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

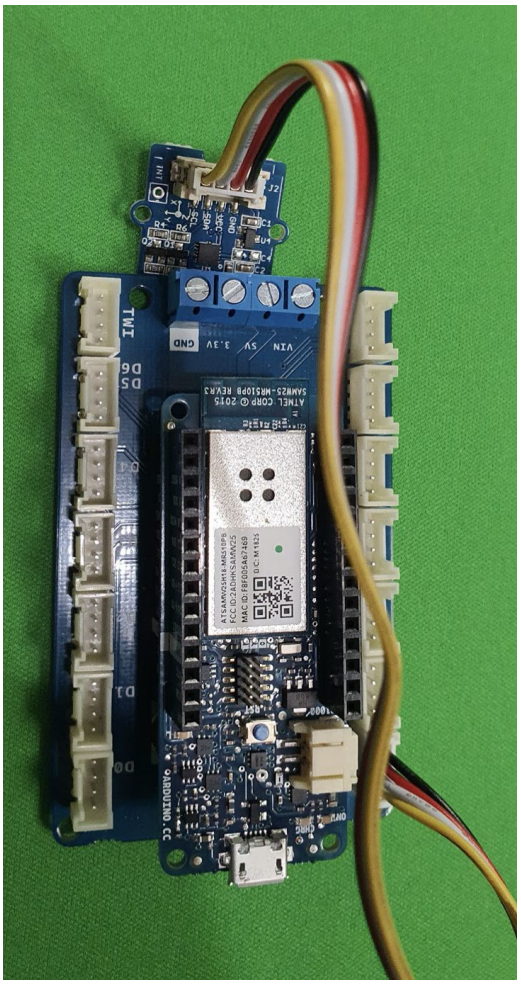
- The accelerometer and Arduino can be attached to a baseball bat securely
- Measures the movement of a baseball bat
- Records the data during a bat swing onto the Arduino IoT cloud

Instructions

The first step is to assemble your device.

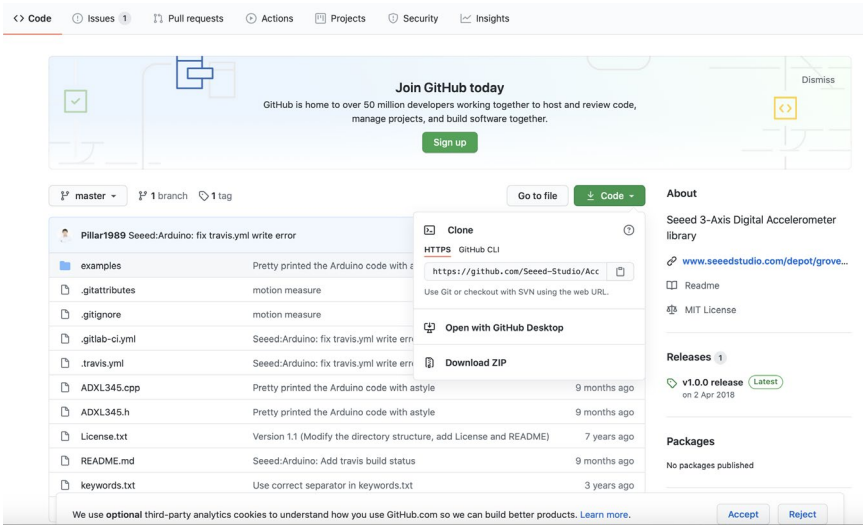
Plug the MKR1000 into the MKR Connector Carrier. Be careful to ensure that you line up the pins correctly and don't cause any of the pins to bend. You need to ensure that all of the pins on the MKR 1000 tie up with the equivalent labels on the connector carrier.

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



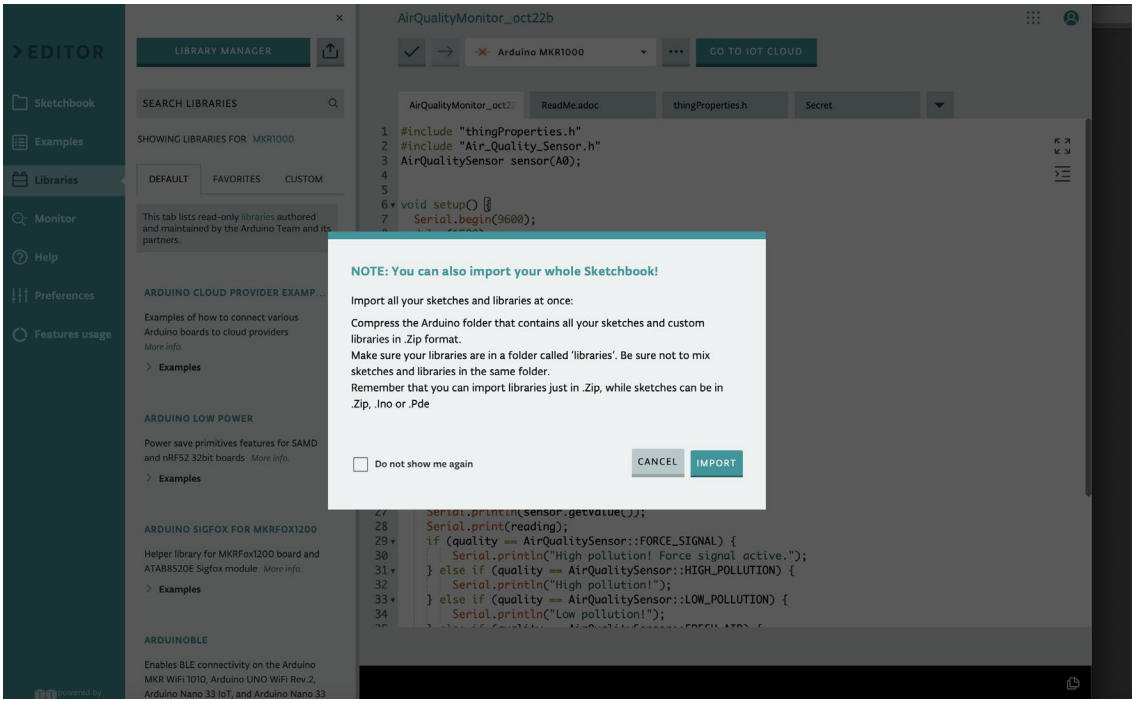
Your device is now assembled and ready to be programmed.

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

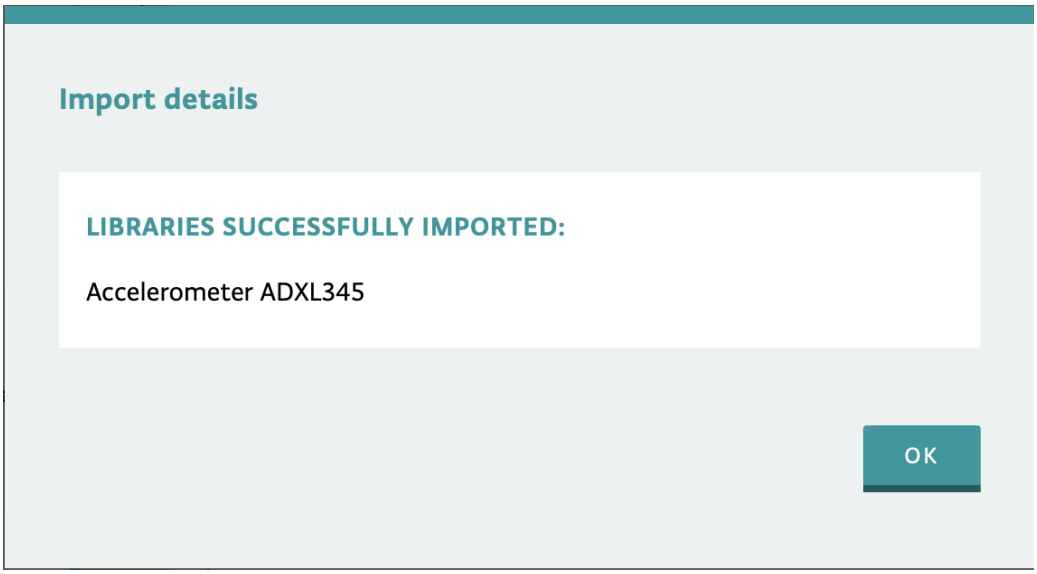


You should the click on ‘Code’ and download the zip file.

You now need to install the library into you 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.

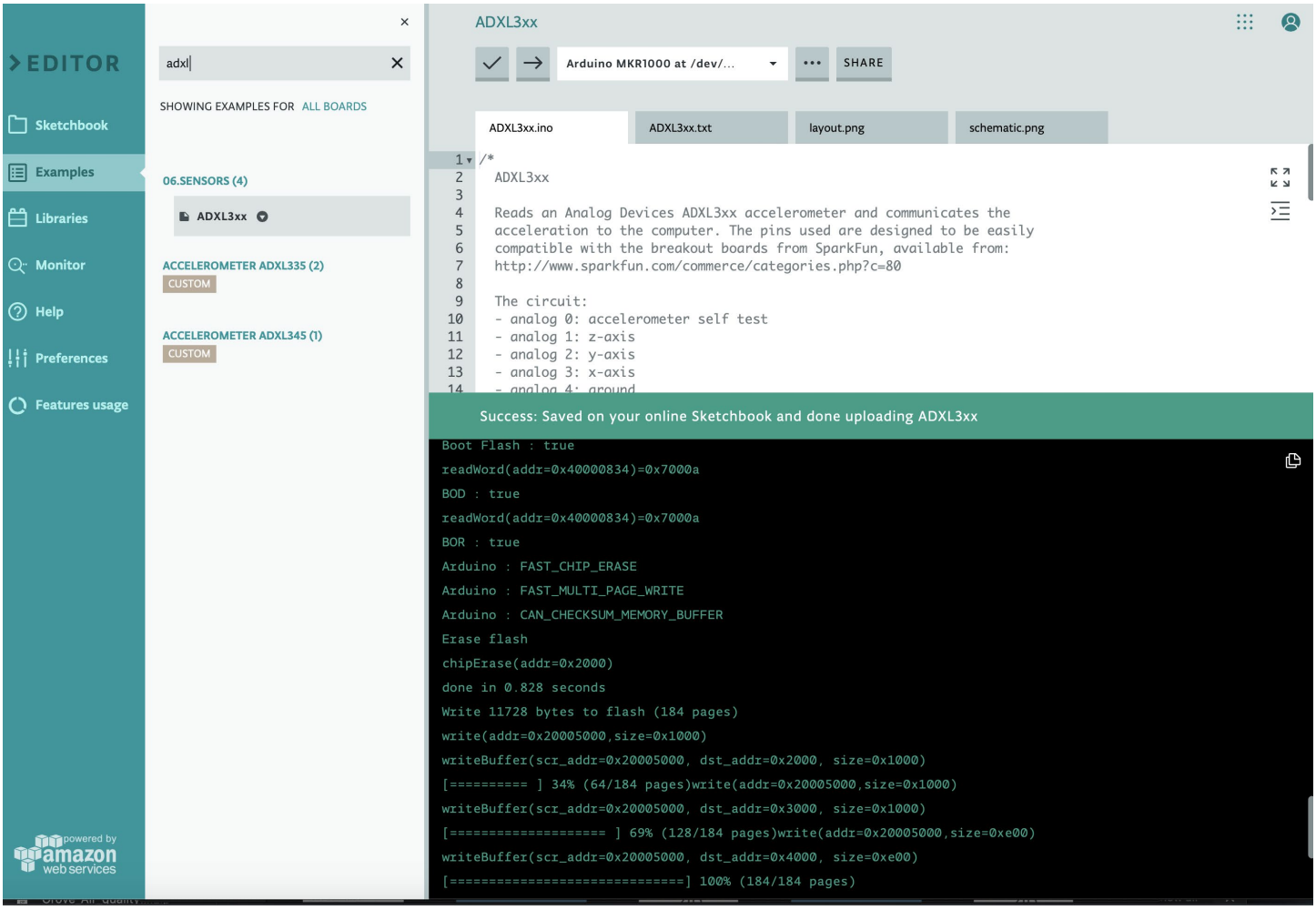


You will then receive a confirmation message to inform you whether the library has been successfully imported.

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.

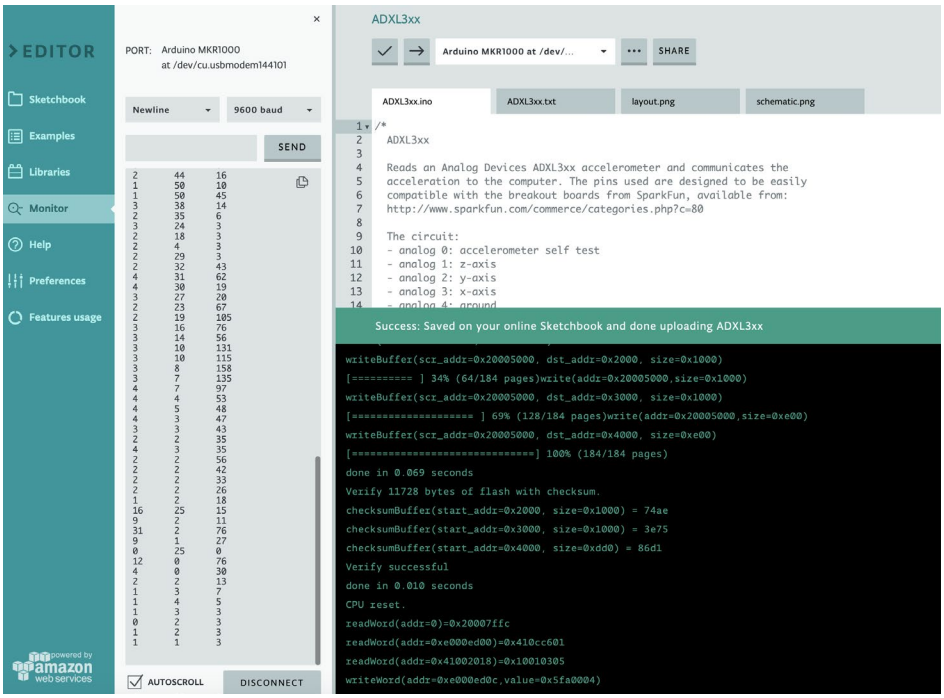
You are now going to test your device using some example code. Click on the examples tab and search for adxl. Then select ADXL3xx.



Upload this file to your device.

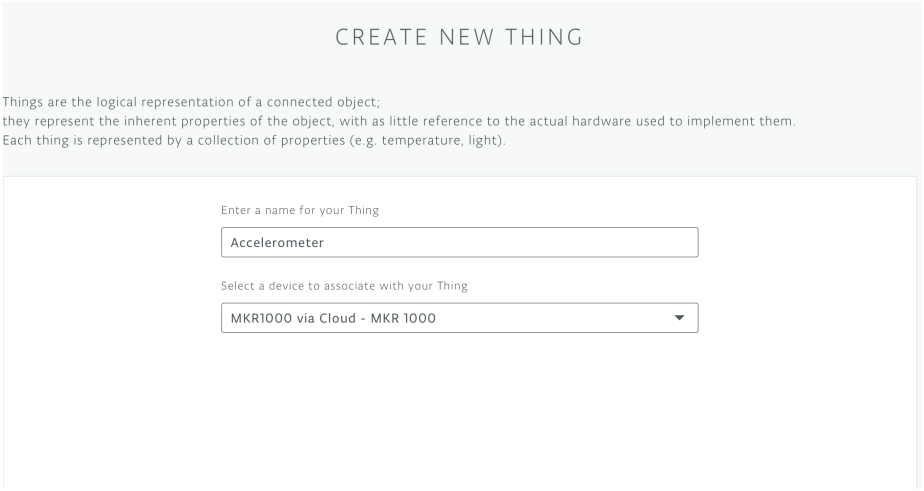
Testing your device while attached to the computer

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.

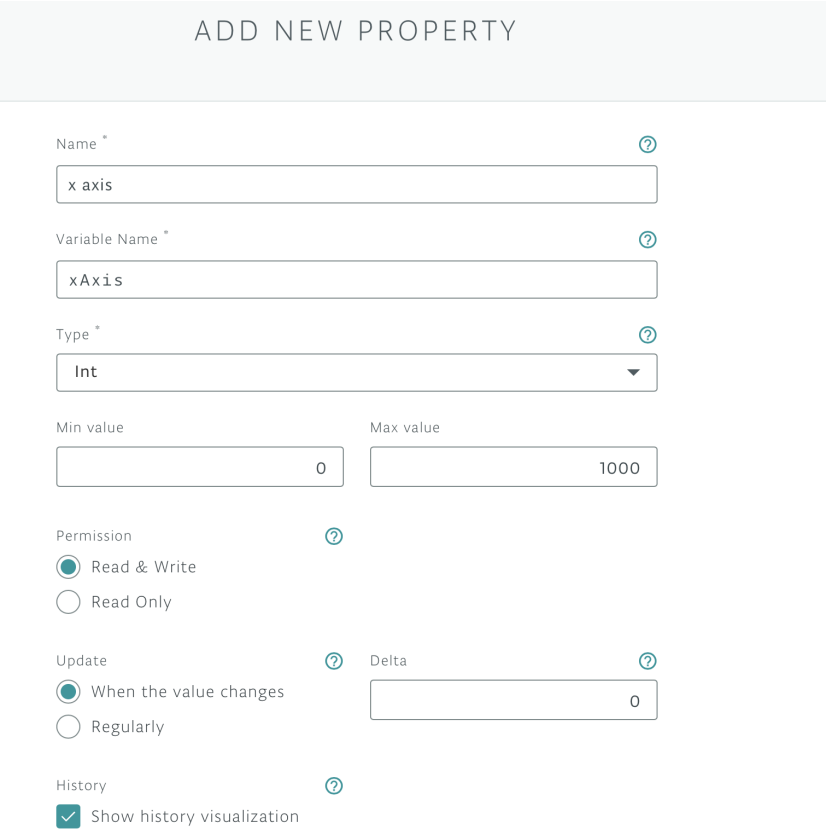


At this stage, you now know whether or not your device is taking readings. We now want to send these readings to the Arduino Cloud so that we can see a log of readings from throughout the day.

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



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.



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

```
#include "thingProperties.h"

void setup() {
  Serial.begin(9600);
  delay(1500);

  initProperties();

  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();
  // Your code here


}

void onXAxisChange() {
  // Do something
}


void onYaxisChange() {
  // Do something
}


void onYAxisChange() {
  // Do something
}
```

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.

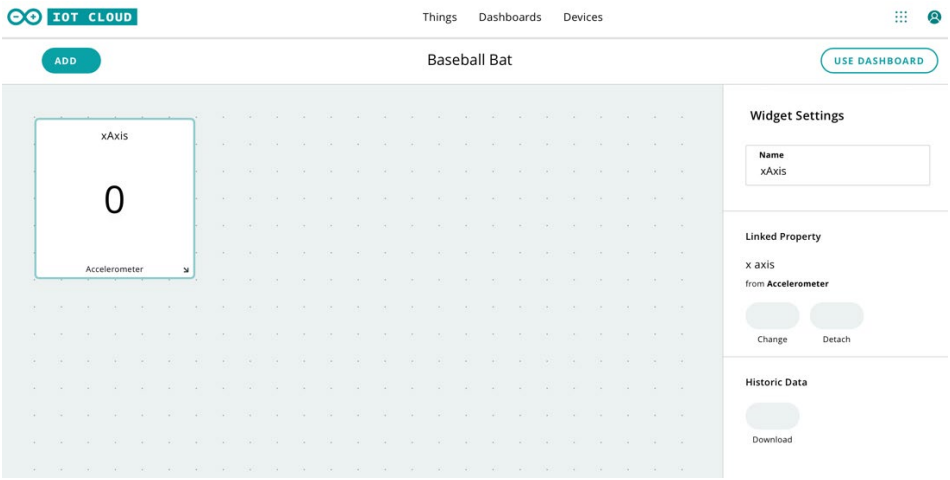
Accelerometer_oct23a.in | ReadMe.adoc | thingProperties.h

```
const int ypin = A2;                    // y-axis
const int zpin = A1;                    // z-axis (only on 3-axis

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();
}

void loop() {
  ArduinoCloud.update();
    // print the sensor values:
  Serial.print(analogRead(xpin));
  xAxis=int(analogRead(xpin));
  // print a tab between values:
  Serial.print("\t");
  Serial.print(analogRead(ypin));
  yAxis=int(analogRead(ypin));
  // print a tab between values:
  Serial.print("\t");
  Serial.print(analogRead(zpin));
  Serial.println();
  zAxis=int(analogRead(xpin));
  // delay before next reading:
  delay(100);

```

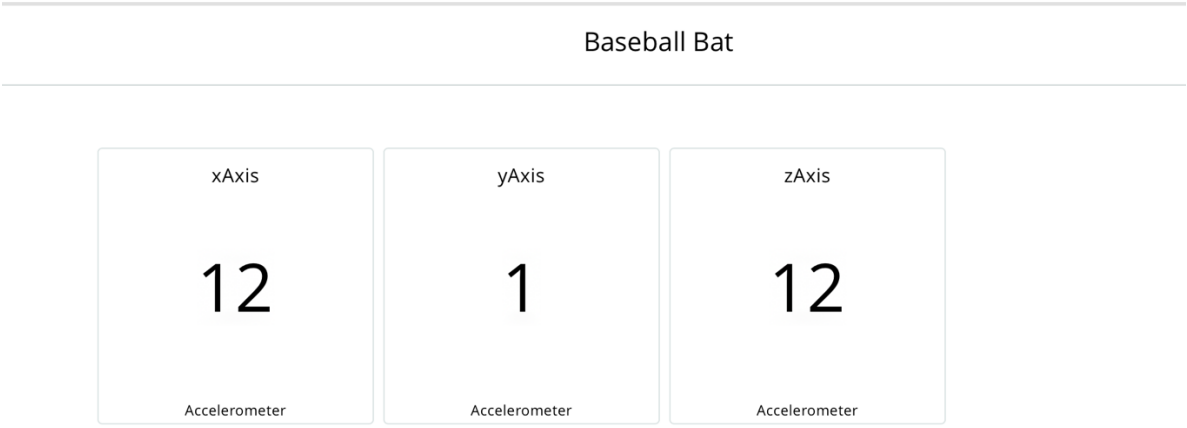
It is important to state that the reading is an integer in order to ensure that the data is successfully transferred to the cloud. You can now upload your program to your device and check in the monitor tab that values are recorded successfully. The Serial.print lines are the commands which are outputting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



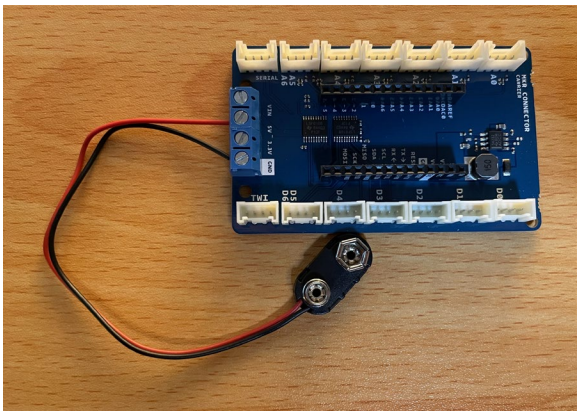
Once you have a set of data ready for analysis you could download all recorded data as a csv file by selecting download. You can then either analyse it using a spreadsheet or a coded application.

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?