Inertial Measurement Unit Simulation Test

Miguel Tamayo

August 11th, 2022

Contents

List of Figures

1 Project Overview

An inertial measurement unit (IMU) is a device usually containing an accelerometer, magnetometer, and gyroscope. These elements obtain the acceleration, magnetic field around the device, and angular rate. By combining this information, we can determine the orientation of the device which it is attached to.

This project focuses on understanding how to use the Adafruit BNO055 9 degrees of freedom sensor through a series of actions. First, the sensor must be calibrated in order to get accurate data. After calibration, the orientation of the device will be displayed and used to rotate an object in a simulation.

The outcome expected of this project is to successfully send the data from the BNO055 sensor through serial communication to a simulator with a space shuttle which will rotate based on the orientation of the sensor.

The components needed for this project are:

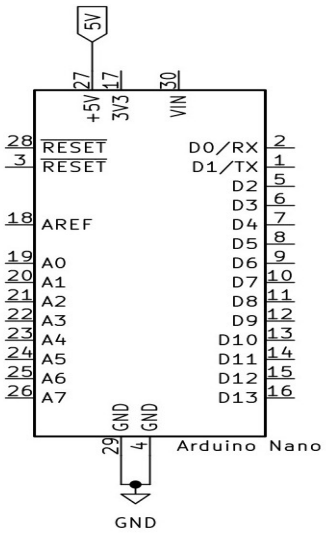
- Breadboard
- Arduino Nano
- Adafruit BNO055 Sensor
- Wires

The software products needed for this project are:

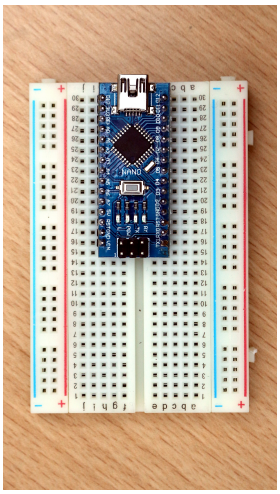
- Arduino IDE
- Processing
- kiCad

2 Hardware Design

2.1 Micro Controller (Arduino Nano)



(a) Arduino Nano Schematic



(b) Arduino Nano Size

Figure 1: Arduino Nano Characteristics

A microcontroller is a small computer designed to do specific tasks of embedded systems such as controlling motors, LEDs, receiving, and sending signals. Arduino has a variety of microcontrollers but the Nano was chosen due to its compact shape which allows the project to be built on a single breadboard (Figure 1).

The Arduino Nano acts as the main computer of the timer. It monitors the signals coming from the buttons and displays the correct number in seconds on the display. The microcontroller also manages the state machine (see Software Design) and changes states according to the signals from buttons and an internal timer.

2.2 Adafruit BNO055 IMU Sensor

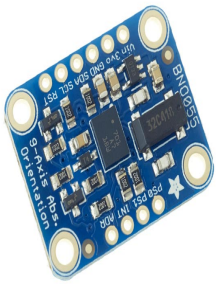


Figure 2: BNO055 Module

The BNO055 sensor is a 9-DOF which uses a MEMS accelerometer, magnetometer, and gyroscope and combines the data into useful readings in Euler angles, Quaternions, and vectors. The BNO055 can be used for many applications such as a compass and a level. This project takes advantage of the Euler total orientation data to orient a space shuttle in a simulation (see software design).

The BNO055 communicates with the Arduino via the I^2C. First, connect the Serial Clock (SCL) pin of the Arduino and sensor together. The SCL pin is a clock which oscillates allowing the receiver to know when it should read bits from the data pin. The next connection is the Serial Data Pin (SDA) which is how the actual data is sent between the devices. In the Arduino Nano, the SCL pin is analog pin A5 and the SDA pin is analog pin A4. To finish the connection between the Arduino and sensor, connect the $3.3V$ and GND pins from the Arduino to the V_{in} and GND pins on the BNO055 sensor respectively. The connections are shown in Figure 3 below.

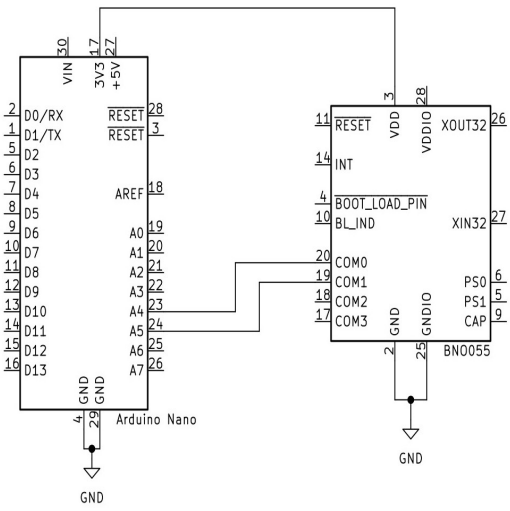


Figure 3: BNO055 Connections

3 Software Design

3.1 Sensor Calibration

Before using the sensor, it must be calibrated to get accurate data. The calibration process is done by taking data and comparing it to expected values in order to get an offset that is subtracted factored from reading to get data that is as error free as possible. This process can be difficult as there are many ways to calibrate a sensor. However, an accurate and reliable algorithms can be hard to find.

Luckily, Adafruit has taken care of this process through their BNO055 library. A calibration script has been already provided and can be used to calibrate the sensor. The only change that has to be made to the code is changing the ID of the device. This allows the Arduino to store the calibration data in the EEPROM for it to be used again in the future.

There are three steps to fully calibrate the sensor. To calibrate the gyroscope, the sensor is left untouched for a couple of seconds. Next, waving the sensor in the air will calibrate the magnetometer. Lastly, to calibrate the accelerometer each axis need to go through positive and negative positions. This is done by making each axis face up and down since acceleration is a vector that points straight down. The script shows signals for each sensor that indicate the level of calibration which goes from 0 (not calibrated) to 3 (calibrated). Once all sensors are calibrated, the biases and scaling factors will be saved for use.

3.2 Relative Orientation

Now that the sensor has been fully calibrated, data can be read accurately. To do this, the sensor is initialized in the IMU setting. This setting zeroes out the axis on start-up allowing the readings to be taken in reference to this point. After loading the calibration parameters, the script read the Euler vector from the sensor which outputs data for each axis (x, y, z). This data is then printed to the serial monitor in order to be read by the simulation software.

3.3 Simulation

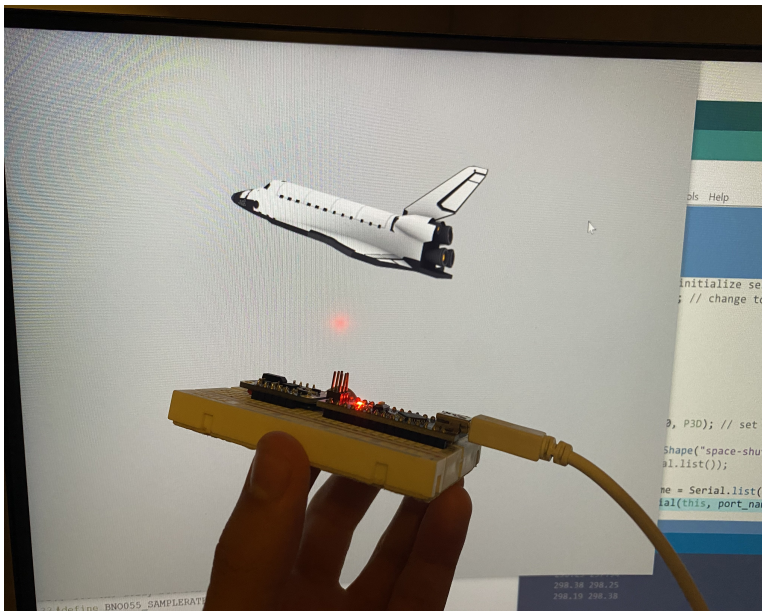


Figure 4: BNO055 Simulation

It can be difficult to understand the data coming from the sensor by just printing the numbers to a screen. For this reason a small simulation (Figure 4) was designed to visually demonstrate how the sensor works. To receive the attitude readings from the Arduino, the simulation script also opens the correct serial port and extracts the angular orientation from the string received. These values which correspond to the angle along the axis are then used to rotate the object in the simulation.

The object used in the simulation is a 3D model of a Space Shuttle found online. The same orientation script is used to render the space shuttle on the screen by loading the *.obj* file and using special functions to draw out the vectors that make up the object. The link to the object can be found on the third link in the references sections.

4 Results

The final build of the IMU testing platform is shown in Figure 5 below:

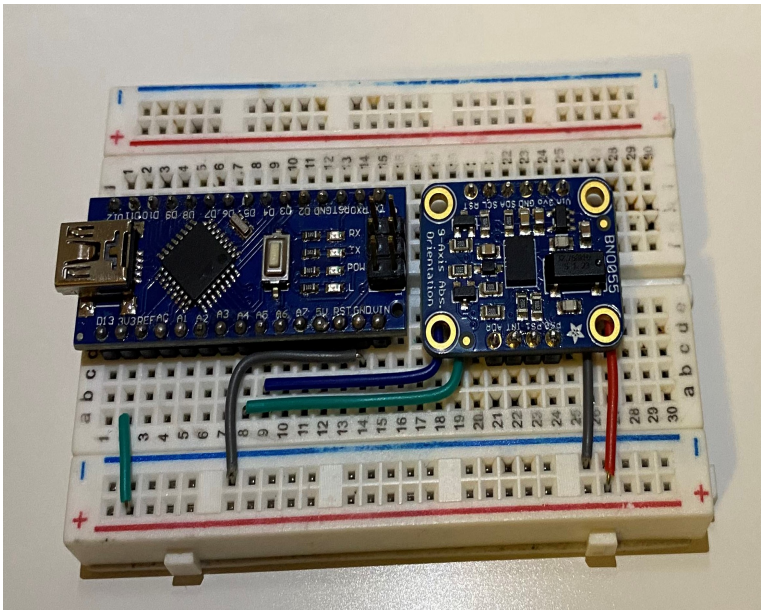


Figure 5: IMU Orientation Final Build

A video showing how this timer works can be found in the YouTube video linked here. Please use the link below if the hyperlink above does not work.

- https://youtube.com/shorts/wP1HaXMB2-M?feature=share

4.1 Discussion

4.1.1 Assumptions and Errors

It was assumed that in the simulation the rotations were being applied to the space shuttle object. However, this assumption was incorrect. After more research was done on the `rotate()` function, it was discovered that the rotation is applied to the screen before loading the object giving the illusion that the object is rotating. Because of this, when the simulation reads a yaw input of 90° degrees the object will turn sideways as expected. In this position is the simulation receives a pitch input, the object appears to roll because the pitch is actually being applied to the screen frame.

The use of Euler angles to represent the orientation of the object raised the issue of gimbal locking. Gimbal lock is the loss of one degree of freedom in a three-dimensional system. In this project, gimbal lock can be seen when the pitch angle reaches 90° degrees and the object cannot move any further. What happens in this case is that the entire frame rotates 180° degrees to compensate for this and the pitch angle becomes negative. To solve this issue, Quaternions are often used as they have an extra characteristic that represents orientation. Quaternions are often used to get a better representation of orientation in a 3D space while solving the gimbal lock problem.

4.1.2 Future Improvements

To make the simulation follow the orientation of the IMU exactly, the inputs from the sensor and the `rotation()` function can be manipulated such that the rotation is applied to the object and not the window frame. Additionally, using position estimation from the IMU, the object can also be moved in the virtual space depending on the location of the IMU device in real life.

5 Conclusion

The purpose of this project is to learn how to obtain data from an Inertial Measurement Unit module and understand how it can be used in real life. This was accomplished by calibrating and sending the attitude data from the module through Serial communication to a virtual simulator. The simulations of a space shuttle allowed us to visualize the effects of moving the IMU and how the values translate to rotations in 3D space.

Additionally, this project shed light to the well known gimbal lock problem which can be solved through the use of Quaternions instead of Euler angles. There were issues when a rotation function was misinterpreted making certain rotations look incorrect based on the real life movements. This can be further solved by creating an algorithm to apply the rotations to the object in the simulation and not the screen frames.

6 References

1. What is an IMU
   https://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu

2. BNO055 Connections
   https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/arduino-code

3. Space Shuttle Object Files
   https://sketchfab.com/3d-models/space-shuttle-ff4b00b7ebb24fdd98fb96b08f2c43c9

4. What is Gimbal lock
   https://en.wikipedia.org/wiki/Gimbal_lock

5. BNO055 Image
   https://components101.com/sensors/bno055-sensor-pinout-datasheet-equivalent-configuration-specs