EE-469 EMBEDDED SYSTEM

Faik Doruk AKGÜNEY 201726003

Buğra Can YAŞAR 201726076

MPU6050 Sensor Promotion

MPU6050 gyroscope and accelerometer sensor consist of a digital motion sensor that performs all complex processing, computations and provides sensor data output to other MCUs over I2C communication. MPU6050 sensor module has 6-axis motion tracking sensors integrated inside the single-chip such as 3-axis measurement for gyroscope and 3-axis measurement for acceleration. In this way, we can use this sensor to measure acceleration, velocity, displacement and many other electro-mechanical features. This sensor provides data output over I2C bus protocol. That means in order to read sensor values with the help of TM4C123G microcontroller. So, we are used the I2C module of TM4C123G microcontroller.

MPU6050 Sensor Properties

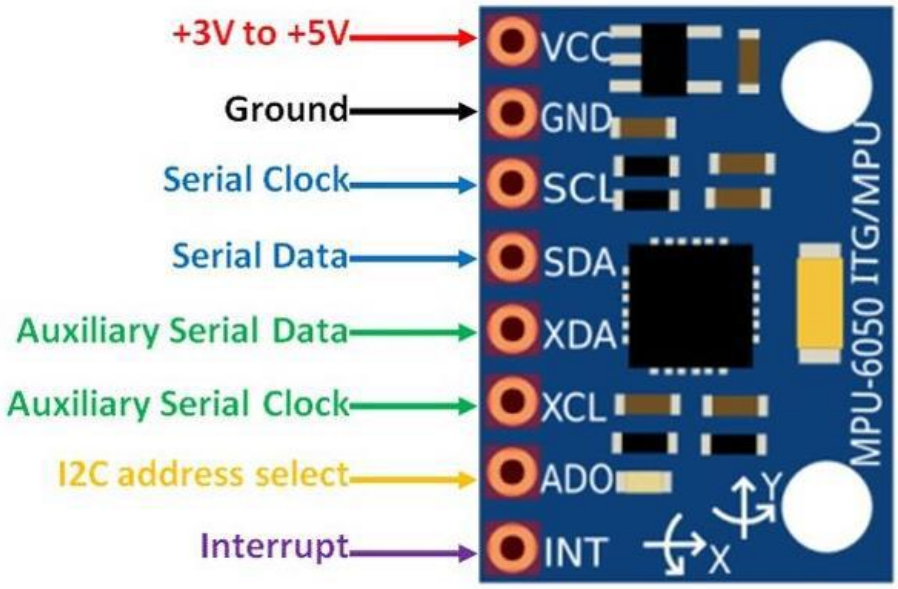
Firstly, we are discussing some important properties of the MPU6050 sensor module. It has a built-in I2C sensor bus which is used to provide gyroscope, accelerometer and temperature sensor data to other devices such as microcontrollers. This sensor module has on board pull-up resistors. So, we do not need to connect external pull resistors which are a requirement for the I2C bus interface.

Some other properties are:

• User Programmable gyroscope and accelerometer with the help of 16-bit Analog to digital converter.

• 1024 Byte FIFO buffer to provide data to the connected microcontroller in high speed and enters the low power mode afterwards.

• Built-in temperature sensor.

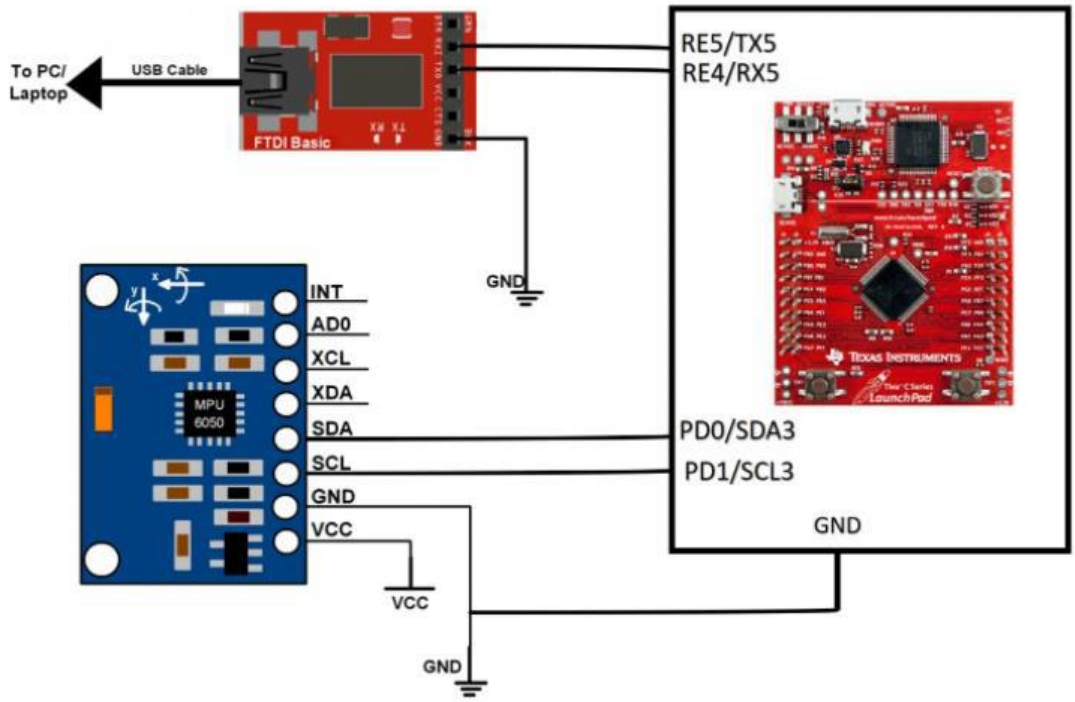


.

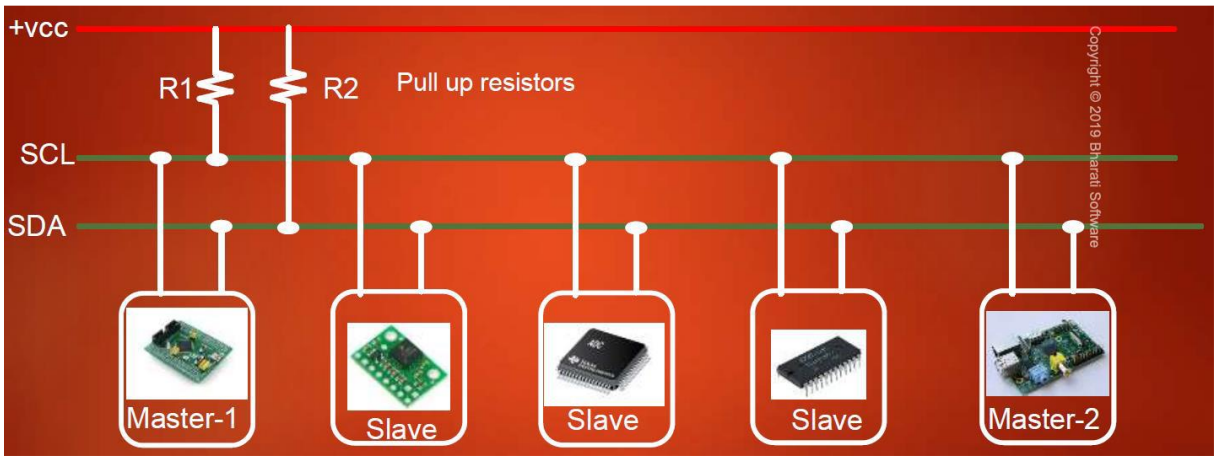
The figure below shows the pinout diagram of the gyroscope and accelerometer sensor. It consists of 8 pins. But to interface it with any microcontrollers, we should only need its four pins such as VCC, GND, SDA and SCL pins. SDA and SCL pins are used to interface it with microcontroller or microprocessor through I2C communication protocol. In a similar way, we are used I2C modules of TM4C123G Tiva C Launchpad to read data from MPU6050 data registers. GND pin connect with the ground connection of microcontroller and V_{cc} pin with 3.3 volt power supply. But pins of MPU6050 can withstand 5 volts.

MPU6050 Circuit Diagram with UART

MPU6050 sensor module provides data through I2C communication. Then, we are used the I2C module of TM4C123G microcontroller and we are connected the I2C module of TM4C123G microcontroller with SCL and SDA pins of MPU6050 sensor as shown in figure. In this project, code of TM4C123G Tiva C Microcontroller, we are printed to measure sensor values such as gyro and acceleration on the hyper terminal of the computer using USB to serial converter. We are used the UART module of Tiva Launchpad to transfer data to the computer. We find MPU6050 sensor values such as gyro and acceleration (X, Y, Z axis values) thanks to this project. This MPU6050 code for TM4C123G Tiva microcontrollers reads the gyroscope and accelerometer axis data through I2C communication and to send data the computer through UART communication.



I2C COMMUNICATION PROTOCOL



I2C is a bus protocol. This protocol designed to allow the user to transfer short range and serial data. I2C is a 2-wire communication protocol basically. The wire using for data is called SDA and another wire using for clock is called SCL. The SDA and SCL lines must be connected to the VCC line with pull-up resistors in order to ensure error-free communication throughout entire line. The reason for connecting upwards with pull-up resistors is related to the microwave theory. This communication protocol can have more than one master and slave devices. In this project, Tiva C is used in master mode while MPU6050 acceleration sensor is used in the slave part. The master device is printed an address and this address is heard by each data. This device sends an ACK bit to whichever slave it matches. If this transmitted address does not match any receiver, the master detects that there is a NACK in the environment. The I2C bus is either in communication mode or idle mode at any time.

ADVANTAGES AND DISADVANTAGES OF I2C

ADVANTAGES

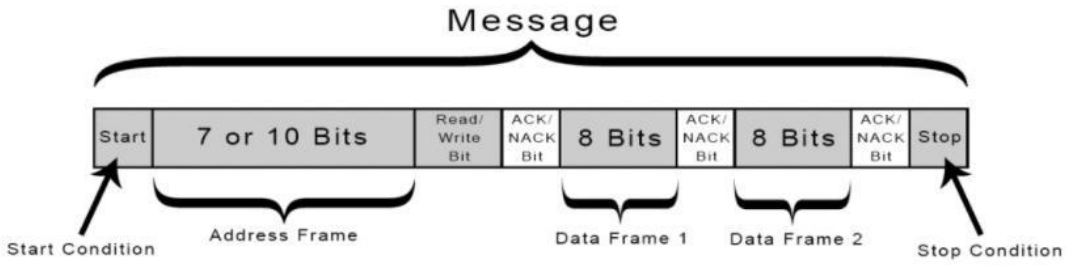
- Only use two wires.
- Support multiple masters and multiple slaves.
- ACK/NACK bit gives confirmation that each frame is transferred successfully.
- Well known and widely used protocol.

DISADVANTAGES

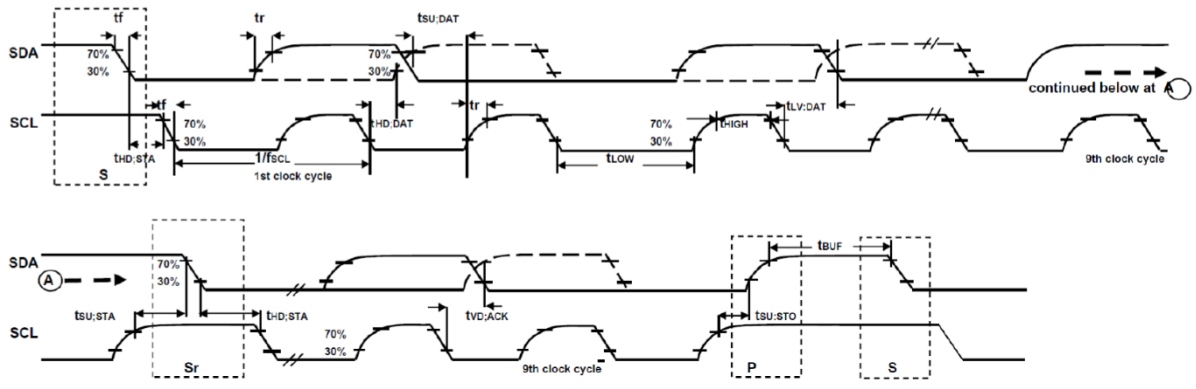
- Slower data transfer rate than other communication systems.

- The size of the data frame is limited to 8 bits.

I2C WORKING PRINCIPLE

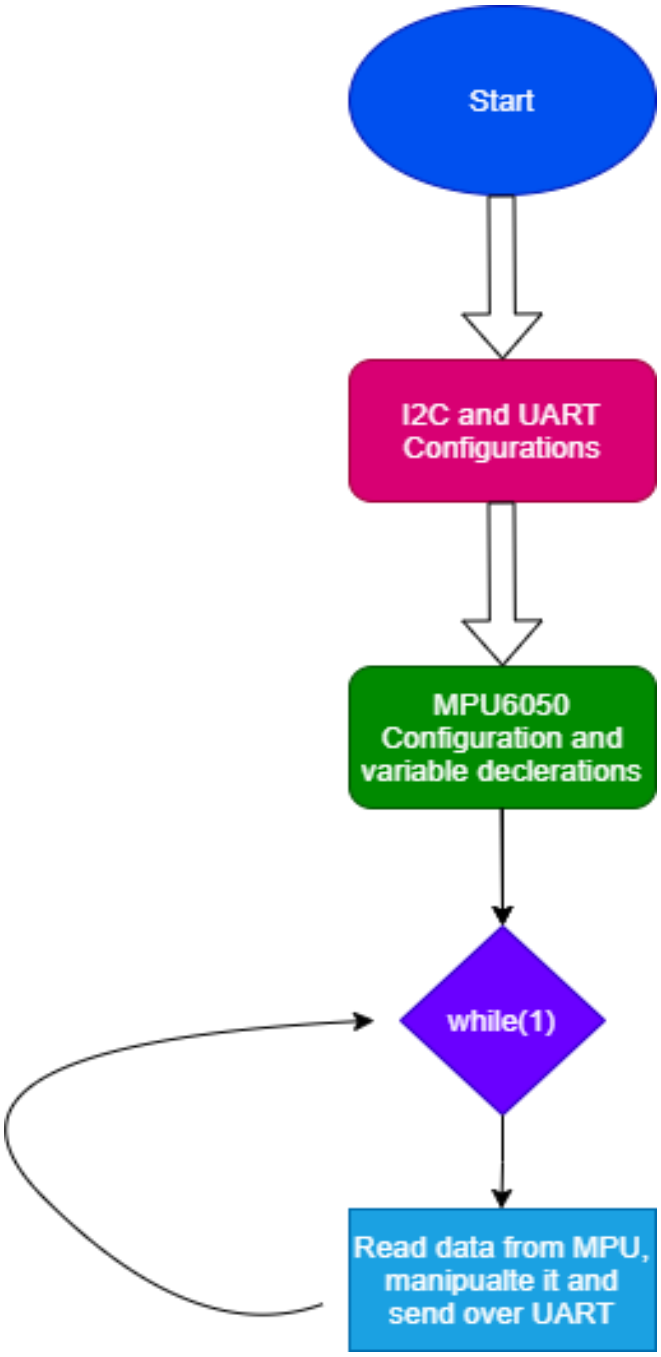


Start Condition: When SCL is logic 1, it occurs by pulling SDA from logic 1 to 0. This process is done by the master.
Stop Condition: When SCL is logic 1, it occurs by pulling SDA from logic 0 to 1. This process is done by the master.
Address Frame: At this stage, the address and communication mode of the slave device is contacted to display by the master device.
Read/Write Bit: This bit specifies the direction of data transfer. If the master device need to send data a slave device, this bit is set zero (Write). If the master needs to receive data from slave device, it is set one (Read).
ACK/NACK Bit: It stands for Acknowledged/Not-Acknowledged bit. If an address frame or data frame are received successfully, an ACK bit is returned the sender from receiving device.



I²C Bus Timing Diagram

FLOW CHART


Start
I2C and UART Configurations
MPU6050 Configuration and variable declerations
while(1)
Read data from MPU, manipualte it and send over UART

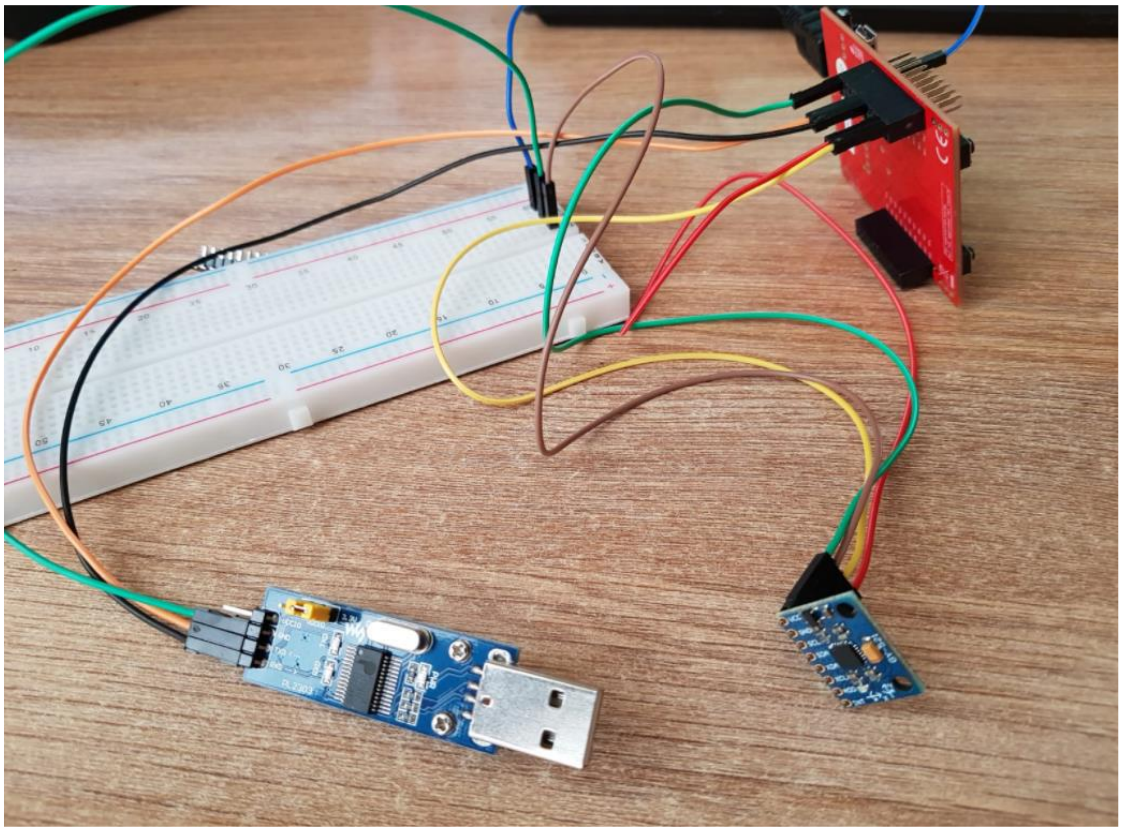
Problem Definition and Solution

In the first part of the Project, we have defined all the register addresses in our code with using macro definitions. With this method, we are avoiding with hardcoding all register values. On the other hand, Keil UVision is generating a device specific header file for us to reach its own registers properly. In the main section of a project, we are defining our variable to store the raw data which come from the MPU6050. After creating those variables, we need to initialize some of our peripherals to communicate with MPU6050. I2C_Init() function is used to achieve this purpose. MPU6050 is one of the sensors that use I2C communication protocol to send and receive data from external devices. In this function that we do first is enabling the clock of I2C and GPIO modules in Tiva C. By default, all clocks are off to save power for some battery-based projects, so we need to enable those firstly. After that we need to configure those pins for I2C communication. All pins in MCU are generally configured in input mode. For I2C communication we need to configure those 2 pins which are for SDA and SCL lines in Alternate Function mode. To do that we have to use datasheet of the MCU and understand the pin multiplexing mechanism. Another configuration style that we must do is GPIO output method. It can be push pull or open drain. For I2C communication it absolutely must be OPEN DRAIN. This is specific for protocol. All protocols have different designs, and we must follow these rules. At the end, PA6 and PA7 GPIO pins were configured in open drain mode with internal pull up resistors and digital enabling mechanism to use I2C communication. I2C protocol can work in different speeds. It is not fast one, but widely used. 100 kHz is configured to talk to the MPU6050 sensor for this project. When I2C is ready for Tiva C, we can configure our sensor to use this peripheral and start reading data from that. MPU6050 has 8 MHz internal clock. With enabling that clock and configuring the sensor according to its datasheet it will start sampling data for us. All we need to do is to read that data and send it to the PC over UART.

For UART Communication we have to follow the same path with I2C configuration method. UART5 is used in this project, so we started enabling its own clock and GPIO clock. PE4 and PE5 are the GPIO pins which are used for Rx and Tx. UART is a serial protocol, and it is sent the data according to baud rate value. Both devices which will use this protocol have to follow same baud rate. Data length can be different in UART. Parity bit can also change this data frame. We used 8 bits data frame without parity bit in this project. After this long configuration we can get the data from MPU6050 and send it to the PC. I2C starts with a Start Condition which is described before. After this start bit Tiva C is sending 7 bits Address of the MPU6050 and Logical LOW for 8. Bit to indicate we will write data into your registers. Tiva C MCU is master and MPU6050 sensor is slave device in this project. So, the slave devices pull the SDA line LOW to send ACK bit to the master. When master gets this ACK bit, it starts sending 8 bits data frame and waits for ACK bit again. If slave devices pull the line down that means ACK and data is sent properly. If slave makes the SDA line HIGH, that means NACK, so master must resend the data again. Proper registers are used to receive 14 bytes data from slave MPU6050 device. If master wants to read data from any slave device, it should send its address and it must add Logical High in the 8. Bit of address frame. When data comes from MPU6050, we need to manipulate its raw data according to the data sheet, so each two 8 bits data frames are compounded as 16 bits data values. With UART protocol, this info will be sent to the PC for a user.

RESULT

There is I2C communication protocol inside the card. We are configured this module properly. In this way, TM4C123G and MPU6050 is communicated between them. So, we can both send and receive data thanks to this configuration. Thanks to UART, we are sent the data, we are obtained to the computer. After this data comes to the computer, the embedded software part of the this job is finished. The code is written and the debugged. At the end of the project, the acceleration data on the X,Y,Z axes and the gyro data on the X,Y,Z axes are accessed.



```
Gx = 0.16    Gy = 0.50    Gz = 0.29    Ax = 0.43    Ay = 0.21    Az = 0.59
Gx = 0.80    Gy = 0.07    Gz = 0.25    Ax = 0.28    Ay = 0.39    Az = 0.28
Gx = 0.92    Gy = 0.69    Gz = 0.93    Ax = 0.17    Ay = 0.21    Az = 0.42
Gx = 0.56    Gy = 0.08    Gz = 0.39    Ax = 0.20    Ay = 0.78    Az = 0.83
Gx = 0.45    Gy = 0.44    Gz = 0.03    Ax = 0.78    Ay = 0.90    Az = 0.14
Gx = 0.93    Gy = 0.52    Gz = 0.95    Ax = 0.28    Ay = 0.37    Az = 0.45
Gx = 0.86    Gy = 0.17    Gz = 0.58    Ax = 0.99    Ay = 0.44    Az = 0.72
Gx = 0.32    Gy = 0.77    Gz = 0.60    Ax = 0.34    Ay = 0.44    Az = 0.51
Gx = 0.78    Gy = 0.33    Gz = 0.79    Ax = 0.99    Ay = 0.29    Az = 0.37
Gx = 0.85    Gy = 0.69    Gz = 0.36    Ax = 0.14    Ay = 0.01    Az = 0.64
```