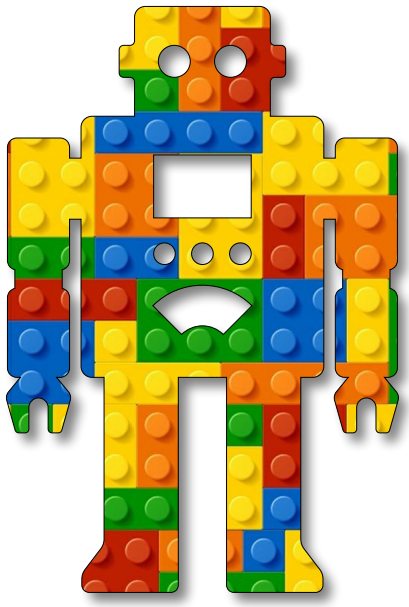
Intelligent Machines Module C Unit #1 collecting data



Content

Module C Unit #1 collecting data

Introduction to collecting data

We are collecting data from the accelerometer from the built-in IMU.

To train a model, we need data. We can create our own data called `synthetic` data, but ideally, you want good-quality `real-world` data and lots of it. Collecting data and then tidying it up is very time-consuming and costly. Yet, companies do this so they can train their models on huge amounts of data.

We could do the following by creating rough synthetic data that mirrors real data, but actually, we can collect real data from the IMU and use that. We want the accelerometer data from the movement in multiple directions. We save the data into a `JSON` file. Labelling each set of data as vertical, horizontal, circular, and none.

These are the four movements (or non-movements) we require. For the sake of brevity, we will have ten bits of data in each group of movements. This is not very much at all, and you might consider increasing it to `100`, but for this simple exercise, we will just do `10`.

There are numerous ways we can do this, and I have selected one, but you might have a better or easier way of doing this.

The index.html file

We have the ml5.js library and the port.js file as before.

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>
      <script src="https://unpkg.com/ml5@1/dist/ml5.min.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

Nothing new.

The port.js file

! This is our port.js, just for reference.

port.js

```
const serial = new p5.WebSerial()
let portButton
let inString
let list = []
let x = 0
let y = 0
let z

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("data", serialEvent)
  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()
}

function serialEvent()
{
  inString = serial.readStringUntil("\r\n")
  if (inString)
```

```
  {
    list = splitTokens(inString, ",")
    if (list.length > 2)
    {
      x = float(list[0])
      y = float(list[1])
      z = float(list[2])
    }
  }
}
```

🗒️ Notes

Just the same as before.

Sketch C1.1 our basic Arduino sketch

❗ The basic Arduino Nano sketch for the IMU.

Arduino sketch

```
#include "Arduino_BMI270_BMM150.h"
float x;
float y;
float z;

void setup()
{
  Serial.begin(9600);
  if (!IMU.begin())
  {
    Serial.println("Failed to initialise IMU");
    while (true);
  }
}

void loop()
{
  if (IMU.accelerationAvailable())
  {
    IMU.readAcceleration(x, y, z);
  }
    Serial.print(x);
    Serial.print(",");
    Serial.print(y);
    Serial.print(",");
    Serial.println(z);
}
```

🗒️ Notes

Nothing new here.

Sketch C1.2 collecting data

❗ Start a new sketch.js.

We repeat this for none, horizontal, vertical, and circular. Each time we run this, you have to change the name of the movement. The line of code: label: "xxxxxx" needs to have the name of the movement inserted here.

sketch.js

```
let count = 0
let data = []
let button

function setup()
{
  createCanvas(400, 400)
  navigation()
  button = createButton('save data')
}

function draw()
{
  background(220)
  let inputs = {
    x: x,
    y: y,
    label: "xxxxxx"
  }

  data.push(inputs)
  count++
  if (count == 10)
  {
    noLoop()
  }
  button.mousePressed(saveData)
  console.log(data)
}
```

```
function saveData()
{
  save(data, 'accelerometer.json', true)
}
```

🗒️ Notes

We will save the data to a file called `accelerometer.json`. You will get `four` sets of data. We do this for `10` bits of data in each set of movements. This is nowhere near enough, but we don't want to be spending all day training at this stage. We want to get the concept first.

🌻 Challenge

For ease of use, I have given each file the same name, but it would be better to have the file name reflect the data labelling as well.

🛠️ Code Explanation

`save(data, 'accelerometer.json', true)`	This creates a json file called accelerometer

Figure 1: no movement


```
[{"x":-0.03,"y":0.02,"label":"none"},{"x":-0.03,"y":0.02,"label":"none"},
{"x":-0.03,"y":0.02,"label":"none"},{"x":-0.03,"y":0.02,"label":"none"},
{"x":-0.03,"y":0.01,"label":"none"},{"x":-0.03,"y":0.02,"label":"none"},
{"x":-0.03,"y":0.01,"label":"none"},{"x":-0.03,"y":0.01,"label":"none"},
{"x":-0.03,"y":0.01,"label":"none"},{"x":-0.03,"y":0.02,"label":"none"}]
```


Figure 2: vertical movement


```
[{"x":0.2,"y":0.07,"label":"vertical"},{"x":0.13,"y":-0.04,"label":"vertical"},
{"x":0.25,"y":-0.01,"label":"vertical"},{"x":0.15,"y":-0.03,"label":"vertical"},
{"x":0.04,"y":-0.02,"label":"vertical"},{"x":0.01,"y":0.02,"label":"vertical"},
{"x":-0.04,"y":0.03,"label":"vertical"},{"x":-0.02,"y":0.01,"label":"vertical"},
{"x":-0.03,"y":0,"label":"vertical"},{"x":-0.07,"y":-0.01,"label":"vertical"}]
```


Figure 3: circular movement


```
[{"x":0.19,"y":-0.29,"label":"circular"},{"x":0.16,"y":-0.37,"label":"circular"},
{"x":0.05,"y":-0.36,"label":"circular"},{"x":-0.02,"y":-0.37,"label":"circular"},
{"x":-0.11,"y":-0.45,"label":"circular"},{"x":-0.12,"y":-0.42,"label":"circular"},
{"x":-0.27,"y":-0.36,"label":"circular"},{"x":-0.32,"y":-0.31,"label":"circular"},
{"x":-0.37,"y":-0.19,"label":"circular"},{"x":-0.44,"y":-0.07,"label":"circular"}]
```


Figure 4: horizontal movement


```
[{"x":0.14,"y":0.29,"label":"horizontal"},{"x":0.1,"y":0.28,"label":"horizontal"},
{"x":0.06,"y":0.14,"label":"horizontal"},{"x":0.03,"y":-0.06,"label":"horizontal"},
{"x":-0.01,"y":-0.11,"label":"horizontal"},{"x":-0.02,"y":-0.22,"label":"horizontal"},
{"x":-0.02,"y":-0.24,"label":"horizontal"},{"x":-0.05,"y":-0.28,"label":"horizontal"},
{"x":-0.08,"y":-0.34,"label":"horizontal"},{"x":-0.13,"y":-0.48,"label":"horizontal"}]
```

Reformatting

We want the format to be like this, so copy the data into a document and use the find and replace function to arrange the data as it shows below. This mirrors the data format from the gesture example. Also, make sure the speech marks are the correct style when you copy this data into the Sketch; for instance, “none” won't work, it needs to be "none". You'll soon find out!

One final note, make sure that the square brackets and the commas all match up in the sketch.

```
[{x: -0.03, y: 0.02, label: "none"},
{x: -0.03, y: 0.02, label: "none"},
{x: -0.03, y: 0.02, label: "none"},
{x: -0.03, y: 0.02, label: "none"},
{x: -0.03, y: 0.01, label: "none"},
{x: -0.03, y: 0.02, label: "none"},
{x: -0.03, y: 0.01, label: "none"},
{x: -0.03, y: 0.01, label: "none"},
{x: -0.03, y: 0.01, label: "none"},
{x: -0.03, y: 0.02, label: “none”}]

[{x: 0.14, y: 0.29, label: "horizontal"},
{x: 0.10, y: 0.28, label: "horizontal"},
{x: 0.06, y: 0.14, label: "horizontal"},
{x: 0.03, y: -0.06, label: "horizontal"},
{x: -0.01, y: -0.11, label: "horizontal"},
{x: -0.02, y: -0.22, label: "horizontal"},
{x: -0.02, y: -0.24, label: "horizontal"},
{x: -0.05, y: -0.28, label: "horizontal"},
{x: -0.08, y: -0.34, label: "horizontal"},
{x: -0.13, y: -0.48, label: "horizontal"}]

[{x: 0.20, y: 0.07, label: "vertical"},
{x: 0.13, y: -0.04, label: "vertical"},
{x: 0.25, y: -0.01, label: "vertical"},
{x: 0.15, y: -0.03, label: "vertical"},
{x: 0.04, y: -0.02, label: "vertical"},
{x: 0.01, y: 0.02, label: "vertical"},
{x: -0.04, y: 0.03, label: "vertical"},
{x: -0.02, y: 0.01, label: "vertical"},
{x: -0.03, y: 0.00, label: "vertical"},
{x: -0.07, y: -0.01, label: “vertical"}]

[{x: 0.19, y: -0.29, label: "circular"},
{x: 0.16, y: -0.37, label: "circular"},
{x: 0.05, y: -0.36, label: "circular"},
{x: -0.02, y: -0.37, label: "circular"},
{x: -0.11, y: -0.45, label: "circular"},
{x: -0.12, y: -0.42, label: "circular"},
{x: -0.27, y: -0.36, label: "circular"},
{x: -0.32, y: -0.31, label: "circular"},
{x: -0.37, y: -0.19, label: "circular"},
{x: -0.44, y: -0.07, label: “circular"}]
```

The order doesn't matter

The above becomes the following: all you need to do is copy and paste, and then do a bit of tidying.

```
[{x: -0.03, y: 0.02, label: “none"},
{x: -0.03, y: 0.02, label: "none"},
{x: -0.03, y: 0.02, label: "none"},
{x: -0.03, y: 0.02, label: "none"},
{x: -0.03, y: 0.01, label: "none"},
{x: -0.03, y: 0.02, label: "none"},
{x: -0.03, y: 0.01, label: "none"},
{x: -0.03, y: 0.01, label: "none"},
{x: -0.03, y: 0.01, label: "none"},
{x: -0.03, y: 0.02, label: “none”},
{x: 0.14, y: 0.29, label: "horizontal"},
{x: 0.10, y: 0.28, label: "horizontal"},
{x: 0.06, y: 0.14, label: "horizontal"},
{x: 0.03, y: -0.06, label: "horizontal"},
{x: -0.01, y: -0.11, label: "horizontal"},
{x: -0.02, y: -0.22, label: "horizontal"},
{x: -0.02, y: -0.24, label: "horizontal"},
{x: -0.05, y: -0.28, label: "horizontal"},
{x: -0.08, y: -0.34, label: "horizontal"},
{x: -0.13, y: -0.48, label: “horizontal"},
{x: 0.20, y: 0.07, label: "vertical"},
{x: 0.13, y: -0.04, label: "vertical"},
{x: 0.25, y: -0.01, label: "vertical"},
{x: 0.15, y: -0.03, label: "vertical"},
{x: 0.04, y: -0.02, label: "vertical"},
{x: 0.01, y: 0.02, label: "vertical"},
{x: -0.04, y: 0.03, label: "vertical"},
{x: -0.02, y: 0.01, label: "vertical"},
{x: -0.03, y: 0.00, label: "vertical"},
{x: -0.07, y: -0.01, label: “vertical”},
[{x: 0.19, y: -0.29, label: "circular"},
{x: 0.16, y: -0.37, label: "circular"},
{x: 0.05, y: -0.36, label: "circular"},
{x: -0.02, y: -0.37, label: "circular"},
{x: -0.11, y: -0.45, label: "circular"},
{x: -0.12, y: -0.42, label: "circular"},
{x: -0.27, y: -0.36, label: "circular"},
{x: -0.32, y: -0.31, label: "circular"},
{x: -0.37, y: -0.19, label: "circular"},
{x: -0.44, y: -0.07, label: “circular”}]
```

Sketch C1.3 adjusting the data

Here I have reduced the values of x and y for "none" to give better results when the Arduino isn't moving. You should have something like this.

```
let data = [
  {x: -0.02, y: 0.02, label: "none"},
  {x: -0.02, y: 0.02, label: "none"},
  {x: -0.02, y: 0.02, label: "none"},
  {x: -0.02, y: 0.02, label: "none"},
  {x: -0.02, y: 0.00, label: "none"},
  {x: -0.02, y: 0.01, label: "none"},
  {x: -0.02, y: 0.00, label: "none"},
  {x: -0.02, y: 0.00, label: "none"},
  {x: -0.02, y: 0.00, label: "none"},
  {x: -0.02, y: 0.01, label: "none"},
  {x: 0.14, y: 0.29, label: "horizontal"},
  {x: 0.10, y: 0.28, label: "horizontal"},
  {x: 0.06, y: 0.14, label: "horizontal"},
  {x: 0.03, y: -0.06, label: "horizontal"},
  {x: -0.01, y: -0.11, label: "horizontal"},
  {x: -0.02, y: -0.22, label: "horizontal"},
  {x: -0.02, y: -0.24, label: "horizontal"},
  {x: -0.05, y: -0.28, label: "horizontal"},
  {x: -0.08, y: -0.34, label: "horizontal"},
  {x: -0.13, y: -0.48, label: "horizontal"},
  {x: 0.20, y: 0.07, label: "vertical"},
  {x: 0.13, y: -0.04, label: "vertical"},
  {x: 0.25, y: -0.01, label: "vertical"},
  {x: 0.15, y: -0.03, label: "vertical"},
  {x: 0.04, y: -0.02, label: "vertical"},
  {x: 0.01, y: 0.02, label: "vertical"},
  {x: -0.04, y: 0.03, label: "vertical"},
  {x: -0.02, y: 0.01, label: "vertical"},
  {x: -0.03, y: 0, label: "vertical"},
  {x: 0.20, y: 0.07, label: "vertical"},
  {x: 0.19, y: -0.29, label: "circular"},
  {x: 0.16, y: -0.37, label: "circular"},
```

```
  {x: 0.05, y: -0.36, label: "circular"},
  {x: -0.02, y: -0.37, label: "circular"},
  {x: -0.11, y: -0.45, label: "circular"},
  {x: -0.12, y: -0.42, label: "circular"},
  {x: -0.27, y: -0.36, label: "circular"},
  {x: -0.32, y: -0.31, label: "circular"},
  {x: -0.37, y: -0.19, label: "circular"},
  {x: -0.44, y: -0.07, label: "circular"}
]
```

🗒️ Notes

The above is how your data needs to look if it is to be read by the model for training purposes. Now we have our data, let's go and do something with it.

🌻 Challenge

Another solution might be to have four files that are loaded. That would be a neater solution, especially if there is a large amount of data.

❗ please keep all this for the next unit