Computer Systems Design Lab

SYSC 4805 (Fall 2022)

Instructor: Dr. Mostafa Taha
Carleton University
Department of Systems and Computer Engineering

Lab 3: Analog Discovery, Laser ToF Sensor and IMU Sept 22nd and 23rd, 2022

In this lab, we will conduct the following Tasks:

- Task 1: Introducing the Analog Discovery 2
  - Experiment 1: Test the AD2 by Loopback
  - Experiment 2: Using AD2 as an Oscilloscope
- Task 2: Testing the VL53L1X Time-of-Flight Distance Sensor
  - Experiment 1: Using one VL53L1X Sensor over I2C
  - Experiment 2: Using two VL53L1X Sensors over I2C
- Task 3: Testing the IMU Sensor.
  - Experiment 1: Using the LSM6 Accelerometer and Gyroscope
  - Experiment 2: Using the LIS3MDL Magnetometer
  - Bonus Experiment: Display movement on Python Visualization

We will start by introducing the Analog Discovery kit, where we test the AD2 as an independent unit, and also as an Oscilloscope. Then, for each of the following tasks, we start by introducing the sensor and its technical description, then a couple of experiments to test its proper functionality and interfacing with the Arduino Due board.

Report Instructions

General:

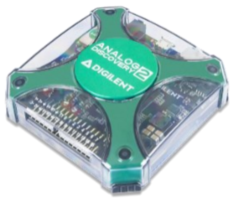
- The report should be submitted before the beginning of the next lab. After that, there will be a penalty of 25% if the report submitted in the following 24 hours. If the report submitted 24 to 48 hours late, it would receive a penalty of 50%. No report will be accepted afterwards.
- The cover page should include your group names, the lab number, and the lab date.
- Organization is extremely important: Divide your report into sections, use captions for figures and headings for tables, …etc.

Lab 3: Your report should include the following

- All the Table completed.
- Answer to all the discussion questions in the experiments of each task.
- The required screenshots and pictures in the experiments of each task.

Task 1: Introducing the Analog Discovery 2

Prepared by Mahmoud Sayed



Introduction

Analog Discovery 2 (AD2) is a pocket size instrument from Digilent that can fairly replace an entire electronics lab, within low limits in frequency, voltage, and power. It can work as oscilloscope, waveform generator, voltmeter, DC power supply, spectrum analyzer, network analyzer, logic analyzer, and much more. In the following experiment, we will use the AD2 to read the I2C signals between the sensor and the Arduino Due board to check whether the sensor is working properly or not.

How it works

In typical configuration, AD2 lies between the circuit under inspection and a computer as shown in Fig. 1. We connect wires and/or coaxial cables (with BNC connectors) to carry analog/digital signals from and to the circuit under testing. Meanwhile, the AD2 is connected to a computer by a USB cable to carry commands and configuration parameters to AD2, and reply with measurement values from the AD2 to the computer. Digilent provides a software called “WaveForms” that works as a GUI for the AD2 among a group of similar supported instruments. Through WaveForms the user can do the required configurations to any module on the AD2, e.g., select trigger source for oscilloscope, waveform type for waveform generator, ...etc. The software also provides graphical display to show the measured quantities in real time (Oscilloscope, spectrum analyzer, ...etc.). User scripts can be also implemented (using JavaScript) on WaveForms to customize the AD2 tasks, which adds a great flexibility to its performance to match the user requirements. Digilent also provides WaveForms SDK with extensive API that allows controlling the AD2 and creating custom PC applications. The SDK allows users to use different programming languages like C/C++, C#, Python, and Visual Basic and provides an extension to Matlab runtime environment.

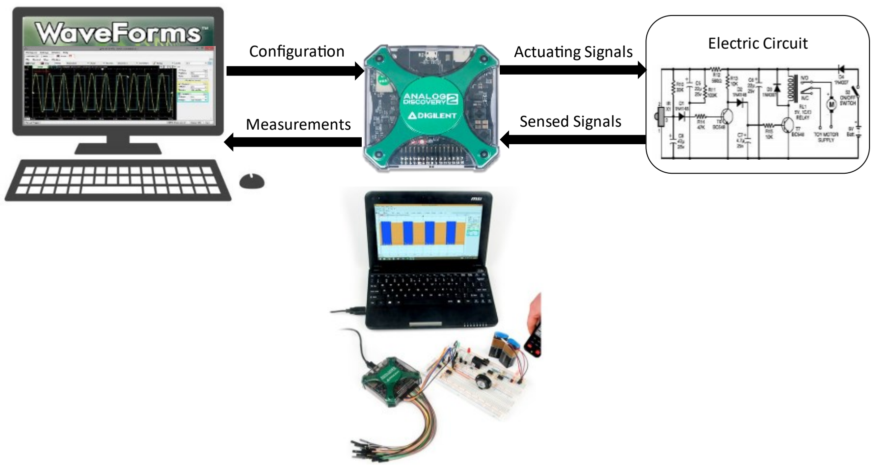


Fig. 1 Working Concept of Analog Discovery 2

Specifications

For the circuit side, the AD2 provides the following interfaces as shown in Fig. 2:

1- **Analog Input Interface**: Two independent analog input channels are provided. These two channels are shared between multiple modules in the AD2 like the scope and voltmeter.
2- **Analog Output Interface**: Two independent analog outputs channels are provided. The two output channels are used mainly by the waveform generator, but they are also used by other modules like the network analyzer.
3- **Digital IO Interface**: 16 digital IO interfaces are provided. They are used mainly for the logic analyzer module, and they are also shared with more advanced functions like serial protocols analyzers.
4- **Power Supply Interface**: Two power supply interfaces are provided, a positive power supply with range 0.5V to 5V and a negative power supply with range -5V to -0.5V.
5- **Trigger Interface**: Two external trigger inputs for the scope functionality.

In the following, a summary of the specs of the commonly used modules in the AD2 is presented. Please refer to the reference manual [Analog Discovery 2 Reference Manual - Digilent Reference] for more detailed information about all modules.

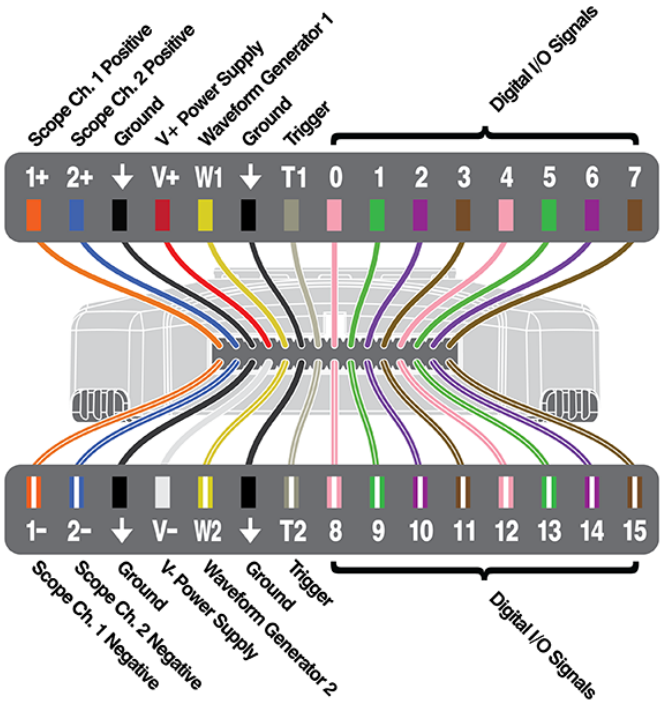


Fig. 2 Pins description of AD2

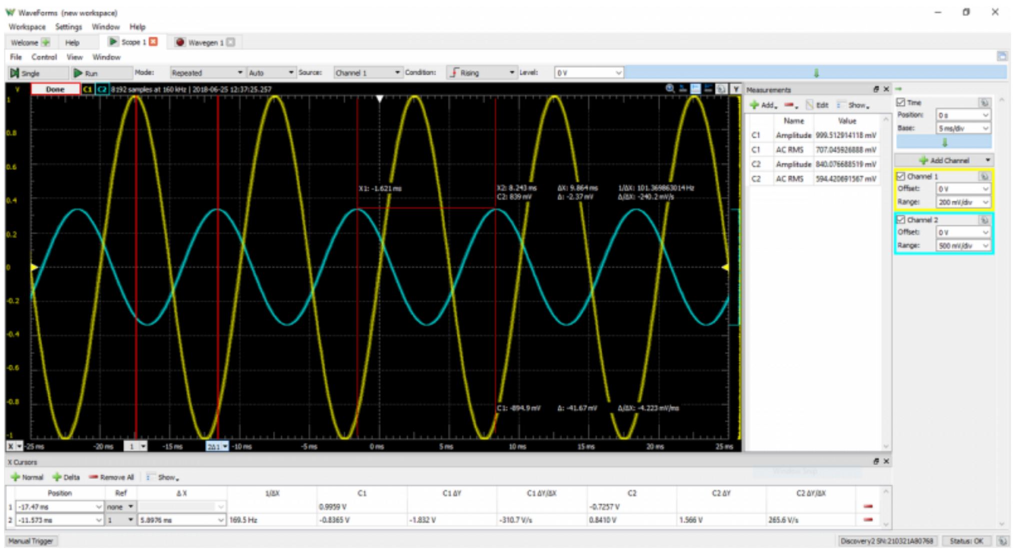
As a 100 MS/s Oscilloscope Scope

The AD2 is used with WaveForms' Oscilloscope instrument to capture analog input data via the analog input ("Scope") channels, using either BNC cables or jumper wires. When this instrument is used, the AD2's analog input channels act as a two-channel, 14-bit, 100 MS/s oscilloscope. A screenshot from typical scope view is presented in Fig. 3.

When used with jumper wires, the oscilloscope channels' + and - pins are differentially paired. The - pins can be attached to a non-ground circuit net, but the circuit under test must still share a common ground with the AD2. However, when using BNC cables, the oscilloscope channels are single-ended and the circuit under test must still share a common ground with the AD2. With BNC cables, the analog input channels have a substantially higher bandwidth than with jumper wires.

In the following experiment, we will only use the jumper wires provided with the kit. We provide an introduction for using the AD as an Oscilloscope as a quick debug tool which can replace the benchtop Oscilloscope while you develop the term project. This is only possible as the clock frequency of the Arduino Due board is relatively small (84 MHz).

Since the AD2's analog input channels are shared, the Oscilloscope instrument cannot be used at the same time as the Voltmeter, Data Logger, Spectrum Analyzer, Network Analyzer, or Impedance Analyzer instruments.



	Name	Value
C1	Amplitude	999.512914118 mV
C1	AC RMS	707.045926888 mV
C2	Amplitude	840.076688519 mV
C2	AC RMS	594.420691567 mV

Time — Position: 0 s; Base: 5 ms/div
Channel 1 — Offset: 0 V; Range: 200 mV/div
Channel 2 — Offset: 0 V; Range: 500 mV/div

	Position	Ref	ΔX	1/ΔX	C1	C1 ΔY	C1 ΔY/ΔX	C2	C2 ΔY	C2 ΔY/ΔX
1	-17.47 ms	none			0.9959 V			-0.7257 V		
2	-11.573 ms	1	5.8976 ms	169.5 Hz	-0.8365 V	-1.832 V	-310.7 V/s	0.8410 V	1.566 V	265.6 V/s

Fig. 3 Screenshot from Scope module in WaveForms

Waveform generator

WaveForms' Wavegen instrument outputs analog voltage waves via either BNC or Jumper wires. Wavegen converts 14-bit digital samples to analog at a rate of up to 100 MS/s on each of two channels. When Wavegen is used, the AD2's analog output channels act as an Arbitrary Waveform Generator. The instrument supports everything from simple waveforms like Sine and Triangle waves, up to more complicated functions like AM and FM modulation. Custom sets of samples can be defined by the user in applications like Excel and imported to WaveForms. Each waveform generator channel is considered a single ended pin; however, a connected circuit must share a ground with the Analog Discovery 2. A screenshot from the module's window is provided in Fig. 4

Since the AD2's analog output channels are shared, the Waveform Generator instrument cannot be used at the same time as the Network Analyzer, or Impedance Analyzer instruments.

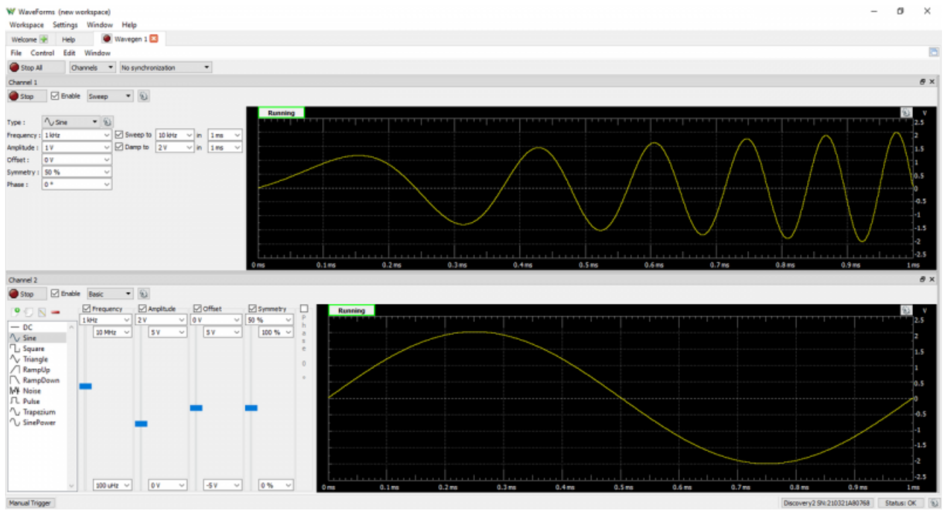
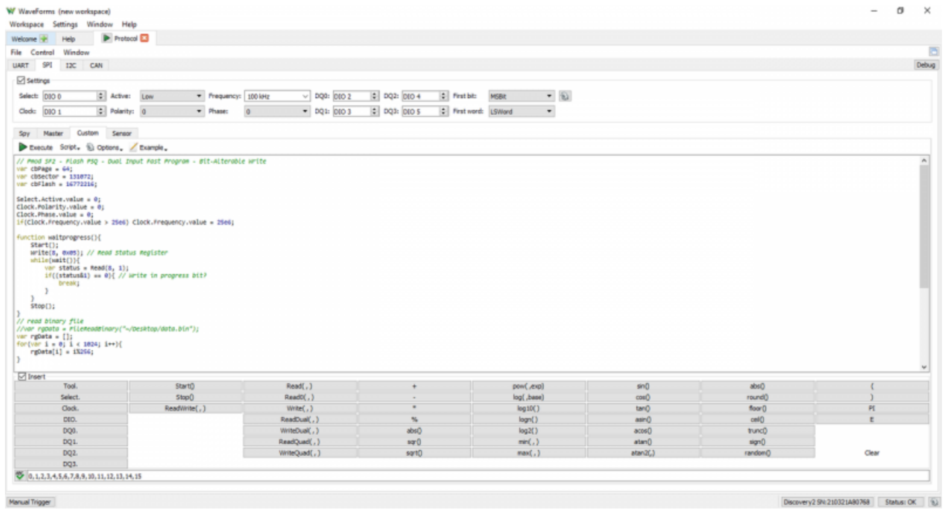

Type: Sine
Frequency: 1 kHz
Amplitude: 1 V
Offset: 0 V
Symmetry: 50 %
Phase: 0 °
Sweep to: 10 kHz in 1 ms
Ramp to: 2 V in 1 ms

Fig. 4 Screenshot from Wavegen interface in WaveForms.

Protocol Analyzer

AD2 can be used with WaveForms' "Protocol" instrument to work with common communications protocols. UART, SPI, I2C, and CAN transactions can be received, transmitted, and/or spied upon by the Analog Discovery 2 using any of the digital input/output channels. Custom scripts can be written within the Protocol Analyzer instrument to generate sequences of SPI or I2C transactions. Since it uses the same hardware resources as the Logic Analyzer and Pattern Generator instruments, the Protocol Analyzer cannot be used at the same time as these instruments. A screenshot of the protocol analyzer interface is shown in Fig. 5.


Select: DIO 0 Active: Low Frequency: 100 kHz DQ0: DIO 2 DQ2: DIO 4 First bit: MSBit
Clock: DIO 1 Polarity: 0 Phase: 0 DQ1: DIO 3 DQ3: DIO 5 First word: LSWord

```
// Pmod SF2 - Flash PSQ - Dual Input Fast Program - Bit-Alterable Write
var cbPage = 64;
var cbSector = 131072;
var cbFlash = 16772216;

Select.Active.value = 0;
Clock.Polarity.value = 0;
Clock.Phase.value = 0;
if(Clock.Frequency.value > 25e6) Clock.Frequency.value = 25e6;

function waitprogress(){
    Start();
    Write(8, 0x05); // Read Status Register
    while(wait()){
        var status = Read(8, 1);
        if((status&1) == 0){ // Write in progress bit?
            break;
        }
    }
    Stop();
}
// read binary file
//var rgData = FileReadBinary("~/Desktop/data.bin");
var rgData = [];
for(var i = 0; i < 1024; i++){
    rgData[i] = i%256;
}
```

Fig. 5 Screenshot from the Protocol Analyzer Interface in Waveforms.

Experiments

In the following experiments, we will try to investigate the basic functionality of the AD2 and WaveForms application to get familiar with the instrument.

Experiment 1: Test the AD2 by Loopback

In this experiment, the AD2 will produce a pre-configured waveform on the analog outputs of the AD2 and measure the analog signals back into the scope interface. We target three goals: 1- test the AD2 as a standalone unit, 2- test the AD2 as a function generator, and 3- test the Oscilloscope functionality of the AD2.

Require Equipment

- Analog Discover 2
- Jumper wires.

Procedure

1- Download, install and run the Digilent WaveForms from: https://digilent.com/shop/software/digilent-waveforms/
   Remember to install the USB drivers.
2- Make connections:
   a. +1 pin to W1 pin.
   b. +2 pin to W2 pin.
   c. -1 and -2 to the ↓ Ground pin.
3- From the Welcome Screen, select the wavegen module of the Digilent WaveForms software. Activate the two channels, by selecting them from the Channels menu.
   Configure the wavegen channels to generate the following.
   a. Channel 1: A sine wave with frequency of 1 KHz and 1 V amplitude.
   b. Channel 2: A square wave with period of 500 usec (frequency of 2kHz), the minimum value is 0V, the maximum value is 2V, and the duty cycle (symmetry) is 25%.
4- From the Welcome screen, select the Scope. Configure the scope module as follows
   a. Source of trigger (at the top) to be Channel 1
   b. Time scale to display 5 periods of signal 1 on horizontal axis, and 500 mv per division for vertical axes of both signals.
   Bonus: From the Welcome screen, use the Specturm tool to display the FFT of both signals, does it match your expectations?

Take a screenshot and include it in your report.

Experiment 2: Using AD2 as an Oscilloscope

In this experiment, we will use the AD2 as an Oscilloscope to display the trigger signal from Lab 2.

Require Equipment

- Analog Discover 2
- Arduino Due Board
- Jumper wires.

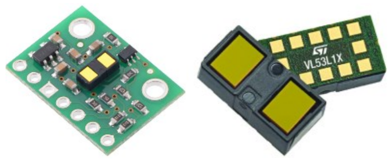
Procedure

1- Connect the Arduino Due board to your computer, and download the code used in Lab 2, Task 3, Procedure for Exp. 1.C: Using the Timer Counter Unit to generate the trigger signal.
2- Disconnect the Arduino Due Board.
3- Connect the Analog Discovery 2,
    a. From the Welcome screen, select the Supplies tool
    b. Make sure the Master Enable is OFF. ✗ Master Enable is Off .
    c. Activate the V+ channel to green check ✓ Positive Supply (V+) Rdy .
    d. Deactivate the V- channel ✗ Negative Supply (V-) Off .
    e. Set the output voltage of the V+ channel to 5V.
    f. Connect the V+ pin and any ↓ ground pin from the AD2, to the + and – pins of the power plug in your kit. Use the screwdriver to gently tight the screws of the power plug.

    g. Connect the power plug to the Arduino Due Board. Remember, you should NEVER try to power the Arduino Due board from the 5V output pin. The Arduino due board can only be powered from the power socket or a USB cable.
4- Make connections:
    a. ↓ pin of the AD2 to GND pin on the Arduino Due board.
    b. +1 pin of the AD2 to pin 2 of the Arduino Board.
    c. -1 pin of the AD2 any Ground pin on the AD2 itself or the Arduino Board.
    d. Enable the Master Enable of the AD2 supply ✓ Master Enable is On
5- Configure the AD2 Scope tool as follows:
    a. Channel 2: Off.
    b. Channel 1: Trigger from Channel 1, rising edge. Time scale as 50 usec/div, 0 offset. Voltage scale as 1 V/div, 0 Offset. This should show a clear pulse.
    c. Try again using the Time scale at 10 ms/div, with position shifted by +30 ms.
6- Take a screenshots and include it in your report.

Task 2: Testing the VL53L1X Time-of-Flight Distance Sensor

Prepared by Mahmoud Sayed



Introduction

The VL53L1X is a long rage distance sensor from ST® that is capable of measuring distances up to 4m in suitable conditions. Its ranging mechanism is very similar to the Radar technology, but with using LASER with much shorter wavelengths in the nanometer range (Infrared or IR range) compared to the conventional radars that work in the radio frequency range. The sensor transmits a pulse of IR and measures its Time-of-Flight, which is basically the total roundtrip time of the ranging pulse to travel from the emitter to the object and back to the sensor again to calculate the object's distance from the sensor. VL53L1X is a very compact smart module with a dimension of less than 0.5 cm as you can see in the figure above (right); however, this tiny size and surface-mount fabrication may make its handling hard for testing and quick integration with hobbyist projects. Thus, we use a carrier Printed Circuit Board (PCB) from Pololu Robotics and Electronics® to provide connections with the VL53L1X using standard pins spacing and to provide suitable electrical interfacing. The relevant datasheets are as follows:
- The VL53L1X chip: https://www.st.com/resource/en/datasheet/vl53l1x.pdf
- The Carrier PCB board: https://www.pololu.com/product/3415
-The I2C bus specifications: https://www.nxp.com/docs/en/user-guide/UM10204.pdf

Features

The VL53L1X comes with a set of unique features and capabilities. There is a microcontroller integrated in the sensor's circuit, which provides a wide range of programmable options for the user to choose based on their project's requirements. Here is a summary of the main features.

- **Fast and accurate long-distance ranging**: The module can measure distances up to 4 m and with frequency of up to 50 Hz.
- **Typical Field-of-View (FoV)**: 27°
- **Programmable region-of-interest (ROI) size** on the receiving array, allowing the sensor FoV to be reduced.
- **Programmable ROI position** on the receiving array, providing multizone operation control from the host
- **I^{2}C Interface** which allows easy communication with host controllers.
- **Shutdown and Interrupt pins** that provide more control options to the host controller.

The sensor provides 3 different distance modes: short, medium, and long modes. As the name suggests, the long mode provides the maximum range (400 cm) but it is sensitive to ambient light compared to the short mode which measures up to 130 cm but it is much more immune to ambient light. In addition to the distance modes, the programmable interface of the sensor allows the user to configure numerous options in the sensor to optimize its performance based on the requirements and surrounding situation.

For example, the user is able is control the "timing budget", the "inter measurement" period, the "Region-Of-Interest" size and location, and other features. You are encouraged to read the sensor's datasheet linked in the Introduction.

The sensor communicates with the host controller using the I^2C interface, which is a popular half duplex serial communications protocol that uses only two wires to send and receive information: Serial Clock (SCL) and Serial Data (SDA). Being a protocol, I^2C has a defined control sequence, message length, message shape, acknowledgment sequence, etc. that must be understood by all communicating ends for the communication to success. A device is said to have I^2C interface usually when it is equipped with a special hardware circuit to handle the I^2C protocol requirements. Both the VL53L1X and the Arduino Due have I^2C interfaces which means they can communicate easily using this protocol. Fig.1 shows the typical connection of an I^2C bus with one master and multiple slaves. The master controls the clock line, chooses which slave to communicate with, and chooses the direction of the communication. The reader is encouraged to read the I^2C bus specifications linked in the Introduction.

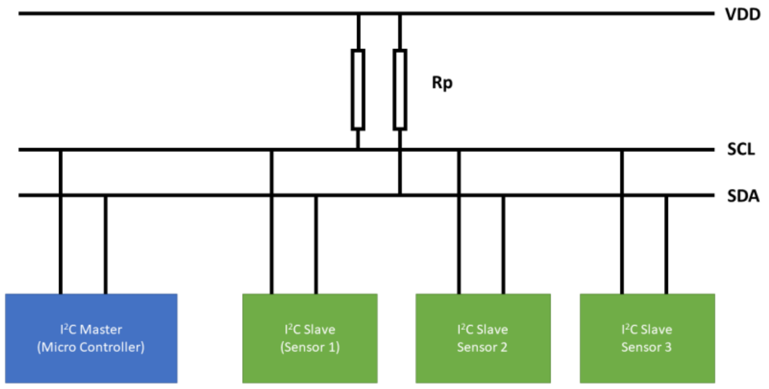


Fig. 6 Typical I^2C bus connection with one master and multiple slaves.

While the nominal operating voltage of the VL53L1X is 2.8V, which makes the interfacing with 5V microcontrollers not straight forward, its maximum rating voltage is 3.5V which makes it compatible with the Arduino Due directly. More importantly, the expansion board implements additional electrical interface circuitry that makes the module's voltage compatibility range extends from 2.6V to 5.5V which is a suitable for wide range of microcontrollers. It should be noted that the I^2C interface pins are usually open-drain or open-collector pins (why?) which means external pull-up resistors must be connected on both lines as shown in Fig. 1. However, the sensor's expansion board integrates these resistors to pull the lines to the power rail, so it is not required to connect them externally. The basic specifications of the sensor integrated within the expansion board is described in Table 1.

Table 1: VMA330 module specifications.

Operating Voltage	Min: 2.6V to Max: 5.5V
Maximum Range	400 cm
Minimum Range	4 cm
Resolution	1 mm
I/O Interface	VIN, VDD, GND, SDA, SCL, XSHUT, GPIO1
Dimensions	17.8 × 12.7 × 2.22 mm

Experiment 1: Using one VL53L1X Sensor over I2C

In this experiment, you should test the basic functionality and performance of VL53L1X ToF distance measurement sensor, and confirm its operation using the Analog Discovery 2.

Required Equipment:

1- VL53L1X sensor (integrated in expansion board)
2- Arduino Due.
3- Analog Discovery 2 kit.
4- White paper (obstacle).
5- Breadboard.
6- Jumper wires.

Procedure:

1- **Install the VL53L1X library from Pololu:**
ST® does not provide a mapping for the internal registers of the sensor. However, they provide a detailed API that can be used to control it. While information about the registers mapping can be extracted from the provided API, ST® recommends using its API directly to ensure the proper operation according to its datasheet specifications. Also, Pololu provides a simple API that is lighter & easier to use but does not contain all the options, accessibility, and robust error checking capabilities available in the original ST® implementation.
In this experiment, we use a the VL53L1X library from Pololu:
   a. In the Arduino IDE, open the "Sketch" menu, select "Include Library", then "Manage Libraries...".
   b. Search for "VL53L1X".
   c. Click on the “VL53L1X by Pololu” entry in the list. Currently in Version 1.3.1.
   d. Click "Install".

2- **Make the connections**: between the Arduino Due and the sensor.
   - 3.3V pin from the Arduino Due Board to the VIN pin from the VL53L1X Sensor.
   - GND pin from the Arduino Due Board to GND from the VL53L1X Sensor, as well as the ↓ ground pin of the AD2.
   - Pin (20 SDA) and (21 SCL) of the Arduino Board to the SDA, and SCL from the VL53L1X Sensor, as well as pin DIO 1 and DIO 0 of the AD2 kit, respectively.
   - Leave VDD, GPIO1, and XHSUT sensor pins floating in this experiment.
   - **Caution: Make sure that you remove the protective liner if it exists**

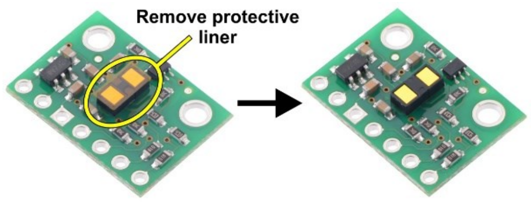


Fig. 7 The sensor might ship with a protective liner covering the sensor IC that must be removed before use. (Figure from https://www.pololu.com/product/3415)

3- **Try Sample codes**: Three example sketches are available that show how to use the library. You can access them from the Arduino IDE by opening the "File" menu, selecting "Examples", and then selecting "VL53L1X". If you cannot find these examples, the library was probably installed incorrectly, and you should retry the installation instructions above. Test the “Continuous” Example, as well as the “ContinuouswithDetails”.

4- **Confirm Operation using AD2:** While the Sensor is sending data to the Arduino Due over the I2C but, open the WaveForms software of the AD2. Select the “Protocol” tool from the Welcome screen. Confirm that you are working in a “Spy” mode, the SCL is selected as “DIO 0”, and SDA is selected as “DIO 1” and the default frequency of 100 kHz.
Hit “Receive”, and you should read data packets in the format of
“Start, xx x xx x xx x, Stop”. This confirms that the sensor module is working and talking to the Arduino Due board. To understand these data packets, you will need to refer to the API interface of the sensor module.

5- **Modify the code**: Modify the code to have a delay of 500 msec between each reading and to test other configurations. Change the “distance mode” to: <u>Short</u>, <u>Medium</u>, and <u>Long</u>
Note that the Timing Budget for each mode should at least be: 20 ms for short distance mode and 33 ms for medium and long distance modes.

6- **Measure and Record**: Now with the sensor functioning as expected, it is time for detailed measurements. For each of the distance mode in step 4 (Short, Medium, and Long), move the paper in front of the sensor, record the measured minimum and maximum distances, and the range in between, with steps of 10 cm. The results should be compared with the actual distances to determine the error (actual – measured).

7- **Plot the results**: You should make three figures, one for each mode (Short, Medium, and Long). Each figure should contain 2 curves: one for the actual distance (on x-axis) Vs measured distance (on y-axis) and one for the error curve.

Experiment 2: Using two VL53L1X Sensors over I2C

In this experiment, you should interface 2 sensors with the Arduino Due, they may work together to confirm the distance to a particular obstacle, or as x-axis and y-axis distances of a robot for localization purposes. To be able to communicate using the I^{2}C bus (one master and two slaves), the sensors must be configured to have different slave addresses i.e. at least one of the sensors will have to change its address from the default. You will need to use the hardware disable pin XSHUT to disable one of the sensors while changing the address of the other before bringing both of them online. Your sketch should display the distances from both sensors on the serial monitor.

Required Equipment:

1- Two VL53L1X sensors (integrated in expansion board), to name them as Sensor A and Sensor B.
2- Arduino Due.
3- White paper (obstacle) – 2 of them.
4- Breadboard.
5- Jumper wires.

Procedure:

1- **Make the connections**: between the Arduino Due and the sensor.

- VIN from both Sensors to the 3.3V pin from the Arduino Due Board.
- GND from both Sensors to the GND pin from the Arduino Due Board.
- SDA, and SCL from both Sensors to pin (20 SDA) and (21 SCL) of the Arduino Board.
- XHSUT pin of Sensor A to pin 2 of the Arduino Due Board.
- XHSUT pin of Sensor B to pin 3 of the Arduino Due Board.
- Leave VDD, and GPIO1 sensor pins floating.

2- **Try sample code:** Use the example code "ContinuousMultipleSensors" from "File" menu, selecting "Examples", and then selecting "VL53L1X".
   a. Modify the code to add 500 msec between each reading.
   b. Modify the sensorCount to 2;
   c. Modify the xshutPins array to {2, 3} to match Sensor A and Sensor B respectively.
   d. Change the Distance Mode and the Timing Budgetof the two sensors to Short and 20 msec respectively.

   If the sensor is not working properly, you can use the AD2 to "Spy" on the I2C data bus as referenced in Task 2, Experiment 1.

3- Take some measurements:
   a. Set two boundaries at a 90^o angle to form a testing arena.
   b. Put the two sensors close to each other and measure the distance from the sensors to the two boundaries at 5 different locations.
   c. Take a picture of your area to include it in your report, along with a screenshot of the terminal monitor readings.

Discussion

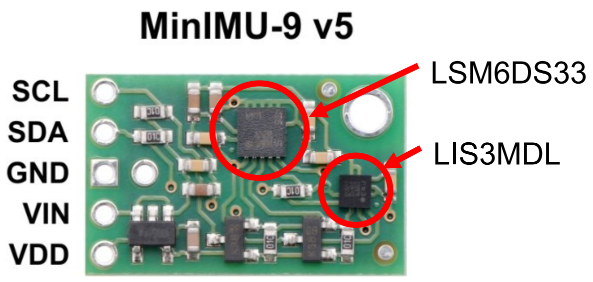
1- What is the theoretical maximum number of slaves that can be connected to a standard I^{2}C bus? Explain.

2- Does changing the size of the Region of Interest (ROI) affects the maximum range of the VL53L1X? If yes, is it a direct or inverse relation? Why do you think so?

3- Does the reflectance of the detected object affect the maximum range of the VL53L1X sensor? Why?

Task 2: Testing the MinIMU-9 v5 IMU Unit

Prepared by Mahmoud Sayed



Introduction

An Inertial Measurement Unit (IMU) is an electronic device that is used to measure or detect the orientation (and additionality attitude for some units) of a body by measuring static and dynamic inertial quantities. Different IMUs have the different capabilities and measurement tools, their measured quantities can include linear acceleration, angular velocity, magnetic field direction, pressure, and temperature. The IMUs measure these different quantities by adopting a set of measurement tools like accelerometers, gyroscopes, magnetometers, barometers, and thermometers. The MinilMU-9 v5 module is a breakout board that hosts three tools: *Accelerometer* and *Gyroscope* inside the LSM6DS33 MEMS IC, and a *magnetometer* in the LIS3MDL IC. The measurements of these tools together can be combined to provide a relatively accurate orientation information of the module in 3D space. The datasheets of these two chips are here:
- LSM6DS33: https://www.pololu.com/file/0J1087/LSM6DS33.pdf
- LIS3MDL: https://www.pololu.com/file/0J1089/LIS3MDL.pdf
- The carrier PCB board: https://www.pololu.com/product/2738

Orientation Angles

A body's change of orientation (rotation) in space relative to an initial reference direction can be completely described by the respective change of its Euler's angles. For simplicity, consider the plane in Fig. 8, a change in its orientation with reference to some initial direction can be precisely described by the change in its Yaw, Roll, and Pitch angles. One of the major tasks of IMUs is to detect the change in orientation of a body by reporting the change in these angles.

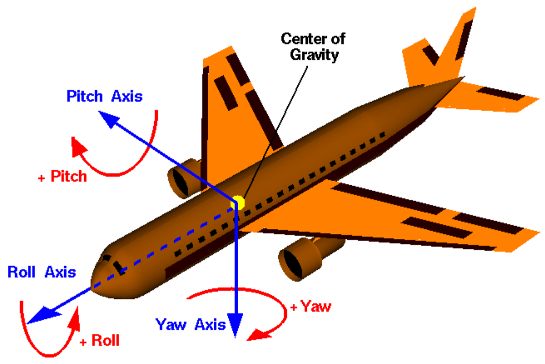


Fig. 8 A graphical description of the 3 rotation angles or Euler's angles (https://www.grc.nasa.gov/www/k-12/airplane/rotations.html)

Accelerometer

The accelerometer is a tool that measures the "proper" *linear acceleration* of a body, which is the acceleration experienced by or "felt" by the body, relative to an inertial observer. The LSM6DS33 MEMS IC from ST®, provides a 3-axis Accelerometer, as shown in Fig.9.

For simplicity, we can look to the accelerometer as a tool that is able to measure or sense the force acting on a body, and it actually does that, so it can calculate its acceleration using the basic $F = ma$ equation with a known proof mass m. An accelerometer at rest on the ground would "feel" its weight as a force acting upon it from the ground, so it will measure an acceleration of 1 G upwards. On the other hand, a free-falling accelerometer will not sense any acceleration because it does not sense any force.

Hence, in addition to measuring the acceleration in any direction, the accelerometer can also be used to measure the absolute orientation of a body with respect to the Roll and Pitch axes.

If the LSM6DS33 sensor is set completely flat, parallel to the ground, in any Yaw angle, the Z (or -Z depending on which face is facing the ground) acceleration component should have a high value due to feeling the weight as an acting force facing down, while the X and Y components should be very low values (almost zero). See Fig.9 for the reference directions on the LSM6DS33 sensor. However, if you change the orientation of the LSM6DS33 sensor so that the shorter edge of the sensor is parallel to the group, while the longer edge is facing up, i.e., the Pitch angle is 90°, the X acceleration component should have a high value while the Y and Z should be low values. Similarly, if you change the orientation of the LSM6DS33 sensor so that the longer edge of the sensor is parallel to the group, while the shorter edge is facing up, i.e., the Roll angle is 90°, the Y acceleration component should have a high value while the Y and Z should be low values.

A change in the Roll or Pitch angles of the sensor will change the weight components distribution on its reference axes, which means the change in these angles can be precisely described by the change of the accelerometer readings. In other words, while the sensor is at rest, the absolute magnitude should always be the same i.e. $\sqrt{X_c^2 + Y_c^2 + Z_c^2}$ is constant. On the other hand, A change in the Yaw angles will not change any of the acceleration components, so the accelerometer is completely oblivious to changes in the Yaw angle which has an axis perpendicular to the earth's surface.

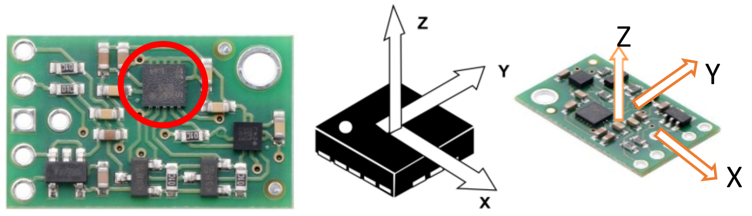


Fig. 9 Direction of the 3 detectable accelerations of the LSM6DS33 MEMS IC.

Now, you may start conducting Experiment 1.A.

Gyroscope

A gyroscope is a tool to measure *rotational or angular velocity* of a body. It does not measure linear velocity or acceleration. In fact, it is designed to eliminate the affects of linear velocity or acceleration on its measurements. The LSM6DS33 MEMS IC from ST®, provides a 3-axis gyroscope, as shown in Fig.10.

While the sensor is lying flat, parallel to the group, the Gyroscopes readings in the X and Y directions, as shown in Fig. 10, aid the accelerometer in calculating the Roll and Pitch angles of the vehicle, not only as absolute values provided by the accelerometer, but also as the velocity of change in these readings. The Gyroscopes reading in the Z direction aid the Magnetometer in calculating the Yaw angle of the vehicle, not only as an absolute value provided by the Magnetometer, but also as the velocity of change in this reading.

Note that the gyroscope can only measure the change in the Yaw angle, i.e., the Yaw angular velocities, not absolute angle since it has no reference for the orientation on earth's surface. This is the task of the Magnetometer.

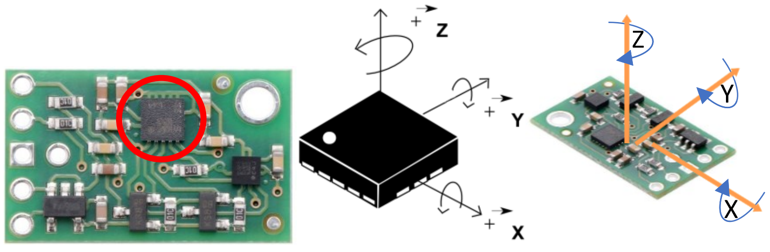


Fig. 10 Direction of the 3 detectable accelerations and the 3 detectable angular rates

Now, you may start conducting Experiment 1.B.

Magnetometer

As the name suggests, a magnetometer is a tool to measure the magnetic field intensity. In IMU units implemented in robotics applications, the magnetometer is usually utilized to measure the magnetic field of the earth, similar to a compass, to give a constant reference for the orientation of the robot.

The LIS3MDL IC from ST® provides a 3-axis magnetometer as shown in Fig.11. Each magnetometer measures the magnetic field in its direction. Hence, for every reading out of the sensor, you shall expect reading 3 values to represent the 3-axis magnetometer readings. While the sensor is parallel to the ground, the Z component should be very small, and only the X and Y components provide the Yaw angle. However, for any change in the Roll or Pitch angles, the Z component will play a role.

The LIS3MDL IC is the smaller chip at the far end of the module.

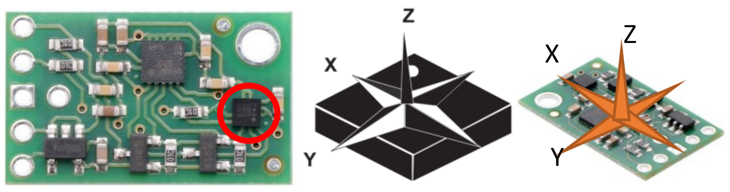


Fig. 11 Direction of the 3 detectable magnetic fields.

Unfortunately, the magnetometer readings are too noisy since there is a lot of interference caused by the induced magnetic fields all around us from different sources. Hence, the magnetometer noisy readings on the absolute reference work with the more sensitive gyroscope readings on the angular velocity to give reliable Yaw angle's measurements.

Features

The MiniIMU-9 v5 module from Pololu hosts a 3-axes Accelerometer and 3-axis Gyroscope in the LSM6DS33 MEMS and a 3-axis magnetometer in the LIS3MDL IC. The carrier PCB provides some electrical conditioning circuitry that combines both ICs on the same I^{2}C line and a voltage regulator and interface circuits to expand the working voltage range of the module. Here are the main features:

- **Sensitivity range:**
    - Gyro: ±125, ±245, ±500, ±1000, or ±2000°/s
    - Accelerometer: ±2, ±4, ±8, or ±16 g
    - Magnetometer: ±4, ±8, ±12, or ±16 gauss.
- **Output width**:
    - Gyro: one 16-bit reading per axis
    - Accelerometer: one 16-bit reading per axis
    - Magnetometer: one 16-bit reading per axis
- **Interface:** I^{2}C serial interface.
- **Operating Voltage:** 2.5V to 5.5V
- **Dimensions:** 20 mm × 13 mm × 3 mm

Increasing the range of any of the IMU tools will decrease its sensitivity since the number of output bits is constant. The range should be selected wisely to get best results.

Experiment 1: Using the LSM6 Accelerometer and Gyroscope

In this experiment, you should test the basic functionality and performance of the 3-axes accelerometer and Gyroscope in the LSM6DS33 MEMS IC of the MiniIMU-9 v5 module.

Required Equipment:

1- MiniIMU-9 v5 module
2- Arduino Due.
3- Breadboard.
4- Jumper wires.

Procedure:

1- **Install the LSM6 library from Pololu**:
   a. In the Arduino IDE, open the "Sketch" menu, select "Include Library", then "Manage Libraries...".
   b. Search for "LSM6", Click on the “LSM6 by Pololu” entry in the list, Click "Install".

2- **Make the connections**: between the Arduino Due and the module.
   - VIN from the VL53L1X Sensor to 3.3V pin from the Arduino Due Board.
   - GND from the VL53L1X Sensor to GND pin from the Arduino Due Board.
   - SDA, and SCL from the VL53L1X Sensor to pin (20 SDA) and (21 SCL) of the Arduino Board.
   - Leave VDD, sensor pins floating.

3- **Exp.1.A: Test the Accelerometer using the following code**:

```
#include <Wire.h>
#include <LSM6.h>
LSM6 imu;
char report[80];
void setup()
{
  Serial.begin(9600);
  Wire.begin();
  if (!imu.init()){
    Serial.println("Failed to detect and initialize IMU!");
    while (1);
  }
  imu.enableDefault();
}
void loop()
{
  float Ax,Ay,Az;
  imu.read();
  Ax = ((int16_t)imu.a.x)*0.061/1000;
  Ay = ((int16_t)imu.a.y)*0.061/1000;
  Az = ((int16_t)imu.a.z)*0.061/1000;
   Serial.println("Scaled Values of Acceleration (+-2g))");
  snprintf(report, sizeof(report), "X-axis:%.3f Y-axis:%.3f Z-axis:%.3f", Ax, Ay, Az);
  Serial.println(report);
  delay(500);
}
```

This code initializes the LSM6 unit using all the default settings. Please refer to the datasheet for these settings, and how to change them. The default settings include:
- Accelerometer full-scale of ±2 g, which has a sensitivity of 0.061 mg/LSB (Table 3, Page 15). This setting is accessible through the register (CTRL1_XL, Page 46), or in Arduino: imu.CTRL1_XL. Hence, the accelerometer readings pulled from the sensor (imu.a) must be multiplied by the sensitivity (0.061/1000) to get the absolute value in g.

If the sensor is not working properly, you can use the AD2 to “Spy” on the I2C data bus as referenced in Task 2, Experiment 1.

Measure and record: You should change orientation of the sensor and notice the following.

a. Put the sensor completely parallel to the ground, check the accelerometer’s X, Y and Z readings.
b. Put the sensor with the short edge parallel to the ground, and long edge facing up, Pitch angle = 90^{O}, check the accelerometer’s X, Y and Z readings.
c. Put the sensor with the long edge parallel to the ground, and short edge facing up, Roll angle = 90^{O}, check the accelerometer’s X, Y and Z readings.

d. From a position parallel to the ground, rotate the device 45 degrees around its x-axis, change the Roll angle, record accelerometer readings to calculate this tilt angle.
e. From a position parallel to the ground, rotate the device 90 degrees around its z-axis, i.e., change the Yaw angle, record accelerometer readings to calculate this Yaw angle.

Note that for all the above readings, while the module is at rest, the $\sqrt{X_c^2 + Y_c^2 + Z_c^2}$ should be constant.
Record all the reading output in your report, with screenshots.

You can also compute the absolute Roll and Pitch angles by adding the following lines of code within the loop function, before the delay():

```
float roll,pitch;
roll  = atan(Ay/Az) * 180 / PI;
pitch = atan(-Ax/sqrt(Ay*Ay + Az*Az)) * 180 / PI;
Serial.println("Pitch and Roll Rotation Angles");
snprintf(report,sizeof(report),"Roll(x): %.3f - Pitch: %.3f",roll,pitch);
Serial.println(report);
```

Tilt the Sensor module to get a 45° Pitch once, and also to get a 45° Roll.

4- **Exp.1.B: Test the Gyroscope using the following code**:

This following code initializes the LSM6 unit using all the default settings. Please refer to the datasheet for these settings, and how to change them. The default settings include:
- Gyroscope full-scale of 245 dps (degree per second), which has a sensitivity of 8.75 mdps/LSB (Table 3, Page 15). This setting is accessible through the register (CTRL2_G, Page 48), or in Arduino: imu.CTRL2_G.
Hence, the accelerometer readings pulled from the sensor (imu.g) must be multiplied by the sensitivity (8.75/1000) to get the absolute value in dps.

If the sensor is not working properly, you can use the AD2 to “Spy” on the I2C data bus as referenced in Task 2, Experiment 1.

Measure and record: You should change orientation of the sensor and notice the following.

a. While the sensor is lying flat, parallel to the group, rotate the module in the Z direction, i.e., the Yaw angle with respect to Fig. 10, in a constant speed <u>back and forth</u>. Note and gyroscope record output on the Z direction <u>as you move the module</u> will change significantly between +ve values and -ve values as you move the rotate the module back and forth. The reading output will be low as soon as you stop rotating the module.
b. Repeat step a, while starting from the sensor parallel to the group, rotate the module in the X direction, i.e., the Roll angle with respect to Fig. 10, in a constant speed.
c. Repeat step a, while starting from the sensor parallel to the group, rotate the module in the Y direction, i.e., the Pitch angle with respect to Fig. 10, in a constant speed.

5- Record some reading outputs in your report, with screenshots.

```
#include <Wire.h>
#include <LSM6.h>
LSM6 imu;
char report[80];
void setup()
{
  Serial.begin(9600);
  Wire.begin();
  if (!imu.init())
  {
    Serial.println("Failed to detect and initialize IMU!");
    while (1);
  }
  imu.enableDefault();
}
void loop()
{
  float Gx,Gy,Gz;
  imu.read();
  Gx = ((int16_t)imu.g.x) * 8.75/1000;
  Gy = ((int16_t)imu.g.y) * 8.75/1000;
  Gz = ((int16_t)imu.g.z) * 8.75/1000;
  Serial.println("Scaled Values of Gyroscope (+- 245dps)");
  snprintf(report, sizeof(report), "X-axis:%.3f Y-axis:%.3f Z-axis:%.3f",Gx, Gy,
Gz);
  Serial.println(report);
  delay(500);
}
```

Experiment 2: Using the LIS3MDL Magnetometer

In this experiment, you should test the basic functionality and performance of the 3-axes magnetometer in the LIS3MDL IC in the MiniIMU-9 v5 module. The equipment and connections are the same as Experiment 1.

Procedure:

1- **Install the LIS3MDL library from Pololu**:
   a. In the Arduino IDE, open the "Sketch" menu, select "Include Library", then "Manage Libraries...".
   b. Search for " LIS3MDL", Click on the “LIS3MDL by Pololu” entry in the list, Click "Install".

2- **Test the magnetometer using the following code**:
   This code initializes the LIS3MDL unit using all the default settings. Please refer to the datasheet for these settings, and how to change them. The default settings include:
   - Magnetometer range of ±4 gauss, which has a sensitivity of 6842 LSB/gauss (Table 3, Page 8).

This setting is accessible through the register (CTRL_REG2, Page 25), or in Arduino: mag.CTRL_REG2.
Hence, the Magnetometer readings pulled from the sensor (mag.m) must be divided by the sensitivity (6842) to get the absolute value in gauss.

3- **Exp. 2A: Check operation (absolute):** With the code running,

   a. Put the sensor completely flat, parallel to the ground, note that the Z reading should be small compared to the X and Y readings. Rotate the sensor module around the Z axis, in the Yaw direction, keeping the sensor parallel to the group, until you identify the direction with maximum reading for the X axis, and the direction with maximum reading for the Y axis.
   b. Repeat step a, while placing the sensor perpendicular to the group on the short edge. Here, the X reading should be small.
   c. Repeat step a, while placing the sensor perpendicular to the group on the long edge. Here, the Y reading should be small.

Note that while the values change (increase and decrease) reasonably with tilting the magnetometer, you will notice that the different components (x,y,z) have different “bias” and “span” values, which makes it difficult to calculate the magnetic north direction.

```
#include <Wire.h>
#include <LIS3MDL.h>
LIS3MDL mag;
char report[80];
void setup()
{
  Serial.begin(9600);
  Wire.begin();
  if (!mag.init())
  {
    Serial.println("Failed to detect and initialize magnetometer!");
    while (1);
  }
  mag.enableDefault();
}
void loop()
{
  float Mx,My,Mz;
  mag.read();
  Mx = ((float)mag.m.x) / 6842;
  My = ((float)mag.m.y) / 6842;
  Mz = ((float)mag.m.z) / 6842;
  Serial.println("Scaled Values of Magnetometer (+- 4 gauss)");
  snprintf(report, sizeof(report), "X-axis:%.3f Y-axis:%.3f Z-axis:%.3f",Mx, My,
Mz);
  Serial.println(report);
  delay(500);
}
```

4- **Exp. 2B: Calibrating the magnetometer:** The magnetometer has 3 spatial axes measurements. Due to various environmental and fabrication issues, the readings of the 3 axes have different bias values and different scales. If the sensor is tilted so that the magnetic field is exactly in the x direction, the x component of the measured field m_x should be at maximum & positive value m_{xmax} and when the sensor if flipped so the magnetic field is exactly in the opposite direction of x the m_x should be in the minimum & negative value m_{xmin}. Ideally, $|m_{xmax}|$ should be equal to $|m_{xmin}|$ and all directions should have the same property. Also, ideally, all directions should have the same span i.e. $m_{xmax} - m_{xmin}$. Practically, this does not happen and readings from different directions have different spans and each direction have a bias i.e. $|m_{xmax}| \neq |m_{xmin}|$ Thus, it is unreliable to depend on the absolute measurements of the magnetic field to calculate its directions.
Fortunately, we are not interested in the absolute values. We are interested in calculating only the direction of the magnetic field. To measure the filed directions, all of its measured components must be comparable to each other i.e. have the same span and bias. To do so, the absolute measurements of each direction should be scaled and shifted according to its maximum and minimum values to give a reading between, let's say, +1 and -1 or other suitable symmetrical range. If the scaled reading of the x component is m_{xs} then it would be calculated as:

$$m_{xs} = 2\frac{m_x - m_{xmin}}{m_{xmax} - m_{xmin}} - 1$$

The following code goes through a 10 sec loop during the setup function to find the maximum and minimum values of the magnetometer readings in all the X, Y and Z directions. Then, during the loop function, it computes the current reading while using the above equation to perform the calibration. We also update the maximum and minimum values whenever needed. During the Calibration phase, you will need to tilt and move your magnetometer in all directions (Roll, Pitch and Yaw) to capture these values before actual reliable usage (Btw, now you should know how to properly calibrate the compass in your smartphone!).

5- **Check the operation (relative):** With the sketch uploaded on the Arduino, you should see the magnetometer relative readings on the serial monitor. Tilting and rotating the device will change the distribution of the measured magnetic field on the magnetometer's axes. Try to tilt and rotate the magnetometer till you get a maximum positive reading at one direction, and (almost) zero readings at other directions. This should be the direction of the Earth's magnetic north (you should find an interesting direction in here in Canada!).

```
#include <Wire.h>
#include <LIS3MDL.h>
LIS3MDL mag;
LIS3MDL::vector<float> mag_min, mag_max, current;
char report[100];
void setup() {
  Serial.begin(9600);
  Wire.begin();
  if (!mag.init()) {
    Serial.println("Failed to detect and initialize magnetometer!");
    while (1);
  }
  mag.enableDefault();
  Serial.println("Calibrating for 10 Seconds. Roll, Pitch and Yaw to get high numbers!");
  unsigned long myTime = millis();
  while (millis() < myTime + 10000) {
    mag.read();
    mag_min.x = min(mag_min.x, mag.m.x); mag_min.y = min(mag_min.y, mag.m.y);
    mag_min.z = min(mag_min.z, mag.m.z); mag_max.x = max(mag_max.x, mag.m.x);
    mag_max.y = max(mag_max.y, mag.m.y); mag_max.z = max(mag_max.z, mag.m.z);
    snprintf(report, sizeof(report), "min: {%.0f, %.0f, %.0f} max: {%.0f, %.0f, %.0f}",
             mag_min.x, mag_min.y, mag_min.z, mag_max.x, mag_max.y, mag_max.z);
    Serial.println(report);
    delay(100);
  }
}
void loop() {
  mag.read();
  mag_min.x = min(mag_min.x, mag.m.x); mag_min.y = min(mag_min.y, mag.m.y);
  mag_min.z = min(mag_min.z, mag.m.z); mag_max.x = max(mag_max.x, mag.m.x);
  mag_max.y = max(mag_max.y, mag.m.y); mag_max.z = max(mag_max.z, mag.m.z);

  current.x = 2 * ((mag.m.x - mag_min.x) / (mag_max.x - mag_min.x)) - 1;
  current.y = 2 * ((mag.m.y - mag_min.y) / (mag_max.y - mag_min.y)) - 1;
  current.z = 2 * ((mag.m.z - mag_min.z) / (mag_max.z - mag_min.z)) - 1;

  snprintf(report, sizeof(report), "Mag: X-axis: %.3f, Y-axis: %.3f, Z-axis: %.3f .",
           current.x, current.y, current.z);
  Serial.println(report);
  delay(500);
}
```

Bonus Experiment: Display movement on Python Visualization

Pololu provides a Python visualization for the IMU. You should follow the installation instructions carefully and precisely while installing the software packages required to operate this animation. This includes the python version itself and all other packages.

Procedure:

1- **Calibrate the magnetometer:** using the code in the previous experiment.

2- **Modify and Upload the Sketch:** Pololu provides a github repository that has an Arduino sketch called MinIMU9AHRS that calculates the angles and send the data, in addition to the python script called MinIMU-9-test.py that is used for visualization. Open the Arduino sketch, and change the magnetometer calibration values to the values acquired in the first step. Follow the instructions to install the required software packages to run the visualization, upload and run the Arduino sketch, then run the python script.

3- **Check the operation:** By now, you should a 3D visualization of the angular orientation of the module along with an arrow directing to Earth's magnetic north. Tilt and rotate the module and notice how the 3D model changes its orientation in real-time according to the module's orientation.
Note: You might need to redo the magnetometer calibration step again if the 3D model response is not good. Try to change the range of the magnetometer to a larger range and redo the calibration.

Discussion

4- What ranges did you use in your experiments for the accelerometer, gyroscope, and magnetometer?

5- How many "g"s, should the accelerometer read in a rocket accelerating, according to an external observer standing on earth, with g perpendicular to the ground near the surface of earth?

6- Can the accelerometer measure the 3D orientation (all 3 angles relative to some reference) of a standing object relative to earth? Why?

7- Research: can the accelerometer be used to measure linear displacement (i.e. how many cm an object moved in x-direction)?

End of Lab 3.