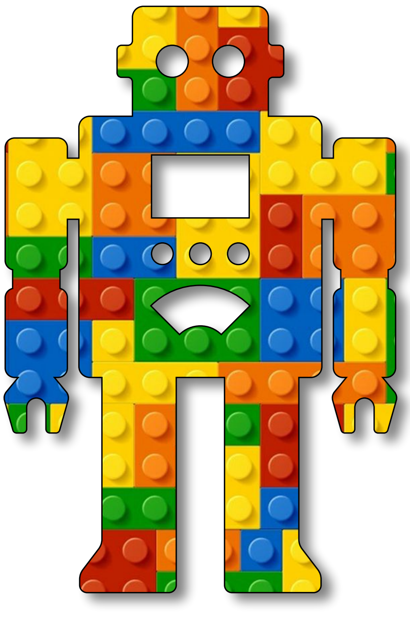
Intelligent Machines Module A Unit #7 serial communication



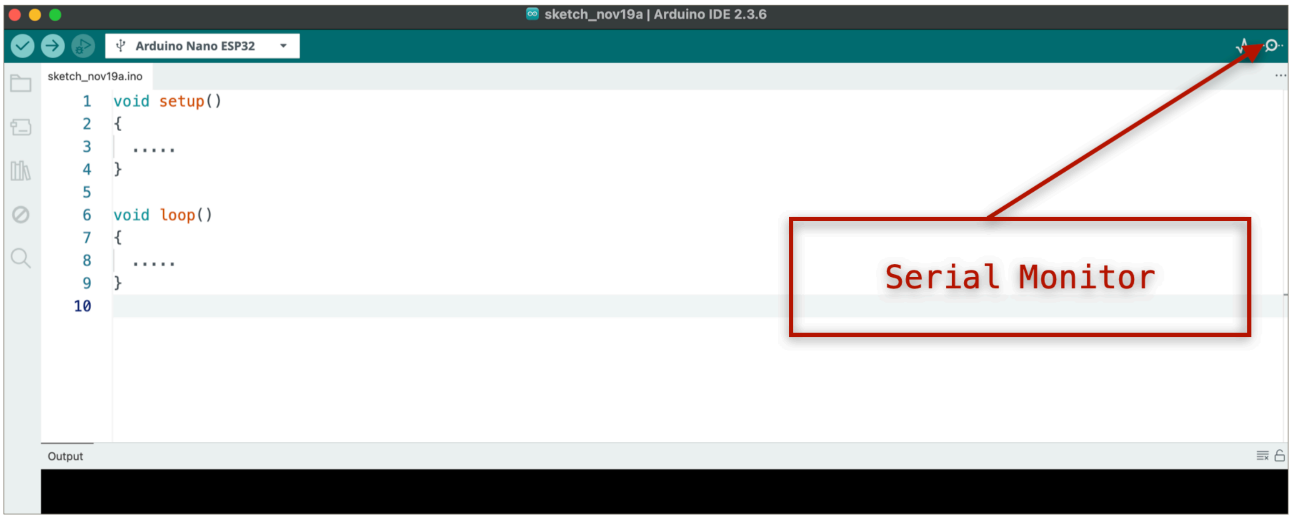
Content

Module A Unit #7 Serial Communication

Introduction to serial communication

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. The box below is where the data being sent from the Arduino to your computer appears.

Figure 1: serial monitor

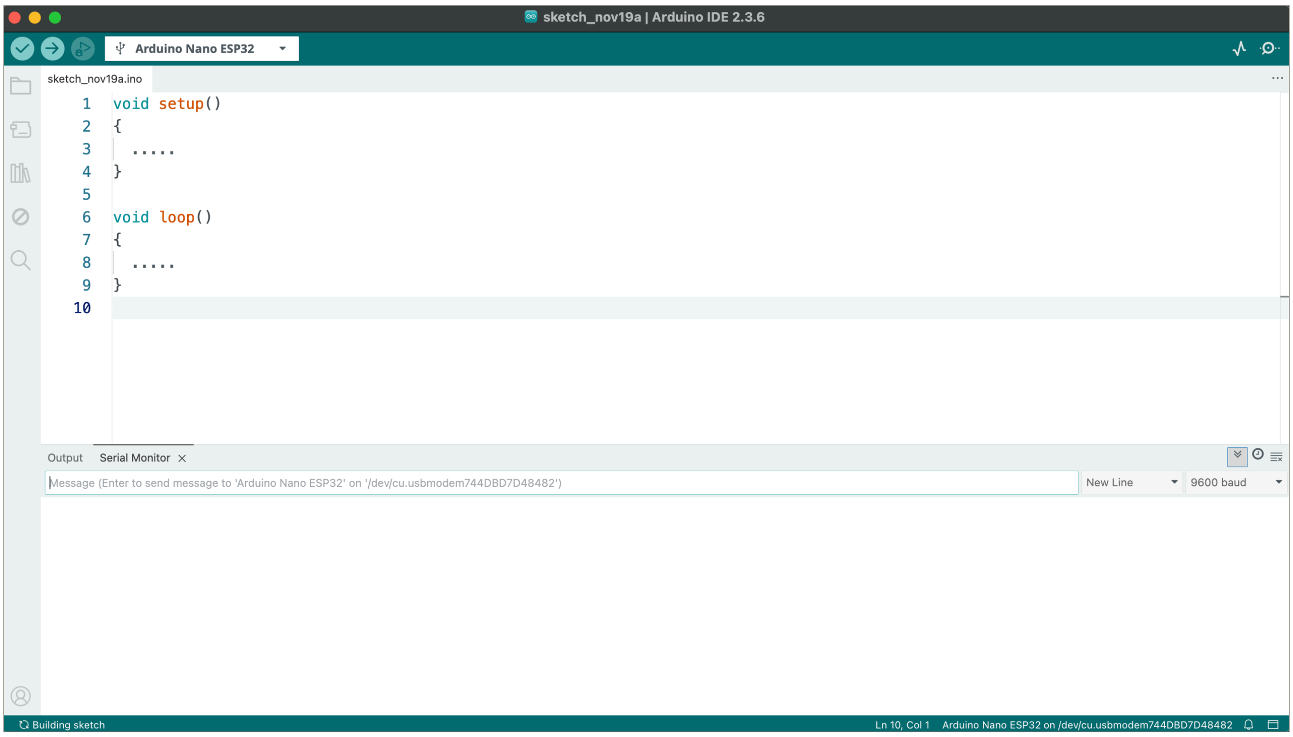


```
void setup()
{
  .....
}

void loop()
{
  .....
}
```

You should get a window pop-up like this...

Figure 2: serial monitor revealed



```
void setup()
{
  .....
}

void loop()
{
  .....
}

```

Message (Enter to send message to 'Arduino Nano ESP32' on '/dev/cu.usbmodem744DBD7D48482')

New Line   9600 baud

Building sketch   Ln 10, Col 1   Arduino Nano ESP32 on /dev/cu.usbmodem744DBD7D48482

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 void 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.

Arduino sketch

```
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. You might need to cancel the serial monitor (click on the x) and re-select it.

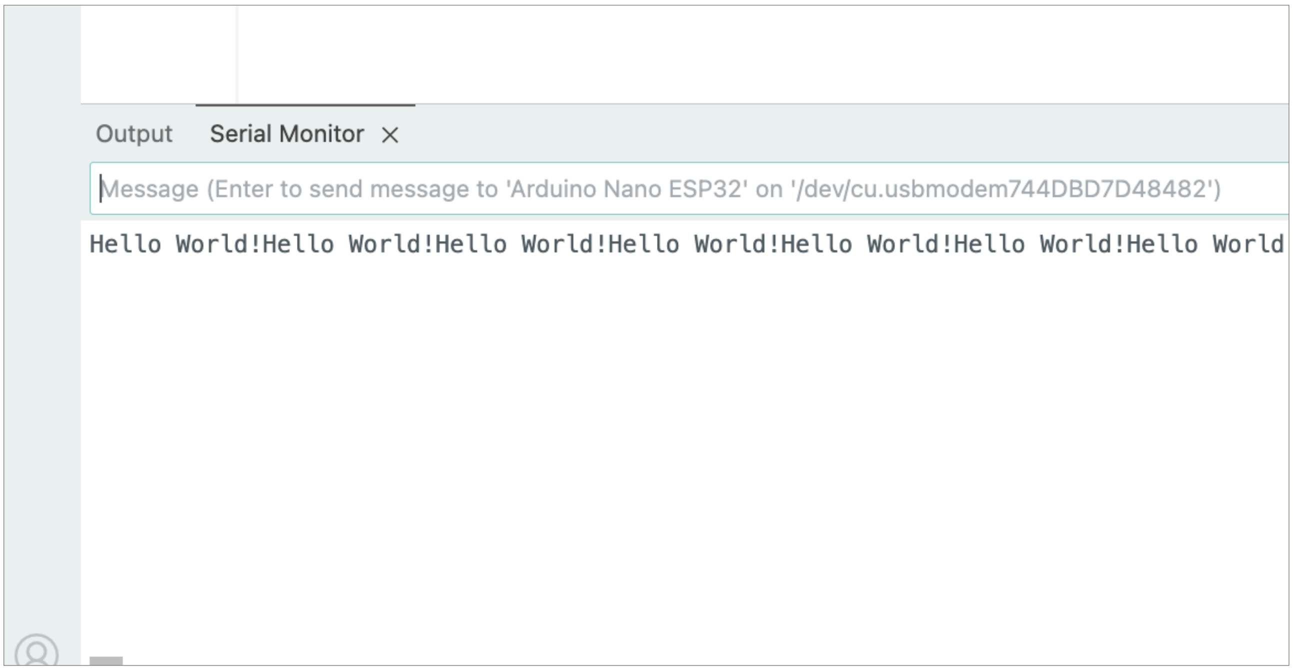
🌻 Challenge

Try some other word or phrase.

🛠️ 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



Message (Enter to send message to 'Arduino Nano ESP32' on '/dev/cu.usbmodem744DBD7D48482')

```
Hello World!Hello World!Hello World!Hello World!Hello World!Hello World!Hello World
```

Sketch A7.2 new line

Now scrolling downwards.

Arduino sketch

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


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

Notes

Remember to use double quotes "...." for text.

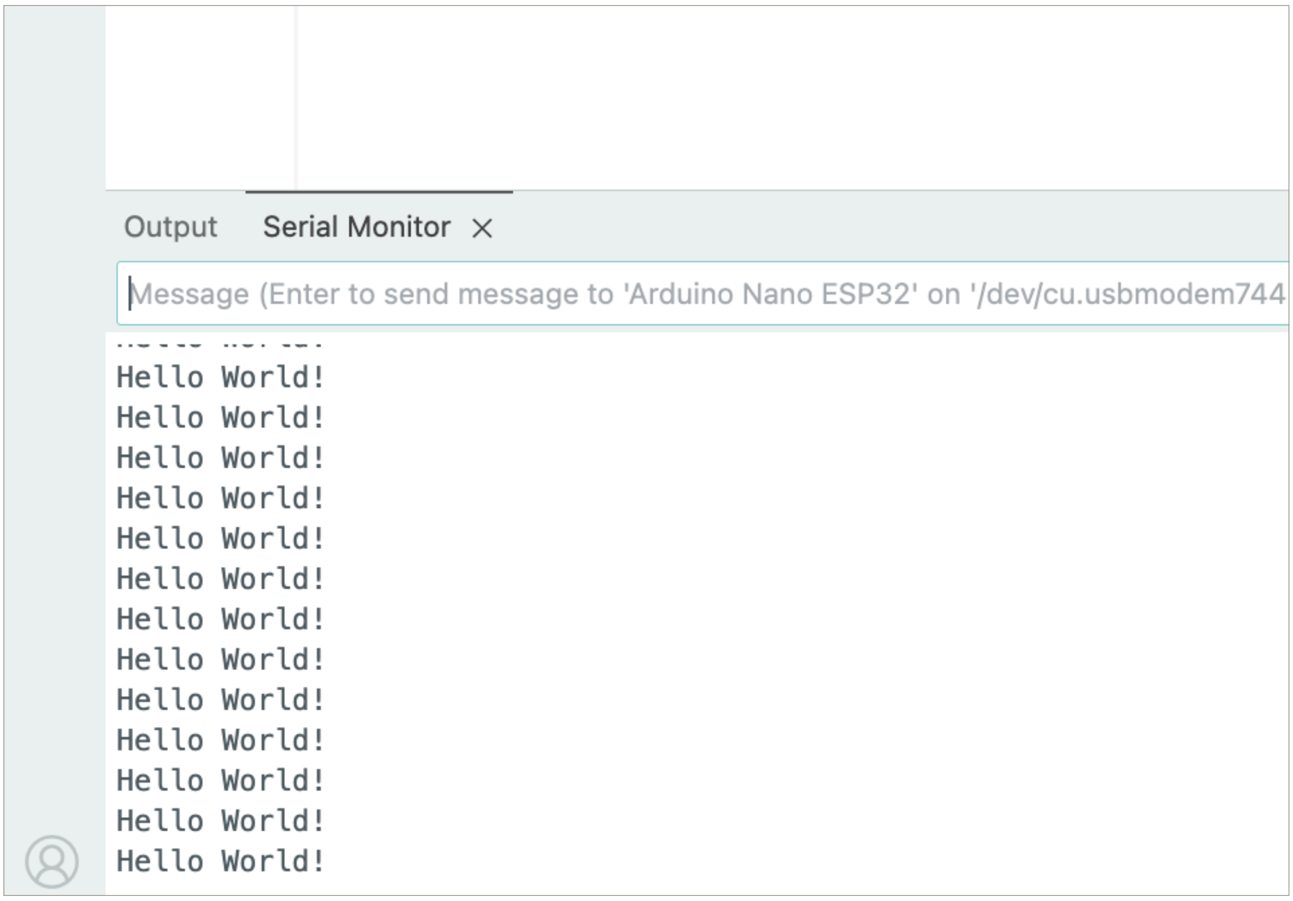
Challenge

See what happens if you remove the speech marks.

Code Explanation

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

Figure A7.2


Output
Serial Monitor
Message (Enter to send message to 'Arduino Nano ESP32' on '/dev/cu.usbmodem744

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

Sketch A7.3 LED on/off

❗ A newish sketch

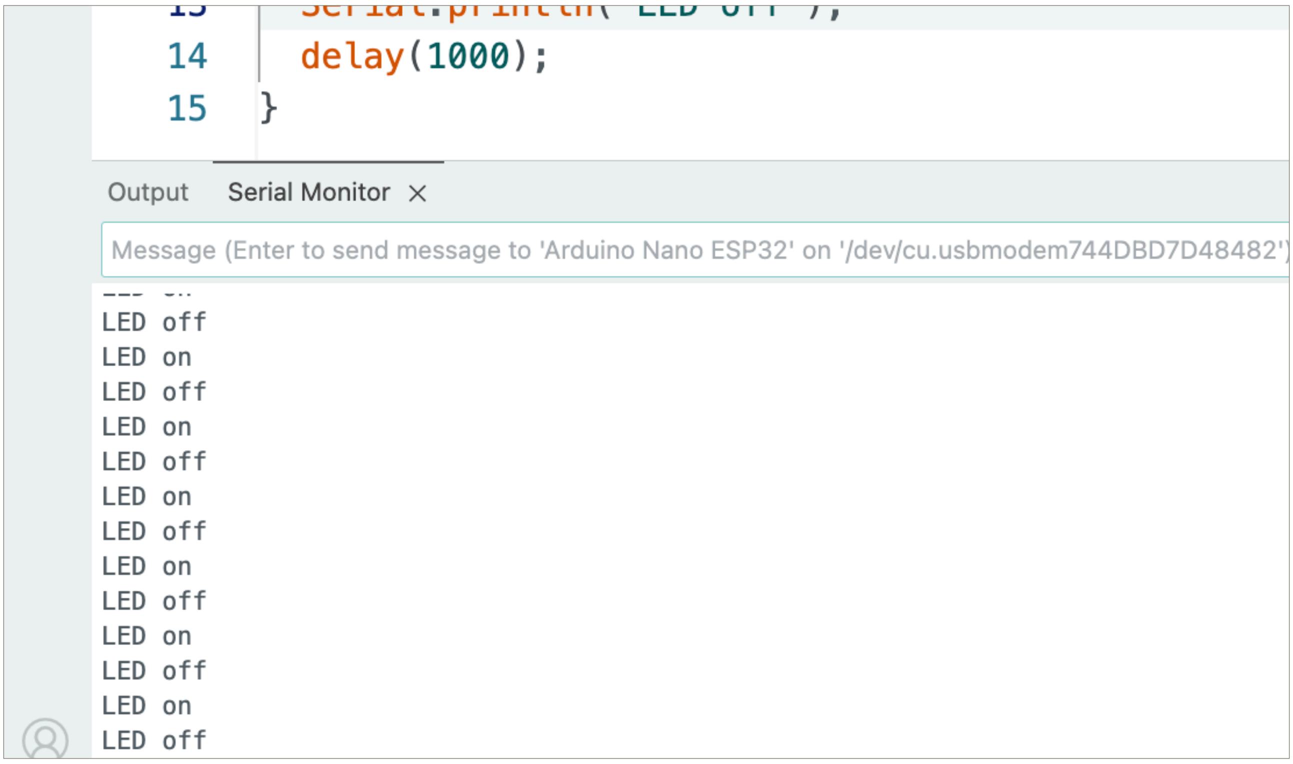
Now, to use the basic `blink` sketch to tell us when the LED is `on` and then `off`.

Arduino sketch

```
void setup()
{
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  Serial.println("LED on");
  delay(1000);
  digitalWrite(13, LOW);
  Serial.println("LED off");
  delay(1000);
}
```

Figure A7.3



```
  delay(1000);
}
```

Output Serial Monitor

Message (Enter to send message to 'Arduino Nano ESP32' on '/dev/cu.usbmodem744DBD7D48482')

```
LED off
LED on
LED off
LED on
LED off
LED on
LED off
LED on
LED off
LED on
LED off
LED on
LED off
```

Sketch A7.4 return Hello World

❗ A new sketch

We are going to type Hello World! into the message box, send it to the Arduino (by pressing return). The Arduino will receive it and send it back in the serial monitor.

Arduino sketch

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

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

Notes

This may seem like a pointless exercise, but I want to introduce the concept of sending information (text or otherwise) via the serial connection to affect change.

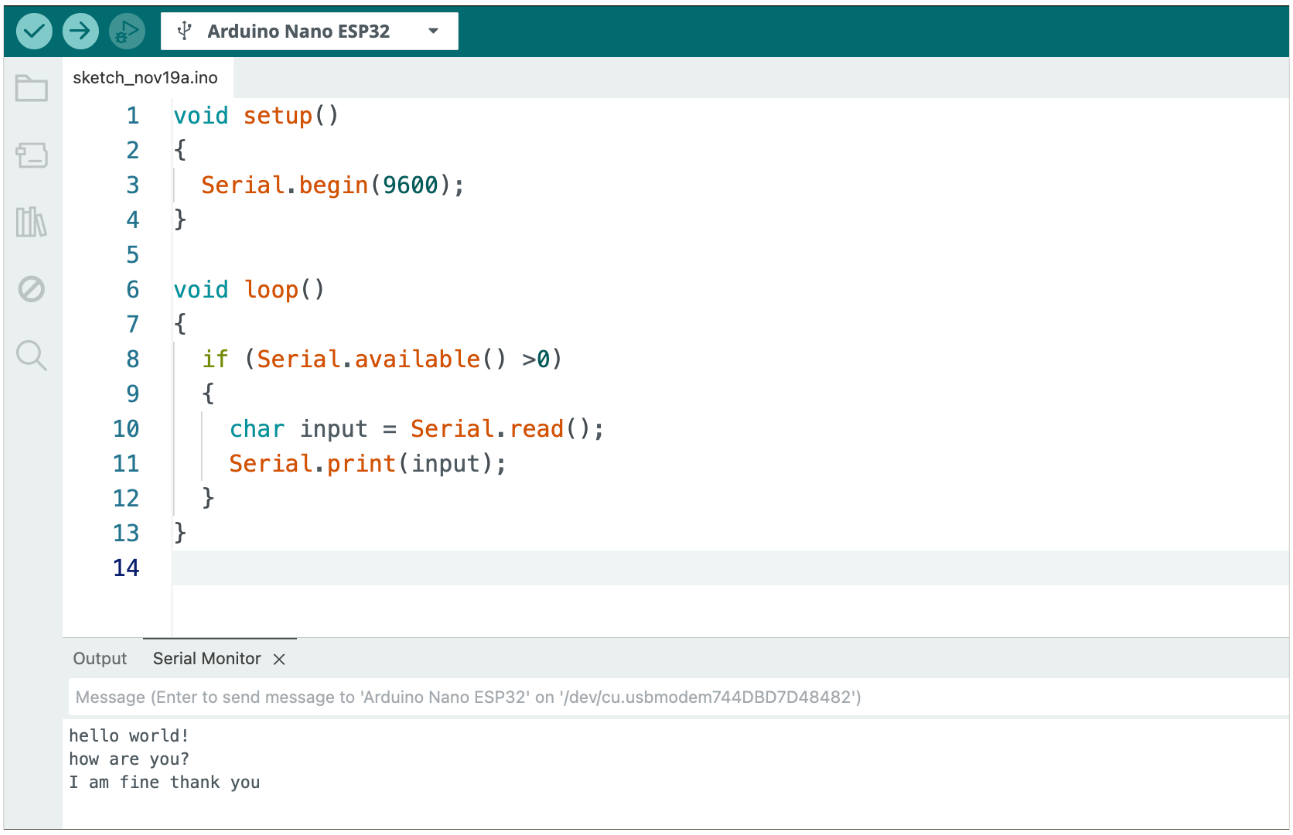
Challenge

Type in your own message.

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 A7.4


Arduino Nano ESP32
sketch_nov19a.ino

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

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

Output
Serial Monitor
Message (Enter to send message to 'Arduino Nano ESP32' on '/dev/cu.usbmodem744DBD7D48482')

```
hello world!
how are you?
I am fine thank you
```

Sketch A7.5 controlling an LED

We are going to use this feature to turn the LED on and off. So that when we type H, it turns the LED on, and L to turn the LED off. As before, we need to check if there is any data coming in.

Arduino sketch

```
void setup()
{
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}

void loop()
{
  if (Serial.available() > 0)
  {
    char input = Serial.read();
    if (input == 'H')
    {
      digitalWrite(13, HIGH);
      Serial.println("LED on");
    }
    if (input == 'L')
    {
      digitalWrite(13, LOW);
      Serial.println("LED off");
    }
  }
}
```

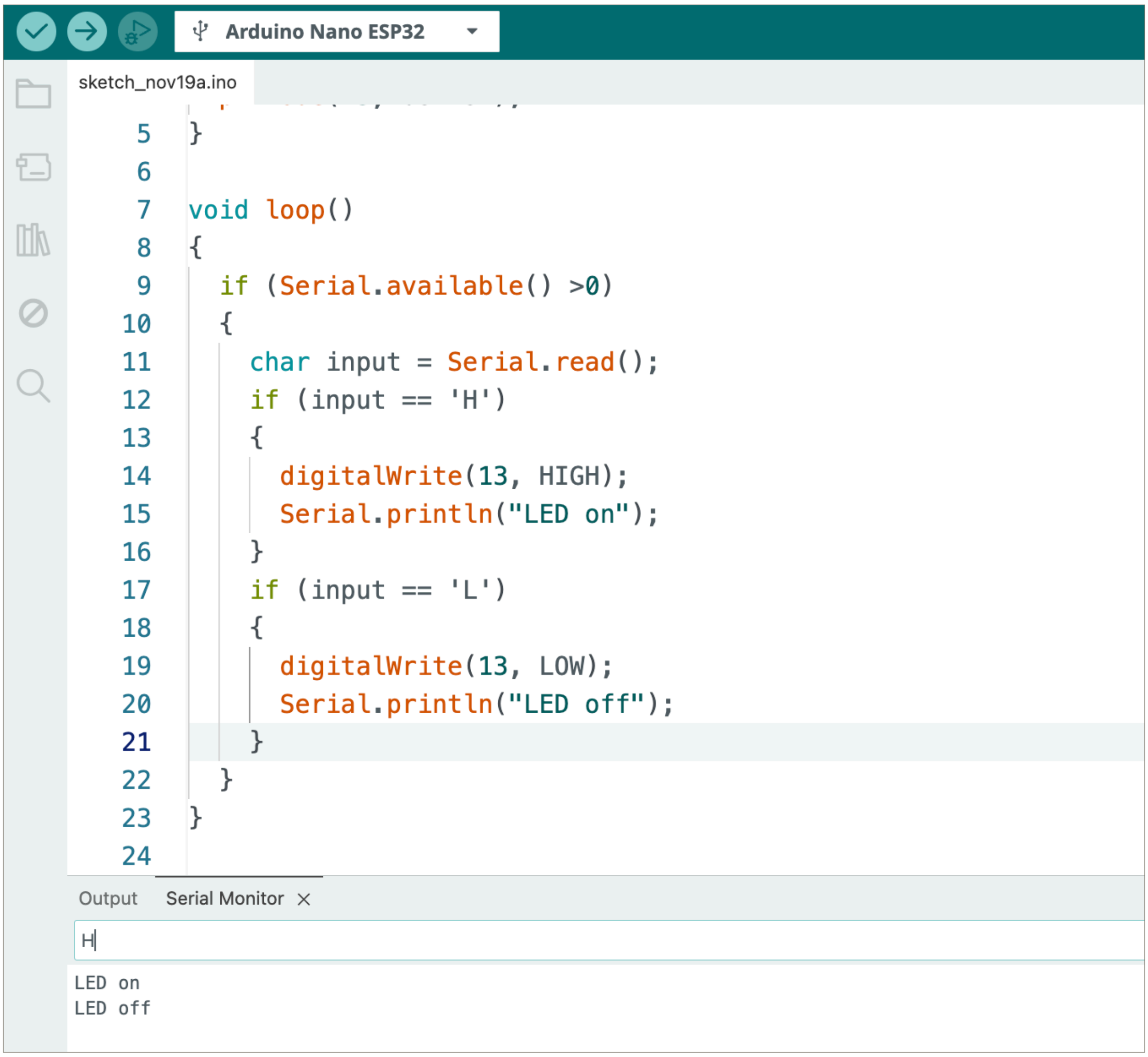
🗒️ Notes

The LED should respond to `H` and `L` being typed in (press return to send).

🌻 Challenge

What happens with lowercase `h` and `l`? How could you mitigate this?

Figure A7.5



```
}

void loop()
{
  if (Serial.available() >0)
  {
    char input = Serial.read();
    if (input == 'H')
    {
      digitalWrite(13, HIGH);
      Serial.println("LED on");
    }
    if (input == 'L')
    {
      digitalWrite(13, LOW);
      Serial.println("LED off");
    }
  }
}
```

```
H
```

```
LED on
LED off
```

Sketch A7.6 levels of brightness

Instead of switching the LED on and off, we can give the brightness a value. To input the value of the brightness, we use the parseInt() function. To test that it works, send in 255, (press return), then send 1, (press return), and then send 255 again followed by 100. You can therefore switch it between these values to control the brightness.

Arduino sketch

```
void setup()
{
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}

void loop()
{
  if (Serial.available() > 0)
  {
    int input = Serial.parseInt();
    if (input != 0)
    {
      analogWrite(13, input);
    }
  }
}
```

Notes

This, again, is a simple illustration. The reason for having the line of code: if(input != 0) . . . is because the Serial Monitor sends a 0 value on a continuous loop, and this solves that problem; otherwise, it would light up only briefly.

🌻 Challenge

Can you think of something else you could control?

🛠️ Code Explanation

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