Introducing Robotics Module A Unit #8 Pot

Contents

Module A Unit #8 variable resistor (pot)

Introduction to the pot (variable resistor)

Most resistors have a fixed value measured in ohms (Ω).

A pot is the short name for a potential divider. It can also be called a variable resistor; this is because you can alter the resistance by turning the top of the resistor. It varies from `0 – 10`kΩ. The symbol for resistance is Ω (ohm). In these instances, the pot is connected to pin `A0`, which is the analog pin. It can measure the voltage between `0` V and `5` V and returns a value of `0 – 1023`.

This simulates an input to an analog pin from a sensor, e.g. a gas sensor. Many sensors have a digital signal output.

What you will need

1 x Arduino Uno
1 x breadboard
1 x LED traffic lights
1 variable resistor (pot)
9 x male-to-male jumper leads

Keeping the traffic lights the same, add the pot as shown in the wiring diagram below. The centre pin connects to analog pin `A0`. Of the two outside pins, one connects to the 5V pin and the other to ground.

Circuit Diagram for unit #8 the pot

Connection pins between the pot, LED traffic lights and the Arduino Uno.

Pot	Arduino Pins
VCC	5v
Out	A0
GND	GND

Traffic Light Pins	Arduino Pins
GND	GND
R (red)	11
Y (yellow)	10
G (green)	9

Figure 1a: Circuit Diagram for pot and LEDs

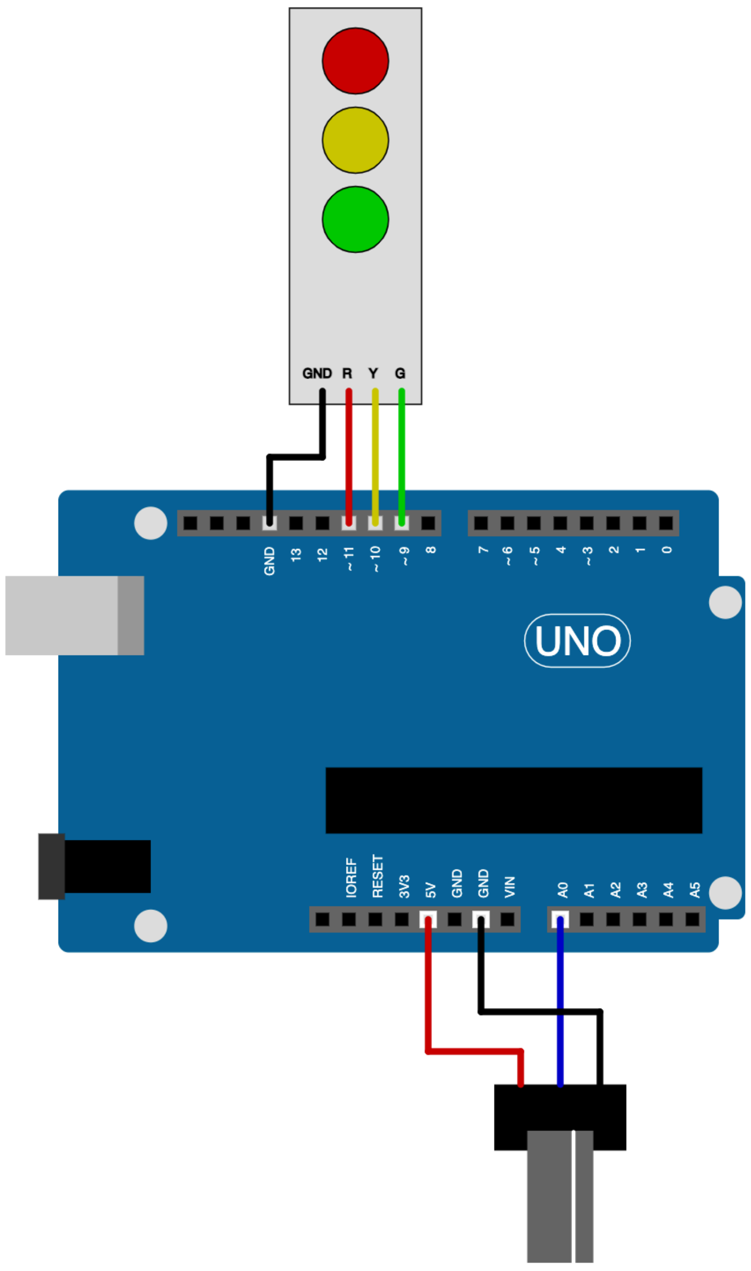
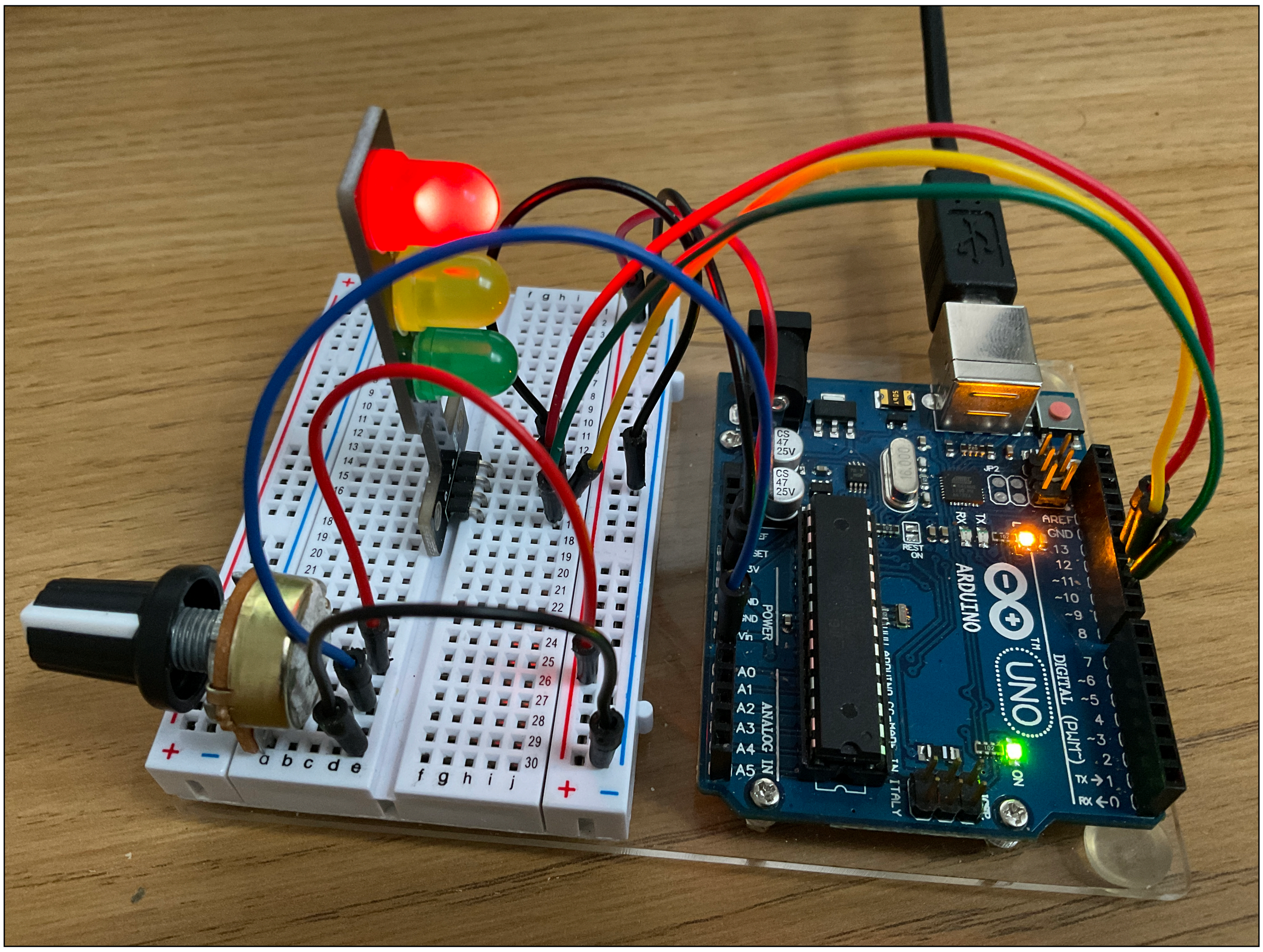


Figure 1b: Connecting pot and LED traffic lights



Important notes

Remember to connect the ground GND and 5V from the Arduino to the blue rail and red rail. You can also connect the GND from the traffic lights LED to the ground rail (blue).

Sketch A8.1 pot basic

The value of the pot is sent to the serial monitor as you turn it with a small screwdriver. It reads from 0 volts to 5 volts, which translates to a reading from 0 to 1023.

```
int potValue = 0;


void setup()
{
  Serial.begin(9600);
}


void loop()
{
  potValue = analogRead(A0);
  Serial.println(potValue);
  delay(100);
}
```

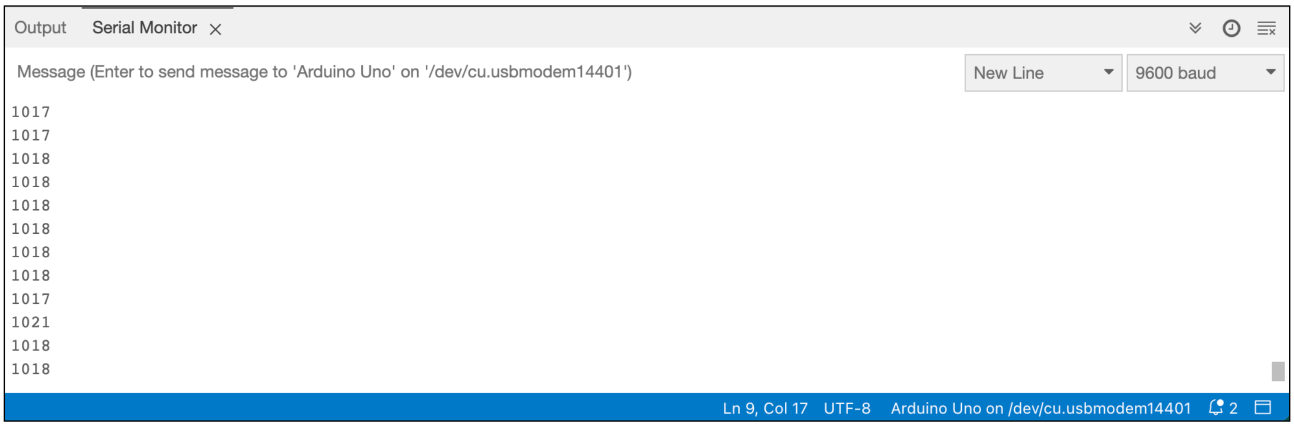
Notes

Now click on the serial monitor icon in the top right. Alternatively, you can click on Tools, then Serial Plotter, but remember to close the Serial Monitor first. Also, whenever you load a new sketch, close the monitor from the previous sketch before continuing whichever monitor you are using. You will notice that it may not go down to zero or up to 1023.

Code Explanation

`potValue = analogRead(A0);`	Here you are reading the value on pin A0 as an analog signal rather than a digital one

Figure A8.1a: Turn the pot for one extreme to the other

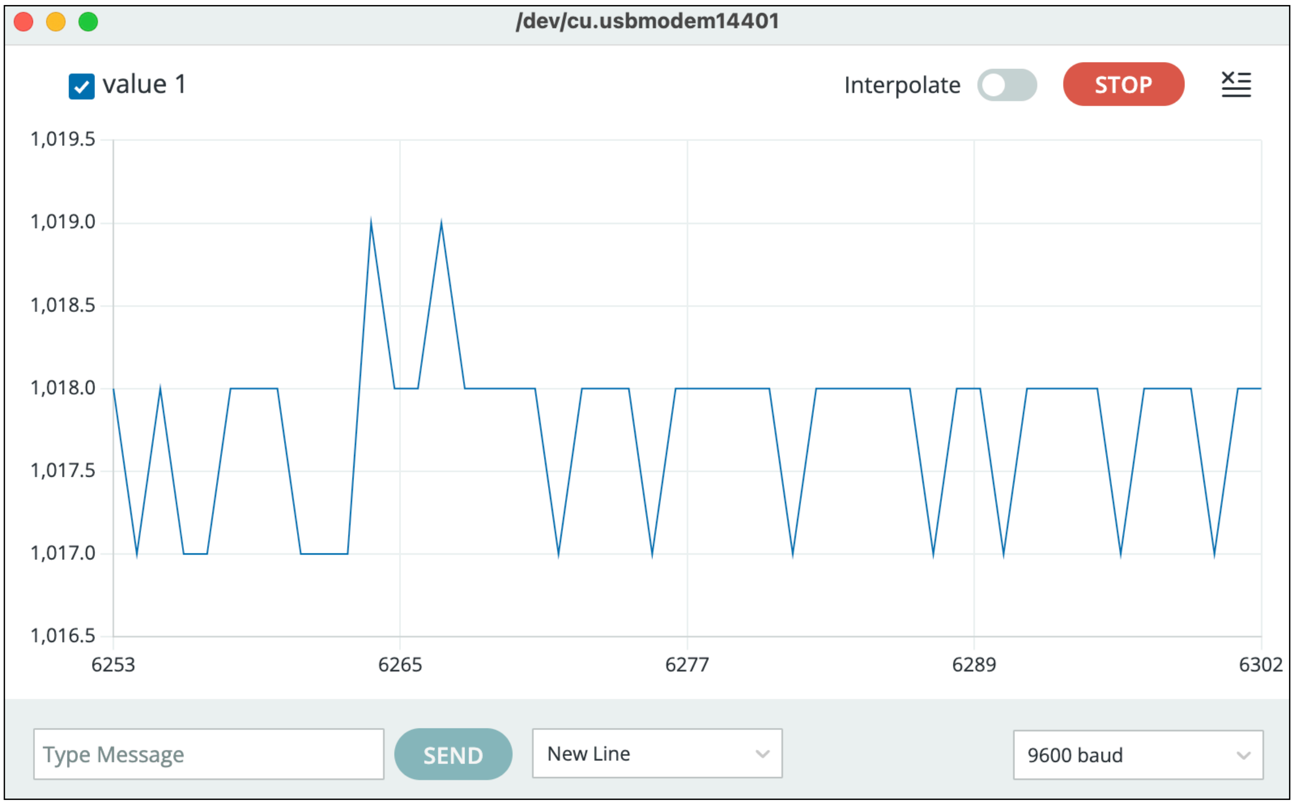


Message (Enter to send message to 'Arduino Uno' on '/dev/cu.usbmodem14401')

```
1017
1017
1018
1018
1018
1018
1018
1018
1017
1021
1018
1018
```

Ln 9, Col 17 UTF-8 Arduino Uno on /dev/cu.usbmodem14401 2

Figure A8.1b: Using the serial plotter turn the pot from one extreme to the other



Tweaking the code

As you can see when you click on the serial plotter (again top right icon), the scale (vertical values move around a little bit annoyingly). You can get a better idea by adding in some fixed values at `0` and `1030`. See the code below.

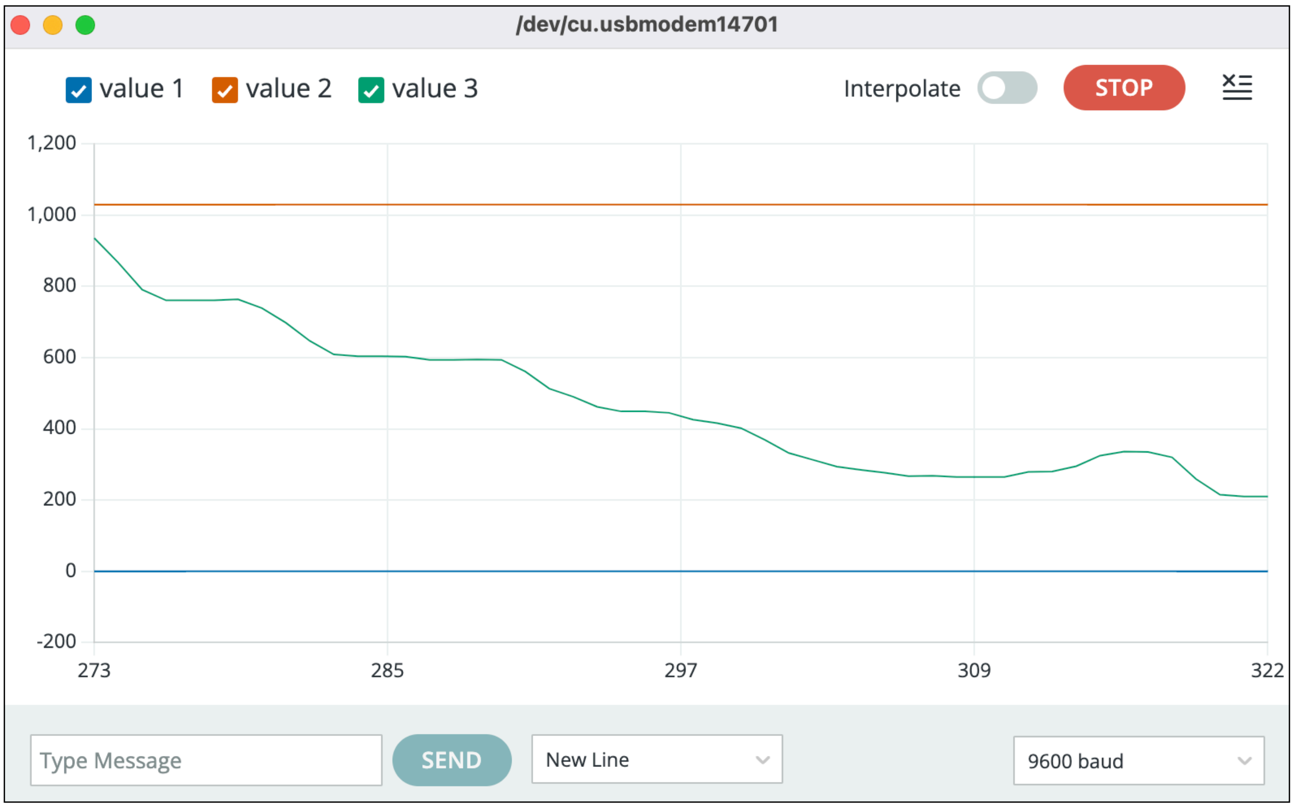
As an alternative to the above code, we can add in some constants so that the chart doesn't alter the scale wildly. This is not essential but a useful tip if you want to keep the y-scale constant.

```
int potValue = 0;
int bottom = 0;
int top = 1030;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  potValue = analogRead(A0);
  Serial.print(bottom);
  Serial.print(" ");
  Serial.print(top);
  Serial.print(" ");
  Serial.println(potValue);
  delay(100);
}
```

Figure A8.1c: As you turn the pot you get a much better idea of where the value of the pot is



Sketch A8.2 the bare minimum

Here we use a function to limit the maximum value (in this case 500), so it prints the minimum value up to 500.

```
int potValue = 0;


void setup()
{
  Serial.begin(9600);
}


void loop()
{
  potValue = analogRead(A0);
  potValue = min(potValue, 500);
  Serial.println(potValue);
  delay(100);
}
```

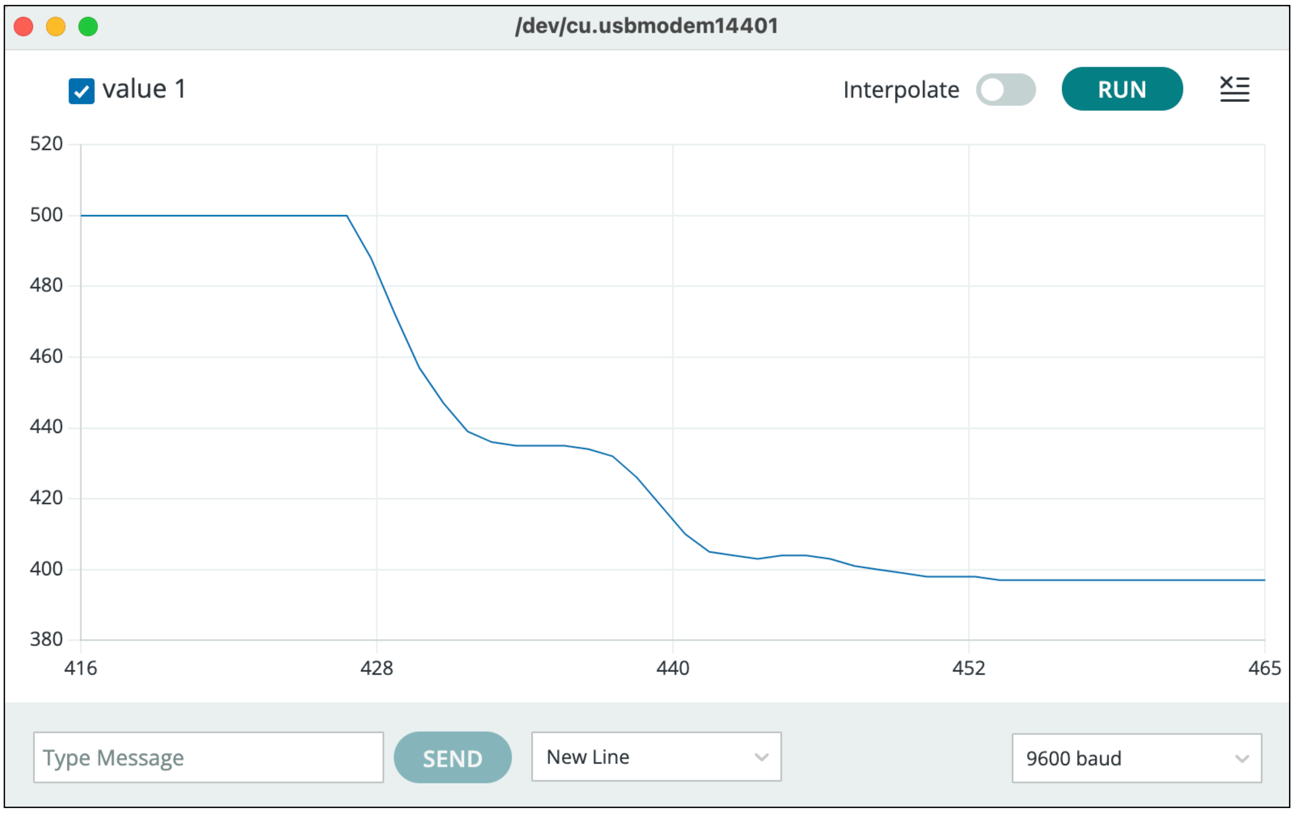
Notes

As you turn the pot, you will find that the value never goes above 500. In the image below, I paused (clicked STOP) the plotter.

Code Explanation

`potValue = min(potValue, 500);`	Returns the lowest value of the two

Figure A8.2: The value never goes above 500



Sketch A8.3 the absolute max

Similar to the min() function we can also have a max() function.

```
int potValue = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  potValue = analogRead(A0);
  potValue = max(potValue, 500);
  Serial.println(potValue);
  delay(100);
}
```

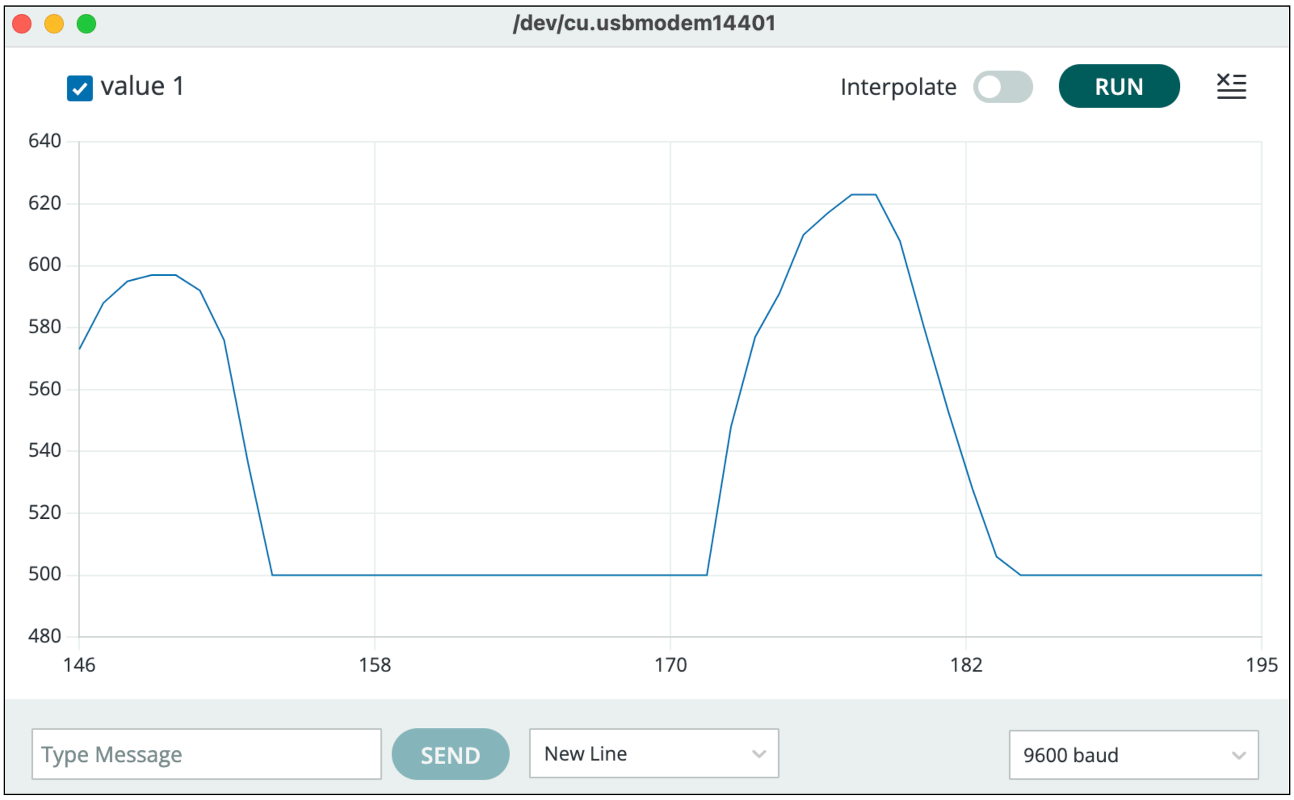
Notes

Now you can't bring it below 500 when you turn the pot.

Code Explanation

`potValue = max(potValue, 500);`	Returns the highest value of the two

Figure A8.3: The lowest value is now 500, it returns the maximum of the two values.



Sketch A8.4 feeling very constrained

We can also constrain the value between two fixed limits using the constrain() function.

```
int potValue = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  potValue = analogRead(A0);
  potValue = constrain(potValue, 100, 500);
  Serial.println(potValue);
  delay(100);
}
```

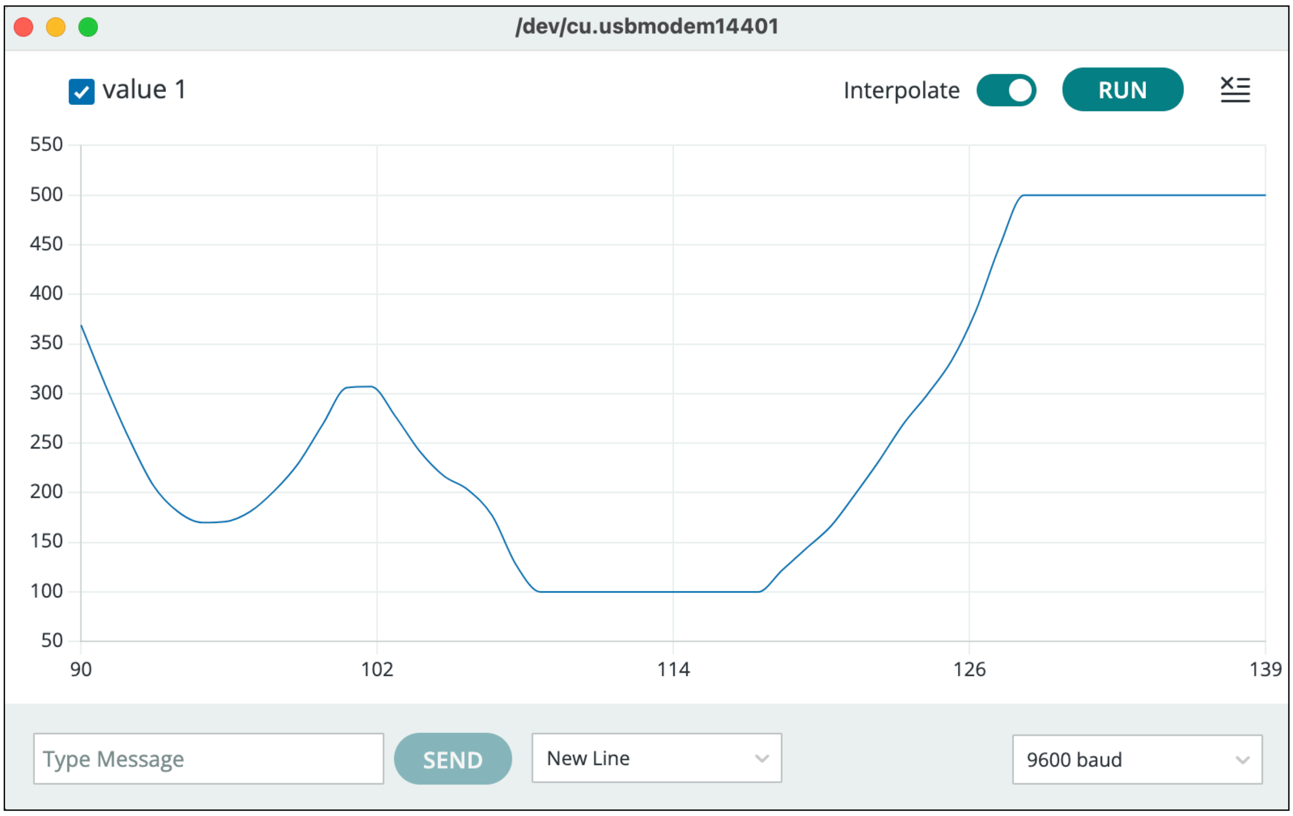
Notes

Now it will only venture between 100 and 500; you have constrained the output.

Code Explanation

`potValue = constrain(potValue, 100, 500);`	Returns values between the two limits

Figure A8.4: The values are limited between 100 and 500, interpolate seems to smooth out the curves



Sketch A8.5 pot blink

Starting a new sketch as we are going to use the traffic lights LEDs. We are going to control the rate of blinks (the amount of delay) for the LED on pin 11 by turning the pot.

```
int delayPeriod = 0;
int ledPin = 11;

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  delayPeriod = analogRead(A0);
  digitalWrite(ledPin, HIGH);
  delay(delayPeriod);
  digitalWrite(ledPin, LOW);
  delay(delayPeriod);
}
```

Mapping

The concept of mapping is used across many coding languages. It allows you to scale one range to another; for instance, if you wanted 0 -> 60 seconds to be represented by 0 -> 100, effectively turning 1 minute (60 seconds) into 100%, then you would use mapping. The line of code might look something like this:

```
int percentValue = map(seconds, 0, 60, 0, 100);
```

The output `percentValue` will take the seconds and convert it to a percentage. It converts the scale of 0 -> 60 to 0 -> 100.

It is probably easier to see it in action. In the example next, we take the pot value, which has a range of 1 -> 1023, and change it so that it outputs a value of 0 -> 255, so we can use it to control the brightness of the LED. So we change the maximum from 1023 to 255 and scale it all the way down to 0.

Sketch A8.6 pot the fade

I recommend starting a new sketch as there are too many changes. As explained above, we will use the pot to control the brightness of the LED on pin 2, but we have to change the scale from 0 -> 1023 to 0 -> 255, and for that, we can use a function called map().

```
int potValue = 0;
int ledBrightness = 0;
int ledPin = 11;

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  potValue = analogRead(A0);
  ledBrightness = map(potValue, 0, 1023, 0, 255);
  analogWrite(ledPin, ledBrightness);
}
```

🛠 Code Explanation

`ledBrightness = map(potValue, 0, 1023, 0, 255);`	Returns a value that is a adjusted from one range to another in proportion

Sketch A8.7 cycling through LEDs

❗ New sketch!

Using the value from the pot to define which LED is on. As you turn the pot a different LED will be lit depending on the value of the pot. These are arbitrary values.

```
void setup()
{
  for (int i = 9; i < 12; i++)
  {
    pinMode(i, OUTPUT);
  }
}


void loop()
{
  int potVal = analogRead(A0);
  if (potVal < 300)
  {
    digitalWrite(9, LOW);
    digitalWrite(10, LOW);
    digitalWrite(11, HIGH);
  }
  if (potVal > 300 && potVal < 600)
  {
    digitalWrite(9, LOW);
    digitalWrite(10, HIGH);
    digitalWrite(11, LOW);
  }
  if (potVal > 600)
  {
    digitalWrite(9, HIGH);
    digitalWrite(10, LOW);
    digitalWrite(11, LOW);
  }
}
```