Lab 7: Interfacing with IMU through I2C and Car Assembly

A) Objectives

- To obtain the Acceleration data and Euler angles through I2C.
- To format and scale the data read in SI units.
- To assemble the basic two-wheeled car platform

B) Background

In this lab, we are going to read the raw sensor data (acceleration and angular velocity) from the MPU-6050 IMU through the sensor native I2C interface. Compared to reading data through the sensor module UART interface, the I2C interface offers lower communication latency and access to low level controls and data.

The main objective for this lab is to read the accelerometer and gyroscope measurements from the MPU6050 and display it. To accomplish this, we must configure the chip to power on (Task 2 – 5), read the raw binary sensor readings from the chip (Task 6 – 7), and format the binary data into meaningful numbers to humans (Task 8).

You will need to refer to the sensor datasheet:

https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

and the sensor register map:

https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf

C) I2C Interface

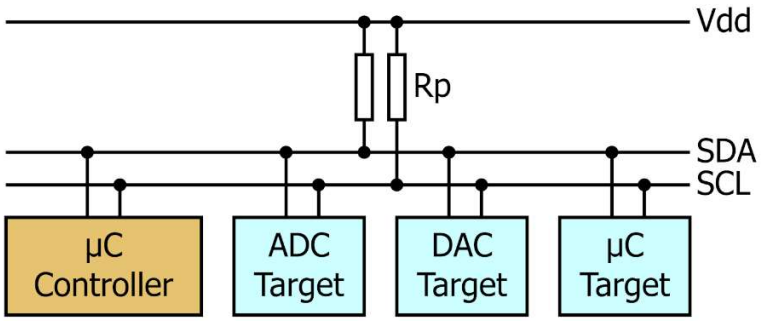
Task 1 – Write Data to Specific Registers

I2C stands for Inter-Integrated Circuit, like its name suggests, it is a way for Integrated Circuits (ICs) to communicate with one another.

Physically, the I2C bus consists of a serial clock line (SCL) and a serial data line (SDA). These two lines can chain together multiple I2C slaves to a I2C master.

Electrically, the I2C bus operates using open drain inputs/outputs (IO), meaning pins wishing to participate in the bus will pull the voltage on the bus lines to logic low, while pull up resistors set the lines to default to logic high.

Source: Wikipedia



This lab requires you to utilize the Arduino Wire library to control the MCU I2C peripheral. You can find the documentation of the library and useful example code here (https://www.arduino.cc/en/Reference/Wire).

Registers are bytes of memory within the IMU. Some registers control the behavior of the device, while some registers stores sensor measurement data. Task 1 specifically asks you to complete a function so that the MCU can successfully write data to specific registers.

Task 2-5 – Power on the Interface

Page 35 of the datasheet specifies the protocol for writing data to s specific register in the IMU, let's try and understand the protocol.

Single-Byte Write Sequence

Master	S	AD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

In an I2C system, the device that initiates communication is the master, which will be the MCU. The device that responds when addressed by the master is the slave, which will be the IMU in this case.

To initiate communication in the I2C bus, the master must first send a start signal. More on how to send this signal below.

Since there can be multiple slaves connected to the master with the same two wires, sharing an I2C bus, and so each I2C slave must have a unique address which the master specifies in the beginning of each communication. I2C devices addresses are usually 7-bit long (denoted as AD in the table above), you should find the I2C address of the IMU and add it to the code in Line 3.

When the master initiates communication, it must also indicate if it wishes to send data to the slave or to receive data from the slave. To read from a slave device, the master appends 1 to the end of the 7-bit address and send the whole 8-bit byte as the first byte of an I2C packet. Conversely, a 0 is appended to the end of the address when the master wishes to write to the slave.

The Wire.beginTransmission(address) function in the Arduino Wire library handles the sending of the start signal and appending of the read bit. Using this function, only the 7-bit address needs to be provided and the S, AD+W part of the transmission will be handled.

The master must wait for the slave device to send an acknowledgement signal (denoted as ACK in the tables, meaning slave acknowledge) before sending anymore data. Lucky for us, this flow control is handled in hardware by the MCU I2C peripheral, so we do not have to pay attention to it in code.

The master proceeds to indicate the 8-bit register address (denoted as RA in the tables) it wishes to read or write into. There are many registers in the IMU, each having different functions or holding different data, so the master must indicate which specific register it wishes to access. The register map document lists all the registers in the IMU.

After another acknowledgement from the slave, the master proceeds to send the byte of content (denoted as DATA in the tables) that needs to be written to the specified register.

You can use the Wire.write(data, length) function to send the two bytes of data (register address and data) to the IMU.

After sending the data, a stop signal (denoted as P in the tables) has to be sent to indicate the end of the communication. The Wire.endTransmission() function will handle this.

Task 6-7 – Read, format and store raw binary sensor readings

In task 6, you are required to read multiple bytes from the IMU. The process of reading multiple bytes is very similar to writing a byte, the process is detailed in the table below:

Burst Read Sequence

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

Challenge: The tricky challenge here is the need to send the repeated start signal (denoted as the second “S” signal in table). Please read the Wire library documentation to figure out which function will produce this signal.

When reading page 29 of the register map, you will notice that the measurement from each axis of the two sensors are 16 bit long, but are stored into 2 separate 8-bit registers. In task 8, you are required to combine the 2 8-bit long bytes into a single 16-bit word. You will have to use the bit shift operator in C to achieve this. Find out more here.

https://www.arduino.cc/reference/en/language/structure/bitwise-operators/bitshiftleft/

Task 8 – Scale the data in units of *g* and *degrees per second*

The 16-bit word you obtained is the IMU's internal representation of the measurement, and it would be nice to translate this machine representation into numbers based on commonly used physical units.

Referring to page 29 and 31 of the register map, we can see how we can achieve this. We have configured the accelerometer to have a +-4g range, and the datasheet tells us the sensitivity of the accelerometer in this range is 8192LSB/g. Note g in this context represents Earth's gravitation acceleration constant, which is $9.81ms^{-2}$. Also note that m stands for milli, which means one thousandth.

You are required to translate the gyroscope data as well, such that it is in unit of dps (degree per second).

D) Car Assembly

Assemble the 2-wheeled robot laying out the components as shown in pictures below.