

Baseball bat metrics

Setting the scene

Have you ever wondered why some people can hit a baseball faster than others? In this project you will create a sensor which measures the acceleration and direction in which a baseball bat is swung when hitting a ball. You will learn how to use the accelerometer and how to send the data to the Arduino IoT cloud ready to be analysed. A metric is a piece of data which allows the user to analyse their performance. In this project, a range of metrics will be used that are taken from the data gathered using the accelerometer.

Success criteria

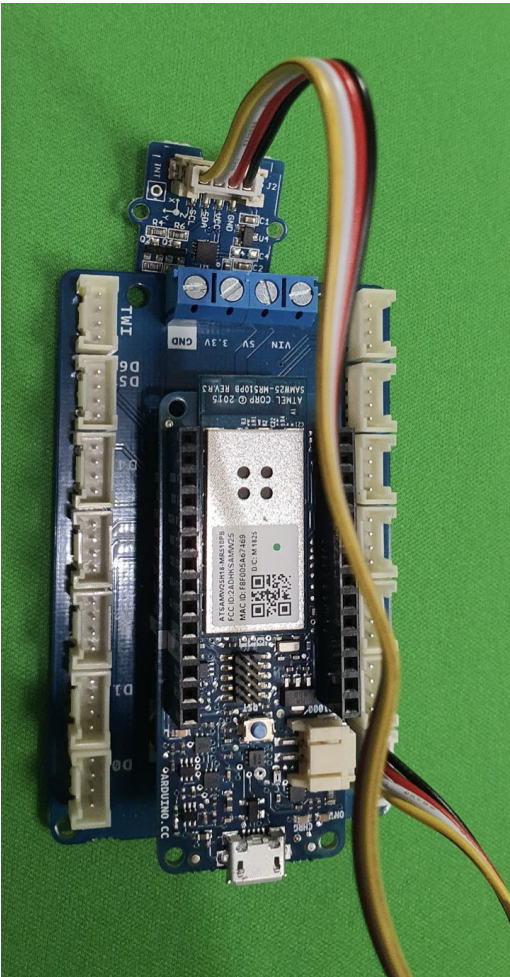
- The accelerometer and Arduino can be attached to a baseball bat securely
- Measures the movement of a baseball bat
- Records the data during a bat swing onto the Arduino IoT cloud

Instructions

The first step is to assemble your device.

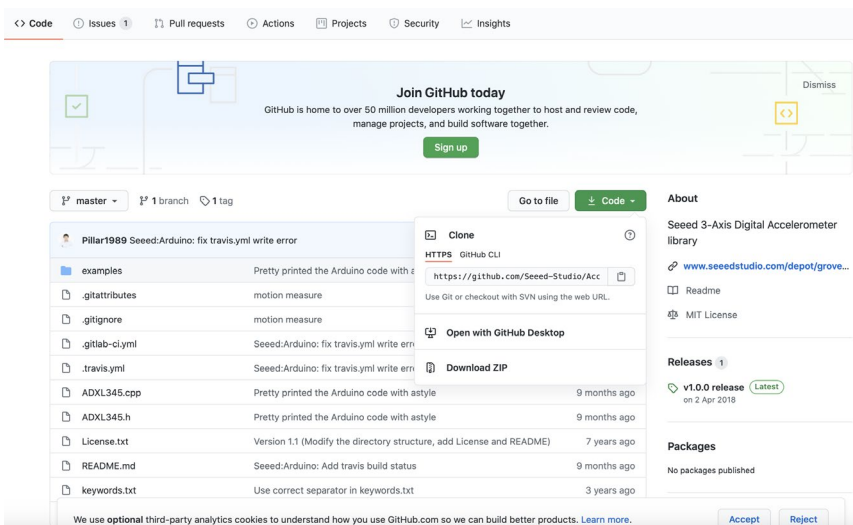
Plug the MKR1000 into the MKR Connector Carrier. Be careful to ensure that you line up the pins correctly and don't cause any of the pins to bend. You need to ensure that all of the pins on the MKR 1000 tie up with the equivalent labels on the connector carrier.

You should then plug in the Grove 3 Axis Digital Accelerometer into port A0 on the connector carrier.



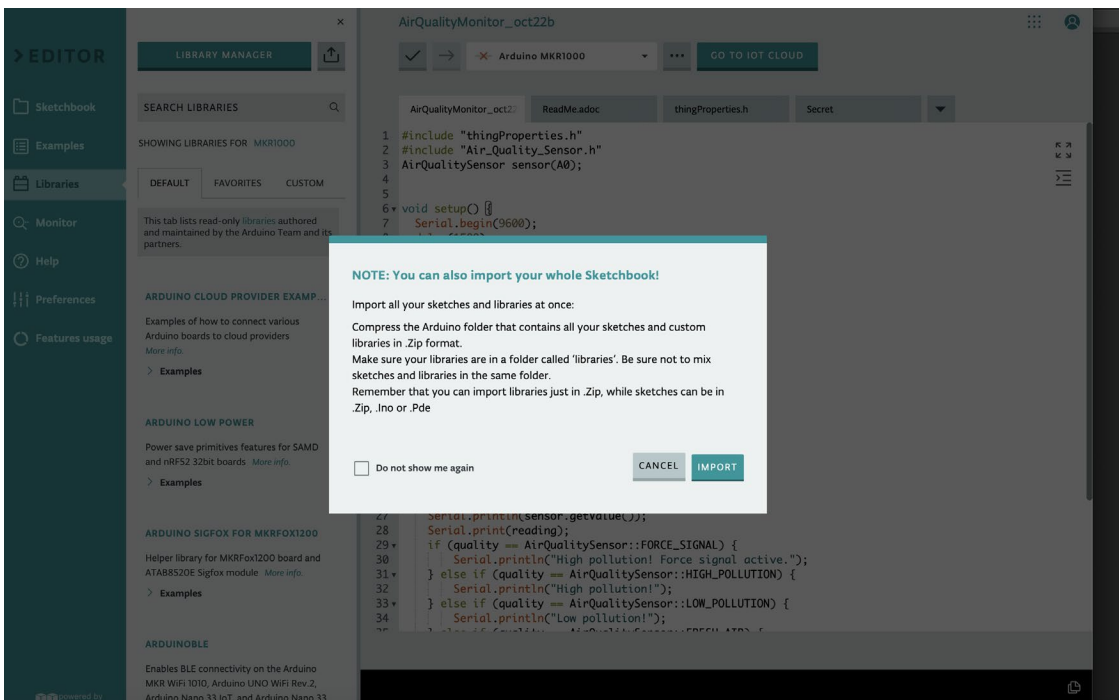
Your device is now assembled and ready to be programmed.

Initially you need to install the relevant libraries. The Accelerometer libraries can be downloaded from: https://github.com/Seeed-Studio/Accelerometer_ADXL345.

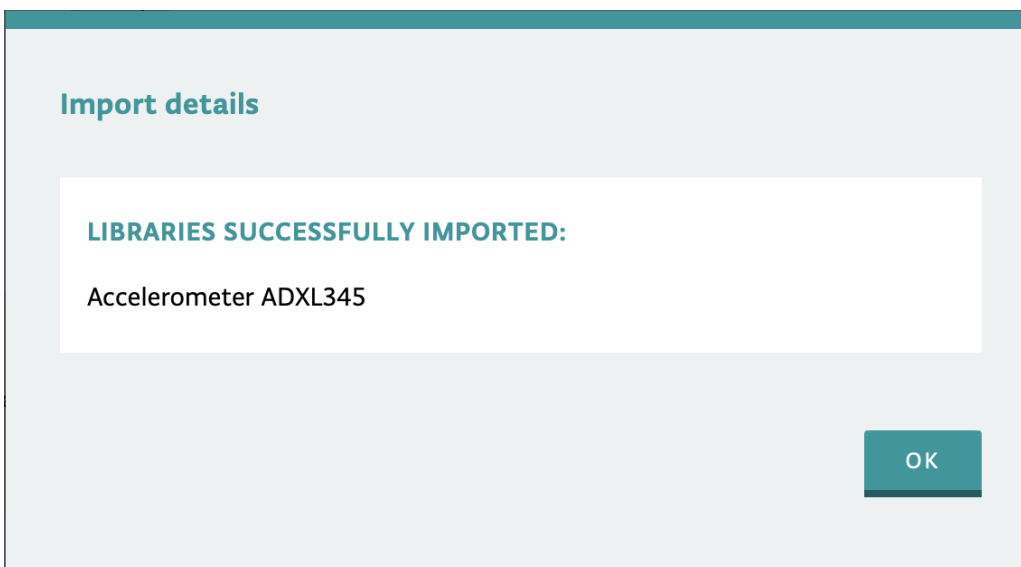


You should the click on 'Code' and download the zip file.

You now need to install the library into your IDE. Sign into the Arduino Cloud and go to the text editor. You should then click on 'Libraries'.



Once in the 'Libraries' menu, you will see an option to import a library. Click on this button and then select your recently downloaded zip file.

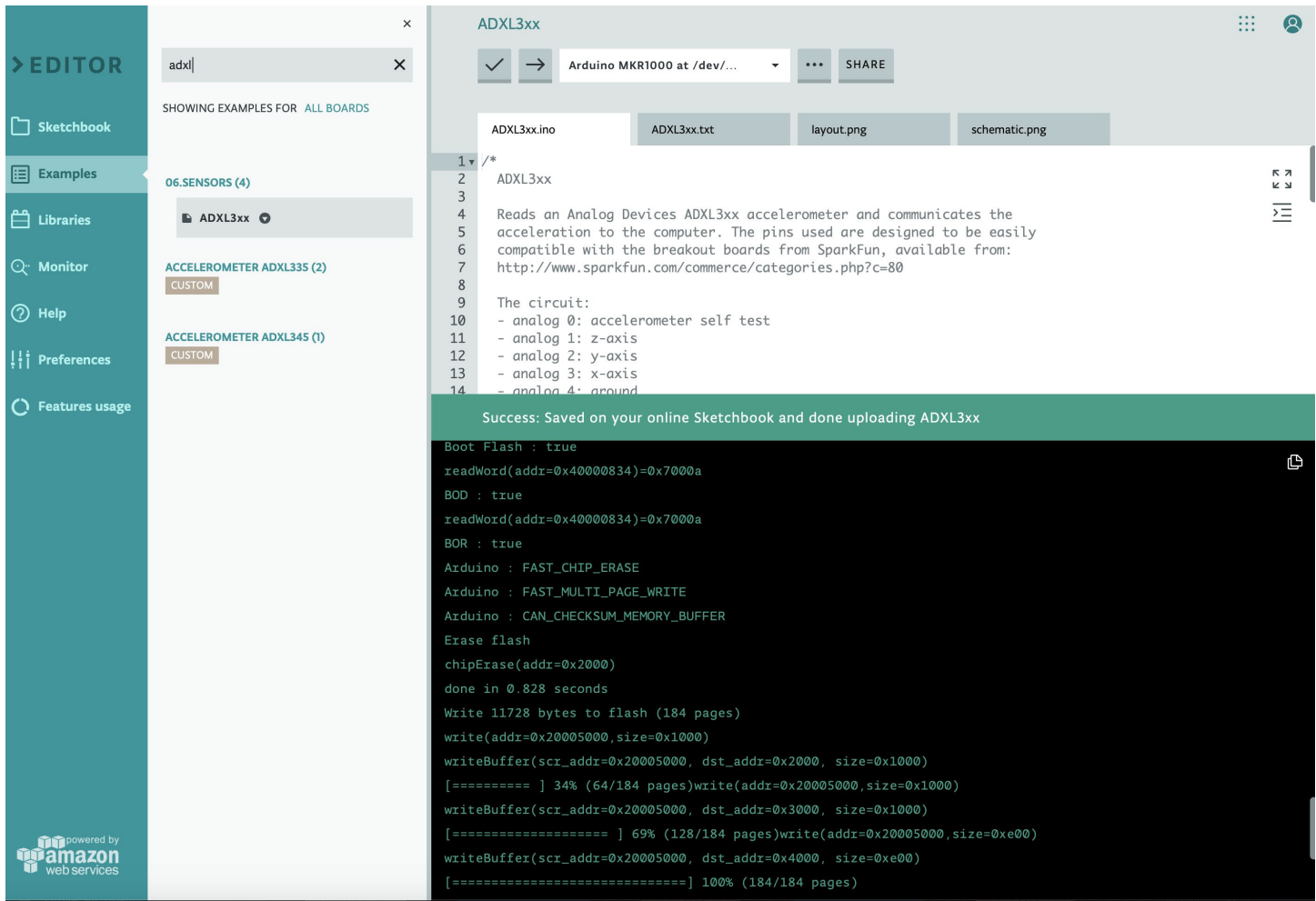


You will then receive a confirmation message to inform you whether the library has been successfully imported.

Pro-tip

When using a new type of sensor for the first time, it is useful to test it using the sample code that is provided by the manufacturer.

You are now going to test your device using some example code. Click on the examples tab and search for adxl. Then select ADXL3xx.



The screenshot shows the Arduino IDE interface. On the left, the 'Examples' tab is selected, and a search for 'adxl' has been performed. The 'ADXL3xx' example is highlighted. The main editor area displays the code for 'ADXL3xx.ino'. The code includes a comment describing the library and a list of pins used for the circuit. A green success message at the bottom states: 'Success: Saved on your online Sketchbook and done uploading ADXL3xx'. Below the message, the upload progress is shown, including the time taken to erase the flash and write the code.

```

1  /*
2  3  ADXL3xx
4  Reads an Analog Devices ADXL3xx accelerometer and communicates the
5  acceleration to the computer. The pins used are designed to be easily
6  compatible with the breakout boards from SparkFun, available from:
7  http://www.sparkfun.com/commerce/categories.php?c=80
8
9  The circuit:
10 - analog 0: accelerometer self test
11 - analog 1: z-axis
12 - analog 2: y-axis
13 - analog 3: x-axis
14 - analog 4: ground

```

Success: Saved on your online Sketchbook and done uploading ADXL3xx

```

Boot Flash : true
readWord(addr=0x40000834)=0x7000a
BOD : true
readWord(addr=0x40000834)=0x7000a
BOR : true
Arduino : FAST_CHIP_ERASE
Arduino : FAST_MULTI_PAGE_WRITE
Arduino : CAN_CHECKSUM_MEMORY_BUFFER
Erase flash
chipErase(addr=0x2000)
done in 0.828 seconds
Write 11728 bytes to flash (184 pages)
write(addr=0x20005000, size=0x1000)
writeBuffer(src_addr=0x20005000, dst_addr=0x2000, size=0x1000)
[===== ] 34% (64/184 pages)write(addr=0x20005000, size=0x1000)
writeBuffer(src_addr=0x20005000, dst_addr=0x3000, size=0x1000)
[===== ] 69% (128/184 pages)write(addr=0x20005000, size=0xe00)
writeBuffer(src_addr=0x20005000, dst_addr=0x4000, size=0xe00)
[=====] 100% (184/184 pages)

```

Upload this file to your device.

Testing your device while attached to the computer

Once you have successfully uploaded the file to your device, you should then select monitor from the left-hand menu. This will show you the readings that your device is currently taking. Move the accelerometer to check that the readings change.



EDITOR

PORT: Arduino MKR1000
at /dev/cu.usbmodem144101

Newline 9600 baud

SEND

ADXL3xx

ADXL3xx.ino ADXL3xx.txt layout.png schematic.png

1 /*
2 ADXL3xx
3
4 Reads an Analog Devices ADXL3xx accelerometer and communicates the
5 acceleration to the computer. The pins used are designed to be easily
6 compatible with the breakout boards from SparkFun, available from:
7 http://www.sparkfun.com/commerce/categories.php?c=88
8
9 The circuit:
10 - analog 0: accelerometer self test
11 - analog 1: z-axis
12 - analog 2: y-axis
13 - analog 3: x-axis
14 - analog 4: ground

Success: Saved on your online Sketchbook and done uploading ADXL3xx

```
writeBuffer(scr_addr=0x20005000, dst_addr=0x2000, size=0x1000)
[===== ] 34% (64/184 pages)write(addr=0x20005000, size=0x1000)
writeBuffer(scr_addr=0x20005000, dst_addr=0x3000, size=0x1000)
[===== ] 69% (128/184 pages)write(addr=0x20005000, size=0xe00)
writeBuffer(scr_addr=0x20005000, dst_addr=0x4000, size=0xe00)
[===== ] 100% (184/184 pages)
done in 0.069 seconds
Verify 11728 bytes of flash with checksum.
checksumBuffer(start_addr=0x2000, size=0x1000) = 74ae
checksumBuffer(start_addr=0x3000, size=0x1000) = 3e75
checksumBuffer(start_addr=0x4000, size=0xddd) = 86d1
Verify successful
done in 0.010 seconds
CPU reset.
readWord(addr=0)=0x20007ffc
readWord(addr=0xe000ed00)=0x410cc601
readWord(addr=0x41002018)=0x10010305
writeWord(addr=0xe000ed0c, value=0x5fa0004)
```

At this stage, you now know whether or not your device is taking readings. We now want to send these readings to the Arduino Cloud so that we can see a log of readings from throughout the day.

You should now return to the Arduino IoT Cloud and create a new thing.

CREATE NEW THING

Things are the logical representation of a connected object;
they represent the inherent properties of the object, with as little reference to the actual hardware used to implement them.
Each thing is represented by a collection of properties (e.g. temperature, light).

Enter a name for your Thing

Accelerometer

Select a device to associate with your Thing

MKR1000 via Cloud - MKR 1000

You should name your thing with a descriptive name and choose the board which it will run on. You should then create a new property. The first property has been named xAxis.



ADD NEW PROPERTY

Name * [?](#)

x axis

Variable Name * [?](#)

xAxis

Type * [?](#)

Int

Min value [?](#) Max value [?](#)

0 1000

Permission [?](#)

☒ Read & Write

☐ Read Only

Update [?](#) Delta [?](#)

☒ When the value changes

☐ Regularly

0

History [?](#)

☒ Show history visualization

Repeat the same process for the yAxis and zAxis.

You should now select 'Edit sketch'. This will take you to the basic code which has been automatically generated. You now need to write the sensor value back to the variable 'reading' so that it can be sent back to the IoT cloud.



```
Accelerometer_oct23a.in  ReadMe.adoc  thingProperties.h  S
1  #include "thingProperties.h"
2
3  void setup() {
4      Serial.begin(9600);
5      delay(1500);
6
7      initProperties();
8
9      ArduinoCloud.begin(ArduinoIoTPreferredConnection);
10     setDebugMessageLevel(2);
11     ArduinoCloud.printDebugInfo();
12 }
13
14 void loop() {
15     ArduinoCloud.update();
16     // Your code here
17
18 }
19
20
21 void onXAxisChange() {
22     // Do something
23 }
24
25
26 void onYAxisChange() {
27     // Do something
28 }
29
30
31 void onYAxisChange() {
32     // Do something
33 }
34
35
```

You should now insert the code from the example file.

```
Accelerometer_oct23a.in  ReadMe.adoc  thingProperties.h  Secret  ▼
#include "thingProperties.h"
// these constants describe the pins. They won't change:
const int groundpin = 18; // analog input pin 4 -- ground
const int powerpin = 19; // analog input pin 5 -- voltage
const int xpin = A3; // x-axis of the accelerometer
const int ypin = A2; // y-axis
const int zpin = A1; // z-axis (only on 3-axis models)

void setup() {
    Serial.begin(9600);
    delay(1500);
    pinMode(groundpin, OUTPUT);
    pinMode(powerpin, OUTPUT);
    digitalWrite(groundpin, LOW);
    digitalWrite(powerpin, HIGH);
    initProperties();
    ArduinoCloud.begin(ArduinoIoTPreferredConnection);
    setDebugMessageLevel(2);
    ArduinoCloud.printDebugInfo();
}
```

Firstly, initiate the constants at the start of the program. You should then complete the set-up loop.

This completes the set-up loop. You now need to add the main loop which handles the readings.

```
Accelerometer_oct23a.in  ReadMe.adoc  thingProperties
21
22 void loop() {
23   ArduinoCloud.update();
24   // print the sensor values:
25   Serial.print(analogRead(xpin));
26   // print a tab between values:
27   Serial.print("\t");
28   Serial.print(analogRead(ypin));
29   // print a tab between values:
30   Serial.print("\t");
31   Serial.print(analogRead(zpin));
32   Serial.println();
33   // delay before next reading:
34   delay(100);
35
36
37 }
```

The above code can be copied from the example file. It is now necessary to add additional code to tie up with the properties which you created earlier.

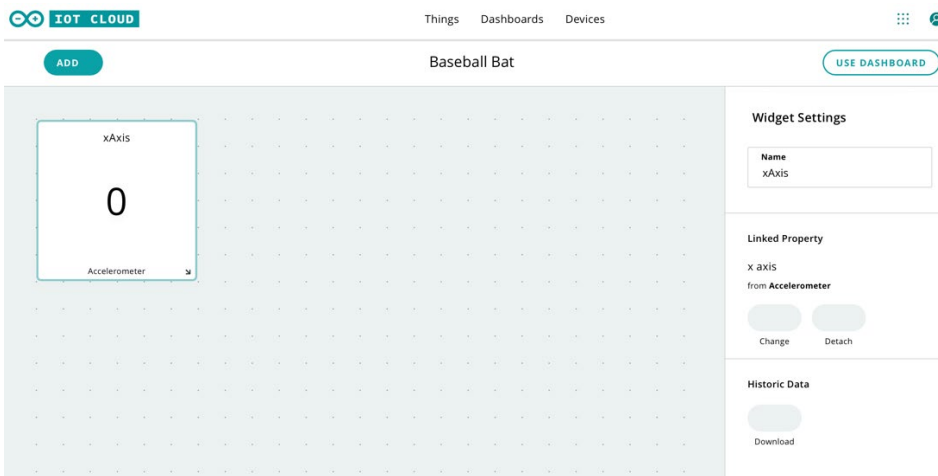


```
Accelerometer_oct23a.in  ReadMe.adoc  thingProperties.h
6  const int ypin = A2;           // y-axis
7  const int zpin = A1;           // z-axis (only on 3-axis)
8
9  void setup() {
10     Serial.begin(9600);
11     delay(1500);
12     pinMode(groundpin, OUTPUT);
13     pinMode(powerpin, OUTPUT);
14     digitalWrite(groundpin, LOW);
15     digitalWrite(powerpin, HIGH);
16     initProperties();
17     ArduinoCloud.begin(ArduinoIoTPreferredConnection);
18     setDebugMessageLevel(2);
19     ArduinoCloud.printDebugInfo();
20 }
21
22 void loop() {
23     ArduinoCloud.update();
24     // print the sensor values:
25     Serial.print(analogRead(xpin));
26     xAxis=int(analogRead(xpin));
27     // print a tab between values:
28     Serial.print("\t");
29     Serial.print(analogRead(ypin));
30     yAxis=int(analogRead(ypin));
31     // print a tab between values:
32     Serial.print("\t");
33     Serial.print(analogRead(zpin));
34     Serial.println();
35     zAxis=int(analogRead(xpin));
36     // delay before next reading:
37     delay(100);
38 }
```

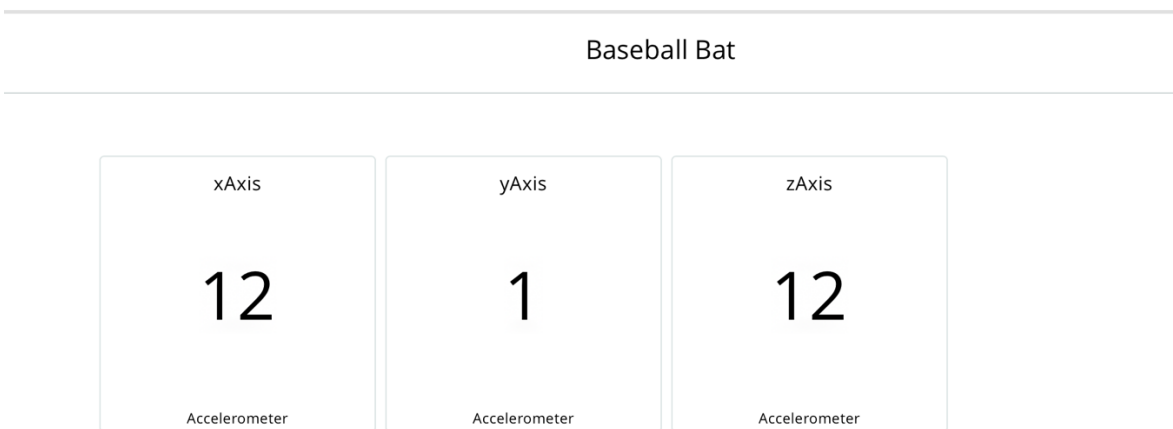
It is important to state that the reading is an integer in order to ensure that the data is successfully transferred to the cloud. You can now upload your program to your device and check in the monitor tab that values are recorded successfully. The Serial.print lines are the commands which are outputting to the monitor.

It is now time to create your IoT dashboard. Click on 'Return to IoT cloud' and then 'Dashboards'.

Create a new dashboard, give it a meaningful name and then add a value widget. Link the widget to the xAxis variable which you created earlier.



Repeat the process for the y and z axis. You will then see your data being displayed live.



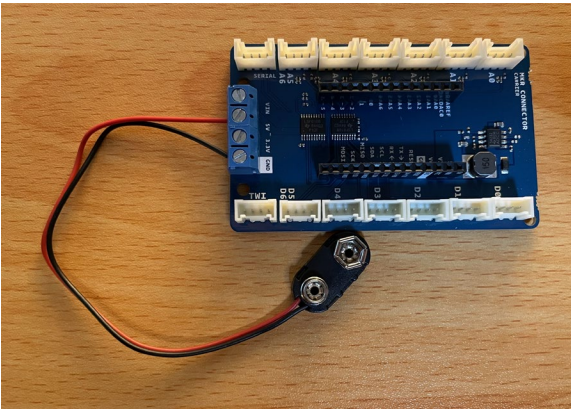
Once you have a set of data ready for analysis you could download all recorded data as a csv file by selecting download. You can then either analyse it using a spreadsheet or a coded application.

Testing your device is communicating with the IoT Cloud

It is now time to test your device. Move your device in a range of different directions. You should see that the values are displayed on the IoT dashboard.

Attaching your device to a baseball bat

When using the device attached to a baseball bat, we recommend that you power the device with a battery so that it doesn't need to be attached to a computer. You can use the battery connected provided with the kit. Connect the red wire to 5V and the black wire to GND.



It will also need to be within Wi-Fi range. We suggest that you wrap the device with a foam padding to help protect it. Mount the device away from the area where the baseball will be struck.

Testing your device whilst attached to a baseball bat

Your device should now be securely attached to the bat. Try swinging it at different speeds and different directions. Look at how the measurements change as you swing the bat.

Stretch tasks

Try purposely hitting a ball in different directions and see which measurements hit the ball the furthest. Can you develop a successful set of metrics which predict the most successful strike?