;
import org.firmata4j.ssd1306.MonochromeCanvas;
import org.firmata4j.ssd1306.SSD1306;
import java.util.TimerTask;

public class CountTask extends TimerTask{

   private int countValue = 10;
   private final SSD1306 theOledObject;

   // Constructor for CountTask
   public CountTask(SSD1306 aDisplayObject){
      theOledObject = aDisplayObject;
   }


   @Override
   public void run() {

      // Fill the OLED memory with something to draw.
      theOledObject.getCanvas().clear();  // clear contents first.

      // draw the String...

      // Update the OLED display (move data from memory onto the screen itself)

      // Update your count variable.

   }

   }
```

Display the countdown on the OLED

Part 4: Display the value on the potentiometer on the OLED & Terminal

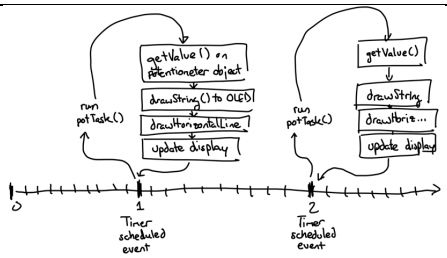
For this part, you will be reading the analog input value of the potentiometer of the Grove board.

Whenever this value changes, you will print out the value to the console.

In order to do this, you can extend the previous work that you did with a timer task.

Specification

When the value of the potentiometer changes, print out the value to the console in the format "Pot is 123" where '123' is the numeric value. Also produce a horizontal bar that is proportionally as long as the number.




```
import org.firmata4j.firmata.FirmataDevice;
import org.firmata4j.ssd1306.SSD1306;

import java.io.IOException;
import java.util.Timer;

public class DisplayMain {

  // Pin definitions (assuming Nano or UNO)
  static final int A0 = 14; // Potentiometer
  static final int A2 = 16; // Sound
  static final int D6 =  6; // Button
  static final int D4 =  4; // LED
  static final byte I2C0 = 0x3C; // OLED Display



  public static void main(String[] args)
      throws InterruptedException, IOException
  {
    var device = new FirmataDevice("/dev/cu.usbserial-0001"); // Change to your serial port

    device.start();
    device.ensureInitializationIsDone();

    // Potentiometer... use getPin, just like you did with the Button in the prvious lab.  But for A0 (or 14)


    // Set up the OLED  display (type, size ...)


    // do the timer thing here, just as you did in the previous lab.

  }
}
```

Main file

```
import org.firmata4j.Pin;
import org.firmata4j.ssd1306.MonochromeCanvas;
import org.firmata4j.ssd1306.SSD1306;

import java.io.IOException;
import java.util.Timer;
import java.util.TimerTask;

public class potTask extends TimerTask {
  private int duration;

  private final SSD1306 display;
  private final Pin pin;
  private final Timer timer;

  // class constructor.
  public DisplayTextTask(SSD1306 display, Pin pin, Timer timer, int duration) {
    this.myOled = display;
    this.pinInput = pin;
    this.myTimer = timer;
  }


  @Override
  public void run() {
    // note you need to convert the int to String.

    // print to the Java console

    // First, write the Potentiometer value.  Attach a "drawString()" method to getCanvas().

    //Next, erase the previous horizontal  line by applying "DARK"

    //  Now, draw a white line that is proportional to the potentiometer value.

   // run the display() method.
  }
}
```

Task file

Procedure

Extend the timer task work that you did earlier, but allow for the OLED display to be updated.

Looking for the names of the methods that you should use? Check out the GitHub project page listed in References below.

References:

1. MonochromeCanvas.java @ GitHub. (https://bit.ly/3vZMymy)
2. SSD1306.java @ GitHub. (https://bit.ly/35IKwfW)

Some hints:

1. Use the getValue() method on your potentiometer object and then convert to a String: String PotValue = String.valueOf(pin.getValue());
2. Use .drawstring(), attached to .getCanvas() to write the potentiometer value to the OLED. It's similar to write() but allows you to directly position it at a specific location on the OLED. Note that drawString() takes three arguments: x position, y position and a String.
3. Use drawHorizontalLine() method to draw a bar on the screen. It takes four arguments: x and y position, the length of the line and then the "colour" of the line (white is MonochromeCanvas.Color.BRIGHT and black is .DARK).
4. Don't forget to use the display() method at the end.

Demo to the TA that you can have the OLED show both the numeric value of the potentiometer (0 to 1023) and a horizontal bar that is proportional to the numeric value.

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: What geometric shapes are supported by the OLED's ssd1306 library. Name at least three by their method names.
- Last Name K -- R: How do you change the text size on the OLED? Explain. Give an example, too.
- Last name S – Z : What is the role of "this" in a constructor?