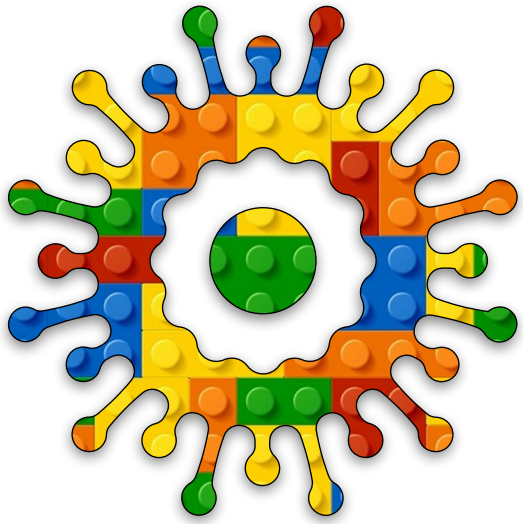
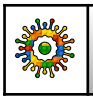
Internet of Things
Module A
Unit #9
the pot





Contents

Module A Unit #9 the variable resistor (pot)

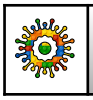
Sketch A9.1	pot basic
Sketch A9.2	the bare minimum
Sketch A9.3	the absolute max
Sketch A9.4	feeling very constrained
Sketch A9.5	pot blink
Sketch A9.6	pot the fade
Sketch A9.7	cycling through LEDs

Introduction to variable resistor (pot)

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 to 10 kOhms. 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 volts and 3.3 volts and returns a value of 0 to 1023. You can use it straight on the breadboard horizontally.

Figure 1: the pot we will use

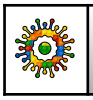


What you will need

1 x Arduino Nano 33 IoT
1 x breadboard
1 x LED traffic lights
1 x variable resistor (pot)
3 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 3.3 volts pin and the other to GND.

! Very important: DO NOT use the 5 volts !



Circuit Diagram for the pot & LEDs

Connection pins between the pot, LED traffic lights, and the Arduino Nano 33 IoT.

Pot	Arduino Pins
First pin	3.3v
Middle pin	A0
Last pin	GND

LED just as before.

Traffic Lights	Arduino Pins
GND	GND
R (red)	2 (D2)
Y (yellow)	3 (D3)
G (green)	4 (D4)

The two outer pins can go to either the 3.3 volts or the GND (ground) on the Arduino Nano 33 IoT board. The middle one has to go to one of the analog pins; any will do, but I have chosen analog pin 0.

I have provided two views of the pot in situ.

Figure 2a: pot front

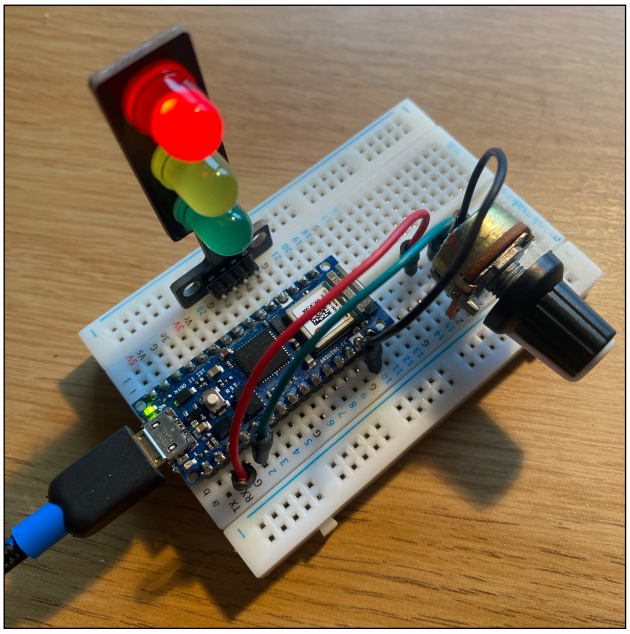


Figure 2b: pot rear

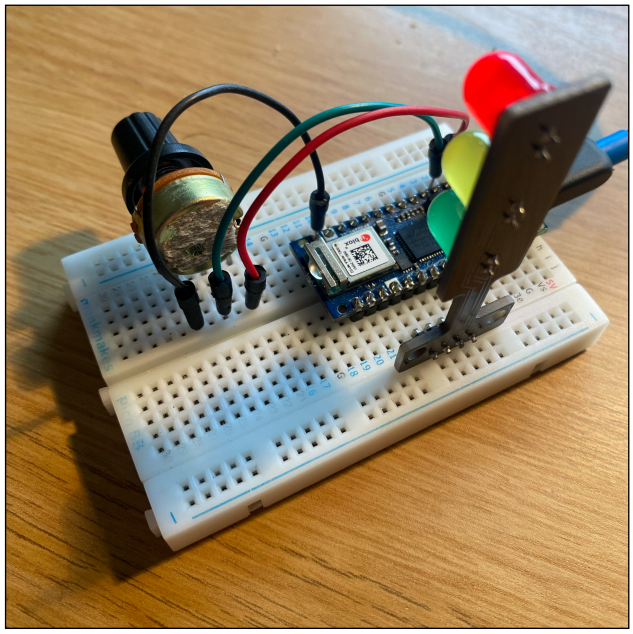
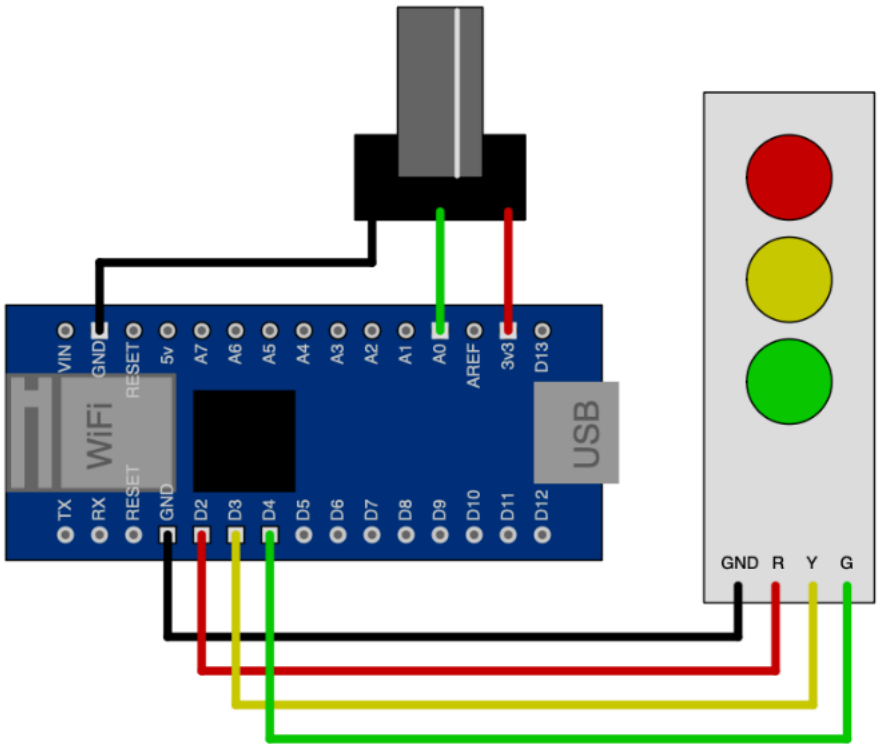
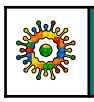


Figure 3: circuit diagram for pot





Sketch A9.1 pot basic

The value of the pot is sent to the serial monitor as you turn it. It reads from 0 volts to 3.3 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);
}
```

🛠️ Code Explanation

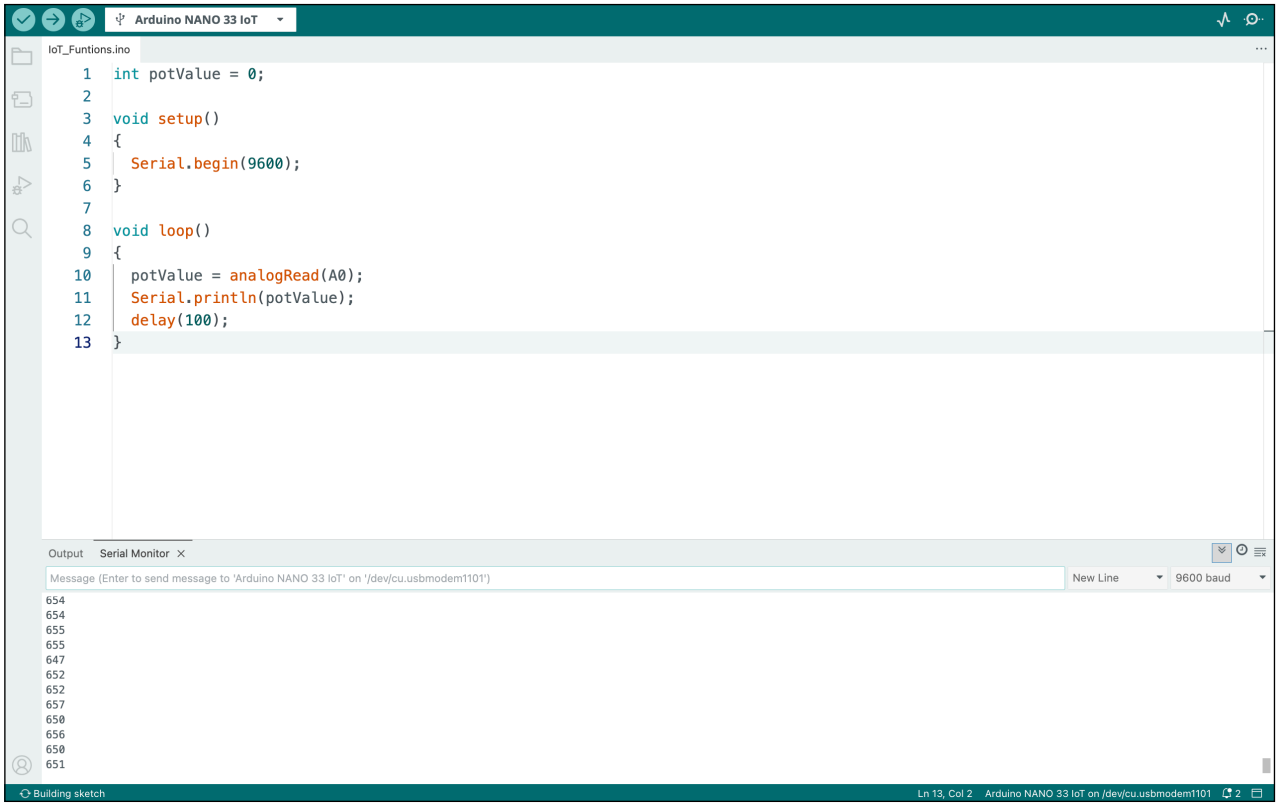
`potValue = analogRead(A0);`	Here you are reading the value on pin A0 as an analog signal rather than a digital one

Once you have uploaded the sketch, click on the serial monitor icon that looks like a magnifying glass (top right-hand corner).

Figure 4: Serial Monitor icon

You will get some values on the console at the bottom of the page. As you turn the pot, those values will change from `0` to `1023`. You will also notice that there is some noise in the values when you stop turning. You can dampen them by having an array take an average of a number of inputs.

Figure A9.1a: Serial Monitor

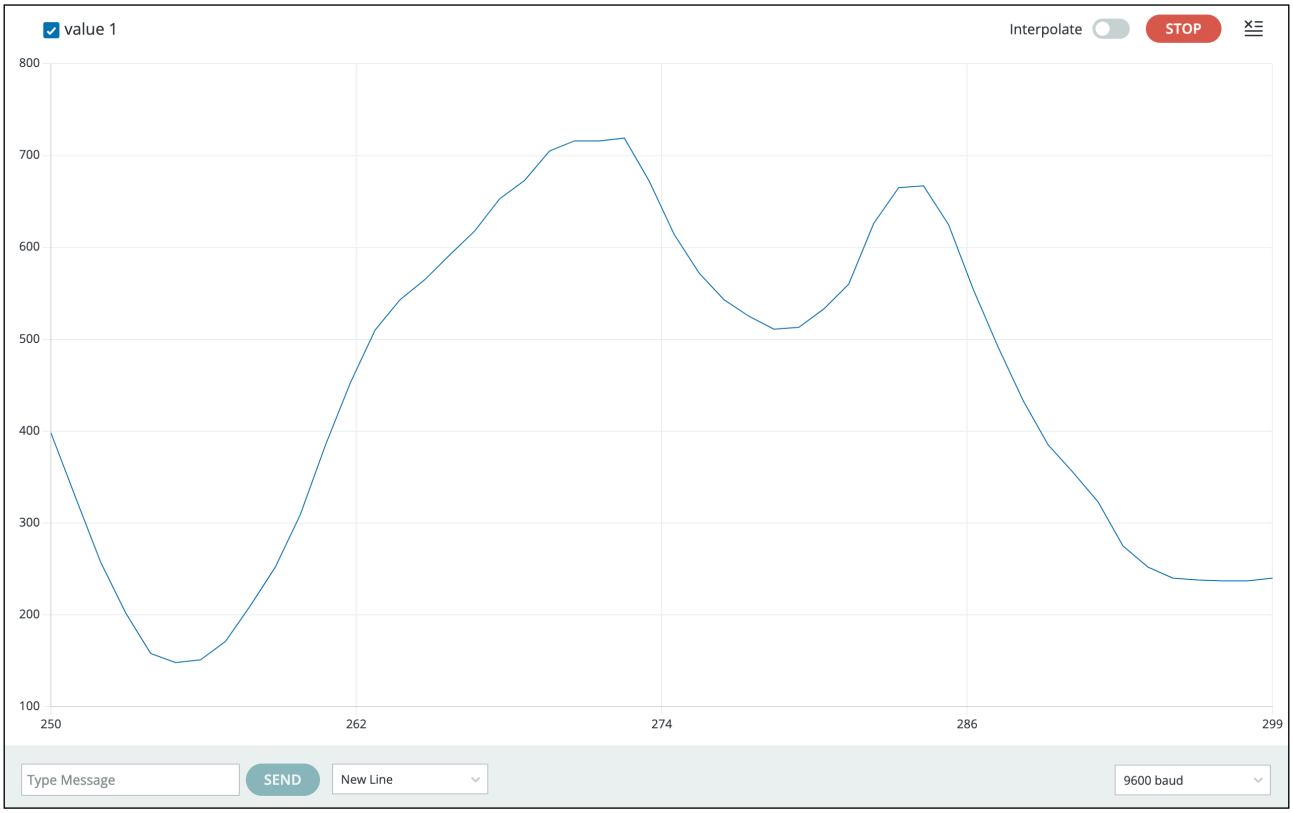


Now click on the other icon that looks like a graph or chart.

Figure 5: Serial Plotter icon

Here we can see graphically the change in values as you turn the pot. This is displayed in a new window. The values can be interpolated (see switch) and the graph stopped (STOP).

Figure A9.1b: Serial Plotter



Serial Plotter

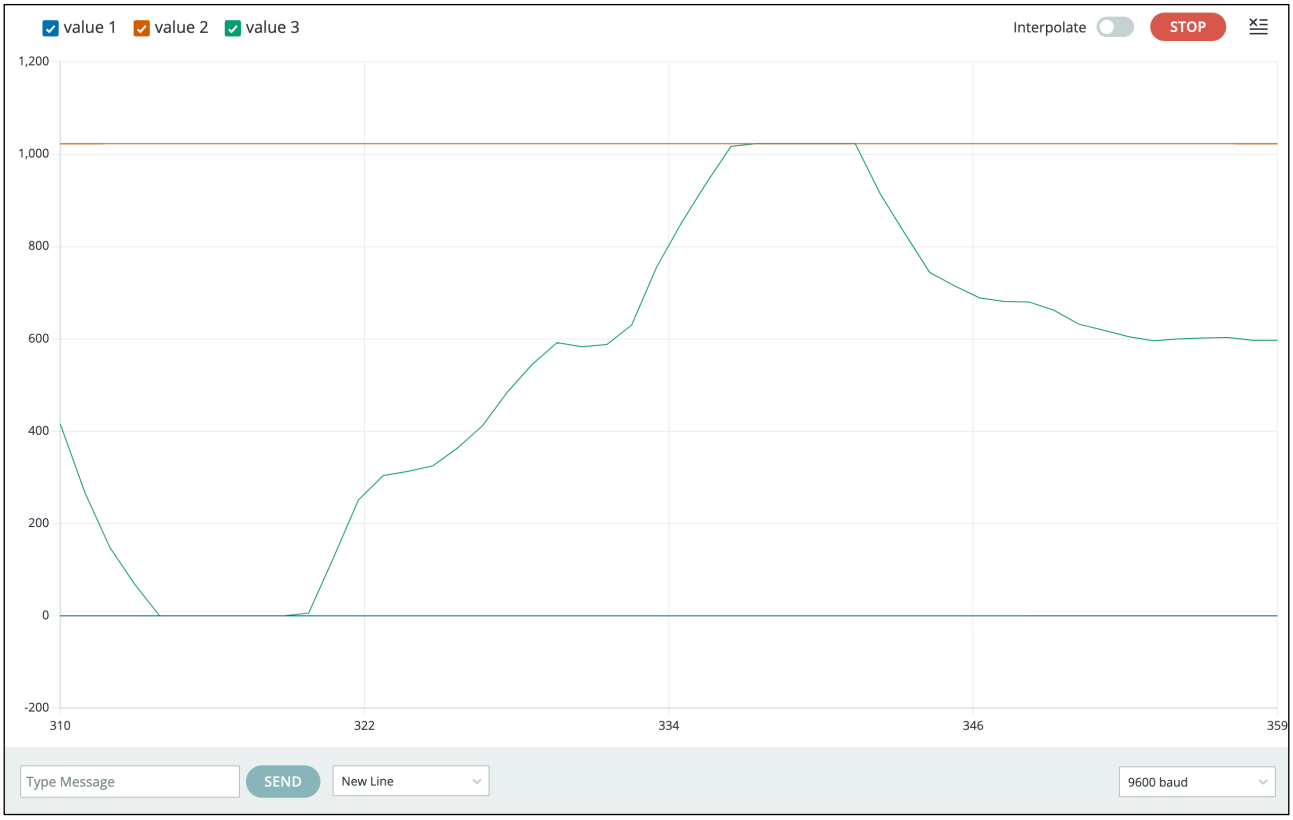
The scale of the values changes wildly as you change the pot. To mitigate this, we can create an upper and lower reference point. This keeps the scale on the left-hand side stable. Just add in the new code, upload, and restart the Serial Plotter. Not essential but useful.

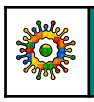
```
int potValue = 0;
int bottom = 0;
int top = 1023;

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);
}
```

Notes

You can see the improvement straight away. You might need to close the old Serial Plotter before it takes effect. The green line is the pot.

Figure A9.1c: upper and lower reference lines





Sketch A9.2 the bare minimum

Here, I 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);
}
```

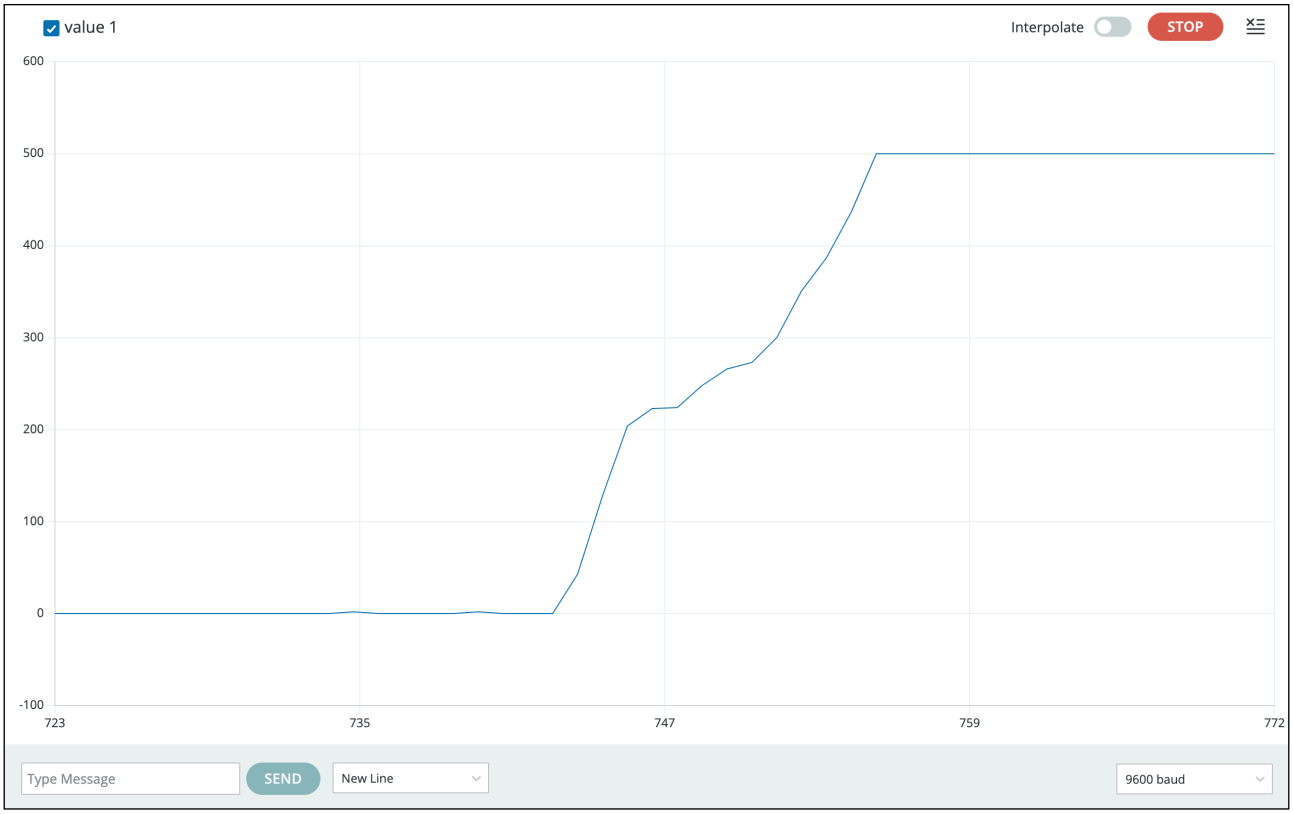
🗒️ Notes

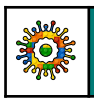
As you turn the pot, you will find that the value never goes above 500.

🛠️ Code Explanation

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

Figure A9.2: limiting the max value, returning the min





Sketch A9.3 the absolute max

Similar to the `min()` function, we can also have a `max()` function. It returns the maximum value above and including `500`.

```
int potValue = 0;


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


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

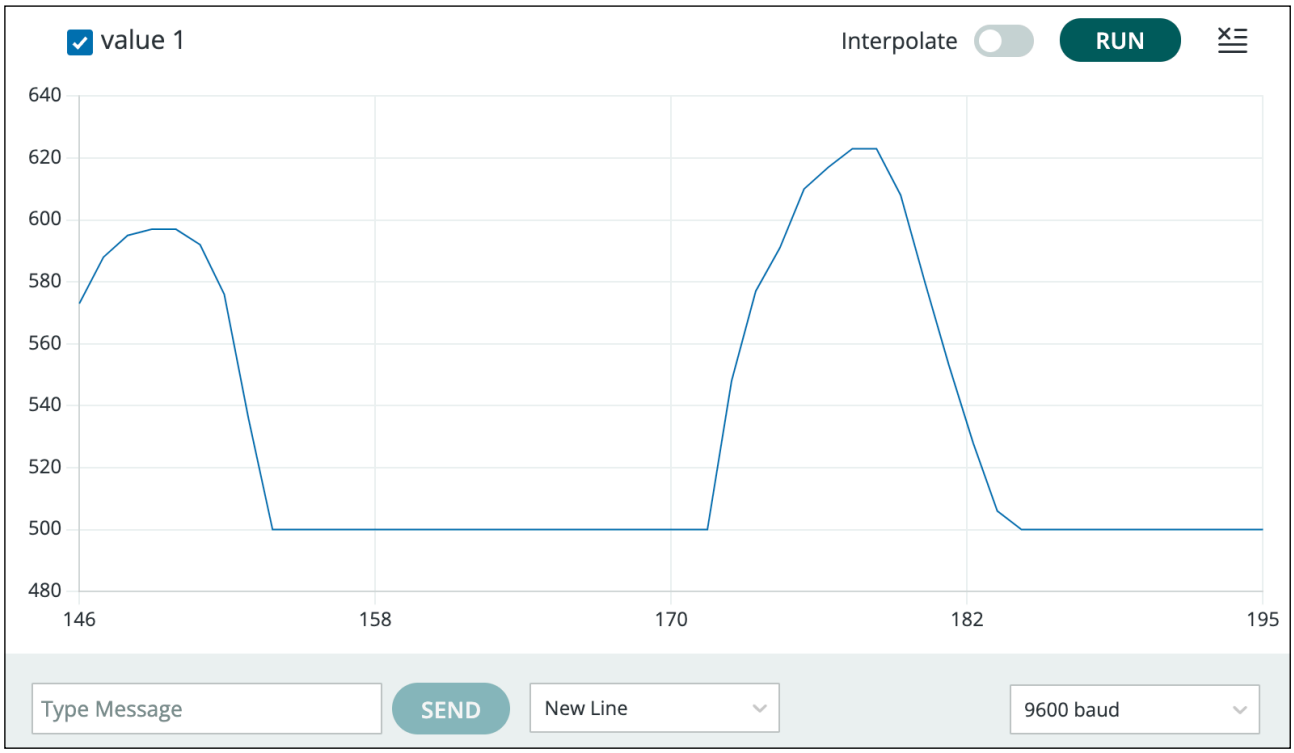
🗒️ Notes

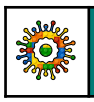
Now you can't bring below `500` when you turn the pot.

🛠️ Code Explanation

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

Figure A9.3: The lowest value is now 500, it returns the maximum of the two values.





Sketch A9.4 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);
}
```

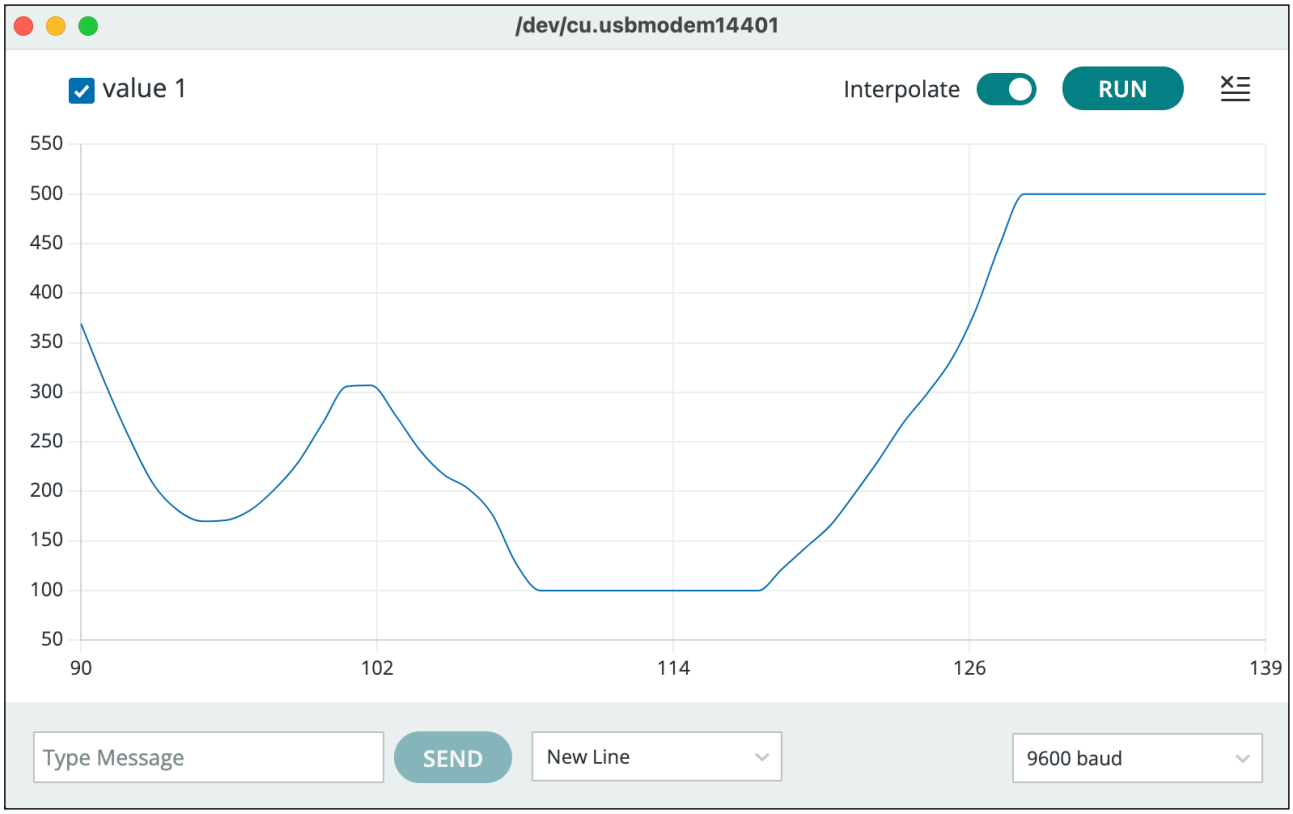
Notes

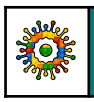
Now it will only venture between `100` and `500`; you have constrained the output.

Code Explanation

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

Figure A9.4: The values are limited between 100 and 500, interpolate seems to smooth out the curves





Sketch A9.5 pot blink

Starting a new sketch as we are going to use the LED traffic lights. We are going to control the rate of blinks (the amount of delay) for the LED on digital pin 2 by turning the pot.

```
int delayPeriod = 0;
int ledPin = 2;

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

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

Notes

Because of the noise, you get a flicker when you turn it down to 0. To stop that, you could always have a constraint function or a max to stop it from going to 0. Or use an `if()` statement for any value below, say, 10, to be equal to 0.

Mapping

The concept of mapping is used across a number of coding languages. It allows you to scale one range to another; for instance, if you wanted 0 to 60 seconds to be represented by 0 to 100, effectively turning 1 minute 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 is probably easier to see it in action. In the example next, we take the pot value, which has a range of 0 to 1023, and change it so that it outputs a value of 0 to 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 A9.6 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 digital pin 2, but we have to change the scale from 0 to 1023 making it 0 to 255 instead, and for that, we can use a function called map().

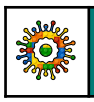
```
int potValue = 0;
int ledBrightness = 0;
int ledPin = 2;

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

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

Notes

Again, some flickering when close to 0.

Code Explanation

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



Sketch A9.7 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 = 2; i < 5; i++)
  {
    pinMode(i, OUTPUT);
  }
}


void loop()
{
  int potVal = analogRead(A0);
  if (potVal < 300)
  {
    digitalWrite(4, LOW);
    digitalWrite(3, LOW);
    digitalWrite(2, HIGH);
  }
  if (potVal > 300 && potVal < 600)
  {
    digitalWrite(4, LOW);
    digitalWrite(3, HIGH);
    digitalWrite(2, LOW);
  }
  if (potVal > 600)
```

```
  {
    digitalWrite(4, HIGH);
    digitalWrite(3, LOW);
    digitalWrite(2, LOW);
  }
}
```

🗒️ Notes

The LEDs change colour up and down as you turn the pot; there is some flickering as you turn it.

🌻 Challenge

Can you think of ways to stop the flickering with the code examples?