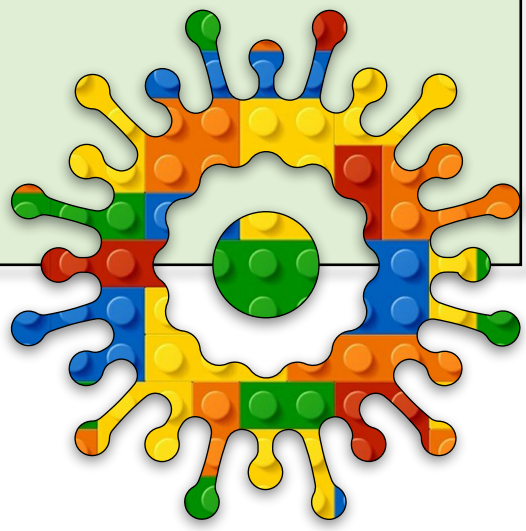
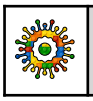
Introducing Robotics

Module A

Unit #7

Serial





Content

Module A Unit #7 serial communication

Introduction to the serial monitor

This is using Serial to connect to your computer and interact directly with the Arduino Uno. There is nothing new needed; we continue with the LEDs.

The serial monitor is accessed by clicking on the button shown below. There is a bar at the top, which is where you can type in commands or data to send to the Arduino (see fig.2). The box below is where the data being sent from the Arduino to your computer appears.

Figure 1: serial monitor

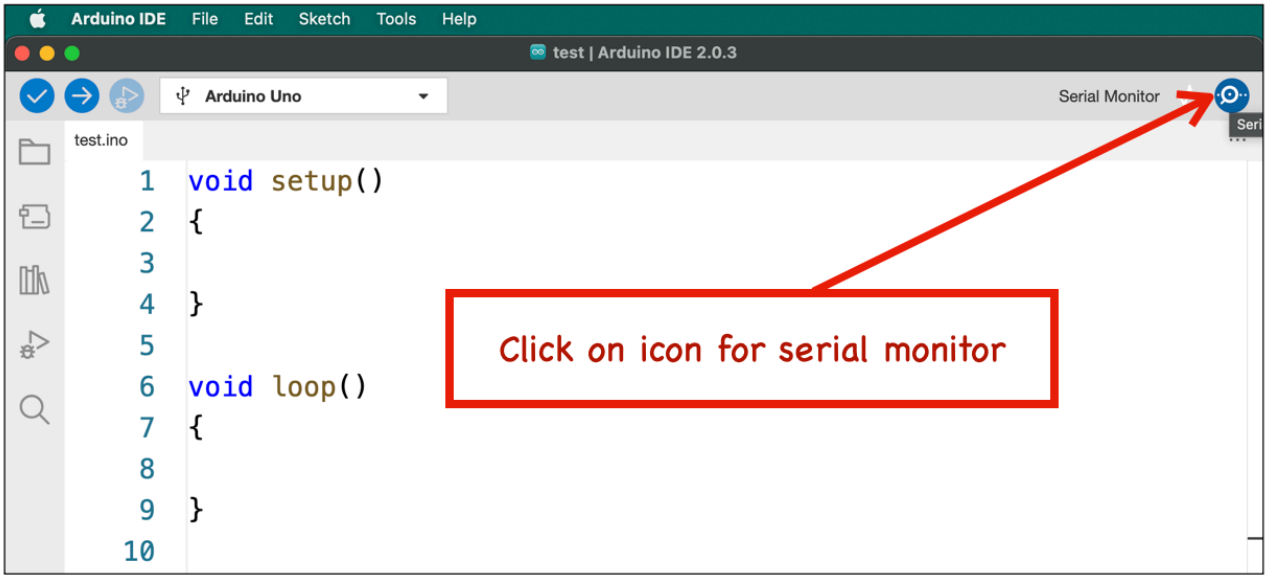
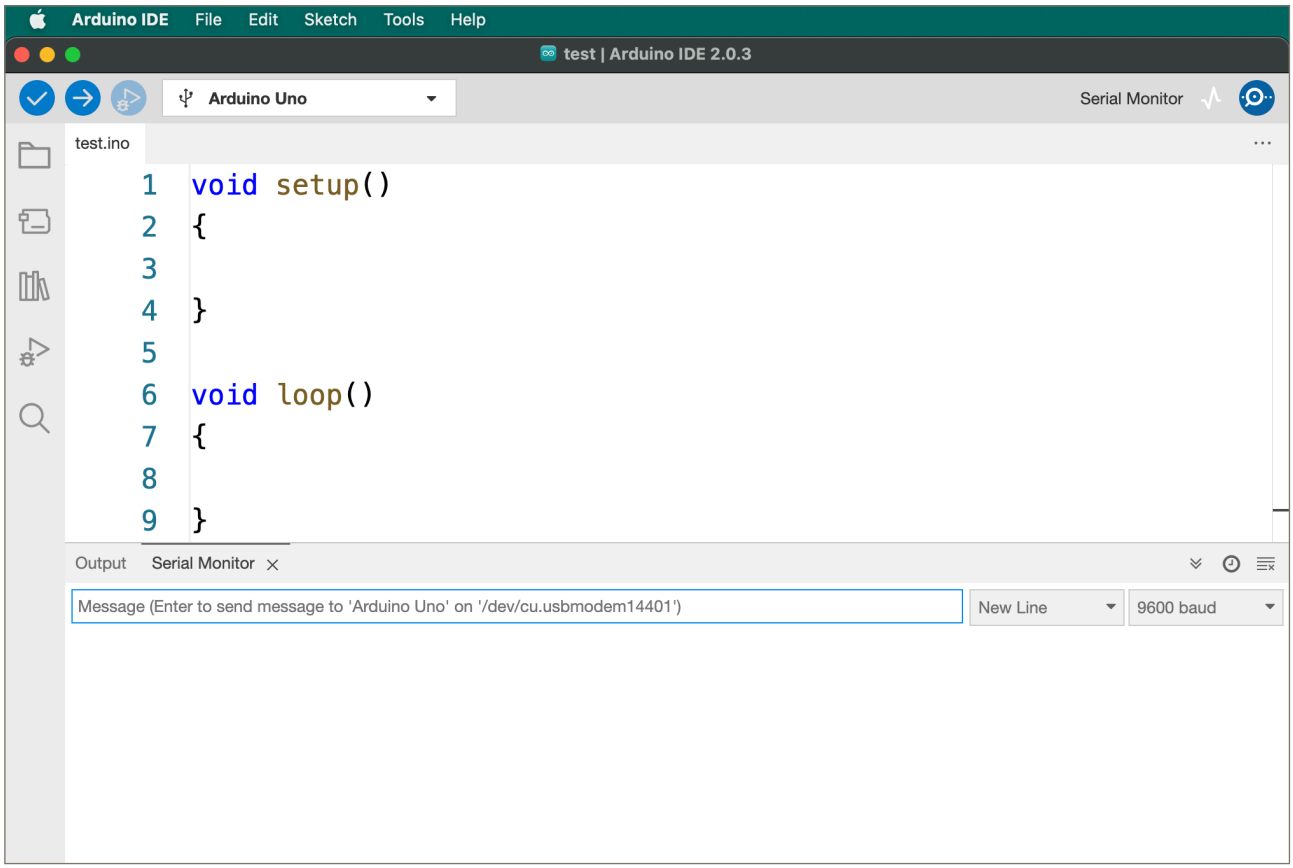


Figure 2: This should appear where the output console was

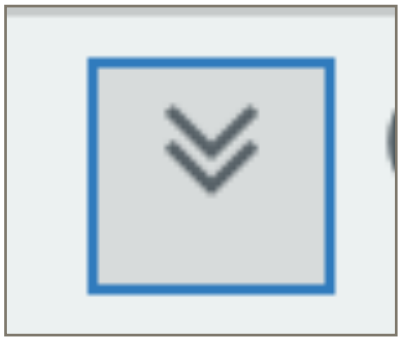


Notes

This is different from the older format, but the principles are the same. An alternative access route is to go to Tools tab and click on the serial monitor tab.

Buttons on the serial monitor

Autoscroll



You can toggle this to stop/start the next line of data appearing.

Timestamp

You can insert a time stamp for each line of data output.

Clear

You can clear the output if you start sending other data.

Sketch A7.1 Hello World!

We can write to the serial monitor. In this first simple example, we will just send the message `Hello World!`. We need to set up communication in the `setup()` function.

The baud rate signifies the data rate in bits per second. The default baud rate in Arduino is `9600` bps (bits per second). So, upload the sketch, then click on the icon (the one that looks like a magnifying glass) near the top right-hand corner.

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

void loop()
{
  Serial.print("Hello World!");
}
```

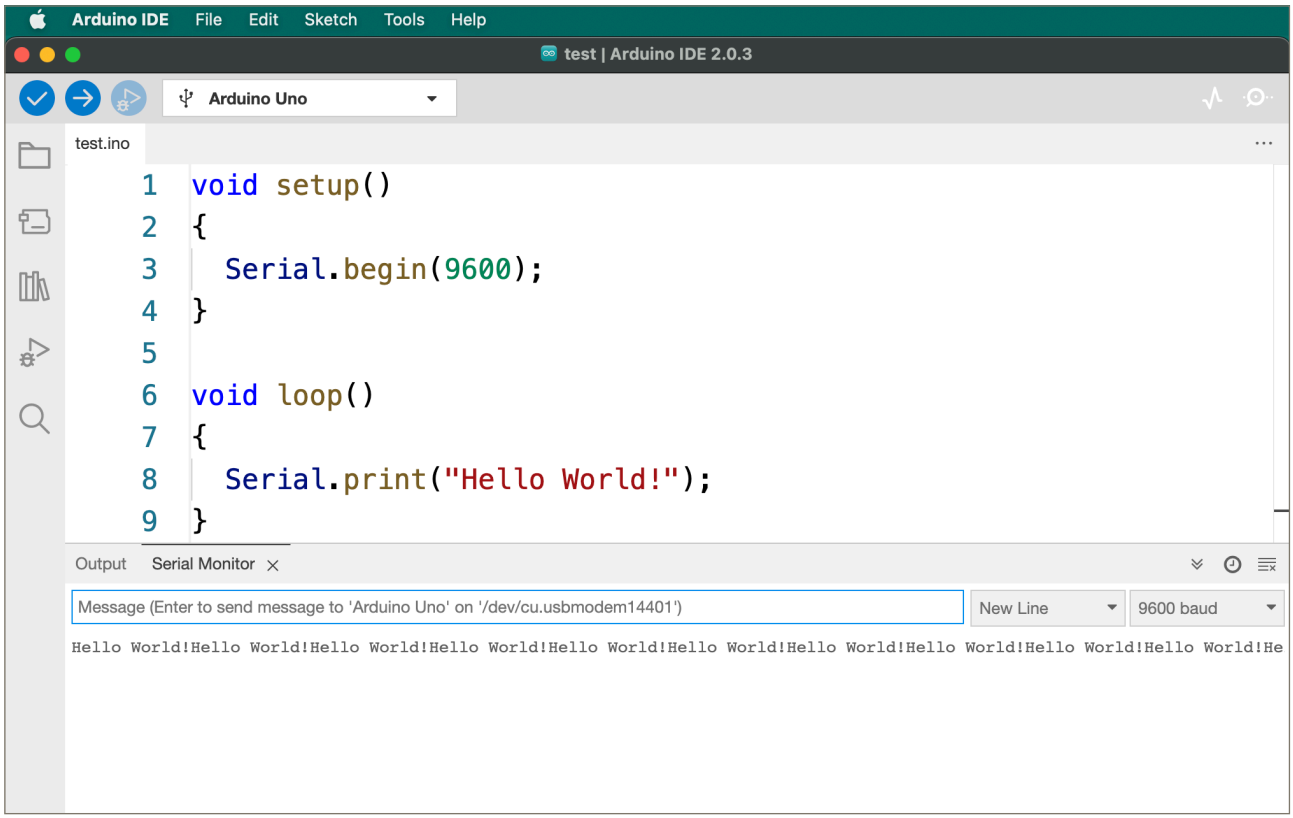
🗒️ Notes

The problem is that it scrolls it across the window; we can get it to scroll downwards in the next sketch.

🛠️ Code Explanation

`Serial.begin(9600);`	Sets up the serial communication, the 9600 is the rate it communicates
`Serial.print();`	Prints a number or text to the monitor

Figure A7.1: It scrolls across the serial monitor



Sketch A7.2 new line

Scrolling downwards, we use `println()` rather than just `print()`.

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

void loop()
{
  Serial.println("Hello World!");
}
```

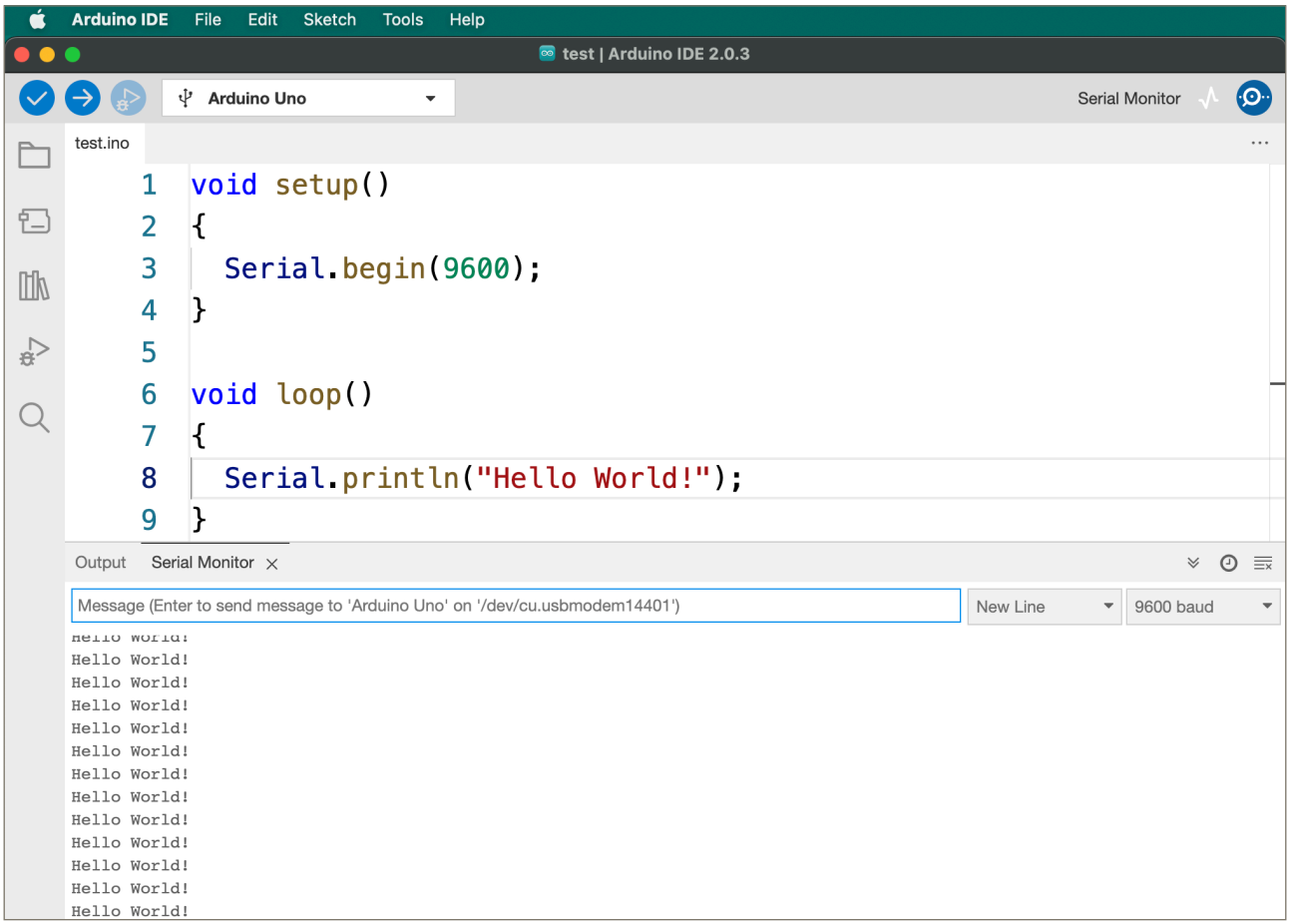
Notes

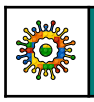
Remember to use double quotes “ ” for text, and if nothing happens, close the window and open it again.

Code Explanation

`Serial.println();`	Prints on a new line if it is repeated on each iteration

Figure A7.2: Much better





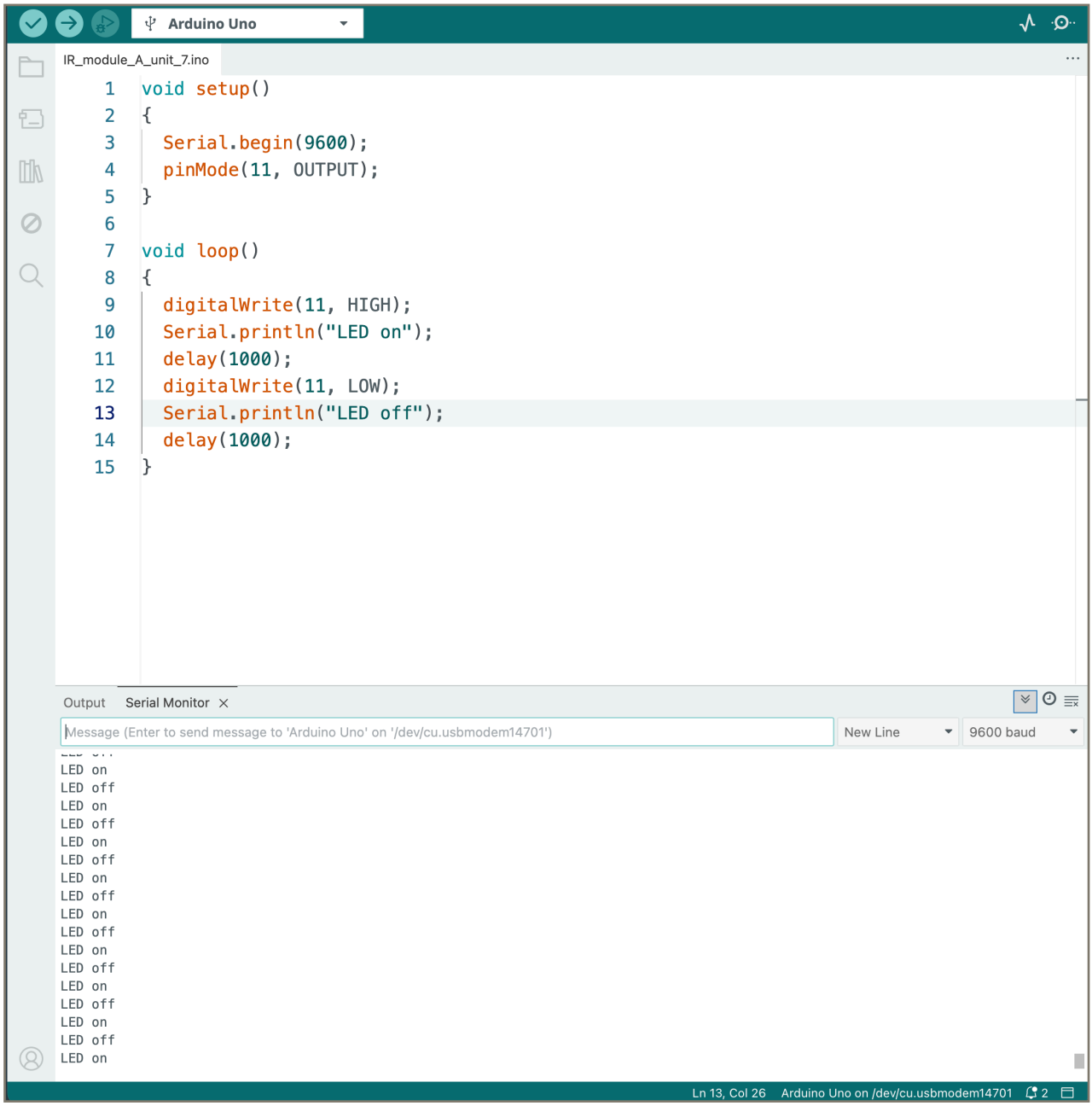
Sketch A7.3 LED on/off

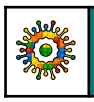
Now, to use the basic blink sketch to tell us when the LED is on, then off.

```
void setup()
{
  Serial.begin(9600);
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  Serial.println("LED on");
  delay(1000);
  digitalWrite(11, LOW);
  Serial.println("LED off");
  delay(1000);
}
```

Figure A7.3: LED in sync with serial monitor





Sketch A7.4 return Hello World

We are going to type `Hello World!` into the message box at the top, send it to the Arduino (press the return button on the keyboard). The Arduino will receive it and send it back.

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

void loop()
{
  if (Serial.available() > 0)
  {
    char input = Serial.read();
    Serial.print(input);
  }
}
```

🛠️ Code Explanation

`Serial.available()`	Checks to see if there is anything
`char input = Serial.read();`	Takes each character one at a time as it reads incoming data

Figure A7.4a: Enter hello world into the top message box

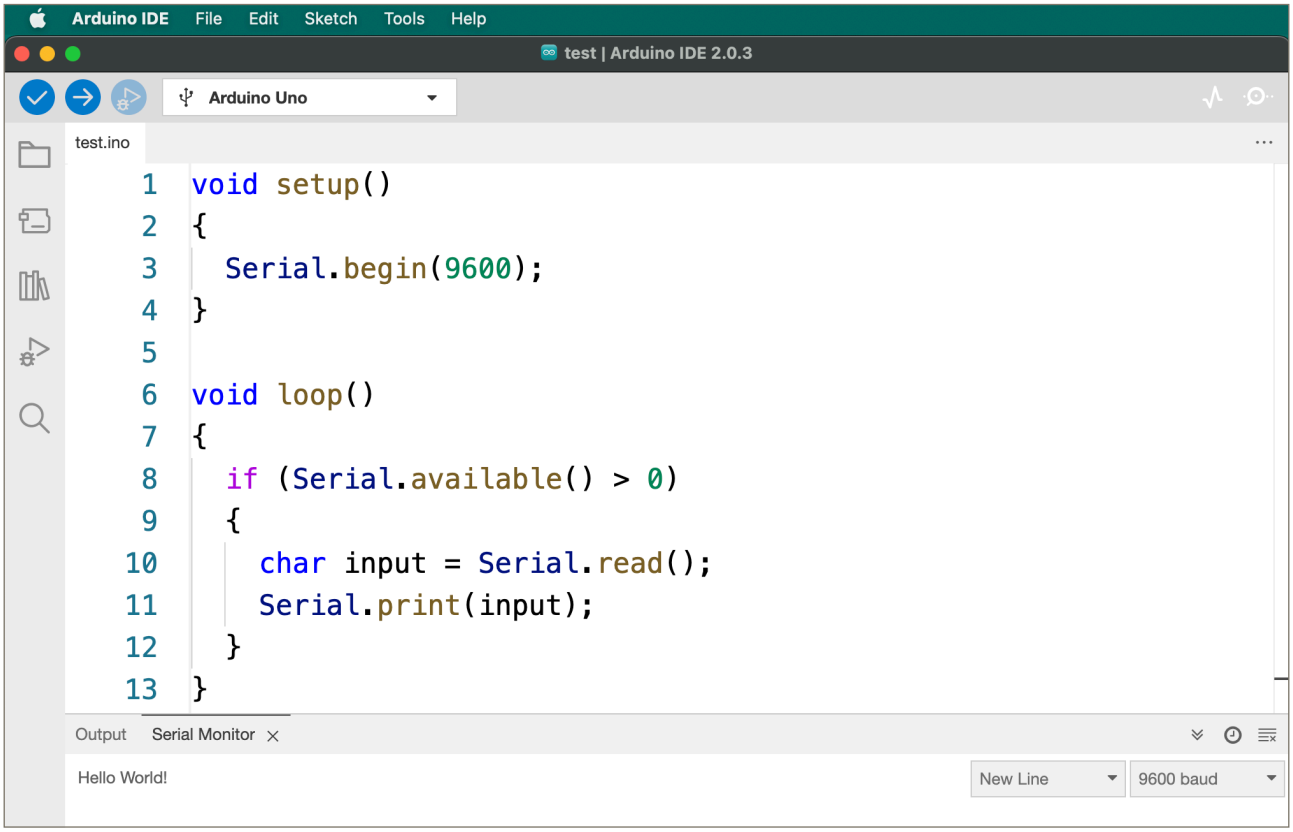
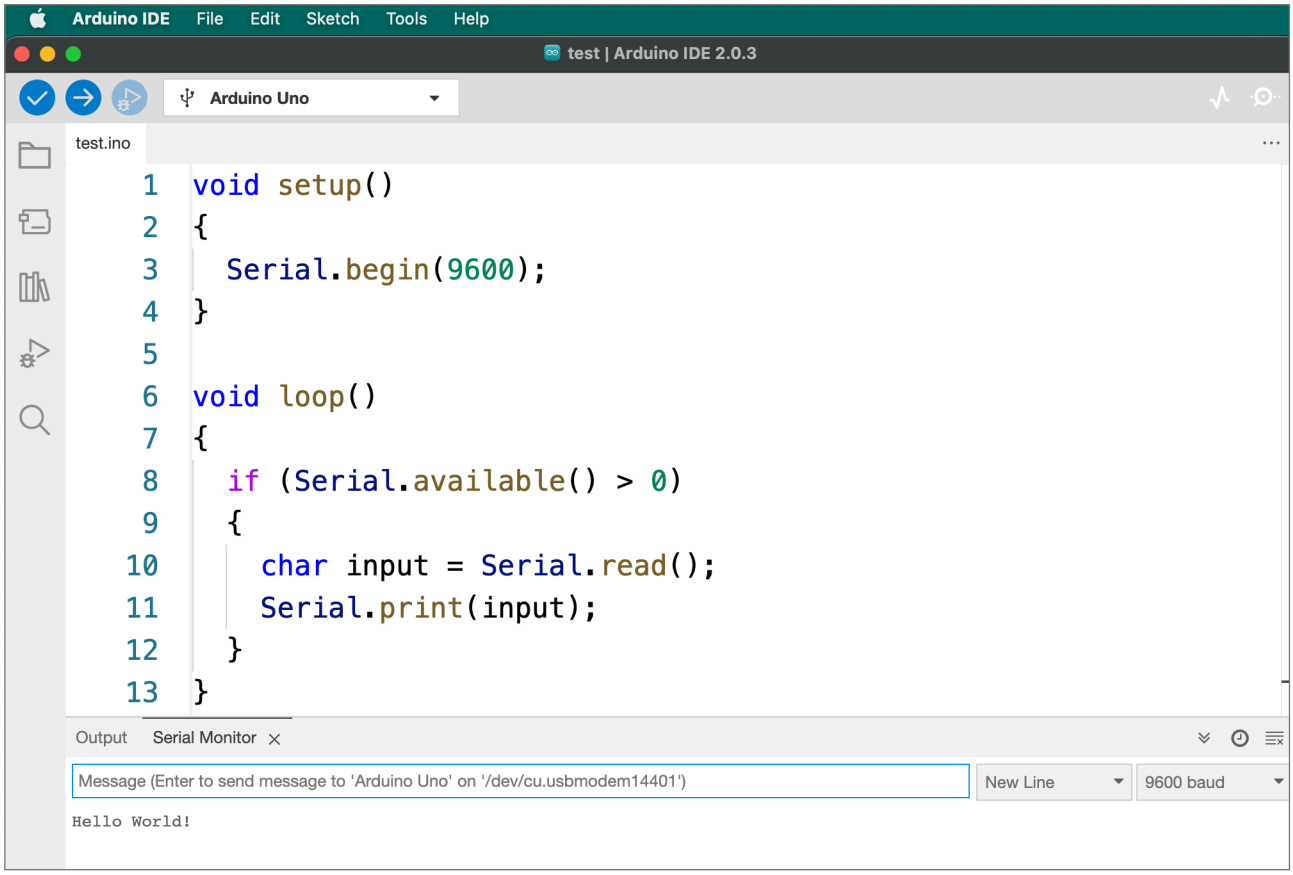
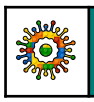


Figure A7.4b: Press return button keyboard





Sketch A7.5 controlling an LED

We are going to use it to turn the LED on and off. Where H is for turn LED on and L is for turn LED off. First, we need to check if there is any data coming in. Once uploaded, click on the serial monitor; if not, open it.

Remove: `Serial.print(input);`

```
void setup()
{
  Serial.begin(9600);
  pinMode(11, OUTPUT);
}

void loop()
{
  if (Serial.available() > 0)
  {
    char input = Serial.read();
    if (input == 'H')
    {
      digitalWrite(11, HIGH);
      Serial.println("LED on");
    }
    if (input == 'L')
    {
      digitalWrite(11, LOW);
      Serial.println("LED off");
    }
  }
}
```

🗒️ Notes

You will need to use capitals for it to work, exactly as the code says. So use `L` and `H`, not `l` or `h`. Also, you can just press return after typing the letter. The LED should be on or off depending on whether you type `H` or `L`.

🌻 Challenge

How would you change the code to accommodate uppercase and lowercase? Hint: use the `||` (the OR logic).

Figure A7.5a: Type capital H in to the text box and press return

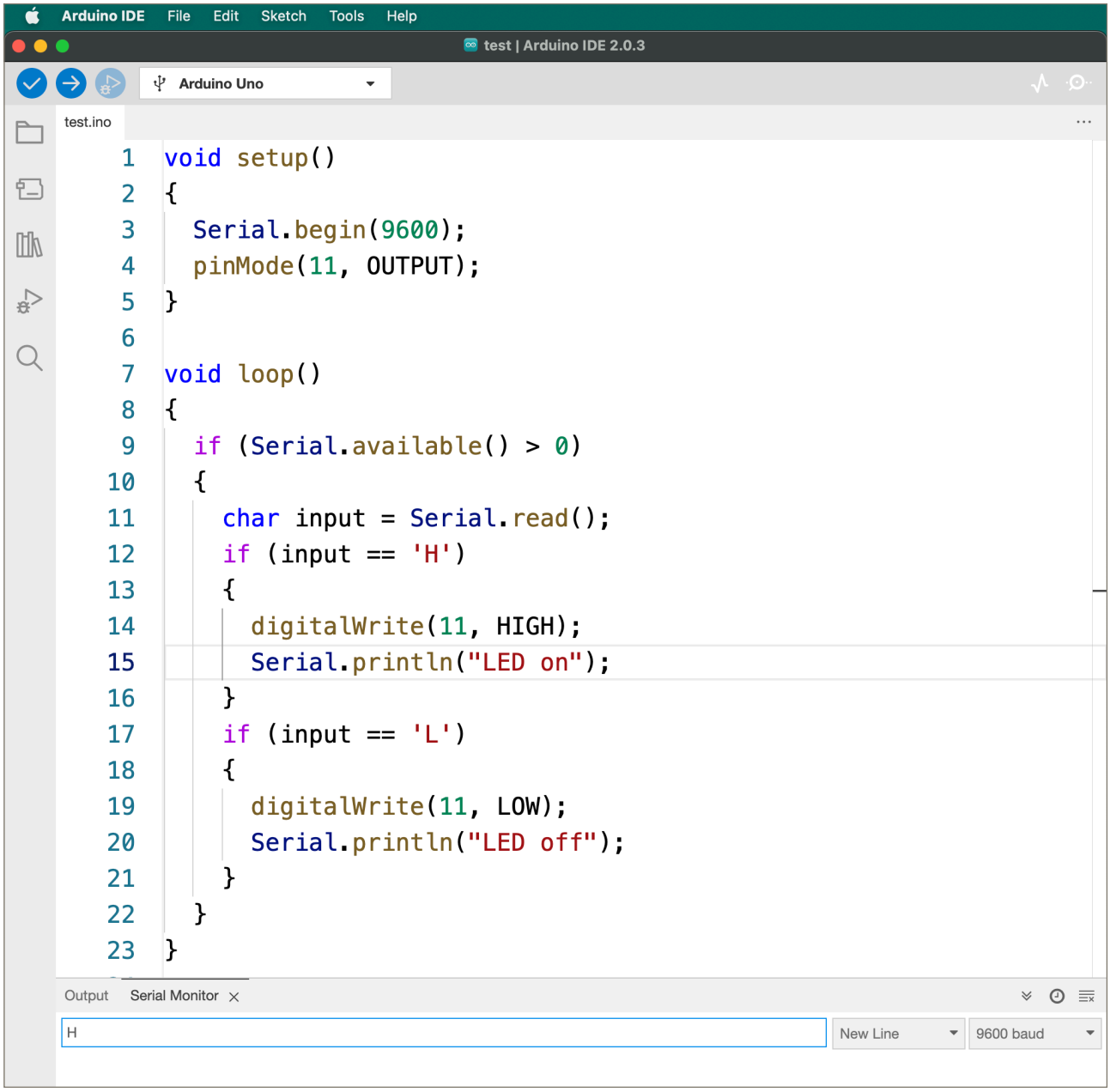
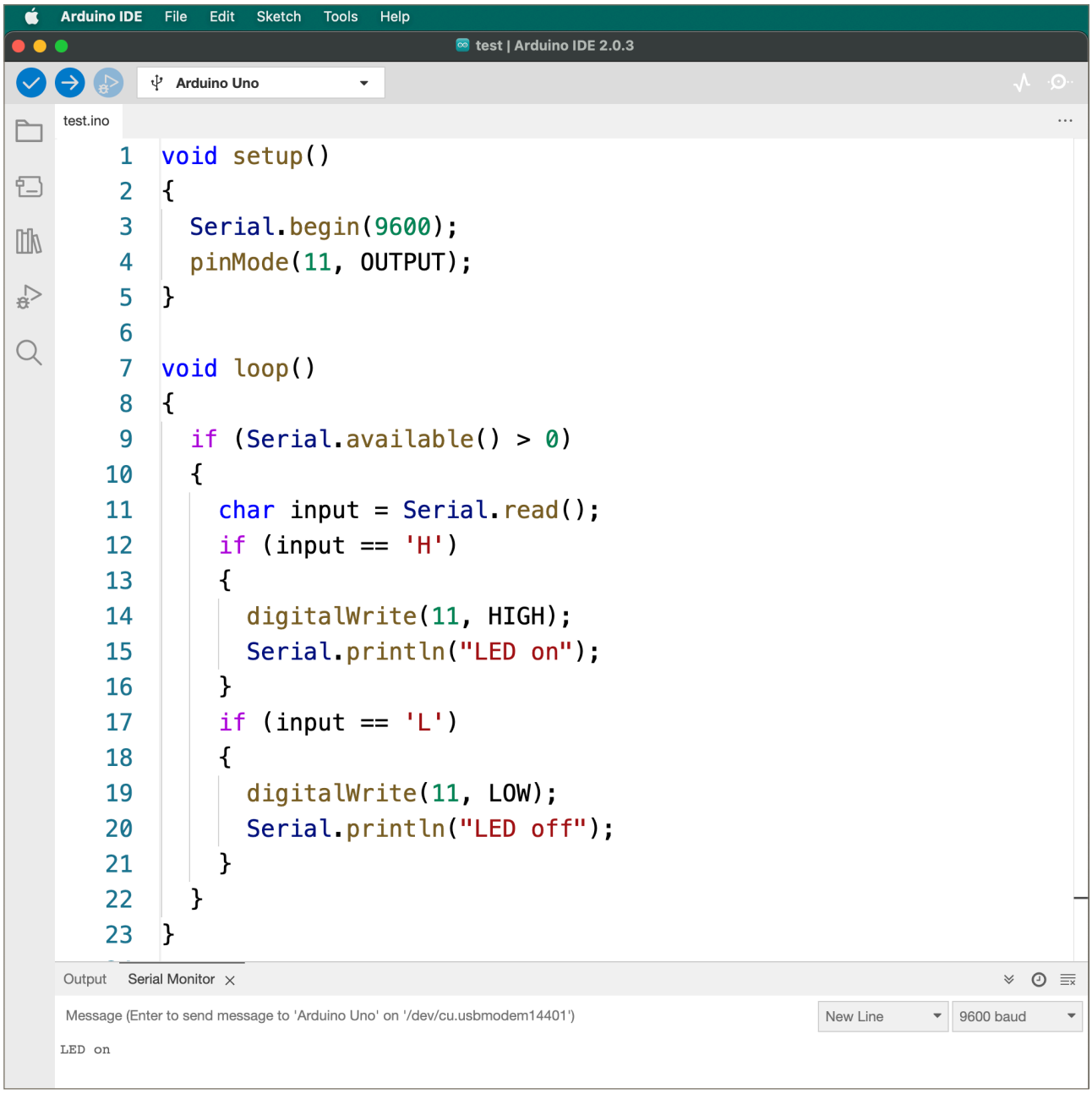


Figure A7.5b: The LED should come on and you will get a message telling you so in the serial monitor



Sketch A7.6 brightness

Instead of inputting on or off (H or L), we want to input the value of the brightness. We use the `parseInt()` function.

❗ Remove everything in the `if()` statement.

To test that it works, send in `255`, then send `0`, and then send `255` again, followed by trying `100`. You can therefore switch it on and off with the values and give it an incremental value also.

```
void setup()
{
  Serial.begin(9600);
  pinMode(11, OUTPUT);
}

void loop()
{
  if (Serial.available() > 0)
  {
    int input = Serial.parseInt();
    analogWrite(11, input);
  }
}
```

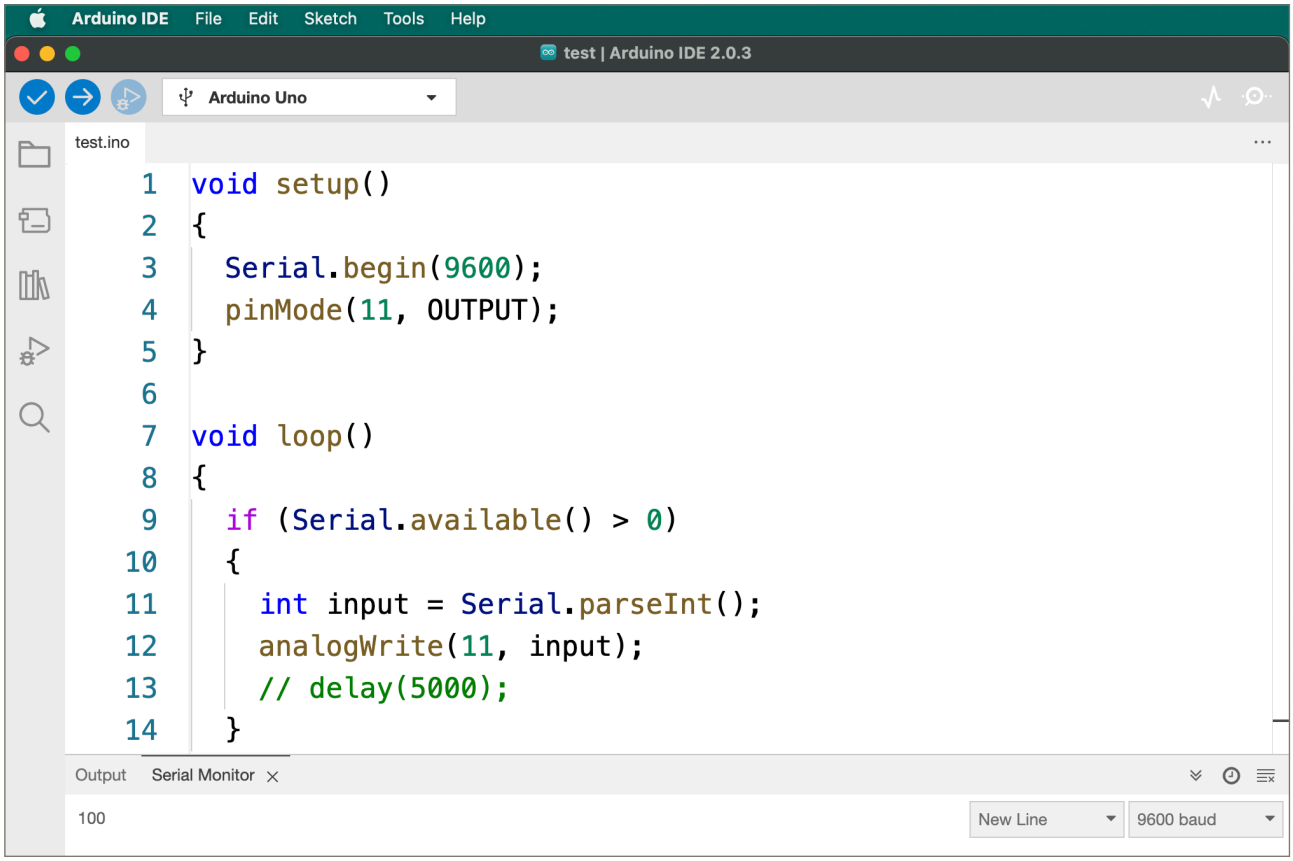
🗒️ Notes

We used lowercase for the input because it is the name of the variable. Notice that it only illuminates briefly; I added a `delay(5000)` to make it last longer.

🛠️ Code Explanation

`int input = Serial.parseInt();`	The input is an integer not a letter (character) hence parseInt() function

Figure A7.6: Type in a value between 0-255 and press send or return



The serial plotter

You have already used the serial monitor, but you have at your disposal a serial plotter. This is where you can plot a graph depending on the inputs it receives. In this initial example, we are going to draw three sine waves.

Figure 3: serial plotter

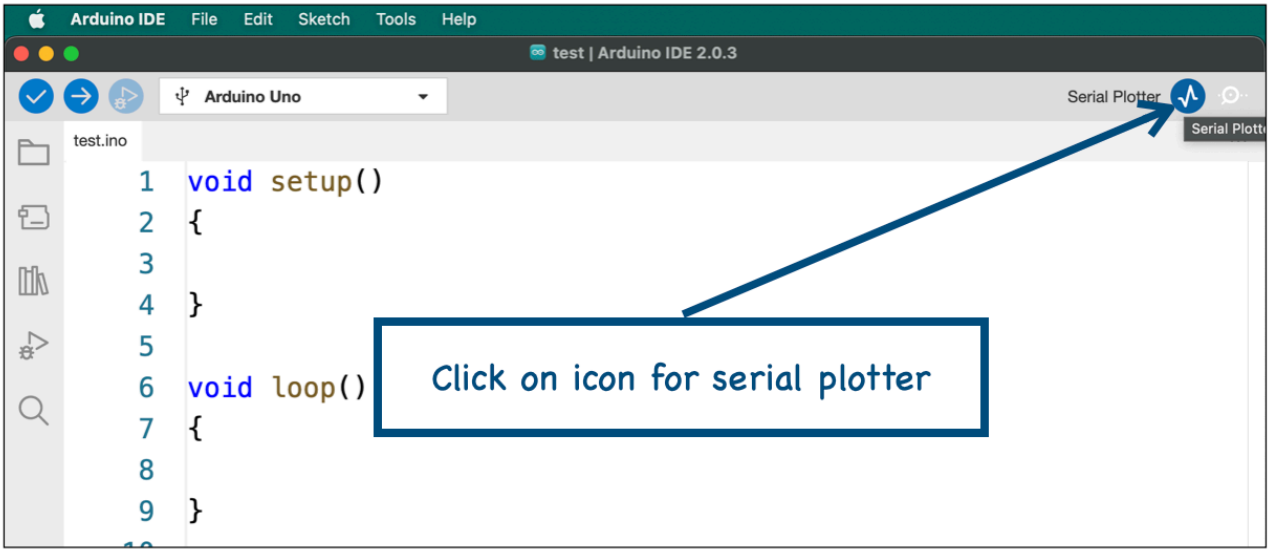


Figure 4: You will get something like this

Sketch A7.7 sine waves

Type in the code below, which will draw a sine wave. After uploading the sketch, click on the serial plotter icon in the top right-hand corner next to the serial monitor icon. You can resize the plotter window by dragging the right edge and the bottom edge.

In Arduino code (C/C++), there are many, many maths functions. Too many to go into just now, but one of them is the constant π (or PI), which is 3.14…. Instead, we can use a handy constant for this, M_PI.

```
float y = 0;

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

void loop()
{
  for (int i = 0; i < 360; i += 10)
  {
    y = 1 * sin(i * M_PI / 180);
    Serial.println(y);
    delay(100);
  }
}
```

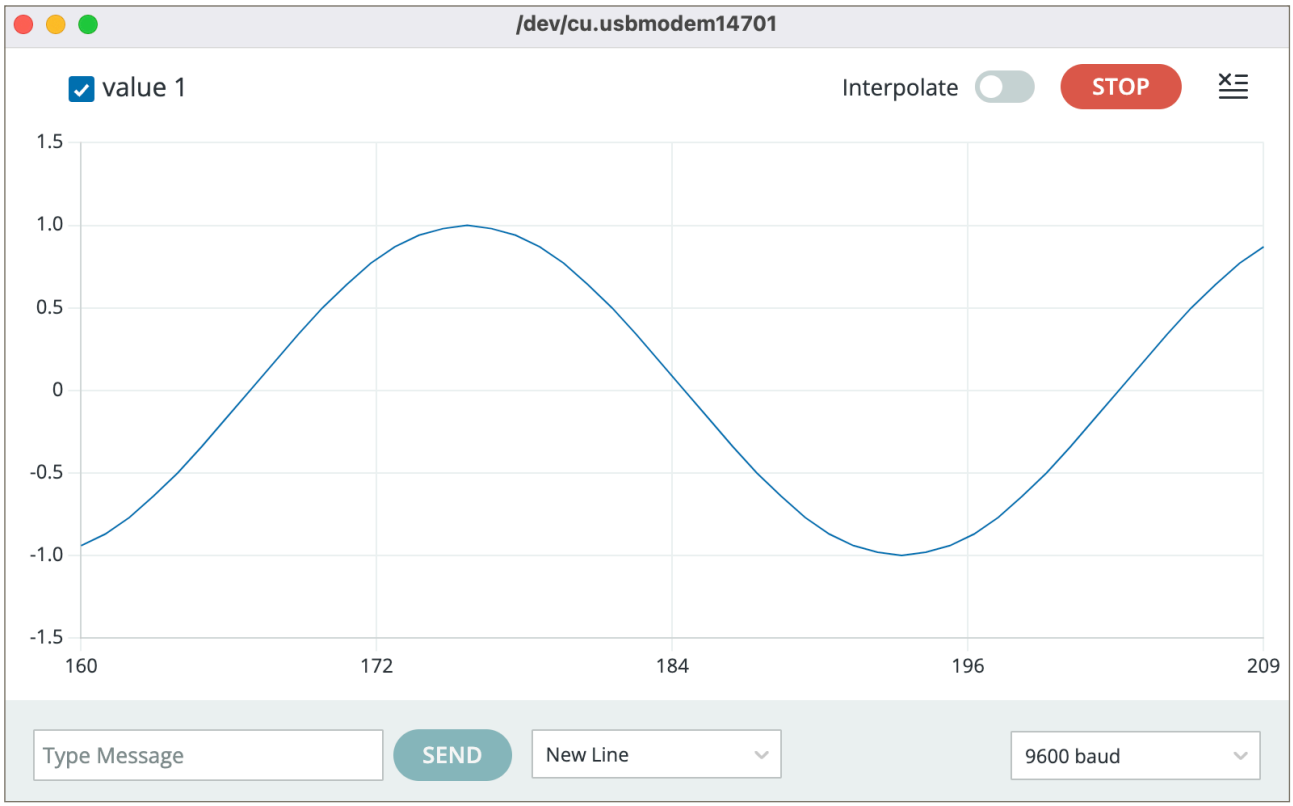
Notes

This was just for illustration purposes; we will make use of the serial plotter when we have data coming from a component such as a variable resistor (pot for short).

🛠️ Code Explanation

`float y = 0;`	The data type is a float which means it can handle numbers after the decimal point, for instance 3.14
`y = 1 *sin(i * M_PI / 180);`	The equation to draw the sine wave, we want it in radians not degrees

Figure A7.7


/dev/cu.usbmodem14701
value 1
Interpolate
STOP
1.5
1.0
0.5
0
-0.5
-1.0
-1.5
160
172
184
196
209
Type Message
SEND
New Line
9600 baud