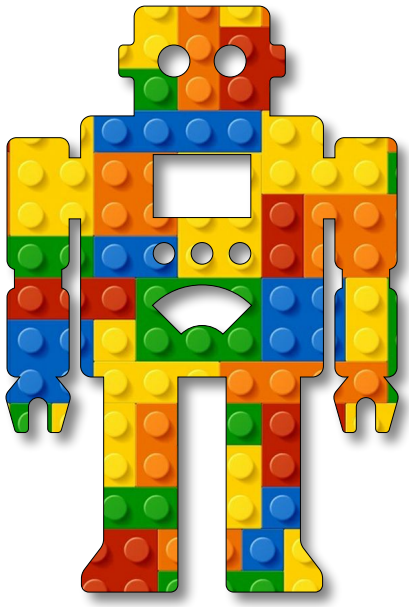
Intelligent Machines Module A Unit #10 accelerometer



Content

Module A Unit #10 Accelerometer

Introduction to the motion sensor

Utilising the built-in BMI270 and BMM150 sensor module (IMU)

IMU stands for: inertial measurement unit. It is an electronic device that measures and reports a body's specific force, angular rate, and the orientation of the body, using a combination of accelerometers, gyroscopes, and oftentimes magnetometers.

The IMU system on the Nano 33 BLE Rev2 is a combination of two modules, the 6-axis BMI270, and the 3-axis BMM150, that together add up to a combined 9-axis IMU system that can measure acceleration, as well as rotation and magnetic fields all in 3D space.

The Arduino BMI270_BMM150 library allows us to use the Nano 33 BLE Rev2 IMU system without having to go into complicated programming. The library takes care of the sensor initialisation and sets its values as follows:

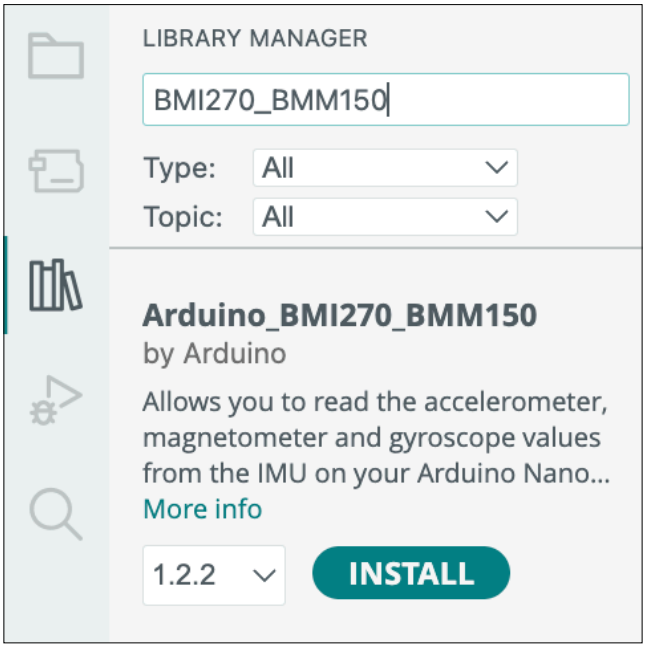
- accelerometer: the range is set at [-4, +4]g +/-0.122 mg and the output data rate is fixed at 104 Hz.

- gyroscope: the range is set at [-2000, +2000] dps +/-70 mdps and the output data rate is fixed at 104 Hz.

- magnetometer: the range is set at [-400, +400] uT +/-0.014 uT and the output data rate is fixed at 20 Hz.

To use the library you will need to install the library by first clicking on the button on the left-hand side that looks like a pile of books (hint: `library`), and then type in the name of the sensor, find it, and click on INSTALL as shown in Fig. 1. You may need to do this for all the libraries needed.

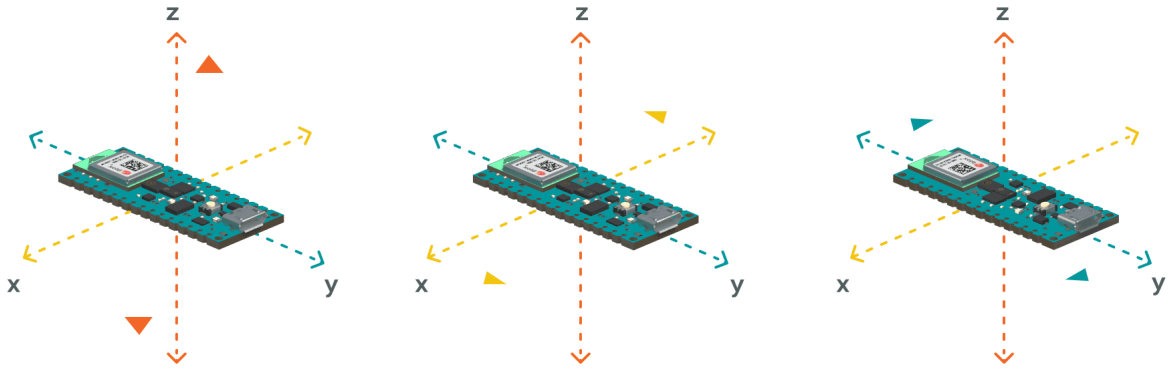
Figure 1: library manager



The Accelerometer

In this unit we will be looking at the accelerometer. An accelerometer is an electromechanical device used to measure acceleration forces. Such forces may be static, like the continuous force of gravity, or, as is the case with many mobile devices, dynamic to sense movement or vibrations.

Figure 2: the accelerometer



In this example, we will use the accelerometer as a level that will provide information about the position of the board. With this application, we will be able to read what the relative position of the board is as well as the degrees, by tilting the board up, down, left, or right.

Sketch A10.1 accelerometer

This will simply print out the raw data for the x, y, and z axes; just move the microcontroller around. Don't forget to press the RESET button.

Arduino sketch

```
#include "Arduino_BMI270_BMM150.h"

void setup()
{
  Serial.begin(9600);
  while (!Serial);
  if (!IMU.begin())
  {
    Serial.println("Failed to initialize IMU!");
    while (1);
  }
  Serial.print("Accelerometer sample rate = ");
  Serial.print(IMU.accelerationSampleRate());
  Serial.println(" Hz");
  Serial.println();
  Serial.println("Acceleration in G's");
  Serial.println("X\tY\tZ");
}

void loop()
{
  float x, y, z;
  if (IMU.accelerationAvailable())
  {
    IMU.readAcceleration(x, y, z);
    Serial.print(x);
    Serial.print('\t');
    Serial.print(y);
    Serial.print('\t');
    Serial.println(z);
  }
}
```

🗒️ Notes

We can see the results if you select the `serial monitor`.

🌻 Challenge

Try the serial plotter if you can find it.

🛠️ Code Explanation

`#include "Arduino_BMI270_BMM150.h"`	This is the library which has to have been already installed
`while (!Serial);`	Just waits for the serial port to open
`if (!IMU.begin())`	Checks to see if it hasn't started
`while (1);`	This creates a continuous loop until the IMU has kicked in
`if (IMU.accelerationAvailable())`	Checks to see if the data is available
`IMU.readAcceleration(x, y, z);`	Reads the data for x, y and z axis
`Serial.print('\t');`	Inserts a tab space

Figure 3: The serial monitor is the one on the right of the two icons

Figure A10.1: the Serial Monitor

Sketch A10.2 accelerometer tilting text

This will give a text indication of the movement tilting up/down/left/right and the degree of tilt.

Arduino sketch

```
#include "Arduino_BMI270_BMM150.h"

float x;
float y;
float z;
int degreesX = 0;
int degreesY = 0;

void setup()
{
  Serial.begin(9600);
  while (!Serial);
  Serial.println("Started");
  if (!IMU.begin())
  {
    Serial.println("Failed to initialize IMU!");
    while (1);
  }

  Serial.print("Accelerometer sample rate = ");
  Serial.print(IMU.accelerationSampleRate());
  Serial.println("Hz");
}

void loop()
{
  if (IMU.accelerationAvailable())
  {
    IMU.readAcceleration(x, y, z);
  }

  if (x > 0.1)
```

```
  {
    x = 100 * x;
    degreesX = map(x, 0, 97, 0, 90);
    Serial.print("Tilting up ");
    Serial.print(degreesX);
    Serial.println("  degrees");
  }

  if (x < -0.1)
  {
    x = 100 * x;
    degreesX = map(x, 0, -100, 0, 90);
    Serial.print("Tilting down ");
    Serial.print(degreesX);
    Serial.println("  degrees");
  }

  if (y > 0.1)
  {
    y = 100 * y;
    degreesY = map(y, 0, 97, 0, 90);
    Serial.print("Tilting left ");
    Serial.print(degreesY);
    Serial.println("  degrees");
  }

  if (y < -0.1)
  {
    y = 100 * y;
    degreesY = map(y, 0, -100, 0, 90);
    Serial.print("Tilting right ");
    Serial.print(degreesY);
    Serial.println("  degrees");
  }
  delay(1000);
}
```

🗒️ Notes

The above is a nice, simple example. The next sketch is an example taken from the documentation.

🛠️ Code Explanation

`while (1);`	This is a continuous loop the library hasn't started
`Serial.print(IMU.accelerationSampleRate());`	The sample rate is measured in Hz
`x = 100 * x;`	Multiply the x value by 100
`degreesX = map(x, 0, 97, 0, 90);`	Maps the value of x from between 0-97 to 0-90 when tilted to the left

Figure A10.2

```
Tilting up 90  degrees
Tilting up 88  degrees
Tilting up 92  degrees
Tilting up 46  degrees
Tilting right 17  degrees
Tilting down 20  degrees
Tilting down 60  degrees
Tilting left 31  degrees
Tilting down 59  degrees
Tilting left 31  degrees
Tilting down 61  degrees
Tilting left 31  degrees
Tilting down 62  degrees
Tilting left 30  degrees
Tilting down 61  degrees
Tilting left 31  degrees
```

Sketch A10.3 a more complex version

This is from another example provided; it does the same thing, roughly, although it looks a little more complex compared to the previous sketch.

Arduino sketch

```
#include "Arduino_BMI270_BMM150.h"

#define MINIMUM_TILT 5
#define WAIT_TIME 500

float x, y, z;
int angleX = 0;
int angleY = 0;
unsigned long previousMillis = 0;

void setup()
{
  Serial.begin(9600);
  while (!Serial);
  if (!IMU.begin())
  {
    Serial.println("Failed to initialise IMU!");
    while (1);
  }

  Serial.print("Accelerometer sample rate = ");
  Serial.print(IMU.accelerationSampleRate());
  Serial.println("Hz");
}

void loop()
{
  if (IMU.accelerationAvailable() && millis() - previousMillis >= WAIT_TIME)
  {
    previousMillis = millis();
    IMU.readAcceleration(x, y, z);
    angleX = atan2(x, sqrt(y * y + z * z)) * 180 / PI;
```

```
    angleY = atan2(y, sqrt(x * x + z * z)) * 180 / PI;
    if (angleX > MINIMUM_TILT)
    {
      Serial.print("Tilting up ");
      Serial.print(angleX);
      Serial.println(" degrees");
    }
    else if (angleX < -MINIMUM_TILT)
    {
      Serial.print("Tilting down ");
      Serial.print(-angleX);
      Serial.println(" degrees");
    }
    if (angleY > MINIMUM_TILT)
    {
      Serial.print("Tilting right ");
      Serial.print(angleY);
      Serial.println(" degrees");
    }
    else if (angleY < -MINIMUM_TILT)
    {
      Serial.print("Tilting left ");
      Serial.print(-angleY);
      Serial.println(" degrees");
    }
  }
}
```

🗒️ Notes

Just plough through the code. Not the place to go into detail here.

🌻 Challenge

Recommend researching atan2 and how the angle is calculated.

🛠️ Code Explanation

`#define MINIMUM_TILT 5`	Threshold for tilt detection in degrees
`#define WAIT_TIME 500`	How often to run the code (in milliseconds)
`angleX = atan2(x, sqrt(y * y + z * z)) * 180 / PI;`	Calculates the tilt angles in degrees