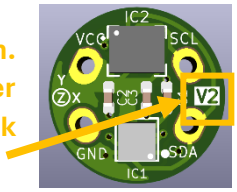
This document corresponds to the V2 of the IMU

This IMU is composed of a 6-axis IMU (3-axis accelerometer + 3-axis gyroscope) thanks to the LSM6DSOX chip (LGA-14L packaging) by STMicroelectronics, and a 3-axis magnetometer thanks to the LIS3MDL chip (LGA-12L packaging) by STMicroelectronics.

Electronic characteristics

The sensors must be powered by 3.3V, and the I2C signals must also be in 3.3V. Connecting it to 5V signals will destroy the IMU.

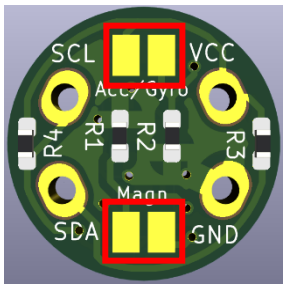
On the V1, the SDA and SCL are reversed on the PCB silkscreen. Connect the SDA of the PCB to the SCL of the microcontroller and the SCL of the PCB to the SDA of the microcontroller. Check the version of the PCB to make sure.



I2C address

The original I2C addresses of the chips are as follows:

- LSM6DSOX – **0x6A**
- LIS3MDL – **0x1C**



These addresses can be modified by soldering the two jumpers at the back of the PCB. The top jumper modifies the address of the accelerometer and gyroscope (LSM6DSOX). The bottom one modifies the address of the magnetometer (LIS3MDL).

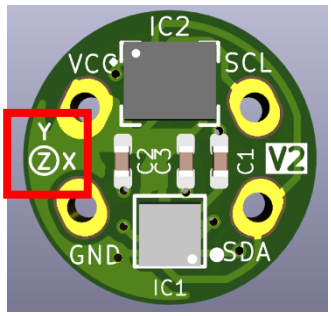
The I2C addresses then become:

- LSM6DSOX – **0x6B**
- LIS3MDL – **0x1E**

This means that maximum 2 IMUs can be connected on any I2C bus. If more IMUs are needed, multiple I2C buses (or an I2C expander) are needed.

Axis

The X, Y and Z axis is shown between the VCC and the GND pins.

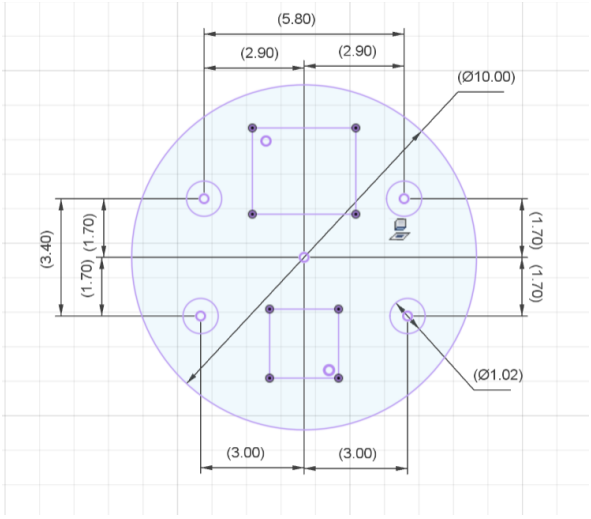


Code

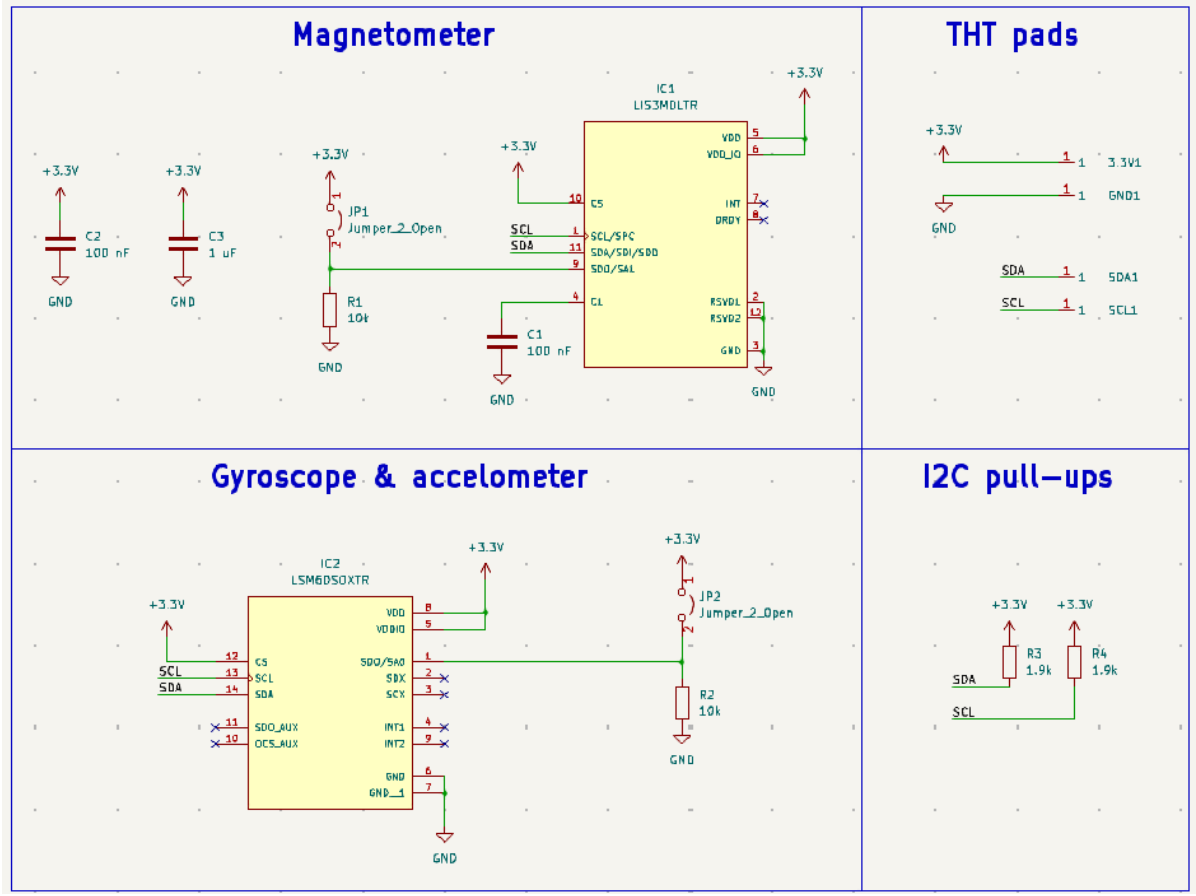
This PCB is based on the one from Adafruit. The Arduino Adafruit library is therefore compatible with this PCB.
The code in annex is the code I use to read data from two PCBs on one I2C bus on a teensy 4.0.

This document corresponds to the V2 of the IMU

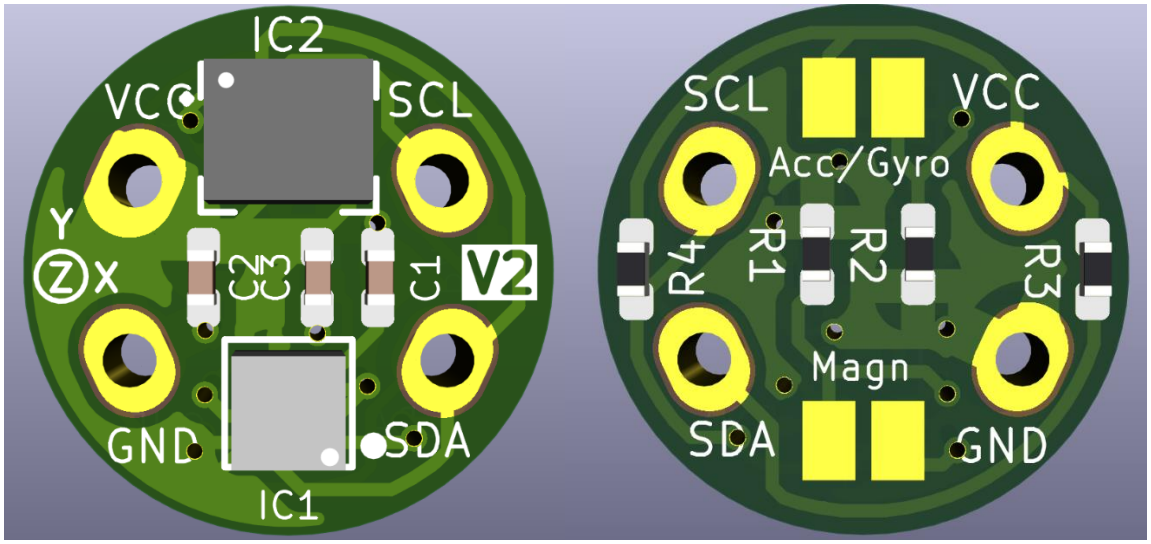
Dimensions



Schematic and PCB



This document corresponds to the V2 of the IMU



This document corresponds to the V2 of the IMU

Annex - Code

```
// SPDX-FileCopyrightText: 2020 Kattni Rembor for Adafruit Industries
// SPDX-License-Identifier: MIT

// Basic demo for accelerometer, gyro, and magnetometer readings
// from the following Adafruit ST Sensor combo boards:
// * LSM6DSOX + LIS3MDL FeatherWing : https://www.adafruit.com/product/4565
// * ISM330DHCX + LIS3MDL FeatherWing https://www.adafruit.com/product/4569
// * LSM6DSOX + LIS3MDL Breakout : https://www.adafruit.com/product/4517
// * LSM6DS33 + LIS3MDL Breakout Lhttps://www.adafruit.com/product/4485

#include <Adafruit_LSM6DSOX.h>
Adafruit_LSM6DSOX lsm6ds;
Adafruit_LSM6DSOX lsm6ds_2;

// To use with the LSM6DS33+LIS3MDL breakout, uncomment these two lines
// and comment out the lines referring to the LSM6DSOX above
//#include <Adafruit_LSM6DS33.h>
//Adafruit_LSM6DS33 lsm6ds;

// To use with the ISM330DHCX+LIS3MDL Feather Wing, uncomment these two lines
// and comment out the lines referring to the LSM6DSOX above
//#include <Adafruit_ISM330DHCX.h>
//Adafruit_ISM330DHCX lsm6ds;

// To use with the LSM6D3TR-C+LIS3MDL breakout, uncomment these two lines
// and comment out the lines referring to the LSM6DSOX above
//#include <Adafruit_LSM6DS3TRC.h>
//Adafruit_LSM6DS3TRC lsm6ds;

#include <Adafruit_LIS3MDL.h>
Adafruit_LIS3MDL lis3mdl;
Adafruit_LIS3MDL lis3mdl_2;

void setup(void) {
 Serial.begin(115200);
 while (!Serial)
   delay(10); // will pause Zero, Leonardo, etc until serial console opens

 Serial.println("Adafruit LSM6DS+LIS3MDL test!");

 bool lsm6ds_success, lis3mdl_success;
 bool lsm6ds_success_2, lis3mdl_success_2;

 // hardware I2C mode, can pass in address & alt Wire

 lsm6ds_success = lsm6ds.begin_I2C();
 lis3mdl_success = lis3mdl.begin_I2C();
 lsm6ds_success_2 = lsm6ds_2.begin_I2C(0x6B);
 lis3mdl_success_2 = lis3mdl_2.begin_I2C(0x1E);

 if (!lsm6ds_success){
   Serial.println("Failed to find LSM6DS chip");
 }
 if (!lis3mdl_success_2){
   Serial.println("Failed to find second LSM6DS chip");
 }
 if (!lis3mdl_success){
   Serial.println("Failed to find LIS3MDL chip");
 }
 if (!lis3mdl_success_2){
   Serial.println("Failed to find second LIS3MDL chip");
 }
 if (!(lsm6ds_success && lis3mdl_success && lsm6ds_success_2 && lis3mdl_success_2)) {
   while (1) {
     delay(10);
   }
 }
```

This document corresponds to the V2 of the IMU

```
 Serial.println("2 LSM6DS and 2 LIS3MDL Found!");

 // lsm6ds.setAccelRange(LSM6DS_ACCEL_RANGE_2_G);
 Serial.print("Accelerometer range set to: ");
 switch (lsm6ds.getAccelRange()) {
 case LSM6DS_ACCEL_RANGE_2_G:
  Serial.println("+-2G");
  break;
 case LSM6DS_ACCEL_RANGE_4_G:
  Serial.println("+-4G");
  break;
 case LSM6DS_ACCEL_RANGE_8_G:
  Serial.println("+-8G");
  break;
 case LSM6DS_ACCEL_RANGE_16_G:
  Serial.println("+-16G");
  break;
 }

 // lsm6ds.setAccelDataRate(LSM6DS_RATE_12_5_HZ);
 Serial.print("Accelerometer data rate set to: ");
 switch (lsm6ds.getAccelDataRate()) {
 case LSM6DS_RATE_SHUTDOWN:
  Serial.println("0 Hz");
  break;
 case LSM6DS_RATE_12_5_HZ:
  Serial.println("12.5 Hz");
  break;
 case LSM6DS_RATE_26_HZ:
  Serial.println("26 Hz");
  break;
 case LSM6DS_RATE_52_HZ:
  Serial.println("52 Hz");
  break;
 case LSM6DS_RATE_104_HZ:
  Serial.println("104 Hz");
  break;
 case LSM6DS_RATE_208_HZ:
  Serial.println("208 Hz");
  break;
 case LSM6DS_RATE_416_HZ:
  Serial.println("416 Hz");
  break;
 case LSM6DS_RATE_833_HZ:
  Serial.println("833 Hz");
  break;
 case LSM6DS_RATE_1_66K_HZ:
  Serial.println("1.66 KHz");
  break;
 case LSM6DS_RATE_3_33K_HZ:
  Serial.println("3.33 KHz");
  break;
 case LSM6DS_RATE_6_66K_HZ:
  Serial.println("6.66 KHz");
  break;
 }

 // lsm6ds.setGyroRange(LSM6DS_GYRO_RANGE_250_DPS );
 Serial.print("Gyro range set to: ");
 switch (lsm6ds.getGyroRange()) {
 case LSM6DS_GYRO_RANGE_125_DPS:
  Serial.println("125 degrees/s");
  break;
 case LSM6DS_GYRO_RANGE_250_DPS:
  Serial.println("250 degrees/s");
  break;
 case LSM6DS_GYRO_RANGE_500_DPS:
  Serial.println("500 degrees/s");
  break;
 case LSM6DS_GYRO_RANGE_1000_DPS:
  Serial.println("1000 degrees/s");
```

This document corresponds to the V2 of the IMU

```
  break;
 case LSM6DS_GYRO_RANGE_2000_DPS:
  Serial.println("2000 degrees/s");
  break;
 case ISM330DHCX_GYRO_RANGE_4000_DPS:
  Serial.println("4000 degrees/s");
  break;
 }
 // lsm6ds.setGyroDataRate(LSM6DS_RATE_12_5_HZ);
 Serial.print("Gyro data rate set to: ");
 switch (lsm6ds.getGyroDataRate()) {
 case LSM6DS_RATE_SHUTDOWN:
  Serial.println("0 Hz");
  break;
 case LSM6DS_RATE_12_5_HZ:
  Serial.println("12.5 Hz");
  break;
 case LSM6DS_RATE_26_HZ:
  Serial.println("26 Hz");
  break;
 case LSM6DS_RATE_52_HZ:
  Serial.println("52 Hz");
  break;
 case LSM6DS_RATE_104_HZ:
  Serial.println("104 Hz");
  break;
 case LSM6DS_RATE_208_HZ:
  Serial.println("208 Hz");
  break;
 case LSM6DS_RATE_416_HZ:
  Serial.println("416 Hz");
  break;
 case LSM6DS_RATE_833_HZ:
  Serial.println("833 Hz");
  break;
 case LSM6DS_RATE_1_66K_HZ:
  Serial.println("1.66 KHz");
  break;
 case LSM6DS_RATE_3_33K_HZ:
  Serial.println("3.33 KHz");
  break;
 case LSM6DS_RATE_6_66K_HZ:
  Serial.println("6.66 KHz");
  break;
 }

 lis3mdl.setDataRate(LIS3MDL_DATARATE_155_HZ);
 // You can check the datarate by looking at the frequency of the DRDY pin
 Serial.print("Magnetometer data rate set to: ");
 switch (lis3mdl.getDataRate()) {
  case LIS3MDL_DATARATE_0_625_HZ: Serial.println("0.625 Hz"); break;
  case LIS3MDL_DATARATE_1_25_HZ: Serial.println("1.25 Hz"); break;
  case LIS3MDL_DATARATE_2_5_HZ: Serial.println("2.5 Hz"); break;
  case LIS3MDL_DATARATE_5_HZ: Serial.println("5 Hz"); break;
  case LIS3MDL_DATARATE_10_HZ: Serial.println("10 Hz"); break;
  case LIS3MDL_DATARATE_20_HZ: Serial.println("20 Hz"); break;
  case LIS3MDL_DATARATE_40_HZ: Serial.println("40 Hz"); break;
  case LIS3MDL_DATARATE_80_HZ: Serial.println("80 Hz"); break;
  case LIS3MDL_DATARATE_155_HZ: Serial.println("155 Hz"); break;
  case LIS3MDL_DATARATE_300_HZ: Serial.println("300 Hz"); break;
  case LIS3MDL_DATARATE_560_HZ: Serial.println("560 Hz"); break;
  case LIS3MDL_DATARATE_1000_HZ: Serial.println("1000 Hz"); break;
 }

 lis3mdl.setRange(LIS3MDL_RANGE_4_GAUSS);
 Serial.print("Range set to: ");
 switch (lis3mdl.getRange()) {
  case LIS3MDL_RANGE_4_GAUSS: Serial.println("+-4 gauss"); break;
  case LIS3MDL_RANGE_8_GAUSS: Serial.println("+-8 gauss"); break;
  case LIS3MDL_RANGE_12_GAUSS: Serial.println("+-12 gauss"); break;
  case LIS3MDL_RANGE_16_GAUSS: Serial.println("+-16 gauss"); break;
```

This document corresponds to the V2 of the IMU

```
 }

 lis3mdl.setPerformanceMode(LIS3MDL_MEDIUMMODE);
 Serial.print("Magnetometer performance mode set to: ");
 switch (lis3mdl.getPerformanceMode()) {
  case LIS3MDL_LOWPOWERMODE: Serial.println("Low"); break;
  case LIS3MDL_MEDIUMMODE: Serial.println("Medium"); break;
  case LIS3MDL_HIGHMODE: Serial.println("High"); break;
  case LIS3MDL_ULTRAHIGHMODE: Serial.println("Ultra-High"); break;
 }

 lis3mdl.setOperationMode(LIS3MDL_CONTINUOUSMODE);
 Serial.print("Magnetometer operation mode set to: ");
 // Single shot mode will complete conversion and go into power down
 switch (lis3mdl.getOperationMode()) {
  case LIS3MDL_CONTINUOUSMODE: Serial.println("Continuous"); break;
  case LIS3MDL_SINGLEMODE: Serial.println("Single mode"); break;
  case LIS3MDL_POWERDOWNMODE: Serial.println("Power-down"); break;
 }

 lis3mdl.setIntThreshold(500);
 lis3mdl.configInterrupt(false, false, true, // enable z axis
                         true, // polarity
                         false, // don't latch
                         true); // enabled!

 Serial.print("Accelerometer range set to: ");
 switch (lsm6ds_2.getAccelRange()) {
  case LSM6DS_ACCEL_RANGE_2_G:
    Serial.println("+-2G");
    break;
  case LSM6DS_ACCEL_RANGE_4_G:
    Serial.println("+-4G");
    break;
  case LSM6DS_ACCEL_RANGE_8_G:
    Serial.println("+-8G");
    break;
  case LSM6DS_ACCEL_RANGE_16_G:
    Serial.println("+-16G");
    break;
 }

 Serial.print("Accelerometer data rate set to: ");
 switch (lsm6ds_2.getAccelDataRate()) {
  case LSM6DS_RATE_SHUTDOWN:
    Serial.println("0 Hz");
    break;
  case LSM6DS_RATE_12_5_HZ:
    Serial.println("12.5 Hz");
    break;
  case LSM6DS_RATE_26_HZ:
    Serial.println("26 Hz");
    break;
  case LSM6DS_RATE_52_HZ:
    Serial.println("52 Hz");
    break;
  case LSM6DS_RATE_104_HZ:
    Serial.println("104 Hz");
    break;
  case LSM6DS_RATE_208_HZ:
    Serial.println("208 Hz");
    break;
  case LSM6DS_RATE_416_HZ:
    Serial.println("416 Hz");
    break;
  case LSM6DS_RATE_833_HZ:
    Serial.println("833 Hz");
    break;
  case LSM6DS_RATE_1_66K_HZ:
    Serial.println("1.66 KHz");
    break;
```

This document corresponds to the V2 of the IMU

```
  case LSM6DS_RATE_3_33K_HZ:
   Serial.println("3.33 KHz");
   break;
  case LSM6DS_RATE_6_66K_HZ:
   Serial.println("6.66 KHz");
   break;
 }

 Serial.print("Gyro range set to: ");
 switch (lsm6ds_2.getGyroRange()) {
  case LSM6DS_GYRO_RANGE_125_DPS:
   Serial.println("125 degrees/s");
   break;
  case LSM6DS_GYRO_RANGE_250_DPS:
   Serial.println("250 degrees/s");
   break;
  case LSM6DS_GYRO_RANGE_500_DPS:
   Serial.println("500 degrees/s");
   break;
  case LSM6DS_GYRO_RANGE_1000_DPS:
   Serial.println("1000 degrees/s");
   break;
  case LSM6DS_GYRO_RANGE_2000_DPS:
   Serial.println("2000 degrees/s");
   break;
  case ISM330DHCX_GYRO_RANGE_4000_DPS:
   Serial.println("4000 degrees/s");
   break;
 }

 Serial.print("Gyro data rate set to: ");
 switch (lsm6ds_2.getGyroDataRate()) {
  case LSM6DS_RATE_SHUTDOWN:
   Serial.println("0 Hz");
   break;
  case LSM6DS_RATE_12_5_HZ:
   Serial.println("12.5 Hz");
   break;
  case LSM6DS_RATE_26_HZ:
   Serial.println("26 Hz");
   break;
  case LSM6DS_RATE_52_HZ:
   Serial.println("52 Hz");
   break;
  case LSM6DS_RATE_104_HZ:
   Serial.println("104 Hz");
   break;
  case LSM6DS_RATE_208_HZ:
   Serial.println("208 Hz");
   break;
  case LSM6DS_RATE_416_HZ:
   Serial.println("416 Hz");
   break;
  case LSM6DS_RATE_833_HZ:
   Serial.println("833 Hz");
   break;
  case LSM6DS_RATE_1_66K_HZ:
   Serial.println("1.66 KHz");
   break;
  case LSM6DS_RATE_3_33K_HZ:
   Serial.println("3.33 KHz");
   break;
  case LSM6DS_RATE_6_66K_HZ:
   Serial.println("6.66 KHz");
   break;
 }

 lis3mdl_2.setDataRate(LIS3MDL_DATARATE_155_HZ);
 Serial.print("Magnetometer data rate set to: ");
 switch (lis3mdl_2.getDataRate()) {
  case LIS3MDL_DATARATE_0_625_HZ: Serial.println("0.625 Hz"); break;
```

This document corresponds to the V2 of the IMU

```
  case LIS3MDL_DATARATE_1_25_HZ: Serial.println("1.25 Hz"); break;
  case LIS3MDL_DATARATE_2_5_HZ: Serial.println("2.5 Hz"); break;
  case LIS3MDL_DATARATE_5_HZ: Serial.println("5 Hz"); break;
  case LIS3MDL_DATARATE_10_HZ: Serial.println("10 Hz"); break;
  case LIS3MDL_DATARATE_20_HZ: Serial.println("20 Hz"); break;
  case LIS3MDL_DATARATE_40_HZ: Serial.println("40 Hz"); break;
  case LIS3MDL_DATARATE_80_HZ: Serial.println("80 Hz"); break;
  case LIS3MDL_DATARATE_155_HZ: Serial.println("155 Hz"); break;
  case LIS3MDL_DATARATE_300_HZ: Serial.println("300 Hz"); break;
  case LIS3MDL_DATARATE_560_HZ: Serial.println("560 Hz"); break;
  case LIS3MDL_DATARATE_1000_HZ: Serial.println("1000 Hz"); break;
 }

 lis3mdl_2.setRange(LIS3MDL_RANGE_4_GAUSS);
 Serial.print("Range set to: ");
 switch (lis3mdl_2.getRange()) {
  case LIS3MDL_RANGE_4_GAUSS: Serial.println("+-4 gauss"); break;
  case LIS3MDL_RANGE_8_GAUSS: Serial.println("+-8 gauss"); break;
  case LIS3MDL_RANGE_12_GAUSS: Serial.println("+-12 gauss"); break;
  case LIS3MDL_RANGE_16_GAUSS: Serial.println("+-16 gauss"); break;
 }

 lis3mdl_2.setPerformanceMode(LIS3MDL_MEDIUMMODE);
 Serial.print("Magnetometer performance mode set to: ");
 switch (lis3mdl_2.getPerformanceMode()) {
  case LIS3MDL_LOWPOWERMODE: Serial.println("Low"); break;
  case LIS3MDL_MEDIUMMODE: Serial.println("Medium"); break;
  case LIS3MDL_HIGHMODE: Serial.println("High"); break;
  case LIS3MDL_ULTRAHIGHMODE: Serial.println("Ultra-High"); break;
 }

 lis3mdl_2.setOperationMode(LIS3MDL_CONTINUOUSMODE);
 Serial.print("Magnetometer operation mode set to: ");
 switch (lis3mdl_2.getOperationMode()) {
  case LIS3MDL_CONTINUOUSMODE: Serial.println("Continuous"); break;
  case LIS3MDL_SINGLEMODE: Serial.println("Single mode"); break;
  case LIS3MDL_POWERDOWNMODE: Serial.println("Power-down"); break;
 }

 lis3mdl_2.setIntThreshold(500);
 lis3mdl_2.configInterrupt(false, false, true,  // enable z axis
                 true,  // polarity
                 false,  // don't latch
                 true);  // enabled!

}

void loop() {

 sensors_event_t accel, gyro, mag, temp;
 sensors_event_t accel_2, gyro_2, mag_2, temp_2;

 // Serial.println("First IMU");
 //  /* Get new normalized sensor events */
 lsm6ds.getEvent(&accel, &gyro, &temp);
 lis3mdl.getEvent(&mag);

 lsm6ds_2.getEvent(&accel_2, &gyro_2, &temp_2);
 lis3mdl_2.getEvent(&mag_2);

 /* Display the results (acceleration is measured in m/s^2) */
 Serial.print("\t\tAccel X: ");
 Serial.print(accel.acceleration.x, 4);
 Serial.print(" \tY: ");
 Serial.print(accel.acceleration.y, 4);
 Serial.print(" \tZ: ");
 Serial.print(accel.acceleration.z, 4);
 Serial.print(" \tm/s^2 ");

 Serial.print("\t\tAccel X (IMU2): ");
 Serial.print(accel_2.acceleration.x, 4);
```

This document corresponds to the V2 of the IMU

```
  Serial.print(" \tY: ");
  Serial.print(accel_2.acceleration.y, 4);
  Serial.print(" \tZ: ");
  Serial.print(accel_2.acceleration.z, 4);
  Serial.println(" \tm/s^2 ");

  /* Display the results (rotation is measured in rad/s) */
  Serial.print("\t\tGyro  X: ");
  Serial.print(gyro.gyro.x, 4);
  Serial.print(" \tY: ");
  Serial.print(gyro.gyro.y, 4);
  Serial.print(" \tZ: ");
  Serial.print(gyro.gyro.z, 4);
  Serial.print(" \tradians/s ");

  Serial.print("\tGyro  X (IMU2): ");
  Serial.print(gyro_2.gyro.x, 4);
  Serial.print(" \tY: ");
  Serial.print(gyro_2.gyro.y, 4);
  Serial.print(" \tZ: ");
  Serial.print(gyro_2.gyro.z, 4);
  Serial.println(" \tradians/s ");

  /* Display the results (magnetic field is measured in uTesla) */
  Serial.print(" \t\tMag   X: ");
  Serial.print(mag.magnetic.x, 4);
  Serial.print(" \tY: ");
  Serial.print(mag.magnetic.y, 4);
  Serial.print(" \tZ: ");
  Serial.print(mag.magnetic.z, 4);
  Serial.print(" \tuTesla ");

  Serial.print(" \tMag   X (IMU2): ");
  Serial.print(mag_2.magnetic.x, 4);
  Serial.print(" \tY: ");
  Serial.print(mag_2.magnetic.y, 4);
  Serial.print(" \tZ: ");
  Serial.print(mag_2.magnetic.z, 4);
  Serial.println(" \tuTesla ");

  Serial.print("\t\tTemp   :\t\t\t\t\t");
  Serial.print(temp.temperature);
  Serial.print(" \tdeg C");
  // Serial.print("");

  Serial.print("\t\tTemp  (IMU2) :\t\t\t\t\t");
  Serial.print(temp_2.temperature);
  Serial.println(" \tdeg C");
  Serial.println();

  delay(1000);

}
```