STUDENT TRAINING PROGRAM, 2016

A PROJECT REPORT ON

INTERFACE OF ADIS16362 INERTIAL SENSOR WITH RASPBERRY PI

BY :- ARUN C. SATHEESH

GUIDED BY :- B. THIYAGARAJAN

NATIONAL CENTRE FOR RADIO ASTROPHYSICS

TATA INSTITUTE OF FUNDAMENTAL RESEARCH

GIANT METREWAVE RADIO TELESCOPE

NATIONAL CENTRE FOR RADIO ASTROPHYSICS
TATA INSTITUTE OF FUNDAMENTAL RESEARCH

GIANT METREWAVE RADIO TELESCOPE
POST BAG NO. 6, NARAYANGAON, TAL-JUNNAR, PUNE - 410 504

CERTIFICATE

This is to certify that the project entitled "**Interface Of Adis16362 Inertial Sensor With Raspberry Pi**" submitted by *Arun C. Satheesh* in the fulfillment of the requirements for the successful completion of Student Training Program from 13th December 2016 to 12th June 2017 at Giant Metrewave Radio Telescope, is an autentic work carried out by him under my supervision and guidance.

Place : GMRT, Pune

Date : 12th June, 2017

B. Thiyagarajan
(Supervisor)

ACKNOWLEDGEMENT

It has been a great honour and privilege to undergo my project training at *Giant Metrewave Radio Telescope.* I show my gratitude to *Mr. B. Thiyagarajan* for providing me this opportunity to work under his guidance and providing me with the support and facilities regarding the project. I am highly thankful to everyone who have hepled me at various point of my time at the institute.

ARUN C. SATHEESH

ABSTRACT

Todays technology being everything about automation, accuration and use of instruments with less size, the use of MEMS devices with the help of microprocessor for measurement can be used to meet todays requirements.

The programming language used is C, which is the most common used language in the embedded programming.

The use of Raspberry Pi micoprocessor gives the compactness and power required by the system.

The BMC2835 library provides a connectivity between the the Sensor, Controller and the Code.

The MEMS sensor provides the accuracy and precision for a perfect result. With the help of micro-electronic technology the sensor can be stable and compact at the same time.

The microprocessor can handle the sensor clock at both normal and burst mode without any major adjustment, since the clock speed of the processor is set through the C program.

INDEX

1. INTRODUCTION

1.1 CONCEPT AND SPECIFICATION

The aim of interfacing a microprocessor and a sensor is the way of getting the parameters of the sensor and desplaying those values to the user in the human understanding language.

To recieve thoes values the code and controller should meet the sensor’s requirements.

The parameters is in the form of hexadecimal values, so the requesting way for the result is to ask the sensor in hexadecimal address.

The protocol SPI provides the communication between the controller and the sensor.

The Raspberry Pi is a ARM controller with 900MHz clock speed and a BCM2835 liberary that helps the user to interact with the protocol.

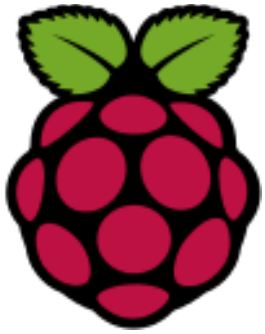
ADIS16362 is a sensor that maps the 6-degree inertia its movements helpful of acceleration and gyroscope determination.

1.2 REQUIREMENT

- **HARDWARE**
  - Operating power --> 4.75V to 5.25V
  - Clock speed --> 0.01 Mhz to 2 Mhz

- **SOFTWARE**
  - Processor --> ARM Cortex A7
  - Operating System --> Raspbian JESSIE OS
  - Kernel Version --> 4.4
  - Library --> BCM2835
  - Programming Language --> C

1.3 RASPBERRY PI



The ***RASPBERRY PI*** is a series of small single-board computers developed in the united kingdom by the Raspberry Pi foundation to promote the teaching of basic computer science in schools and in developing countries.The original model became far more popular than anticipated,selling outside of its target market for uses such as robotics. Peripherals (including keyboards, mice and cases) are not included with the Raspberry Pi. Some accessories however have been included in several official and unofficial bundles.

All models feature a Broadcom system on a chip (SoC), which includes an ARM compatible central processing unit (CPU) and an on-chip graphics processing unit (GPU, a VideoCore IV). CPU speed ranges from 700 MHz to 1.2 GHz for the Pi 3 and on board memory range from 256 MB to 1 GB RAM. Secure Digital (SD) cards are used to store the operating system and program memory in either the SDHC or MicroSDHC sizes. Most boards have between one and four USB slots, HDMI and composite video output, and a 3.5 mm phono jack for audio. Lower level output is provided by a number of GPIO pins which support common protocols like I²C. The B-models have an 8P8C Ethernet port and the Pi 3 and Pi Zero W have on board Wi-Fi 802.11n and Bluetooth.

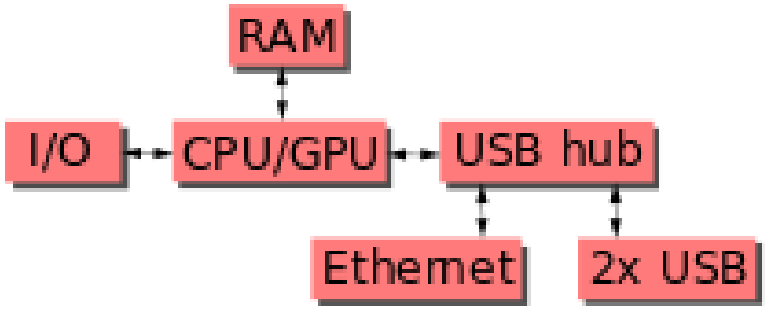
The Foundation provides ***Raspbian***, a Debian-based Linux distribution for download, as well as third party Ubuntu, Windows 10 IOT Core, RISC OS, and specialised media center distributions. It promotes Python and Scratch as the main programming language, with support for many other languages.

HARDWARE

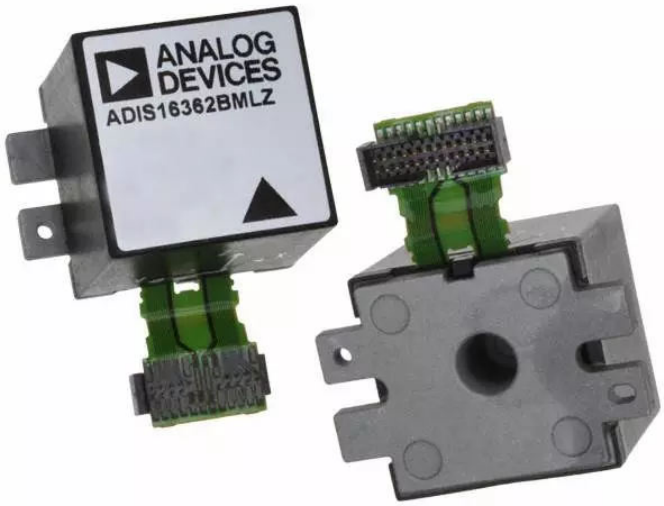
The Raspberry Pi 2 uses a Broadcom BCM2836 SoC with a 900 MHz 32-bit quad-core ARM Cortex-A7 processor, with 256 KB shared L2 cache.

Raspberry Pi 2 includes a quad-core Cortex-A7 CPU running at 900 MHz and 1 GB RAM. It is described as 4–6 times more powerful than its predecessor. The GPU is identical to the original. In parallelized benchmarks, the Raspberry Pi 2 could be up to 14 times faster than a Raspberry Pi 1 Model B+.

The Raspberry Pi 2 has a 1 GB of RAM.



1.4 ADIS16362



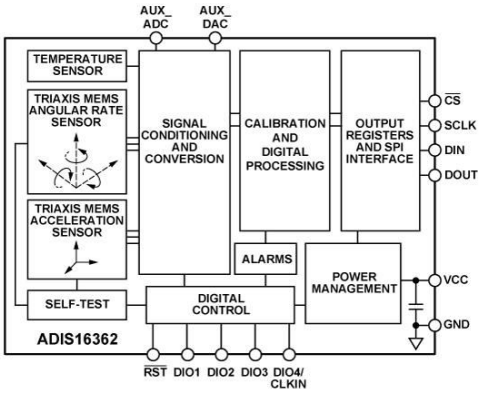
The ***ADIS16362*** is a complete inertial system that includes a triaxis gyroscope and triaxis accelerometer. Each sensor in the ADIS16362 combines industry-leading MEMS technology with signal conditioning that optimizes dynamic performance. The factory calibration characterizes each sensor for sensitivity, bias, alignment, and linear acceleration (gyro bias). As a result, each sensor has its own dynamic compensation formulas that provide accurate sensor measurements over a temperature range of −20°C to +70°C.

The ADIS16362 provides a simple, cost-effective method for integrating accurate, multiaxis, inertial sensing into industrial systems, especially when compared with the complexity and investment associated with discrete designs.

All necessary motion testing and calibration are part of the production process at the factory, greatly reducing system integration time. Tight orthogonal alignment simplifies inertial frame alignment in navigation systems.

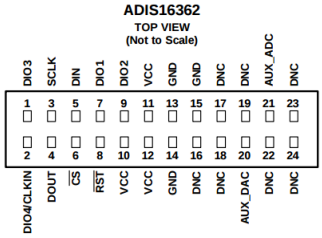
An improved SPI interface and register structure provide faster data collection and configuration control.

This compact module is approximately 23 mm × 23 mm × 23 mm and provides a flexible connector interface that enables multiple mounting orientation options.



HARDWARE

The sensor has 24 pins

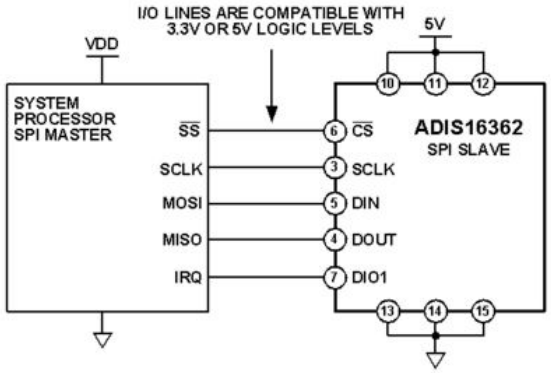


Pin No.	Mnemonic	Type[1]	Description
1	DIO3	I/O	Configurable Digital Input/Output.
2	DIO4/CLKIN	I/O	Configurable Digital Input/Output or Sync Clock Input.
3	SCLK	I	SPI Serial Clock.
4	DOUT	O	SPI Data Output. Clocks output on SCLK falling edge.
5	DIN	I	SPI Data Input. Clocks input on SCLK rising edge.
6	$\overline{CS}$	I	SPI Chip Select.
7, 9	DIO1, DIO2	I/O	Configurable Digital Input/Output.
8	$\overline{RST}$	I	Reset.
10, 11, 12	VCC	S	Power Supply.
13, 14, 15	GND	S	Power Ground.
16, 17, 18, 19, 22, 23, 24	DNC	N/A	Do Not Connect.
20	AUX_DAC	O	Auxiliary, 12-Bit DAC Output.
21	AUX_ADC	I	Auxiliary, 12-Bit ADC Input.

[1] I/O is input/output, I is input, O is output, S is supply, N/A is not applicable.

THEORY OF OPERATION

The ADIS16362 is an autonomous sensor system that starts up after it has a valid power supply voltage and begins producing inertial measurement data at the factory default sample rate setting of 819.2 SPS. After each sample cycle, the sensor data is loaded into the output registers, and DIO1 pulses high, which provides a new data ready control signal for driving systemlevel interrupt service routines. In a typical system, a master processor accesses the output data registers through the SPI interface.



MEMORY MAP

Name	R/W	Flash Backup	Address[1]	Default	Register Description
XACCL_OUT	R	No	0x0A	N/A	X-axis accelerometer output
YACCL_OUT	R	No	0x0C	N/A	Y-axis accelerometer output
ZACCL_OUT	R	No	0x0E	N/A	Z-axis accelerometer output

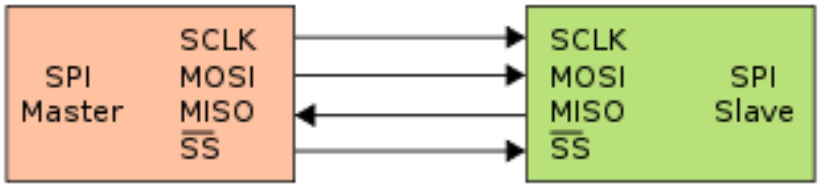
The address is a 16-bit value. The address of the lower byte is displayed. The address of the upper byte is equal to the address of the lower byte plus one.

1.5 SPI PROTOCOL

The ***Serial Peripheral Interface bus (SPI)*** is a synchronous serial communication interface specification used for short distance communication, primarily in embedded systems. The interface was developed by Motorola in the late 1980s and has become a de facto standard. Typical applications include Secure Digital cards and liquid crystal displays.

SPI devices communicate in full duplex mode using a master-slave architecture with a single master. The master device originates the frame for reading and writing. Multiple slave devices are supported through selection with individual slave select (SS) lines.

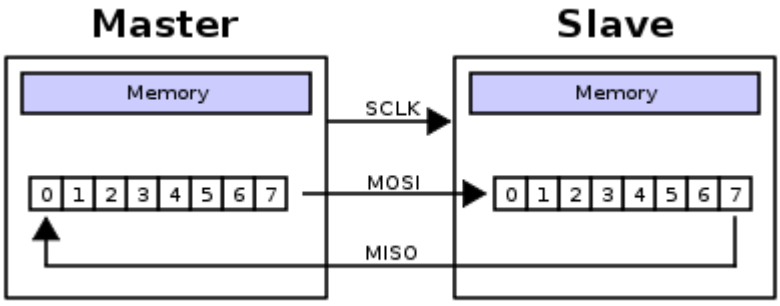
Sometimes SPI is called a four-wire serial bus, contrasting with three-, two-, and one-wire serial buses. The SPI may be accurately described as a synchronous serial interface, but it is different from the Synchronous Serial Interface (SSI) protocol, which is also a four-wire synchronous serial communication protocol. But SSI Protocol employs differential signaling and provides only a single simplex communication channel.



Data transmission

To begin communication, the bus master configures the clock, using a frequency supported by the slave device, typically up to a few MHz. The master then selects the slave device with a logic level 0 on the select line. If a waiting period is required, such as for an analog-to-digital conversion, the master must wait for at least that period of time before issuing clock cycles.

Transmissions normally involve two shift registers of some given word size, such as eight bits, one in the master and one in the slave; they are connected in a virtual ring topology. Data is usually shifted out with the most-significant bit first, while shifting a new least-significant bit into the same register. At the same time, Data from the counterpart is shifted into the least-significant bit register. After the register bits have been shifted out and in, the master and slave have exchanged register values. If more data needs to be exchanged, the shift registers are reloaded and the process repeats. Transmission may continue for any number of clock cycles. When complete, the master stops toggling the clock signal, and typically deselects the slave.



1.6 BCM2835 LIBRARY

A C library for Broadcom BCM 2835 as used in Raspberry Pi.

This is a C library for Raspberry Pi (RPi). It provides access to GPIO and other IO functions on the Broadcom BCM 2835 chip, as used in the RaspberryPi, allowing access to the GPIO pins on the 26 pin IDE plug on the RPi board so you can control and interface with various external devices.

It provides functions for reading digital inputs and setting digital outputs, using SPI and I2C, and for accessing the system timers. Pin event detection is supported by polling.

LIBRARY FOR SPI PINS

The bcm2835_spi_* functions allow you to control the BCM 2835 SPI0 interface, allowing you to send and received data by SPI (Serial Peripheral Interface).

When **bcm2835_spi_begin()** is called it changes the bahaviour of the SPI interface pins from their default GPIO behaviour in order to support SPI. While SPI is in use, you will not be able to control the state of the SPI pins through the usual bcm2835_spi_gpio_write(). When **bcm2835_spi_end()** is called, the SPI pins will all revert to inputs, and can then be configured and controled with the usual bcm2835_gpio_* calls.

The Raspberry Pi GPIO pins used for SPI are:

- P1-19 (MOSI)
- P1-21 (MISO)
- P1-23 (CLK)
- P1-24 (CE0)
- P1-26 (CE1)

Although it is possible to select high speeds for the SPI interface, up to 125MHz (**bcm2835_spi_setClockDivider()**) you should not expect to actually achieve those sorts of speeds with the RPi wiring. Our tests on RPi 2 show that the SPI CLK line when unloaded has a resonant frequency of about 40MHz, and when loaded, the MOSI and MISO lines ring at an even lower frequency. Measurements show that SPI waveforms are very poor and unusable at 62 and 125MHz. Dont expect any speed faster than 31MHz to work reliably.

API's -->

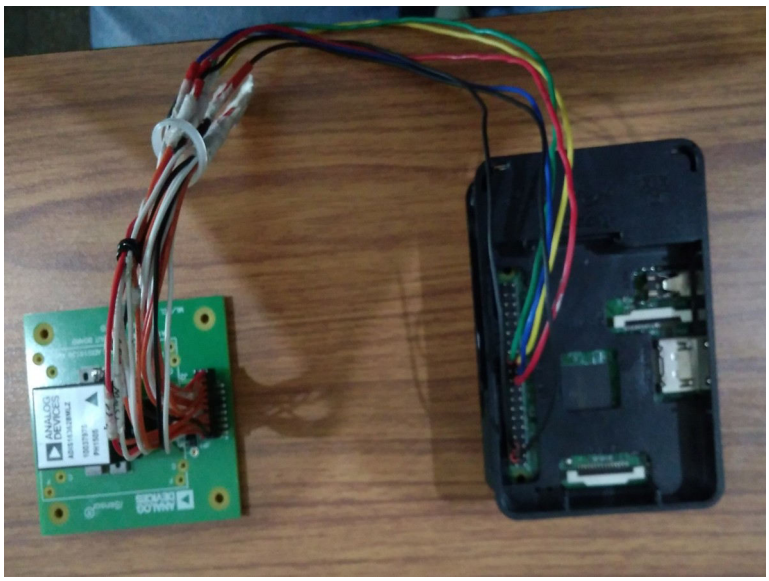
- int bcm2835_spi_begin (void)
- void bcm2835_spi_end (void)
- void bcm2835_spi_setBitOrder (uint8_t order)
- void bcm2835_spi_setClockDivider (uint16_t divider)
- void bcm2835_spi_setDataMode (uint8_t mode)
- void bcm2835_spi_chipSelect (uint8_t cs)
- void bcm2835_spi_setChipSelectPolarity (uint8_t cs, uint8_t active)
- uint8_t bcm2835_spi_transfer (uint8_t value)
- void bcm2835_spi_transfernb (char *tbuf, char *rbuf, uint32_t len)
- void bcm2835_spi_transfern (char *buf, uint32_t len)
- void bcm2835_spi_writenb (char *buf, uint32_t len)

2. INTERFACE

The Microprocessor and the sensor has a physicsl interface using jumper cables through which the communication takes place using SPI protocol.

The ADIS16362 sensor is mounted over a ADIS16IMU1/PCBZ Breakout Board for access to the sensors delicate pins.

THE Microprocessor communicates with the sensor with the help of the lines through the pins on the breakout board.

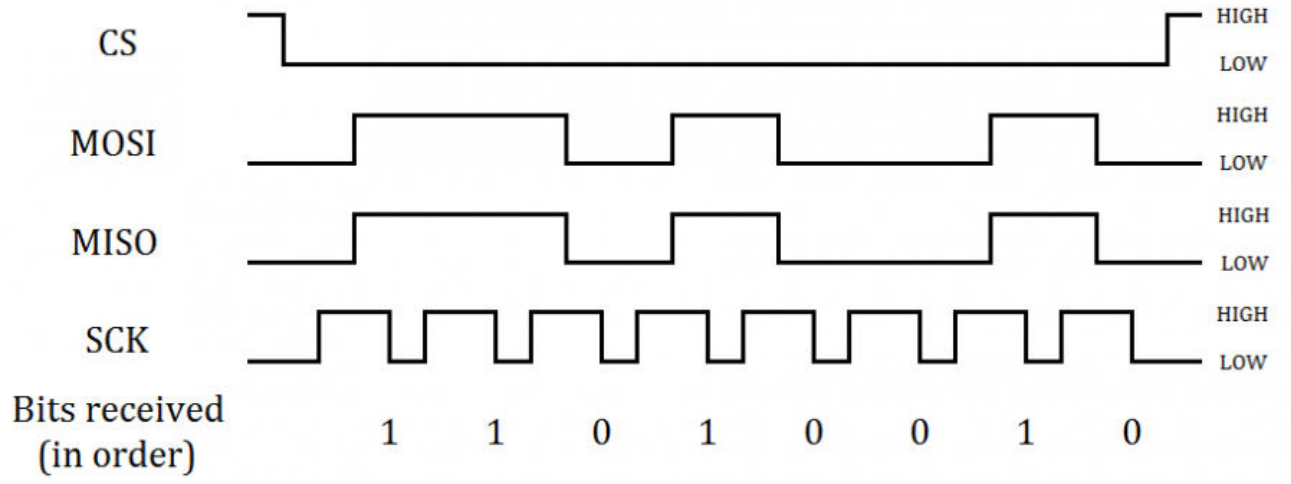


3. COMMUNICATION

The communication is written in C programming language and uses BCM2835 library for communication specific keywords termed APIs. The RaspberryPi-BCM2835 APIs are used for access to the controllers GPIO ports.

To start the communication the microcontroller has to send a ADDRESS to the sensor through the MOSI line,this address contains or points to the parameter that is required as the result.

The address is a 8-bit hex value which is unique for every axis.
After the sensor receives the address it checks for the corresponding data and sends it back to the microcontroller on the MISO line.



4. PROGRAM

```
#include <stdio.h>
#include <bcm2835.h>
#include <time.h>
#include <unistd.h>


typedef union{
        short int t2:14;
        unsigned char t1[2];
}DATA;


int main(int argc, char **argv){

    if (!bcm2835_init()){
        return 1;
    }

    DATA xdata, ydata, zdata;

    bcm2835_spi_begin();
    bcm2835_spi_setDataMode(BCM2835_SPI_MODE0);
    bcm2835_spi_setClockDivider(BCM2835_SPI_CLOCK_DIVIDER_512);
    bcm2835_spi_chipSelect(BCM2835_SPI_CS0);
    bcm2835_spi_setChipSelectPolarity(BCM2835_SPI_CS0,LOW);

    float MF = 0.000333;
    float X,Y,Z,Gx,Gy,Gz;
```

```
        int i;
        for(i=0;i<30;i++){

        unsigned char send_lx[2] = {0x0A,0x0B};
        unsigned char readx[2];
        bcm2835_spi_transfernb(send_lx,readx,sizeof(4));
        xdata.t1[0] = readx[1];
        xdata.t1[1] = readx[0];
        X = xdata.t2 * MF;
        Gx = X * 9.8;
        printf("Gx::%f m/s2\t\t",Gx);

        unsigned char send_ly[2] = {0x0C,0x0D};
        unsigned char ready[2];
        bcm2835_spi_transfernb(send_ly,ready,sizeof(4));
        ydata.t1[0] = ready[1];
        ydata.t1[1] = ready[0];
        Y = ydata.t2 * MF;
        Gy = Y * 9.8;
        printf("  Gy::%f m/s2\t\t",Gy);

        unsigned char send_lz[2] = {0x0E,0x0F};
        unsigned char readz[2];
        bcm2835_spi_transfernb(send_lz,readz,sizeof(4));
        zdata.t1[0] = readz[1];
        zdata.t1[1] = readz[0];
        Z = zdata.t2 * MF;
        Gz = Z * 9.8;
        printf("  Gz::%f m/s2\n",Gz);
        printf("------------------\t\t----------------------\t\t--------------------\n");

        bcm2835_delay(3000);
        }
bcm2835_spi_end();
bcm2835_close();
return 0;
}
```

5. RESULT

Gx::7.343473 m/s2	**Gy:: 6.758352 m/s2**	**Gz::7.564932 m/s2**
----------	**----------**	**----------**
Gx::7.846473 m/s2	**Gy::-8.456399 m/s2**	**Gz::7.956743 m/s2**
----------	**----------**	**----------**
Gx::8.638264 m/s2	**Gy::-7.937856 m/s2**	**Gz::6.776841 m/s2**
----------	**----------**	**----------**
Gx::-7.946583 m/s2	**Gy::8.594536 m/s2**	**Gz::7.649632 m/s2**
----------	**----------**	**----------**
Gx::-6.735823 m/s2	**Gy::6.327543 m/s2**	**Gz::8.453784 m/s2**
----------	**----------**	**----------**
Gx::7.740566 m/s2	**Gy::8.611835 m/s2**	**Gz::-6.493643 m/s2**
----------	**----------**	**----------**
Gx::7.649376 m/s2	**Gy::8.253499 m/s2**	**Gz::-6.493463 m/s2**
----------	**----------**	**----------**
Gx::8.437452 m/s2	**Gy::7.645294 m/s2**	**Gz::8.364822 m/s2**
----------	**----------**	**----------**
Gx::8.437452 m/s2	**Gy::7.645294 m/s2**	**Gz::8.364822 m/s2**
----------	**----------**	**----------**
Gx::8.437452 m/s2	**Gy::7.645294 m/s2**	**Gz::8.364822 m/s2**
----------	**----------**	**----------**

6. FUTURE SCOPE

In my project I have interfaced for accelerometer only. But the ADIS16361 sensor consists a gyroscope and a temperature sensor also which are similarly communicated through SPI protocol.

The microprocessor can also handle some more sensors at the same time, because of its high fanout capacity.

Wireless methords can be used to receive the data form a remort location.

The setup can be used as a gadget for measuring the parameters and desplay on a screen fixed on the microprocessor.

7. REFERENCES

1. http://www.analog.com/media/en/technical-documentation/data-sheets/ADIS16362.pdf

2. https://cdn-learn.adafruit.com/downloads/pdf/introducing-the-raspberry-pi-2-model-b.pdf

3. http://www.airspayce.com/mikem/bcm2835/group__spi.html#ga6f0330a183f3c5765a36f1839e029a44