

PROJECT REPORT ON

**TILT SENSOR APPLICATION USING MPU 6050 INTERFACED
WITH BEAGLEBONE BLACK**

SUBMITTED IN PARTIAL
FULFILLMENT OF
THE REQUIREMENTS OF
THE AWARDS OF THE

PG-DIPLOMA IN EMBEDDED SYSTEM DESIGN

OFFERED BY

C-DAC HYDERABAD

BY

ISHWARI MODAK

220350330011

NEHA SINGH

220350330012



**ADVANCED COMPUTING TRAINING SCHOOL
C-DAC
HYDERABAD-500005**

MARCH 2022

CERTIFICATE

This is to certify that this is a Bonafide record of project entitled “**Tilt Sensor Application using MPU6050 interfaced with BeagleBone Black**”. ISHWARI MODAK-220350330011, NEHA SINGH-220350330012 have completed project work as part of **Diploma in Embedded System Design (March,2022-Batch)**, a PG course offered by C-DAC Hyderabad. They have completed project work under the supervision of Mr. Prasad Nissi. Their Performance was found to be good.

Signature of Project guide

(Mr. Prasad Nissi)

DATE: 15/09/22

PLACE: C-DAC, HARDWARE PARK, HYDERABAD.

ACKNOWLEDGMENT

Tilt Sensor application using MPU 6050 interfaced with Beaglebone Black project has been presented. This project marks the final hurdle that we tackle, of hopefully what would be one of the many challenges we have taken upon and are yet to take. However, we could not have made it without the support and guidance from the following. Firstly, I want to take this opportunity to have special thanks to our guide **Mr. Prasad Nissi** who helped us throughout this project by providing valuable guidance and advice as well as acquiring all components needed for this project to become a success.

(PG-DESD MARCH 2022)

ISHWARI MODAK

220350330011

NEHA SINGH

220350330012

TABLE OF CONTENTS

ABSTRACT

- 1. Introduction**
 - 1.1. Problem Statement**
 - 1.2. Software & Hardware Requirements**
 - 1.2.1. Hardware Specification**
 - 1.2.2. Software Specification**
 - 1.3. Coding and Testing**
- 2. Implementation**
 - 1. USB to TTL Converter**
 - 2. Beaglebone Black Rev c**
 - 3. MPU6050**
 - 4. I2C Protocol**
 - 5. Eclipse IDE**
 - 6. C Program**
 - 7. Operating System**
- 3. Schematic Diagram**
 - 3.1. Block Diagram**
 - 3.2. Implemented System**
- 4. Flow of project work**
- 5. Literature Survey**
 - 5.1. Paper 1**
 - 5.2. Paper 2**
- 6. Conclusion**
- 7. References**

ABSTRACT

Tilt is one of the important altitude parameters and its measurement is widely applied in a number of applications such as engineering machinery alignment, human body motion detection, game controllers, ground motion and land subsidence detection. Hence, various tilt sensors and systems have been developed accordingly. The main concept behind the tilt sensor application is based on the measurement of physical response of a mass moving relative to the fixed casing due to the induced inertial force caused by gravity. The types of moving mass can be solid, liquid or gas and the response of the mass pendulum with respect to gravity can be sensed in various ways including capacitive, resistive, fiber optic, magnetic and thermal. The MEMS technology has revolutionized the sensors to enable them with a smaller size and lower cost. Measuring the gravity acceleration using MEMS accelerometers has become one of the common methods to calculate the tilting angle.

The main aim of project is to measure Tilt Angle using MPU6050 & BeagleBone Black Microcontroller. The MPU6050 sensor uses I2C protocol to communicate with beaglebone black. The 3- axis Accelerometer sends the raw X, Y, and Z acceleration forces which are then converted into 'g' values by selecting the appropriate full-scale sensitivity. Similarly, the 3-axis gyroscope also sends the raw values of tilt over X, Y, Z axes which are converted into deg/sec for pitch, roll and yaw respectively. Based on these values, calibration of the application can be performed to maintain a balanced level.

1. PROJECT REQUIREMENTS

1.1 Problem Statement:

- To monitor the angle at which a mobile phone or tablet is held for the auto-rotate function.
- To detect the position of hand-held game systems and in game controllers.
- To indicate the roll of boats, vehicles and aircraft.
- To measure the angle at which a satellite antenna 'looks' toward a satellite.
- To estimate the height of a tree or building.
- To measure the steepness of a ski slope.
- To provide a warning system for the surface tilt angle of cryogenic liquids during transportation.
- To monitor laser levels and seismic activity.

1.2 Software & Hardware Requirements:

1.2.1 Hardware Specification:

- Development Board- Beaglebone Black Rev C
- Sensor- MPU 6050
- Serial communication- USB to TTL converter
- SD Card

1.2.2 Software Specification:

- Operating System: Debian
- IDE: Eclipse IDE
- Toolchain: arm-linux-gnueabi-
- Partition: GParted

1.3 Coding and Testing:

In this we develop a code using following software languages:

- C programming language
- Shell commands

2.

IMPLEMENTATION

The main aim of project is to measure Tilt Angle using MPU6050 & BeagleBone Black Microcontroller. The MPU6050 sensor uses I2C protocol to communicate with beaglebone black. The 3- axis Accelerometer sends the raw X, Y, and Z acceleration forces which are then converted into 'g' values by selecting the appropriate full-scale sensitivity. Similarly, the 3-axis gyroscope also sends the raw values of tilt over X, Y, Z axes which are converted into deg/sec for pitch, roll and yaw respectively. Based on these values, calibration of the application can be performed to maintain a balanced level.

1.USB to TTL Converter-

To access the beaglebone black through serial communication we are using the USB to TTL converter (CP 2102) as shown in Figure 1. Serial communication is a communication method used in telecommunications where data is transmitted one bit at a time in sequential order over a communication channel. It has different protocols such as CAN, ETHERNET, I2C, SPI, RS232, USB, UART, 1-Wire, and SATA. The beagle bone board comes with separate UART pins for serial communication. They are defined as J1 pins on board. So, to connect Beaglebone with the converter we require only 3 pins of the UART.

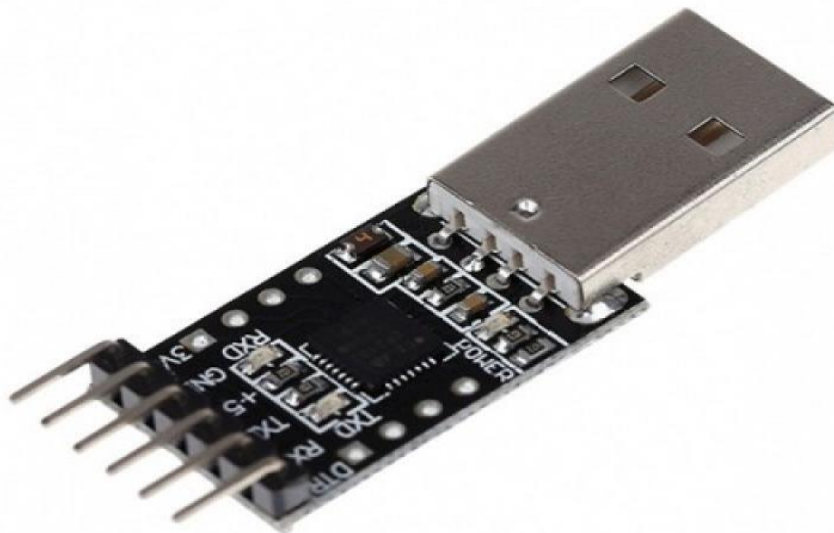


Figure 1.USB to TTL Converter (CP2102)

The ground pin of the converter (GND) is connected to pin no 1 of UART -J1 pin, the transmitter pin that is TXD of the converter is connected to the 4th pin of the J1- UART pin and the receiver pin present as RXD on the converter is connected to J1- UART 5th pin as shown in Figure 2. This ensures serial communication of the beaglebone and our Linux to which the USB of the converter will be connected.

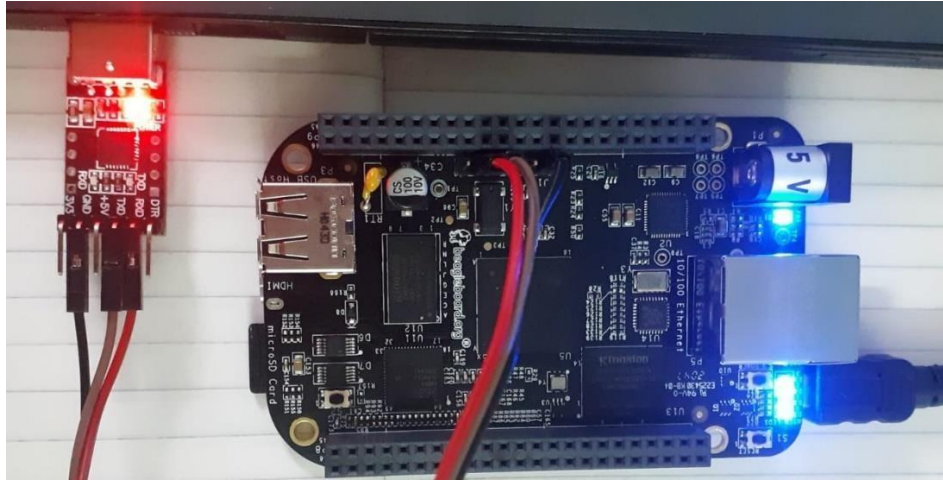


Figure 2 Connection of USB to TTL Converter to the Beaglebone and Linux

After connecting the TTL converter's ground, receiver and transmitter pins to the UART – J1 pins of the Beaglebone Board connect the USB to the Linux host system. In the terminal of our Linux system, we check if the converter is detected by the system, to do so we put the following command-

ls /dev => lists the devices attached to the system

OR

dmesg => lists the ongoing device files/modules running

The first command will give you the list of devices attached to the system or the ones that it currently holds active. Here in the list you will find “ttyUSB0” this means that the system has detected the converter. So, we give further commands-

sudo apt install minicom

sudo minicom -s

The first command installs minicom on the system. The second command opens the settings of the minicom, here the “-s” suggests settings to the system. In the settings we go to the serial Port settings edit the port name as ttyUSB0 and save the settings as default after that we exit the settings of minicom and are directed to the

minicom terminal window. In the minicom window the basic information about our port is displayed now we power up the beaglebone board. The beaglebone has two ways to boot up the default boot is of OS debian which is through the eMMC. The boot up info of bootloader and starting kernel is displayed on the minicom terminal. The last 5-6 lines of the boot process display basic information of the board, its IP address and the username-password which we need to type again to access the terminal of the board. So after the boot we type debian (default username) and password is tempwd (also default which is already mentioned on the screen). This finishes the boot up process of the board and gives us access to the terminal of the board.

2.Beaglebone Black Rev C-

Understanding the boot up mechanism of the BeagleBone Black is important to be able to modify it.

The BeagleBone Black provides by alternative boot sequences which are selectable by the boot switch (S2). In default mode (S2 not pressed) it tries to boot from

- MMC1 (onboard eMMC)
- MMC0 (microSD)
- UART0
- USB0.

Usually, it will find something in the onboard eMMC and boot from there. If S2 is pressed during power-up the boot sequence is changed to

- SPI0
- MMC0 (microSD)
- UART0
- USB0.

Whichever way it is booted in, the basic power supply and led functionality is same. The basic LED sequence is same and works as follows-

1. Connect the small connector on the USB cable to the board as shown in Figure 3. The connector is on the bottom side of the board.



Figure 3. USB Connection to the Board

2. Connect the large connector of the USB cable to your PC or laptop USB port.
3. The board will power on and the power LED will be on as shown in Figure 4.

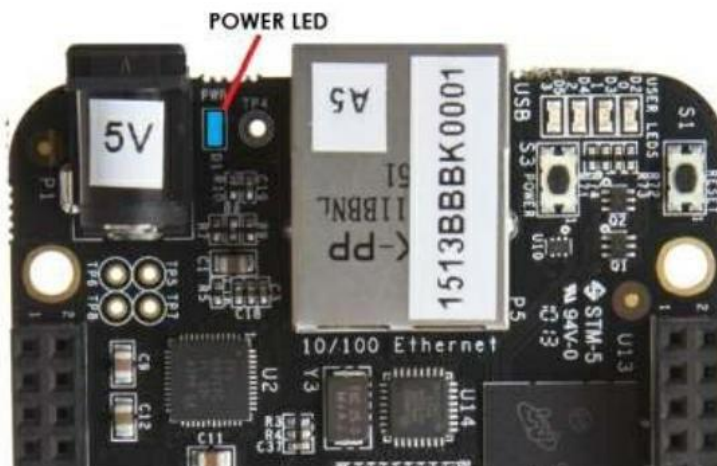


Figure 4. Board Power LED REF: BBONEBLK_SRM BeagleBone Black System Reference Manual Rev A5.2 Page 16 of 108

4. When the board starts to boot the LEDs will come on in sequence as shown in Figure 5 below. It will take a few seconds for the status LEDs to come on, so be patient. The LEDs will be flashing in an erratic manner as it boots the Linux kernel.

- USER0 is the heartbeat indicator from the Linux kernel.
- USER1 turns on when the SD card is being accessed.
- USER2 is an activity indicator. It turns on when the kernel is not in the idle loop.
- USER3 turns on when the onboard eMMC is being accessed.

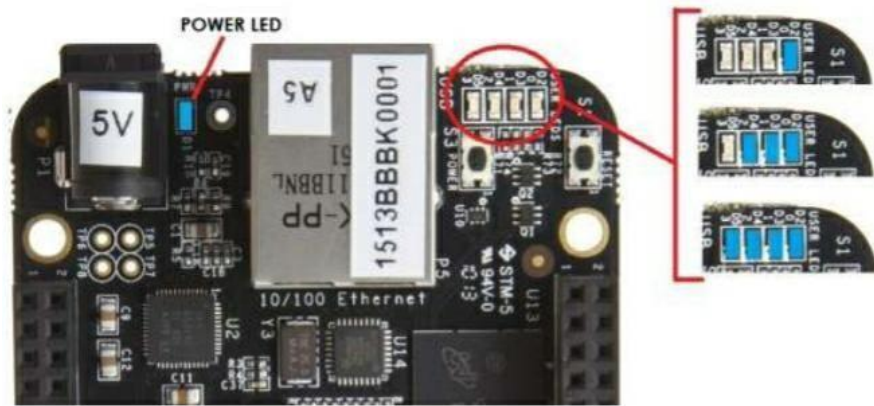


Figure 5. Board Boot Status

3.MPU6050-

MPU6050 sensor module is a complete 6-axis Motion Tracking Device. It combines 3-axis Gyroscope, 3-axis Accelerometer and Digital Motion Processor all in small package. It has I2C bus interface to communicate with the microcontrollers.

It has Auxiliary I2C bus to communicate with other sensor devices like 3-axis Magnetometer, Pressure sensor etc.

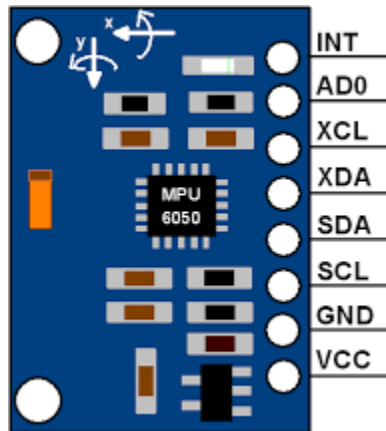


Figure 6. MPU6050 Pins

The MPU6050 has registers from 0x0D to 0x75. Each of these registers are 8-bits wide and have specific functions. The accelerometer and gyroscope data are 16 bits wide and so data from each axis uses two registers high and Low.

Register	Address
ACCEL_XOUT_H	0x3B
ACCEL_XOUT_L	0x3C
ACCEL_YOUT_H	0x3D
ACCEL_YOUT_L	0x3E
ACCEL_ZOUT_H	0x3F
ACCEL_ZOUT_L	0x40

GYRO_XOUT_H	0x43
GYRO_XOUT_L	0x44
GYRO_YOUT_H	0x45
GYRO_YOUT_L	0x46
GYRO_ZOUT_H	0x47
GYRO_ZOUT_L	0x48

Table 1. Register addresses for MPU6050

The registers on the table have zero values initially. This is because we need to wake the MPU6050 up before it gives data. To wake up the MPU6050, we need to write zero to the PWR_MGMT_1 register which is at address 0x6B.

The values received from the sensors are raw values which need to be limited within a certain range. By setting the FS_SEL value and AFS_SEL value we can select the range which means the maximum range you can measure.

In our case we set both FS_SEL and AFS_SEL 0 which means I can measure from -250deg/s to +250deg/s and from -2g to +2g respectively.

There is one more important value which rises when we configure these two registers. It is Sensitivity Scale Factor.

CONDITION	TYP
AFS_SEL_0	16384
AFS_SEL_1	8192
AFS_SEL_2	4096
AFS_SEL_3	2048

Table 2. Full-Scale modes for MPU6050 Accelerometer

CONDITION	TYP
FS_SEL_0	131
FS_SEL_1	65.5
FS_SEL_2	32.8
FS_SEL_3	16.4

Table 3. Full-Scale modes for MPU6050 Gyroscope

As shown above if we select *FS_SEL=0*, *Sensitivity Scale Factor becomes 131* which means angular velocity of 1 deg/sec is given as 131 in gyro sensor. Therefore,

angular velocity (deg/sec) can be obtained by dividing sensitivity scale factor (in my case 131).

Same method can be applied to the accelerometer. Since I set $AFS_SEL=0$, *Sensitivity Scale Factor becomes 16384* which means 1g is given as 16384 in accelerometer.

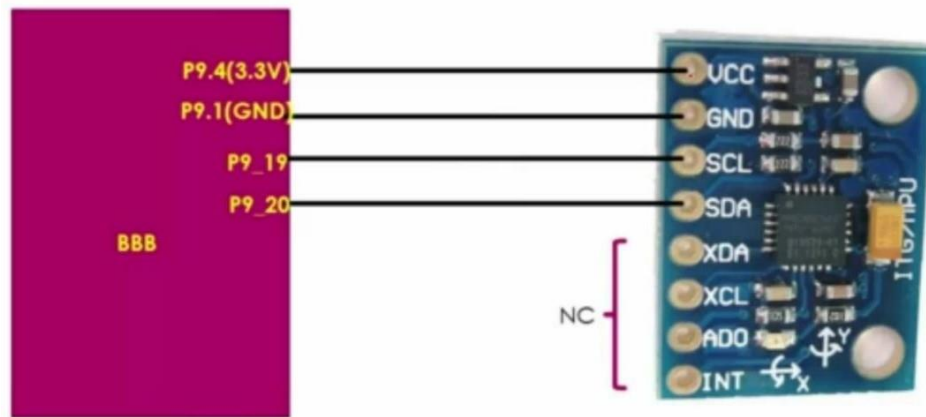


Figure 7. MPU 6050 to BBB Connection

Functions Used:

1. **void mpu6050_init ()**: To disable its sleep mode and also configure the full-scale ranges for gyro and acc.
2. **void mpu6050_read_acc (short int *pBuffer)**: read accelerometer values of x,y,z in to the buffer "pBuffer".
3. **void mpu6050_read_gyro (short *pBuffer)**: read gyro values of x,y,z in to the buffer "pBuffer".
4. **int mpu6050_read (uint8_t base_addr, char *pBuffer, uint32_t len)**: read "len" many bytes from "addr" of the sensor in to the address indicated by "pBuffer".
5. **int mpu6050_write (uint8_t addr, uint8_t data)**: write an 8bit "data" to the sensor at the address indicated by "addr".

4.I2C Protocol-

I2C stands for Inter-Integrated Circuit. It is a bus interface connection protocol incorporated into devices for serial communication. Recently, it is a widely used protocol for short-distance communication. It is also known as Two Wired Interface (TWI).

It uses only 2 bi-directional open-drain lines for data communication called SDA and SCL. Both these lines are pulled high; hence a pull-up resistor is required so that the lines are high since the devices on the I2C bus are active low.

- Serial Data (SDA) – Transfer of data takes place through this pin.
- Serial Clock (SCL) – It carries the clock signal.

I2C operates in 2 modes:

- Master mode
- Slave mode

Working of I2C Communication Protocol:

Each data bit transferred on SDA line is synchronized by a high to the low pulse of each clock on the SCL line.

According to I2C protocols, the data line cannot change when the clock line is high, it can change only when the clock line is low. The data is transmitted in the form of packets which comprises 9 bits.

START and STOP bits can be generated by keeping the SCL line high and changing the level of SDA. To generate START condition the SDA is changed from high to low while keeping the SCL high. To generate STOP condition SDA goes from low to high while keeping the SCL high, as shown in the figure below.

With I2C, data is transferred in *messages*. Messages are broken up into *frames* of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted.

The message also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame:

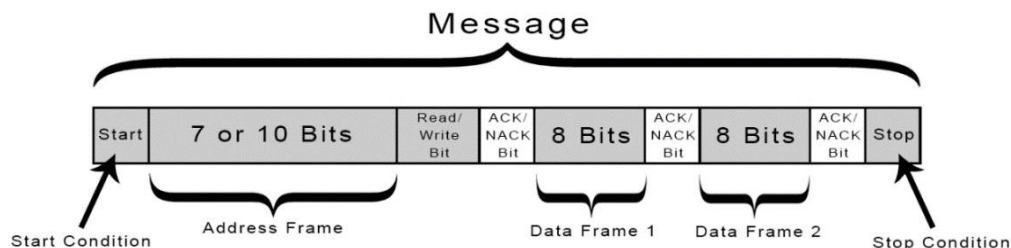


Figure 8. I2C protocol Message Frame Format

The slave address of MPU 6050 is 0X68

Single-Byte Write Sequence

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

Burst Write Sequence

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

Figure 9. Frame Format for Single-Byte and Burst Write sequence

Single-Byte Read Sequence

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

Burst Read Sequence

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

Figure 10. Frame Format for Single-Byte and Burst Read sequence

There are three I2C buses on the Beaglebone Black according to the AM335X Technical Reference Manual and their memory addresses are:

i2c-0: 0x44E0_B000

i2c-1: 0x4802_A000

i2c-2: 0x4819_C000

PIN	PROC	NAME	MODE0	MODE1	MODE2	MODE3	MODE4	MODE5	MODE6	MODE7
1,2						GND				
3,4						DC_3.3V				
5,6						VDD_5V				
7,8						SYS_5V				
9						PWR_BTN				
10	A10	SYS_RESETn	RESET_OUT							
11	T17	UART4_RXD	gpmc_wait0	mii2_crs	gpmc_csn4	mii2_crs_dv	mmc1_sdcd		uart4_rxd_mux2	gpio0[30]
12	U18	GPIO1_28	gpmc_be1n	mii2_col	gpmc_csn6	mmc2_dat3	gpmc_dir		mcasp0_aclkr_mux3	gpio1[28]
13	U17	UART4_TXD	gpmc_wpn	mii2_rxerr	gpmc_csn5	mii2_rxerr	mmc2_sdcd		uart4_txd_mux2	gpio0[31]
14	U14	EHRPWM1A	gpmc_a2	mii2_bxd3	rgmii2_tcl3	mmc2_dat1	gpmc_a18		ehrpwm1A_mux1	gpio1[18]
15	R13	GPIO1_16	gpmc_a0	gmii2_bxen	rgmii2_tcl1	mii2_bxen	gpmc_a16		ehrpwm1B_mux1	gpio1[16]
16	T14	EHRPWM1B	gpmc_a3	mii2_bxd2	rgmii2_tcl2	mmc2_dat2	gpmc_a19		ehrpwm1B_mux1	gpio1[19]
17	A16	I2C1_SCL	spi0_cs0	mmc2_sdwp	I2C1_SCL	ehrpwm0_synci				gpio0[5]
18	B16	I2C1_SDA	spi0_d1	mmc1_sdwp	I2C1_SDA	ehrpwm0_tripzone				gpio0[4]
19	D17	I2C2_SCL	uart1_rtsn	timer5	dcan0_rx	I2C2_SCL	spi1_cs1			gpio0[13]
20	D18	I2C2_SDA	uart1_ctsn	timer6	dcan0_tx	I2C2_SDA	spi1_cs0			gpio0[12]

Figure 11. Pin modes of BBB

The i2c-0 bus is not accessible on the header pins while the i2c-1 bus is utilized for reading EEPROMS on cape add-on boards and may interfere with that

function when used for other digital I/O operations. Hence, we use the i2c-2 bus on pins 19 and 20.

To find out which pins are in which modes by default we can refer to the Header pins map which gives the corresponding software pin number.

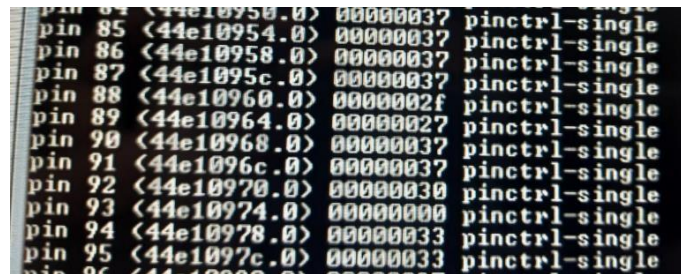
Header pin number	Software pin number
P9_17	87
P9_18	86
P9_19	95
P9_20	94

Table 4. Header and S/W pin table

The corresponding modes of the software pins can be checked in:

/sys/kernel/debug/pinctrl/44e10800.pinmux/cat pins

As seen in the image below the pins 87 & 86 are by default in mode-7. Whereas pins 95 and 94 are by default in mode-3 which corresponds to I2C2_SCL & I2C2_SDA respectively.



```
pin 84 (44e10950.0) 00000037 pinctrl-single
pin 85 (44e10954.0) 00000037 pinctrl-single
pin 86 (44e10958.0) 00000037 pinctrl-single
pin 87 (44e1095c.0) 00000037 pinctrl-single
pin 88 (44e10960.0) 0000002f pinctrl-single
pin 89 (44e10964.0) 00000027 pinctrl-single
pin 90 (44e10968.0) 00000037 pinctrl-single
pin 91 (44e1096c.0) 00000037 pinctrl-single
pin 92 (44e10970.0) 00000030 pinctrl-single
pin 93 (44e10974.0) 00000000 pinctrl-single
pin 94 (44e10978.0) 00000033 pinctrl-single
pin 95 (44e1097c.0) 00000033 pinctrl-single
pin 96 (44e10980.0) 00000033 pinctrl-single
```

Figure 10. Default modes of S/W pins

5. Eclipse IDE-

The Eclipse IDE is a multi-language integrated development environment which can be used to develop, install, and debug embedded applications. For this project we used Eclipse IDE to build and load the code into BeagleBone Black. The following steps were followed to setup Eclipse IDE:

- **Downloaded the latest Java JDK:** From [Java Downloads | Oracle](#) and extract the tar file in the location `/usr/lib/jvm`. We then changed the environment variables `JAVA_HOME` and `PATH`.
- **Install the ARM EABI Cross Toolchain:** To compile programs for ARM processors we will need a cross-compiler toolchain. So we downloaded the

toolchain from <http://releases.linaro.org/components/toolchain/binaries/> and extracted it.

- **Install and extract Eclipse IDE:** From the website www.eclipse.org downloaded the Eclipse IDE for C/C++ Developers for Linux and extracted it.
- **Eclipse Project creation and Build settings:** A new workspace was created for project launching.
 - Create new project from File-->New-->C Project
 - In project type select Empty project and in Toolchains select Cross GCC. Click on Next.
 - In select Configurations open Advanced Settings-->C/C++ Build-->Environment-->PATH-->"Add the path of /bin of extracted toolchain."
 - In GCC Cross Command add the **Cross Compiler Prefix** as "arm-linux-gnueabi-" and **Cross Compiler Path** as "path of /bin of extracted toolchain"
- **Implement code and Build project:** Created a new src folder within project and a new file named "Mpu6050.c". After implementing logic, Build the project and checked if Binary file is generated in the "Binaries" folder.
- **Load binary into BeagleBone Black:** The following steps were followed before loading the binary into BeagleBone Black:
 - Installing Remote System Explorer:
 - In Help-->Install New Software-->Work with: All Available Sites-->Mobile and Device Development-->Next
 - Agree and install the software.
 - After installation is complete:
 - Window-->Show View-->Other-->Remote Systems-->Remote Systems
 - A New window will appear at the bottom:
 - Click on Define a connection to remote system-->Select Remote System Type: SSH Only-->Host name: IP Address of BBB (192.168.7.2)-->Connection name: Anything and Finish.
 - After connecting the BBB to PC via Minicom we will be able to see our BBB files on the newly created Remote system connection under Sftp files.
 - Drag and Drop our Application Binary into any Folder of BBB.

- In order to be able to execute the binary in our BBB, we need to give the file execution permission: “chmod +x “binary name””
- The output of the Code can be checked either using Minicom or the terminal within Eclipse IDE.

6.Application code-

The main application program is written in the Eclipse IDE in C language. The program can be divided into 3 parts for better understanding viz definitions, functions and main.

The first part is definitions, we have to define macros to the address spaces of the MPU for easy access of those addresses later in the program. These are the addresses of mpu6050 from which you will fetch accelerometer x,y,z high and low values

```
#define MPU6050_REG_ACC_X_HIGH    0x3B
#define MPU6050_REG_ACC_X_LOW    0x3C
#define MPU6050_REG_ACC_Y_HIGH    0x3D
#define MPU6050_REG_ACC_Y_LOW    0x3E
#define MPU6050_REG_ACC_Z_HIGH    0x3F
#define MPU6050_REG_ACC_Z_LOW    0x40
```

These are the addresses of mpu6050 from which you will fetch gyro x,y,z high and low values

```
#define MPU6050_REG_GYRO_X_HIGH    0x43
#define MPU6050_REG_GYRO_X_LOW    0x44
#define MPU6050_REG_GYRO_Y_HIGH    0x45
#define MPU6050_REG_GYRO_Y_LOW    0x46
#define MPU6050_REG_GYRO_Z_HIGH    0x47
#define MPU6050_REG_GYRO_Z_LOW    0x48
```

We also define macros for the different sensitivities at which the mpu 6050 can work. Following are the sensitivities offered by the accelerometer-

```
#define ACC_FS_SENSITIVITY_0          16384
#define ACC_FS_SENSITIVITY_1          8192
#define ACC_FS_SENSITIVITY_2          4096
#define ACC_FS_SENSITIVITY_3          2048
```

Following are the sensitivities offered by the gyroscope-

```
#define GYR_FS_SENSITIVITY_0          131
#define GYR_FS_SENSITIVITY_1          65.5
#define GYR_FS_SENSITIVITY_2          32.8
#define GYR_FS_SENSITIVITY_3          16.4
```

This is the I2C slave address of mpu6050 sensor

```
#define MPU6050_SLAVE_ADDR 0x68
#define MAX_VALUE 50
```

This is the linux OS device file for hte I2C3 controller of the SOC

```
#define I2C_DEVICE_FILE  "/dev/i2c-2"
int fd;
```

After this definition of macros we move on to the read and write function of the dev file and also the init function of the mpu 6050. Then we define the accelerometer and gyroscope read function.

In the int mpu6050_write(uint8_t addr, uint8_t data) function we basically write a 8bit "data" to the sensor at the address indicated by "addr" and the int mpu6050_read(uint8_t base_addr, char *pBuffer,uint32_t len) function is to read "len" many bytes from "addr" of the sensor in to the address indicated by "pBuffer". The mpu 6050 is by default in sleep mode so the mpu_6050_init function disables its sleep mode and also configure the full-scale ranges for gyro and acc as well. The last two functions are read functions of accelometer and gyroscope, since the axis value is 2 bytes a buffer is defined inside the functions of 6 bytes.

```
void mpu6050_read_acc(short int *pBuffer)
```

=>read accelerometer values of x,y,z in to the buffer "pBuffer"

```
void mpu6050_read_gyro(short *pBuffer)
```

=>read gyro values of x,y,z in to the buffer "pBuffer"

Now comes the main function where everything comes into action. Firstly we open the I2C device file using open function then we set the I2C address using ioctl I2C_SLAVE command then we disable the sleep mode of mpu6050 for this we call the init function. After this we read the values and compute them and display.

```
int main(void)
{
    short acc_value[3],gyro_value[3];
    double accx,accy,accz,gyrox,gyroy,gyroz;
    if ((fd = open(I2C_DEVICE_FILE,O_RDWR)) < 0) {
        perror("Failed to open I2C device file.\n");
        return -1;
    }
    if (ioctl(fd,I2C_SLAVE,MPU6050_SLAVE_ADDR) < 0) {
        perror("Failed to set I2C slave address.\n");
        close(fd);
        return -1;
    }
    mpu6050_init();
    while(1)
    {
        mpu6050_read_acc(acc_value);
        mpu6050_read_gyro(gyro_value);
        /*Convert acc raw values in to 'g' values*/
        accx = (double) acc_value[0]/ACC_FS_SENSITIVITY_0;
```

```

    accy = (double) acc_value[1]/ACC_FS_SENSITIVITY_0;
    accz = (double) acc_value[2]/ACC_FS_SENSITIVITY_0;
    /* Convert gyro raw values in to "°/s" (deg/seconds) */
    gyro_x = (double) gyro_value[0]/GYR_FS_SENSITIVITY_0;
    gyro_y = (double) gyro_value[1]/GYR_FS_SENSITIVITY_0;
    gyro_z = (double) gyro_value[2]/GYR_FS_SENSITIVITY_0;

    /* print just the raw values read */
    printf("Acc(raw)=>X:%dY:%dZ:%d gyro(raw)=>X:%dY:%dZ:%d\n",
    \acc_value[0],acc_value[1],acc_value[2],gyro_value[0],gyro_value[1],gyro_value[
    2]);

    /* print the 'g' and '°/s' values */
    printf("Acc(g)=> X:%.2f Y:%.2f Z:%.2f gyro(dps)=> X:%.2f Y:%.2f Z:%.2f \n",
    \accx,accy,accz,gyrox,gyroy,gyroz);

    }

}

```

7. Operating System-

The second part of the project is to create our own OS and run the executable generated on eclipse IDE. The basic requirement for any OS is -

- Bootloader
- Kernel
- Root file system

So to make our own OS we first downloaded a bootloader and extracted it in copied it in the boot partition. The next step is to download the Kernel, extract and copy the extracted files and images into Boot partition as well. Finally copy the extracted root file system into the rootfs partition.

We start of by formatting our SD card and creating partitions which can be done in the terminal using fdisk command. For that our SD card should be loaded as a sdb in the system. Another way to part the SD card is by using the Gparted

application. If we choose to do this process through fdisk we are simply suppose to give command-

sudo fdisk /dev/sdb

This will take us into the fdisk terminal which is like the gdb terminal with its own commands like-

p => print

d => delete

n => create new partition

a => make partition as boot

The first partition that is the boot partition should be of about 1024 bytes that is around 1 GB and it is of type fat32 type. The second partition consists of all the remaining capacity of the SD card. This partition is the rootfs partition of type ext4 this contains the root file system. This process can be done using the gparted application as well.

For the downloads we used the following-

<https://angstrom.s3.amazonaws.com/demo/beagleboard/index.html>

<https://source.denx.de/u-boot/u-boot/-/tree/v2020.04>

<https://www.kernel.org/>

Once the downloads are completed and all the files are extracted, we are supposed to copy all the files extracted from the Angstrom-Beagleboard-demo-image-glibc-ipk-2011.1-beagleboard.rootfs.tar.bz2 into the rootfs partition. From the u-boot extractions we have to copy MLO image and u-boot.img into the boot partitions. Similarly, from the linux kernel extractions we have to copy zImage which is the bootable kernel image available in arch/arm/boot of the Linux source.

We also need device tree, blob am335x-boneblack.dtb and copy it in the boot partition as well. While powering up the board we press the boot button and power on by using usb cable. The board automatically boots from the SD card showing u-boot prompt. Press space bar to abort autoboot. Then set the bootargs for the kernel using setenv command:

**=>setenv bootargs console=ttyS0,115200 rw root=/dev/mmcblk0p2
rootfstype=ext4 rootwait**

Load the kernel and device tree into RAM

=> load mmc 0:1 0x82000000 zImage

=> load mmc 0:1 0x84000000 am335x-boneblack.dtb

Boot the kernel from RAM using bootz

=> bootz 0x82000000 – 0x84000000

After the board is booted through the SD card we copy the executable generated in eclipse IDE into the system and run file to get the output.

3.

SCHEMATIC DIAGRAM

3.1 Block diagram

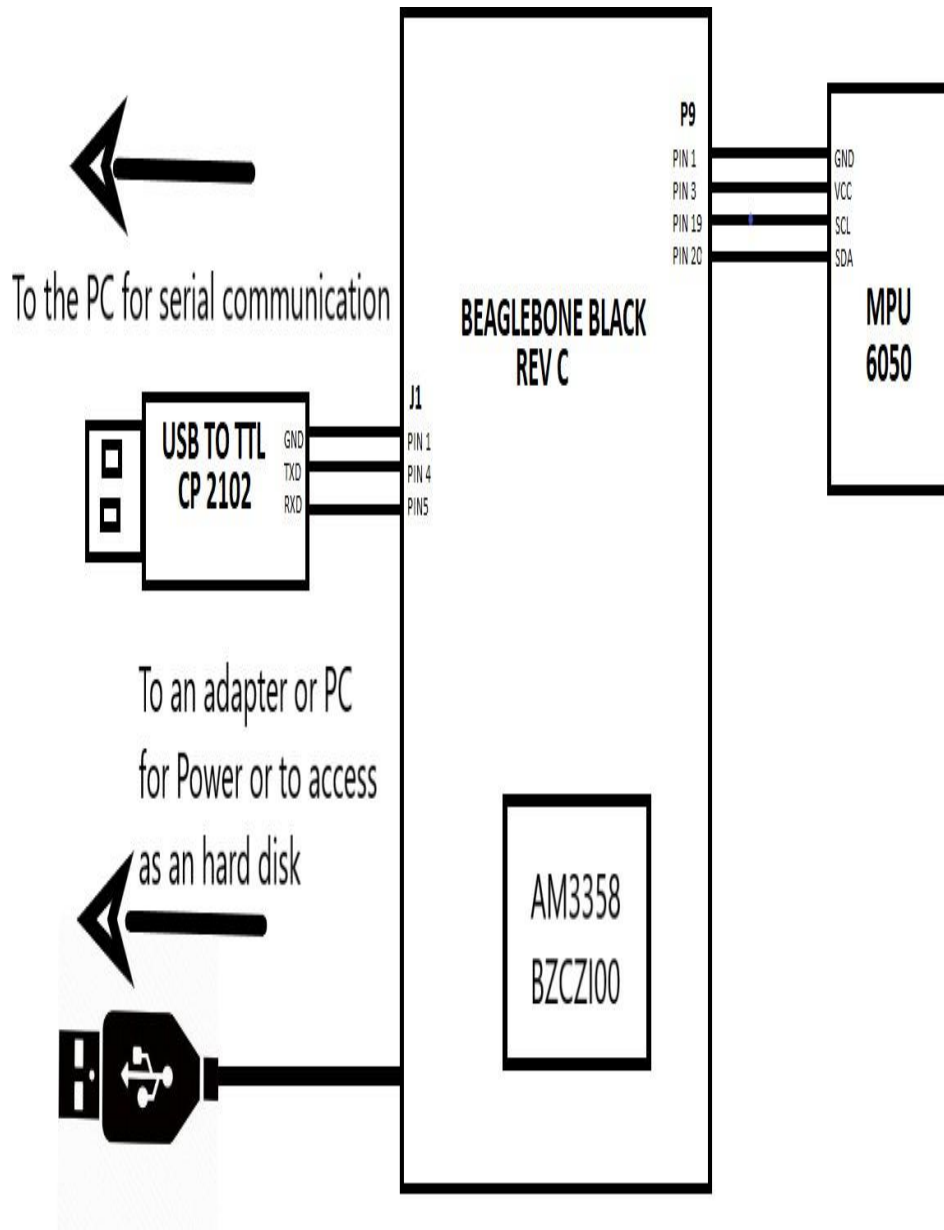


Figure 11. Block Diagram of the Interface

3.2 Implemented system:

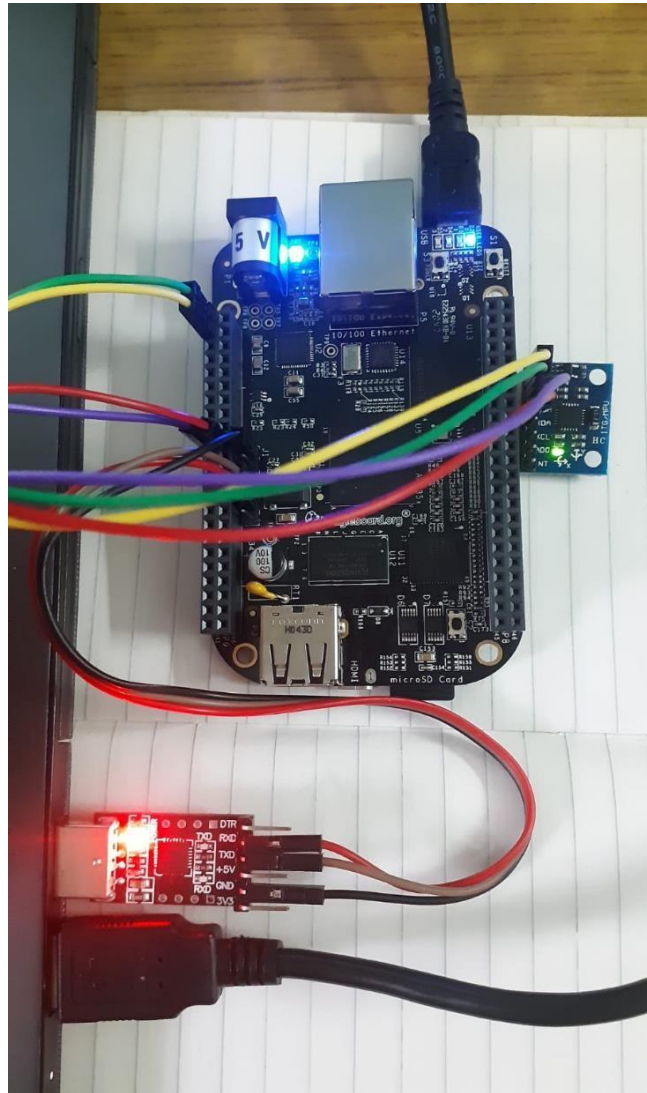


Figure 12.Implemented System

4.

FLOW OF PROJECT WORK

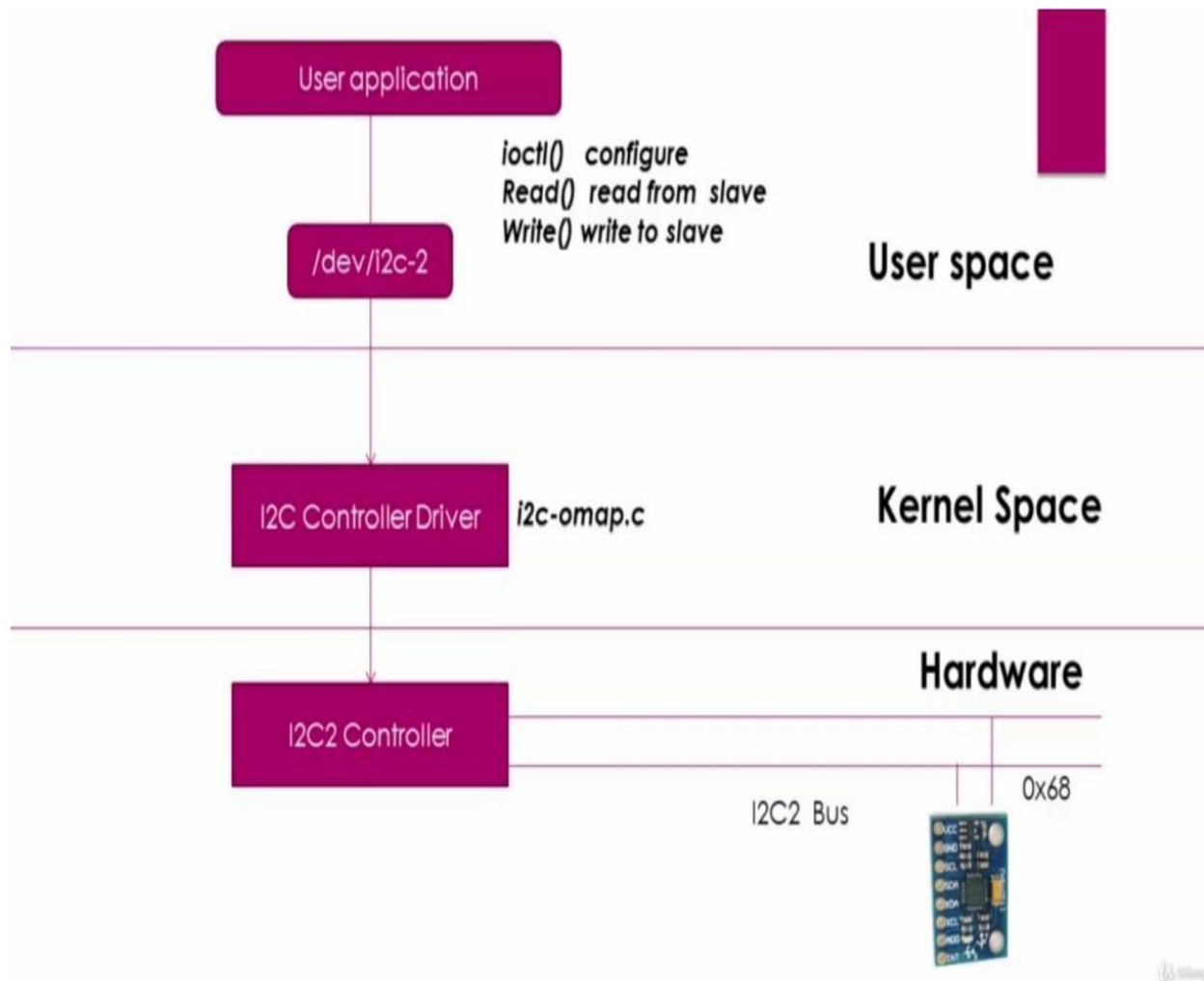


Figure 13. Workflow of the code

This diagram represents the workflow that is the path of accessing every file along the way to get the output. This tells us about which file whether the application file or device driver file where it exists that is in which space and the chronological order of opening, accessing, fetching, computing and returning the output to the application program. The `ioctl` plays an important role in communicating with the device driver.

Based on the raw values and sensitivity we calculated the acceleration in terms of 'g' along the X, Y, Z axis and plotted a graph as shown in figure 14. Similarly, we also calculated the rotational velocity that is '°/s' as plotted in the graph in the figure 15.

Graph for Accelerometer X, Y, Z axes -

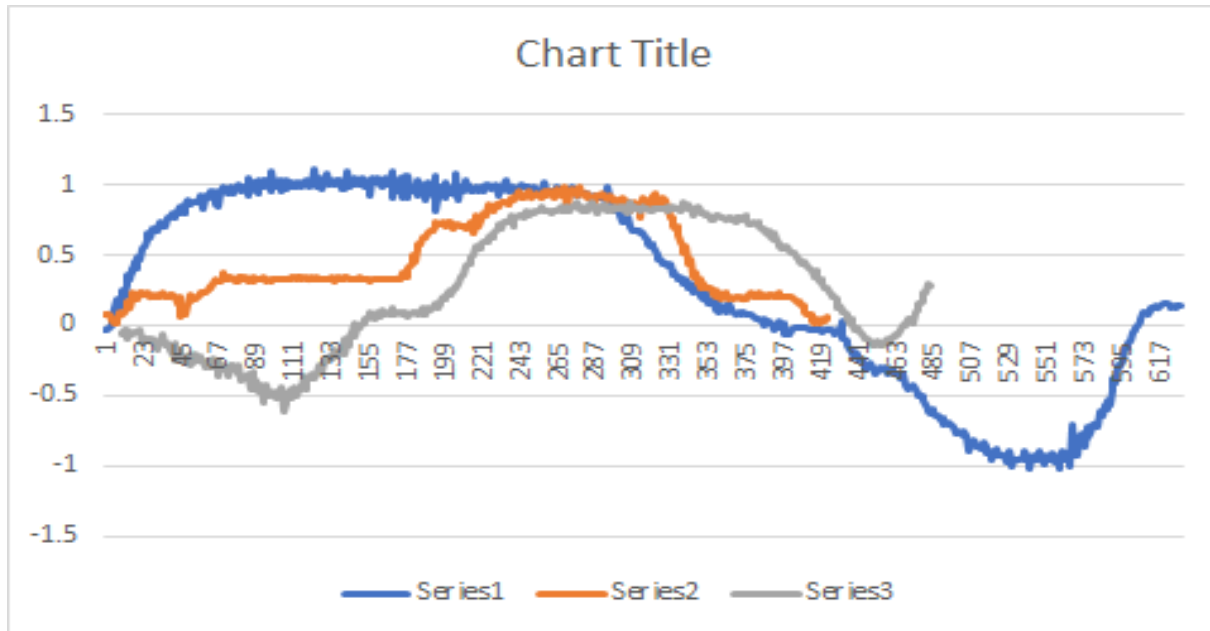


Figure 14. Acceleration along X, Y, Z

Graph for Gyroscope X, Y, Z axes -

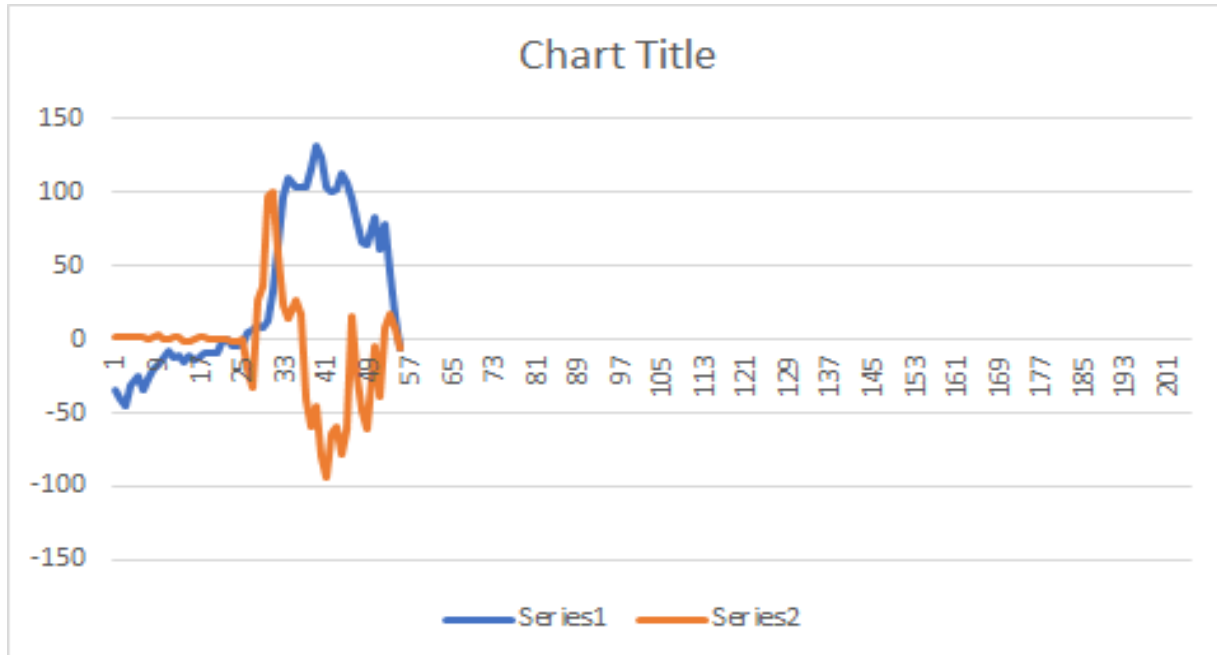


Figure 15. Rotational velocity along X, Y axis

5.

LITERATURE SURVEY

In Literature Survey we have done research on two papers. The first paper is **Design of Angle Detection System Based on MPU6050** by Jian Huang and second paper is **Flex sensors and MPU6050 sensors responses on smart glove for sign language translation** by A Yudhana, J Rahmawan and C U P Negara.

Paper-1

Survey on **Design of Angle Detection System Based on MPU6050** by Jian Huang, XiJing University, Xi'an, Shaanxi, China at Advances in Computer Science Research (ACSR), volume 73, 7th International Conference on Education, Management, Information and Computer Science (ICEMC 2017).

Abstract:

This paper describes the method of using MPU6050 to detect the dip angle, and designs the hardware circuit. In order to make the detection angle more accurate, the Calman filter algorithm is used in the software programming, which can effectively remove the interference and make the measurement precision higher. In the test, the experimental platform is built, which can accurately measure the horizontal angle and vertical angle, this design can be used for the inclination detection of flight control. It has a practical value.

Summary:

In this paper, the principle of using MPU6050 to detect the inclination is described. The hardware circuit is designed and the software is programmed. In the program, the MPU6050 is initialized, and the DMP method is used to detect the inclination angle, which reduces the data fusion and the interference. The test results show that the design can accurately detect the angle of moving objects, and has a certain practical value.

Paper-2

Flex sensors and MPU6050 sensors responses on smart glove for sign language translation by A Yudhana, J Rahmawan and C U P Negara., Indonesia at 2017 1st International Conference on Engineering and Applied Technology (ICEAT).

Abstract:

Flex sensor is a transducer that measures the level of curvature of the sensor. One type of flex sensor is a resistive flex sensor. This type of sensor has output resistance value that proportionally to the level of flexibility of the sensor. The gyroscope and accelerometer sensor used in the research is the MPU6050 that has measurements on earth gravity. The glove was attached by the flex sensor and gyroscope sensors for sensing fingers and hand movement. This paper presented data on flex and MPU6050 sensor response. The microcontroller is used in the research was Arduino Nano controller to read flex sensors and MPU6050 sensors. It has a practical value.

Summary:

This paper has presented data flex and accelerometer sensor in forming gesture sign language. Flex sensor placed on each finger and MPU6050 placed on the hand. Then the sensor is read using the Arduino Nano controller, and is displayed on the PC using the serial path. Hand finger forms a certain curve and angle according to the gesture of sign language and the accelerometer's flex sensor reads each flex and slope and obtained data according to the paper presented. Using additional accelerometer sensors can detect gestures that have a different slope than other gestures, such as the letters G, H, J, P and Q. The gesture accelerometer value has a significant difference value than the other letters

6.

CONCLUSION

In this way the project of Tilt Sensor using MPU6050 interfaced with Beaglebone Black was implemented. The project has several applications in real life like posture detection, changing mobile phones orientation from portrait to landscape, to monitor health of buildings and Cruise/Aircraft control mechanisms. The sensor was moved and rotated in different directions across all three axes. The corresponding raw and 'g' values were noted and plotted to not the change across each axis.

REFERENCES

1. Yan Li, Shen Mingxia, Yao Wen, Lu Mingzhou,, et al. Method of gesture recognition for lactating sows based on MPU6050 sensor [J], Chinese Journal of agricultural machinery,.2015,46 (5): 279-285.
2. A Yudhana, J Rahmawan and C U P Negara,, based on Flex sensors and MPU6050 sensors responses on smart glove for sign language translation, Indonesia at 2017 IOP Conf. Series: Materials Science and Engineering 403 (2018) 012032 doi:10.1088/1757-899X/403/1/012032
3. BBB_SRM Datasheet-
https://cdn-hop.adafruit.com/datasheets/BBB_SRM.pdf
4. MPU6050 Datasheet - <https://pdf1.alldatasheet.com/datasheet-pdf/view/1132807/TDK/MPU-6050.html>