Understanding GY-521

June 1, 2020

As we seen in the hardware section gy-521 is more efficient than gy-87 and ADXL335 for our purpose of use.

Gy-521 board features:

1. Chip built-in 16bit AD converter, 16-bit data output
2. Driver Chip: MPU6050
3. Operating Voltage: 3-5V DC
4. Communication: I2C Protocol
5. Gyro Range: ± 250, 500, 1000, 2000 °/s
6. Accelerometer Range: ± 2, ± 4, ± 8, ± 16 g

There are lot more other features too but not of our interest for now.

In this documentation we are going to look at glance the registers that are required and affects to our experiments. Actually to study the MPU 6050 the datasheet is enough required documentation.

The MPU 6050 supports various range of sensor measurement and also adjustable sample rate of sensor. It is having 16 bit resolution of data over its configured full range. It is having gyroscope and accelerometer both on one chip. We will communicate to it with the use of Arduino over I2C protocol.

As aur aim to understand MPU 6050 basics, to communicate with it on I2C bus, we will use “Wire.h” default library of Arduino platform. In problem statement folder you were provided by a skeleton file where you need to complete code to solve the given problem statement based experiment.

In IIC protocol for communicate, read or write to a register we need to follow following sequence

1. Start communication
2. Send device address
3. Send register address
4. Release bus to let MPU put data if we are reading register
5. Send data to be write on the register if we are modifying register.
6. End communication

Task 1:

Our board gy-521 is having IIC address 0X68 when AD0 pin is set low. Let we study registers that we are going to write or read.

Register 0X6B:

The register is power management register or in data sheet it's called as PWR_MGMT_1. Setting this register to 0X00 will set all things at default.

Register 0X19:

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
19	25	SMPLRT_DIV[7:0]							

This register specifies the divider from the gyroscope output rate used to generate the Sample Rate for the MPU-60X0.

The sensor register output, FIFO output, and DMP sampling are all based on the Sample Rate. The Sample Rate is generated by dividing the gyroscope output rate by SMPLRT_DIV:

$$SampleRate = GyroscopeOutputRate/(1 + SMPLRT\_DIV)$$

where Gyroscope Output Rate = 8 kHz when the DLPF is disabled (DLPF_CFG = 0 or 7), and 1kHz when the DLPF is enabled (see Register 26).

Note: The accelerometer output rate is 1 kHz. This means that for a Sample Rate greater than 1kHz,the same accelerometer sample may be output to the FIFO, DMP, and sensor registers more than once.

For a diagram of the gyroscope and accelerometer signal paths, see Section 8 of the MPU-6000/MPU-6050 Product Specification document.

As our DLPF_CFG bits are 0 the gyroscope o/p rate will be 8 kHz, we will set this register to 8000 hz/500hz = 16. Therefor the value to be write in 0X19 is 16.

Register 0X1B:

The register will be modified to set our desired sensitivity of gyroscope.

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]		-	-	-

In this register bit – 0,1,2 are not used so we will make it zero. bit 5,6,7 are used for self-test purpose and its must set to be zero. FS_SEL selects the full scale range of the gyroscope outputs according to the following table.

FS_SEL	Full Scale Range
0	± 250 °/s
1	± 500 °/s
2	± 1000 °/s
3	± 2000 °/s

Table 1: FS_SEL bits (ref. datasheet)

So for setting sensitivity to 1000 dps FS_SEL = 2 and the register 0X1B will be written with 0X10.

Register 0X1C:

The register will be modified to set our desired sensitivity of accelerometer.

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		-		

In this register bit – 0, 1, 2 are not used so we will make it zero. Bit 5, 6, 7 are used for self-test purpose and its must set to be zero. AFS_SEL selects the full scale range of the accelerometer outputs according to the following table.

AFS_SEL	Full Scale Range
0	± 2*g*
1	± 4*g*
2	± 8*g*
3	± 16*g*

Table 2: AFS_SEL bits (ref. datasheet)

So for setting sensitivity to +/-2g, AFS_SEL = 0 and the register 0X1C will be written with 0X00.

Data Register: 0X3D to 0X40 and 0X43 to 0X44

This registers stores most recent sampled data of respective sensors as shown in table below.

Now for getting raw data of sensors we need to read this registers as per our requirement according to this table.

Task 2 3:

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							

Table 3: Register configurations (ref. data sheet)

Note: Data for one axis is of 2 bytes and will be written to 2 separate registers. So for getting our data we need to read 2 registers and then combine them to gather in one 16 bit signed int variable with the logic below.

1. Store both high byte and low byte in temporary variable temp1 and temp2 respectively
2. Now left shift variable temp1 (high byte) 8 times
3. Perform or operation on temp1 and temp2 (temp1 || temp2)
4. Store the resultant data in a variable.
5. Multiply it with a factor to get convert this raw data in to MKS unit.

How to derive multiplication factor:

Suppose we are having full scale range of sensor is +/- X unit. Here we are having 16 bit data to contain it. So when our 16 bit data will be 0XFFFF or in decimal 65536, it means for our full scale sensor output. Therefor when our raw data will have decimal value Y, our MKS unit value of that measurement will be

$$MKS\ value\ of\ raw\ data = Y * (65536/2)/X.$$

$$So, multiplication\ factor = 65536/(2 * X) MKS\ unit.$$

Note : how to write programme is explained in the file “mpu_reading_data_solu.ino” by comments in solution folder.