Efforts Towards the Detection of Perturbations in the Earth's Magnetic Field: Design and Development of Solid State Sensor Array

by

Chandra Shekar Reddy Yerrabothu

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 6, 2011

Abstract

The following thesis provides a detailed description of the design and development of magneto-resistive unexploded ordinance (UXO) sensor array to detect the perturbations in the Earth’s magnetic field due to the presence of buried ferrous objects. Over the years there has been a substantial development in the detection of unexploded ordinance. The objective of this research is to use the multiple axis sensors in the design to detect UXO. This thesis describes the theory of different magnetization techniques and the characteristics of Anisotropic Magneto-Resistive (AMR) sensors. The description of each hardware component employed in the system is given in detail. The softwares used for the data handling and processing is also addressed. The verification of a working design is achieved by acquiring baseline data set.

Acknowledgements

I would like to convey my most sincere thanks to Dr. Lloyd Stephen Riggs for his guidance and support in writing of this thesis. For his encouragement and motivation I will forever be grateful. I would also like to thank my committee members, Dr. Stuart M. Wentworth and Dr. Michael Baginski for their valuable suggestions. Thanks are due for Dr. Don-Terry Veal, Patrick Rose, and Sheree Wilson from Center for Governmental Services for their continuous financial support. Finally, a very special thanks to all my family and friends for supporting me every step along the way.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The objective of this research is to design, develop, and test a simple, user friendly, cost efficient unexploded ordinance (UXO) sensor. There are several techniques that are developed for the detection of unexploded ordinance. This thesis will address the magnetization techniques and key topics related to anisotropic magneto-resistive (AMR) sensors. The design and working of the hardware components used in the system shall be addressed. This design uses an array of magneto-resistive (MR) sensors for the detecting a ferrous object.

The sensor used in the design is the HMC2003 from Honeywell. The objective of the system is to collect magnetic field measurements through the MR sensors by detecting perturbations in the earth's magnetic field due to buried ferrous materials. This collected data will be transmitted to a computer and a graphical user interface will be used to display the collected data. The design of an unexploded ordinance sensor mainly consists of four parts: magneto-resistive sensor, computer architecture, wireless transmission, and graphical user interface (GUI). The components in the computer architecture are the microcontroller, multiplexers, and analog to digital converter (ADC). The microcontroller handles all on-board sensor operations. The micro controller processes MR sensor measurements, analog to digital conversion of measurements, data packaging and processing of serial wireless communication through ZigBee. The microcontroller used in the design is the Arduino Fio board and the ADCs implemented in the design are the ADS1115 from Texas Instruments.

The number of sensors that can be supported by the microcontroller is derived from three differential inputs available on four external ADCs, each of which can be multiplexed to 16 analog signals. This yields 192 available analog inputs. Each MR sensor has an input for each of axis (X-, Y-, and Z), yielding a maximum of 64 sensors. The sensor data is transferred through ZigBee to the computer wirelessly and it will be displayed on the GUI.

The communication between the computer and the micro-controller are done using a wireless network. The wireless network consists of ZigBee digital radios. ZigBee is a low power and low cost high level communication protocol. It is based on IEEE 802.15.4 standard. The Arduino Fio microcontrollers have a specially designed interface for ZigBee. The data is gathered and graphically represented using Math Work's Matlab and Python software. The software is used to program the serial port communication handling, data collection and to achieve graphical visualization.

Design verification is achieved by conducting a baseline data set acquisition process. Future design improvements are described in detail.

Chapter 2: Theory

Magnetic sensors have been serving mankind in numerous applications through decades. The ranges of magnetic sensing applications include magnetic storage discs to applications in airplanes. There are several ways of sensing magnetic fields based on connections between electric and magnetic fields. Most common magnetic sensing technologies are listed in the following table [4].

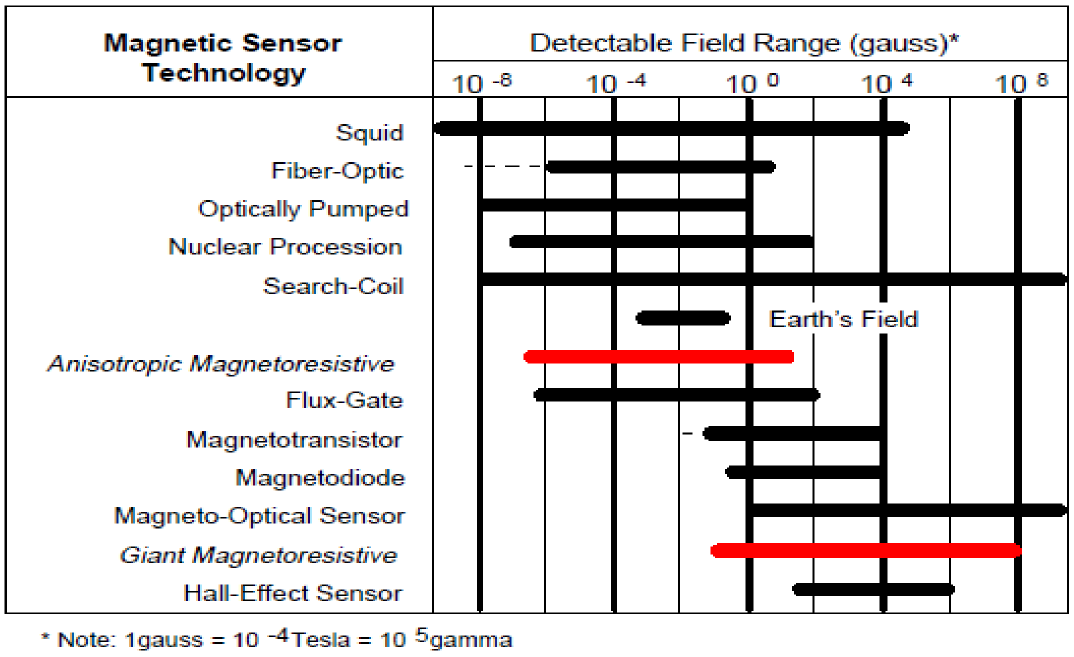


Table 2.1 Field Ranges of different magnetic sensor technologies

2.1 Different Magnetic sensing technologies

Low Field Sensors

Low field sensors are used in medical and military applications. Some of the low field sensors are search coil, nuclear precession, optically pumped, and fiber optics magneto meter. They are usually bulky and costly compared to other magnetic field sensors.

Search Coil Magnetometer

Search coil magnetometer technology is based on Faraday's law of induction. If the magnetic flux through a coil changes a voltage will be generated proportional to the rate of change of flux through the coil [4]. The sensitivity of the search coil depends on the permeability of the core as well as the area and the number of turns of the coil. The disadvantage of the search coil is that it only works if the magnetic field is changing with time. It cannot be used detect static or slowly varying magnetic fields. The sensitivity range of the search coil is from a field as low as 10^{-6} G to a field without an upper limit. These sensors are very easy to manufacture, are very low cost, and are widely used in road traffic signaling system.

SQUID Magnetometer

SQUID stands for Superconductive Quantum Interference Device. It is highly sensitive and it can measure extremely low magnetic fields. Its sensitivity ranges from several fempto-Teslas to 9 Tesla. Due to the high sensitivity SQUID magnetometers are frequently used in medical applications.

Earth’s Field Sensors

Earth’s field sensors are used in compass navigation and vehicle detection etc. These include the fluxgate magnetometer, magneto inductive magnetometer, and anisotropic magneto resistive technology.

Flux Gate Magnetometer

Flux gate magnetometer consists of small, magnetically susceptible and high permeability ferromagnetic core wrapped by two coils of wire; called the primary and secondary winding. When current is passed through primary (drive) coil, it drives the core to oscillate between two magnetic saturation points. This changing field will induce current in the secondary (sense) coil which is measured by a detector. In magnetically neutral field the input and output currents will match. In an alternating magnetic field the input and out currents will be different. This difference depends on the strength of applied magnetic field [4]. The sensitivity of this magnetometer ranges from 10^{-6}G to 100G. Flux gate magnetometers are widely used in compass navigation systems and were developed during World War II to detect submarines. The flux gate magnetometer circuit is shown in the figure 2.1 [4].

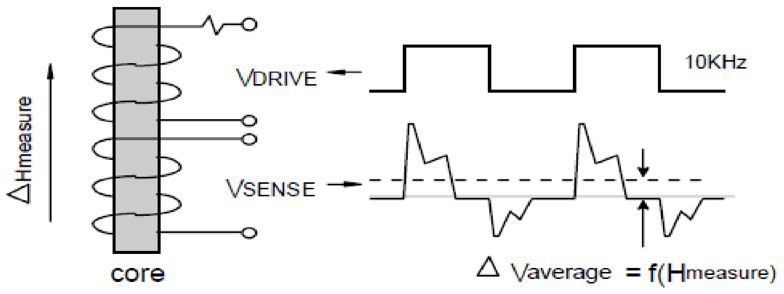


Figure 2.1 Flux Gate Magnetometer circuit

Magnetoinductive Magnetometer

A magnetoinductive magnetometer has a sense coil that has a ferromagnetic winding which changes permeability in the Earth's magnetic field. The sense coil is an inductance element in L/R relaxation oscillator. The oscillator frequency is proportional to field being measured. When a static dc current is applied, the frequency will shift as the sensor is rotated 90^0 from the applied magnetic field. The frequency shift can be recorded using a microprocessor capture port. These magnetometers, shown in the figure 2,2 [4],are simple, use little power, and are low cost.

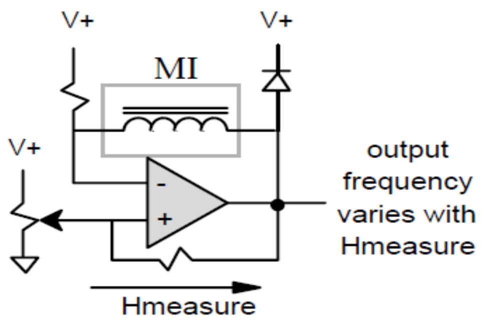


Figure 2.2 Magneto Inductive Magnetometer Circuit

Anisotropic Magnetoresistive Sensors (AMR)

AMR sensors are based on the Magnetoresistive effect. There is a change in resistance of an iron when its magnetization is altered. When a current is passed through a film of Permalloy, the magnetic orientation of the material will be in the direction parallel to the current. If a magnetic field is applied perpendicular to the direction of current, the magnetization will rotate towards the direction of magnetic field. The resistance of the film decreases as the magnetization rotates away from the current flow. AMR sensors have a sensitivity range between

10^{-6} G and 50G. These sensors are widely used in vehicle detection, for automotive wheel speed measurement, current sensing and many other applications.

Giant Magnetoresistive Sensor (GMR)

GMR Sensors consist of a thin anti-ferromagnetic layer sandwiched between two ferromagnetic layers. When an external magnetic field is applied, the mean distance the electrons have to travel through the structure changes causing the resistance to change. A change in the resistance up to 70% is observed when large magnetic fields are applied. In AMR, the change in the resistance is only a few percent, hence the name "Giant Magneto Resistors (GMR)" is given.

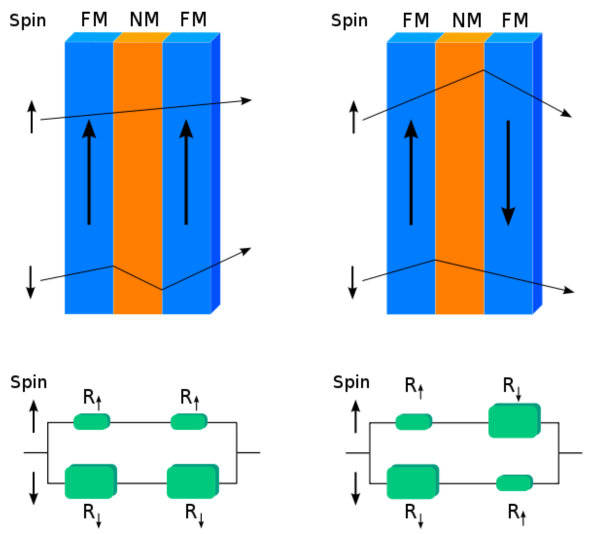


Figure 2.3 GMR Working Principle

Both GMR and AMR are ideal for array sensing as multiple sensors can be fabricated on the same IC. The sensing technique implemented in this design is the anisotropic Magnetoresistive (AMR) technique.

2.2 Theory of Anisotropic Magneto Resistive Sensors

In 1857, William Thompson discovered the basic anisotropic magneto resistive effect. He discovered a change in the electrical resistance of an iron bar when its magnetization was altered. This phenomenon is called “Anisotropic Magnetic Effect”. AMR effect can be explained with reference to figure 2.4.

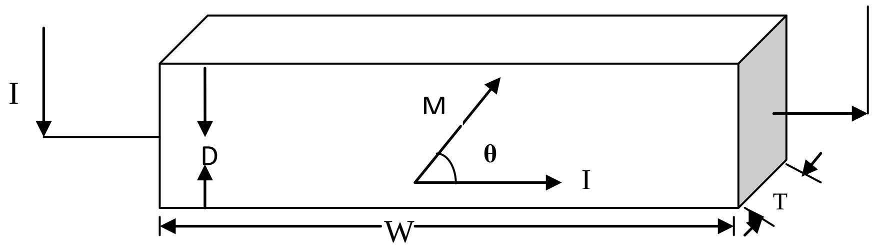


Figure 2.4 Principle of Magneto-Resistive Effect

In a thin magnetic film, the angle between magnetization M and sensing current I is θ, then the electrical resistance can be defined as,

$$R = R_0 + \Delta R \, COS^2 \theta$$

In the above equation R_0 is the fixed part and ΔR is the maximum value of variable resistance. AMR effect takes place in ferrous materials. The transducer of magneto-resistive elements is in the form of a Wheatstone bridge. Resistance of all four resistor elements in the bridge is R, and supply voltage is V_b. The supply voltage causes a current to flow in the resistors. Due to the applied field H, the magnetization in two opposite resistors rotates in the direction of current and in the other two resistors it rotates away from the direction of current. An increase in the resistance (R) of the two resistors with magnetization in the direction of current is observed. The other two resistors have a decrease in the resistance (R). The output will be proportional to applied field H, $\Delta V = H\ S\ V_b$. AMR sensors undergo a resistance change of 2-3% in the presence of magnetic field. The figure 2.7 [4] shows the AMR circuit.

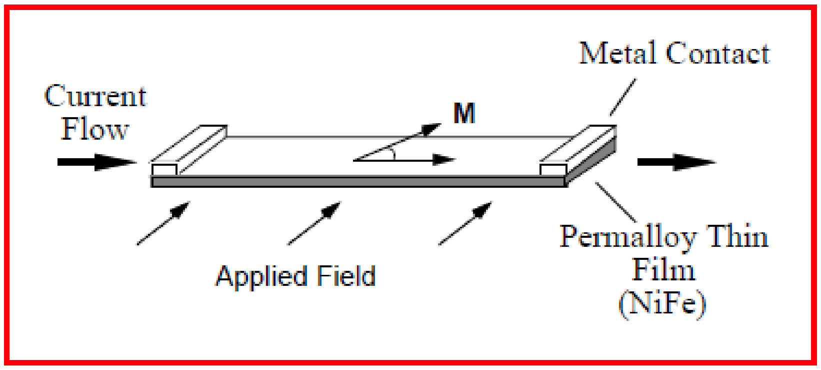


Figure 2.5 Operation of MR sensors

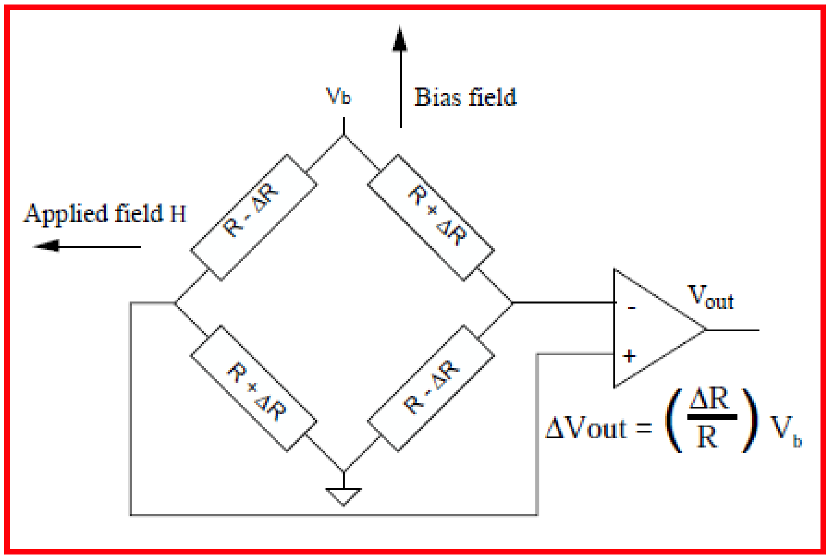


Figure 2.6 Transducer of MR elements

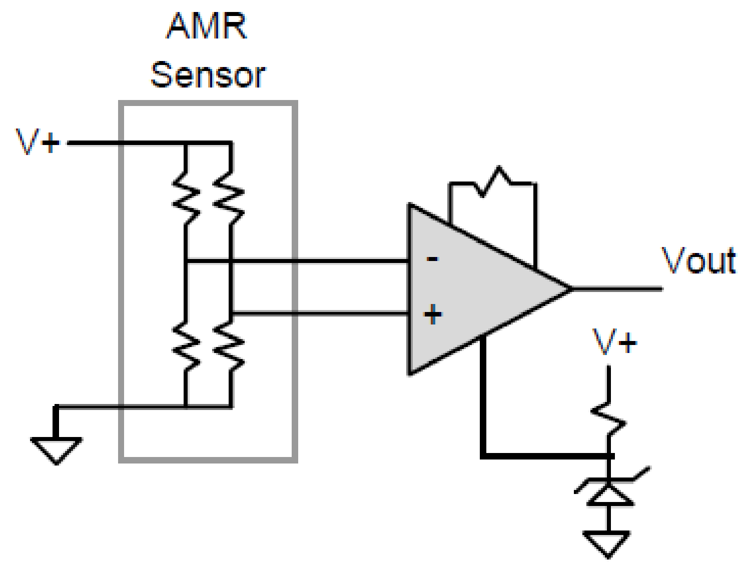


Figure 2.7 AMR Sensor circuit

AMR sensors are made of a nickel-iron (Permalloy) thin film deposited on silicon wafer in a resistor bridge pattern. The deposition of Permalloy film on the wafer is done in a strong magnetic field. This strong magnetic field sets the orientation of magnetization vector in the direction preferred. The magnetization vector is parallel to the length of the resistor and can be set to either left or right in the film. The magnetic range of AMR sensors lies in the range of the Earth's magnetic field. AMR sensors can measure both linear and angular position displacement in the Earth's magnetic field.

2.3 Issues in AMR sensors

The main issue with AMR sensors is the Bridge offset voltage. AMR sensors are designed in Wheatstone bridge configurations with four identical resistors. A typical AMR Wheatstone bridge as shown in the figure 2.8 [10] will have two null points. Ideally the output taken at these null points should be same. If V_b is the supply voltage and V_{0+} , V_{0-} are the output voltages at the null points, V_{0+} and V_{0-} should be equal. Due to the in-exact resistor values, these voltages will differ in values. The difference between V_{0+} and V_{0-} is known as "Bridge offset voltage". The bridge offset voltage is an undesired output and it has significant impact on the sensor output. A small difference in the value of resistor elements will cause a sizable bridge offset voltage.

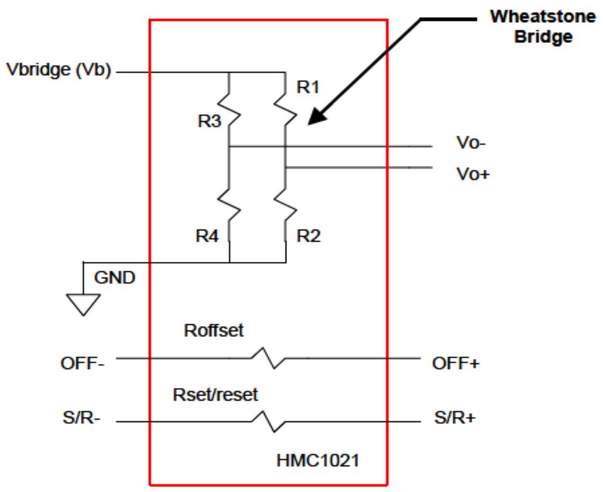


Figure 2.8 Wheatstone bridge of typical AMR sensor

There are several methods to eliminate bridge offset voltage in the AMR sensors. Some of them are the shunt resistance method, amplifier biasing method, electronic feedback method, and offset strap current method. In shunt resistance method, a resistor is added to one or more legs of the bridge. This resistance will force the output at the null points to be identical. Addition of a resistor should be done in zero magnetic field environments so that it does not get affected by the Earth' magnetic field. Usually it is done in a closed shielded chamber. The drawback of this method is the labor involved in finding the exact value of shut resistance for each resistor in the bridge. The figure 2.9 [10] shows the shunt resistance circuitry.

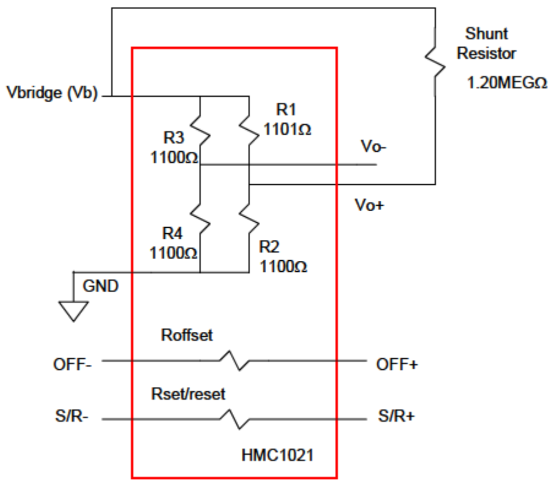


Figure 2.9 Shunt Resistance Method

In amplifier bias nulling method, a bias voltage of half of the supply voltage minus offset voltage is also applied. This bias voltage will create a current in opposite direction that nulls the offset. This method can be achieved by adding an instrumentation amplifier to the bridge as shown in the figure 2.10 [10].

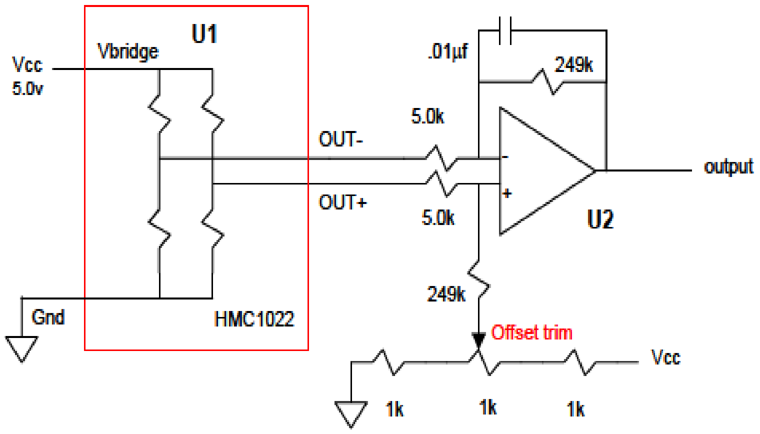


Figure 2.10 Amplifier Bias nulling method

In set/reset with microprocessor method, bridge offset is removed by using subtraction. First a "set" pulse is activated and the reading is taken as V_{set} with an applied field $H_{applied}$. Next a "reset" pulse is activated and the reading is taken as V_{reset}. The subtraction of V_{set} and V_{reset} gives the doubled value of desired field.

$$V_{set} = S^* H_{applied} + V_{offset} \quad \& \quad V_{reset} = -S^* H_{applied} + V_{offset}$$

$$V_{set} - V_{reset} = 2^* S^* H_{applied}$$

The result of the subtraction is that the bridge offset voltage is removed and this method also removes temperature drift of the bridge offset voltage.

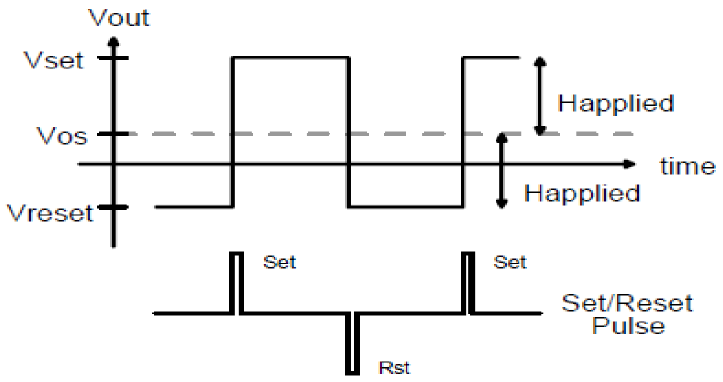


Figure 2.11 Set/Reset Effect on bridge Output

In the Electronic feedback method, a feedback signal is generated to null the bridge offset voltage. The following figure 2.12 [4] shows the schematic of the electronic feedback circuitry. In this method, the output of the bridge is made to switch between V_{set} and V_{reset} by applying set and reset pulses. This process modulates the output signal to a higher band. The amplifier#1 outputs a negative dc signal compared to the bridge offset and this signal is fed back to the input of amplifier#2. This process will eliminate the offset voltage. In the third stage the output which is free from bridge offset is demodulated.

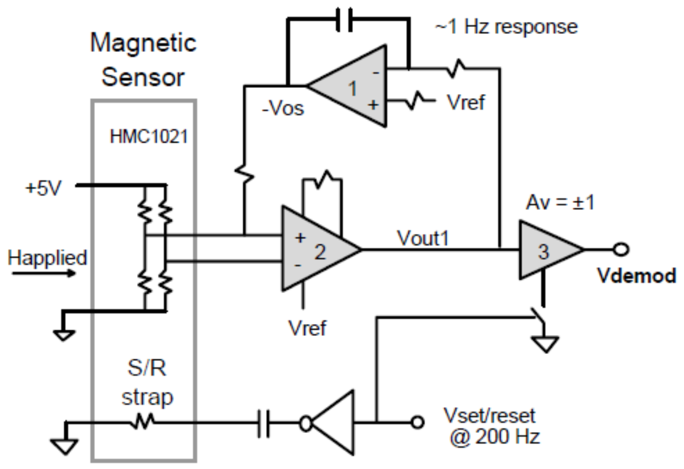


Figure 2.12 Electronic Feedback Circuitry

In the Offset strap method, a coil is placed near the bridge. When a current is passed through this coli, a field which is equal to the bridge offset voltage is created and it will nullify the bridge offset. Any of the above methods can be used to solve the issue bridge offset voltage in the AMR sensors.

Chapter 3: Hardware and Wireless Network Components

3.1 Sensor

The sensor used in the design is Honeywell's HMC2003. It is a three-axis hybrid magnetic sensor with high sensitivity used to measure low magnetic fields. It is a combination of Honeywell's most sensitive sensors HMC1001 and HMC1002. HMC2003 provides maximum user flexibility due to its analog interface with critical nodes available for pin interface. It has signal conditioning circuitry with operational amplifiers, and low-pass filters which eliminate the unwanted noise in the outputs. HMC2003 features include internal excitation current source and offset resistors which reduce the temperature errors and offset drift. Because of its signal conditioning circuitry, HMC2003 is chosen over other AMR sensors sold by other manufacturers.

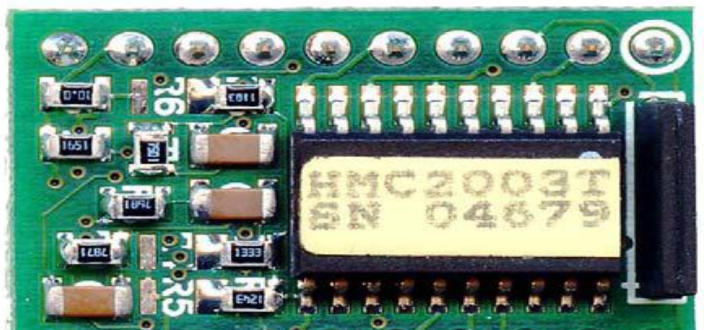


Figure 3.1 Honeywell's HMC2003

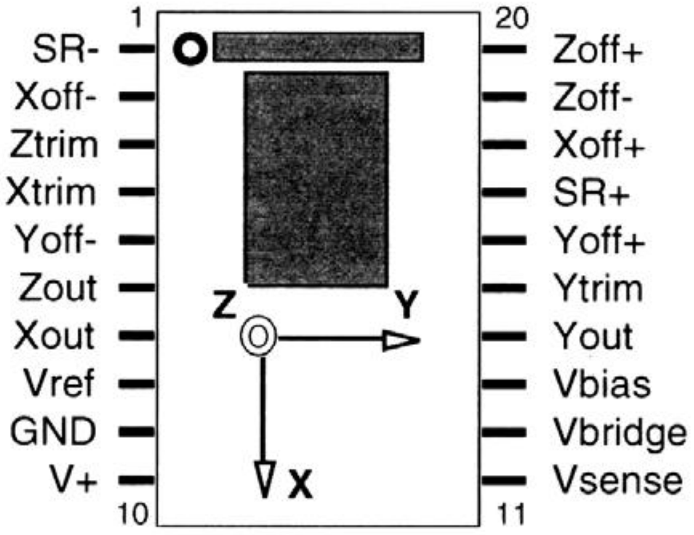


Figure 3.2 Pin Diagram of HMC2003

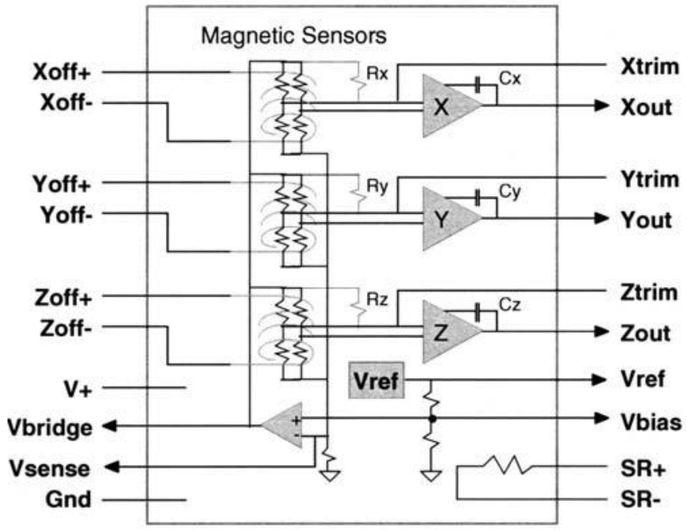


Figure 3.3 Block Diagram of HMC2003

3.1.1 HMC2003 Features

HMC2003 comes from Honeywell's HMR/HMC series. HMC2003 is a three axis magnetic sensor hybrid. It measures strength and direction of magnetic field using three Permalloy magneto-resistive sensors. These sensors can measure the field along the length, width, height (X, Y, Z axis). Analog output interface is provided for each X, Y, Z axis from the hybrid. HMC2003 has a reference voltage of 2.5 Volt, with a power supply range of 6-15 Volts. HMC20003 is configured in a 20 pin layout. Its dynamic range is from less than 40 micro gauss to +/-2 gauss. It is integrated with magnetically coupled straps to eliminate unwanted magnetic fields. The HMC2003 sensor output can be affected by high momentary magnetic fields. To eliminate this effect, a magnetic switching technique is applied to the bridge using Set/Reset pins.

HMC2003 sensors are easy-to-use, highly reliable, and provide several desirable features. They have high sensitivity. Due to high sensitivity, even at very low field ranges, the sensors provide highly sensitive output. These sensors can measure the changes in the field accurately even if the target is well away from the sensor. They provide very low noise compared to other technologies. The provision of the offset strap on chip eliminates the need for trimming resistor. They can easily be mounted on a printed circuit board (PCB). These sensors are provided with feedback electronics to minimize temperature effects.

HMC2003 sensors can be used to measure the presence, magnitude, and direction of a magnetic field. They can sense changes in the magnetic field in the presence of ferromagnetic objects. They can also be used to measure the earth's field for navigation and compassing purposes. The HMC2003 have a wide spectrum of applications. HMC2003 are widely used in precision compassing, navigation systems, traffic detection, proximity detection, medical devices, and attitude reference. Their applications also include mineral prospecting, food processing, agricultural equipment, and underground boring equipment.

3.1.2 Set/Reset Driver

A Set/Reset driver is designed to eliminate the effect of latch on the output. The circuit operates in a push-pull output stage (totem pole stage) method. One end of the set/reset strap is grounded and the other end is driven by push-pull stage as shown in the figure 3.4. The Set/Reset circuitry features two CMOS IRF 7509 IC chips. These chips are driven by an active low PWM signal of 2ms period with 1 percent duty. In steady state, the capacitors work as open circuits. When the driving PWM signal is applied, the capacitors will get charged and discharged as per the transition of the driven signal. This will form transient currents. The original set/reset driver proved not to provide enough current to set/reset the MR sensor. Hence the two totem pole circuits in push/pull method working complementary to each other were implemented. This technique produced double the amount of output current. The output measured across strap resistance is shown in the figure 3.6.

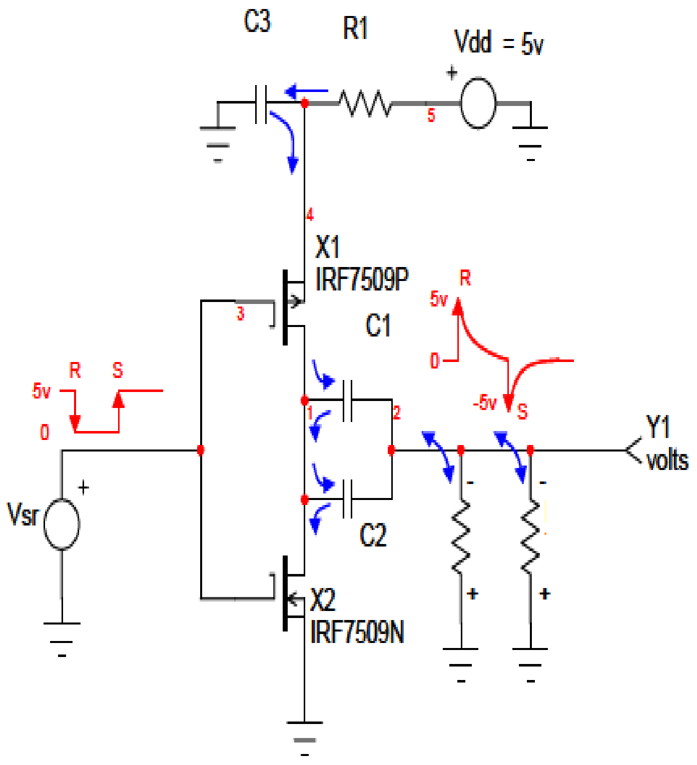


Figure 3.4 Totem pole Circuit

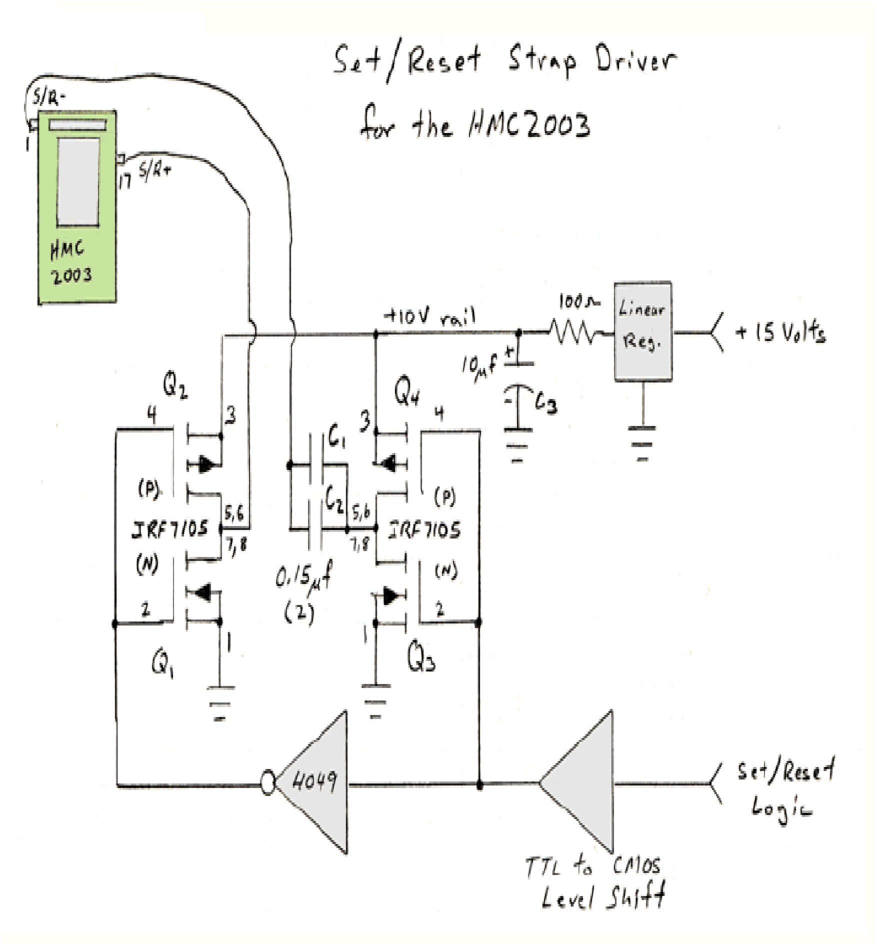


Figure 3.5 Set/Reset Circuitry

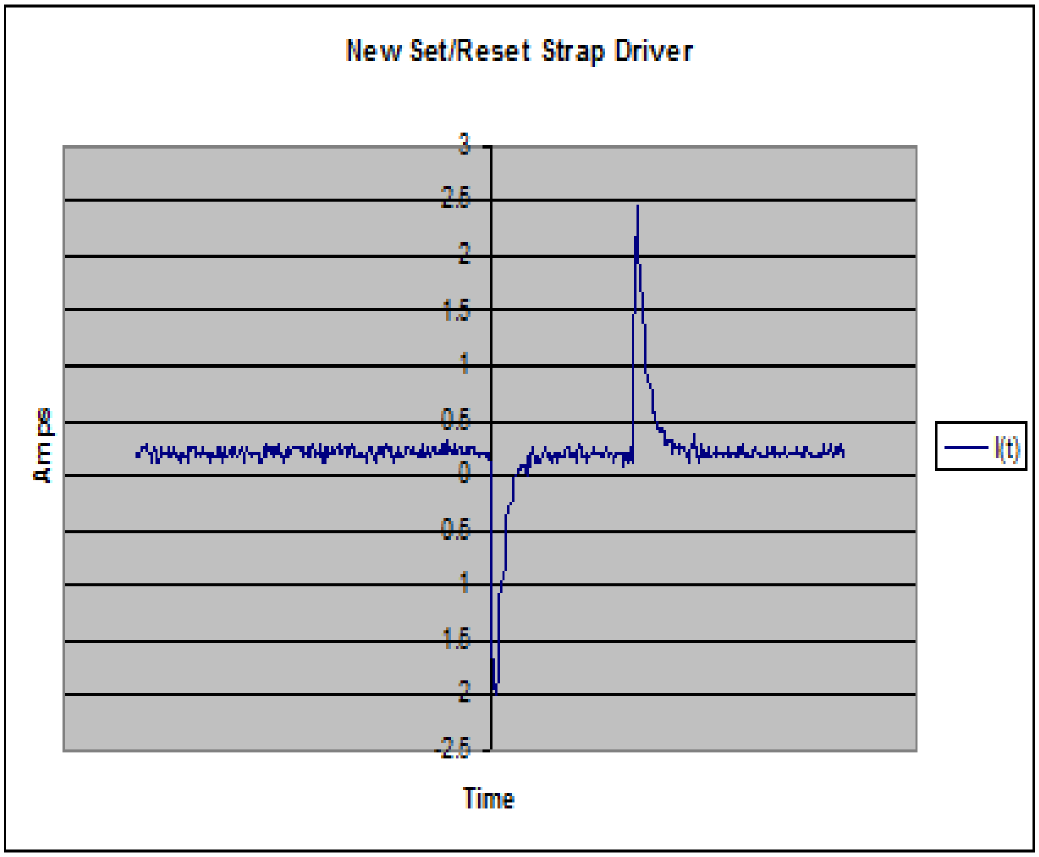


Figure 3.6 Output of Set/Reset circuitry

From the output in figure 3.6, it is clear that the output current is not sufficient to set/reset the MR sensor. The required current to set/reset the MR sensor is 3.2 Amperes. One solution was, by increasing the power supply of the push/pull strap driver, the amount of current supplied by the circuit can be increased. But the output becomes unbalanced as the reset stage of current is greater than that of the set stage. By altering the capacitor values used in the circuit, more charge to can be stored and dumped by the driver.

3.2 Microcontroller

The microcontroller part in the design is very complex as the design involves several dozen sensors. In the design the microcontroller is used for data storage and control logic, and a set of external analog to digital convertors is used for signal digitization, and a set of analog signal multiplexers for input expansion.

In the design the micro-controller collects data and communicates with the PC wirelessly using Digital ZigBee radios. The selection of micro-controller is based on several design requirements. The micro-controller should have a built in Inter-Integrated-Circuit Bus (I2C) to communicate with external ADC devices. Pulse Width Modulated Output is required to implement a set/reset circuit. At least 4 Digital Input/output (DIO) pins are required to operate the multiplexer. It should have a serial port for communication to a computer using FTDI cable or ZigBee. The ZigBee radios use the same physical interface as FTDI, which can be done in software. To make it a cost efficient design the micro-controller with open source programming environment is a factor. The micro-controller data storage capacity is a very important factor in the selection of a micro-controller. It should have a memory space to hold sensor readings, flash memory to hold code-base, RAM for temporary storage and simple calculations. Three most suitable micro-controller boards are Atmel (Arduino), PIC (mini-Bully), and Freescale (Adapt9S12). The Mini-Bully, shown in the figure 3.8, is a PIC series microcontroller communicates with computers through a FTDI cable. Its hardware features are adequate for the design, but it has a limited software support. Freescale (Adapt9S12) board, shown in the figure 3.7, meets all the EEPROM/Flash/RAM, I/O requirements, but its development environment does not support 64 bit Windows. In addition, its cost is higher than the other two options.

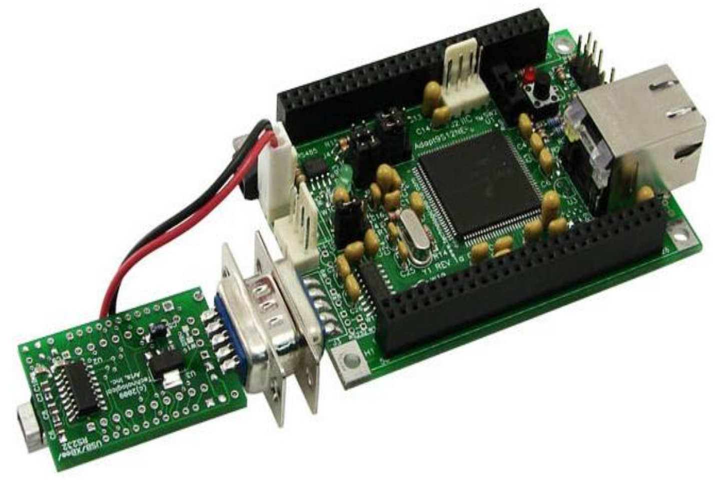


Figure 3.7 Adapt9S12

Figure 3.8 Mini-Bully

Arduino microcontroller is the option chosen which covers all requirements. It uses the Arduino platform with programming environment as simple as C. This programming language is based on wiring. Wiring is an open source programming environment. The Arduino Fio features also include predefined libraries with one line commands. This feature makes the programming of the microcontroller very simple. The Arduino Fio board is developed for wireless applications. It communicates using ZigBee radios. It also features built-in sockets for a ZigBee radio.

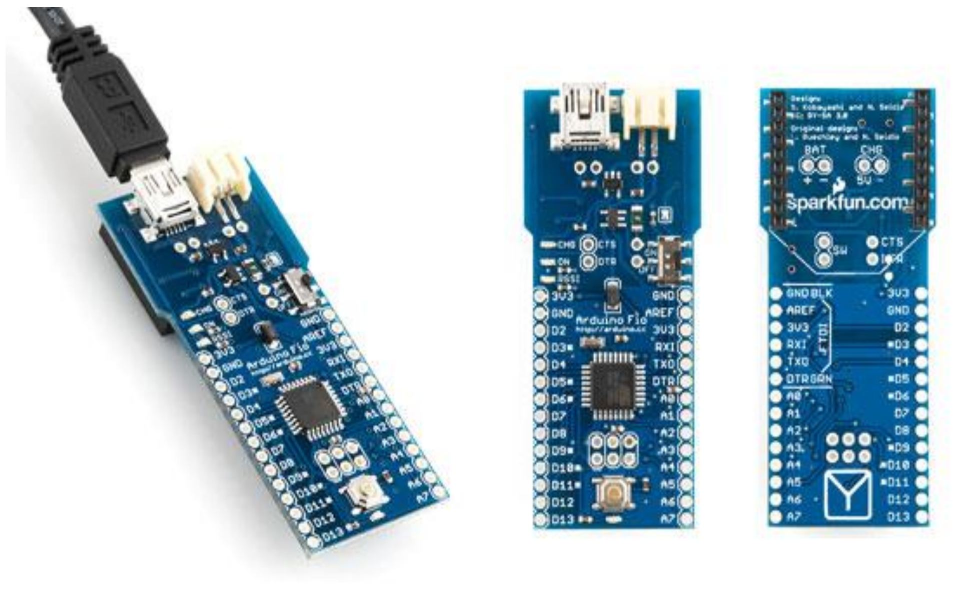


Figure 3.9 Arduino Fio

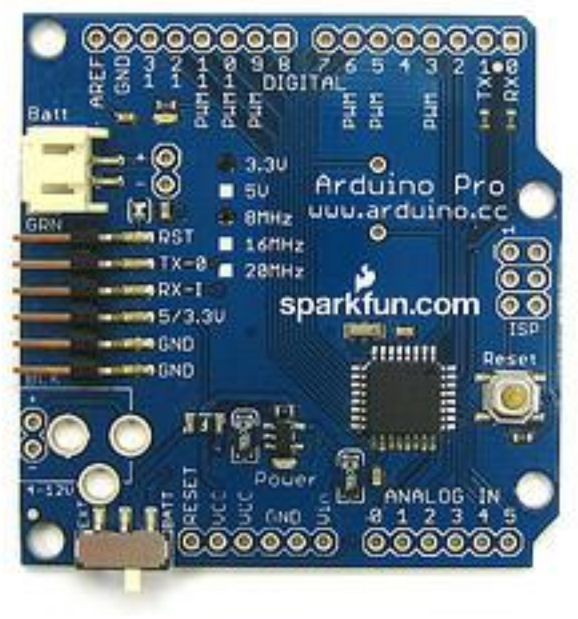


Figure 3.10 Arduino Fio board

3.2.1 Arduino Fio Features

Arduino Fio microcontroller board was designed by Shigeru Kobayashi and manufactured by sparkfun electronics. It is developed based on ATmega328P microcontroller. It can be powered with 3.3V regulated power supply. It is compatible with a lithium polymer battery. Its power pins are, BAT for lithium polymer battery, 3V3 pin is for regulated 3.3V power supply and GND ground pins. The Arduino Fio has a clock speed of 8 MHz. It has flash memory of 32 Kb, 2kb of SRAM, and 1Kb of EEPROM. Arduino Fio has 14 digital pins which can be used as both input and output pins. Its serial pins are RX1 and TX0. RX1 is used to receive and TX0 is used to transmit TTL serial data. These pins can be connected to ZigBee using DIN and DOUT pins of the ZigBee socket. Pins 2 and 3 are used to configure and trigger external interrupts. 3, 5, 6, 9, 10, and 11 pins are used as pulse width modulation (PWM) outputs. The figure 3.11 shows the mapping between the Arduino Fio and ATmega328P/ATmega168P.

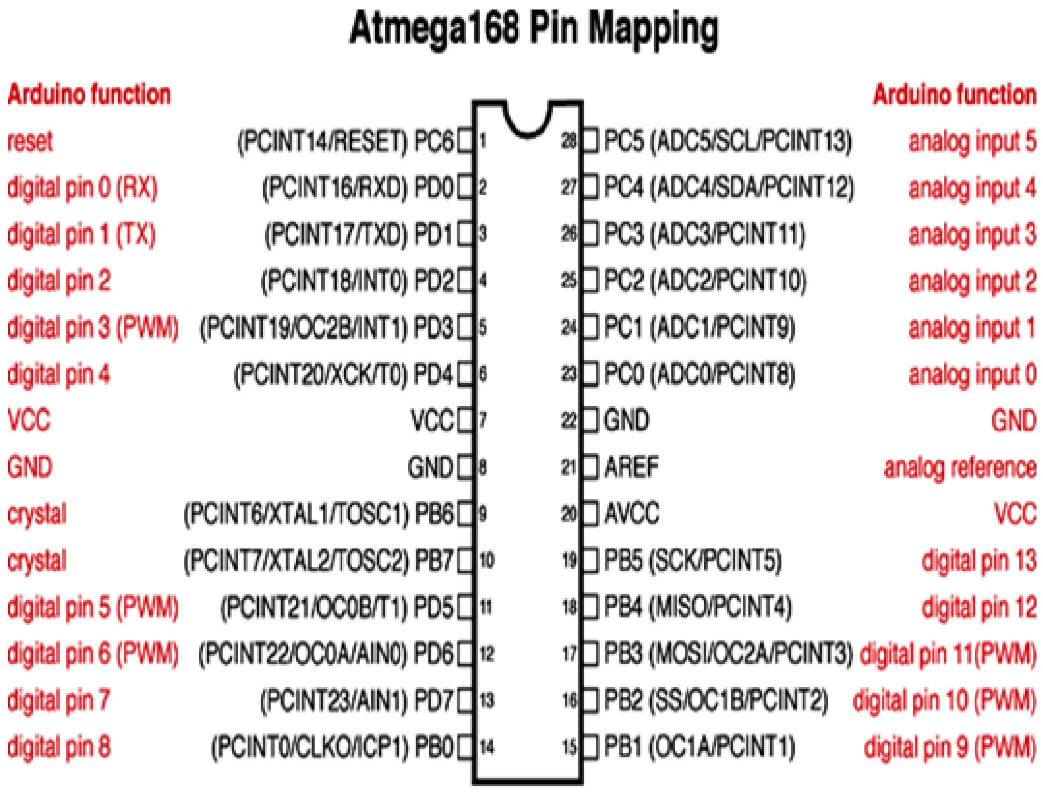


Figure 3.11 Pin Mapping of the Arduino Fio with ATmega168/328P

The 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) pins are provided to support SPI communication. Arduino Fio has an in-built LED connected to pin 13. When this pin is HIGH, LED is on and when it is LOW, LED is off. Pin 4 (SDA) and Pin 5 (SCL) are provided to support I2C communication. Additionally the Fio has 8 pins for analog input. AREF pin is provided for reference voltage for these analog inputs. The DTR pin is used to add a reset button. Arduino Fio has provisions to communicate with a computer or another Arduino or other micro-controller. Its RX1 and TX0 pins are used for TTL communication. Arduino software features a serial monitor to transmit textual data to and from the Fio board. The Fio board also has a mini-USB connector for charging, it is not used for serial communication. The physical dimensions of the PCB are approximately 1.1" x 2.6".

3.2.2 Programming the Arduino Fio

Arduino Fio can be programmed using Arduino software. The Fio comes with a bootloader, which is preburned. It follows STK500 protocol for communication. The uploading of new code can be done wirelessly or using connector cables. The FTDI USB-to-serial or USB-to-serial adaptor board can be used. The figure 3.12 shows the FTDI cable adaptor. The FTDI cable adaptor can be connected to program the Arduino Fio as shown in the figure 3.13. It can be programmed wirelessly to talk through ZigBee radios. The Fio board does not come with a built-in serial adaptor.

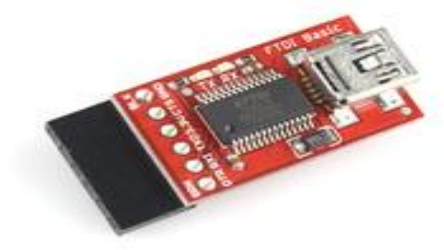


Figure 3.12 FTDI cable adaptor from Sparkfun Electronics

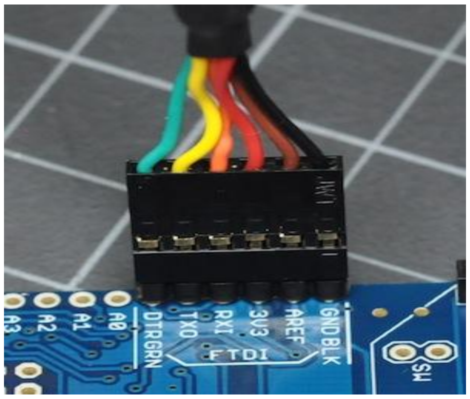


Figure 3.13 Connection of FTDI cable to program Fio

3.3 External Analog to Digital Convertor

Arduino Fio has a built-in ADC with a range of 0 to 5 volts and 10 bit accuracy. It has a resolution of 4.9 mV. The MR sensor, HMC2003, has a resolution of 40 micro-Volts. HMC2003 can detect very small changes in the earth's magnetic field due to buried unexploded ordinance. To take advantage of HMC2003's high resolution, it requires a 16-bit ADC. An external 16-bit ADC can give up to 75 micro-Volts resolution over a 5 Volt range. With an external ADC, differential readings relative to non-zero voltages with different dynamic ranges can also be taken.

The external ADC used in the design is the ADS1115 available from Texas Instruments. It communicates via I2C communication interface. The ADS1115 has 4 input pins. One of those input pins is used as a reference for differential readings.

3.3.1 Overview of ADS1115

The ADS1115 is a 16-bit, small, low power analog to digital convertor. It has a wide spectrum of applications and is very easy to configure. The convertor has an A/D core with adjustable gain and communicates using I2C serial interface. It has an internal voltage reference and a clock oscillator thus reducing the need for an external circuitry yielding improved performance.

The A/D core reads the difference of input signals at AIN_P and AIN_N. The measured differential signal is compared to an internal reference voltage. The ADC has a multiplexer which gives two differential inputs. As shown in the figure 3.14, the A/D core has a differential, switched capacitor modulator and a digital filter. The digital filter converts high speed bit stream into a code. The ADS1115 has two conversion modes, single shot mode and continuous conversion mode. Single shot mode is a power saving mode in which conversion is done by ADC upon request. The result will be stored in an internal register. After conversion, the ADC moves into auto shut down mode or idle mode. The ADC operating in this mode greatly reduces power consumption. In continuous conversion mode, data conversion starts once the previous conversion is completed. In this mode, the resultant data is the most recent completed conversion and the conversion rate is equal to the programmed data rate. The ADS1115 can convert analog data into digital data at rates up to 860 samples per second. In continuous conversion mode, the ADC consumes 150 micro-AMP of current. The ADS1115 is widely used in portable Instrumentation. Its applications also include temperature measurement and battery monitoring. It is also used in consumer goods and factory automation and process controls.

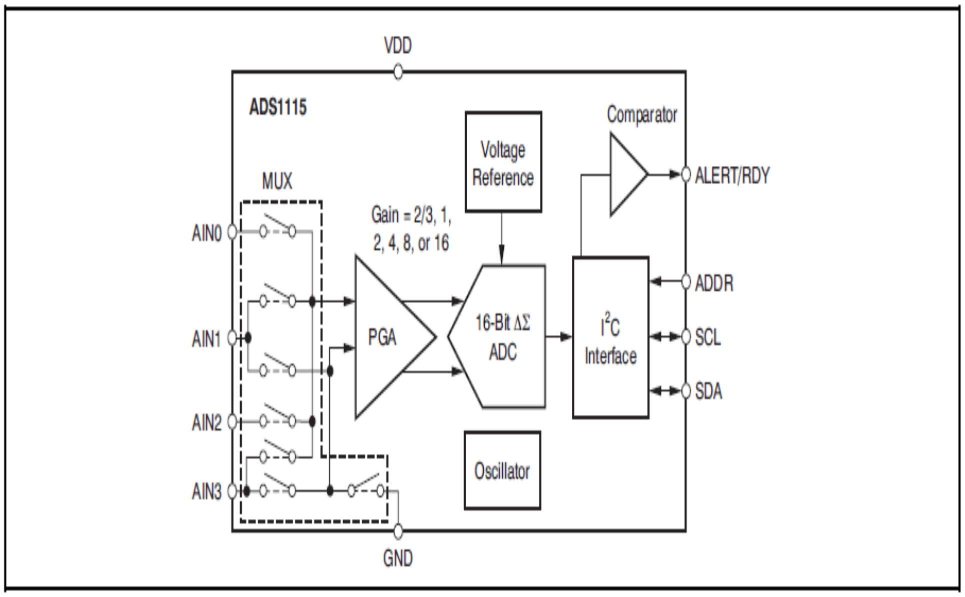


Figure 3.14 Functional Block Diagram of ADS1115

3.3.2 Features of ADS1115

ADS1115 is 16-bit analog to digital convertor with very low current consumption. It operates in two modes. In continuous mode, it consumes 150 µA of current. In single shot mode, power automatically goes down after a conversion. Figure 3.18 shows the basic hardware configuration of the ADS1115. The ADS1115 has a wide supply range from 2V to 5.5V. Its programmable data rate is from 8 samples per second to 860 samples per second. It features an internal oscillator. To measure both small and large signals with high resolution, The ADS1115 has an on-board PGA. It also features a programmable comparator. The ADC1115 communicates with the master (micro controller) via I2C Interface. It can act as slave device only. The figure 3.15 shows the pin configuration of the ADS115.The address pin (ADDR) sets the I2C address. ALERT/RDY pin is an output pin for digital comparator output. SDA pin is a digital input output pin. It is used to transmit and receive serial data. SCL is digital input pin, it clocks the data on SDA. VDD is the power supply pin, and AIN0, AIN1, AIN2, AIN3 are analog input pins.

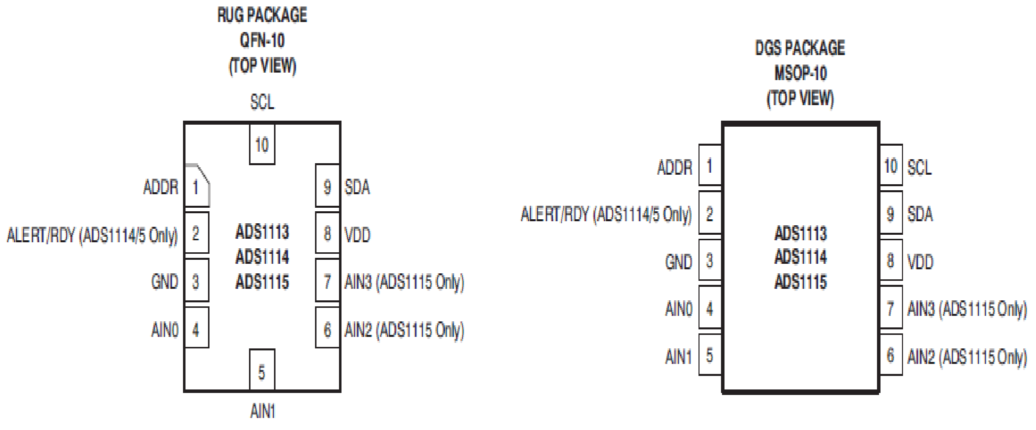


Figure 3.15 Pin Diagram of the ADS1115

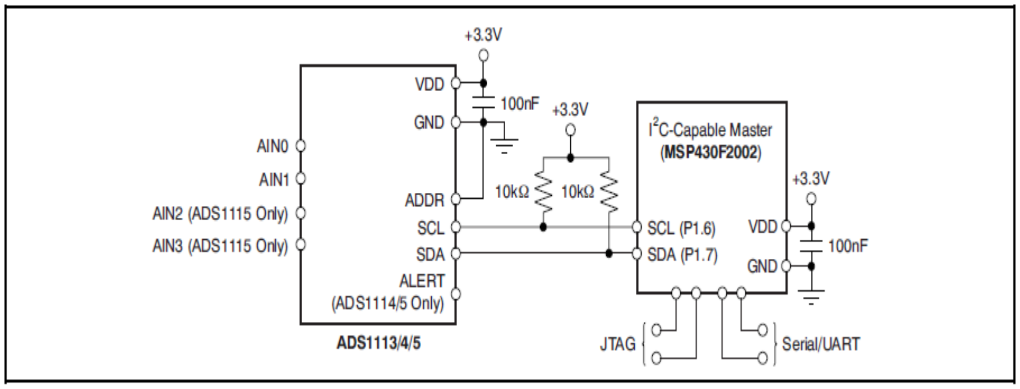


Figure 3.16 Basic Hardware Configuration of the ADS1115

3.4 Multiplexer

The multiplexers used in the design are the CD74HC4067 from Texas Instruments. The CD74HC4067 are digitally controlled analog multiplexers. They use silicon-gate operating technology. They consume very low power. They have a wide analog input voltage range. Their switching and propagation speeds are very high. These multiplexers have low "on" resistance and low "off" leakage. They are available on sparkfun breakout boards.

Figure 3.17 CD74HC4067 sparkfun breakout board

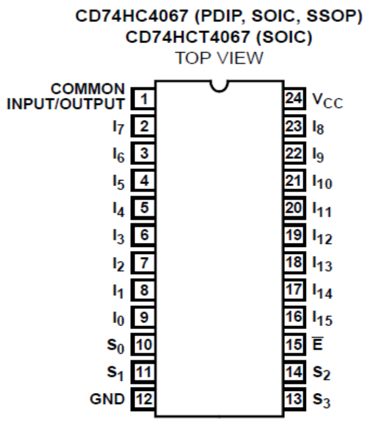


Figure 3.18 Pin Diagram of CD74HC4067

3.5 Wireless Network and ZigBee Radios

Wireless network is designed to communicate between the Arduino Fio and a laptop. A wireless network is one way to ensure that the operator can be removed to a safe distance away from the sensor. There were several options available for wireless networking. Software defined radios (SDR), IEEE 802.15.1 (Bluetooth) radios, IEEE 802.11 b/g (standard 2.4 GHz wireless), and ZigBee radios (802.15.4) are more suitable options for the design.

Software defined radios are open source and emerging technology. With SDR, manipulation of frequency, modulation, data rate, time synchronization, and power are very easy. SDRs have few significant draw backs compared to other options. As mentioned earlier, they employ very new technology. Their applications are not yet fully developed. Another drawback is that they are costly compared to other options. Bluetooth, standard 2.4 GHz devices are common and cheap options. Like SDR, standard 2.4 GHz hardware devices are more compatible with motherboards than micro-controllers. Bluetooth technology is widely used in mobile devices. They are semi-open standard and utilize low power. ZigBee standard and the Bluetooth standard both share the IEEE 802.15. The Arduino Fio board has a built-in socket designed for ZigBee radios. This is the biggest advantage of ZigBee radios over Bluetooth standard. ZigBee is an open standard created by ZigBee Alliance. It follows the IEEE802.15.4 spectrum and protocol. ZigBee uses orthogonal quadrature phase shift keying (OQPSK) digital modulation scheme with direct-sequence spread spectrum coding (DSSS) in the 2.4 GHz band. For greater distances, like more than 20 feet line-of-sight, high power ZigBees are available in the 900 MHz band. ZigBee standard is widely used in radio frequency applications where low data rates are required and long battery life is desired. They also offer very secure networking.

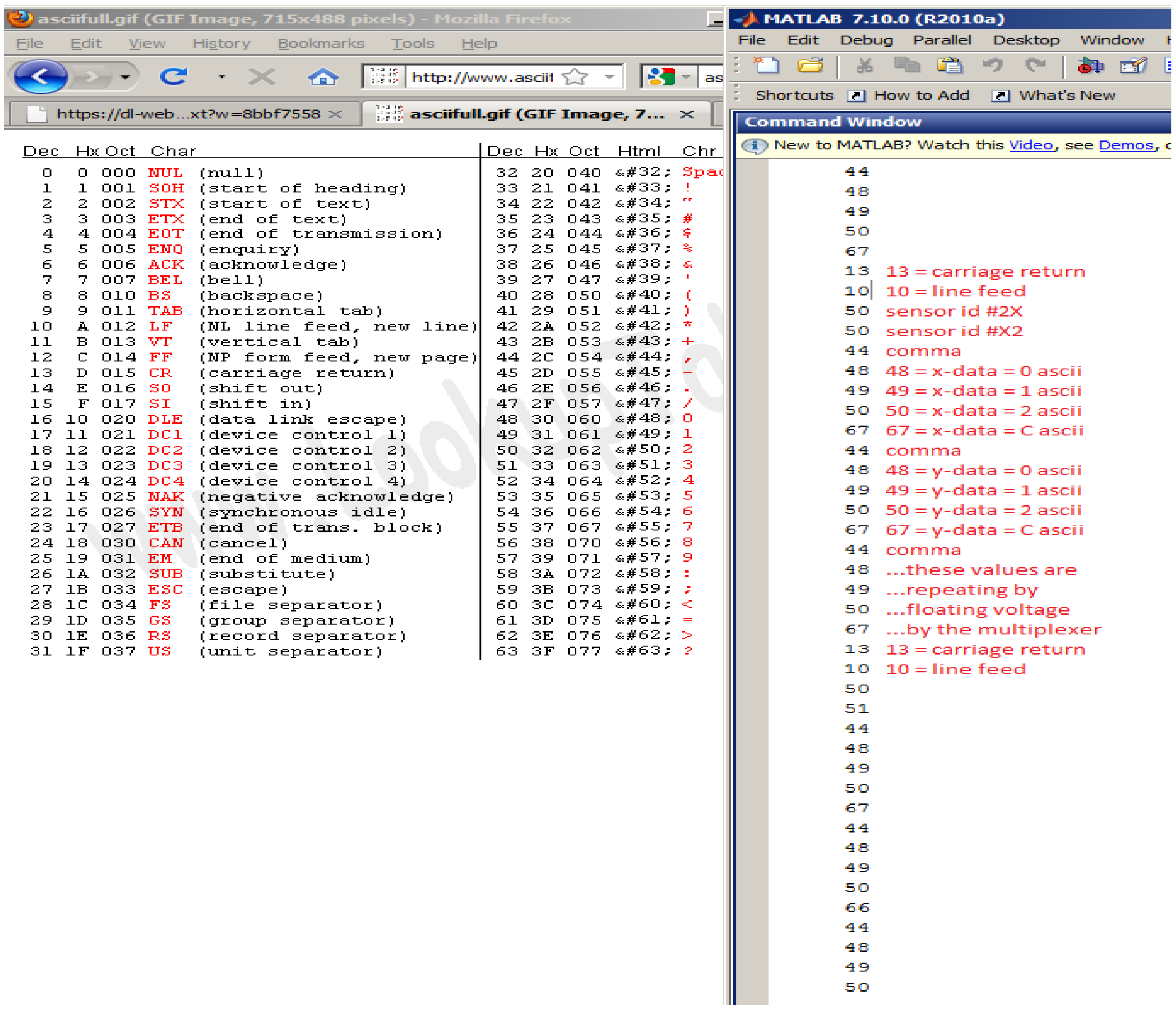
Figure 3.19 ZigBee

ZigBee applications include home entertainment and control like smart lighting, advanced temperature control, security devices, movies and music. They are used in home awareness applications like smoke detectors, water sensors, power sensors etc. In mobile services, they are used for security, access control, monitoring and control. They are also used in commercial and industrial building applications like energy monitoring, HVAC, access control, process control, energy management, and environmental management. ZigBee features including adjustable power, low cost, user-friendliness, and flexibility make them safe and secure option for wireless networking.

Chapter 4: Software Overview and System Architecture

ZigBee is setup to communicate between the microcontroller and a laptop. The Micro controller sends the ASCII data over ZigBee to the USB port of a laptop. The output of the microcontroller is in comma separated value (CSV) format. The output from microcontroller in CSV format is transmitted through ZigBee network and received by USB buffer on the client-side. This data is put into an array. The figure 4.1 shows the data received by the client-side software. Data reading and visualization are done in both Math work's Matlab and Python. GNU Octave and Oasis Montaj are the other options that can be to serially read the data and plot in 3D. GNU Octave is an open-source application and Oasis Montaj is a closed-source application. Oasis Montaj is much more capable software used in high level applications by Army Core of Engineers.

The python code written to handle the serial port communication and Matlab code is written to handle the data formatting and visualization. Matlab code can be converted into Python script including predefined libraries such as Matplotlib. The main difference between these two codes is Python starts the serial communication by calling library pySerial to communicate with the serial port.


asciifull.gif (GIF Image, 715x488 pixels) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.ascii
https://dl-web...xt?w=8bbf7558
asciifull.gif (GIF Image, 7...
MATLAB 7.10.0 (R2010a)
File Edit Debug Parallel Desktop Window
Shortcuts How to Add What's New
Command Window
New to MATLAB? Watch this Video, see Demos,

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	&#32;	Spa
1	1	001	SOH	(start of heading)	33	21	041	&#33;	!
2	2	002	STX	(start of text)	34	22	042	&#34;	"
3	3	003	ETX	(end of text)	35	23	043	&#35;	#
4	4	004	EOT	(end of transmission)	36	24	044	&#36;	$
5	5	005	ENQ	(enquiry)	37	25	045	&#37;	%
6	6	006	ACK	(acknowledge)	38	26	046	&#38;	&
7	7	007	BEL	(bell)	39	27	047	&#39;	'
8	8	010	BS	(backspace)	40	28	050	&#40;	(
9	9	011	TAB	(horizontal tab)	41	29	051	&#41;	)
10	A	012	LF	(NL line feed, new line)	42	2A	052	&#42;	*
11	B	013	VT	(vertical tab)	43	2B	053	&#43;	+
12	C	014	FF	(NP form feed, new page)	44	2C	054	&#44;	,
13	D	015	CR	(carriage return)	45	2D	055	&#45;	-
14	E	016	SO	(shift out)	46	2E	056	&#46;	.
15	F	017	SI	(shift in)	47	2F	057	&#47;	/
16	10	020	DLE	(data link escape)	48	30	060	&#48;	0
17	11	021	DC1	(device control 1)	49	31	061	&#49;	1
18	12	022	DC2	(device control 2)	50	32	062	&#50;	2
19	13	023	DC3	(device control 3)	51	33	063	&#51;	3
20	14	024	DC4	(device control 4)	52	34	064	&#52;	4
21	15	025	NAK	(negative acknowledge)	53	35	065	&#53;	5
22	16	026	SYN	(synchronous idle)	54	36	066	&#54;	6
23	17	027	ETB	(end of trans. block)	55	37	067	&#55;	7
24	18	030	CAN	(cancel)	56	38	070	&#56;	8
25	19	031	EM	(end of medium)	57	39	071	&#57;	9
26	1A	032	SUB	(substitute)	58	3A	072	&#58;	:
27	1B	033	ESC	(escape)	59	3B	073	&#59;	;
28	1C	034	FS	(file separator)	60	3C	074	&#60;	<
29	1D	035	GS	(group separator)	61	3D	075	&#61;	=
30	1E	036	RS	(record separator)	62	3E	076	&#62;	>
31	1F	037	US	(unit separator)	63	3F	077	&#63;	?

```
    44
    48
    49
    50
    67
    13   13 = carriage return
    10   10 = line feed
    50   sensor id #2X
    50   sensor id #X2
    44   comma
    48   48 = x-data = 0 ascii
    49   49 = x-data = 1 ascii
    50   50 = x-data = 2 ascii
    67   67 = x-data = C ascii
    44   comma
    48   48 = y-data = 0 ascii
    49   49 = y-data = 1 ascii
    50   50 = y-data = 2 ascii
    67   67 = y-data = C ascii
    44   comma
    48   ...these values are
    49   ...repeating by
    50   ...floating voltage
    67   ...by the multiplexer
    13   13 = carriage return
    10   10 = line feed
    50
    51
    44
    48
    49
    50
    67
    44
    48
    49
    50
    66
    44
    48
    49
    50
```

Figure 4.1 Matlab Command Window

4.1 Matlab

Mathwork’s Matlab is a closed-source application. It is able to serially read and plot 3-D data. Matlab code is written to handle three main duties. They are Serial port communication, data collection and data visualization. The baud rate of serial port is set up at 57600 bps. COM3 is used as serial port on windows operating system by ZigBee. On UNIX and LINUX systems, /dev/ttyUSB is used as the serial port. ‘fwrite’ function is used for writing to the serial port. The data read from the sensors is in CSV format. This data is put into a zeroed array using Matlab code. The code is written to create a loop which iterates until the last sensor is read. The code converts hexadecimal data into decimal by using the function ‘hex2dec’. From the looped data each sensor reading can be accessed by pointer. The visualization of the data is done using open-source Matlab library named ‘vectarrow’. This library is set up to plot in both 2-D and 3-D space. Using Matlab options more than one sensor reading can be viewed at a time. A sensor reading is plotted as an arrow in 3-D space as shown in figure 4.2. This Matlab code can be translated into GNU Octave code also except for the serial communication portion. To be able to communicate with the serial port in Octave, it requires linking of Python library (pySerial) to Octave. For the visualization of data in Octave, ‘vectarrow’ library can be imported by ‘addpath’ option and 3-D output can be plotted.

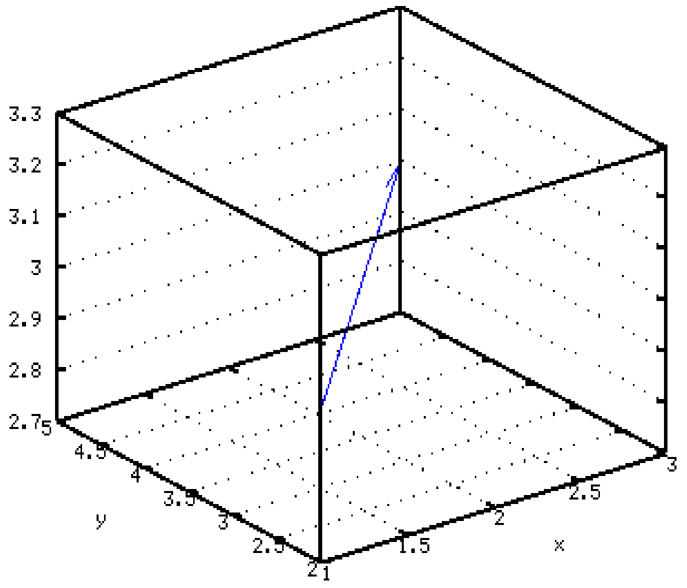


Figure 4.2 Matlab Plot of Vectarrow

4.2 Python

Serial communication in Matlab has a serial buffer issue, which takes time to communicate with the serial port. Python uses pySerial library to communicate with the serial port. PySerial library is faster when compared to Matlab. Python is an open-source application. It can handle multiprocessors and multi-thread hence it is more time efficient than Matlab.

Data visualization in Python can be done in two ways. Python has a visual 3-D library for plotting 3-D data. This library consists of a predefined function named ‘arrow’, which can be called by code and can be used for real time arrow changes. Another technique that can be adapted for 2-D visualizations is by drawing a vector using Python code. This method is very simple and more than one sensor reading can be plotted using the Matplotlib library.

4.3 System Architecture

The microcontroller is the heart of the design. The microcontroller configures the ADC and reads it through I2C. It controls the analog multiplexers via DIO. It collects the data from sensors and transmits it serially via ZigBee radios. The analog input 5 of the Arduino Fio is connected to the SCL line and analog input 4 is connected to the SDA line of I2C bus. These connections are to configure and read ADC by the Arduino Fio microcontroller. When the I2C bus is in idle mode, no communication occurs. Only master devices, in this design Arduino Fio, can start the communication by initiating a START condition. When the SDA line changes the state between START and STOP conditions, the SCL line will be clocked accordingly. If the SDA line moves into START condition, the SCL line goes from HIGH to LOW. IF the SDA line moves into STOP condition, the SCL line goes from LOW to HIGH. Each byte of the data or address sent on I2C bus is acknowledged with an *acknowledge* bit. When the Arduino Fio sends a byte of data over I2C to the ADC, it waits for the acknowledge byte. The ADC acknowledges by driving the SDA low. Then Arduino clocks the acknowledge bit. In a similar fashion, when Arduino reads the byte of information, it drives SDA low to acknowledge ADC and drive clocks the acknowledge bit. Only the micro-controller (master) can clock the clock line.

The code implemented in the Arduino Fio wiring language configures the initiation settings of I2C bus. The “address” parameter in the code refers to the 7 bit I2C address of ADS1115. The “channel” parameter refers to the inputs of ADS1115 being read. In this design the channel should be 0, 1, or 2 as only the readings of the first three inputs are read. The “index” parameter is a 16 bit address for the EEPROM, its values range from 0 to 511. It represents the address on the EEPROM where the reading is to be stored. The code generates a Pulse Width Modulation (PWM) signal used by the Set/Reset circuit. The “config” parameter in the code is an array that carries the configuration data byte that the ADS1115 must send to read an input. After the configuration data is sent, Arduino Fio requests the ADC data by sending a byte of zeroes.

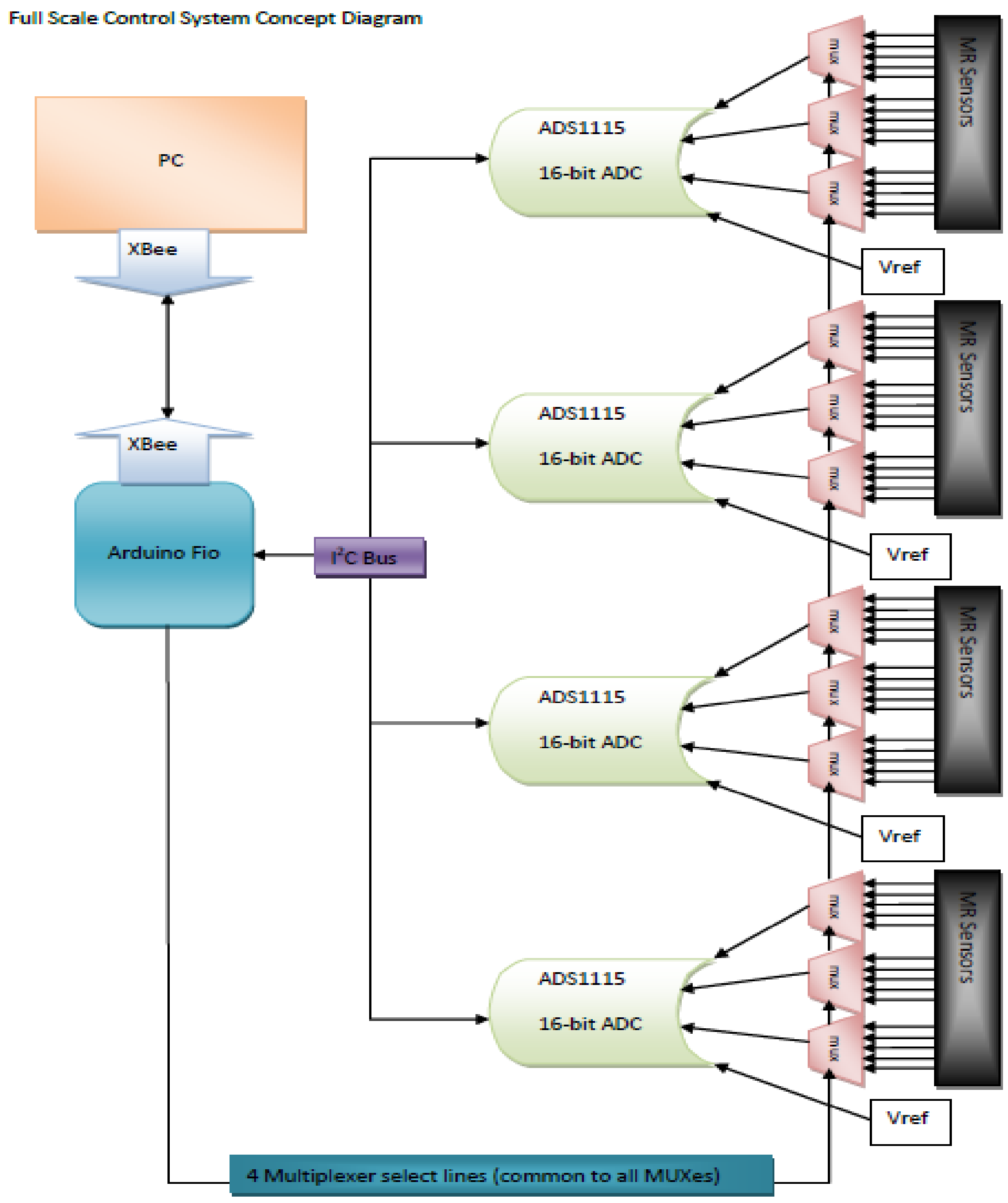

Full Scale Control System Concept Diagram
PC
XBee
XBee
Arduino Fio
I²C Bus
ADS1115
16-bit ADC
mux
MR Sensors
Vref
4 Multiplexer select lines (common to all MUXes)

Figure 4.3 Concept Diagram of the Design

The Arduino Fio has a time out limit to receive the data from the ADC. It does not wait more than 10ms, before this time out the data from the ADS1115 will be received by the Arduino and stored in the EEPROM memory. The purpose the time limit is to protect the Arduino Fio from reading data from any other ADC that is not connected to it.

The multiplexer control settings are also configured by the code developed in the Arduino Fio wiring language. The code reads “channel” argument that refers the input channel to be read and sets the DIO. The output of each multiplexer is connected to one input of the ADS1115. All the control signals of multiplexers are connected in parallel. The code is developed to read the EEPROM contents and write them in a matrix fashion. The code consists of a for-loop that iterates until all the sensor readings are read. Each sensor reading will have X, Y, Z axes. The code assumes, each x-axis of 16 sensors will be connected to input-0 of one ADC through multiplexer, Y-axis to the input-1 of the same ADC, and Z-axis to the input-2 of the same ADC. To print the contents of EEPROM, “printMatrix” function is developed. The format of the contents in matrix fashion is shown below:

```
01,XXXX,YYYY,ZZZZ

02,XXXX,YYYY,ZZZZ

03,XXXX,YYYY,ZZZZ

04,XXXX,YYYY,ZZZZ

...

NN,XXXX,YYYY,ZZZZ
```

Where XXXX is the ADC reading from the sensor in the x-axis, YYYY is for the y-axis, ZZZZ is for the z-axis, and NN is from 00 to 64. This data is sent through a serial interface over ZigBee to the laptop.

Chapter 5: Design Verification & Results

5.1 Baseline Data Set Acquisition

The working of the design is verified by the acquisition of a baseline data set. The baseline data set provides a data package for testing of the sensors' performance and functionality. The acquisition module was a National Instruments NI9219 4-channel series module. The NI9219 is a 24-bit analog input ADC designed for multipurpose testing in a compact DAQ or CompactRio chassis.

An inert 105mm HEAT-T M456A1 was used as a target. This is a highly explosive anti tank cartridge. The figure shows the test set-up for the baseline data set acquisition.

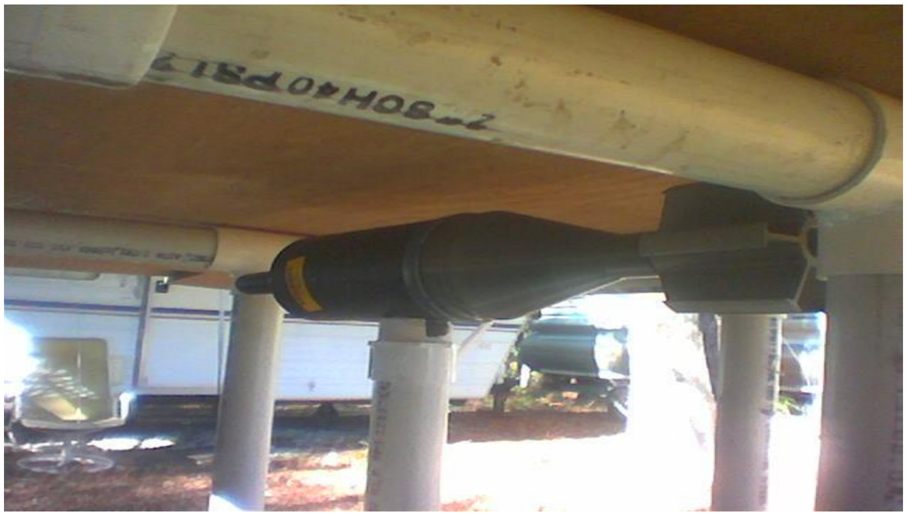


Figure 5.1 HEAT-T suspension in the test set-up

The test set-up has a 36”x36” grid with two inch spacing. The grid was marked off above the target for data points. It has total 361 data points. HEAT-T was suspended above the ground to reduce the noise effects of the ferrous objects in the ground. The grid used in the set-up is shown in the figure 5.2.

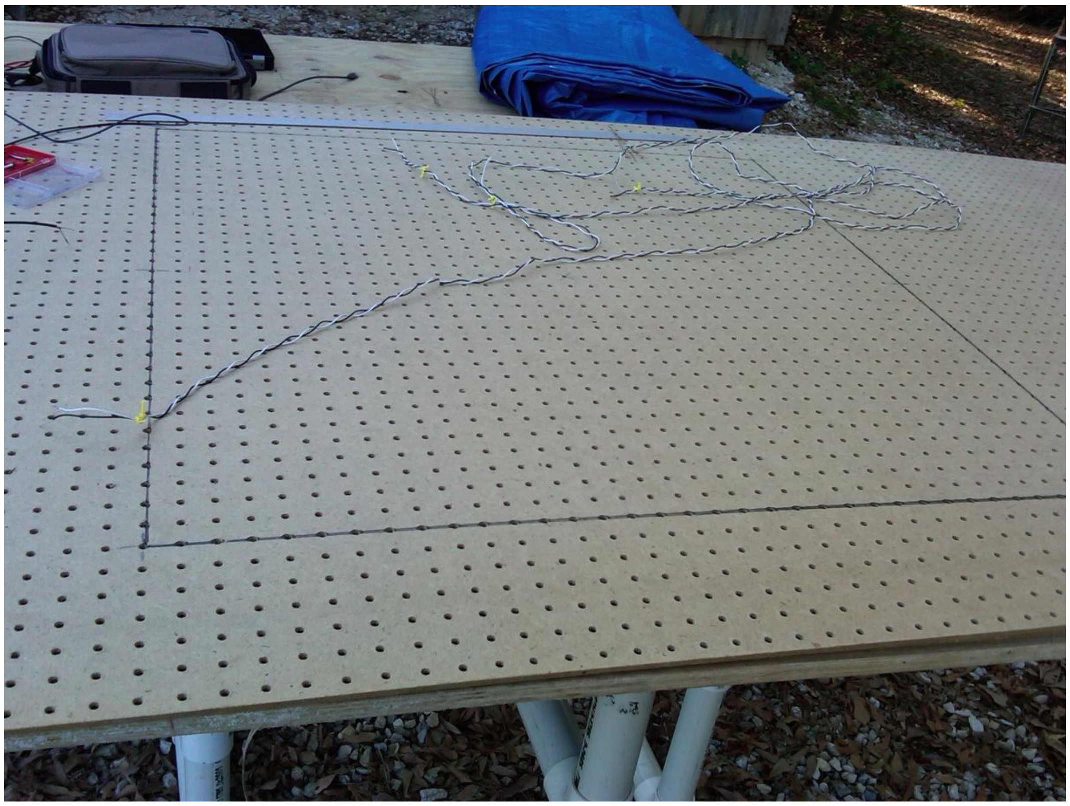
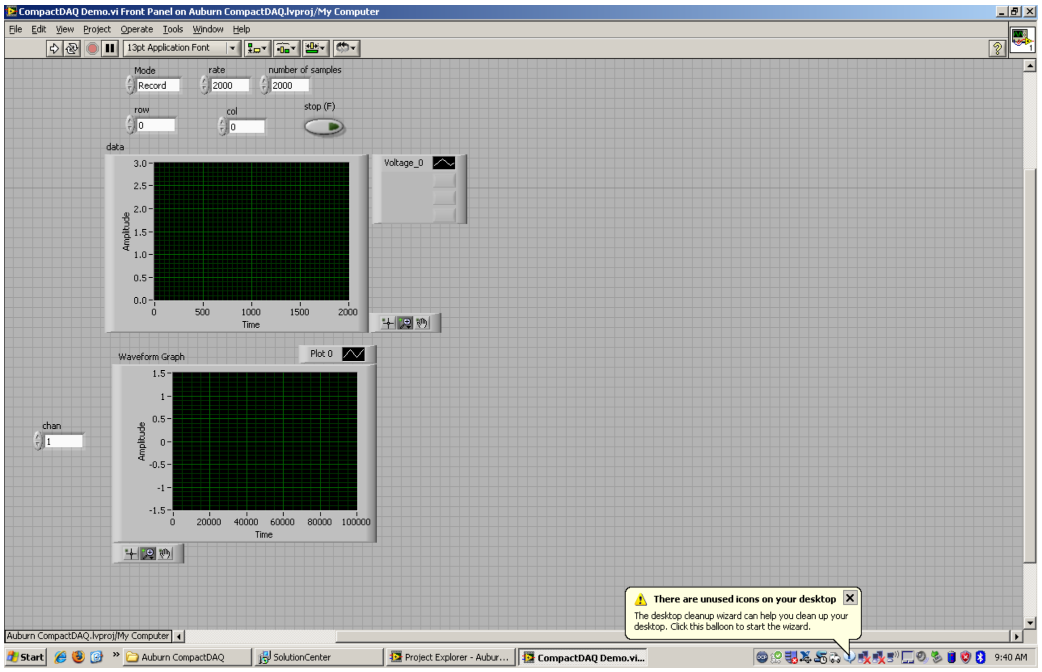


Figure 5.2 the grid in the test set-up

A simple LABVIEW VI was constructed to control the NI9219 and CompactDAQ chassis in the acquisition process. The sample rate was set to 25 KHz. The three axes (X, Y, and Z) of HMC2003 are connected to an0, an1, an2 channels of NI9219. The collected data is exported to a column defined excel spreadsheet. The data at each point is saved as an individual file. Each data point was sampled for approximately one second and by using MATLAB, the first 100 points were averaged and placed in a spreadsheet. The baseline data set acquisition was successful in proving the working of the design and it helped in the partial development of the GUI.



There are unused icons on your desktop
The desktop cleanup wizard can help you clean up your desktop. Click this balloon to start the wizard.

Figure 5.3 Screen Shot of LABVIEW VI

5. 2 GUI display

Graphical user interface display of the four sensor readings at a single data point is shown in the figure 5.4. GUI display is achieved by using open-source Matlab library called ‘vectarrow’. This library can plot both 2-D and 3-D and add an arrow at the end of the vector.

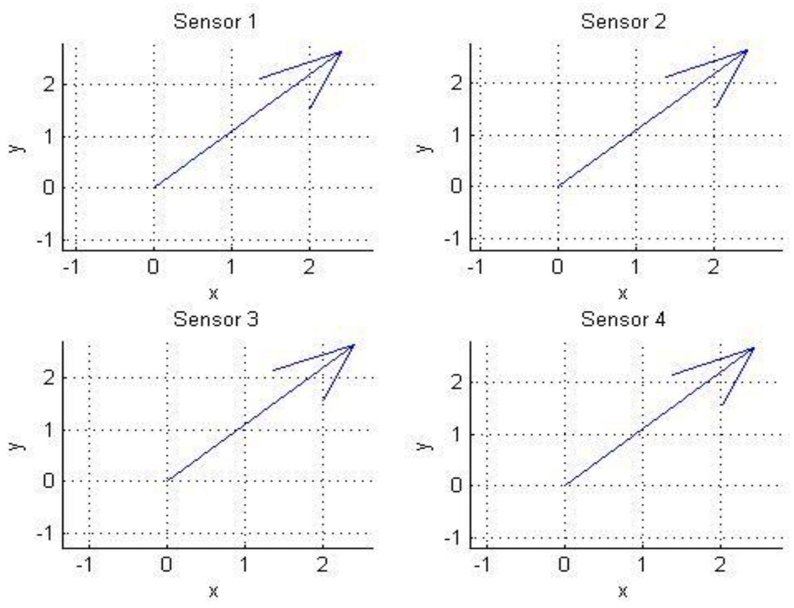


Figure 5.4 GUI display

5.3 Results

The Earth contains a magnetic field created by sources in the core. The Earth's magnetic field would be similar to that of a magnetic dipole located at the center of the Earth with its axis aligned sub parallel to the geographic axis. Several objects and minerals that are weakly magnetic can cause anomalies in the Earth's magnetic field. Many man-made objects containing highly magnetic materials can cause large anomalies in the Earth's magnetic field. In this experiment the sensor's performance is tested against a UXO target at fixed location. The target is suspended under a grid as shown in the figure 5.4. The sensor is moved along the grid and the data is collected at each point located with a spacing of 2-inch. The data collected through the acquisition is process is stored in a MS Excel file.

The geometry of the grid is shown in the figure 5.4. The figure 5.5, figure 5.6, figure 5.7 show the magnitude plot of sensor readings at position A, B, and C. Position A refers to the data collected at each data point on the first row of the grid. Position B is the fifth row on the grid. Position C is the row at the center of the grid where the HEAT-T is located. The plots interpret the variations in the field induced by magnetic dipoles in the ferrous object as the sensor approaches the center of the grid.

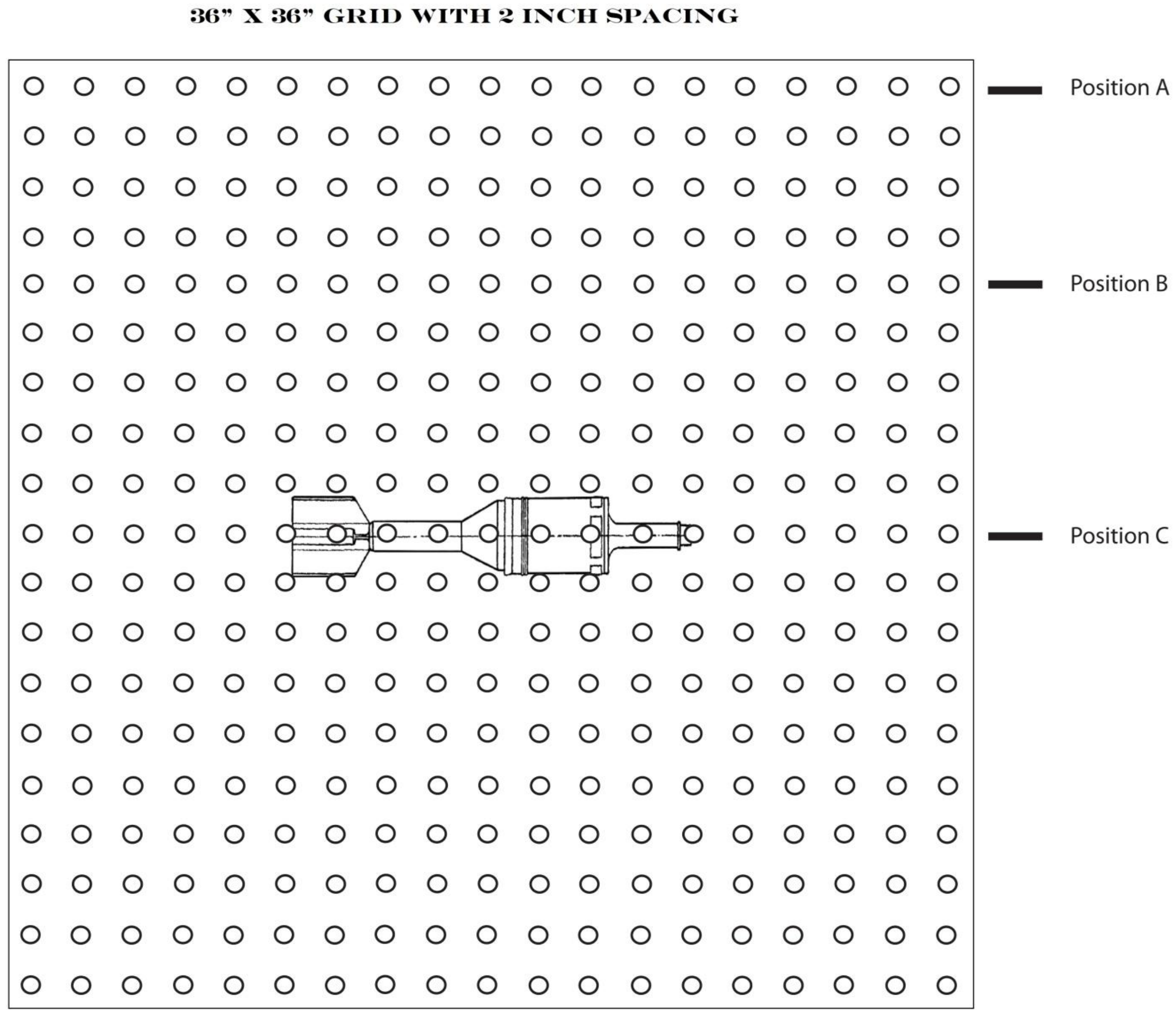


Figure 5.5 the geometry of the grid

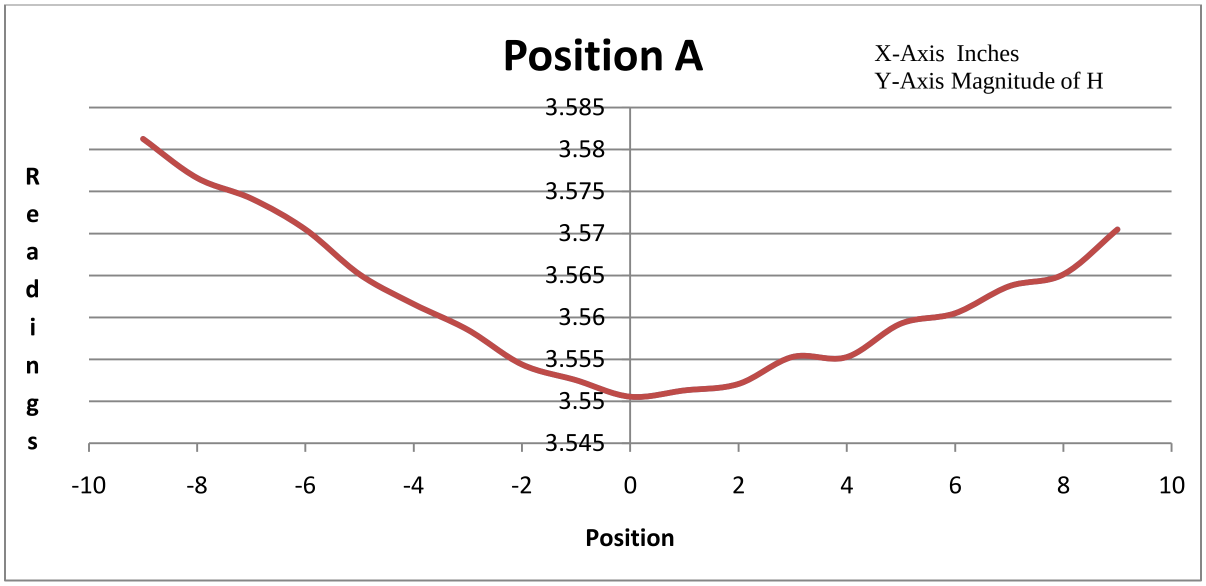


Figure 5.6 Results plot at position A on the grid

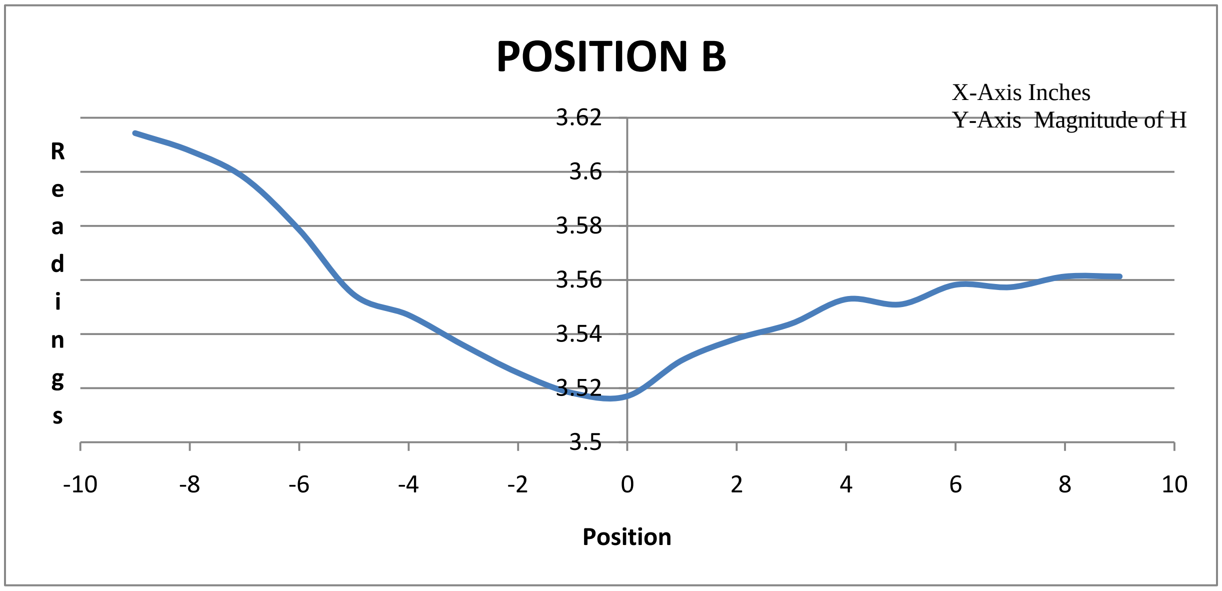


Figure 5.7 Results plot At Position B on the grid

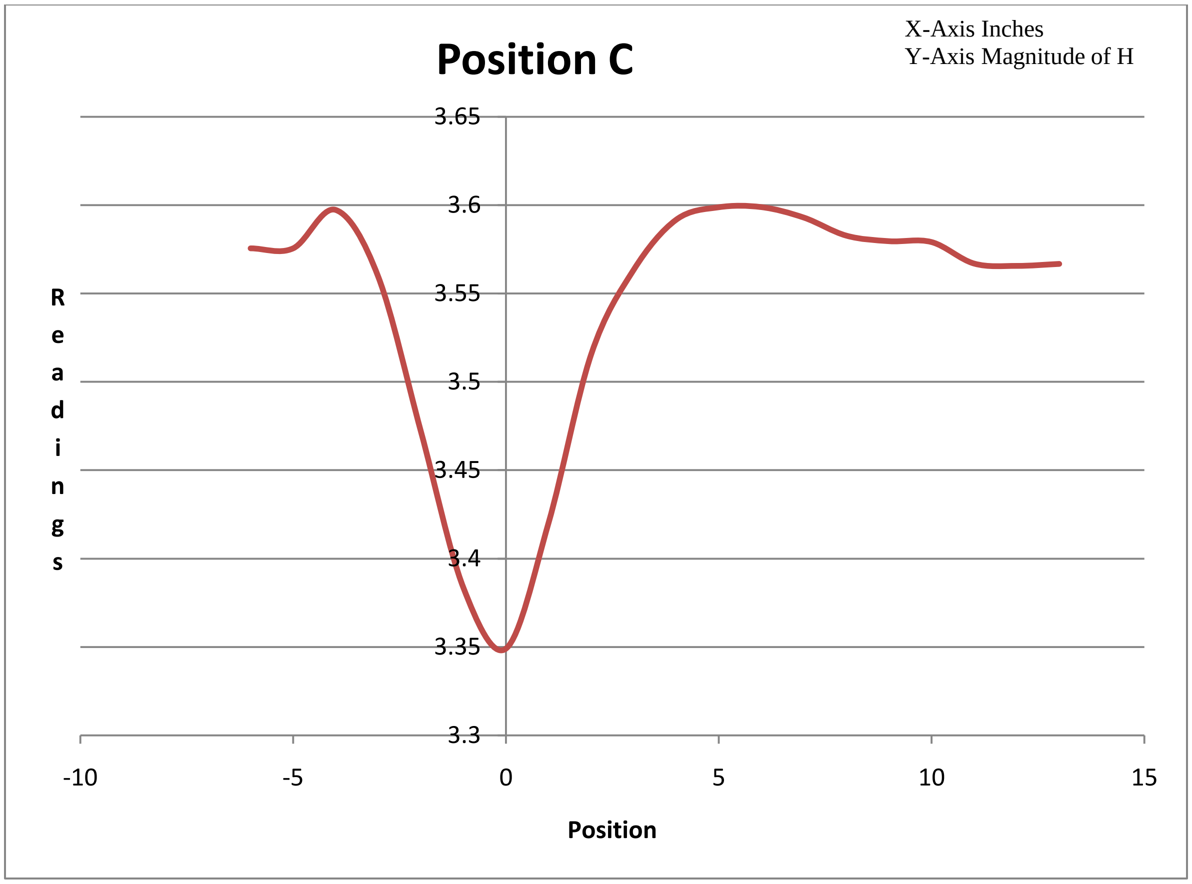


Figure 5.8 Results plot at Position C on the grid

Chapter 6: Conclusion

The body of this thesis describes the design and development of a simple, user friendly, and cost efficient magneto-resistive unexploded ordinance (UXO) sensor. In the beginning of the chapters, the theory of different magnetic induction techniques is described in detail. In the theory, a detailed look was given to the principle of magneto-resistive effect. After introducing Magnetoresistive effect in ferrous materials, a detailed theory of Anisotropic Magneto-Resistive sensors (AMR) is described to provide a better understanding of the underlying principles of the sensors used in the design. Having introduced the theory, the details of hardware equipment employed in the design are described in the hardware chapter. The details of different components available and the requirement of each component in the design is explained. Next, the development of a wireless network and the software used to process the data are describes in detail. With hardware, wireless network, and software setup complete, a baseline data set acquisition was completed for the design verification.

This effort towards the development of magneto-resistive unexploded ordinance sensor array proved to be successful. This design offers flexibility in hardware selection along with the ability to expand the number of sensors used.

There are several modifications that can be employed to improve the design. The revision of VI would simplify and speed up the acquisition process. The removal of large ferrous objects like vehicles should be done to make the test site more ideal. The multiplexing technique employed in the design needs further improvement. The technique used in this design uses three multiplexers for multiplexing of analog signals from 16 sensors into one external ADC. By employing de-multiplexing on the I2C bus, there would be a need of only 2 multiplexers (one for SDA, and one for SCL). The set/reset driver circuitry also needs to be improved. The push/pull driver implemented in the design does not meet Honeywell's recommended current level. For accurate optimization and utilization of HMC2003 data, fully operational set/reset driver is very important.

References

[1] Lenz, E., James, “A review of magnetic sensing”, Proceedings of IEEE, 78, No. 6:973-989 (1990).

[2] Ripka, Pavel, “Magnetic Sensors and Magnetometers”, (Artech House Remote Sensing Library).

[3] Pant, B., B., Fall 1987, "Magnetoresistive Sensors," Scientific Honeyweller, Vol. 8, No. 1:29-34.

[4] Caruso, J., Michael, Bratland, Tamara, Smith, H., Carl, and Schneider, W., Robert, “A new perspective on magnetic field sensing”, Nonvolatile Electronics, Inc. December 1, 1998.

[5] Magnetic Sensors International Data Book 1995, (Siemens Aktiengesellschaft, München, 1994).

[6] Janicke, J., M., The Magnetic Measurement Handbook, 1994, New Jersey: Magnetic Research Press.

[7] Daughton, M., J., “GMR and SDT sensor applications,” IEEE Trans. Magn. 36, 2773 - 2778, (2000).

[8] Billings, D., Stephen, Pasion, Catherine, Walker, Sean, and Laurens, Beran, “Magnetic Models of Unexploded Ordnance”, IEEE Transactions on Geoscience and Remote sensing, Vol.44, No.8, August, 2006.

[9] HMC 2003: http://www.magneticsensors.com/datasheets/hmc2003.pdf. Retrieved 2010-5 29.

[10] “Handling of Sensor Bridge”, http://www51.honeywell.com/aero/commom/documents/myaerospacecatalogdocuments/Defence _ Brochures-documents/Magnetic__Literature_Application_notes-documents/AN212_Handling_of_Sensor_Bridge_Offset.pdf. Retrieved 2010-5-29.

APPENDIX – SOURCE CODE

```
/*
Arduino Fio Pins Used:

 3V3 - Power
 GND - Ground

 Analog Input 5 - I2C SCL
 Analog Input 4 - I2C SDA

 Digital I/O 3  - Set/Reset Line

 Digital I/O 9  - Multiplexer LSB Select Line
 Digital I/O 10 - Multiplexer 2nd LSB Select Line
 Digital I/O 11 - Multiplexer 2nd MSB Select Line
 Digital I/O 12 - Multiplexer MSB Select Line
 Digital I/O 13 - On-Board LED

 */

//Headers
#include <EEPROM.h>
#include <avr/pgmspace.h>
#include <Wire.h>


// String Constants
// This is done so that the strings do not use RAM, only Flash memory
prog_uchar VERSION[] PROGMEM = "\n\rAU GMR - v2.1.0 ";
prog_uchar MAIN_MENU[] PROGMEM = "Main Menu:\n\rE:   EEPROM Menu\n\rS:   Read Sensors\n\r";
prog_uchar EE_MENU[] PROGMEM = "\n\rEEPROM MENU:\n\rR: Read EEPROM\n\rC: Clear EEPROM\n\rF: Fill EEPROM\n\rX: Exit\n\r";

// Constants
#define ADC_W 0x48
#define ADC_X 0x49
#define ADC_Y 0x4A
#define ADC_Z 0x4B

// Global Variable Declarations
char inByte = 0;         // incoming serial byte
int addr = 0;                                  // initializing EEPROM Write
int val;                        // outgoing EEPROM Write data
unsigned long now;          // non-busy-waiting timing mechanism
unsigned long then;          // non-busy-waiting timing mechanism
int ledState = LOW;                // most current time reading

// Implementation Globals

// Axes = number of axes per sensor
// ADCs = number of ADCs in implementation
// Snsr = number of Sensors implemented per ADC
int numAxes = 3; // Full Scale - 3
int numADCs = 4; // Full Scale - 4
int numSnsr = 16; // Full Scale - 16

/*
This is the function called when the Arduino starts up.

 Opens the Serial Port on 57600 baud
 Clears the EEPROM

 calls establishContact()
 */
void setup()
{
  // start serial port at 57600 bps:
  Serial.begin(57600);
  Wire.begin();

  pinMode(3, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);

  // Clear EEPROM Function
  // write a 0 to all 1024 bytes of the EEPROM for EEPROM initialization
  clearEEPROM();

  establishContact();  // send a byte to establish contact until receiver responds
}

/*
Functionally Equivalent to void main(){while(1){ [CODE] }}

 Provides the Main interface to the menu system, which navigates the functions of the code.
 */
void loop()
{
  // if we get a valid byte, read analog ins:
  if (Serial.available() > 0) {
    // get incoming byte:
    inByte = Serial.read();
    // Use ASCII codes to differentiate input instead of ASCII Characters
    if(inByte == 83 || inByte == 115) // S/s
      readGMR();
    else if(inByte == 69 || inByte == 101) // E/e
      menuEE();

    printMenu();
  }

  // A sanity check LED blink, so we know the Arduino is running the loop.
  activityLED();
}

/*
A function that prints the strings that comprise the Main Menu
 */
void printMenu(){
  int i = 0;
  while(i < sizeof(VERSION)){
    Serial.print(pgm_read_byte_near(VERSION + i));
    i++;
  }
  i = 0;
  while(i < sizeof(MAIN_MENU)){
    Serial.print(pgm_read_byte_near(MAIN_MENU + i));
    i++;
  }
}

/*
  A sub-menu for control of the EEPROM

 4 Options:

 R - Read the EEPROM, print contents to the Serial Port
 F - Write all 1's to the EEPROM
 C - Write all 0's to the EEPROM
 X - Exit to Main Menu
 */
void menuEE(){
  int i = 0;
  // Print the EEPROM Menu
  while(i < sizeof(EE_MENU)) {
    Serial.print(pgm_read_byte_near(EE_MENU + i));
    i++;
  }

  // Clear the buffer
  Serial.flush();

  // Block, wait for new input
  while (Serial.available() <= 0) activityLED();

  inByte = Serial.read();

  // Use ASCII codes to differentiate input instead of ASCII Characters
  if(inByte == 82 || inByte == 114)  // R/r
    readEEPROM();

  else if(inByte == 67 || inByte == 99) // C/c
    clearEEPROM();

  else if(inByte == 70 || inByte == 102) // F/f
    fillEEPROM();

  else if(inByte == 88 || inByte == 120) // X/x
    return;

  menuEE();
}

// Passively blinks the LED every 2 seconds or so as non-serial feedback
void activityLED(){
  now = millis();
  if (now - then > 2000) {
    then = now;
    if (ledState == LOW)
      ledState = HIGH;
    else
      ledState = LOW;

    digitalWrite(13, ledState);
  }
}

/*
Blocks, sending '.' characters to the Serial port until the Serial port gets a response
```

```
 */
void establishContact() {
  while (Serial.available() <= 0) {
    Serial.print('.', BYTE);   // send instruction
    activityLED();
    delay(1000);           // operational timer set to 1 sec.
  }
}

/*
Takes in the channel argument for the MUX channel that
 needs to be read (0-15) and sets Digital I/O accordingly
 */
void setMUX(byte channel){
  if (channel > 15) return;

  if ((channel & 0x08) != 0) digitalWrite(12, HIGH);
  else digitalWrite(12, LOW);

  if ((channel & 0x04) != 0) digitalWrite(11, HIGH);
  else digitalWrite(11, LOW);

  if ((channel & 0x02) != 0) digitalWrite(10, HIGH);
  else digitalWrite(10, LOW);

  if ((channel & 0x01) != 0) digitalWrite(9, HIGH);
  else digitalWrite(9, LOW);
}

// Sets up the ADC with the given address for readings on the given channel
void readADC(byte address, byte channel, int index){
  byte config[3];
  byte a, b;
  long timeoutTick, timeoutTock;
  boolean skip = false;

  // PWM generates a signal that the Set/Reset Circuit needs
  for(int i = 0; i <= 5; i++){
    digitalWrite(3, LOW);
    delayMicroseconds(20);
    digitalWrite(3, HIGH);
    delayMicroseconds(1980);
  }

  a = 0;
  b = 0;

  config[0] = 0x01;
  config[1] = 0x85 | (((channel + 1) << 4) & 0xF0);
  config[2] = 0x83;

  Wire.beginTransmission(address);
  Wire.send(config[0]);
  Wire.send(config[1]);
  Wire.send(config[2]);
  Wire.endTransmission();

  Wire.beginTransmission(address);
  Wire.send(0x00);
  Wire.endTransmission();
  timeoutTick = millis();
  Wire.requestFrom(address, byte(2));
  while(Wire.available() < 2){
    timeoutTock = millis();
    if((timeoutTock - timeoutTick) > 10){
      skip = true;
      break;
    }
  }
  if (!skip){
    a = Wire.receive();
    b = Wire.receive();
  }

  EEPROM.write((2 * index), a);
  EEPROM.write((2 * index) + 1, b);
}

/*
A function that will eventually become the GIANT-GREEN-GO-BUTTON code.
 */
void readGMR() {
  int i = 0;
  byte currentADC = 0x00;

  // connect sensor input to EEPROM write/read here

  for(int i = 0; i < (numAxes*numSnsr*numADCs); i++){
    digitalWrite(13, HIGH);

    currentADC = ADC_W + (( i / numAxes ) % numADCs );
    setMUX(i/(numAxes*numADCs));

    if (i % numAxes == 0){
      // Read from ADC, CH-0
      readADC(currentADC, 0x00, i);
    }
    else if (i % numAxes == 1){
      // Read from ADC, CH-1
      readADC(currentADC, 0x01, i);
    }
    else if (i % numAxes == 2){
      // Read from ADC, CH-2
      readADC(currentADC, 0x02, i);
    }
  }

  // GO
  printMatrix();
  digitalWrite(13, LOW);
}

/*
Read EEPROM and send the contents to the Serial port
 */
void readEEPROM(){
  int doneLooping = 0;
  while(doneLooping == 0){
    // read a byte from the current address of the EEPROM
    val = EEPROM.read(addr);
    if((addr % 8) > 0 ){
      Serial.print(", ");
      Serial.print(val, HEX);
    }
    else if(addr != 1024){
      Serial.println();
      Serial.print(addr);
      Serial.print("\t");
      Serial.print(val, HEX);
    }

    // advance to the next address of the EEPROM
    // there are only 1024 bytes of EEPROM, from 0 to 1023, so if we're
    // on address 1024, wrap around to address 0
    if (addr <= 1023){
      addr++;
    }
    else{
      doneLooping = 1;
      addr = 0;
    }
  }
  Serial.println();
  addr = 0;
}

/*
Read EEPROM for the sensor data and send the contents to the Serial port in a 3x85 matrix of unsigned 16-bit HEX values
 */
void printMatrix(){
  int doneLooping = 0;
  int s = 0;
  while(doneLooping == 0){
    // read a byte from the current address of the EEPROM
    val = EEPROM.read(addr);

    if((addr % (numAxes*2)) > 0 ){
      Serial.print(",");
      byte msb = byte(val);
      val = EEPROM.read(++addr);
      if(msb <= 15) Serial.print(0, HEX);
      Serial.print(msb, HEX);
      if(val <= 15) Serial.print(0, HEX);
      Serial.print(val, HEX);
    }
    else if(addr != (numAxes*numSnsr*numADCs*2)){
      Serial.println();
      if(++s < 10) Serial.print(0, DEC);
      Serial.print(s);
      Serial.print(",");
      byte msb = byte(val);
      val = EEPROM.read(++addr);
      if(msb <= 15) Serial.print(0, HEX);
      Serial.print(msb, HEX);
      if(val <= 15) Serial.print(0, HEX);
      Serial.print(val, HEX);
    }

    // advance to the next address of the EEPROM
    // there are only (numAxes*numSnsr*numADCs*2) bytes of EEPROM used for sensor
    // data, from 0 to (numAxes*numSnsr*numADCs*2)-1, so if we're
    // on address (numAxes*numSnsr*numADCs*2), wrap around to address 0
    if (addr < (numAxes*numSnsr*numADCs*2)){
      addr++;
    }
    else{
      doneLooping = 1;
      addr = 0;
    }
  }
  Serial.println();
  addr = 0;
}


// Write all 0's to the EEPROM
void clearEEPROM(){
  for (int i = 0; i < 1024; i++)
    EEPROM.write(i, 0);
}

// Write all 1's to the EEPROM
void fillEEPROM(){
  for (int i = 0; i < 1024; i++)
    EEPROM.write(i, 0xFF);
}
```