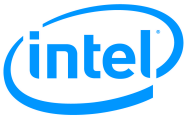
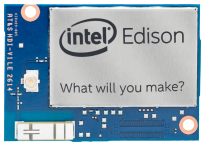
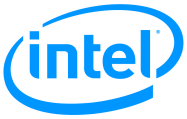




Intel® Edison Tutorial: 9DOF IMU SparkFun Kit



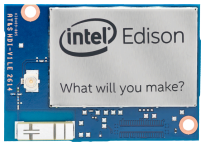
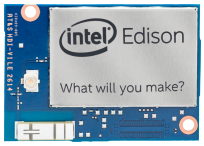


Table of Contents



Introduction

In this tutorial, users will:

1. Assemble the 9DOF IMU SparkFun kit.
2. Write C code to acquire data generated by the 9DOF IMU SparkFun kit.

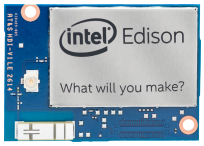
Prerequisite Tutorials

Users should ensure they are familiar with the documents listed below before proceeding.

1. Intel Edison Tutorial – Introduction, Linux Operating System Shell Access, and SFTP
2. Intel Edison Tutorial – Introduction to Linux
3. Intel Edison Tutorial – Introduction to Vim

List of Required Materials and Equipment

1. 1x Intel Edison Compute Module.
2. 1x 9DOF IMU SparkFun Kit:
    1. 1x Base Block https://www.sparkfun.com/products/13045
    2. 1x 9DOF IMU https://www.sparkfun.com/products/13033
    3. 1x Battery Block https://www.sparkfun.com/products/13037
    4. 1x Hardware Pack https://www.sparkfun.com/products/13187
3. 2x USB 2.0 A-Male to Micro B Cable (micro USB cable).
4. 1x Roll of Electrical Tape.
5. 1x Personal Computer.
6. 1x Wi-Fi network with access to the Internet.



9DOF IMU SparkFun Kit

The **9 D**egrees **o**f **F**reedom (9DOF) SparkFun Kit for the Intel Edison uses the LSM9DS0 9DOF IMU for full-range motion sensing. This system-in-package features a 3D digital linear acceleration sensor (accelerometer), 3D digital angular rate sensor (gyroscope), and a 3D digital magnetic sensor (magnetometer).

Refer to the figures shown below to understand the data collected by the 9DOF IMU.

Some of the potential 9DOF projects include:

- Head tracking for virtual reality (gyroscope).
- Fall detection (accelerometer).
- Athletic performance tracker (accelerometer and gyroscope).
- Indoor localization (accelerometer and gyroscope).

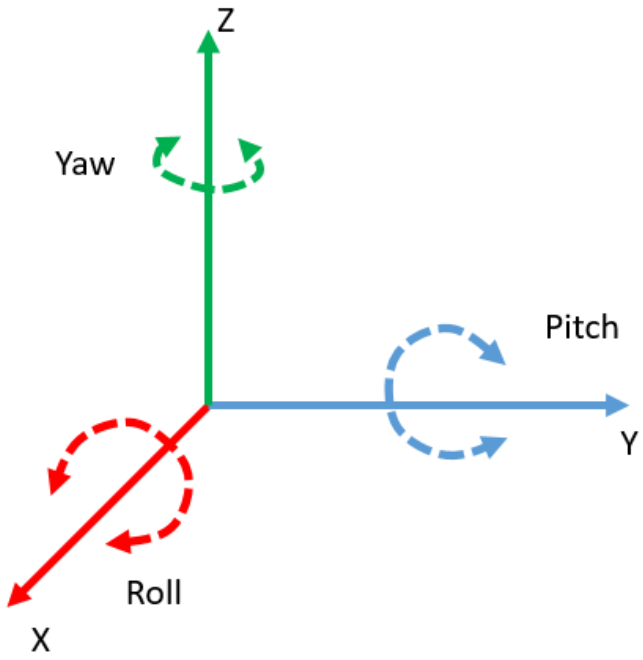


Figure 1: Visualization of data provided by gyroscope

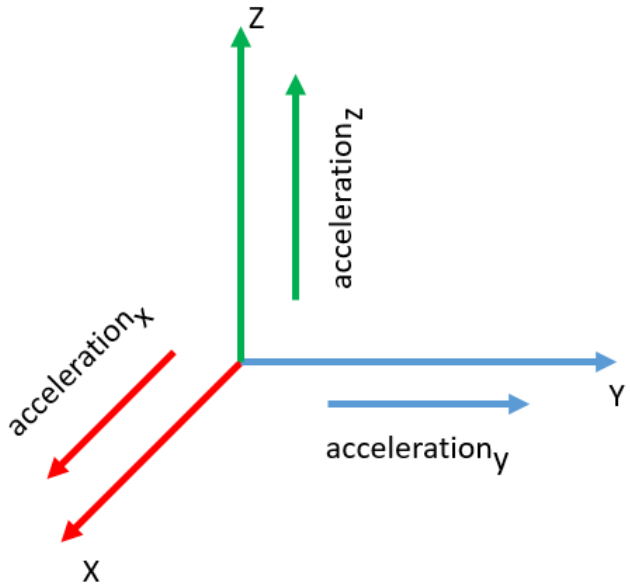


Figure 2: Visualization of data provided by accelerometer

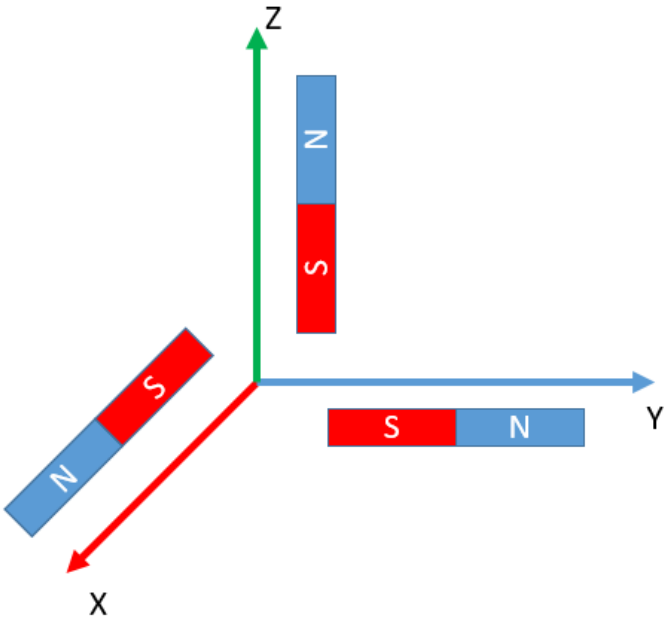


Figure 3: Visualization of data provided by magnetometer

Hardware Assembly

Users should check the list of items below to ensure they have the correct list of materials before proceeding with the tutorial.

1. Before performing any tasks, power down the Intel Edison device. Users must also unplug all USB and power cables from the Intel Edison.
2. Base Block.

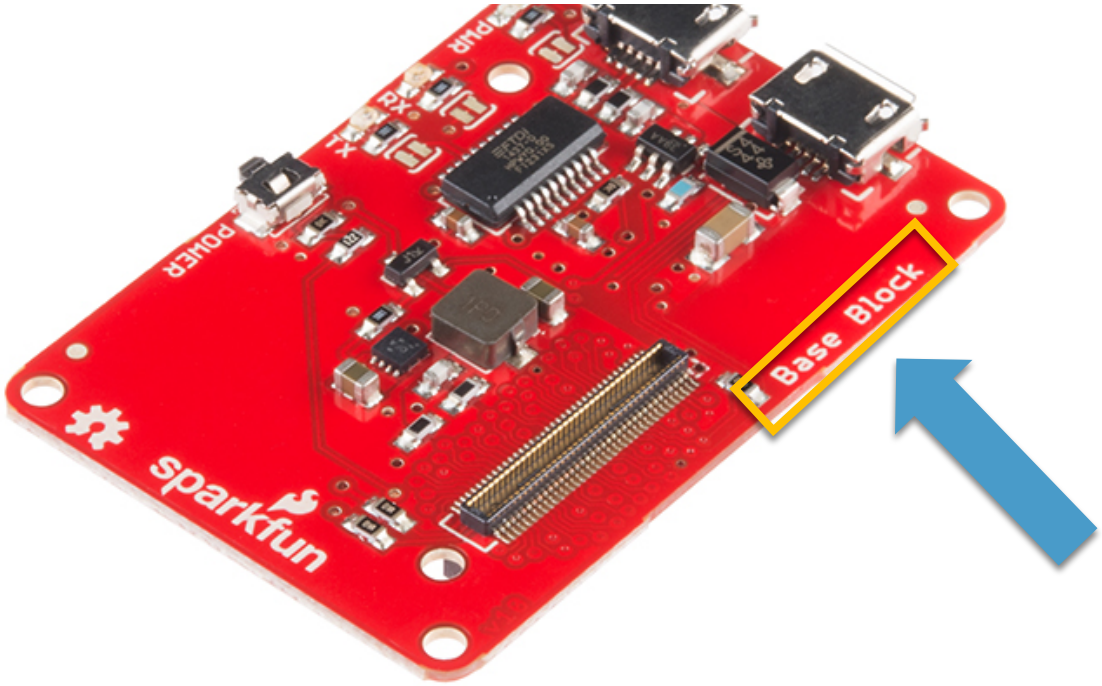


Figure 4: Base Block, required to access the shell on the Intel Edison Compute Module using a USB cable

3. 9DOF Block.

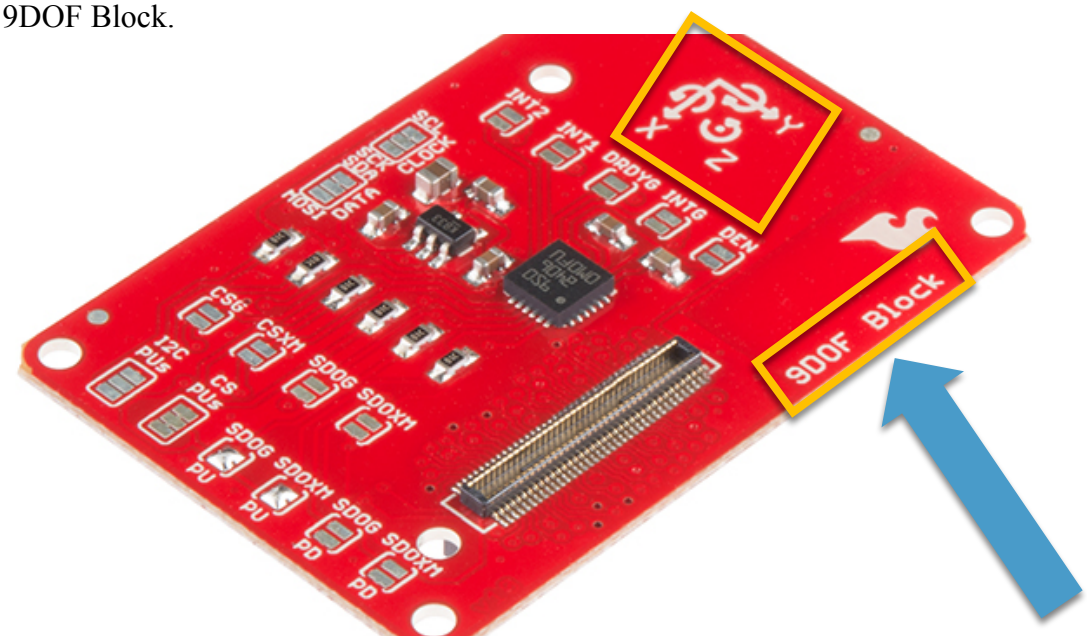


Figure 5: The 9DOF Block hosts accelerometer, gyroscope and magnetometer

4. Battery Block.

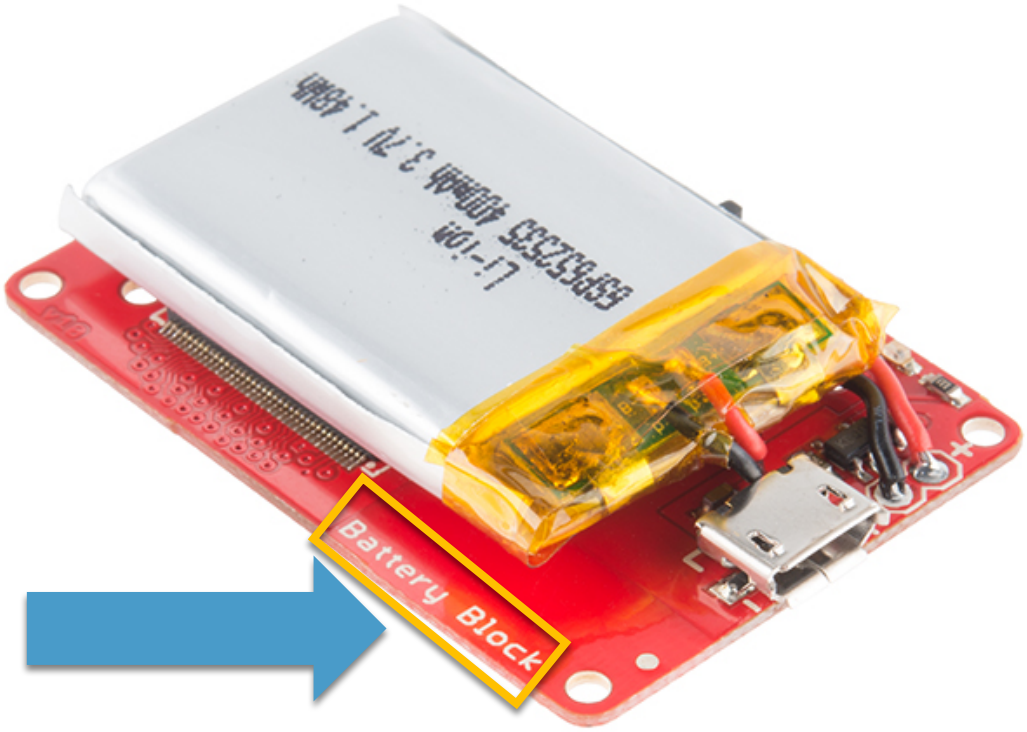


Figure 6: Battery Block, provides energy to 9DOF IMU SparkFun Kit

5. Hardware Pack.

Figure 7: Hardware Pack, use these components to ensure mechanical stability 9DOF IMU SparkFun Kit

6. There are two exposed metal contacts on the Battery Block. Users should ensure these contacts are electrically insulated from each other to prevent issues such as short circuits. Cut a strip of electrical tape and place it on top of the contacts as described below.

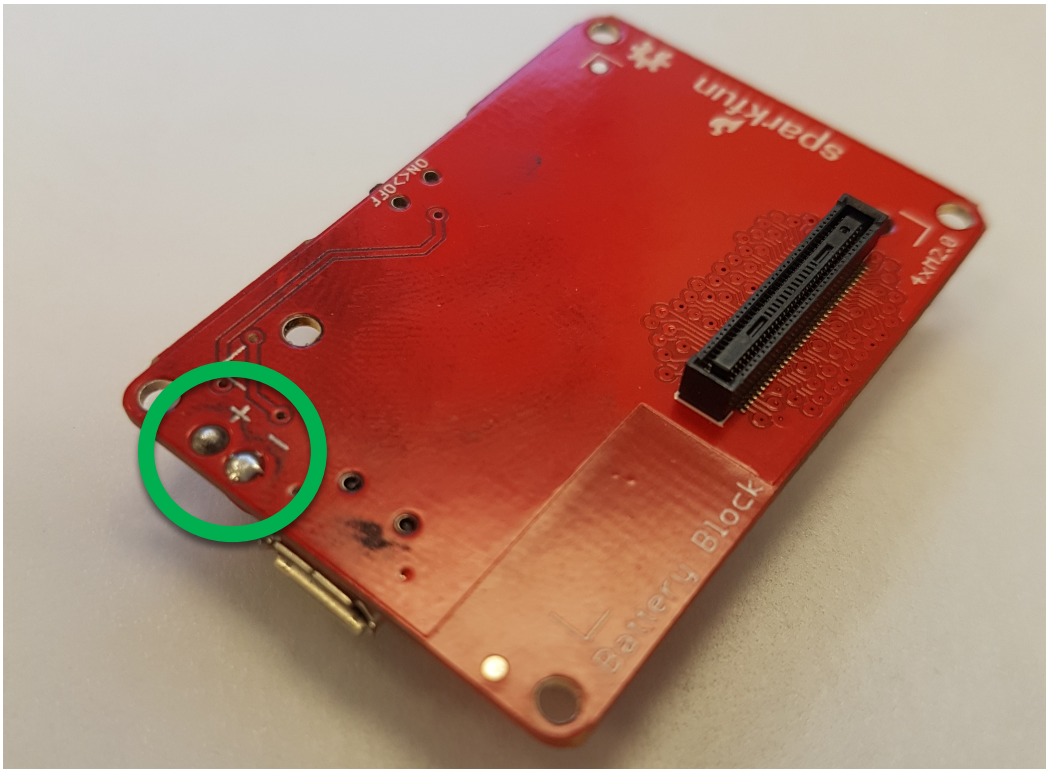


Figure 8: Exposed electrical contacts. These are connected to the terminals of the Li-ion battery

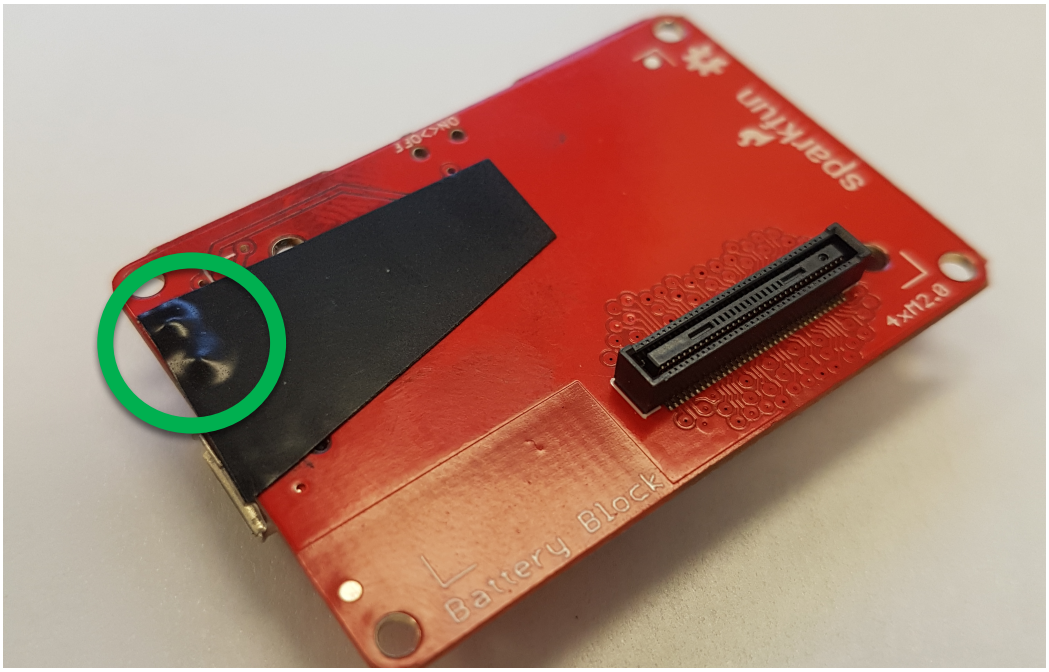


Figure 9: Electrically isolating the contacts for the positive and negative terminals of the Battery Block

7. Re-orient the board to match the below figure, and place a small silver screw in the corner.

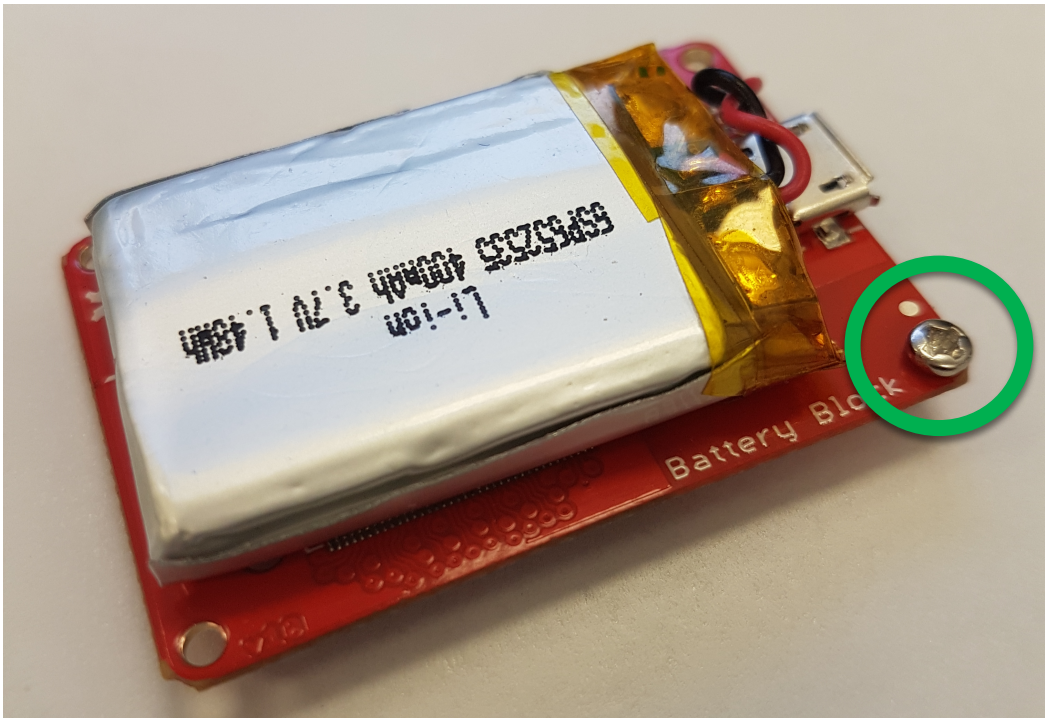


Figure 10: Correct screw placement

8. Re-orient the board to match the figure below and screw the silver screw into the standoff (golden screw).

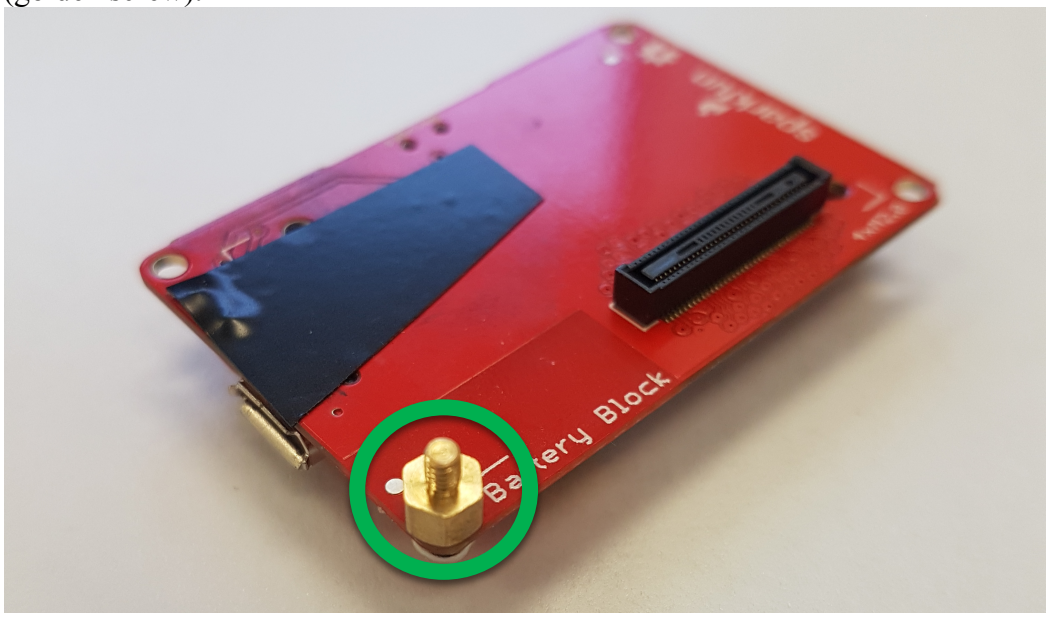


Figure 11: Correct standoff placement

9. Attach screws and standoffs to the remaining four corners.

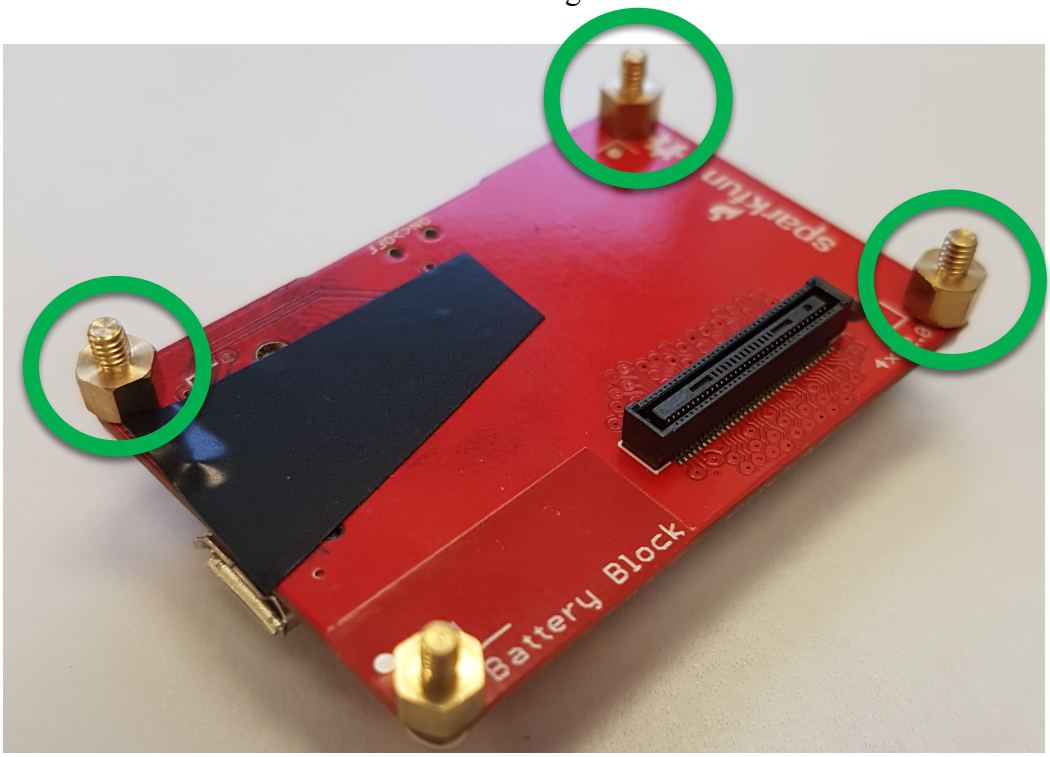


Figure 12: Battery Block after correctly inserting 4 screws and 4 standoffs

10. Attach the 9DOF block as shown. Ensure that the black headers align, and push **firmly** until there is a noticeable **click**.

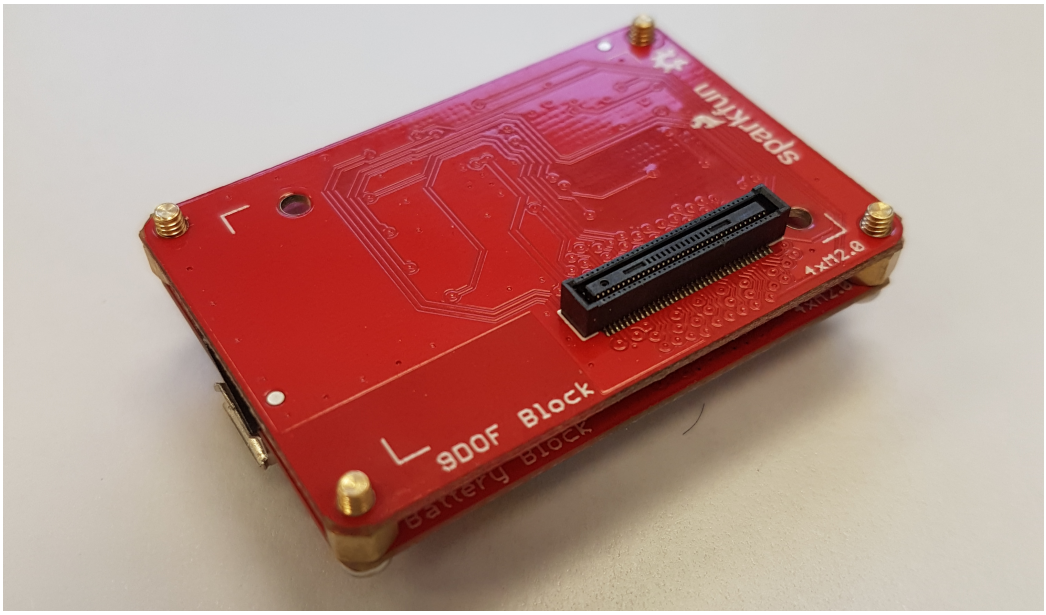


Figure 13: Attaching the 9DOF block to the Battery Block in a secure manner

11. Attach three more standoffs (golden screws) as indicated.

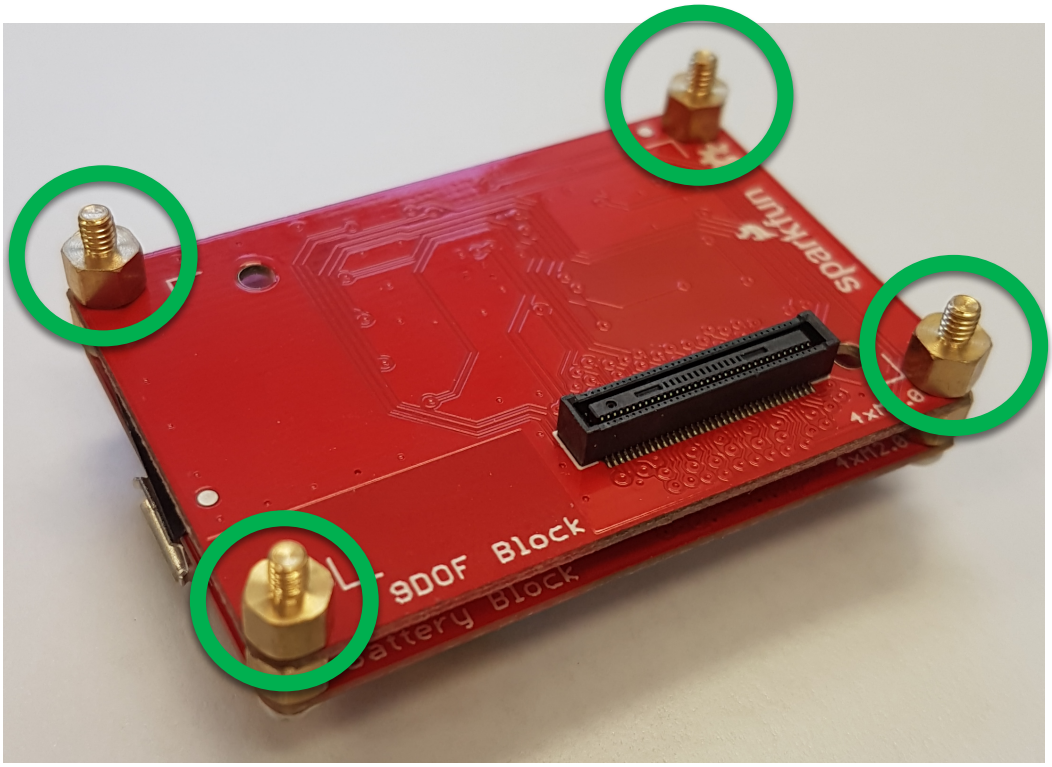


Figure 14: Attaching standoffs to the 9DOF Block

12. Assemble the base block according to the following instructions. Place a screw in the indicated slot.

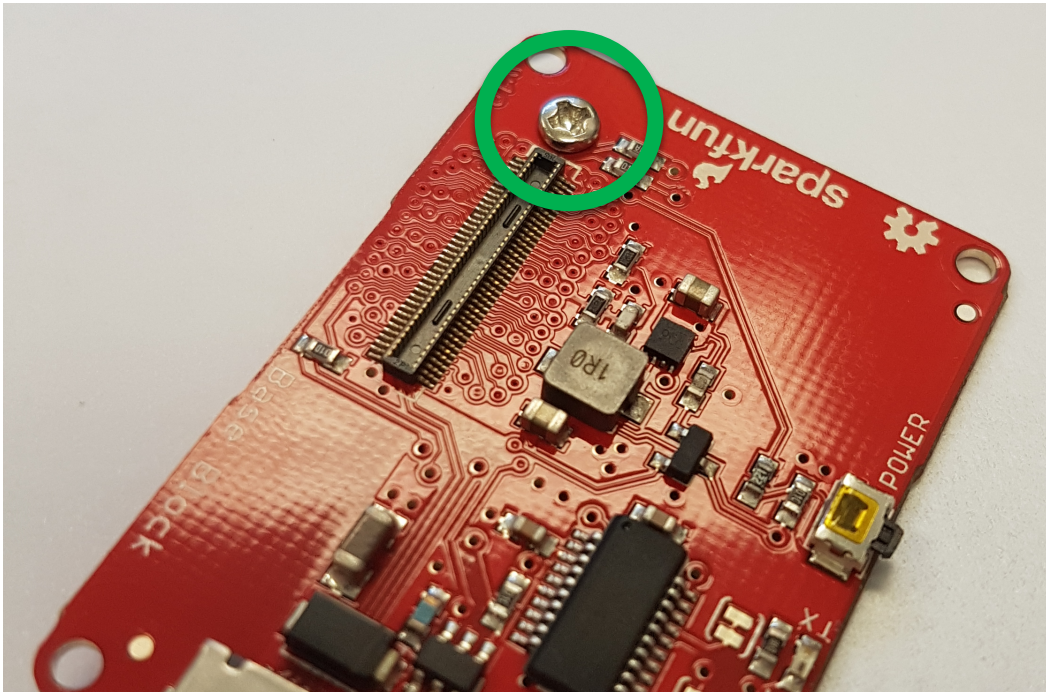


Figure 15: Screw placement for most mechanically secure hardware configuration

Attach a standoff (golden screw) to this screw.

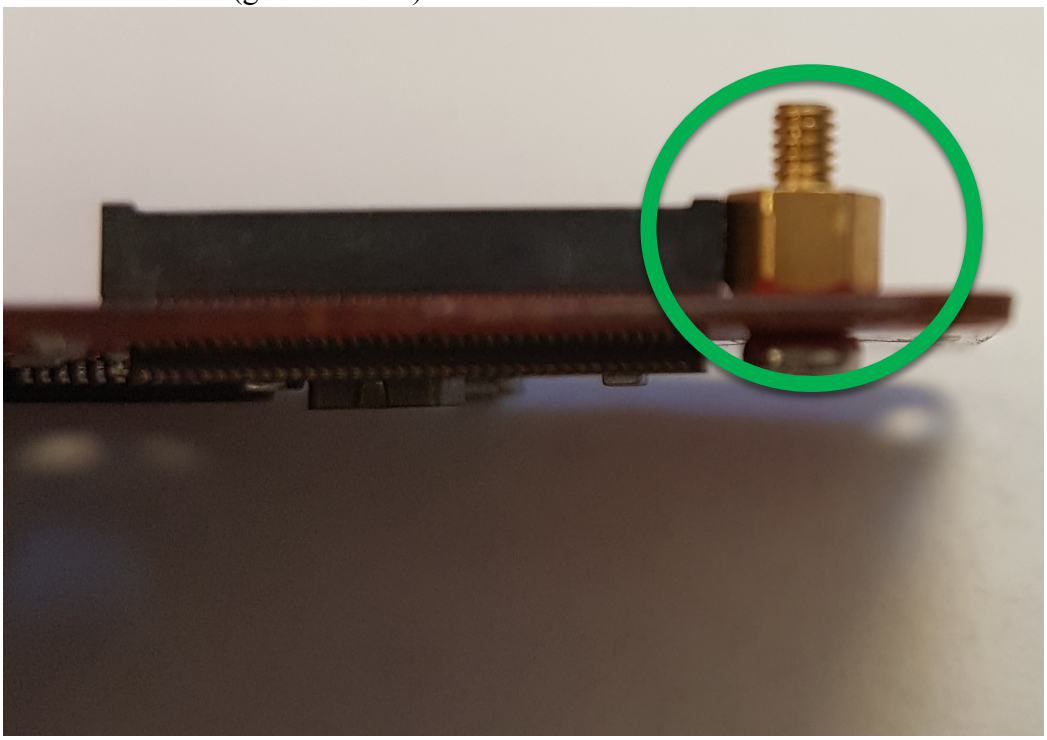


Figure 16: Assembled standoff and screw for Base Block

Attach another screw and standoff to the indicated slot.

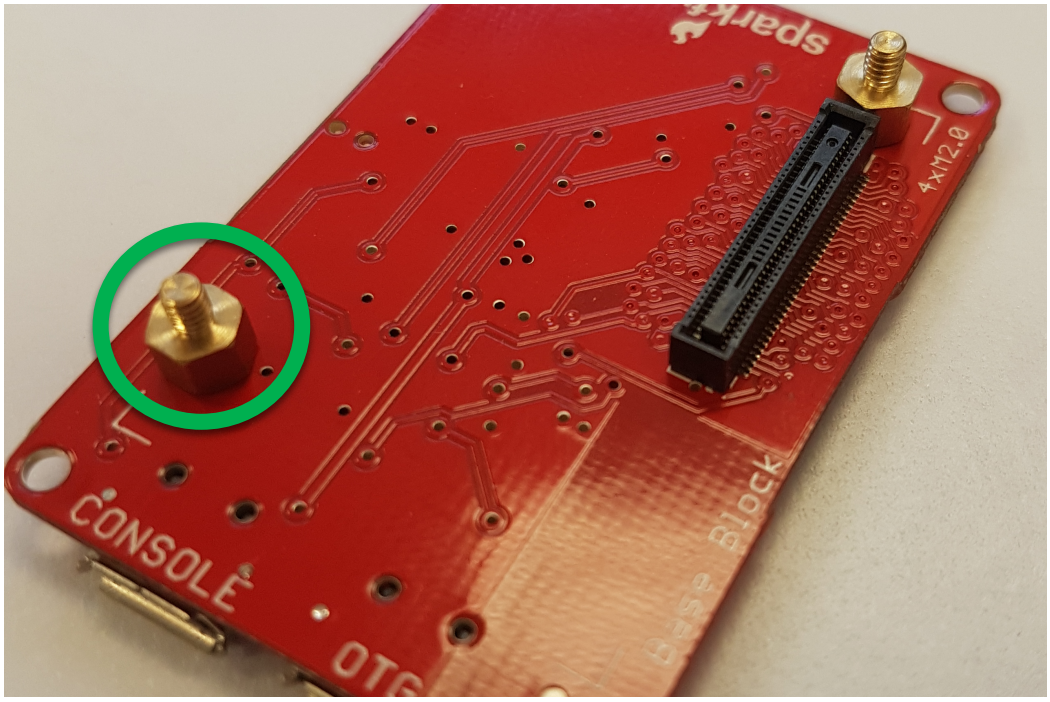


Figure 17: Configuration of screws for mechanically stable Base Block

13. Attach the Intel Edison Compute Module to the base block. Press down until there is a noticeable **click**.

    Note: Make sure the Intel Edison Compute Module has no power supplied to it prior to detaching it from any boards such as the Edison Board for Arduino.

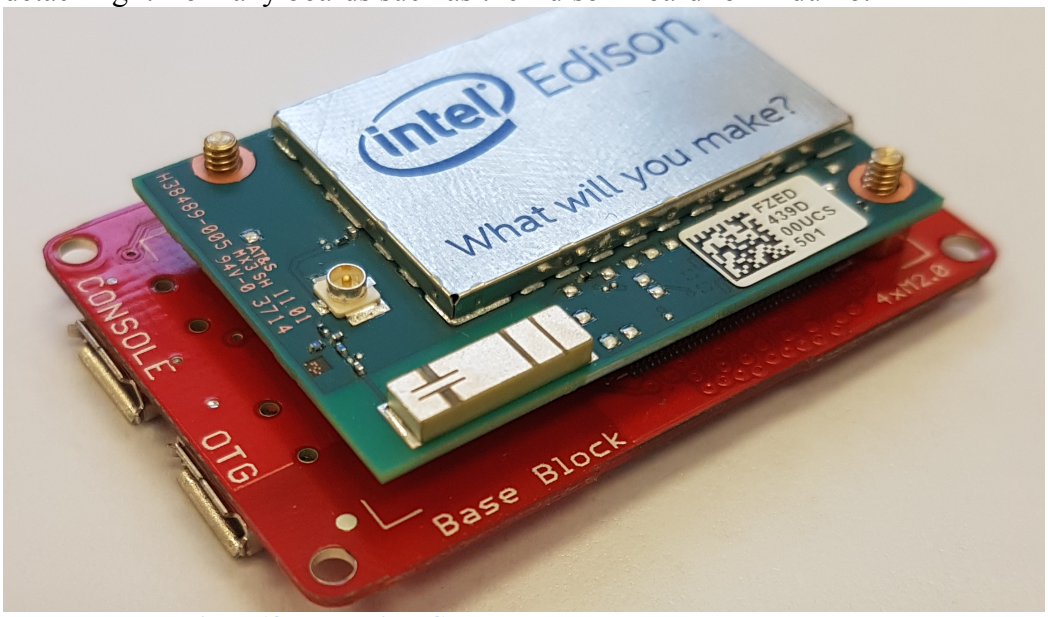


Figure 18: Intel Edison Compute Module attached to the base block

14. Attach the Intel Edison Compute Module to the Base Block. Secure the Intel Edison Compute Module with the nuts available in the Hardware Pack.

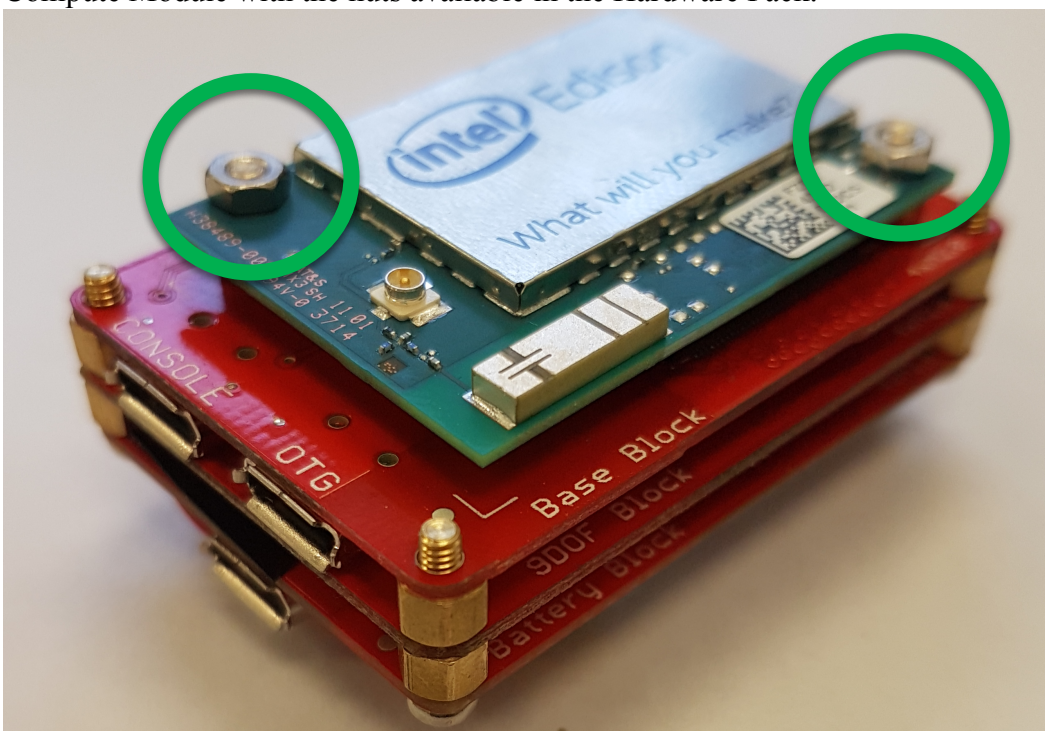


Figure 19: Fully assembled 9DOF SparkFun Kit

15. To charge the battery, supply power to the interface marked in the figure shown below via a micro-B USB cable.

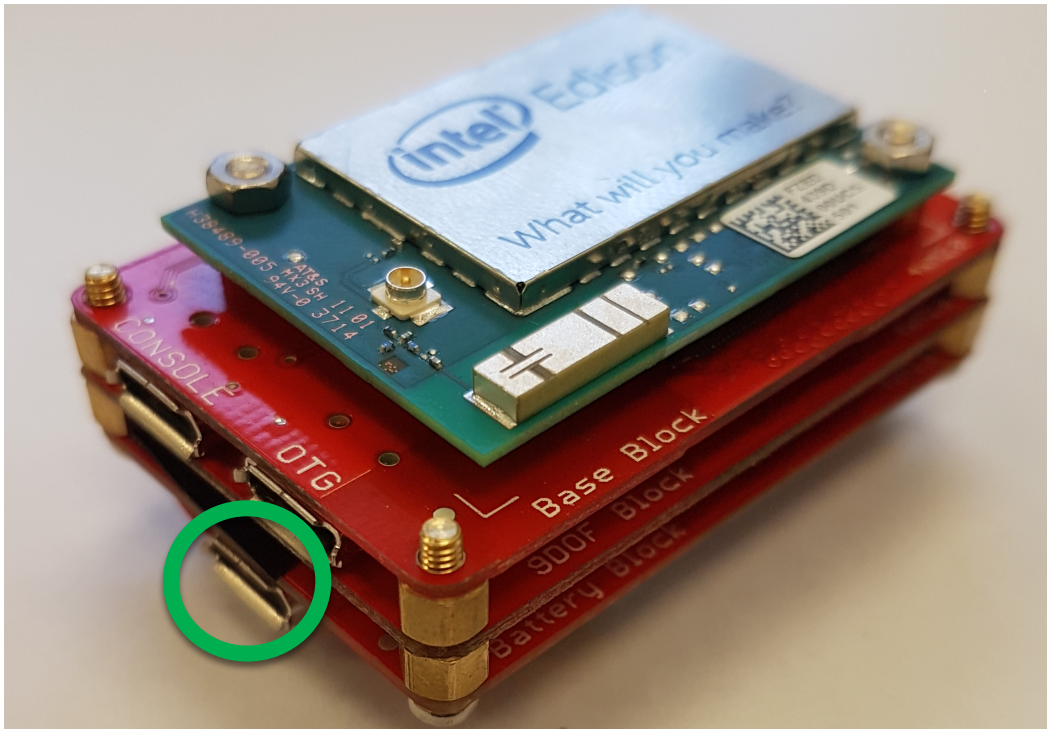


Figure 20: Correct interface to attach micro-B side of a USB cable to charge the battery on the SparkFun battery block

16. Connect a micro USB cable to the interface labelled **CONSOLE** as shown in the figure below to access the shell.

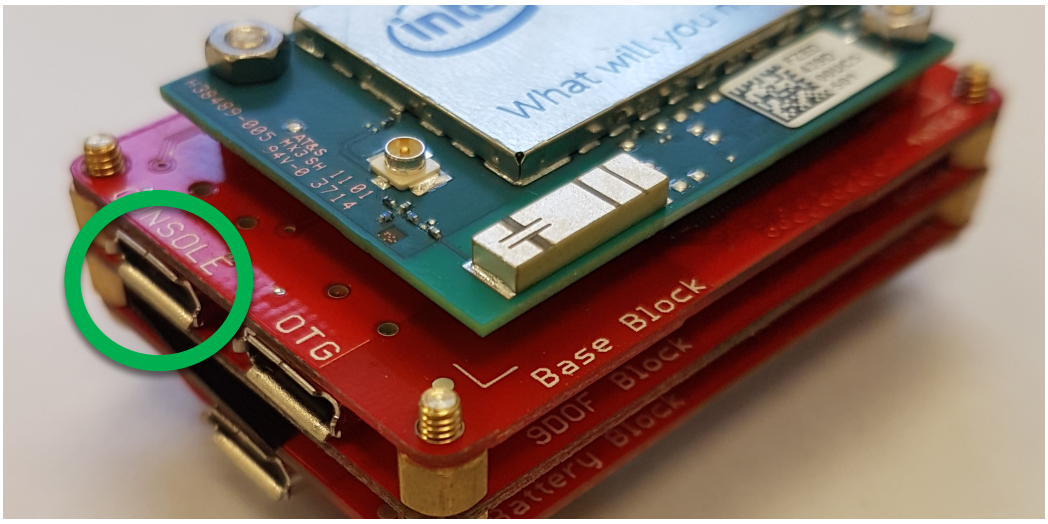


Figure 21: Interface to access the shell on the Intel Edison Compute Module via a wirelines serial interface

17. To enable data collection without the use of **any** cables (power OR USB) follow the steps below.
18. Adjust the switch position such that the kit can draw power from the battery block as indicated in the figures below.

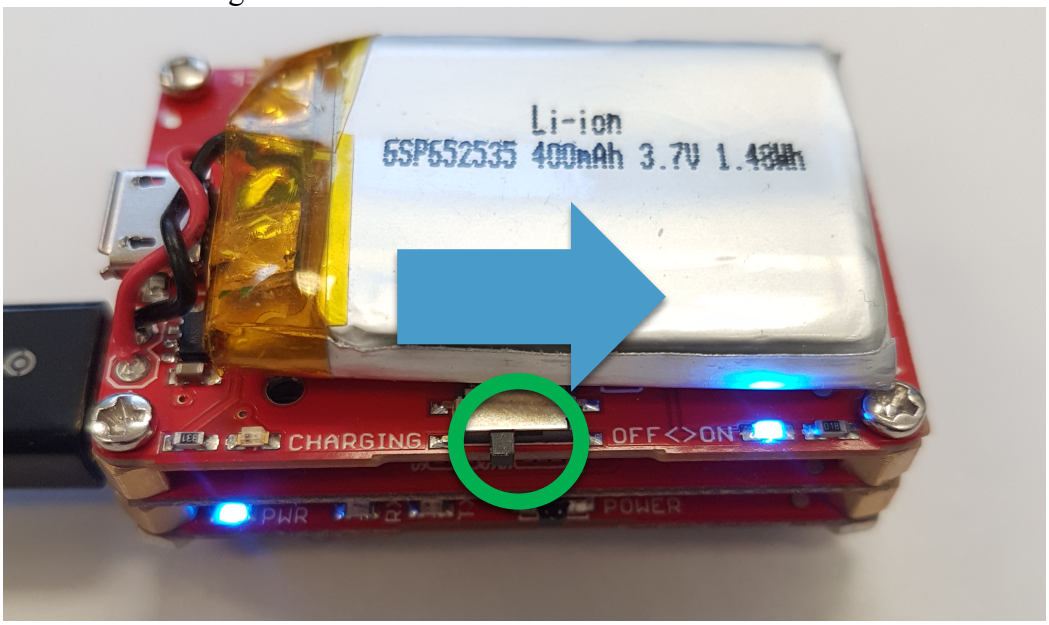


Figure 22: Battery block in 'off' position

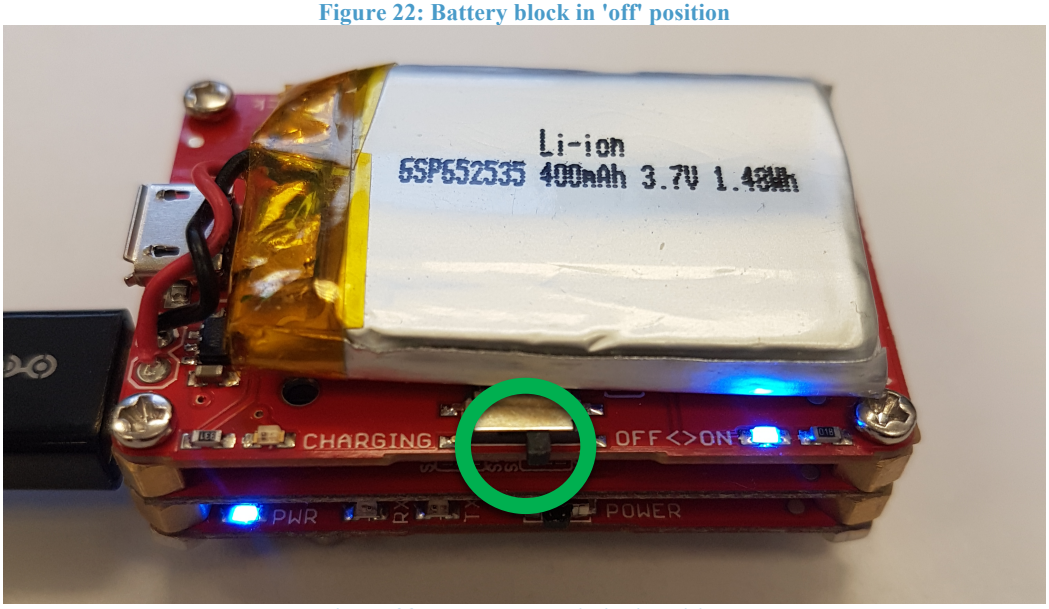
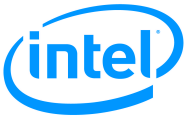
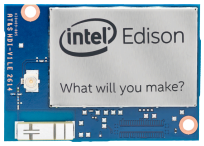


Figure 23: Battery block in 'on' position

19. Access the Linux Operating System Shell on the Intel Edison using the serial interface.
20. Establish a connection between the Intel Edison and a Wi-Fi network.
21. Discover the IP address assigned to the Intel Edison.
22. Access the shell program using SSH.



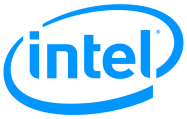


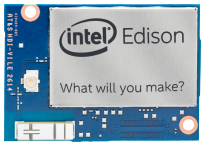
C Library Overview

Users have been provided with a set of functions to interface with the 9DOF block. These set of functions are defined in the header file **LSM9DS0.h** and implemented in the C code source file **LSM9DS0.c**.

Some of the functions defined in the **LSM9DS0.h** header are listed below.

1. **Initialization Functions:**
    a. **accel_init**: This function initializes the I2C connection between the Edison and the accelerometer/magnetometer. It then enables the IMU, sets the sample rate to 200Hz, and sets the scale to 2g.
    b. **gyro_init**: This function initializes the I2C connection between the Edison and the gyroscope. It then enables the IMU, sets the sample rate to 760Hz, and sets the scale to 245 dps.
    c. **mag_init**: This function initializes the I2C connection between the Edison and the magnetometer. It then enables the IMU, sets the sample rate to 100Hz, sets the scale to 2 Gauss, and sets the IMU mode to continuous conversion mode.
2. **Data Scaling:**
    a. **set_accel_scale:** set the upper and lower bounds for data collected by the accelerometer.
    b. **set_gyro_scale:** set the upper and lower bounds for data collected by the gyroscope.
    c. **set_mag_scale:** set the upper and lower bounds for data collected by the magnetometer.
3. **Output Data Rate (ODR):**
    a. **set_accel_ODR:** set ODR of accelerometer.
    b. **set_gyro_ODR:** set ODR and bandwidth of gyroscope.
    c. **set_mag_ODR:** set ODR of magnetometer.
4. **Resolution Determining Functions:**
    a. **calc_accel_res**: calculates g / (ADC tick).
    b. **calc_gyro_res**: calculates degrees per second / (ADC tick).
    c. **calc_mag_res**: calculates gauss / (ADC tick).
5. **Offset Calculation:**
    a. **calc_gyro_offset:** calculates and returns the average output value of each gyroscope axis.
6. **Read Data:**
    a. **read_accel:** acquire data gathered from accelerometer.
    b. **read_gyro:** acquire data gathered from gyroscope.
    c. **read_mag:** acquire data gathered from magnetometer.





Software Configuration and Details

This section will enable users to write C code that can collect data from the 9DOF IMU SparkFun Kit. In particular, the topics of IMU initialization, scale setup, sample rate setup, resolution calculation, and IMU data collection are covered.

IMU Initialization

1. Access the shell on the Intel Edison.
2. Create a new directory and navigate to it.

   $ mkdir ~/IMU
   $ cd ~/IMU

3. Open the following link on a web browser on a personal computer.

   https://drive.google.com/drive/folders/0B4NGslzPqDhvYkgxeVpKVEpYNUk

4. Download and extract the contents of the zip archive labelled **FILES**. The archive should contain the two files listed below.

   LSM9DS0.c
   LSM9DS0.h

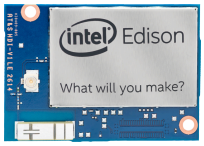
5. Upload the files labelled **LSM9DS0.c** and **LSM9DS0.h** to the folder **~/IMU**.
6. Open a file.

   $ vi imu_init.c

7. Enter the C code below into the Vim editor.

```c
#include <stdio.h>
#include <mraa/i2c.h>
#include "LSM9DS0.h"

int main() {
    mraa_i2c_context accel, gyro, mag;

    accel = accel_init();
    gyro = gyro_init();
    mag = mag_init();

    return 0;
}
```

Figure 24: C code source file imu_init.c



8. The initialization functions (**accel_init**, **gyro_init**, and **mag_init**) return a data structure of type **mraa_i2c_context**. This data structure is used to handle communication using the I2C protocol, and is found in the **mraa** library. Therefore, users must include the **mraa** library as shown in **line 2**.

   In order to use the functions defined in the **LSM9DS0.h** header, users must include **LSM9DS0.h** header file as shown in line 3. Header files typically **define** functions, and C code source files typically **implement** the defined functions. As such, it is recommended that the **LSM9DS0.h** and **LSM9DS0.c** files are located in the same directory as the file **imu_init.c**. Ensuring these files are all in the same directory will ease the compilation process.

9. Save the file, and terminate the Vim editor.
10. Compile the C code source files into a binary executable file.

    $ gcc –lmraa –o imu_init imu_init.c LSM9DS0.c

11. Run the code. Notice that the program does not print anything to standard output.

    $ ./imu_init

Scaling

1. Access the shell on the Intel Edison.
2. Navigate to the IMU directory.

   $ cd ~/IMU

3. Create a copy of the file **imu_init.c** and label it **imu_scale.c**.

   $ cp imu_init.c imu_scale.c

4. Open the C code source file **imu_scale.c**.

   $ vi imu_scale.c

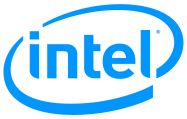
5. Modify the file such that it matches Figure 25.

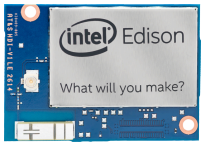
```
1  #include <stdio.h>
2  #include <mraa/i2c.h>
3  #include "LSM9DS0.h"
4
5  int main() {
6      mraa_i2c_context accel, gyro, mag;
7
8      accel = accel_init();
9      set_accel_scale(accel, A_SCALE_2G);
10
11     gyro = gyro_init();
12     set_gyro_scale(gyro, G_SCALE_245DPS);
13
14     mag = mag_init();
15     set_mag_scale(mag, M_SCALE_2GS);
16
17     return 0;
18 }
19
```

Figure 25: C code source file imu_scale.c

6. The 9DOF IMU SparkFun kit is capable of generating data within a specific range (i.e. sensor saturation point). To select this range, users may use the appropriate **scale** function. The scale function requires two input arguments: an I2C context (**accel**, **gyro** or **mag**), and a scale factor. Consult the list below for more information.

   Accelerometer
   - A_SCALE_2G *(±2 g)*
   - A_SCALE_4G *(±4 g)*
   - A_SCALE_6G *(±6 g)*
   - A_SCALE_8G *(±8 g)*
   - A_SCALE_16G *(±16 g)*





Gyroscope

- G_SCALE_245DPS *(±245 degrees per second)*
- G_SCALE_500DPS *(±500 degrees per second)*
- G_SCALE_2000DPS *(±2000 degrees per second)*

Magnetometer

- M_SCALE_2GS *(±2 gauss)*
- M_SCALE_4GS *(±4 gauss)*
- M_SCALE_8GS *(±8 gauss)*
- M_SCALE_12GS *(±12 gauss)*

7. Save the file, and terminate the Vim editor.
8. Compile the C code source files into a binary executable file.

   $ gcc –lmraa –o imu_scale imu_scale.c LSM9DS0.c

9. Run the code. Notice that the program does not print anything to standard output.

   $ ./imu_scale

Output Data Rate Selection

1. Access the shell on the Intel Edison.
2. Navigate to the IMU directory.

   $ cd ~/IMU

3. Create a copy of the file **imu_scale.c** and label it **imu_sample.c**.

   $ cp imu_scale.c imu_sample.c

4. Open the C code source file **imu_sample.c**.

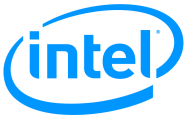
   $ vi imu_sample.c

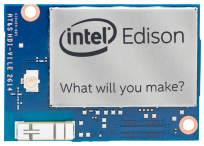
5. Modify the file such that it matches Figure 26.

```
init_imu.c
1  #include <stdio.h>
2  #include <mraa/i2c.h>
3  #include "LSM9DS0.h"
4
5  int main() {
6      mraa_i2c_context accel, gyro, mag;
7
8      accel = accel_init();
9      set_accel_scale(accel, A_SCALE_2G);
10     set_accel_ODR(accel, A_ODR_100);
11
12     gyro = gyro_init();
13     set_gyro_scale(gyro, G_SCALE_245DPS);
14     set_gyro_ODR(gyro, G_ODR_190_BW_70);
15
16     mag = mag_init();
17     set_mag_scale(mag, M_SCALE_2GS);
18     set_mag_ODR(mag, M_ODR_125);
19
20     return 0;
21 }
22
```

Figure 26: C code source file imu_sample.c

6. The 9DOF IMU SparkFun kit is capable of generating data at different rates. To select the output data rate (ODR), users may use the appropriate ODR function. The ODR function requires two input arguments: an I2C context (**accel**, **gyro** or **mag**), and a ODR value. It is important to note that selecting the **set_gyro_ODR** function sets both the





ODR and the bandwidth (BW) for the gyroscope. Consult the list below for more information.

Accelerometer

- A_POWER_DOWN *(Power-down mode)*
- A_ODR_3125 *(3.125 Hz)*
- A_ODR_625 *(6.25 Hz)*
- A_ODR_125 *(12.5 Hz)*
- A_ODR_25 *(25 Hz)*
- A_ODR_50 *(50 Hz)*
- A_ODR_100 *(100 Hz)*
- A_ODR_200 *(200 Hz)*
- A_ODR_400 *(400 Hz)*
- A_ODR_800 *(800 Hz)*
- A_ODR_1600 *(1600 Hz)*

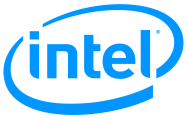
Gyroscope

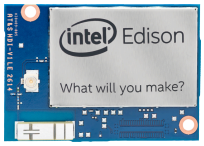
- G_ODR_95_BW_125 *(rate: 95 Hz, BW:12.5)*
- G_ODR_95_BW_25 *(rate: 95 Hz, BW: 25)*
- G_ODR_190_BW_125 *(rate: 190 Hz, BW: 12.5)*
- G_ODR_190_BW_25 *(rate: 190 Hz, BW: 25)*
- G_ODR_190_BW_50 *(rate: 190 Hz, BW: 50)*
- G_ODR_190_BW_70 *(rate: 190 Hz, BW: 70)*
- G_ODR_380_BW_20 *(rate: 380 Hz, BW: 20)*
- G_ODR_380_BW_25 *(rate: 380 Hz, BW: 25)*
- G_ODR_380_BW_50 *(rate: 380 Hz, BW: 50)*
- G_ODR_380_BW_100 *(rate: 380 Hz, BW: 100)*
- G_ODR_760_BW_30 *(rate: 760 Hz, BW: 30)*
- G_ODR_760_BW_35 *(rate: 760 Hz, BW: 35)*
- G_ODR_760_BW_50 *(rate: 760 Hz, BW: 50)*
- G_ODR_760_BW_100 *(rate: 760 Hz, BW: 100)*

Magnetometer

- M_ODR_3125 (3.125 Hz)
- M_ODR_625 (6.25 Hz)
- M_ODR_125 (12.5 Hz)
- M_ODR_25 (25 Hz)
- M_ODR_50 (50 Hz)
- M_ODR_100 (100 Hz)

7. While the above functions set the ODR, the user may still choose to sample the data at a different rate. In most cases, selecting a lower ODR, BW or sample rate will cause the 9DOF IMU to consume less energy.
8. Save the file, and terminate the Vim editor.



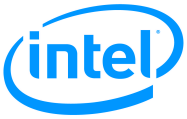


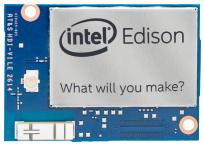
9. Compile the C code source files into a binary executable file.

   $ gcc –lmraa –o imu_sample imu_sample.c LSM9DS0.c

10. Run the code. Notice that the program does not print anything to standard output.

    $ ./imu_sample





Resolution Calculation

1. Access the shell on the Intel Edison.
2. Navigate to the IMU directory.

   $ cd ~/IMU

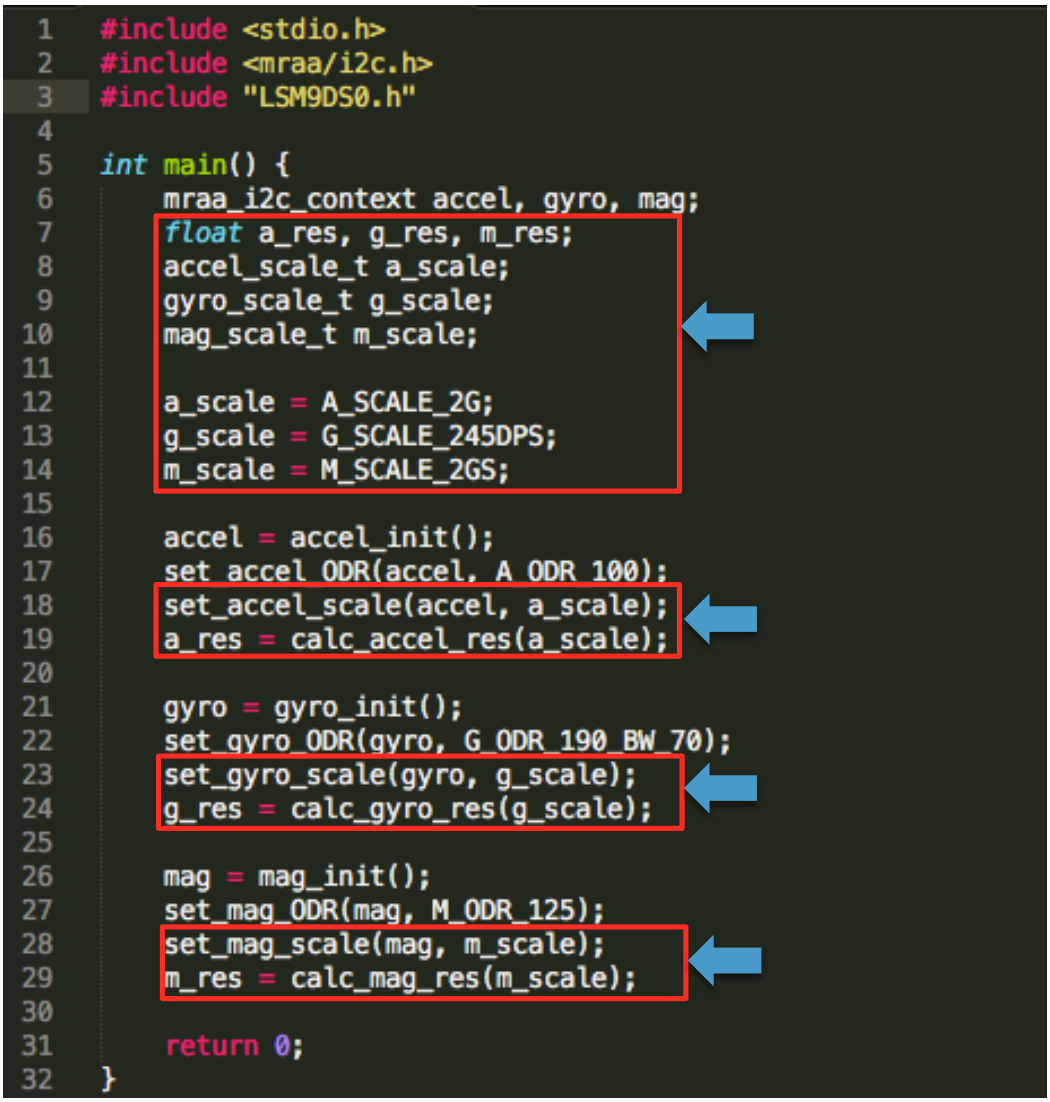
3. Create a copy of the file **imu_sample.c** and label it **imu_res.c**.

   $ cp imu_sample.c imu_res.c

4. Open the C code source file **imu_res.c**.

   $ vi imu_res.c

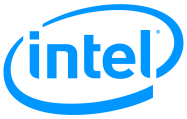
5. Modify the file such that it matches Figure 27.

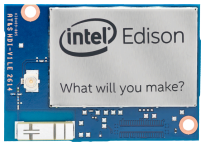


```
#include <stdio.h>
#include <mraa/i2c.h>
#include "LSM9DS0.h"

int main() {
    mraa_i2c_context accel, gyro, mag;
    float a_res, g_res, m_res;
    accel_scale_t a_scale;
    gyro_scale_t g_scale;
    mag_scale_t m_scale;

    a_scale = A_SCALE_2G;
    g_scale = G_SCALE_245DPS;
    m_scale = M_SCALE_2GS;

    accel = accel_init();
    set_accel_ODR(accel, A_ODR_100);
    set_accel_scale(accel, a_scale);
    a_res = calc_accel_res(a_scale);

    gyro = gyro_init();
    set_gyro_ODR(gyro, G_ODR_190_BW_70);
    set_gyro_scale(gyro, g_scale);
    g_res = calc_gyro_res(g_scale);

    mag = mag_init();
    set_mag_ODR(mag, M_ODR_125);
    set_mag_scale(mag, m_scale);
    m_res = calc_mag_res(m_scale);

    return 0;
}
```

Figure 27: C code source file imu_res.c

6. The 9DOF IMU SparkFun kit is capable of generating data at different resolutions. However, the resolution of the output data is dependent on the scale factor. To determine the resolution of the generated data, users may use the appropriate calculate resolution function. The calculate resolution function requires the scaling factor as the input argument. To ensure the same scale is provided to both the scale selection function and the resolution calculation function, the user defined scale should be stored in a variable.
7. There is a common trade-off in sensor technologies between **sensor resolution** and **sensor scale**. If measurements are taken of over a large scale (e.g. accelerometer operating over the range of ± 16g), the resolution will be worse than measurements being taken at a smaller scale (e.g. accelerometer operating over the range ± 2g). However, if a sensor operates at a smaller scale (± 2g), the readings will saturate if the acceleration is outside the ±2g range.
8. Save the file, and terminate the Vim editor.
9. Compile the C code source files into a binary executable file.

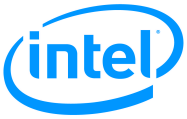


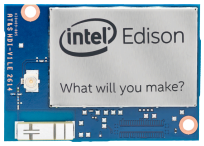


$ gcc –lmraa –o imu_res imu_res.c LSM9DS0.c

10. Run the code. Notice that the program does not print anything to standard output.

$ ./imu_res





Data Collection:

1. Access the shell on the Intel Edison.
2. Navigate to the IMU directory.

   $ cd ~/IMU

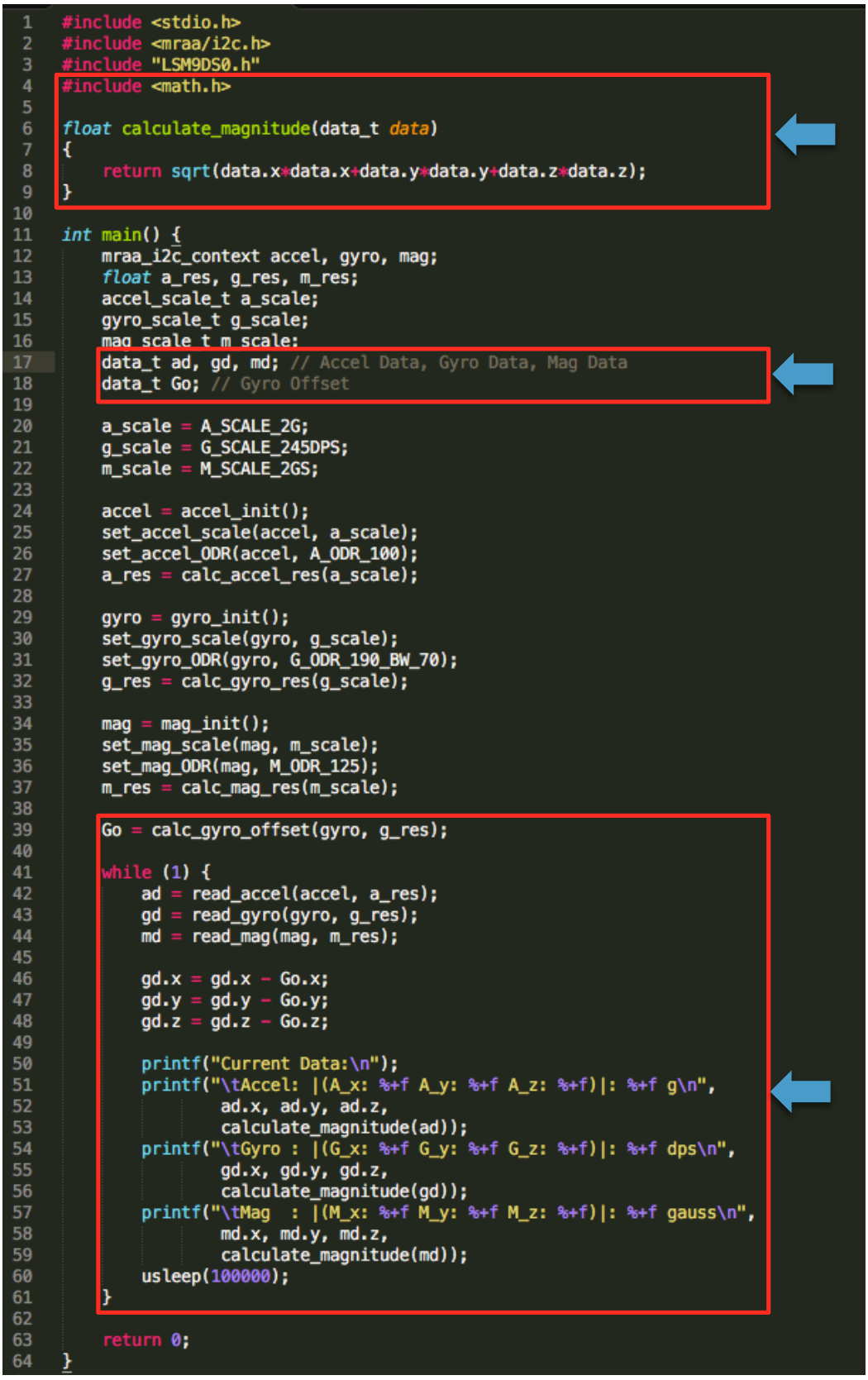
3. Create a copy of the file **imu_res.c** and label it **imu_data.c**.

   $ cp imu_res.c imu_data.c

4. Open the C code source file **imu_data.c**.

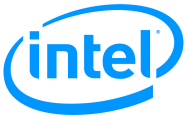
   $ vi imu_data.c

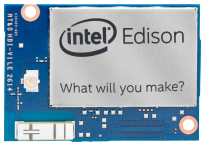
5. Modify the file such that it matches Figure 28.



```c
#include <stdio.h>
#include <mraa/i2c.h>
#include "LSM9DS0.h"
#include <math.h>

float calculate_magnitude(data_t data)
{
    return sqrt(data.x*data.x+data.y*data.y+data.z*data.z);
}

int main() {
    mraa_i2c_context accel, gyro, mag;
    float a_res, g_res, m_res;
    accel_scale_t a_scale;
    gyro_scale_t g_scale;
    mag_scale_t m_scale;
    data_t ad, gd, md; // Accel Data, Gyro Data, Mag Data
    data_t Go; // Gyro Offset

    a_scale = A_SCALE_2G;
    g_scale = G_SCALE_245DPS;
    m_scale = M_SCALE_2GS;

    accel = accel_init();
    set_accel_scale(accel, a_scale);
    set_accel_ODR(accel, A_ODR_100);
    a_res = calc_accel_res(a_scale);

    gyro = gyro_init();
    set_gyro_scale(gyro, g_scale);
    set_gyro_ODR(gyro, G_ODR_190_BW_70);
    g_res = calc_gyro_res(g_scale);

    mag = mag_init();
    set_mag_scale(mag, m_scale);
    set_mag_ODR(mag, M_ODR_125);
    m_res = calc_mag_res(m_scale);

    Go = calc_gyro_offset(gyro, g_res);

    while (1) {
        ad = read_accel(accel, a_res);
        gd = read_gyro(gyro, g_res);
        md = read_mag(mag, m_res);

        gd.x = gd.x - Go.x;
        gd.y = gd.y - Go.y;
        gd.z = gd.z - Go.z;

        printf("Current Data:\n");
        printf("\tAccel: |(A_x: %+f A_y: %+f A_z: %+f)|: %+f g\n",
                ad.x, ad.y, ad.z,
                calculate_magnitude(ad));
        printf("\tGyro : |(G_x: %+f G_y: %+f G_z: %+f)|: %+f dps\n",
                gd.x, gd.y, gd.z,
                calculate_magnitude(gd));
        printf("\tMag  : |(M_x: %+f M_y: %+f M_z: %+f)|: %+f gauss\n",
                md.x, md.y, md.z,
                calculate_magnitude(md));
        usleep(100000);
    }

    return 0;
}
```

Figure 28: C code source file imu_data.c





6. The 9DOF IMU SparkFun kit is capable writes the generated data to a register. To acquire the generated data, users may use the appropriate read function. The read function requires two input arguments: an I2C context, and the resolution.
7. The return type of the data is **data_t**. This structure is defined as follows.

   typedef struct {
   float x, y, z
   } data_t;

   To declare a **data_t** structure, use the following syntax.

   data_t ad;

   To access the **x**, **y**, and **z** components of the **data_t** structure, use the following syntax.

   acceleration_x: ad.x
   acceleration_y: ad.y
   acceleration_z: ad.z

8. Save the file, and terminate the Vim editor.
9. Compile the C code source files into a binary executable file. Take note of the **–lm** flag. This enables access to the **sqrt** function used in the **calculate_magnitude** function.

   $ gcc –lmraa –lm –o imu_data imu_data.c LSM9DS0.c

10. Run the code. Notice how data is being printed to standard output.

    $ ./imu_data

11. While the code is running, move the 9DOF IMU SparkFun kit and examine the change in data printed to standard output.
12. To terminate the **imu_data** process, issue the **ctrl-C** keystroke.
13. Edit the C code source file such that instead of printing an acceleration magnitude, the code will print “UP” if the acceleration in the z-axis is positive, and “DOWN” if the acceleration in the z-axis is negative.
14. Compile and test the code resulting from step 13.
15. Adjust the scale parameters and see how this affects the data printed to standard output.