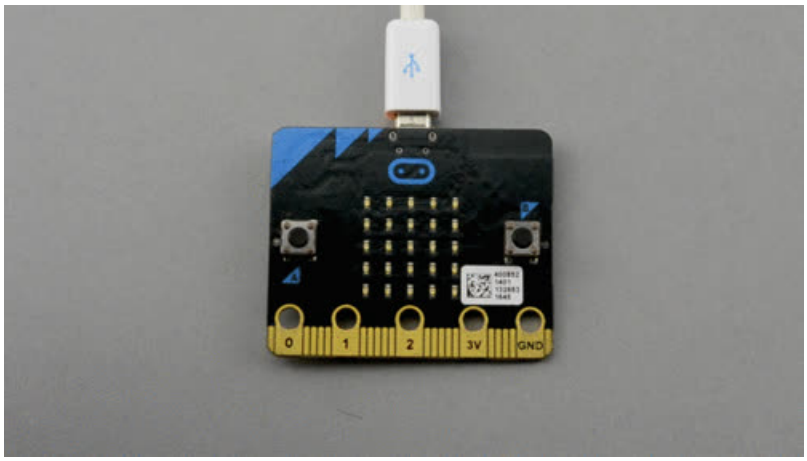
adafruit learning system

micro:bit Lesson 1. Using the Built-in Sensors

Created by Simon Monk



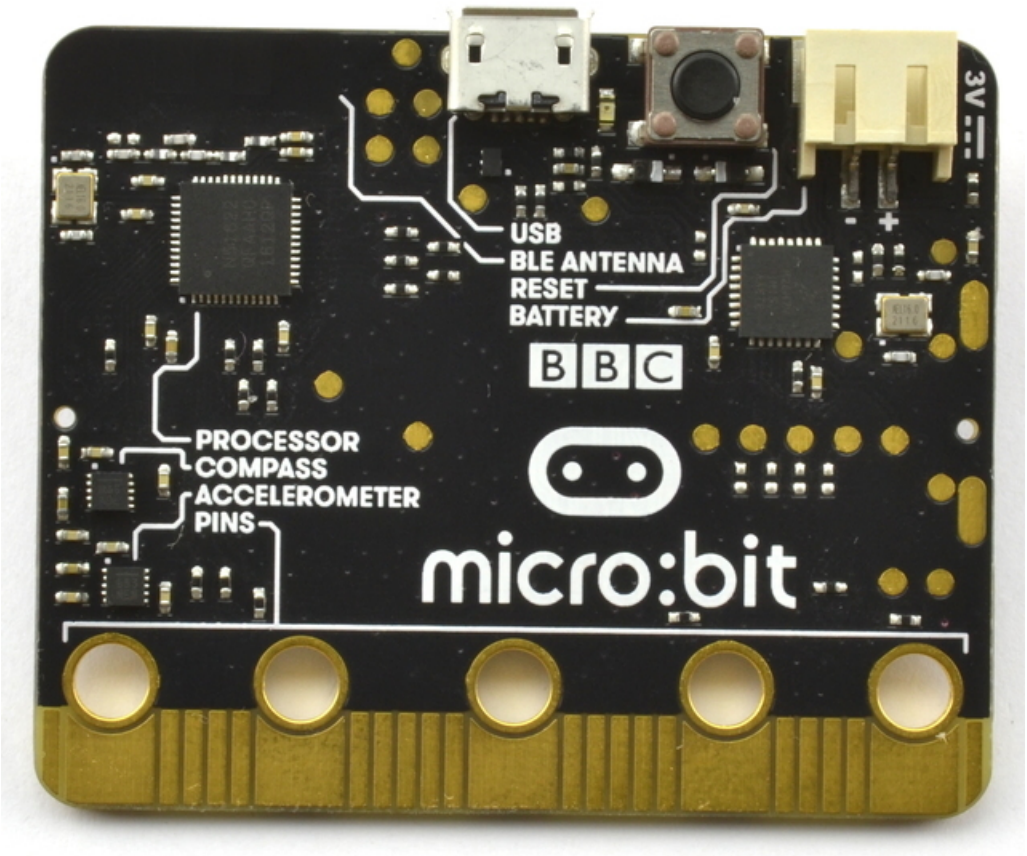
Last updated on 2018-08-22 04:05:20 PM UTC

Guide Contents

Overview

The BBC micro:bit has a number of built-in sensors that make it really easy to get started with a few simple projects.

In this lesson you will learn how to use the micro:bit's accelerometer to make a simple movement activated alarm. You will also learn how to use the micro:bit's magnetometer chip to detect the presence of a magnet.



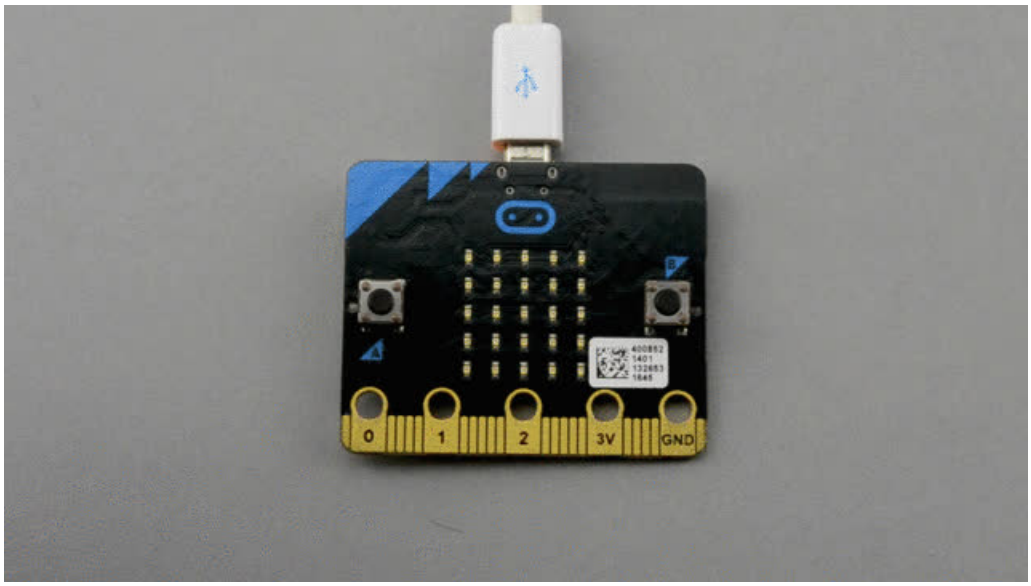
If you flip your micro:bit over, you can see both the accelerometer and magnetometer (compass) marked on the bottom left of the photograph above.

Magnetometer

The micro:bit's built-in magnetometer chip is intended for use as compass to detect magnetic north. Like the compass app on your phone, this requires calibration and in my experience doesn't work *very* reliably. However, it is great for detecting the presence of a magnet!

Magnet Detector

The following project will make the micro:bit's display show a cross when a magnet comes near. This works best with a strong 'neodymium' magnet like the one in the side panel.



When you are looking at the top of the micro:bit, the compass chip is more or less behind button B.

Note that I have had problems with this project using battery power from the battery connector. However, its fine powered from a USB lead.

JavaScript Block Code

When the program starts the micro:bit will automatically take you through a calibration process for the compass chip. This involves tilting the micro:bit around in a circular motion in order to light up a ring of LEDs on the display.

The JavaScript Blocks Code editor is embedded directly on this page below. From the editor, you can click on Download button (bottom right) and then copy the downloaded file onto your micro:bit. Alternatively, you can Click here (https://adafru.it/Cj4) to open the editor in a separate browser tab.

In this program the block *magnetic force* is used to measure the intensity of the magnetic field near the compass chip on your micro:bit. This is used in the `on start` block to establish a `baseline` reading of magnetic strength with no magnet near the micro:bit. This happens when the micro:bit is first powered up or the reset button is pressed.

The `forever` loop repeatedly takes readings and checks to see if the absolute value of the difference between the new reading and the baseline reading is above 100 and if it is it displays the cross symbol on the display. The `absolute` value is just the value with the sign removed. This is necessary because, depending on which pole of the magnet is pointing towards the compass chip, the `magnetic force` reading may decrease rather than increase.

You may need to increase the threshold of 100 if the cross appears even without a magnet being near.

MicroPython

To run the MicroPython version of the code, open up the online Python editor here (https://adafru.it/CiK) and paste the following code into the editor window.

```
from microbit import *

baseline = compass.get_field_strength() # Take a baseline reading of magnetic strength

while True:
  if abs(compass.get_field_strength() - baseline) > 10000:
      # Magnetic field strength increased by 10000
      display.show(Image.NO)    # Show a cross symbol
  else:
      display.clear()
```

The program first `import`s the `microbit` library that contains the `compass` and `display` objects needed in the program.

The `while` loop works just like its JavaScript Blocks sibling. It repeatedly takes readings and checks to see if the absolute value of the difference between the new reading and the `baseline` reading is above 10000 and if it is it displays the cross symbol on the display. The `abs` value is just the value with the sign removed. This is necessary because, depending on which pole of the magnet is pointing towards the compass chip, the *magnetic force* reading may decrease rather than increase.

You may need to increase the threshold of 10000 if the cross appears even without a magnet being near.

Arduino

Make sure that you have your Arduino environment set up for micro:bit by following this guide (https://adafru.it/CiL).

Now start a new Sketch by clicking on the *File* menu and *New*. Then paste the following code into the editor window.

```
#include "Wire.h"
#include <SparkFun_MAG3110.h>
#include <Adafruit_Microbit.h>

Adafruit_Microbit_Matrix microbit;
MAG3110 compass = MAG3110();  // The compass chip
int baseline = 0;

void setup() {
  microbit.begin();
  compass.initialize();       // Initializes the compass chip
  compass.start();            // Puts the sensor in active mode
  baseline = readStrength();  // Take a baseline reading of magnetic strength
  delay(500);

}

void loop() {
  if (abs(readStrength() - baseline) > 15000) {
    // Magnetic field strength increased by 15000
    microbit.show(microbit.NO);
  }
  else {
    microbit.clear();
  }
}

int readStrength() {
  int x, y, z;
  while(!compass.dataReady()) {};   // Wait for data to become available to read
  compass.readMag(&x, &y, &z);      // Read data into variables
  // calculate the RMS power combining x, y and z readings
  int power = sqrt((sq(float(x)) + sq(float(y)) + sq(float(z))) / 3);
  return power;
}
```

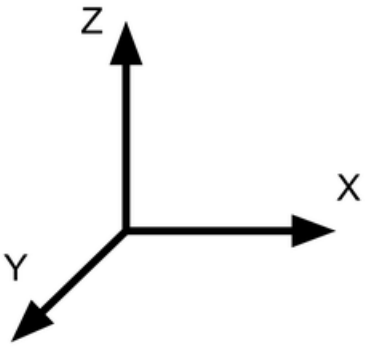
If you get an error such as: fatal error: ble_gatts.h: No such file or directory when you go to upload, then make sure you have selected a Soft Device of S110 from the Arduino Tools menu.

The Arduino version of the compass library does not include an equivalent to the *magnetic force* block of the Blocks editor or the `get_field_strength` of MicroPython. This means that we have to write our own, called `readStrength`. This takes the x, y and z readings from the compass chip and combines them using a mathematical technique called RMS (https://adafru.it/Bf5) (root mean square). That takes a kind of average of the readings.

You may need to increase the threshold of 15000 if the cross appears even without a magnet being near.

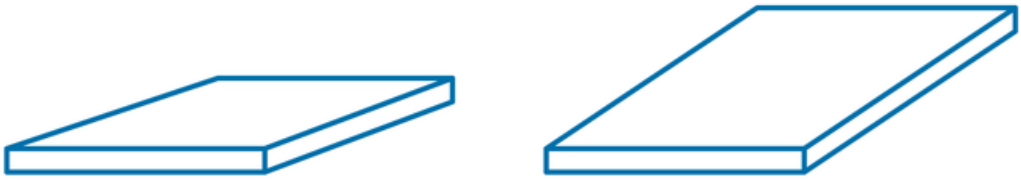
Accelerometer

An accelerometer chip measures the forces acting on a tiny weight in each of three dimensions. If you imagine the micro:bit sitting flat on a desk, the x dimension is left to right; the y dimension front to back and the z dimension into and out of the desk.



Flat
x = 0
y = 0
z = -1000

Back tipped up
x = 0
y = 400
z = -700



If the micro:bit is perfectly flat, the forces on the x and y dimensions will be zero. The force acting on the z axis will have a value (when read in your JavaScript Blocks or Python code) of about -1000 due to the pull of gravity. If you then tilt the back of the micro:bit up then the force in the y dimension will increase a little.

Its these changes that allow you to use an accelerometer to detect the orientation of the micro:bit, since gravity will always be acting in the same downwards direction.

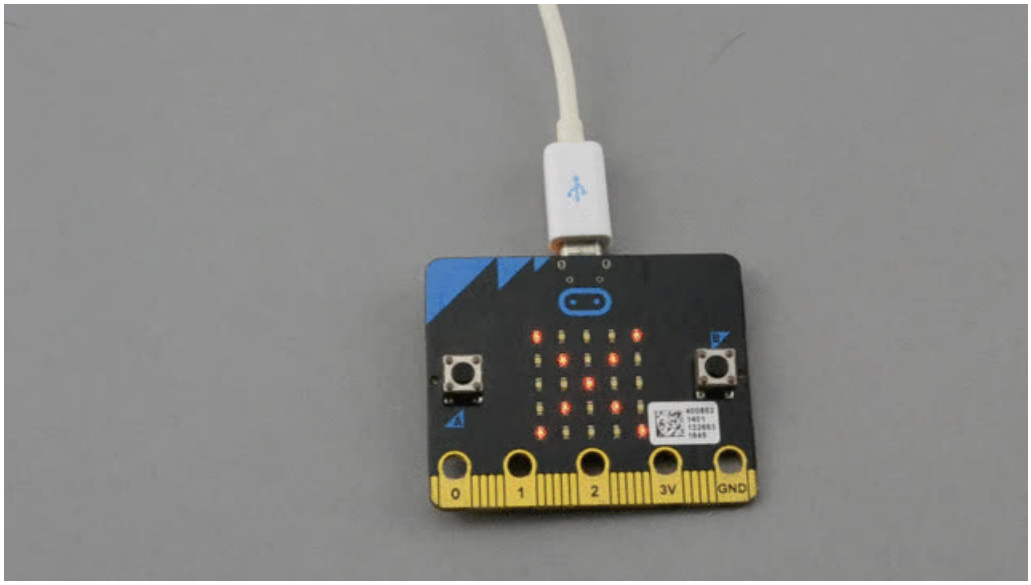
Movement Alarm Example

As an example of using the accelerometer lets make a sensitive movement alarm from out micro:bit.

The project works like this:

When you press button A, the display changes top show a triangle, indicating that the alarm will arm itself after two seconds. Once armed, the display will be blank.

If you now try and move the micro:bit then the LEDs will all light indicating that the alarm has been triggered. Put the micro:bit down, so that it's still and then press button A to arm it again.



JavaScript Block Code

Here is the JavaScript Block version of the alarm program. The JavaScript Blocks Code editor is embedded directly on this page below. From the editor, you can click on Download button (bottom right) and then copy the downloaded file onto your micro:bit. Alternatively, you can Click here (https://adafru.it/Cj6) to open the editor in a separate browser tab.

The code works like this:

The *on start* block will be run the first time the program runs after the micro:bit has been powered-up or reset. This sets a variable `z` (the acceleration in the z axis) to zero. It also sets a Boolean (true or false) variable (`alarm triggered`) to false. The variable `alarm triggered` will be used to indicate to other parts of the program that the alarm has been triggered.

The `forever` loop will run over and over again and is in three parts, each part being an `if` / `then` block.

The first `if` / `then` block checks for a press of button A, and if button A is pressed, it displays a triangle symbol, waits two seconds (2000 milliseconds) then reads the acceleration in the z axis into the variable `z`. Finally it clears the screen.

The second if/then in the `forever` block checks the current acceleration in the z axis against the value stored in the variable `z` and if it is less than the stored reading by a margin of 30, it sets the variable `alarm triggered` to be true.

The final part of the code determines what happens when the `alarm triggered` variable is true. If the alarm is triggered, then the *cross* symbol will be shown on the display otherwise the display will be cleared. The `alarm triggered` variable will remain true until button A is pressed again.

MicroPython

Here is the MicroPython version of the code for the movement alarm.

```
from microbit import *

z = 0
alarm_triggered = False

while True:
    if button_a.was_pressed():
        # Wait for 2 seconds while the alarm is armed
        display.show(Image.TRIANGLE)
        sleep(2000)
        z = accelerometer.get_z() # take a reading while still
        alarm_triggered = False   # the alarm is now armed
    # Check to see if acceleration on z axis changed by 30
    if accelerometer.get_z() < z - 30:
        alarm_triggered = True
    if alarm_triggered:
        display.show(Image.NO)    # display a cross
    else:
        display.clear()
```

The code is almost a line by line copy of the JavaScript Blocks code, but translated into Python.

Arduino

The Arduino version of the program is listed below and contains a few differences from the other two versions of the code.

```
#include "Wire.h"
#include "MMA8653.h"
#include <Adafruit_Microbit.h>

const int buttonA = 5;     // the number of the pushbutton pin

const uint8_t triangle[] = {
  B00000,
  B00100,
  B01010,
  B11111,
  B00000
};

Adafruit_Microbit_Matrix microbit;
MMA8653 accelerometer;
int z = 0;
boolean alarmTriggered = false;

void setup() {
  pinMode(buttonA, INPUT);
  microbit.begin();              // start the display
  accelerometer.begin(false, 2); // 8-bit mode, 2g range
}

void loop() {
  accelerometer.update();
  if (digitalRead(buttonA) == LOW) {
    // Wait for 2 seconds while the alarm is armed
    microbit.clear();
    microbit.show(triangle);
    delay(2000);
    z = accelerometer.getZ();         // take a reading while still
    alarmTriggered = false;           // the alarm is now armed
  }
  // Check to see if acceleration on z axis changed by 5
  if (accelerometer.getZ() < z - 5) {
    alarmTriggered = true;
  }
  if (alarmTriggered) {
    microbit.clear();
    microbit.show(microbit.NO);  // display a cross
  }
  else {
    microbit.clear();
  }
}
```

The main thing about the Arduino version of the code is that there are no predefined triangle symbol to use with the display and so you have to define this yourself as the `triangle` byte array.

Each byte array is an array of 5 bytes each of eight bits. Each bit can be either on or off (1 or 0). If the bit is 1 then the LED in that position will be lit otherwise it won't. If you look carefully at the byte arrays triangle and cross, you can just make out the bit pattern.

The `setup` function (like the on start block in JavaScript Blocks) is only run when the micro:bit starts or is reset. In this case the button A pin has to be defined as an input explicitly. The `microbit.begin` method starts the refreshing of the

micro:bit's screen.

The `loop` function will be run repeatedly and does actually work just like the JavaScript Blocks and MicroPython code.

Other Things to Do

You might like to experiment with the code for the projects in this lesson. You could for example make both the magnet detector and accelerometer project more sensitive. You could change the magnet detector project so that it counted the number of times a magnet was brought close to it. This technique of a magnet and magnetic sensor can be used to count the rotations of a wheel.

About the Author

If you have found this lesson useful, and want to learn Python, then you might like my book Programming the BBC micro:bit: Getting Started with MicroPython.