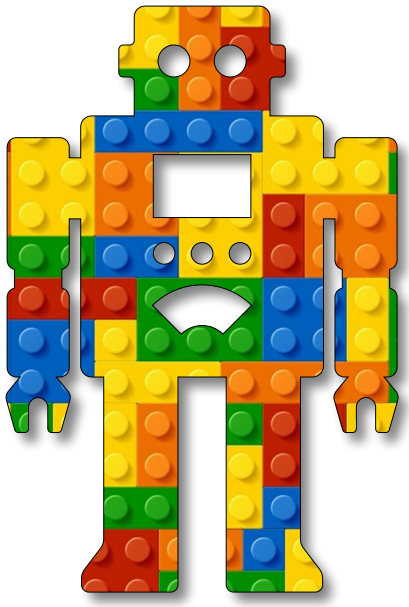
Intelligent Machines Module B Unit #2 LED on/off



Content

Module B Unit #2 p5.js LED

Introduction to LED with p5.js

Previously, we were able to control the LED with code inside the sketch, either to blink, switch it on or off. We could even use the serial port to send messages to switch it on or off or control its brightness.

Here, we are going to take this a step further and use `p5.js` to do just that. In the first unit, we are going to simply switch it on with a click of the mouse on the canvas. This will lead us onto being able to connect our Arduino with AI. The Machine Learning tool (`ml5.js`) will be able to directly engage with it, both sending data to it and also receiving data from its many sensors.

This can get a bit complicated because we are coding in two different places: the `p5.js web editor` and the `Arduino IDE`. To, hopefully, make it clear which is which, I have colour-coded them separately. We will start with the Arduino code. There are no components to add, just the `Arduino Nano 33 BLE` connected, as before, to your computer.

Sketch B2.1 LED

❗ Arduino sketch

Using pin 13 for the LED, waits for the input words either HIGH or LOW to turn the LED on (HIGH) or off (LOW).

Arduino sketch

```
const int ledPin = 13;

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

void loop()
{
  if (Serial.available() > 0)
  {
    String command = Serial.readStringUntil('\n');
    command.trim();
    if (command == "HIGH")
    {
      digitalWrite(ledPin, HIGH);
    }
    else if (command == "LOW")
    {
      digitalWrite(ledPin, LOW);
    }
  }
}
```

Notes

All we need to do now (as far as the Arduino is concerned) is wait for the incoming command.

🛠️ Code Explanation

`const int ledPin = 13;`	LED is connected to pin 13
`Serial.begin(9600);`	Match the baud rate in p5.js
`String command = Serial.readStringUntil('\n');`	Read until newline
`command.trim();`	Remove leading/trailing whitespace

Talking to the USB port

Head over to the p5.js web editor. Here, we will need to do a number of things. The first is getting the web editor to talk to the port. This requires some extra code and software. The great thing is that someone has done the hard work for us, and so we can use a webserial library created for just this purpose.

We simply add it into the index.html file.

```
<script src="https://unpkg.com/p5-webserial@0.1.1/build/p5.webserial.js"></script>
```

One issue is that you have to use Chrome or Edge as the web browser (not Safari).

Because you will be communicating on the same port as the Arduino Nano 33 BLE, you probably can't update the Arduino (upload) if the p5.js web editor is connected to it. If, for some reason, you need to change the code on the Arduino, then you first have to stop the p5.js web browser before proceeding with the Arduino upload, and conversely, you can't have the serial monitor open if you are communicating from the web editor. You'll find out soon enough.

Although we have added a library to take care of the heavy lifting, we do need some code to make the web editor talk to the Arduino and vice versa. To do that, we will create another file called port.js and add that to the index.html file.

The port.js functions

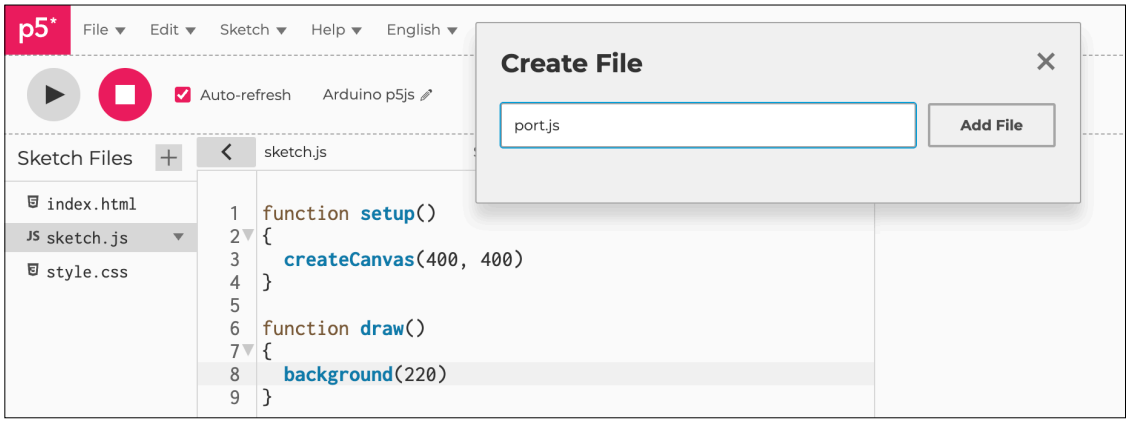
As part of the library, we have to create a number of functions to fulfil a number of actions. They are relatively self-evident. Rather than trying to analyse each line of code, I will summarise each one in turn.

function navigation()
function makePortButton()
function choosePort()
function openPort()
function portError(err)
function serialEvent()
function portConnect()
function portDisconnect()
function closePort()

Creating the port.js file

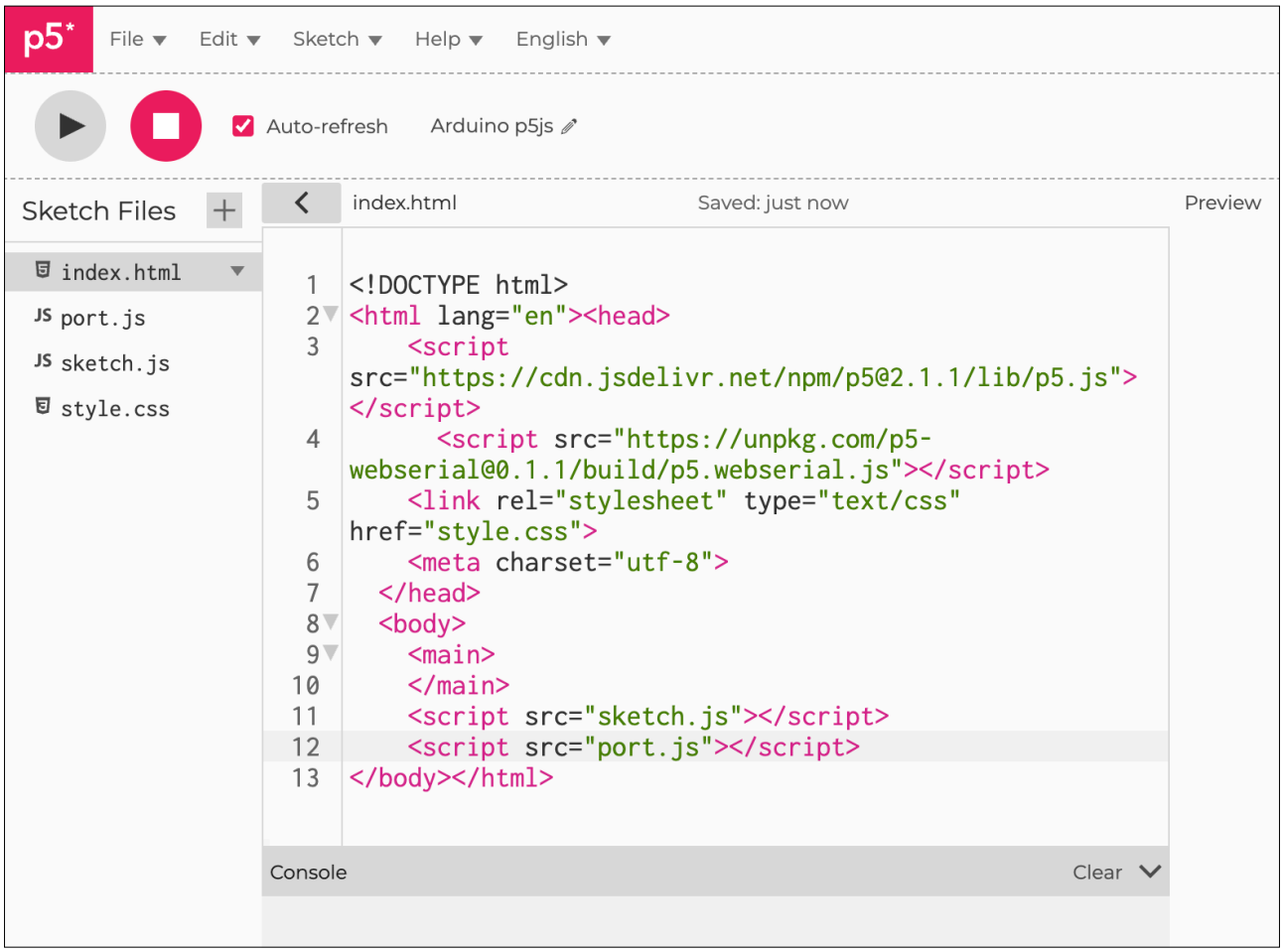
Creating the `port.js` file as we have done for other files previously.

Figure 1: port.js file



Adding the webserial library and the port.js file to the index.html file.

Figure 2: webserial and port.js to index.html



Sketch B2.2 index html

In the index.html file, we need to add the reference to include the webserial. This will allow us to connect the website with the port we want to use when connected to our Arduino.

index.html

```
<!DOCTYPE html>
<html lang="en"><head>
    <script src="https://cdn.jsdelivr.net/npm/p5@2.1.1/lib/p5.js"></script>
      <script src="https://unpkg.com/p5-webserial@0.1.1/build/
p5.webserial.js"></script>
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta charset="utf-8">
  </head>
  <body>
    <main>
    </main>
    <script src="sketch.js"></script>
    <script src="port.js"></script>
</body></html>
```

Notes

It will look like figure 2.

Sketch B2.3 function navigation()

❗ port.js

The main function holding all the others together is the `navigation()` function. The first challenge is to check whether the browser is suitable for this.

port.js

```
function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}
```

🗒️ Notes

There are lots of elements to this. In one sense, you really don't need to know the details, but being aware is what coding is all about.

🛠️ Code Explanation

`if (!navigator.serial)`	Checks to see if the browser is compatible
`alert ("WebSerial is not supported in this browser. Try Chrome")`	If not then it sends a message to you that you need to consider a different browser
`serial.getPorts()`	Gets a list of available ports it has permission to use
`serial.on("noport", makePortButton)`	We go to the make a button function when we haven't connected with a port yet
`serial.on("portavailable", openPort)`	If a port is now connected start the process of sending/receiving data
`serial.on("requesterror", portError)`	Means there might be a problem or error of some kind
`serial.on("close", makePortButton)`	This is when the port closes for any reason, intentional or in error

Sketch B2.4 making a button

When you start off, you will need to choose a port. We have a button that we click on to take us to the port menu, and from there, select the port the Arduino is on.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}
```

Notes

You can position the button anywhere you like. When you click on the button, it will take you to another function called `choosePort()`, which we haven't created yet.

Code Explanation

`portButton = createButton("choose port")`	Creating a button
`portButton.position(10, 10)`	Putting it somewhere other than the default
`portButton.mousePressed(choosePort)`	When clicked the button action takes us to a function called choosePort

Sketch B2.5 choosing a port

There is a useful function that shows the available ports.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
  serial.requestPort()
}
```

Notes

This operation gives you a drop-down menu (later) for available ports; you will just have to find the one that the Arduino is connected to.

🛠️ Code Explanation

`serial.requestPort()`	Gives a selection of available ports

Sketch B2.6 port available

This opens the port that is connected to the Arduino.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
  serial.requestPort()
}

function openPort()
{
  serial.open().then(initiateSerial)
  function initiateSerial()
```

```
  {
    console.log("port open")
    if (portButton) portButton.hide()
  }
}
```

🛠️ Code Explanation

`serial.open().then(initiateSerial)`	The connection has successfully been made to the microcontroller

Sketch B2.7 error report

If there is a problem with the connection or the port, an error message is displayed.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
  serial.requestPort()
}

function openPort()
{
  serial.open().then(initiateSerial)
  function initiateSerial()
```

```
  {
    console.log("port open")
    if (portButton) portButton.hide()
  }
}
```

```
function portError(err)
{
  alert("Serial port error: " + err)
}
```

🗒️ Notes

Helps to debug.

🛠️ Code Explanation

`alert("Serial port error: " + err)`	This puts a message on the screen

Sketch B2.8 confirmation

This lets you know all is well.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
  serial.requestPort()
}

function openPort()
{
  serial.open().then(initiateSerial)
  function initiateSerial()
```

```
  {
    console.log("port open")
    if (portButton) portButton.hide()
  }
}

function portError(err)
{
  alert("Serial port error: " + err)
}
```

```
function portConnect()
{
  console.log("port connected")
  serial.getPorts()
}
```

Sketch B2.9 disconnected

Fairly obvious, it lets you know when it has disconnected from the port.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
  serial.requestPort()
}

function openPort()
{
  serial.open().then(initiateSerial)
  function initiateSerial()
```

```
  {
    console.log("port open")
    if (portButton) portButton.hide()
  }
}

function portError(err)
{
  alert("Serial port error: " + err)
}

function portConnect()
{
  console.log("port connected")
  serial.getPorts()
}

function portDisconnect()
{
  serial.close()
  console.log("port disconnected")
}
```

🛠️ Code Explanation

`serial.close()`	This stops the communication with the port

Sketch B2.10 closing the port

Does the same function as above but without the messaging.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
  serial.requestPort()
}

function openPort()
{
  serial.open().then(initiateSerial)
  function initiateSerial()
```

```
  {
    console.log("port open")
    if (portButton) portButton.hide()
  }
}

function portError(err)
{
  alert("Serial port error: " + err)
}

function portConnect()
{
  console.log("port connected")
  serial.getPorts()
}

function portDisconnect()
{
  serial.close()
  console.log("port disconnected")
}

function closePort()
{
  serial.close()
}
```

Let's test it out

Now we have the opening of the port code working (hopefully), and you have connected and uploaded your Arduino code, we can start to interact with it. All we are going to do is click the mouse, and it will switch the LED `on` or `off`.

For dramatic effect, we will have a circle change colour accordingly.

Sketch B2.11 starting sketch

❗ Now in the sketch.js file,
Sticking a circle in the middle of the canvas.

sketch.js

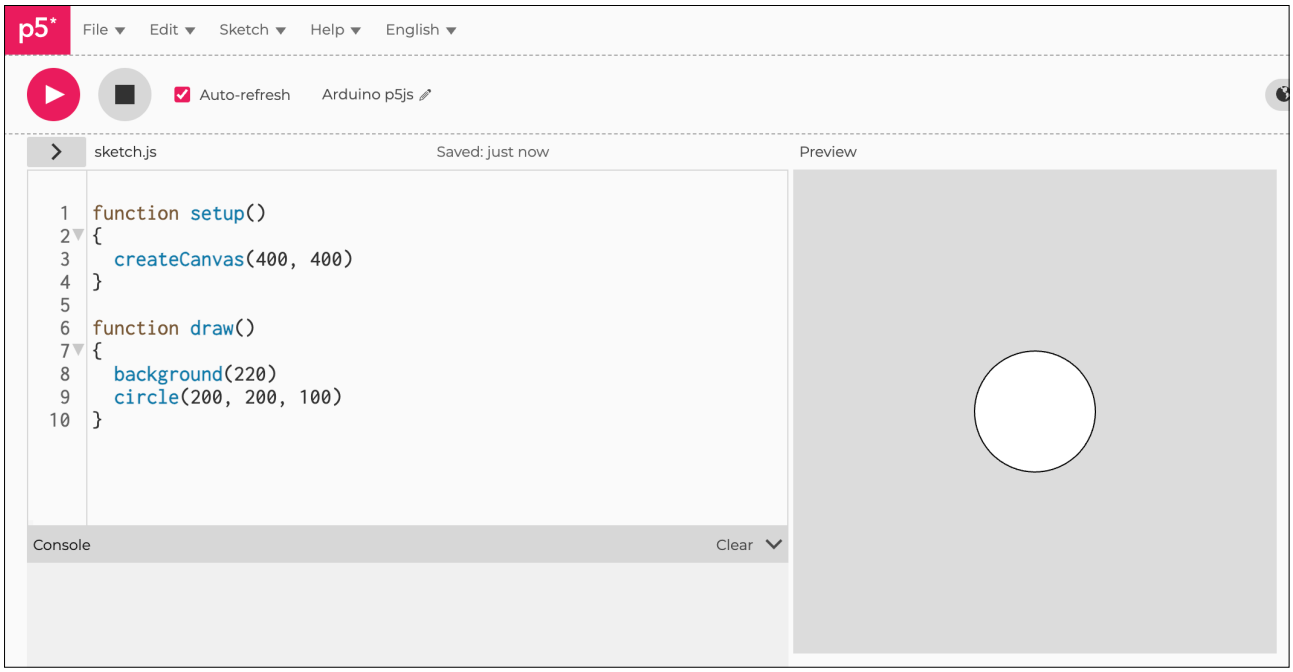
```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  circle(200, 200, 100)
}
```

Notes

Just the usual.

Figure B2.11



Sketch B2.12 LED on/off

Now, when you click on the canvas, the LED is on; when you release the mouse button, the LED is `off`. We have also included the `navigation()` function in setup to activate the serial communication.

sketch.js

```
function setup()
{
  createCanvas(400, 400)
  navigation()
}

function draw()
{
  background(220)
  if (mouseIsPressed)
  {
    serial.write("HIGH\n")
  }
  else
  {
    serial.write("LOW\n")
  }
  circle(200, 200, 100)
}
```

🗒️ Notes

The `\n` means new line, which corresponds with the Arduino code that means the command has finished.

🛠️ Code Explanation

`serial.write("HIGH\n")`	Sends data followed by a new line command

Sketch B2.13 the circle response

Now when you click on the canvas, the LED goes on/off, and the circle changes from grey to yellow.

sketch.js

```
function setup()
{
  createCanvas(400, 400)
  navigation()
}

function draw()
{
  background(220)
  if (mouseIsPressed)
  {
    serial.write("HIGH\n")
    fill('yellow')
  }
  else
  {
    serial.write("LOW\n")
    noFill()
  }
  circle(200, 200, 100)
}
```

Figure B2.13a: mouse pressed

Figure B2.13b: mouse not pressed