2020F_ESE_3014_1

SEMESTER: 3rd SEM

INSTRUCTOR: Prof. Linchen Wang

LAB 4

SUBMISSION DATE: 29th Oct, 2020

NAME AND ID:

VISHAL HASRAJANI (C0761544)

Goutham Reddy Alugubelly (C0747981)

PARTH PATEL (C0764929)

INTRODUCTION:

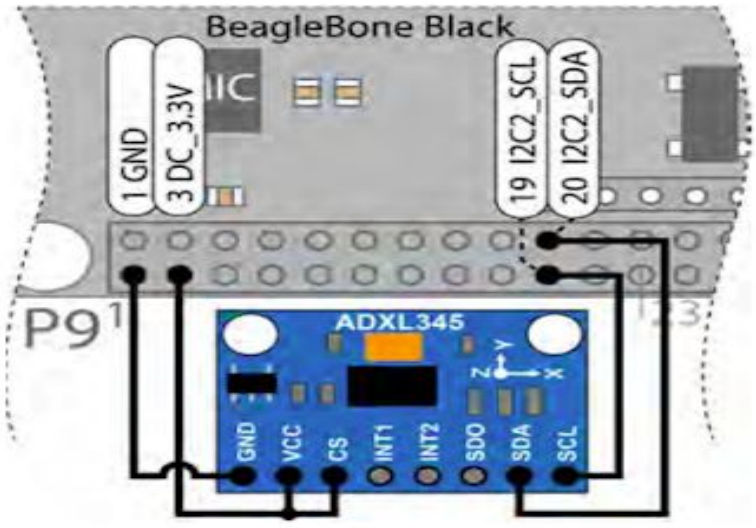
I2c is the most famous communication protocol that is utilized by the different devices and sensors.

In this lab, we will set up and achieve I2C interfacing communication for BBB as master and ADXL345 Accelerometer as slave.

DESCRIPTION :

At first, we connected the beaglebone black with an ADXL345 accelerometer.

- Connecting BeagleBone Black with ADXL345

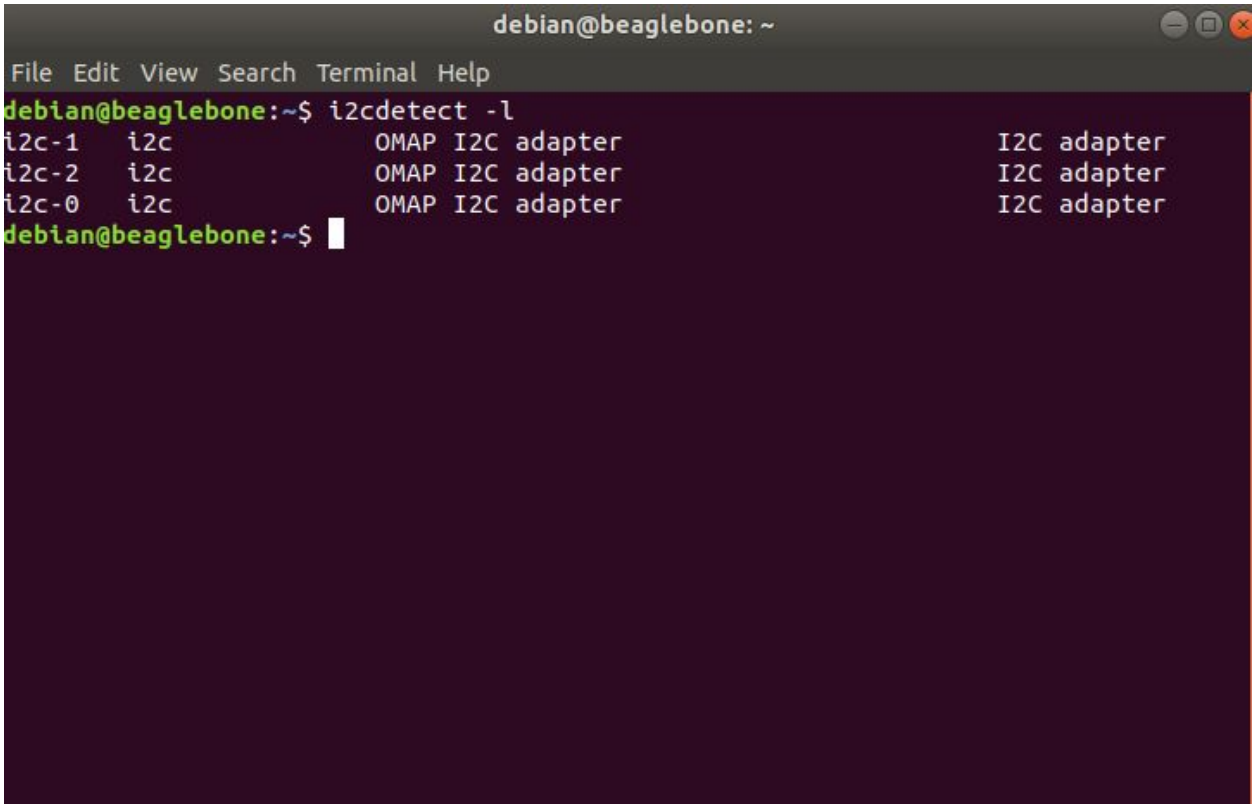


Here including these connections we also connected the SDO pin to GND.

After the connections were made,we checked whether the Accelerometer is detected by the beaglebone or not.

So in order to check if the ADXL345 is configured properly, we used the command in terminal as:

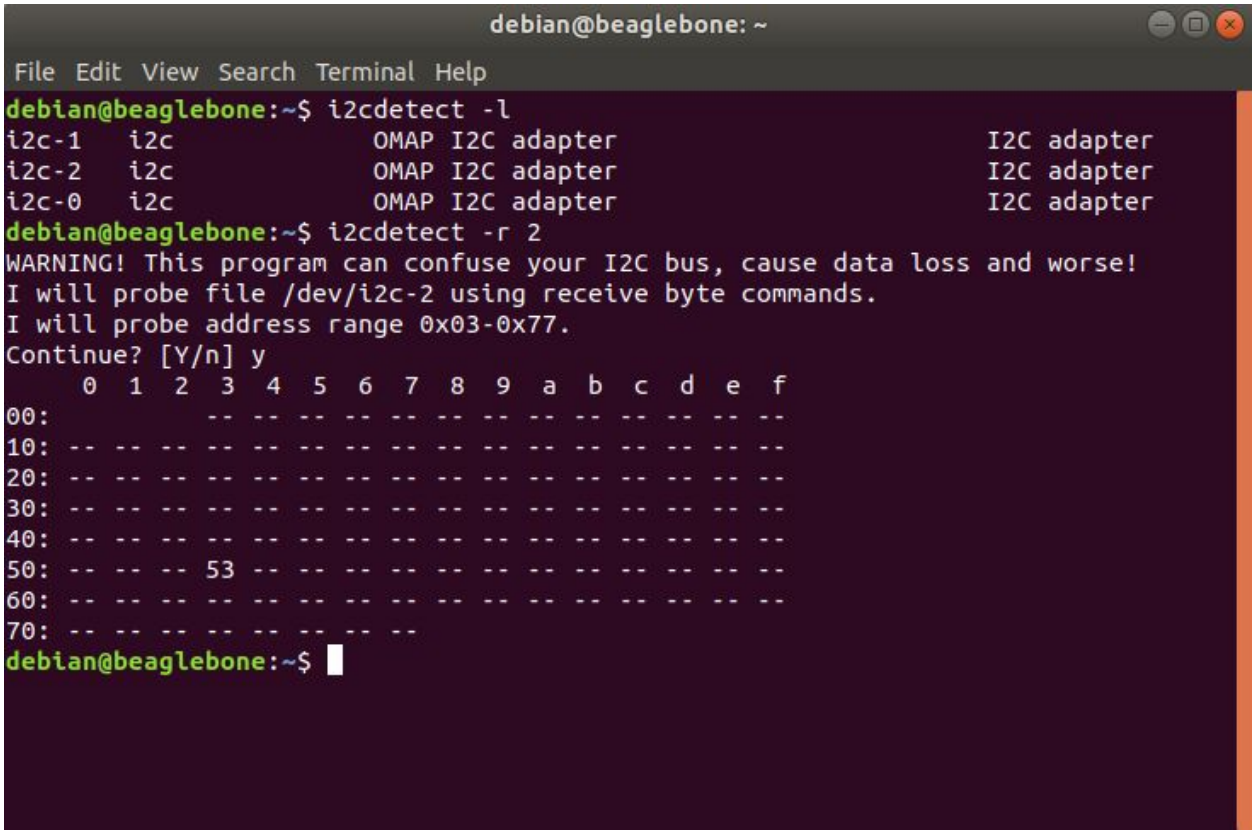
1) i2cdetect -l



So these are the i2c protocols that are available.But usually the devices are connected at i2c-2.

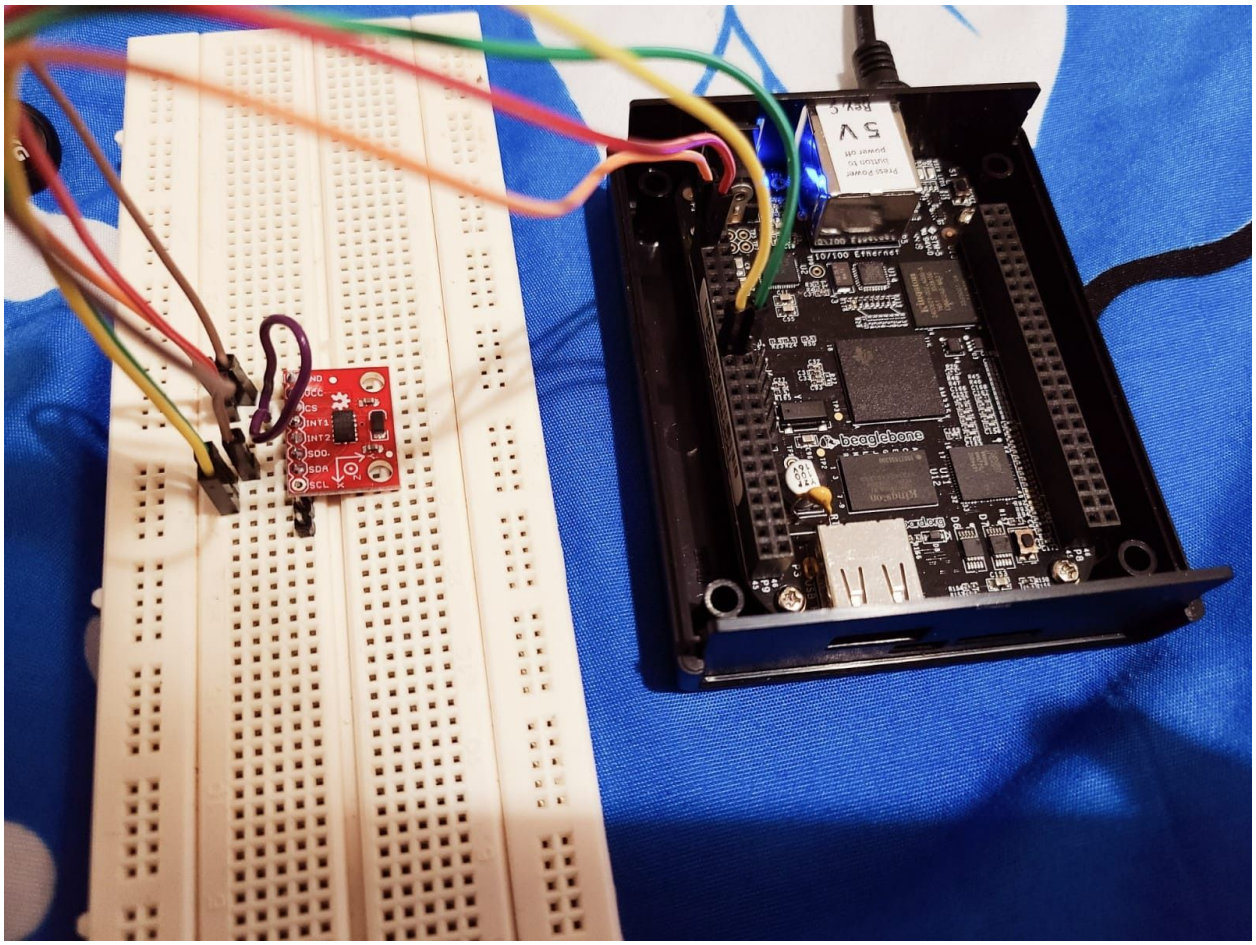
So,we referred the i2c-2 pin to check whether it is there or not.

2) i2cdetect -r 2



So, here,we can see that our ADXL345 is detected properly by beaglebone black and has an address of 53.

Actual Connections :



Now, in order to get the values of x-axis , y-axis and z-axis, we wrote a c code .

The explanation of the c code is shown is the recorded video .

The link is below :

https://www.youtube.com/watch?v=L7F7VS2Cb3M

C code :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>

#define ADXL345_SLAVE_ADDR                  0x53
         // Setting ADXL345 as a Slave with Slave Address
#define I2C_FILE_PATH_BEAGLEBONE          "/dev/i2c-2"
         //  Path of files for the i2c-2.

// Registers taken from the ADXL345 Datasheet
#define ADXL345_REG_BW_RATE      0x2C
#define ADXL345_REG_POWERCTL     0x2D
#define ADXL345_REG_DATA_FORMAT      0x31
#define ADXL345_REG_DATAX0       0x32
#define ADXL345_REG_DATAX1       0x33
#define ADXL345_REG_DATAY0       0x34
#define ADXL345_REG_DATAY1       0x35
#define ADXL345_REG_DATAZ0       0x36
#define ADXL345_REG_DATAZ1       0x37

int point_file; //pointing to file
```

```
int adxl345_init()
{
     // Buffer with 2 elements to hold the data and
address.
     char Buff_write[2]; //it is 8 bit buffer

     // Opening up file
     if ((point_file=open(I2C_FILE_PATH_BEAGLEBONE,
O_RDWR)) < 0)
     {
          perror("Failed to open the bus. \n");
          return -1;
     }

     // Connecting to slave
     if ((ioctl(point_file, I2C_SLAVE, ADXL345_SLAVE_ADDR))
< 0)
     {
          perror("Failed to connect to the sensor\n");
          return -1;
     }

     // Configuring the BW Rate to 3200 and turn low power
off
     Buff_write[0] = ADXL345_REG_BW_RATE;
     Buff_write[1] = 0x0A;
     if (write(point_file, Buff_write, 2) != 2)
     {
          perror("Failed to write to the register\n");
          return -1;
     }
```

```
    // Configuring the Data Format register to set full
resolution
    Buff_write[0] = ADXL345_REG_DATA_FORMAT;
    Buff_write[1] = 0x08;
    if (write(point_file, Buff_write, 2) != 2)
    {
        perror("Failed to write to the register\n");
        return -1;
    }

    // Configuring the Power Control register to turn on
measurment mode
    Buff_write[0] = ADXL345_REG_POWERCTL;
    Buff_write[1] = 0x08;
    if (write(point_file, Buff_write, 2) != 2)
    {
        perror("Failed to write to the register\n");
        return -1;
    }

    sleep(1);
    close(point_file);

    return 0;
}

int adxl345_read()
{
    short x_axis, y_axis, z_axis; //we have selected short
as we want 16 bits

    // Opening up file
    if ((point_file=open(I2C_FILE_PATH_BEAGLEBONE,
```

```
O_RDWR)) < 0)
    {
        perror("\n");
        return -1;
    }

    // Connecting to slave
    if ((ioctl(point_file, I2C_SLAVE, ADXL345_SLAVE_ADDR))
< 0)
    {
        perror("Failed to connect to the sensor\n");
        return -1;
    }

    // Set the buffer pointer to the start of the data
registers (X0) and then increment the buffer.
    char buffer_pointer[1] = {0x32};
    if (write(point_file, buffer_pointer, 1) != 1)
    {
        perror("Failed to write to the register\n");
        return -1;
    }

    // Reading the six reisters of ADXL345
    char buffer_reg[6];
    if (read(point_file,  buffer_reg, 6) != 6)
    {
        perror("Failed to read from the buffer\n");
        return -1;
    }

    close(point_file);
```

```
/*
Left shifting the 8 bit data of MSB to the left by 8
bits(buffer_reg is of 16 bits) and then doing OR operation
with LSB bits to get the actual data.
*/
     x_axis = ((short)buffer_reg[1] << 8) |
(short)buffer_reg[0];
     y_axis = ((short)buffer_reg[3] << 8) |
(short)buffer_reg[2];
     z_axis = ((short)buffer_reg[5] << 8) |
(short)buffer_reg[4];

// getting the values ....

     printf("Accelerometer Readings\n");
        printf(" X:%d\n Y:%d\n Z:%d\n",
x_axis,y_axis,z_axis);

printf("****************************************\n");


     return 0;
}

int main(void)
{
     adxl345_init();
     usleep(1000);

     while(1)
     {
          adxl345_read();
```

```
            sleep(1);
        }
        return 0;
}
```

CONCLUSION :

To conclude , we can say that,in this lab we successfully implemented the i2c protocol communication between beaglebone black and ADXL345.