

# The RPI Pico in the Lab Session 1 Tasks

# Session 1 Lab Tasks

## Install Software

### Arduino IDE

<https://www.arduino.cc/en/software>



### Arduino IDE 2.3.0

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

#### SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

#### DOWNLOAD OPTIONS

**Windows** Win 10 and newer, 64 bits

**Windows** MSI Installer

**Windows** ZIP file

**Linux** AppImage 64 bits (X86-64)

**Linux** ZIP file 64 bits (X86-64)

**macOS** Intel, 10.14: "Catalina" or newer, 64 bits

**macOS** Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

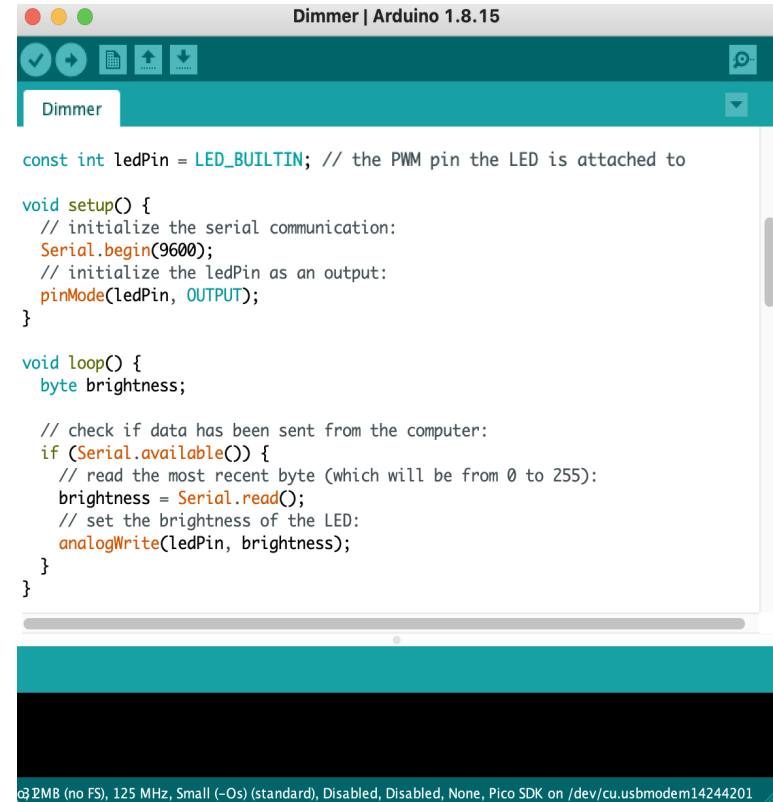
### Arduino-core for RPI Pico Installation

<https://arduino-pico.readthedocs.io/en/latest/>

# Session 1 Lab Tasks

## Write your first RPI pico program

- Look at menu “File/Examples”. Try out some sketches that use serial comms between PC and pico to become familiar with the programming environment, e.g. Dimmer and Temperature.



The screenshot shows the Arduino IDE interface with the 'Dimmer' sketch loaded. The title bar reads 'Dimmer | Arduino 1.8.15'. The code in the editor is as follows:

```
const int ledPin = LED_BUILTIN; // the PWM pin the LED is attached to

void setup() {
  // initialize the serial communication:
  Serial.begin(9600);
  // initialize the ledPin as an output:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  byte brightness;

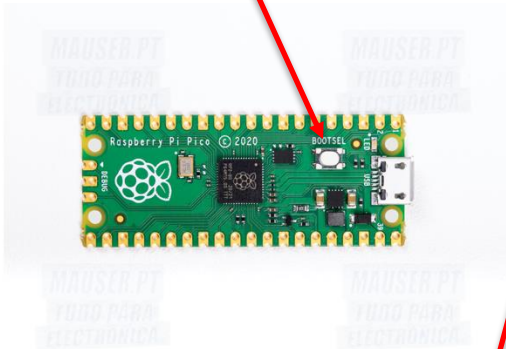
  // check if data has been sent from the computer:
  if (Serial.available()) {
    // read the most recent byte (which will be from 0 to 255):
    brightness = Serial.read();
    // set the brightness of the LED:
    analogWrite(ledPin, brightness);
  }
}
```

At the bottom of the IDE, the status bar displays: '2MB (no FS), 125 MHz, Small (-Os) (standard), Disabled, Disabled, None, Pico SDK on /dev/cu.usbmodem14244201'.

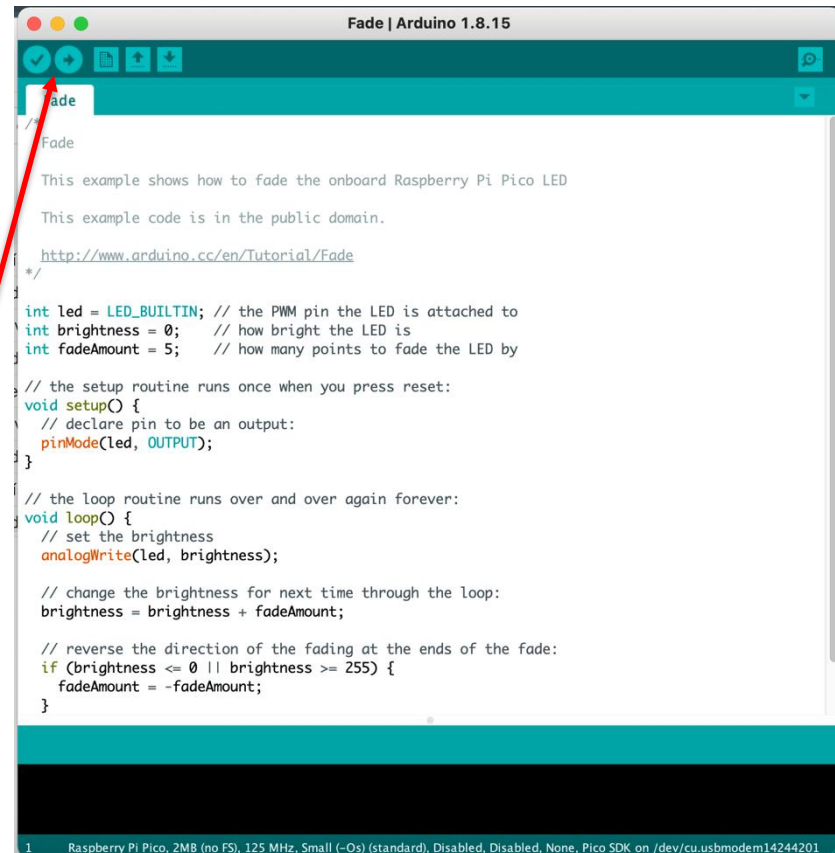
# Session 1 Lab Tasks

## Load Firmware

The first upload requires holding the BOOTSEL button before powering up the microcontroller (connecting to USB).



For the next times, just hit the upload button.

A screenshot of the Arduino IDE interface. The title bar reads 'Fade | Arduino 1.8.15'. The code editor shows the 'Fade' example code, which is used to fade the onboard Raspberry Pi Pico LED. The code includes comments and C++ syntax for setting up and looping through brightness levels. A red arrow points from the text above to the 'Upload' button (a right-pointing arrow) in the top toolbar of the IDE.

```
Fade

Fade

This example shows how to fade the onboard Raspberry Pi Pico LED

This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/Fade
*/

int led = LED_BUILTIN; // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness
  analogWrite(led, brightness);

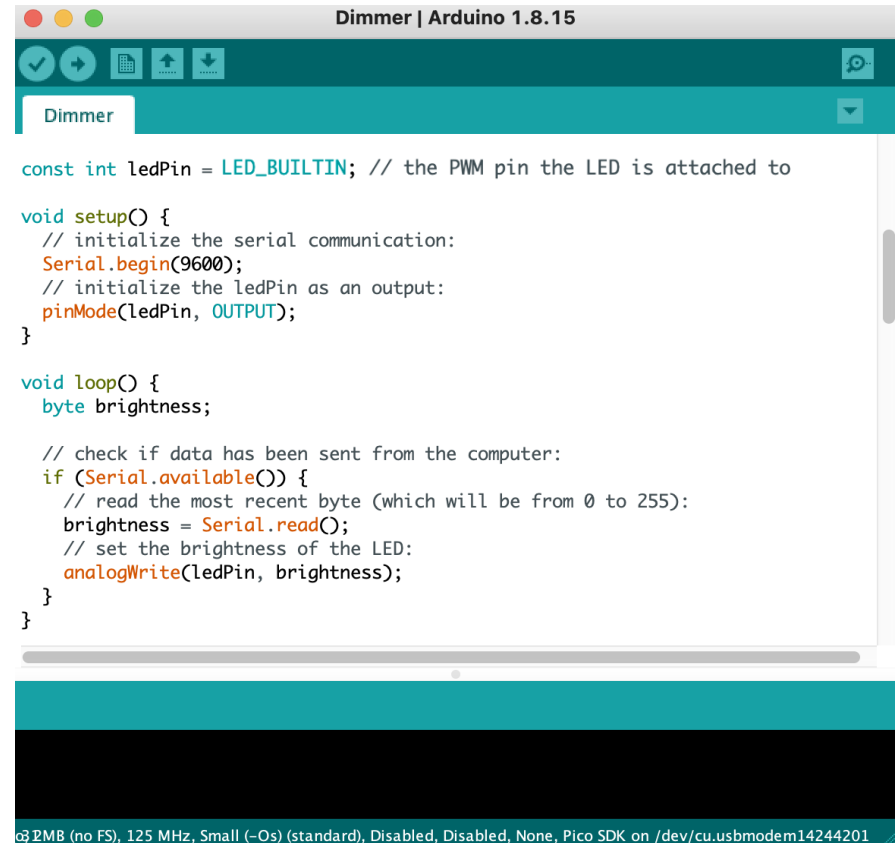
  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
}
```

# Week 1 Lab Tasks

## Serial USB Interface

- Look at some examples that use serial communication between PC and pico, e.g. Dimmer and Temperature.

A screenshot of the Arduino IDE interface. The title bar reads "Dimmer | Arduino 1.8.15". The menu bar includes "File", "Edit", "Tools", and "Help". The toolbar shows icons for opening, saving, uploading, and downloading. The main text area displays the following C++ code:

```
const int ledPin = LED_BUILTIN; // the PWM pin the LED is attached to

void setup() {
  // initialize the serial communication:
  Serial.begin(9600);
  // initialize the ledPin as an output:
  pinMode(ledPin, OUTPUT);
}

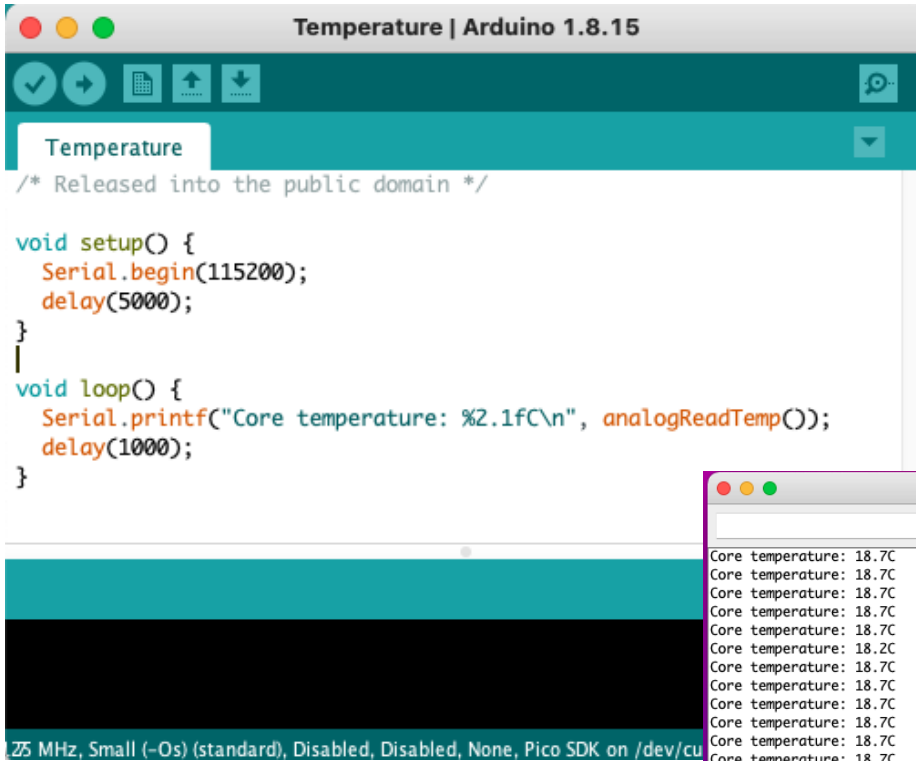
void loop() {
  byte brightness;

  // check if data has been sent from the computer:
  if (Serial.available()) {
    // read the most recent byte (which will be from 0 to 255):
    brightness = Serial.read();
    // set the brightness of the LED:
    analogWrite(ledPin, brightness);
  }
}
```

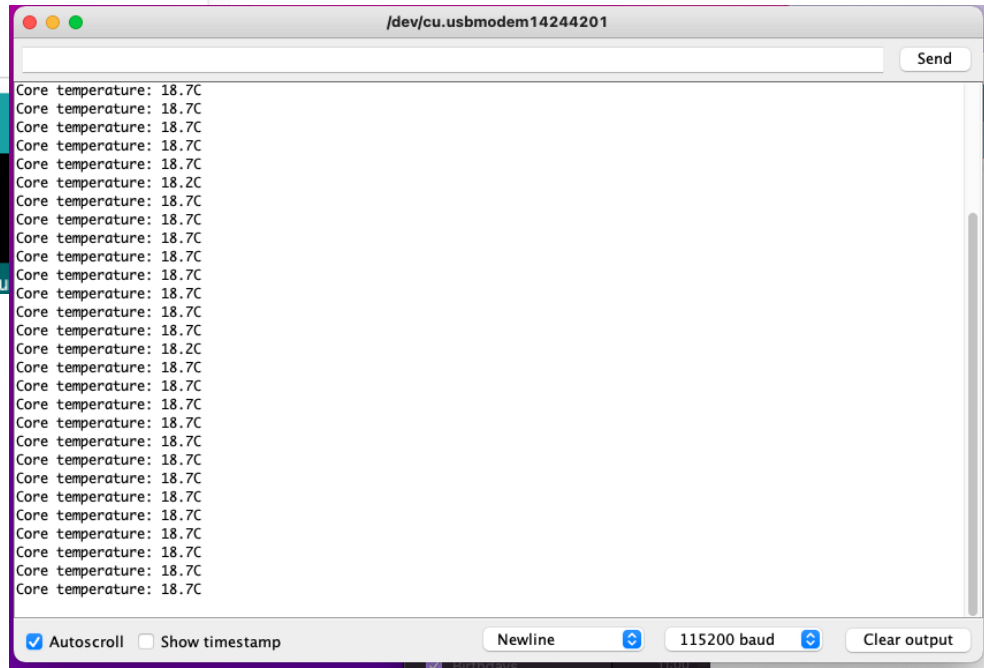
The status bar at the bottom indicates: "32MB (no FS), 125 MHz, Small (-Os) (standard), Disabled, Disabled, None, Pico SDK on /dev/cu.usbmodem14244201".

# Session 1 Lab Tasks

## Serial USB Interface



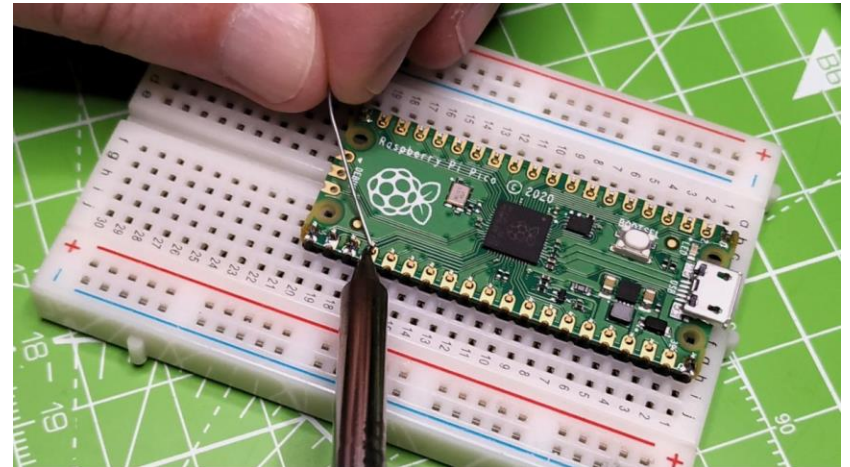
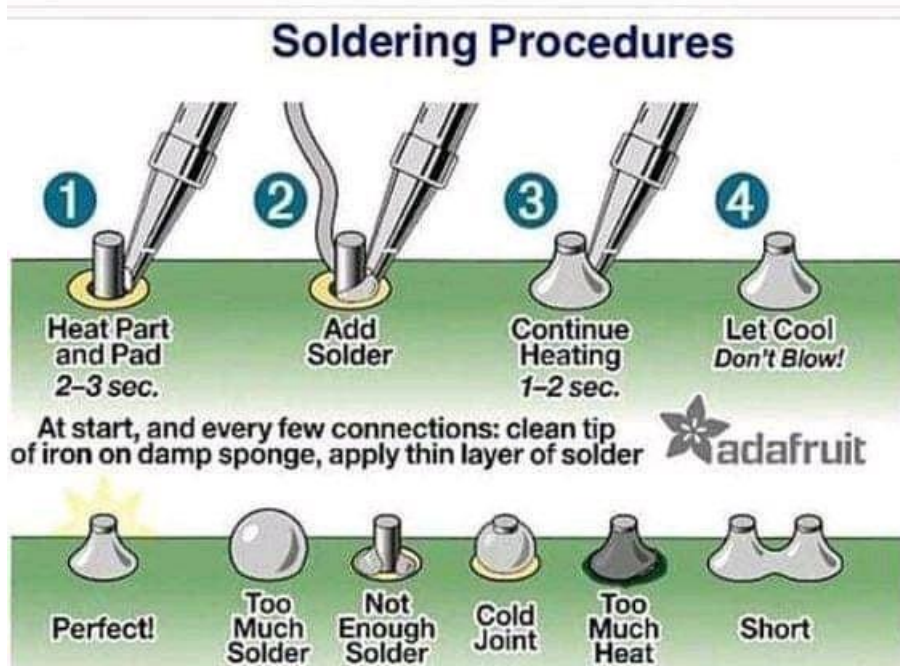
## Tools / Serial Monitor



# Session Lab Tasks

## Prepare the Pico

- Solder the header pins of rpi-pico and put it on the breadboard.

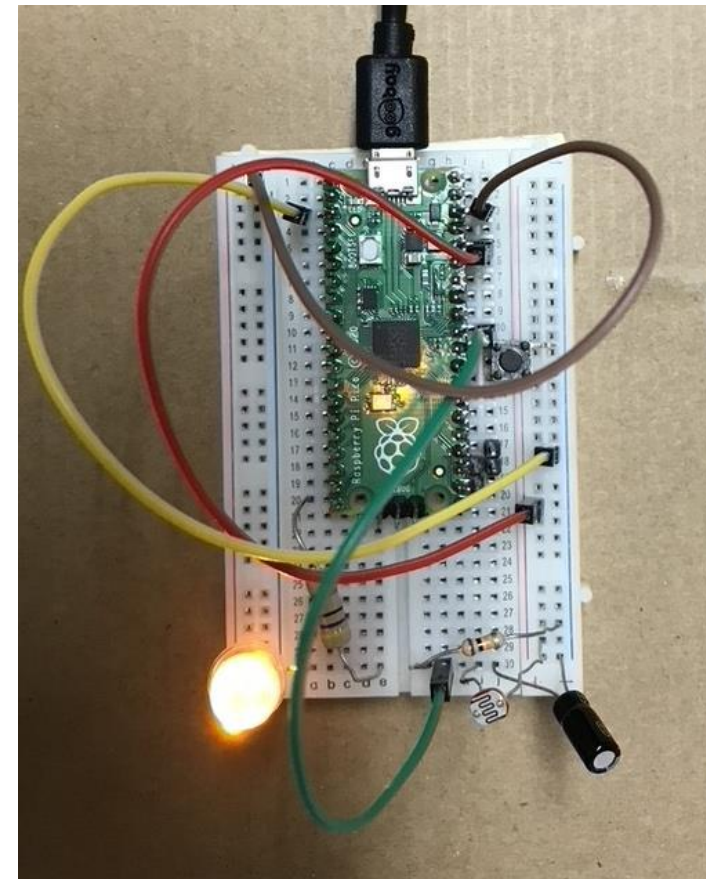
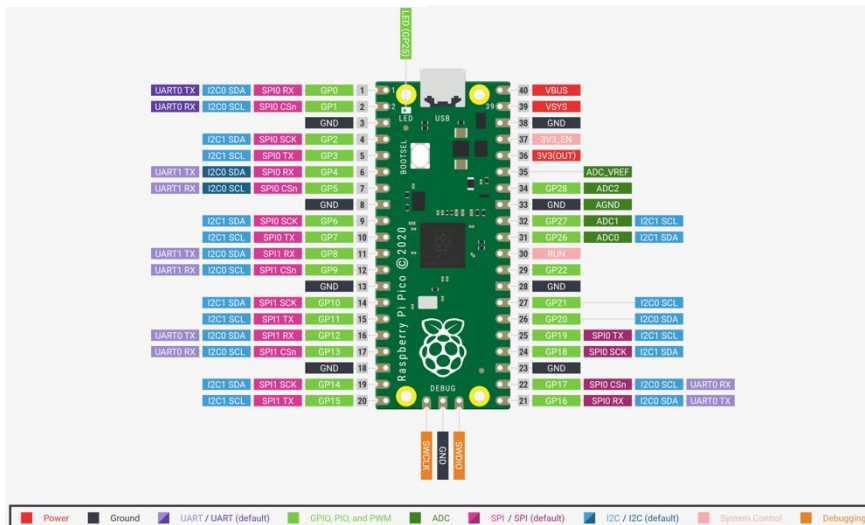
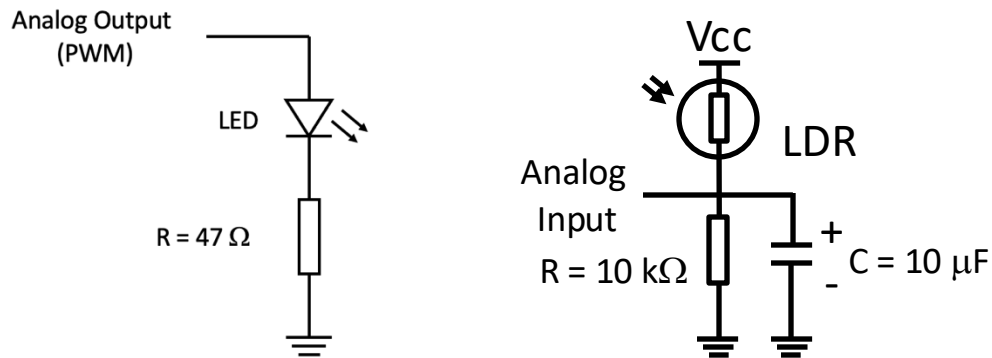




# Session 1 Lab Tasks

## Assemble the Breadboard Components

**Use tweezers to insert components into the breadboard.**





# Session 1 Lab Tasks

## Basic IO

```
const int LED_PIN = 15;

const int DAC_RANGE = 4096;

int counter = 0;

void setup() { // the setup function runs once
  Serial.begin(115200);

  analogReadResolution(12); //default is 10
  analogWriteFreq(60000); //60KHz, about max
  analogWriteRange(DAC_RANGE); //100% duty cycle
}

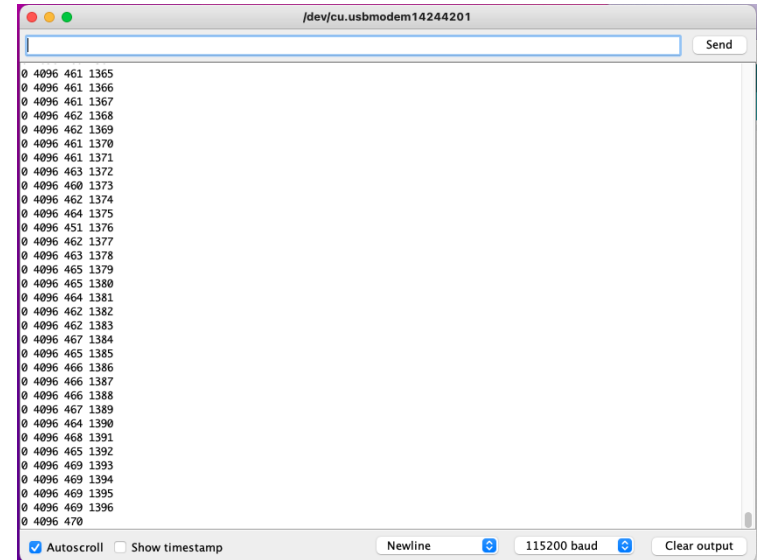
void loop() { // the loop function runs cyclically
  int read_adc;

  analogWrite(LED_PIN, counter); // set led PWM
  delay(1); //delay 1ms
  read_adc = analogRead(A0); // read analog voltage
  counter = counter + 1;

  if (counter > DAC_RANGE) // if counter saturates
    counter = 0; // reset counter

  //format that Serial Plotter likes
  Serial.print(0); Serial.print(" ");
  Serial.print(DAC_RANGE); Serial.print(" ");
  Serial.print(read_adc); Serial.print(" ");
  Serial.print(counter); Serial.println();
}
```

## Serial Monitor



## Serial Plotter

