Tutorial 04 – Electrical Circuit and Signals

Make sure you complete **Setup** and
Test Your Adafruit Feather Huzzah32 Connection
document before starting this Tutorial 04.

Learning Objectives

1. Understand the use of the tactile switch and potentiometer on electrical circuits.
2. Understand the difference between digital and analogue signals, as well as the difference between sampling rate and bit rate in the Analogue-to-Digital Conversion (ADC) process.
3. Understand the basics of Pulse Width Modulation (PWM) in simulating analogue output from digital signal.
4. Understand the use of Serial Plotter in monitoring the output from sensors.
5. Understanding the basics of programming i.e., the use of compound operators (i.e., `--`).
6. Implement function calls related to analogue signals, i.e., `analogWrite()`, `map()`, on Adafruit Feather Huzzah32.

Theory

Analogue-to-Digital Conversion (ADC) or sampling, i.e., measuring at regular intervals, is a process of converting analogue information into digital data. For example, your voice on the phone gets converted from the sound wave into electrical signals before it can be transmitted over the mobile network. Figure 1 shows an ADC example with two different sampling rates.

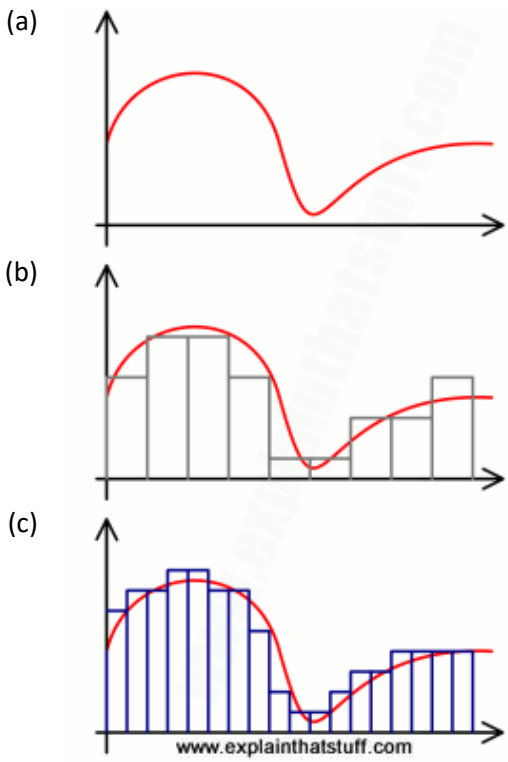


Figure 1: Analogue sound wave (top), digital approximation with a low sampling rate (middle) and a more accurate digital approximation with doubling the sampling rate (bottom) (Woodford, 2021).

In Figure 1 (b), there are half as many digital approximations than Figure 1 (c), and therefore requires half as much data to be stored and transmitted. The sampling rate or how many times per second a measurement is converted into a number. If each bar chart of the middle graph represents one second of time, then the sound wave (top) can be represented using nine numbers. In Figure 1 (b) for example, there values could be 5, 7, 7, 5, 1, 1, 3, 3, 5.

To address the lost of information in this process, more measurements can be taken and converted per second. In Figure 1 (bottom), there are twice any many measurements and therefore a more accurate approximation e.g., 6, 7, 7, 8, 8, 7, 7, 5, 2, 1, 1, 2, 3, 3, 4, 4, 4, 4.

Bit rate is the amount of information captured each time analogue information is sampled. The higher the bit rate, the more analogue information is captured and converted into digital information. For example, typically CDs and MP3 are sampling at 44.1 kHz (about 44,000 times per second which is twice the highest frequency that the human ear can hear i.e., 20 kHz. The typical bit rate for MP3 tracks is around 128 kbps (128,000 binary digits or bits per second) and higher quality tracks can have a bit rate between 128 kbps to 256 kbps.

Electrical circuits can be used to accept input from users. We have looked at the use of Serial Monitor to accept user input from the keyboard in Tutorial 02. Figure 2 shows two additional input devices which will be investigated in this tutorial, they are the tactile button and potentiometer.

Figure 2: Tactile button (left) (Digilent, n.d.), and Potentiometer (right) (Digikey, 2022).

The tactile button or momentary contact button makes contact when actively being pressed which output VDD (Device Supply Voltage). When at rest (not being pressed), these tactile buttons output GND. The potentiometer is a resistor with a rotating contact to adjust the output voltage to a fraction of its input voltage. However, it is rarely used to directly control the amounts of power but is commonly used to adjust the level of analogue signals e.g., volume controls. Figure 3 shows the different purposes of the 3 terminals (legs) of the potentiometer.

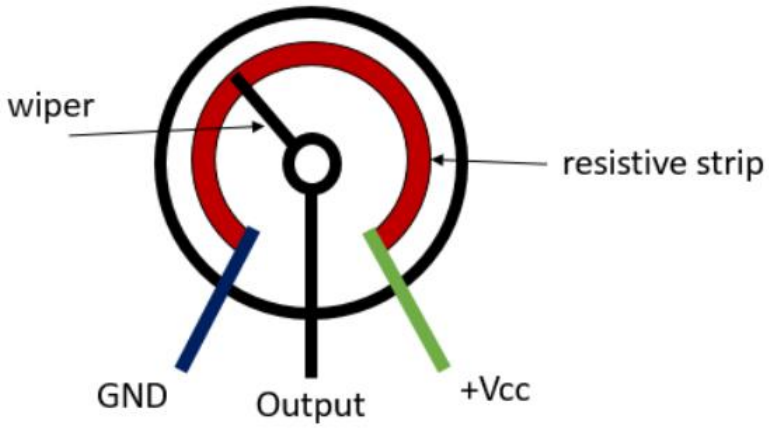


Figure 3: Potentiometer legs.

If you look closely at the potentiometer from the kit, you will note a small number i.e., 1, and 3, written on two of the 4 corners. The terminal by the number 1 should be connected to GND (if the terminals are facing away from you, this would be on the top right corner of the potentiometer). The terminal by the number 3 should be connected to VCC or Common Collector Voltage or supply voltage (if the terminals are facing away from you, this would be on the top left corner of the potentiometer).

Read the following Arduino reference pages which will be used in this Tutorial 03. You may also wish to revisit Tutorial 02, if you have already forgotten how to use `digitalWrite()` and `analogRead()`.

1. `map()`: https://www.arduino.cc/reference/en/language/functions/math/map/ .
2. `analogWrite()`: https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/ .

Unlike `digitalWrite()` which has two possible values i.e., `LOW`, or `HIGH`, `analogWrite()` has 256 possible values i.e., 0 - 255. As shown in Figure 4, Adafruit Huzzah32 uses Pulse Width Modulation (PWM) to simulate analogue results by modifying the "on time" or pulse width. When the value 255 is used, 100% duty cycle is requested and therefore the voltage is always on. Unlike `digitalWrite()`, the function `pinMode()` is not required for `analogWrite()`.

The function `map()` is useful in remapping a number from one range to another, e.g., from 1-10 to 1-100. The function can also be used to reverse a range of numbers, e.g., from 1-10 to 10-1. This is particularly useful when converting an output from `analogRead()`, i.e., 0-4095, to an input for `analogWrite()`, i.e., 0-255.

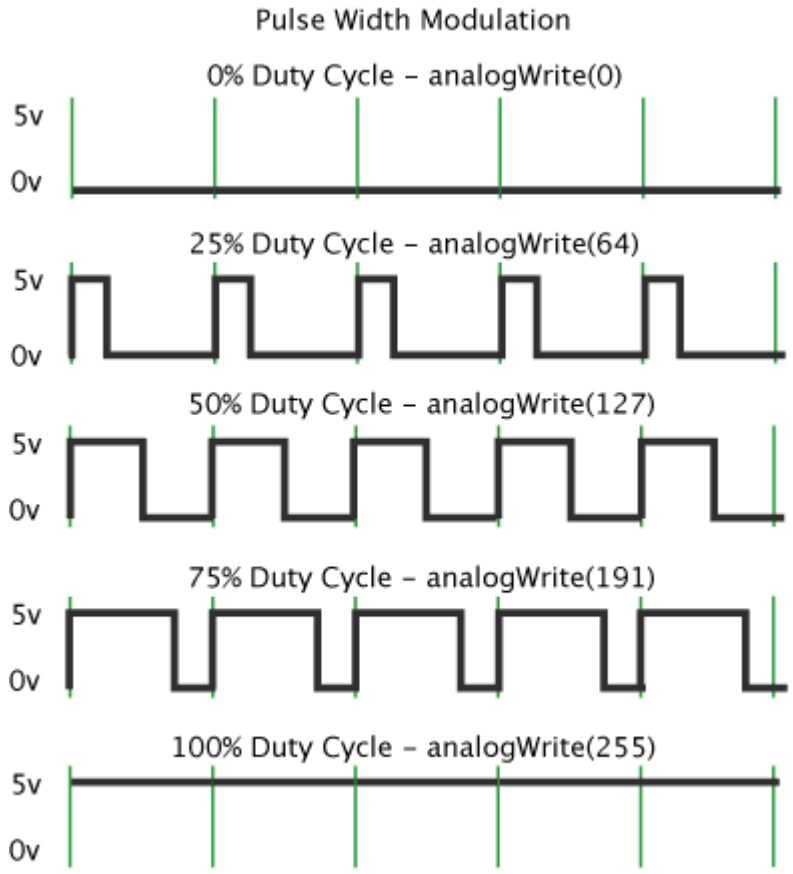


Figure 4: Pulse Width Modulation (Arduino, 2022).

In addition to the Serial Monitor which we have used in previous tutorials to monitor the outputs from the codes, Arduino IDE also provides a Serial Plotter which can be used to provide visual representations of the output as shown in Figure 5.

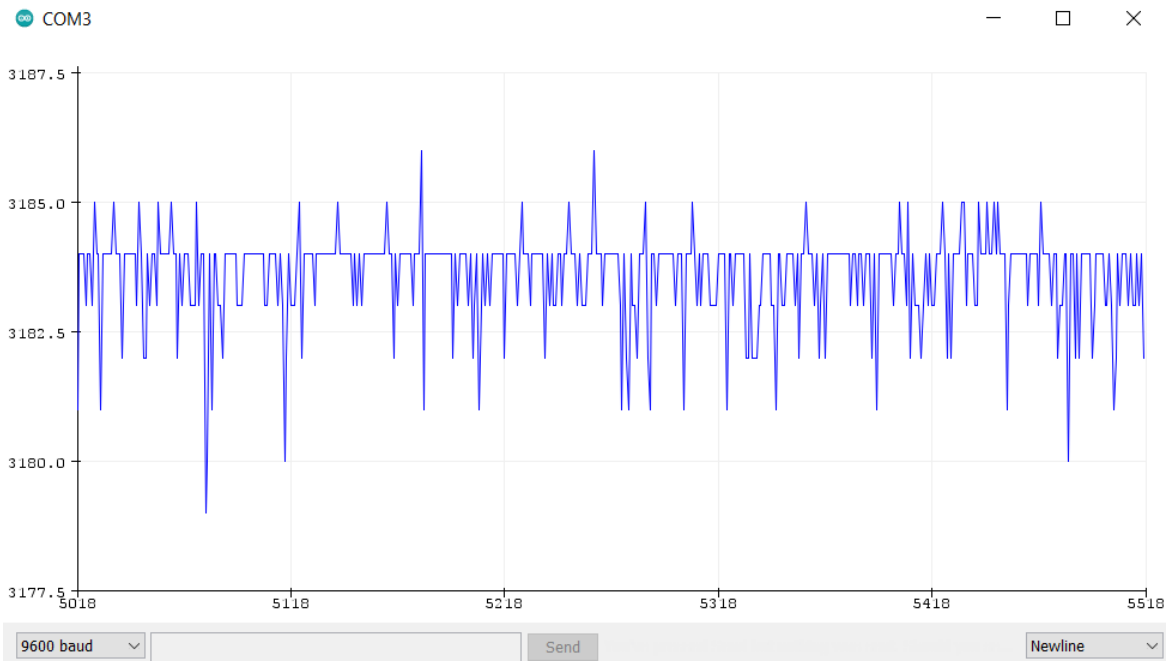


Figure 5: Serial Plotter.

As shown in Figure 6, you can access this Serial Plotter by going to **Tools** and then select **Serial Plotter**. The output should appear on the Serial Plotter, either in a separate window as shown in Figure 5, or in a Serial Plotter tab under the coding area (if you use Arduino IDE 2.0).

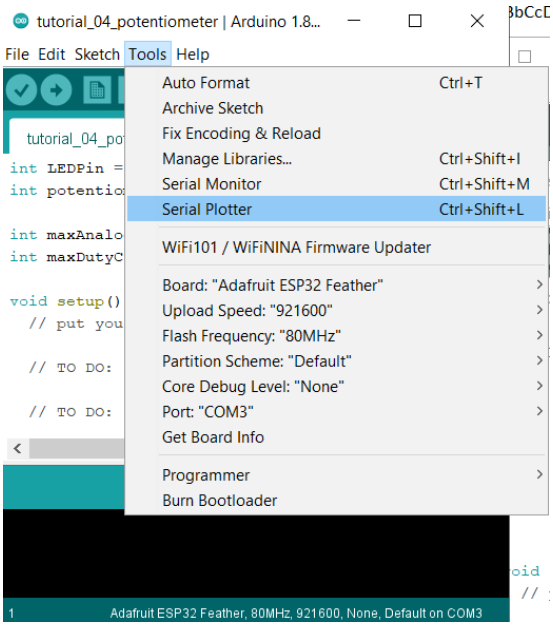


Figure 6: Accessing Serial Plotter.

To display multiple graphs in the same plot, each graph label can be added using `Serial.print("Name:")`, and the additional graphs can be added by adding `Serial.print(" ")` to separate the graphs as shown in Figure 7.

Figure 7: Adding labels and additional graph to the same plot.

It is important to close the Serial Plotter after each use (or before you upload and rerun a new piece of code) to avoid any unexpected results. As shown in Figure 8, the labels are incorrectly displayed i.e., there are 3 labels on the plot but there are only 2 graphs.

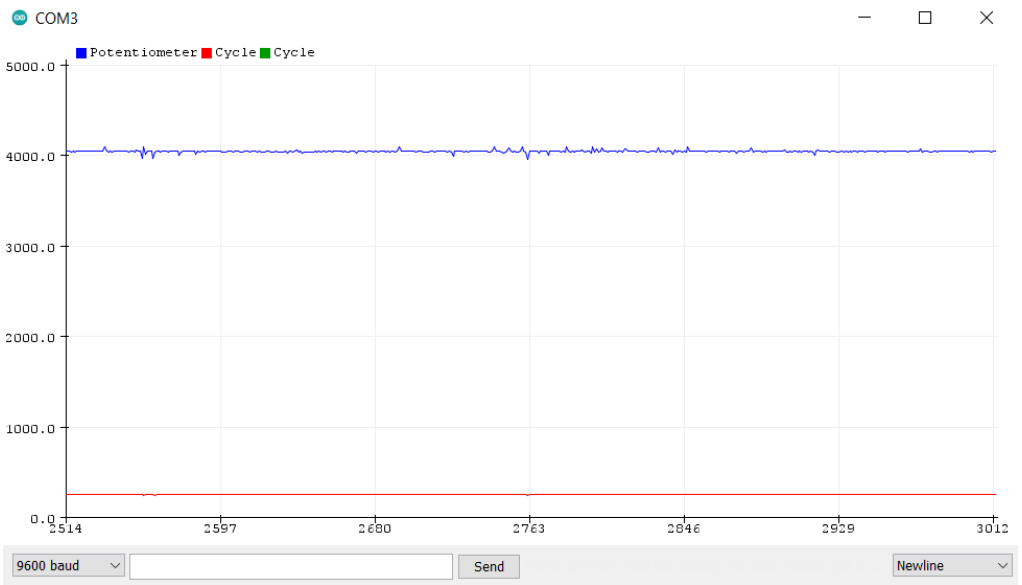


Figure 8: Serial Plotter with Unexpected Results.

Reference

Woodford, C. (2021). Analog and Digital. [Website] Available at: https://www.explainthatstuff.com/analog-and-digital.html

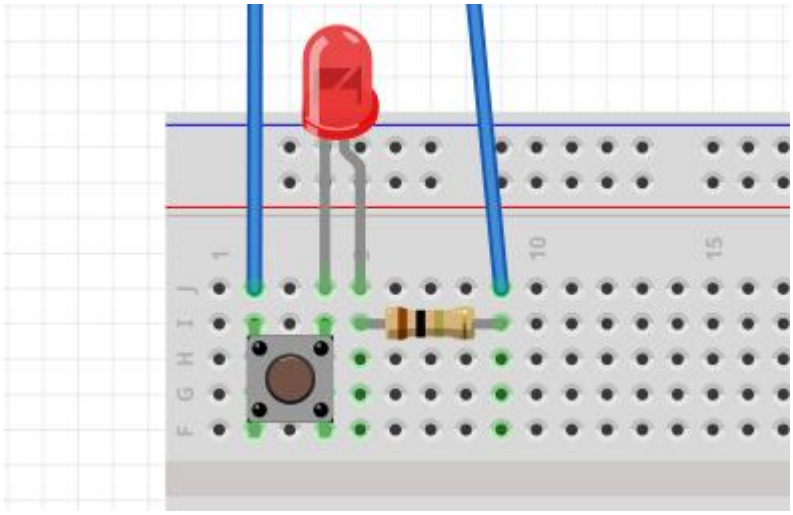
Digilent. (n.d.). Basic Digital I/O: Slide Switch, Push Button, and LED. [Website] Available at: https://digilent.com/reference/learn/fundamentals/circuits/basic-digital-io/start#:~:text=Switches%20and%20Push%20Buttons%20are,input%20devices%20for%20digital%20system.

Digikey. (2022). Potentiometer. [Website] Available at: https://www.digikey.co.uk/en/products

Arduino. (2022). Basic of PWM (Pulse Width Modulation). [Website] Available at: https://docs.arduino.cc/learn/microcontrollers/analog-output

Practice 4.1

1. Connect a tactile button, resistor, and LED diode according to the following schema. Please note that unlike previous Tutorials, the following schema is incomplete. It should be noted that the jumper wire on the left and the right is for power and ground respectively. These should be connected to the power (3V), and the Ground (GND) on Adafruit Huzzah32 correctly.



Source: https://fritzing.org/projects/led-w-button

2. Type the code below into the coding area of your Arduino IDE. If you haven't done so already, use the links provided in the Theory part of this tutorial to make sure that you thoroughly understand the meaning of every line of the code below.

```
int LEDPin = 15;

void setup() {
  // put your setup code here, to run once:
  pinMode(LEDPin, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:

  digitalWrite(LEDPin, HIGH);
  delay(1000);
  digitalWrite(LEDPin, LOW);
  delay(1000);
}
```

Note: Use practice4_1.json for WOKWI.

3. **Verify** the code. You will then be asked to save the sketch. Enter a **meaningful name** for your file, e.g., tutorial_04_switch. Please **take note where you save your file**, so you know where to find it, should you need it afterwards.

4. **Upload** the code.

5. Press (and hold) the tactile button and you should now see the LED diode lights up.

   Note: Avoid looking right at the LED!!! They can be very bright.

TO DO

1. Note the LEDPin value in tutorial_04_switch.ino, move the jumper wire into the correct pin.

2. Press (and hold) the tactile button and you should now see the LED diode blinks.

3. Make sure that you understand the difference when a tactile button is used to close the electrical circuit with and without output to the data pin.

Troubleshooting

1. LED lights are on when the current flow from anode (+) to cathode (-). Make sure that you connect the LED long leg (+) to the power/data pin, and the short leg (-) to Ground. With this set up, when you set the data pin to `HIGH` then the LED will light up and when you set the data pin to `LOW` then the LED will be off.

   However, if you connect the LED short leg (-) to the data pin, and the long leg (+) to power, the opposite will happen. So, when you set the data pin to `LOW` then the LED will light up, and when you set data pin to `HIGH` then the LED will be off.

Practice 4.2

1. Use the same connection as Practice 4.1 TO DO.

2. Type the code below into the coding area of your Arduino IDE. If you haven't done so already, use the links provided in the Theory part of this tutorial to make sure that you thoroughly understand the meaning of every line of the code below.

```
int LEDPin = 15;                                  – change 15 to 21 for WOKWI

void setup() {
  // put your setup code here, to run once:

  //You do not need to call pinMode() to set the pin for analogWrite()
}

void loop() {
  // put your main code here, to run repeatedly:

  int maxValue = 255;

}
```

   Note: Use practice4_2.json for WOKWI.

3. **Verify** the code. You will then be asked to save the sketch. Enter a **meaningful name** for your file, e.g., tutorial_04_analogwrite. Please **take note where you save your file**, so you know where to find it, should you need it afterwards.

4. **Upload** the code.

5. Press (and hold) the tactile button and you should now see the LED diode lights up.

TO DO

1. Reading the following Arduino reference page on `--` https://www.arduino.cc/reference/en/language/structure/compound-operators/decrement/ .

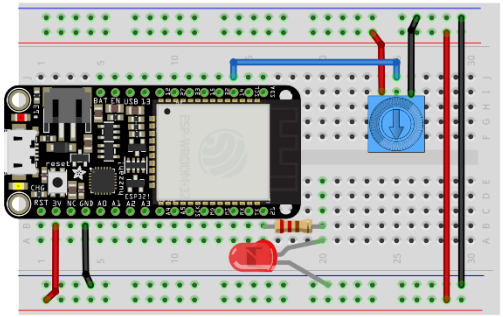
   Note: you may find it useful to read the Arduino reference pages on other compound operators too.
   https://www.arduino.cc/reference/en#compound-operators .

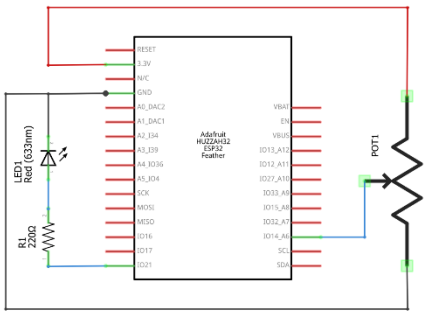
2. Modify tutorial_04_analogwrite.ino to fade the LED diode output by using `analogWrite()`. You should make sure that the brightness goes up and down smoothly and consistently. You may consider using loops to control the brightness rather than hardcoding all the steps manually.

3. **Verify** your modification and **Upload** the code to see if it still works correctly.

Practice 4.3

1. Connect potentiometer, LED diode and resistor according to the following schema. Note where the power (3V), the Ground (GND) and the data (Pin 14 for the potentiometer, and Pin 21 for the LED diode) are, and connect them correctly.





Source: https://makeabilitylab.github.io/physcomp/esp32/pot-fade.html

2. Type the code below into the coding area of your Arduino IDE. If you haven't done so already, use the links provided in the Theory part of this tutorial to make sure that you thoroughly understand the meaning of every line of the code below.

```
int LEDPin = 21;
int potentiometerPin = A0; // GPIO 26                    – change A0 to 14 for WOKWI

int maxAnalogue = 4095;
int maxDutyCycle = 255;

void setup() {
 // put your setup code here, to run once:

 // TO DO: set up the serial communication

 // TO DO: set up the pins and their modes

}

void loop() {
 // put your main code here, to run repeatedly:

 // TO DO: obtain input from the potentiometer
 int potV = 0;
 // TO DO: remap the possible input range into the correct possible output range
 //       and assign the remapped value to cycle
 int cycle = 0;

 // TO DO: change the brightness of the LED diode
```

```
  // Display 2 graphs on the Serial Plotter
  Serial.print("Potentiometer:");
  Serial.print(potV);
  Serial.print(" ");
  Serial.print("Cycle:"); // Label with a space will be made into 2 labels
  Serial.print(cycle);
  Serial.println();

  delay(10);

}
```

Note: Use practice4_3.json for WOKWI.

3. **Verify** the code. You will then be asked to save the sketch. Enter a **meaningful name** for your file, e.g., tutorial_04_potentiometer. Please **take note where you save your file**, so you know where to find it, should you need it afterwards.

4. If you have a Serial Plotter open already, please close it.

5. **Upload** the code.

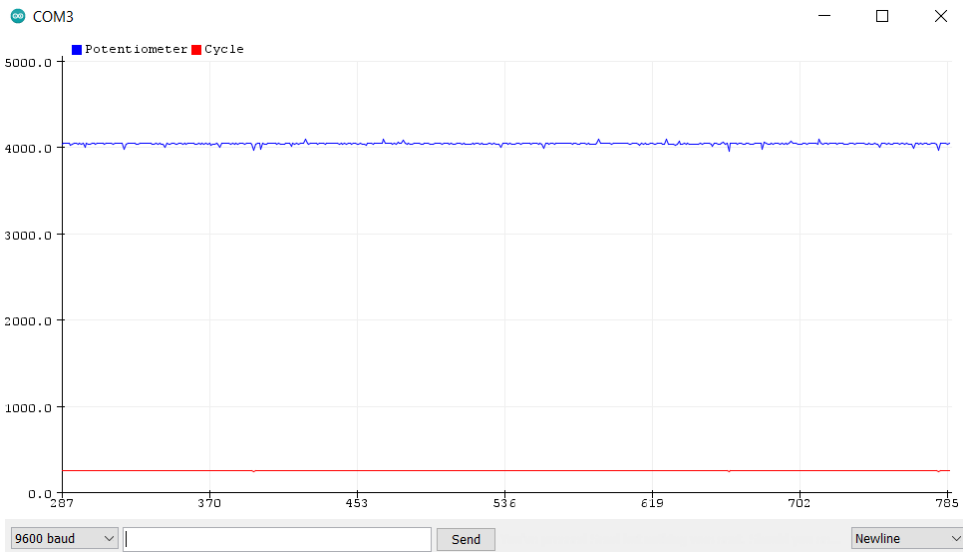
TO DO

1. Complete all of the TO DO tasks as shown in the comments in tutorial_04_potentiometer.ino.

2. If you have a Serial Plotter open already, please close it. **Verify** your modification.

3. **Upload** the code.

4. Go to **Tools** and then select **Serial Plotter**. You should now see the output of similar to what is shown below.


COM3
Potentiometer
Cycle
5000.0
4000.0
3000.0
2000.0
1000.0
0.0
287
370
453
536
619
702
785
9600 baud
Send
Newline

Practice 4.4

1. Modify the completed tutorial_04_potentiometer.ino (including the TO DO tasks) by remapping the original range of values for `potV` and `cycle` to the range of values between 0-100 and 0-50 respectively.

   Note: Use practice4_4.json for WOKWI, and remember to change potentioMeterPin in the sketch from A0 to 14 for WOKWI.

2. If you have a Serial Plotter open already, please close it.

3. **Verify** the code. You will then be asked to save the sketch. Enter a **meaningful name** for your file, e.g., tutorial_04_plotter. Please **take note where you save your file**, so you know where to find it, should you need it afterwards.

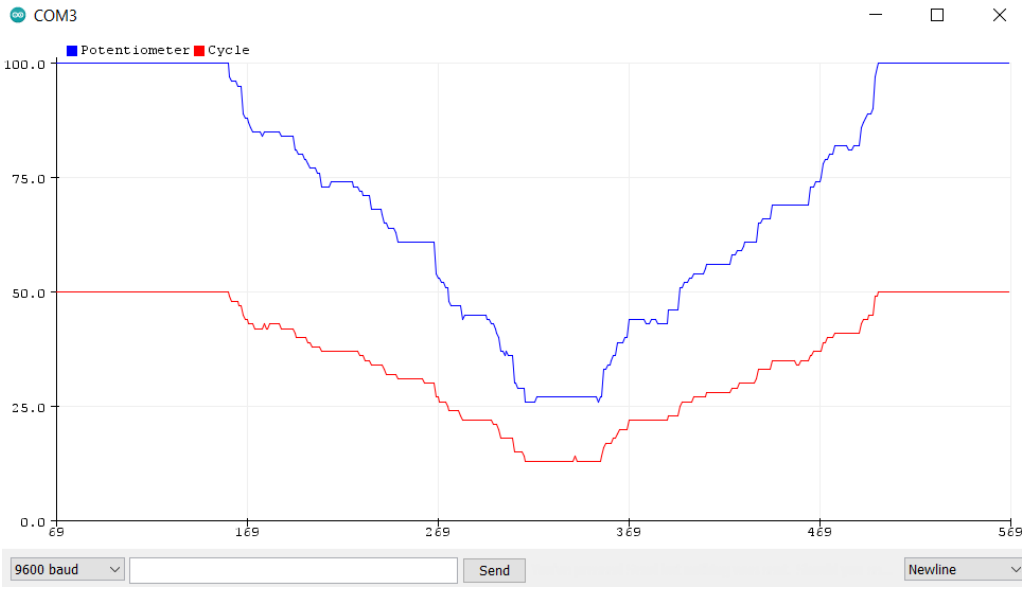
4. **Upload** the code.

TO DO

1. Read the following page, specifically on how to add min and max limits for the y-axis https://diyrobocars.com/2020/05/04/arduino-serial-plotter-the-missing-manual/ .

   Hint: the process is very similar to adding the labels.

2. Modify tutorial_04_plotter.ino to now limit the y-axis values between 0 and 100. Your code should produce an output similar to what is shown below.



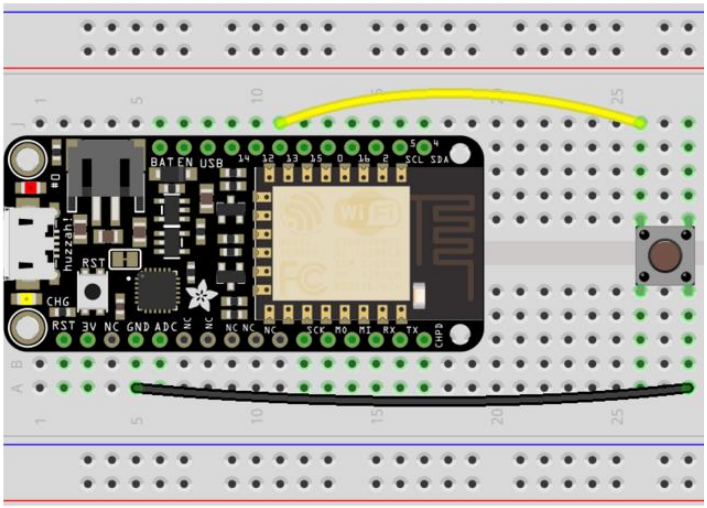
3. If you have a Serial Plotter open already, please close it.

4. **Verify** your modification and **Upload** the code.

5. Go to **Tools** and then select **Serial Plotter**. You should now see the output of similar to what is shown above.

Challenge 4.1

1. Connect a tactile button according to the following schema. Note where the Ground (GND) and the data (Pin 13) are and connect them correctly.



Source: https://learn.adafruit.com/adafruit-feather-huzzah-esp8266/usage

2. Type the code below into the coding area of your Arduino IDE. If you haven't done so already, use the links provided in the Theory part of this tutorial to make sure that you thoroughly understand the meaning of every line of the code below.

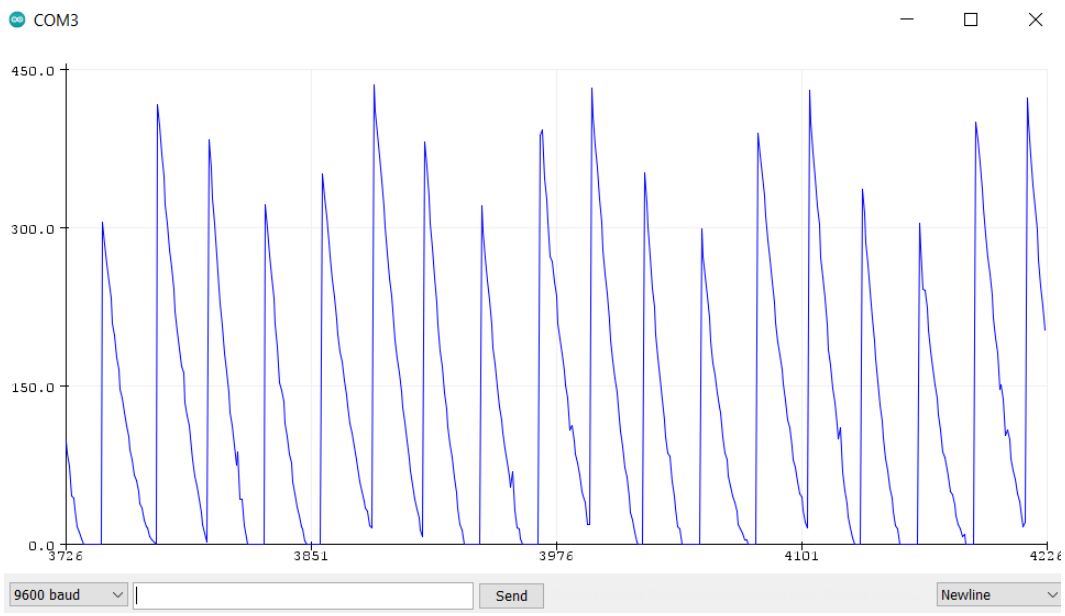
```
int buttonAPin = 13;

void setup() {
  // put your setup code here, to run once:

  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:

  Serial.println(analogRead(buttonAPin));
}
```

Note: Use challenge4_1.json for WOKWI.

3. If you have a Serial Plotter open already, please close it. **Verify** the code. You will then be asked to save the sketch. Enter a meaningful name for your file, e.g., tutorial_04_button_challenge. Please **take note where you save your file**, so you know where to find it, should you need it afterwards.

4. **Upload** the code.

5. Go to **Tools** and then select **Serial Plotter**. You should now see an output similar to what is shown below when the tactile button is NOT pressed.



6. The oscillation shown above indicates the "floating pin" problem which occurs because pins on Adafruit Huzzah32 are default as input pins, i.e., they do not need to be explicitly declared as inputs with `pinMode()`. Pins configured this way take very little current, e.g., electrical noise from the environment, or the state of a nearby pin, to move the pin from one state to another. This is one of the reasons to add a resistor to steer the input to a known state e.g., pulldown resistor to ground (0 V).

TO DO

1. In addition to `INPUT`, and `OUTPUT`, the function `pinMode()` also has a third option of using `INPUT_PULLUP` which is software based pullup resistors. As opposed to an external resistor pulling towards 0 V, this option connects the pin to the power, i.e., pulls towards 3.3 V. This, therefore, inverting the on/off state (or `HIGH` and `LOW`).

   You may wish to read about `pinMode()` and floating pin problem in more detail from https://roboticsbackend.com/arduino-input_pullup-pinmode/ .

2. Insert `pinMode(potAPin, INPUT_PULLUP);` in `setup()`. If you have a Serial Plotter open already, please close it. **Verify** and **Upload** the code.

3. You can test the inversion of on/off states by pressing the tactile button. The tactile button will now report `LOW` when the button is pressed (the red LED will be off), and the button will report `HIGH` when the button is not pressed (the red LED will be on).

Note: Pin 13 is also connected to the red LED next to the USB port and that is why the red LED is changing its state in corresponding to the pressing (or not pressing) of the tactile button.

4. Connect another tactile button to Pin 15 and an LED diode to Pin 21. Write a piece of code where the LED is turned on when the first tactile button (Pin 13) is pressed, and the LED is turned off when the second tactile button (Pin 15) is pressed. You can use `digitalRead(pin)` to check if the state of the tactile buttons.

   Hint: If you make changes from tutorial_04_button_challenge.ino, do make sure to remove/comment the `analogRead(potAPin)`.

   Note: Use challenge4_1_2.json for WOKWI.

5. **Verify** your modification and **Upload** the code to see if it now produces the required output.