DETERMINING HEAD POSITION TO ASSIST ELECTRIC-POWERED WHEELCHAIR OPERATION FOR PERSONS WITH TRAUMATIC BRAIN INJURY

by

Alex James Bevly III

Bachelor of Science in Computer Engineering, University of Pittsburgh, 2002

Submitted to the Graduate Faculty of

the School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science in Electrical Engineering

University of Pittsburgh

2005

UNIVERSITY OF PITTSBURGH

SCHOOL OF ENGINEERING

This thesis was presented

by

Alex James Bevly III

It was defended on

April 7, 2005

and approved by

Donald M. Spaeth, Adjunct Assistant Professor, Rehabilitation Science and Technology

Rory A. Cooper, Distinguished Professor and FISA/PVA Chair, Rehabilitation Science and Technology

Ronald G. Hoelzeman, Associate Professor, Electrical Engineering

J.T. Cain, Professor, Electrical Engineering

Thesis Director: Marlin H. Mickle, Nickolas A. DeCecco Professor, Electrical Engineering

DETERMINING HEAD POSITION TO ASSIST ELECTRIC-POWERED WHEELCHAIR OPERATION FOR PERSONS WITH TRAUMATIC BRAIN INJURY

Alex James Bevly III, MSEE

University of Pittsburgh, 2005

Monitoring head position in persons with a traumatic brain injury may provide a means for independent powered mobility. Given the often limited residual functions of attending, visual processing, and motor control, the operation of an electric-powered wheelchair must be constantly monitored to ensure the safety of these users. Human-directed support is not always available and does not encourage independent mobility. The solution proposed for this problem is placement of a magnet on the rear of the person's head. Strategically placed linear analog Hall effect sensors that are fixed in a stationary headrest can then track the magnet; thus, accurately determining head positioning. With this proposed head tracking, a specialized interface to the electric-powered wheelchair controller can be used to ensure the person's head is attending the direction of travel asserted by a conventional, direction-sensing joystick.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

PREFACE

Without their contributions in knowledge, this research would not be possible. Thank you:

Dr. Rory A. Cooper

Dr. Donald M. Spaeth

Dr. Marlin H. Mickle

Dr. Songfeng Guo

Karl W. Brown

To my mother Elaine H. Bevly, who never let me settle for less than my potential.

1.0 INTRODUCTION

A resounding need to care for people with enduring head injury is prevalent in the world. With the different types of injuries, those that damage the brain are the most likely to result in death or chronic disability [1]. In the U.S., the leading causes of traumatic brain injury (TBI) are vehicular accidents, falls, and violence. TBI related to falls has an outcome of fatal injury effecting 11% of these people [1]. Supporting this, the Center for Disease Control and Prevention (CDC) [2] has found yearly:

- *Motor vehicle crashes* are the leading cause of hospitalization.
- *Falls* are the leading cause of a TBI among the elderly.
- *Suicidal violence* and *personal assaults* involving firearms are the leading causes in death related to a TBI.

Of the persons surviving these serious injuries, many will have to live with permanent disability. Concerning the effects of these injuries, the CDC [2] also reported that yearly:

- 230,000 people survive through hospitalization.
- 80,000 to 90,000 people experience long-term or permanent disability.
- 50,000 people die.

In the U.S. military, the Defense and Veterans Brain Injury Center (DVBIC) found that TBI accounts for at least 14 – 20% of surviving casualties in times of combat [3].

The CDC further reported that an estimated incidence of 1.5 million Americans experience a traumatic brain injury each year [2]. The number of Americans sustaining permanent TBI-related disability is estimated at 5.3 million people at any given time [2]. The

CDC also recognizes that the outcomes of TBI result in “cognitive, emotional, sensory, and motor impairments” that profoundly affect a person’s quality of life. The high incidence of a TBI among Americans justifies demand to facilitate and expand technology options that can assist with the care for this population.

People experiencing serious spinal cord injuries or a TBI often need alternative means of mobility. Examples are walkers, scooters, manual wheelchairs, and electric-powered wheelchairs (EPW). Depending on the body size, life style, and diagnosis of the user, clinicians recommend these alternatives best suited to the person’s needs. Although an EPW is often prescribed to persons with a TBI, the users often sustain significant motor impairments. Unfortunately, the residual motor function after experiencing a TBI often leaves a person with inadequate driving skills to operate an EPW.

The presence of severe disabilities makes it difficult for people to return to their “pre-trauma” activities, or embark on new activities in life. The research done by Keren points out that the people who survive a TBI often suffer residual impairments in motor control, cognition, tremor in the limbs, spasticity, and weakened motor functions [4]. Keren’s research states that residual motor control can sometimes be improved, but may require years of “comprehensive and integrated rehabilitation” [4]. TBI recovery is a slow process, which may hinder or prolong the steps toward independent mobility.

Some resounding problems for persons sustaining a TBI are impairment of motor skills, visual processing difficulties, and poor attention to environment [4,5]. These residual deficits not only can make EPW driving difficult, but dangerous for the person with a TBI. Two types of residual deficits of a TBI impact mobility. The first group is movement disorders such as tremors, inadequate range of motion, spasticity, and weakness. These movement disorders hinder EPW driving and can render participation in purposeful activity unattainable. The second set of residual deficits hindering independent EPW driving is poor attention and visual neglect of surroundings. This second set of deficits seems to be most promising to be improved upon by using some type of “cued guidance” to better facilitate a person’s residual motor skill and control for EPW driving. To administer “cued guidance” for these deficits, a specialized assistive technology (AT) that rewards the person for attending the desired direction of travel is explored.

1.1 CUED GUIDANCE

Cued guidance is a treatment in which a therapist directs the attention of a person with a TBI to a particular head position or orientation. Motivating attention can be performed either by another person, or through specialized AT designed to cue a person for a desired response. Numerous researchers have shown that cued guidance can significantly improve the attention deficits and visual impairments of persons who have experienced a TBI [5,6,7,8,9]. The methods reported by the literature addressing “cued guidance” used either a human interface or a specialized AT device.

One study called “The Lighthouse Strategy” tested 16 people who had cerebral vascular accidents (i.e., strokes) who exhibited inattentiveness and substantial lateral neglect. In this study, Niemeier encouraged the participants to mimic a lighthouse, by standing up straight and “sweeping their heads back and forth” [5]. The visual imaging technique used as a stimulus yielded significant results, where the inattentiveness and visual neglect of the study participants had considerably improved.

Two more reviewed case studies with human-directed cued guidance had similar results. Evidence of improvement in the participant's lateral visual neglect was confirmed ultimately by a turn of the head. In Karnath's study [6], there were three persons with noted neglect of their left side. The method of resolving this neglect was to turn the body's trunk or head to the left to a measured degree. Schindler made similar observations with a study of five people who demonstrated visual neglect on their left sides [7]. Schindler's comparable solution was to have the participants turn left to some measured degree. These two studies are further affirmation that cued guidance, at least directed by a person, can produce significant improvement in processing neglected areas of residual vision due to a TBI.

The approach for cued guidance by Niemann demonstrated the effectiveness of a computer-assisted AT program to reestablish attentiveness in persons with moderate to severe TBI conditions [8]. Niemann tested 29 subjects, where they were randomly assigned to two groups. One of these groups of TBI participants was being tested for memory, and the other group for attention. The latter of these assessments revealed some promise, where a significant improvement in attentiveness compared to a control group was observed. This study imparts something, perhaps, more important than the mere rehabilitation advantage of cued guidance. It also shows that cuing does not necessarily need to be administered by another person, but can be sufficient when using an appropriate electronic device as assistive technology.

The analysis conducted by Kirsch provides one of the most convincing supports not only for cued guidance but also in support of cuing with an appropriate electronic device. Kirsch reviewed two case studies of persons with a TBI, where assistive technology (AT) was used to attain progress with cognitive processing [9]. In the first case study, a 19-year old man with a disoriented cognitive condition used an iPaq personal digital assistant (PDA) for a navigation task through a building [9]. There were “distinctively colored” circles attached to the walls throughout the building. Using wireless Internet, the PDA was updated with different colors to look for on the walls so that the subject could walk to them. In short, the subject was given cued guidance to navigate a building.

The effect on the average number of navigational errors was observed in this first study when using the PDA. It was shown that these errors significantly reduced when this specialized AT were utilized. Kirsch notes that the results were “remarkable,” given the subject previously had an “inability to learn navigational routes for a prior period of nearly 6 weeks of daily rehearsal” with therapist intervention [9].

Kirsch’s second case study was of a 71-year old woman whose cognitive performance had degraded due to a TBI. The subject’s task was setting an alarm clock. Also being cued by a PDA, a digital image of each step was communicated to accomplish setting the alarm clock. The average number of accumulated errors in the step to achieve setting the alarm clock was recorded, as well as the number of steps attempted. With the use of this AT, the subject was able to accomplish all the steps to set the alarm clock within the allotted time [9]. From this study, the data suggests that “repetitive, highly structured, and interactive cueing facilitated skill acquisition” [9].

The two case studies reviewed by Kirsch followed the same trend, where errors in performance decreased when using a specialized AT. Both case studies that Kirsch reviewed showed significant improvement when using AT to augment the subject’s cognitive processing [9]. Kirsch concluded that assistive technology “can facilitate functional performance and contribute to learning of specific adaptive skills” [9].

Therefore, cued guidance can promote learning and improve upon residual motor skills, visual impairment, and cognitive processing. The discussed research studies further show that cued guidance for a TBI does not necessarily need to be administered by another person. It can be accomplished just as well with a specialized AT. Given the appropriate AT device, a person with a TBI can "re-learn" to use their residual functions after their injury [9].

The mechanism for relearning is to provide a specialized AT for EPW driving. One such AT was developed by Brienza that incorporated a force feedback joystick and control algorithms for EPW obstacle avoidance [10]. Brienza stresses the importance of such AT because the inability to maneuver with an EPW "can lead to dissatisfaction with and/or abandonment of equipment" [10]. It was deduced that an AT system resident to an EPW "would assist in teaching necessary driving skills" [10].

Considering the supports for cued guidance both with human and machine direction, the research further suggest that there is potential for AT to teach EPW driving skills. Furthermore, this literature supports cued guidance as a potential therapeutic approach. The premise of this research is that persons with a TBI can use assistive technology to drive an EPW; thus conferring major steps toward realizing independent mobility.

An EPW equipped with a cuing system will accelerate the development strategy of using aggressive head movement to overcome inattentiveness and visual loss. Drivers with a TBI can be persistently cued using AT until they have "relearned" their remaining motor control in attaining independent mobility. This new AT would help the person with a TBI remember to turn their head towards the direction of travel asserted by a direction-sensing joystick. The intent of this methodology is for drivers with a TBI to improve their visual attentiveness and reduce the risk of collisions to acceptable levels.

2.0 PROBLEM STATEMENT

The purpose of the Head Position Monitor (HPM) is to provide a head-monitoring interface for persons with impaired visual, cognitive, and motor function. Primarily, the target population is for person's who have experienced a traumatic brain injury. However, its function has hopes to serve other specific populations with severe upper extremity impairment. The HPM is to be a reliable assistive technology, where it can provide safety for these populations when operating an EPW where it tracks head orientation to a reasonable degree of accuracy

To meet these objectives as an assistive technology, the mean resolution for the HPM should detect a person's head position within *0.30 inch*. Through the stages of development, a proof of concept for head tracking will be established for the HPM design. Following a proof of concept, characterization of accuracy will be determined by using a milling machine to dial precise coordinates and verify results from the HPM. The coordinates tested with the mill will be randomly generated to ensure impartiality.

A mean resolution within *three tenths of an inch* would accordingly show that the HPM follows both the vertical and horizontal movements of a person's head. In addition to this, it serves as a means not only for safety in wheelchair usage, but as a potential gateway for the population with a TBI to perform in areas typically unavailable to persons with severe disabilities.

3.0 DESIGN ANALYSIS AND APPROACH

A person sustaining a TBI typically encounters difficulties with attending, motor control, and visual processing that can hinder independent powered mobility [4,5]. Poor attending and visual processing can make it unsafe to operate an electric-powered wheelchair (EPW) without supervised "cuing." Dependence upon cued guidance by a person does not encourage independent mobility, and often exceeds the rehabilitation services available to the user. An alternative to cuing by a clinician is electronic monitoring of the user's head position that enforces head and hand congruence. By congruence, it is only meant that restrictions on driving would be enforced if the person's head were not aligned with the direction of travel. A block diagram for the realized *TBI System* is illustrated in Figure 1.

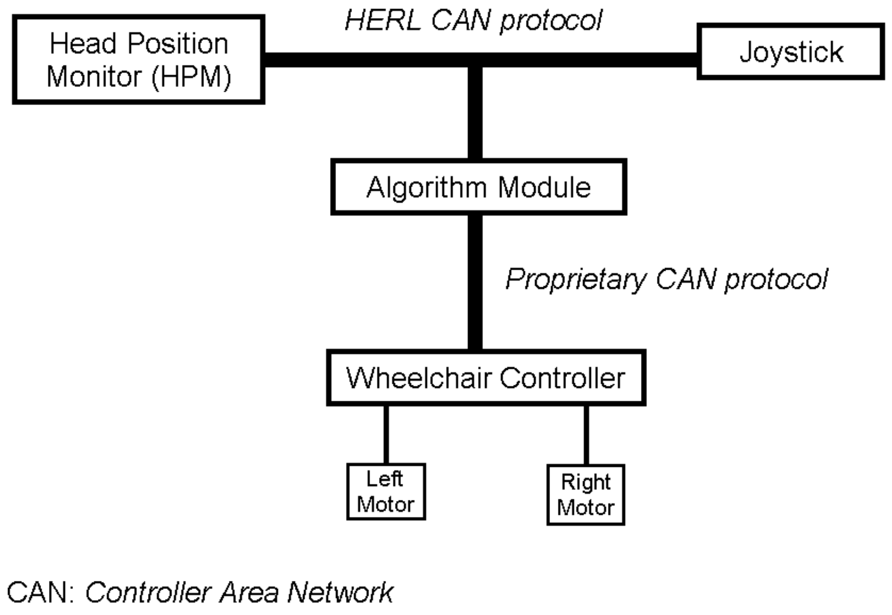


Figure 1: TBI System

3.1 TECHNOLOGY ALTERNATIVES

The ideal monitoring system for a person with a TBI would track the gaze of the user's eye. However, currently available eye-tracking systems are not suitable for mounting on an EPW. The alternative is to implement a non-stigmatizing head tracking system. Head-controlled assistive technology has already been explored for computer access with individuals sustaining a spinal cord injury [11]. Persons with disabilities often exhibited certain limitations with neck range of motion, neck strength, and fine motor control [11], but such AT was deemed suitable if the system was tailored to the user. This finding builds on existing confidence that the HPM will be a suitable interface to an EPW.

In examining the options to accurately track head position, various existing technologies are available. Technologies currently used to track human body movement include ultrasonic sensors, infrared sensors, gyroscopic measurements, and digital Hall effect sensors. Each technology has its benefits, but all have disadvantages that prevent them from being used in a practical HPM implementation. This can be seen in the following analysis where three main criteria were established for head tracking technology.

The first criterion of head tracking technology is for it to exhibit "absolute head positioning." Absolute head positioning in a tracking system does not "drift" during extensive monitoring periods. It is common for tracking systems that experience drift to often need to calibrate its sensors, or the entire system, to maintain some degree of accuracy. It is necessary to avoid any technology that experiences drift in head tracking technology for persons with a TBI. This is mostly due to the need for a reliable electronic cuing system for the users to use for extended periods without the system accumulating substantial error.

An important quality of head tracking technology, which is the second criterion, is to be a non-stigmatizing interface for the user. Stigmatizing characteristics can include mechanical linkages to the user's head, requiring a helmet being worn to operate the EPW, invasive connections to the user, or anything else that has a similar effect on the person's physical appearance.

The third and last criterion analyzed with the mentioned technologies is immunity to noise potentially inherent in different environments. Noise immunity is an obvious part of the criteria due to the fact the system is made for persons who are in need of constant and reliable cuing. To be reliable, the system must provide the most unfettered head tracking for maximal mobility to the user. Characterization of "noise" can be ambient light, electromagnetic interference, or any other potentially hindering feature of environments that could cause distortion depending on the type of technology used. These types of noise are what hinder some of the technologies mentioned in Table 1.

Table 1: Head Tracking Technology

Technologies	**Absolute Head Position**	**Non-Stigmatizing**	**Noise Immunity**
Infrared	Yes	Yes	No
Ultrasound	Yes	Yes	No
Digital Hall Effect Sensor Array	Yes	Yes	Yes
Gyroscope	No	Yes	Yes

Of the reviewed mentioned technologies, only the use of digital Hall effect sensors met all the necessary criteria. Examination of Table 1 shows how the other reviewed technologies do not meet the three main criteria. Both infrared and ultrasound technologies for head tracking do not have noise immunity. Infrared systems are often susceptible to interference from ambient-light in an environment [13], while ultrasonic systems can experience interference due to reflections of its own waves or other sound waves in the surroundings [14]. As for gyroscopic technologies, the tracking is not *absolute* where the accuracy typically degrades due to accumulated error.

At first glance, a digital Hall effect sensor array seems to be the best choice among other technologies. However, the cost of a head monitoring system using these digital sensors must be considered. A brief inductive analysis using a digital sensor array requires a sensor for each potential head position. Considering a limited resolution of $6.35\ mm^2$ (0.25 inch^2) in a region of about $457\ mm^2$ (12 x 6''), it would require more than 1000 sensors to realize each potential head position. Due to the impractical number of sensors in this prospective design, it should not be implemented.

To maintain all of the met criteria of the digital Hall effect sensor array, an *analog* sensor of this type is used instead of *digital*. Since an analog sensor provides a range of signals that varies with a magnet's proximity, it is possible to interpolate between a set of sensors. Given this characteristic of the analog Hall effect sensor, the HPM implementation is realized by placing a magnet on the rear of the user's head. The magnetic field created by the magnet enables the use of linear (or vector) algebra to determine the magnet's position with an array of linear analog Hall effect sensors (MLX90215). A small set ($n < 25$) of the *analog* sensors can then be used to observe the user's head position rather than the impractical number of *digital* sensors.

The next section will discuss the practical design of the head position monitor (HPM) using analog sensors, as opposed to digital.

3.2 HPM DESIGN APPROACH

Monitoring head position in persons with a TBI may provide a means for independent powered mobility. Given the often limited residual functions of attending, visual processing, and motor control, the operation of an electric-powered wheelchair must be constantly monitored to ensure the safety of these users. Thus, the HPM serves as a safety feature for this population. The solution proposed for this problem is placement of a magnet on the rear of the person's head. Strategically placed linear analog Hall effect sensors on a stationary headrest can then track the magnet; thus, accurately determining head positioning. Two different perspectives of a Solid Works computer-aiding draft (CAD) drawing of the HPM shell are shown in Figure 2.

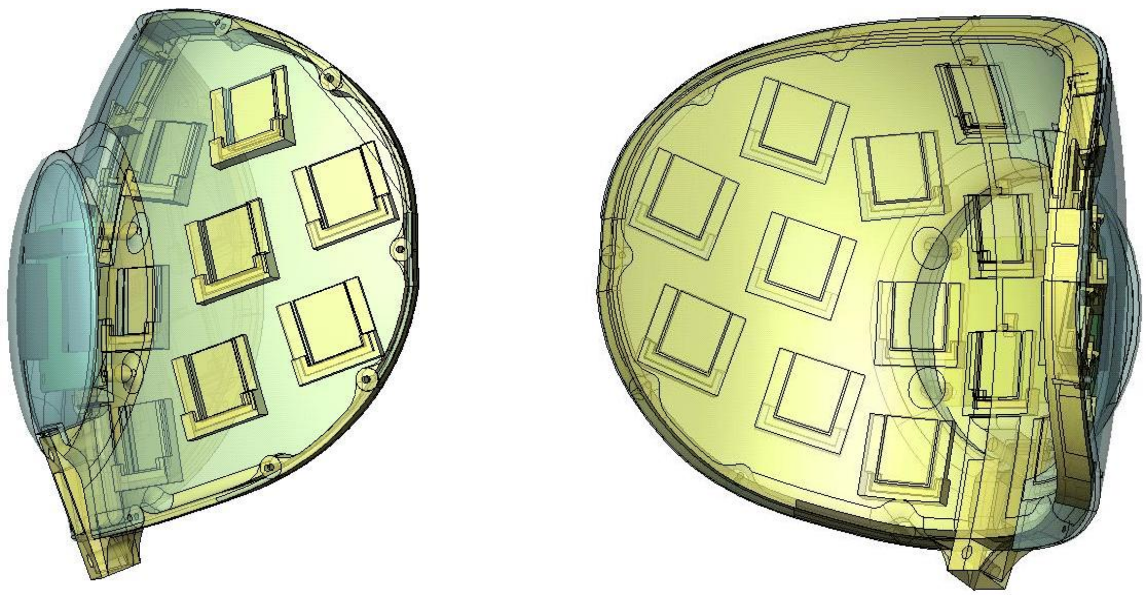


Figure 2: Head Position Monitor (HPM) Shell CAD drawing

The physical structure, as a *hemispherical* shell, of the headrest holding the sensors was previously constructed to follow an assumed contour head movement of a person with a TBI. From Figure 2, it can be seen that pockets are resident in the HPM shell for the linear analog Hall effect sensors. Given this prior information, the three-dimensional coordinates are known for the sensors; therefore, this establishes a configuration conducive to calculating a magnet's position through linear (or vector) algebra. This position of the magnet then implies the manner that the head is currently oriented. Given the necessary three-dimensions to realize the problem, the result of the head position is expressed as a vector of three components.

The engineering design of all components in the *TBI System*, as well as the communication protocols between them, will not be discussed in detail. The topic of this thesis is not the *TBI System* as a whole, but only verification of the HPM as a functional prototype. Other components of the *TBI System* will only be highlighted if a given explanation about an aspect of the HPM requires such clarification.

3.3 LINEAR ANALOG HALL EFFECT SENSOR

The linear analog Hall effect sensor used is made by Melexis. This analog sensor works similar to how the digital sensor works where it is triggered by the presence of a magnetic field. The main difference is that a digital sensor has some threshold for the strength of the magnetic field of which it triggers in a binary fashion. This is different from the analog sensor because its output is a gradient voltage in between the *power* and *ground* of the power supply.

When triggering a digital Hall effect sensor, a magnet must be present to create a magnetic field strong enough to surpass the threshold of the sensor. If the magnet is not close enough to the sensor, it will typically sustain an output voltage equal to *ground* of the power supply. Once the magnetic field from the magnet exceeds the threshold, the output voltage of the sensor then goes to its *high voltage* provided by the power supply. Given a digital sensor, it does not create any responses in between *ground* and its *high voltage*. With a *5 V* power supply, this digital (or binary) response is illustrated in Figure 3.

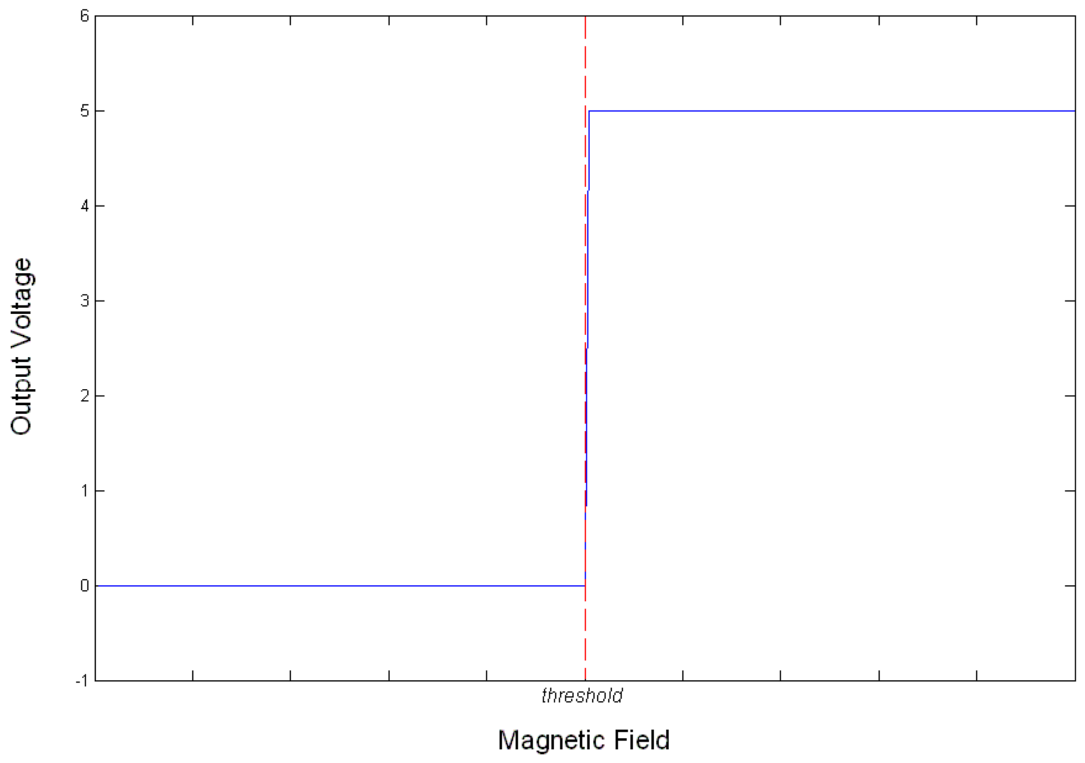


Figure 3: Digital Hall Effect Sensor Response

The response of the digital Hall effect sensor is different from its *analog* counterpart where the analog sensor's output voltage has a gradient response in between the *ground* and *high voltage* of the power supply. The specific analog Hall effect sensor used to implement the HPM is made by Melexis, known as the *MLX90215*. This analog sensor can be programmed with various attributes affecting the sensitivity to magnetic field, idle voltage (output voltage when magnetic field is not present or below threshold), and range of graded response for the output voltage [12]. An illustration of the *MLX90215* is shown in Figure 4 where the sensor is mounted on a printed circuit board (PCB).

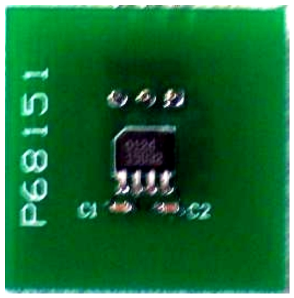


Figure 4: ***MLX90215*** **linear analog Hall effect sensor on a PCB**

Powered by a *5 V* power supply, the HPM implementation uses the *MLX90215* with an idle voltage of *2.5 V*, and a sensitivity of $100\,{}^{mV}\!/_{mT}$. This sensor has a versatile feature of triggering on both the North and South Poles of a magnet. When triggering on the North Pole of the magnet, the *MLX90215* gives an output voltage in between *2.5 V* and *5.0 V* with a linearly gradient response. In essence, the output voltage linearly increases to *5.0 V* as the strength of the magnetic field (due to the North Pole of the magnet) increases. This is typically when the magnet gets closer to the sensor. Intuitively, as the sensor gets farther away, the *MLX90215* output voltage will converge back to *2.5 V*. Figure 5 shows how the output voltage of the sensor corresponds to the strength of the magnetic field caused by the North Pole of a magnet.

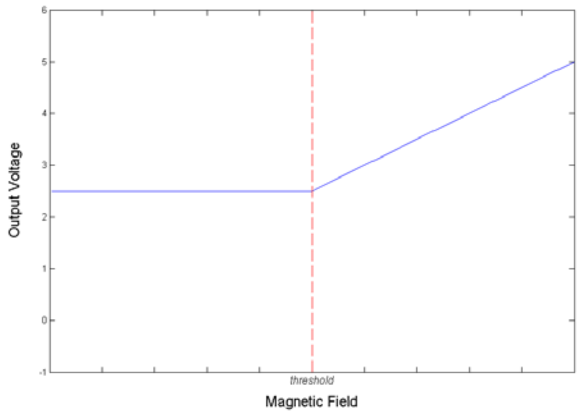


Figure 5: ***MLX90215*** **response to North Pole of magnet**

When the *MLX90215* is triggered with the South Pole of the magnet, it is similar in operation to the North Pole. The only difference being that it responds with a linearly gradient response in between *2.5 V* and *0 V*, rather than *2.5 V* and *5.0 V*. In this case the output voltage linearly *decreases* to *0 V* as the strength of the magnetic field (due to the South Pole of the magnet) increases. This occurs when the magnet gets closer to the sensor. Characteristically, the *MLX90215* output voltage will revert to *2.5 V* as the sensor gets farther away. The effect of a magnet's South Pole on the sensor is shown in Figure 6.

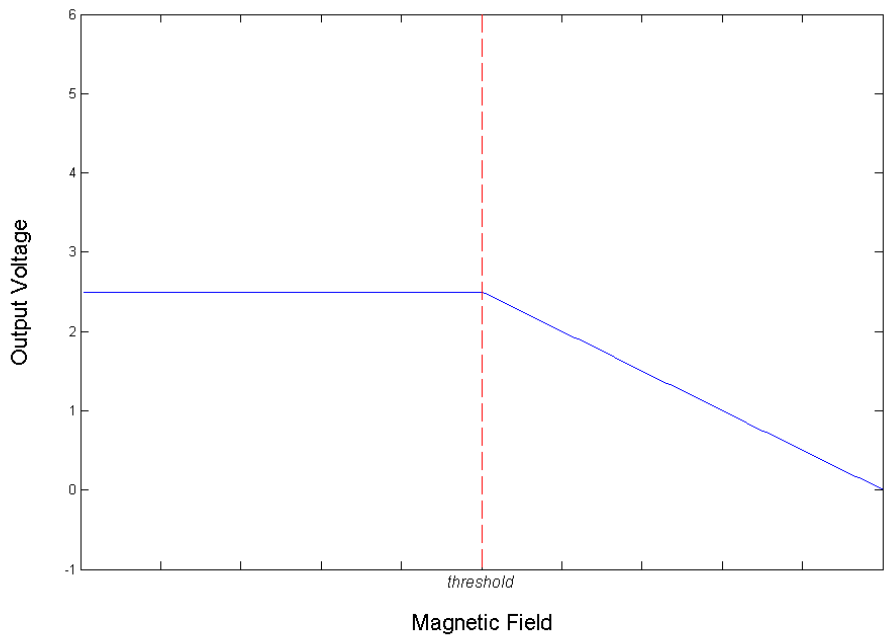


Figure 6: *MLX90215* response to South Pole of magnet

It can be seen that with the *MLX90215*, that a range of distances with respect to a magnet can be detected. It also follows that when using a set of these sensors with fixed coordinates, accurate positioning of a magnet can be determined. This premise is the basis for head position tracking in the *TBI System*.

3.3.1 Determining Absolute Position

In determining absolute head position, the HPM in the *TBI System* utilizes a *triangulation set*. A triangulation set is typically a group of three sensors that sense when a magnet is in their proximity. Given the gradient response from each sensor in the set, accurate positioning of the magnet can be determined with respect to the stationary sensors. Absolute position calculations were empirically determined by using the North Pole of a magnet and a *5 V* power supply. A photo of the magnet used for empirical testing is shown in Figure 7. The magnet is a rare earth $^5\!/_8$ *inch* Neodymium-Iron-Boron (NdFeB) magnet [15].

Figure 7: *NdFeB* Magnet Used for Empirical Testing

The triangulation set in the HPM is created by placing the analog Hall effect sensors equal distances from each other, as a result creating an equilateral triangle. To realize the largest distance at which sensors may be equally distanced, what has been termed the “falloff distance” ($D_{falloff}$) must be found. This distance is the measure where, beyond this point, the sensor can no longer detect a (North Pole) magnetic field. When a magnet is directly at a sensor, the output voltage is *5 V* assuming a *5 V* power supply. As the magnet moves away from the sensor, the output voltage linearly decreases to *2.5 V* as it approaches the falloff distance. Naturally, if the magnet exceeds a displacement from the sensor greater than the falloff distance, then its output voltage will remain idle at *2.5 V*. Empirically, it has been determined that a falloff distance ($D_{falloff}$) of 101.6 mm (4'') is characteristic of the sensor with the *NdFeB* magnet. The relationship between the analog Hall effect sensor's output voltage and the *NdFeB* magnet's relative displacement is illustrated in Figure 8.

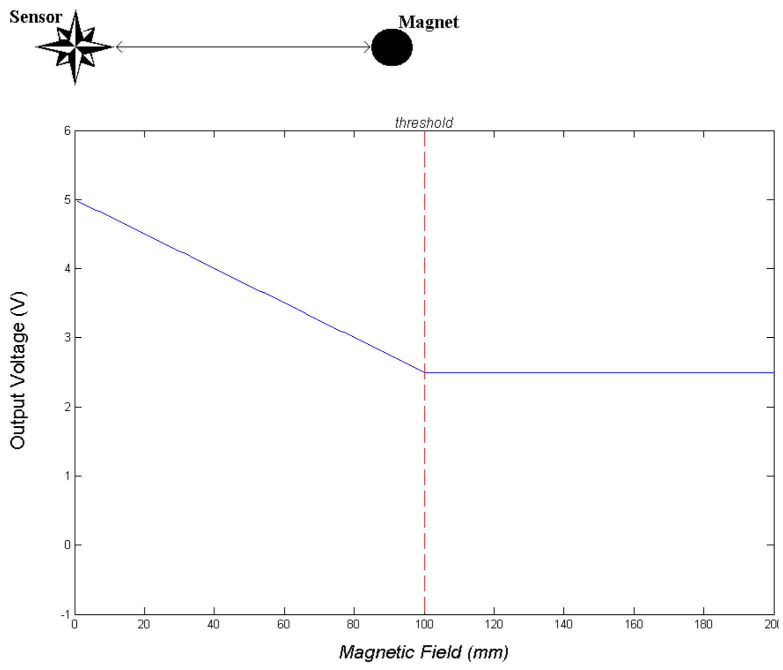


Figure 8: Magnet Displacement vs. Output Voltage

For triangulation to work properly, at least three analog Hall effect sensors need to detect the magnet within their range of graded response. In other words, the distance between the sensors in the triangulation set must be *less than* or *equal to* the falloff distance. Figure 9 shows an example of a triangulation set where the eight-point stars represent an analog Hall effect sensor and the black circle corresponds to the magnet used.

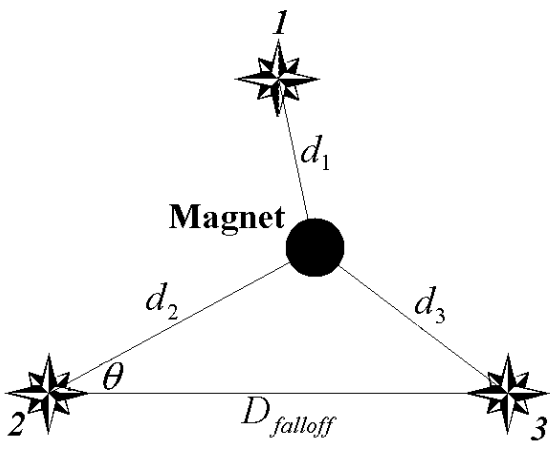


Figure 9: Magnet Triangulation

In reference to Figure 9, the distance between a given sensor and the magnet can be determined by taking the difference of *5 V* and a sensor's output voltage ($V_{i_{out}}$), then dividing it by the idle voltage of *2.5 V*. This quotient results in a fraction where the difference between one (1) and this quantity is multiplied by the falloff distance ($D_{falloff}$). This distance calculation for a given sensor is shown in Equation 1.

$$d_i = \left(1 - \frac{5 - V_{i_{out}}}{2.5}\right) \cdot D_{falloff}$$

Equation 1: Distance Calculation of Magnet from Sensor

Given the linear sensitivity of the sensors, the magnet distance with respect to sensors *1*, *2*, or *3* can be determined using Equation 1. Since the distance from a given sensor is known, only the angle with respect to a sensor needs to be found to associate a *direction* with the *magnitude* calculated in Equation 1. Using Figure 9 as a model, the angle between the horizontal of *sensor 2* and the distance d_2 can be found by using the *Law of Cosines* described as in Equation 2 [16].

$$(d_3)^2 = (d_2)^2 + (D_{falloff})^2 - (2)(d_2)(D_{falloff}) \cdot \cos\theta$$

Equation 2: Law of Cosines according to Figure 9

Algebraic manipulation of Equation 2 leads to Equation 3, which evaluates the angle *θ*.

$$\theta = \cos^{-1}\left[\frac{(d_3)^2 - (d_2)^2 - (D_{falloff})^2}{(-2)(d_2)(D_{falloff})}\right]$$

Equation 3: Angle *theta* corresponding to Figure 9

Although the distance from *sensor 1* to the magnet is not used in the position calculation, it is needed to verify positioning when dealing with multiple triangulation sets. Without this third sensor reference, it would be ambiguous whether the magnet was in the current triangulation set or one potentially adjacent to it.

The characteristics of the analog Hall effect sensor, as well as the relative position calculations, are a basis for the head position tracking of the HPM.

4.0 ELECTRICAL DESIGN WITH INTEGRATED CIRCUITS

The electrical design of the HPM is handled by integrated circuits (ICs). As a head-monitoring control interface, the design follows a number of steps to accomplish the task at hand. The sequential method of the hardware is the following:

- Sense presence of the North Pole of a magnet.
- Filter out spurious and erroneous signals.
- Channel signals for processing.
- Calculate "instantaneous centroid" of magnetic field.
- Communicate information about the position of the magnet.

Considering the method for operation, there were four types of ICs used to implement the HPM. These ICs are a set of linear analog Hall effect sensors, analog multiplexer, microcontroller, and protocol converter (for computer interfacing). The functional dependency, as well as overall system architecture, of these components will be discussed in detail in this chapter.

4.1 SET OF LINEAR ANALOG HALL EFFECT SENSORS

The HPM contains a set of 19 linear analog Hall effect sensors to track a magnet when in proximity of the concave surface of the HPM shell. This sensor is the *MLX90215* made by Melexis Microelectronic Integrated Systems; it is fabricated utilizing silicon-CMOS technology. The robust architecture of the *MLX90215* allows programming of slope and magnitude of sensitivity to magnetic flux density, as well as the sensitivity of the output voltage [12]. A functional diagram of the sensor's architecture is depicted in Figure 10 [12].

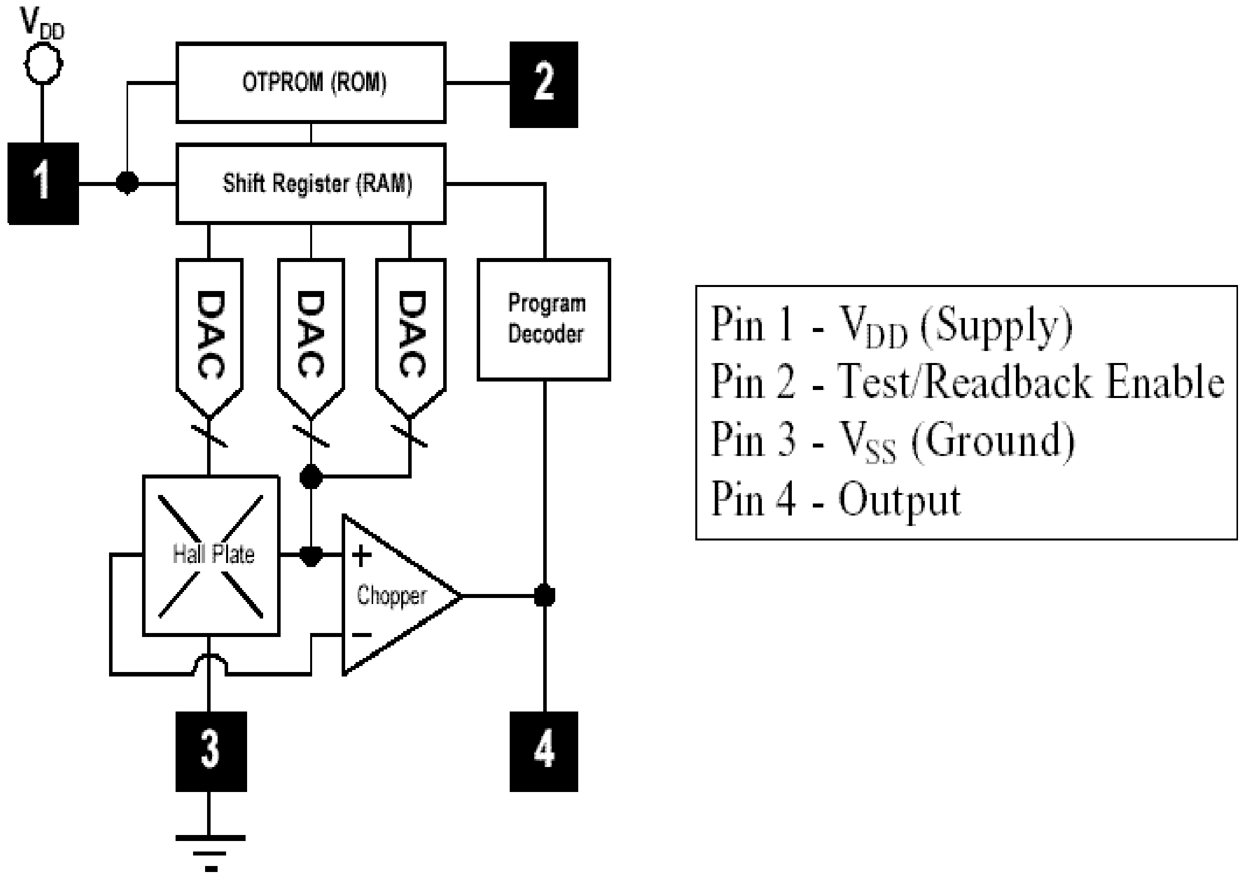
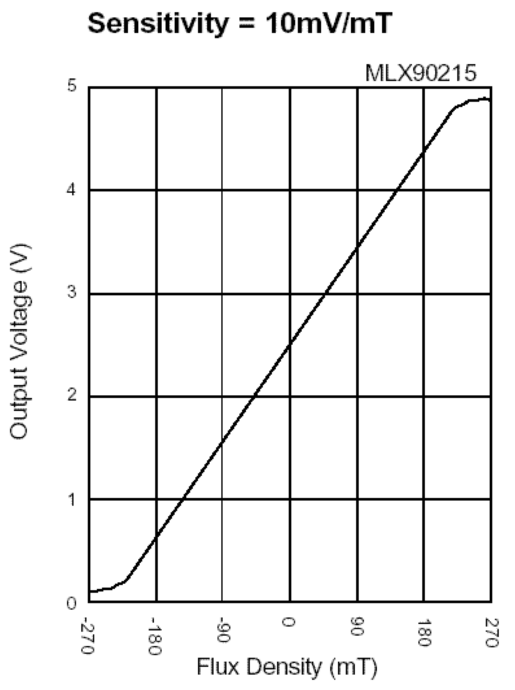


Figure 10: ***MLX90215*** **Functional Diagram**

With this architecture, it is possible to tailor the aspects desired for "linearity." What makes the *MLX90215* "linear" is that a significant interval of its function, output voltage (V) versus magnetic flux density (mT), follows a predictable slope. Figure 11 shows that the vast majority of the operating range of the sensor with sensitivities of $10\,{}^{mV}\!/_{mT}$ (on the left) and $140\,{}^{mV}\!/_{mT}$ (on the right) have linear responses [12].



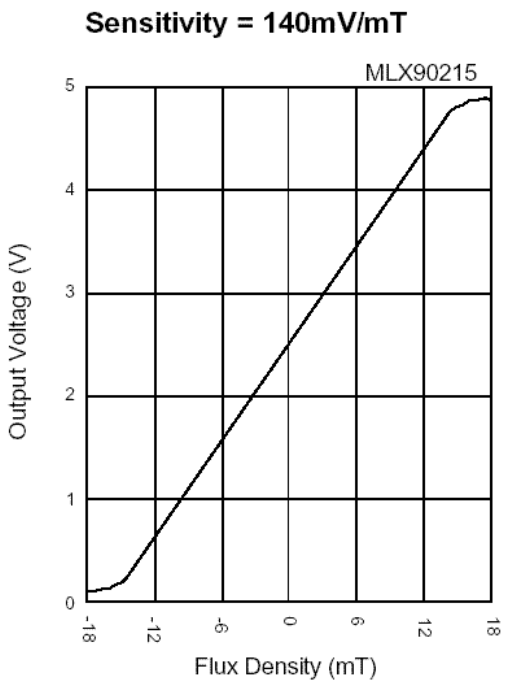


Figure 11: *MLX90215* Slope of Operating Range Graphs

The implementation of the HPM does not utilize either of the sensitivities rendered in Figure 11, but still conveys valuable information about the design. The sensitivity used in the HPM is $100\, {}^{mV}\!/_{mT}$, which is in between the tested measures conducted my Melexis. Therefore, this shows that the selected sensor sensitivity can be expected to perform as a linear function with respect to output voltage (V) versus magnetic flux density (mT). Given this linearity, this further provides for a linear gradient response with respect to the displacement of a magnet.

Realization of the *MLX90215* on a printed circuit board (PCB) has a relatively simple electrical configuration of two *2.5 nF* capacitors [12]. This is all that is required to operate the sensor; however, a small "indicator circuit" was added to convey if the sensor was working properly for debugging purposes. The indicator circuit only consists of a light-emitting diode (LED) in series with a resistor connected to ground. The entire circuit schematic for the sensor, pertaining to the HPM implementation, is depicted in Figure 12.

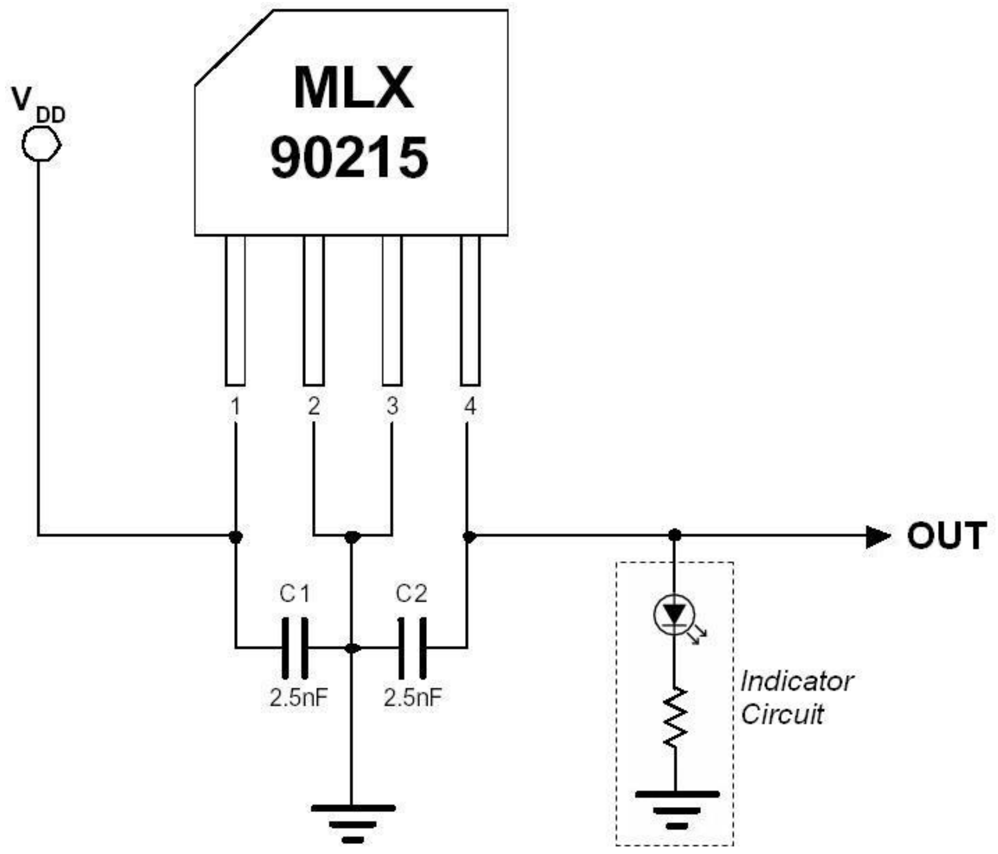


Figure 12: ***MLX90215*** **Circuit Schematic**

Using the set of 19 sensors, they each will independently respond to the magnetic field of a magnet in the concave proximity of the HPM shell. Accordingly, the LED of the *Indicator Circuit* of Figure 12 will increase its light intensity as the magnet approaches. Likewise, the light intensity will decrease as the magnet moves further away. The signals from each of the sensors must now be "channeled," which is mostly handled by the analog multiplexer.

4.2 ANALOG MULTIPLEXER

The role of the analog multiplexer is to facilitate sampling of the *MLX90215* Hall sensors. This IC is known as the *DG406* and is made by Maxim Integrated Products. The *DG406* is a 1 of 16 multiplexer and de-multiplexer [17]. The reason that it is also considered a “de-multiplexer” is due to its equal current conduction in both directions through the IC. The functional diagram of the *DG406* is illustrated in Figure 13 [17].

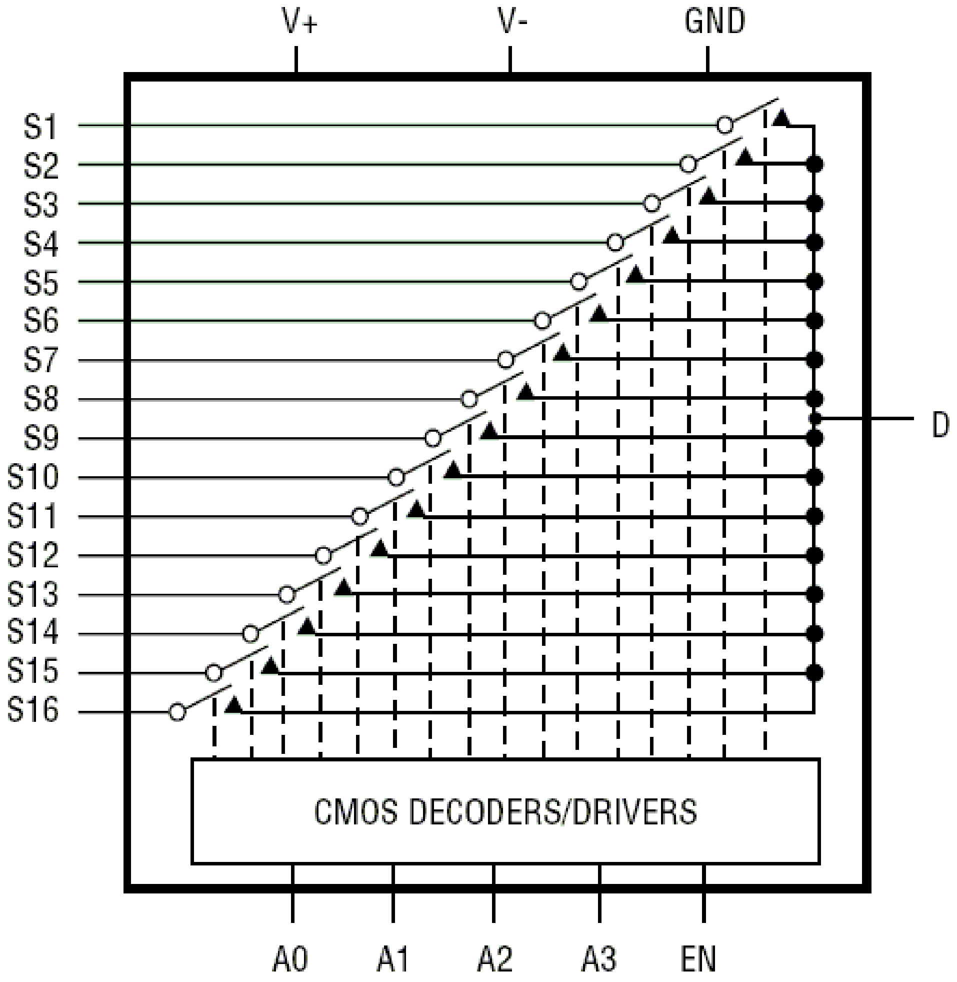
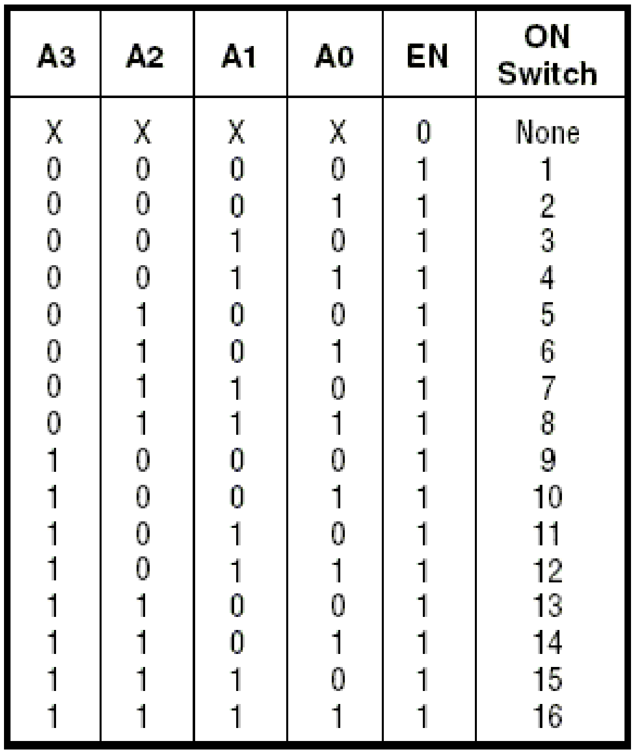


Figure 13: *DG406* Functional Diagram

For the HPM implementation, the output voltage of the *MLX90215* sensors will always be connected to the *S1 - S16* terminals of the *DG406*. As for all multiplexers, the conductive connection between any given *S* terminal and the *D* output is mutually exclusive to other *S* terminals. In essence, only one *S* terminal can be electrically connected to the *D* output terminal at any given time. The *S* terminal selected is determined by the electrical configuration of the address terminals *A0 – A3*. The truth table in Figure 14 [16] is for the *CMOS Decoders/Drivers* from Figure 13. This part of the IC's architecture determines which signals to pass through the analog multiplexer.



A3	A2	A1	A0	EN	ON Switch
X	X	X	X	0	None
0	0	0	0	1	1
0	0	0	1	1	2
0	0	1	0	1	3
0	0	1	1	1	4
0	1	0	0	1	5
0	1	0	1	1	6
0	1	1	0	1	7
0	1	1	1	1	8
1	0	0	0	1	9
1	0	0	1	1	10
1	0	1	0	1	11
1	0	1	1	1	12
1	1	0	0	1	13
1	1	0	1	1	14
1	1	1	0	1	15
1	1	1	1	1	16

Figure 14: *DG406* Truth Table

Assuming that the enable terminal (EN) of Figure 14 is a logical "1," the IC will perform as a standard analog multiplexer. Pertaining to the truth table in conjunction with the power supply of the HPM, a logical "0" and logical "1" refers to *0 V* and *5 V* respectively applied to the terminals of the *DG406*. The addressing and enable signals are supplied by the microcontroller to sample the sensors to process the instantaneous centroid calculation. Figure 15 has a partial schematic of the *DG406* analog multiplexer including connectivity to the microcontroller and the set of 19 sensors.

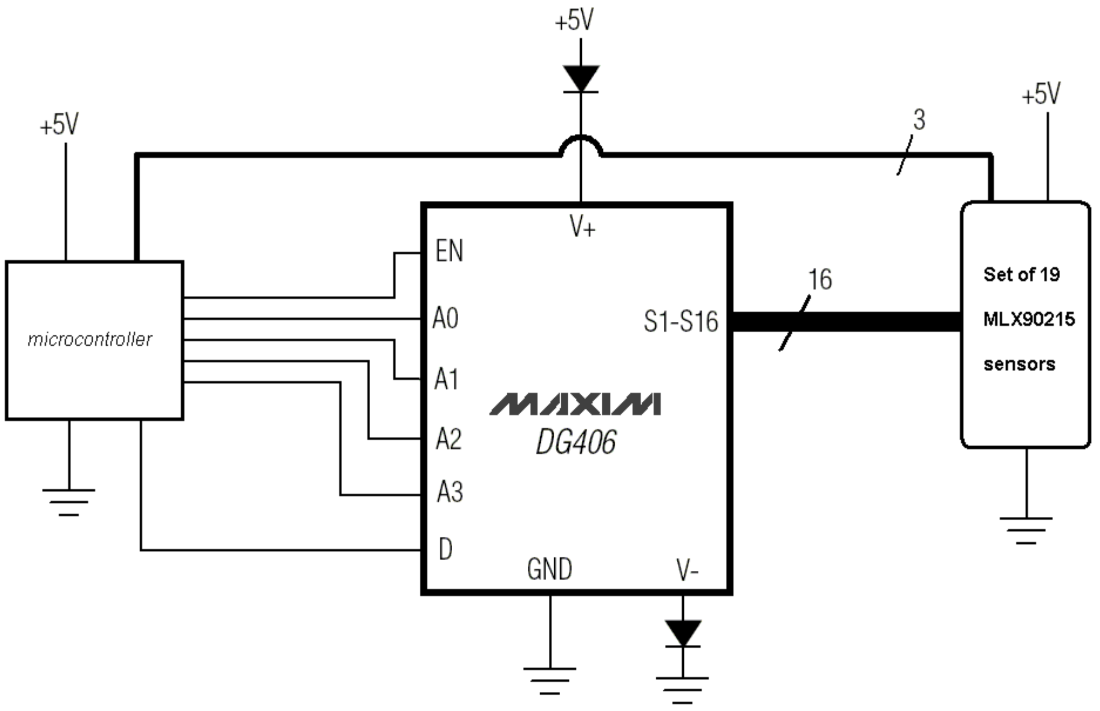


Figure 15: ***DG406*** **Schematic**

There is one more important thing to recognize about the analog multiplexer in the HPM configuration. This is only a "1 of 16" multiplexer; therefore, three more sensors of the set of 19 need to be accounted. Figure 15 depicts that the remaining three output voltages from the sensors are connected directly to the microcontroller. The reason for this is simply that 19 sensors were needed to track head position, but an analog multiplexer only capable of channeling 16 signals was available. The next section will detail the configuration of the microcontroller used.

4.3 MICROCONTROLLER

The processing necessary to calculate the instantaneous centroid of the magnetic field produced by the *NdFeB* magnet is handled by the *MC68HC908GZ16* microcontroller made my Motorola

Incorporated. This microcontroller is robust when it comes to supplementary features that facilitate some of the tasks of the HPM. The features that support the operations of the HPM will be detailed as of the role they carry in the system. In particular, the *MC68HC908GZ16* has an analog-to-digital converter (ADC) with resolution of 10 bits, ample general-purpose input/output (I/O) terminals, and a Serial Communications Interface (SCI) Module [18]. The pin layout of the microcontroller is shown in Figure 16 with minor reference descriptions.

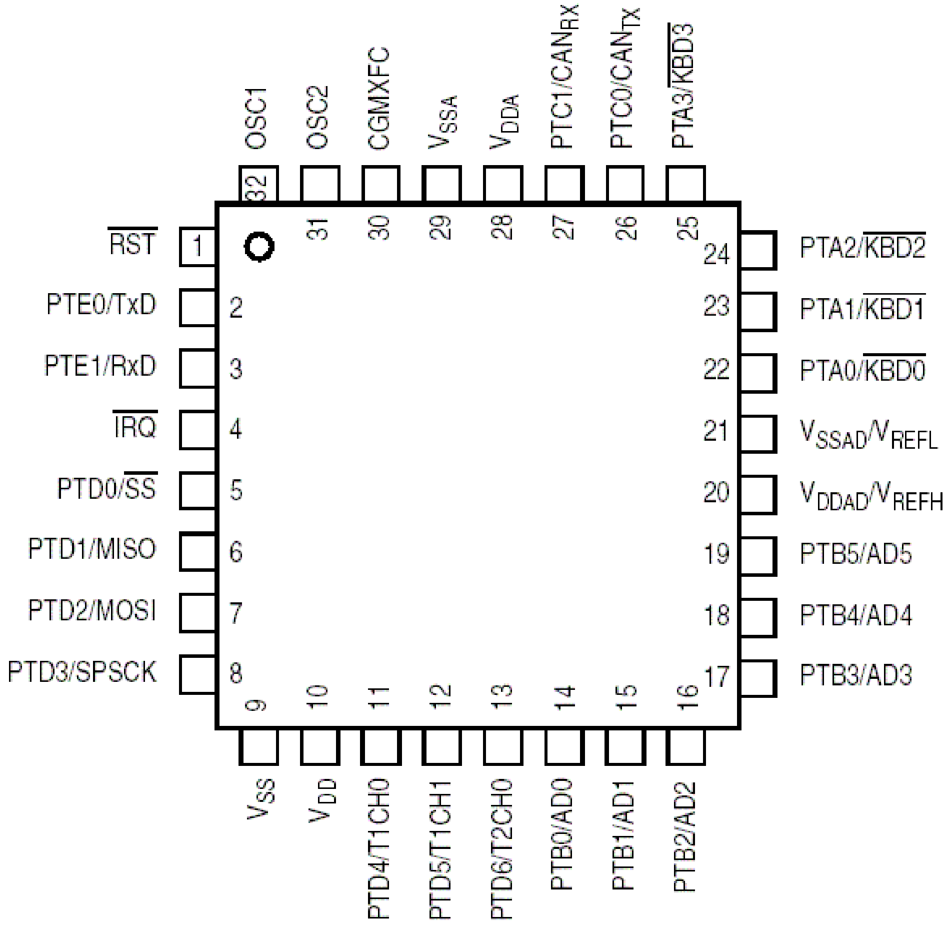


Figure 16: *MC68HC908GZ16* Pin Layout

The pins labeled *PTB/AD* (14 – 19) in Figure 16 are the pins that feed to the ADC. These pins are used to convert the analog signal from the *MLX90215* sensors to a 10-bit digital value.

PTD0 – PTD4 (pins: 5 – 8, 11) are used as I/O pins to command the *DG406* analog multiplexer to channel specific signals to the microcontroller. *PTD4* is connected to the enable terminal of the multiplexer, while *PTD0 – PTD3* are used for the address terminals (A0 – A3). The *TxD* and *RxD* shown in Figure 16 are the transmission and receive pins (2 & 3) of the SCI module.

4.4 PROTOCOL CONVERTER

The protocol converter (MAX3232) for the HPM is simply a voltage level-shifter that serves as an *RS-232* transceiver [19]. The *MAX3232* is also fabricated by Maxim Integrated Products, where it seamlessly interfaces with the SCI module of the *MC68HC908GZ16* microcontroller. The SCI is configured to transmit data in the framed format illustrated in Figure 17 [18].

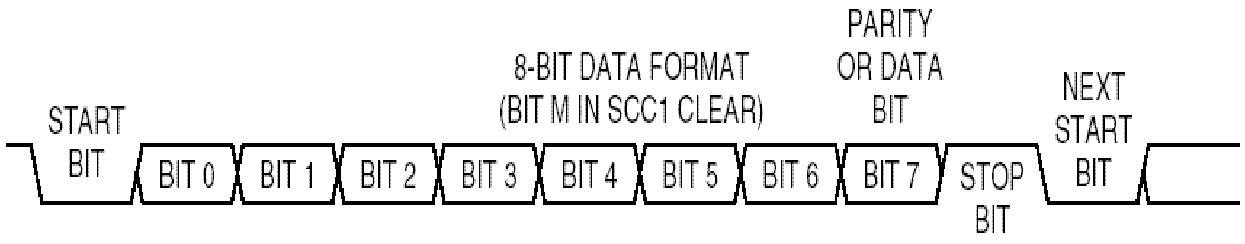


Figure 17: SCI/RS-232 Transmission Frame

The SCI module transmits using the standard non-return-to-zero format, which is synonymous to *RS-232* except for the voltage level of the bits transmitted. The HPM transmission parameters from the SCI module to *RS-232* is eight data bits, one stop bit, and no parity bits as shown in Figure 17.

The circuit configuration of the *MAX3232* is shown in Figure 18 [19].

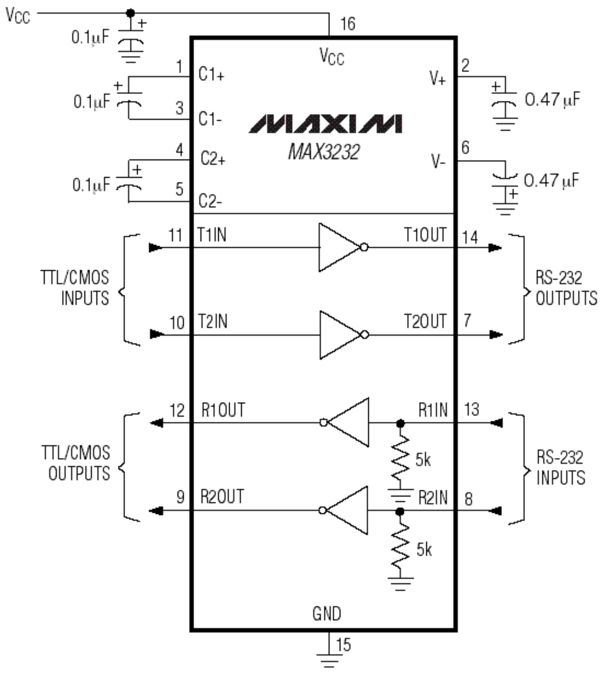


Figure 18: *MAX3232* Schematic

The *RS-232 INPUTS* and *OUTPUTS* in Figure 18 are typically connected to the appropriate terminals of a DB-9 connector. To see the *MAX3232* DB-9 connections, refer to Appendix A. The *MAX3232* and DB-9 serial connection complete the interface to a computer (PC) that has a serial communications port. The functionality of this part of the HPM is an essential part of the design for testing and verification of results.

4.5 COMPLETE HPM CIRCUITRY

The complete schematic for the integrated circuit design of the HPM is too detailed for a conventional figure to be shown. The full schematic, was drawn in Protel's Altium Design Explorer, and can be found in the Appendix A.

5.0 ASSEMBLY OF ELECTRICAL AND MECHANICAL COMPONENTS

The HPM is an assembly of electrical components on printed circuit board (PCB) with a mechanical housing to be integrated into a headrest. To clarify the intended practice for the HPM, a typical EPW is illustrated in Figure 19.

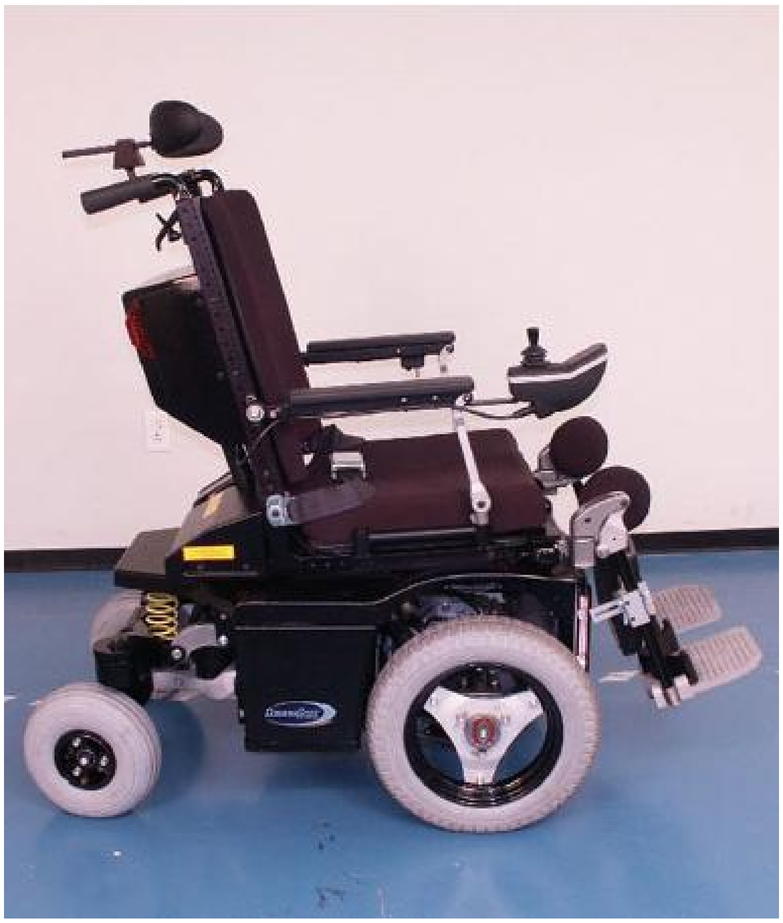


Figure 19: Typical EPW with Headrest

From Figure 19, we can see a headrest attached in the rear of the EPW where a person can relax their head. For persons with a TBI, the general role of the EPW's headrest is substituted by the HPM. This substitution is made on a different model EPW, and is shown in Figure 20.

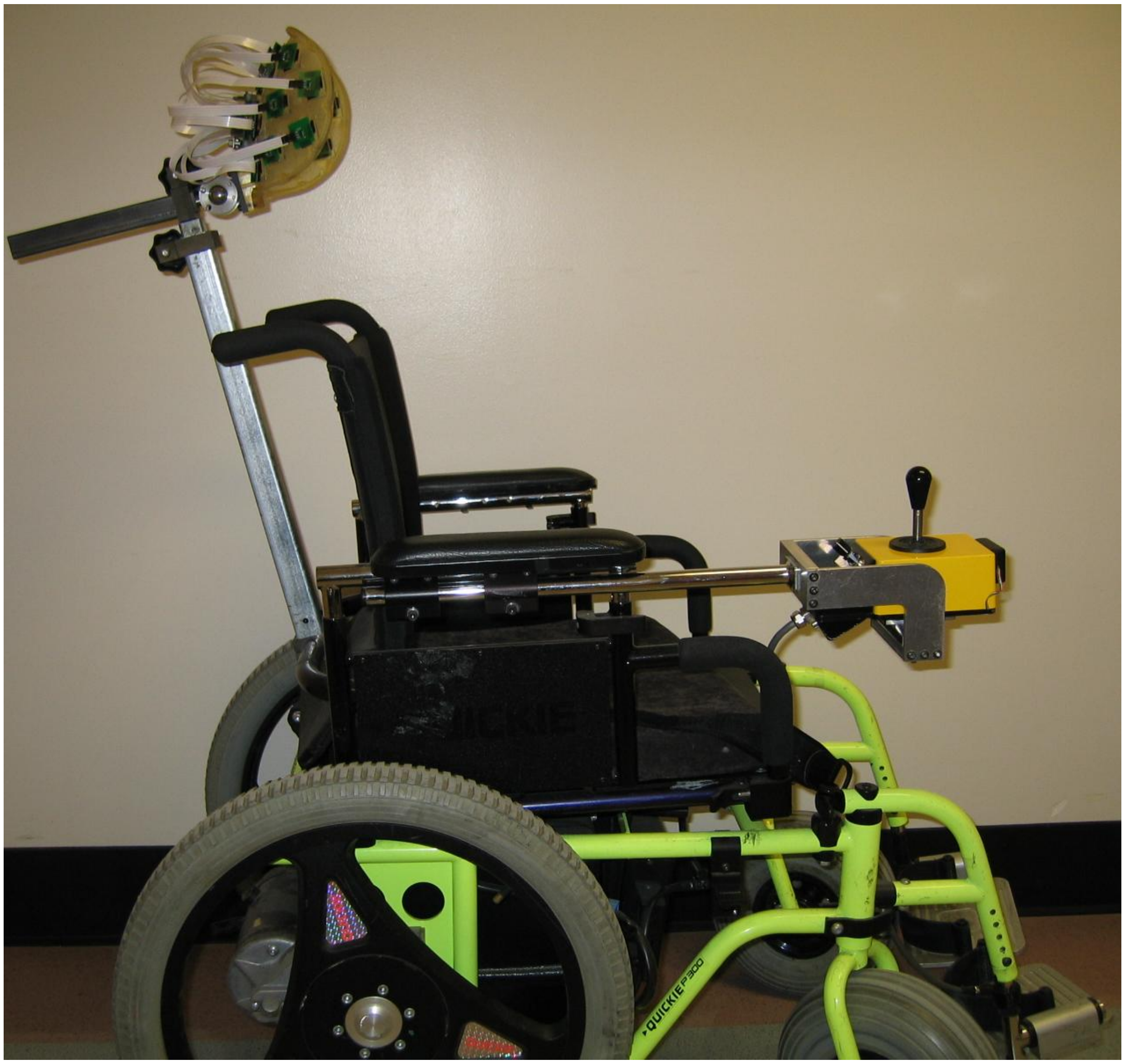


Figure 20: HPM attached to EPW

The HPM in Figure 20 is mounted in the rear of the EPW where the head would normally rest. Given the HPM is only a prototype, the circuitry and interconnections are exposed for testing and debugging, as well to better illustrate the assembly. A closer view of the HPM shell displaying the electronics and interconnections is depicted in Figure 21.

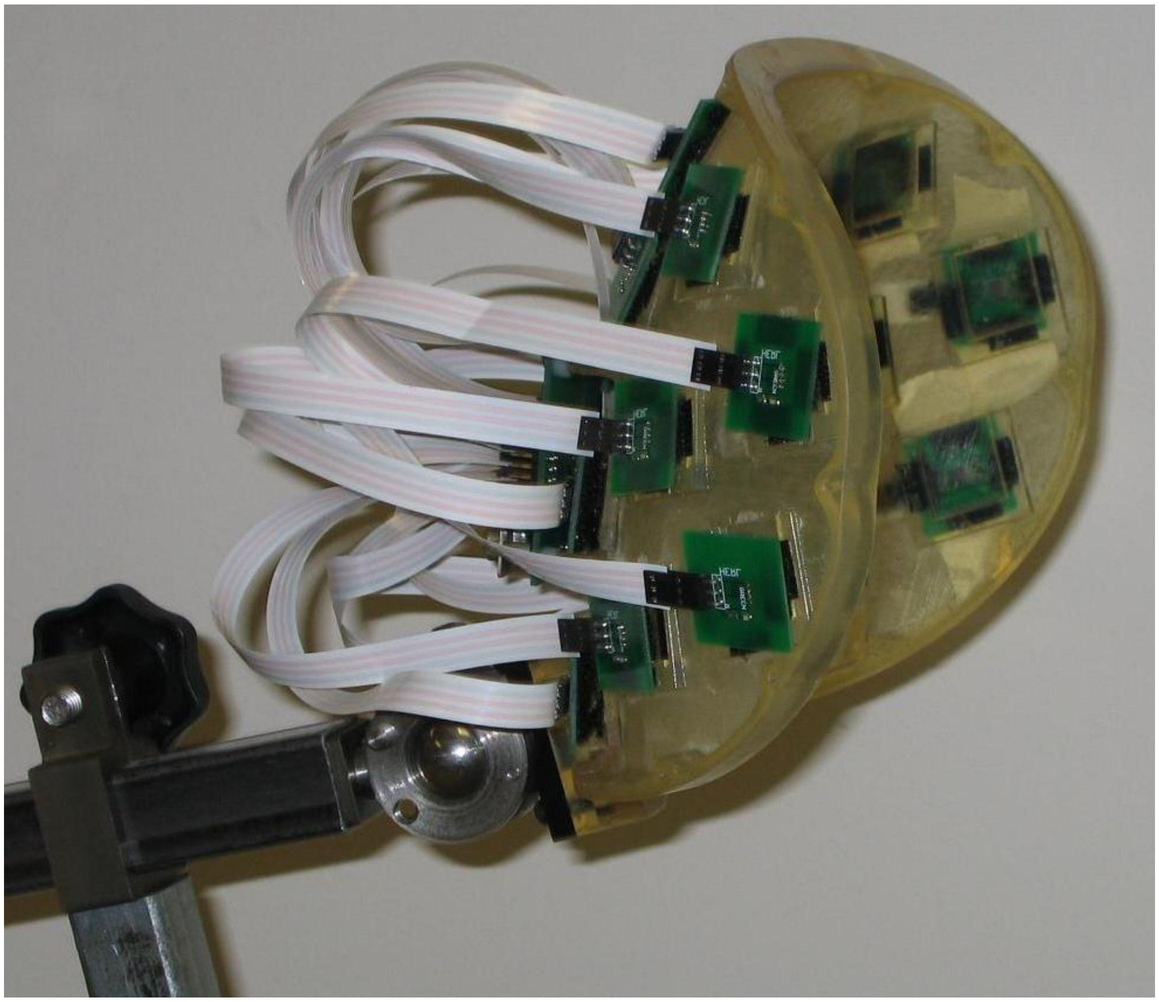


Figure 21: Close Up View of HPM Shell

The small square PCBs distributed around the shell are resident to the *MLX90215* linear analog Hall effect sensors. Each of the *MLX90215* PCBs are connected to a central PCB through a triple-conductor flex ribbon-cable. Given the current HPM is a prototype, further developments will shorten the length of cables and encapsulate the circuitry with a specialized cover. The central PCB of the HPM is shown in Figure 22 disconnected from the 19 *MLX90215* sensors.

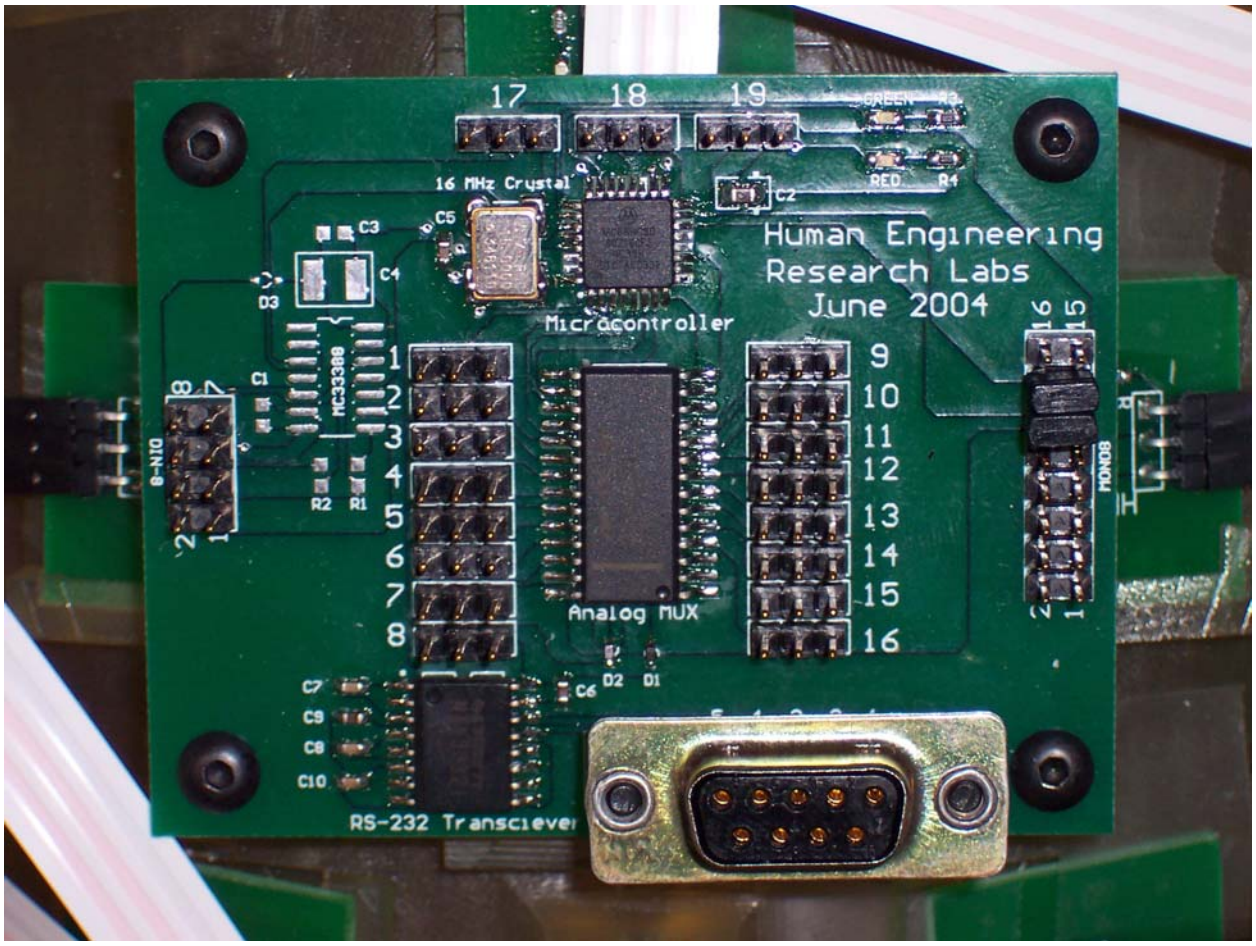


Figure 22: Central PCB on HPM Shell

The numbers *1 – 19* depicted on the central PCB is imprinted next to each three-pin header that electrically connects to each *MLX90215* PCB through the triple-conductor flex ribbon-cable. The center rectangular IC in Figure 22 is the *DG406* analog multiplexer, and is so labeled adjacent to its PCB mounting. The square IC at the top center of the central PCB is the *MC68HC908GZ16* microcontroller. The two components at the base of Figure 22 are the *MAX3232* voltage-level shifter (on the left) and a DB-9 connector (on the right). The DB-9 connector is only used to verify the output of the HPM by analyzing the data through the serial communications port of a PC.

6.0 DETERMINING COORDINATE POSITIONS OF MAGNET

As covered in chapter 3.0 of this work, the basis of finding the “instantaneous centroid” of the magnet is using a *triangulation set*. When a magnet comes in the proximity of the HPM shell’s concave surface, it intrinsically becomes resident to a triangulation set. Inspection of the HPM mechanical housing shows that there is actually a collection of these triangulation sets. It can be seen in Figure 23, viewing from the concave surface of the HPM shell, that each set is positioned in the shell with different orientations around the shell’s curve.

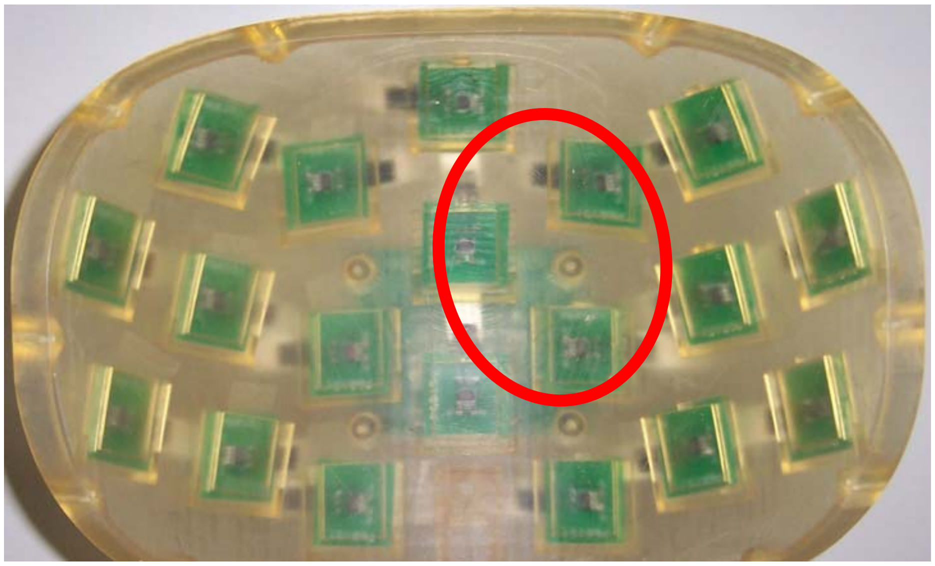


Figure 23: Sensor Placement in HPM Shell

An arbitrary triangulation set is encircled in *red* in Figure 23, where the centers of each *MLX90215* PCB in the set are equidistant from each other. This, in turn, forms a virtual three-dimensional equilateral triangle in the HPM.

6.1 STATIONARY SENSOR COORDINATES OF HPM

Before proceeding to calculate the precise coordinates of a magnet, the sensor coordinates associated with the respective triangulation set in the HPM must first be determined. These sensors are typically those that detect the strongest magnetic field. The sensor's coordinates are already known, since they are defined in the Solid Works CAD software that was used to design the HPM shell's physical characteristics.

For the purpose of linear algebra calculations, the HPM is treated as a *subspace*, where the origin is a reference point in space in front of the shell where it is assumed that a person's axis of rotation resides. The subspace of the HPM shell is shown with its coordinate vectors in Figure 24 in a profile view. The circles in the graph are the locations of the sensors, and they have a solid line connecting them with the origin of the subspace.

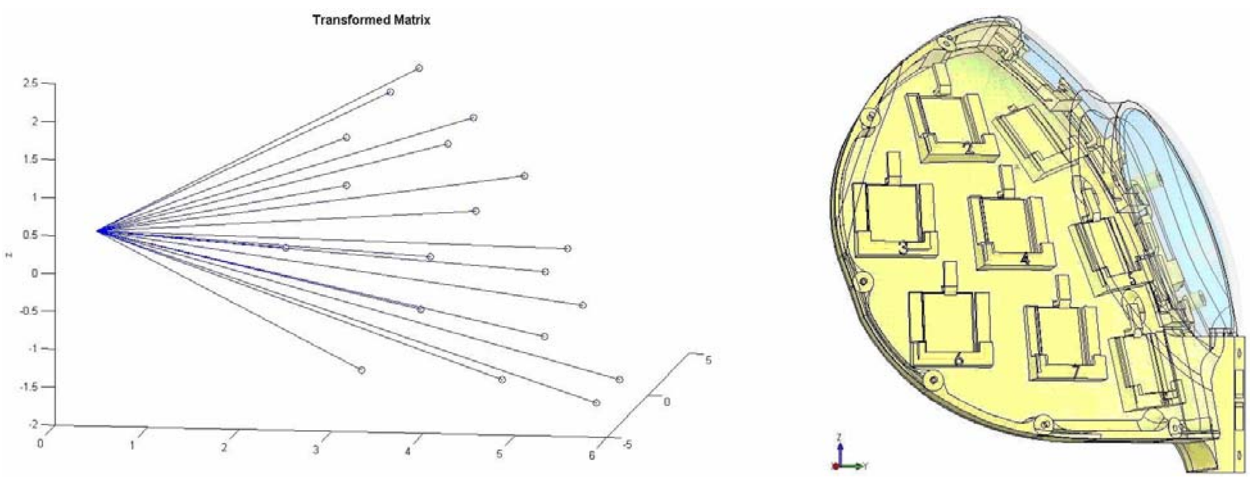


Figure 24: HPM Subspace Profile View

The sensor vectors in the HPM subspace were plotted in MATLAB to emphasize the mathematical relationship of the sensors. The horizontal, depth, and vertical aspects of the HPM shell are realized by the three-dimensional *x*, *y*, and *z* axes respectively. To see the graphical curve of the HPM shell, Figure 25 shows a top-level view of the subspace aligned with the *z* axis. Therefore, the HPM shell is viewed only in terms of *y* and *z*. The origin of the HPM subspace can be seen to be isolated in space in front of the physical shell.

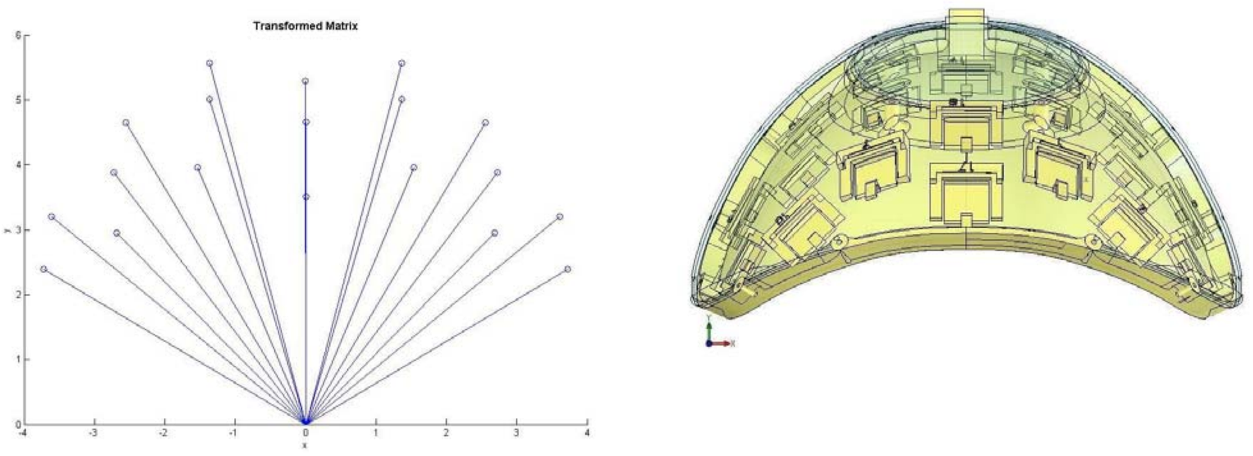


Figure 25: HPM Subspace Top View

In the design specification of the HPM shell, it was established that adjacent sensors would form equilateral triangles in three-dimensional space. This meaning that there is a two-dimensional plane that is common to the sensors within a triangulation set. This physical specification was implemented to simplify calculations of vectors using linear algebra.

A list of the sensor coordinates is in Table 2. The columns for *x* (horizontal), *y* (depth), and *z* (vertical) represent the three dimensions in inches, while *θ* and *Φ* represent the spherical coordinates in degrees. The spherical coordinates are only shown to give some reference as to the number of degrees between any two given sensors in either the horizontal (*θ*) or vertical (*Φ*) directions. This becomes important when determining whether a person's head lies within a certain degree of accuracy when cued for a particular head response. As illustrated in Figure 24 and Figure 25, the origin of the HPM subspace resides in space in front of the physical shell.

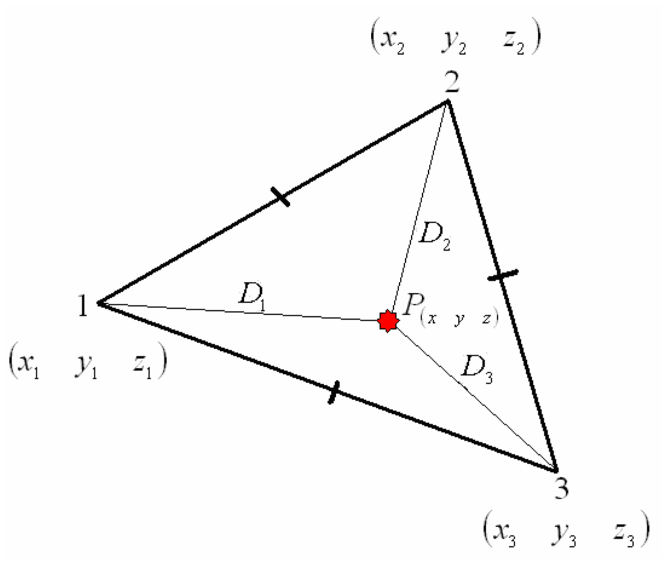
Table 2: Sensor Coordinates (Cartesian and Spherical)

Sensor	x	y	z	θ	Φ
1	1.5413	3.9609	1.4346	68.7377	18.6509
2	2.6835	2.9549	1.6179	47.7555	22.0638
3	3.7198	2.3870	0.2601	32.6884	3.3677
4	2.7288	3.8780	0.0733	54.8672	0.8851
5	1.3624	5.0058	-0.2443	74.7752	-2.6957
6	3.6127	3.2039	-1.3425	41.5678	-15.5369
7	2.5512	4.6432	-1.5463	61.2135	-16.2706
8	1.3643	5.5625	-1.9560	76.2197	-18.8563
9	-1.5413	3.9609	1.4346	111.2623	18.6509
10	-2.6835	2.9549	1.6179	132.2445	22.0638
11	-3.7198	2.3870	0.2601	147.3116	3.3677
12	-2.7288	3.8780	0.0733	125.1328	0.8851
13	-1.3624	5.0058	-0.2443	105.2248	-2.6957
14	-3.6127	3.2039	-1.3425	138.4322	-15.5369
15	-2.5512	4.6432	-1.5463	118.7865	-16.2706
16	-1.3643	5.5625	-1.9560	103.7803	-18.8563
17	0.0000	3.5106	2.2480	90.0000	32.6331
18	0.0000	4.6541	0.8579	90.0000	10.4443
19	-0.0097	5.2937	-0.8246	90.1045	-8.8539

The program code used to generate all MATLAB plots and tables can be found in Appendices B.2 and B.3.

6.2 CALCULATING COORDINATES OF A POINT INSIDE A TRIANGLE

To determine precise coordinates of the magnet within a triangulation set, this problem is solved by the methodology of finding a point inside a triangle. Although the sensors are in a subspace of three dimensions, sensors of the same triangulation set share a common plane. When a magnet enters the proximity of the HPM shell, it has a projection on the plane of a triangulation set. Thus, this reduces the complexity from a three-dimensional to a two-dimensional problem. Solving for the projection of a magnet on the plane of a triangulation set is shown in Figure 26. One thing to note is that all quantities in this problem are known, except for coordinates in the plane of the triangle.



$(x_2 \quad y_2 \quad z_2)$
2
D_2
1
D_1
$P_{(x \quad y \quad z)}$
$(x_1 \quad y_1 \quad z_1)$
D_3
3
$(x_3 \quad y_3 \quad z_3)$

Figure 26: Point Inside Triangle

Figure 26 represents the typical problem of a three-dimensional triangle that has an unresolved (magnet) coordinate in the HPM. When a magnet enters the proximity of the HPM shell, the magnet becomes resident to some triangulation set. Accordingly, the magnet’s projection then becomes a point with coordinates within the plane of a triangle whose vertices are the exact coordinates of the sensors making up the triangle. To determine the solution, an arithmetic process utilizing the theorems established in linear algebra yields a vector of three components.

The red star illustrated in Figure 26 represents the magnet whose position is to be determined. Vertices *1, 2,* and *3* of the triangle have the exact vector coordinates as the predefined placement of the *MLX90215* PCBs around the curve of the HPM shell. To begin solving this problem mathematically, a perpendicular is extended to one of the sides of the triangle. From this, we get two more quantities that arise in solving for the magnet's coordinates. This process is shown in Figure 27.

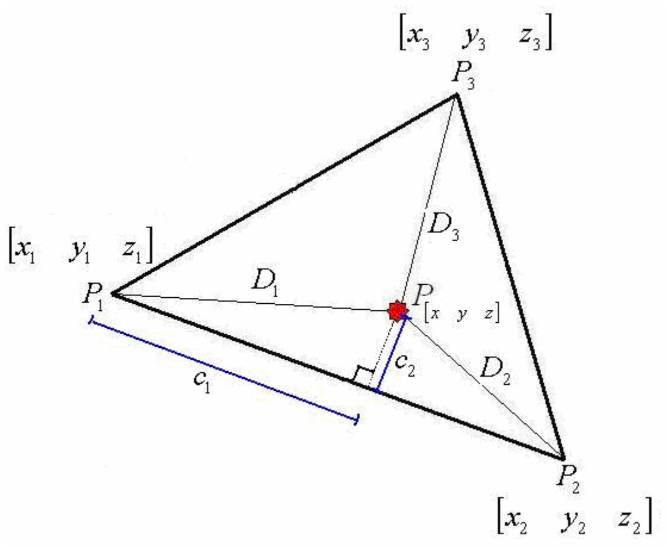

$[x_3 \quad y_3 \quad z_3]$
P_3
D_3
$[x_1 \quad y_1 \quad z_1]$
P_1
D_1
P
$[x \quad y \quad z]$
c_2
D_2
c_1
P_2
$[x_2 \quad y_2 \quad z_2]$

Figure 27: Point Inside Triangle with Extended Perpendicular

The reason we have extended this perpendicular as shown in Figure 27 is to exploit the *Parallelogram Rule* for addition of vectors [20]. This rule gives a relationship between $\underline{P}_1$ and our point of interest $\underline{P}$ that is described in Equation 4.

$$\underline{P} = \underline{P}_1 + c_1\underline{u}_1 + c_2\underline{u}_2$$

Equation 4: Vector Relationship between *P* and *P1*

The vectors $\underline{u}_1$ and $\underline{u}_2$ are the unit vectors of the triangle corresponding to $\overrightarrow{P_1P_2}$ and $\underline{u}_1$'s orthogonal vector contingent to the plane of the triangle respectively. The quantities c_1 and c_2 are the coefficients of the respective unit vectors.

6.2.1 Unit Vector Calculations

A unit vector is simply a vector with a magnitude equal to one (*1*). This is equivalent to taking a vector and dividing it by its own magnitude where: $\underline{u}_i = \frac{\underline{v}}{\|\underline{v}\|}$. The same methodology follows when determining the first unit vector $\underline{u}_1$ as expressed in Equation 5.

$$\underline{u}_1 = \frac{\overrightarrow{P_1P_2}}{s_{1-2}}$$

Equation 5: Unit Vector 1

The $\overrightarrow{P_1P_2}$ notation can be substituted with the calculation: $[(x_2 - x_1) \quad (y_2 - y_1) \quad (z_2 - z_1)]$. This is the difference in two vectors, where each component of one vector is subtracted from the other. It is evident from Figure 27 that the unit vector $\underline{u}_1$ along $\overrightarrow{P_1P_2}$ has a magnitude (or length) of simply a side of the triangle. The side of the triangle is denoted by *s*, and is typically the same in all calculations given the HPM shell was designed to have nearly equilateral triangles. Furthermore, s_{1-2} is identical to the magnitude of the vector from $\overrightarrow{P_1P_2}$ evaluated as the square-root of the difference of the points: $\left\|\overrightarrow{P_1P_2}\right\| = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2 - (z_2 - z_1)^2}$.

The second unit vector $\underline{u}_2$ is a bit more involved to determine. Given it must be orthogonal to $\underline{u}_1$ [20], half the vector $\overrightarrow{P_1P_2}$ is taken from $\overrightarrow{P_1P_3}$, then divided by its magnitude as according to Figure 27. The calculable result of $\underline{u}_1$ is mathematically conveyed in Equation 6.

$$\underline{u}_2 = \frac{\overrightarrow{P_1P_3} - \frac{1}{2}\overrightarrow{P_1P_2}}{\frac{\sqrt{3}}{2} s_{1-3}}$$

Equation 6: Unit Vector 2

From Equation 6, one can deduce that $\frac{\sqrt{3}}{2} s_{1-3}$ was used as the magnitude of the second unit vector $\underline{u}_2$, given this is necessary to provide a vector with length of one (1). For $\underline{u}_2$ to be orthogonal to $\underline{u}_1$, the following steps were taken:

1. Extend a perpendicular from $\underline{P}_3$ onto $\overrightarrow{P_1P_2}$. This typically splits the triangle in half as shown in Figure 28; thus performing the sub-operation of the *numerator* in Equation 6.

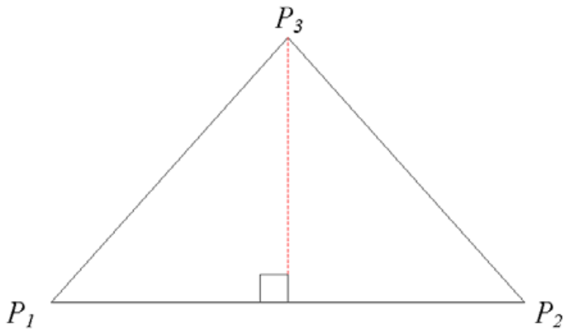


Figure 28: [*P*$_3$] perpendicular onto [*P*$_1$*-to-P*$_2$]

2. It is known that the angle at $\underline{P}_1$ is 60°; therefore, the side of the left *right triangle* opposite to $\underline{P}_1$'s angle corresponds to $\sin(60°)$ in Figure 28. Accordingly, the unit vector magnitude (or length) is shown in Equation 7.

$$\|\underline{u}_2\| = \sin(60°) \cdot s_{1-3} = \boxed{\sqrt{3}/2 \cdot s_{1-3}}$$

Equation 7: Magnitude of Unit Vector 2

From the preceding calculations, the unit vectors for any given triangulation set can be determined. The unit vectors give a general direction in a triangle to determine the coordinates of the magnet position. To lock in on the distance away from the initial point $\underline{P}_1$ where the magnet position lies, the coefficients of the unit vectors are needed for some indication of the length along these vectors [20].

6.2.2 Unit Vector Coefficient Calculations

The calculations of the unit vector coefficients can be found by applying the *Pythagorean Theorem*, and then using algebraic manipulation. In Figure 27, we find that a perpendicular was used to denote the significance of the unit vector coefficients c_1 and c_2. When determining these coefficients, this calculation involves a smaller triangle that consists of two adjacent right triangles. This smaller triangle is illustrated in Figure 29.

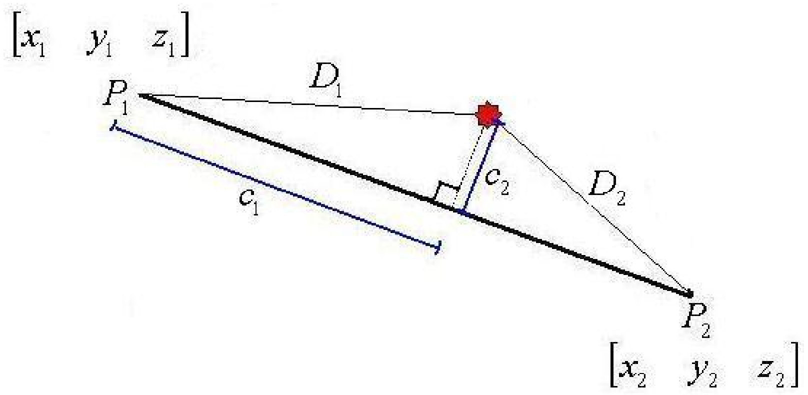


Figure 29: Unit Vector Coefficient Triangle

The Pythagorean Theorem can be utilized for both right triangles in Figure 29. From this, we have the preliminary precepts for the unit vector coefficients c_1 and c_2. The first precept to build upon is for the hypotenuse D_1, where: $(D_1)^2 = (c_1)^2 + (c_2)^2$. The second precept for the unit vector coefficients is for the other hypotenuse D_2, where: $(D_2)^2 = (s_{1-2} - c_1)^2 + (c_2)^2$. With these two precepts, c_1 and c_2 are constructed in Equation 8 and Equation 9.

$$c_1 = \frac{(s_{1-2})^2 + (D_1)^2 - (D_2)^2}{2 \cdot s_{1-2}}$$

Equation 8: Unit Vector 1 Coefficient

$$c_2 = \sqrt{(D_1)^2 + (c_1)^2}$$

Equation 9: Unit Vector 2 Coefficient

By inspection of Equation 8 and Equation 9, it is apparent that the calculation of c_2 is dependent upon c_1. Therefore, c_1 must be calculated prior to c_2. The remaining unknowns are the quantities D_1 and D_2. This needs to be found to complete the calculation of the point coordinates inside a triangle.

6.2.3 Relative Distance Calculation from Sensor to Magnet

Through empirical testing with the present HPM shell, *1.8 inches* was estimated as the distance where the magnetic field drops below noise level with respect to the given sensor. This is while using the rare earth $5/8$ *inch* Neodymium-Iron-Boron (NdFeB) magnet. As reported in section

3.3.1, a linear distance of about *4 inches* was found to be characteristic for the current sensitivity configuration of the *MLX90215* sensor and NdFeB magnet. However, given the curvature of the HPM shell, the magnet does not necessarily enter the proximity of a triangulation set "straight on," or orthogonal to the Hall sensor. Most of the time, the *MLX90215* sensor will sense the magnet at angles less than 90°.

It should be noted that the signal strength from a Hall sensor is best when the magnetic field is nearly orthogonal to the sensor plate. In Figure 30, the scenario of a magnet being orthogonal to the Hall sensor's plate is depicted.

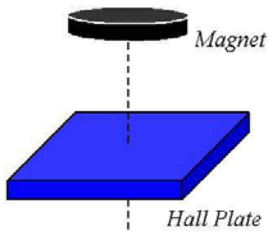


Figure 30: Hall Plate Best Response

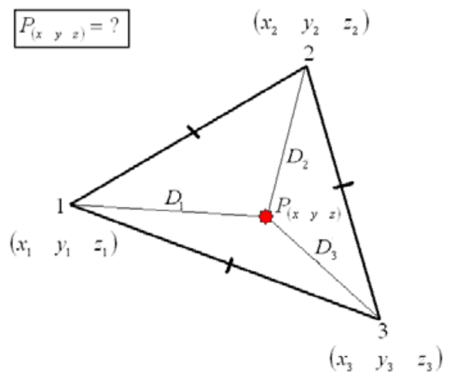
Intuitively, *non-orthogonal* penetration of the magnetic field yields signal strength detractive from the sensor's optimal response. This means that a magnet relatively far away, yet orthogonal to the Hall plate, will induce a weak response as if the magnet was relatively close, yet at an acute angle to the Hall plate's normal. This sensitivity to the magnetic field's penetration angle makes determining the distance between a given sensor and a magnet in a triangulation set more involved than the methodology set forth in Equation 1 of section 3.3.1.

There are two ways to determine how far a magnet is from a given sensor within a triangulation set. The one method would follow the measures taken in Equation 1 of section 3.3.1, where the autonomous distance to a particular sensor is found. Another method is to determine a relative distance with respect to the other sensors in the triangulation set. Equation 10 shows the relative distance (RD) calculation with respect to the first sensor in a triangulation set like that found in Figure 27. For the current design of the HPM shell, *1.8 inches* is used for the *RD* calculation because it was the design specification for the distance between sensors.

$$RD_1 = 1.8\left(1 - \frac{D_1}{D_1 + D_2 + D_3}\right)$$

Equation 10: Relative Distance with respect to Sensor 1

From Equation 10, it should be understood that the distances denoted by D_i are the result of a 10-bit analog-to-digital conversion from the *MC68HC908GZ16* microcontroller's ADC. Therefore, this is the analog output of the *MLX90215* sensor, then converted to a digital number for mathematical processing. Likewise, the calculations of the three relative distances of a triangulation set are shown in Figure 31. Furthermore, the RD_i result should substitute for D_i in all algorithmic calculations.



$$RD_1 = 1.8\left(1 - \frac{D_1}{D_1 + D_2 + D_3}\right)$$

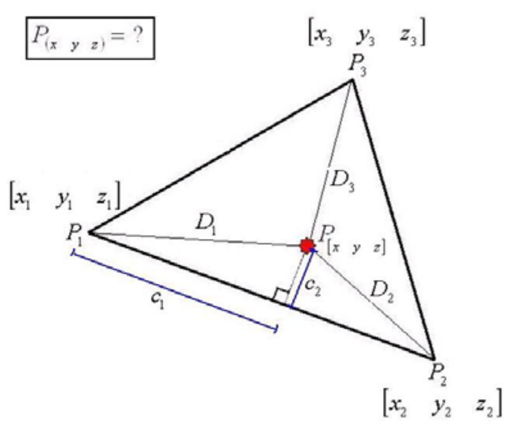
$$RD_2 = 1.8\left(1 - \frac{D_2}{D_1 + D_2 + D_3}\right)$$

$$RD_3 = 1.8\left(1 - \frac{D_3}{D_1 + D_2 + D_3}\right)$$

Figure 31: Relative Distances of Triangulation Set

6.3 DETERMINING COORDINATE VECTORS

The coordinate vector composed of the *x, y,* and *z* component of the magnet position is the desired result for this algorithm. A result vector can be produced using Equation 4, utilizing the appropriate quantities. This is shown in Figure 32 for succinct explanation of the algorithm to determine the coordinates of a point inside a three-dimensional triangle.



$$RD_1 = 1.8\left(1 - \frac{D_1}{D_1 + D_2 + D_3}\right)$$

$$RD_2 = 1.8\left(1 - \frac{D_2}{D_1 + D_2 + D_3}\right)$$

$$\overrightarrow{P_aP_b} = \left[(x_b - x_a) \quad (y_b - y_a) \quad (z_b - z_a)\right]$$

$$s_{a-b} = \left\|\overrightarrow{P_aP_b}\right\| = \sqrt{(x_b - x_a)^2 - (y_b - y_a)^2 - (z_b - z_a)^2}$$

$$P = P_1 + c_1 u_1 + c_2 u_2$$

$$u_1 = \frac{\overrightarrow{P_1P_2}}{s_{1-2}} \qquad u_2 = \frac{\overrightarrow{P_1P_3} - \frac{1}{2}\overrightarrow{P_1P_2}}{\frac{\sqrt{3}}{2} \cdot s_{1-3}}$$

$$c_1 = \frac{(s_{1-2})^2 + (RD_1)^2 + (RD_2)^2}{2 \cdot s_{1-2}} \qquad c_2 = \sqrt{(RD_1)^2 + (c_1)^2}$$

Figure 32: Algorithm Overview of Finding Point Inside a Triangle

7.0 SOURCE CODE DESIGN

All programming for the HPM was done through the *MC68HC908GZ16* microcontroller. The microcontroller handles sampling of *MLX90215* sensor's analog output by commanding the channels through the *DG406* analog multiplexer and retrieving the associated value of each signal through the resident ADC. After sampling, the microcontroller is programmed to do the necessary processing for both filtering and the algorithm for calculating the coordinates for a point inside a triangle. The linear progression of the microcontroller's program code is illustrated in Figure 33.

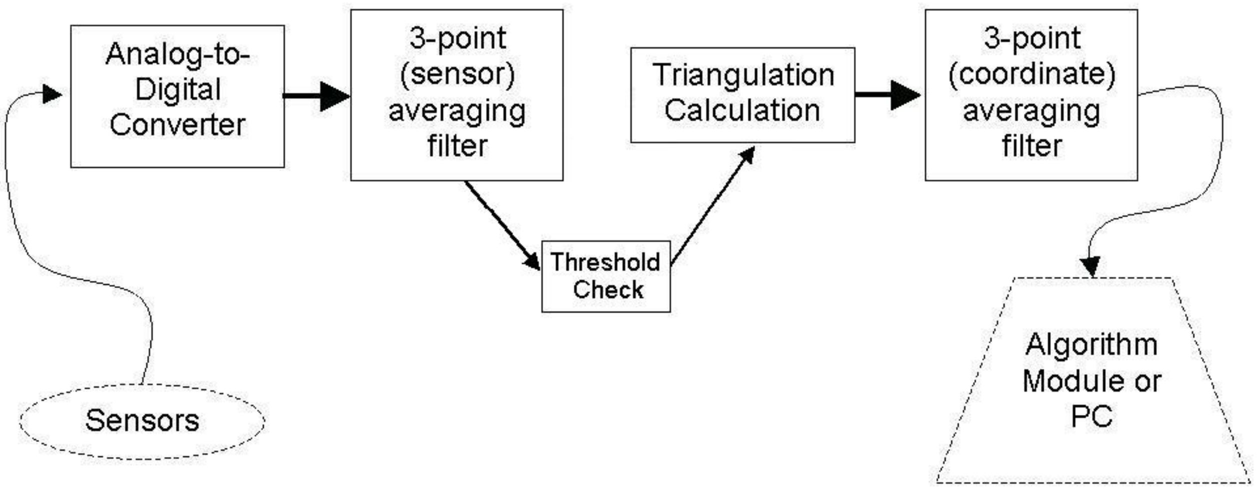

Analog-to-Digital Converter
3-point (sensor) averaging filter
Triangulation Calculation
3-point (coordinate) averaging filter
Threshold Check
Algorithm Module or PC
Sensors

Figure 33: Microcontroller Program Process

From Figure 33, it can be seen that the sensor's analog outputs are fed to the ADC. The result of each sensor's AD conversion is stored into a *running three-point averaging filter* for each sensor. Therefore, there are 19 averaging filters for the sensors in the HPM shell. The purpose of averaging the sensor's analog response is because there is some noise produced between the ADC and the "non-uniformity" of a magnetic field triggering the sensors. This provides smoother transitions in the sensor response for the tracking algorithm.

Prior to performing the triangulation calculation, the sensors in a given triangulation set must have a response above the "noise" threshold. After the triangulation algorithm calculates the coordinate vector, the result is then put into another *running three-point averaging filter* for each component (x, y, and z) of the result vector. After the coordinate filtering, the result is then transmitted to an external device for analysis. In the case for prototyping the HPM, the external device was a PC through the serial communications port via *RS-232*. This is described in the section 4.4.

The preliminary source code for the HPM was done in MATLAB to verify that the calculation of the coordinates for a point inside a triangle could be done in reasonable time. This preliminary MATLAB source code is in Appendix B.1. As for the source code resident on the microcontroller, it was written in the language *C* using the Metrowerks Codewarrior IDE. The microcontroller *C* source code can be found in Appendix C.

8.0 PROOF OF CONCEPT

As a preliminary affirmation of electrical and software design coherence, a proof of concept was established by analyzing the HPM's data via *RS-232* through a PC's serial communications port. There were two separate applications to analyze the HPM data. The first application affirms tracking of the magnet around the HPM shell's grid of sensors. The second application emphasizes any patterns in data made by the HPM, whether they are correct or a deviation from desired response. In each case, the routine of the HPM that transmits results is tailored to interface with the application intended to support analysis. The *C* source code for the *transmit_results* routine to interface with the two applications can be found in Appendix C.2.

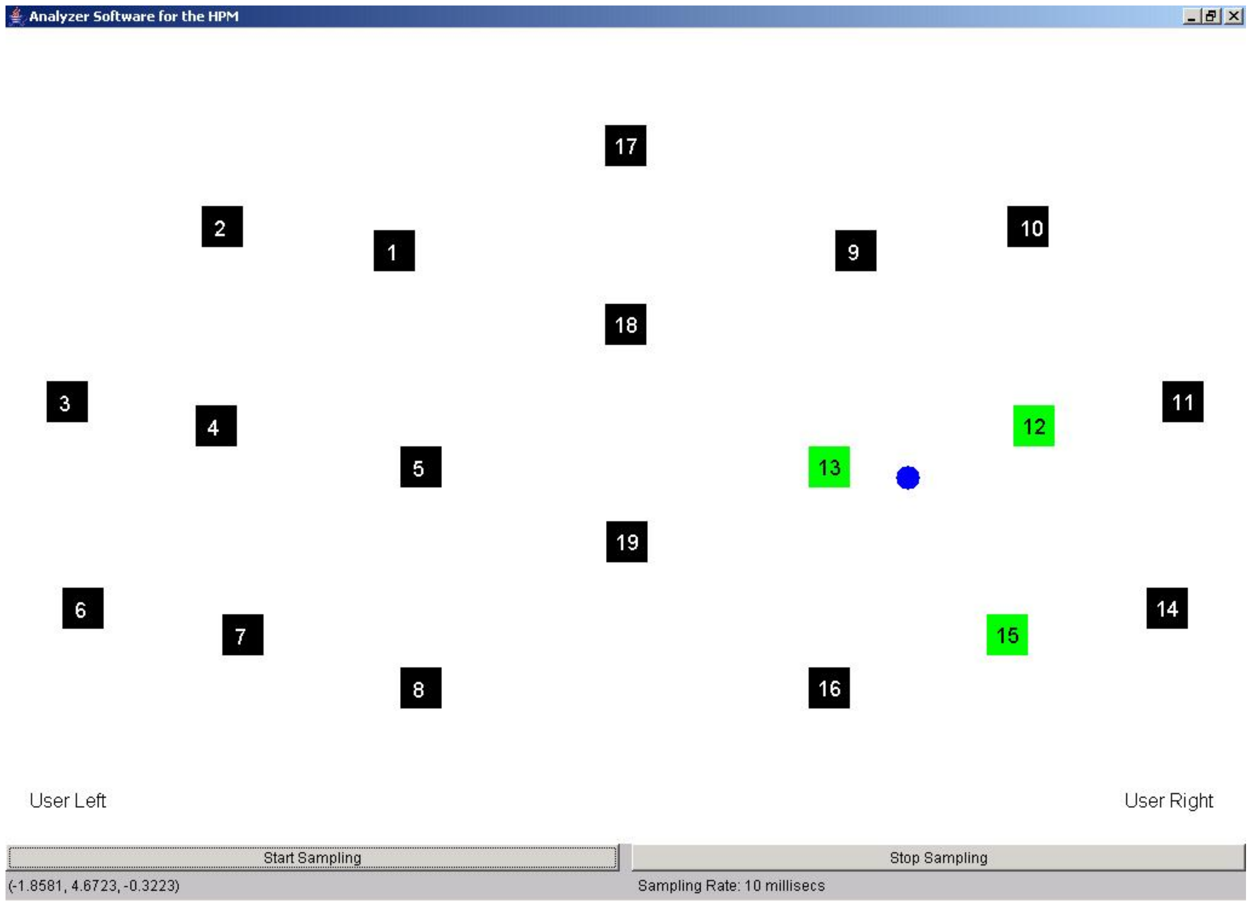
8.1 CONFIRMATION OF HPM TRACKING

To analyze the reliability of the HPM, a program was developed to display the position of the magnet with respect to the *MLX90215* sensors. The information in the transmission frame of Figure 34 are ASCII characters transmitted through the serial communications port. The number of bytes (of 8 bits) are at the bottom of each.

A	***F***	[X]	***F***	[Y]	***F***	[Z]	***N***	[***1st Sensor***]	[2nd ***Sensor***]	[3rd ***Sensor***]
1	1	6 or 7	1	6 or 7	1	6 or 7	1	*1*	*1*	*1*

Figure 34: Serial Communication Frame for Reliability Tracking

The first block of the transmission frame in Figure 34 is the ASCII character "A," where it serves as the preamble to the frame. When the PC's analyzer software sees the preamble, it then knows that coordinate information about the magnet is about to follow. Before the transmission of each floating-point coordinate (as a string of ASCII characters), an "F" is transmitted for prompting. Lastly, an "N" character is transmitted to indicate that the next three bytes in succession will be the numbers (1-19) of the sensors detecting the magnet in a triangulation set. Figure 35 is a rendered screenshot of the analyzer software when the HPM is tracking the magnet with sensors *12, 13*, and *15*.

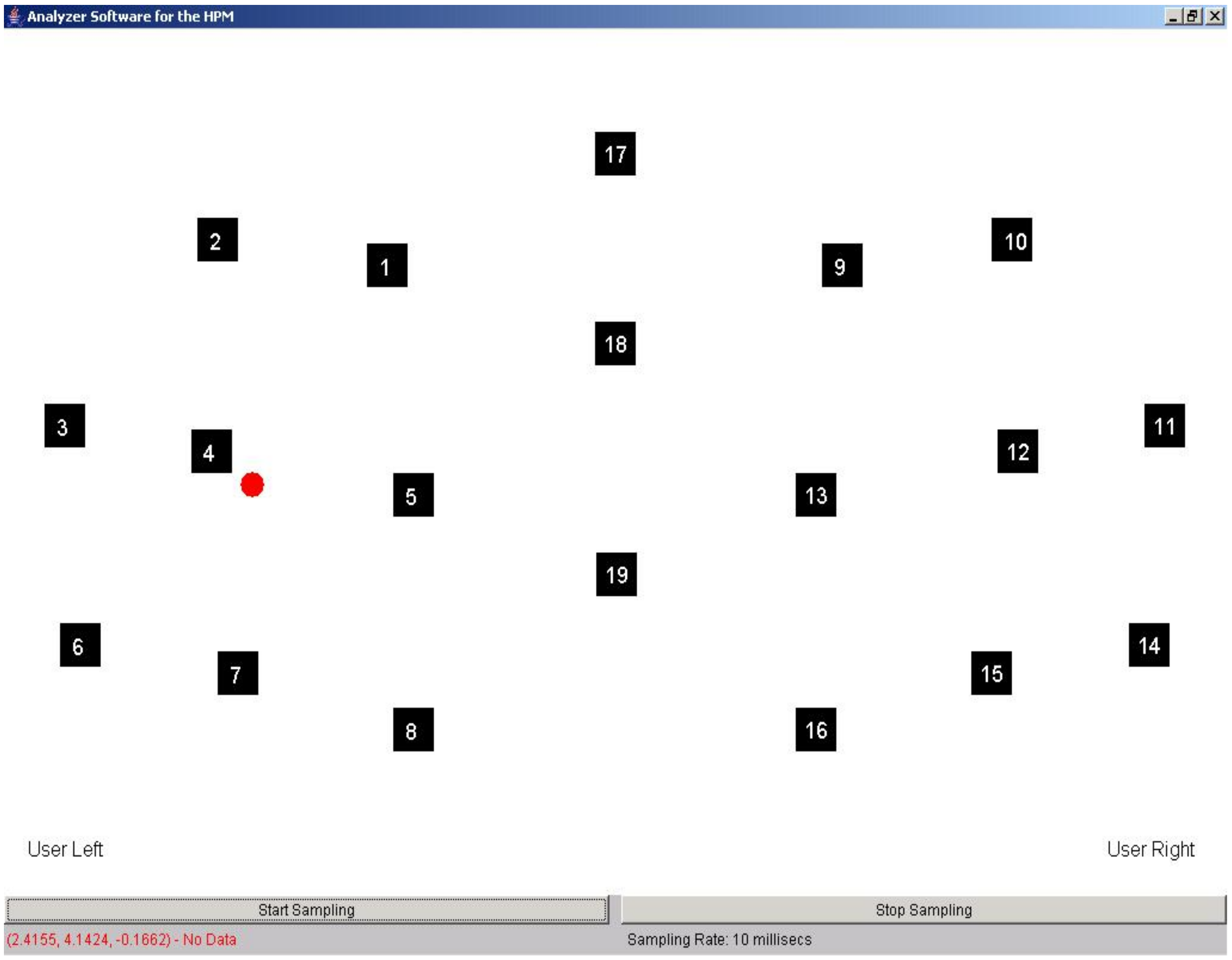


(-1.8581, 4.6723, -0.3223)

Sampling Rate: 10 millisecs

Figure 35: Position Tracking Reliability Software

The blue-filled circle in Figure 35 signifies the magnet, while the black squares with white numbers represent the sensors in the HPM shell. Figure 35 also depicts that the analyzer software toggles the sensor squares to green when they are used in the current triangulation set in tracking the magnet position. When the HPM cannot determine a magnet position, it sends an "A" and then an immediate "Z" for the transmission frame. This signals the analyzer software that there is no new data. Accordingly, the software will toggle the magnet dot to *red* with the last acceptable coordinates. These actions are illustrated in Figure 36.



(2.4155, 4.1424, -0.1662) - No Data

Sampling Rate: 10 millisecs

Figure 36: Position Tracking Reliability Software (No Data)

The position tracking affirmation software is a two-dimensional representation of the HPM shell, where only the horizontal (x) and vertical (z) components are used for display. The depth (y) component can be ignored for testing since the HPM is only intended to track the orientation of the user's head, and not necessarily the displacement from shell's surface. The test for position tracking reliability has shown that the HPM can successfully convey the general direction of a user's head.

8.2 RAW DATA PLOTTING

Another important test for the HPM is for consistency when there is a fixed magnet position. This test would show any deviation, jitter, or error, in the output response of the HPM. The test was conducted using a program that plots the raw data from the HPM for visual display in a 10 second interval. This plotting application receives a different transmission frame than the position tracking reliability software. The transmission frame for the plotting software is rendered in Figure 37. The results of the HPM calculations were converted to suit this transmission format.

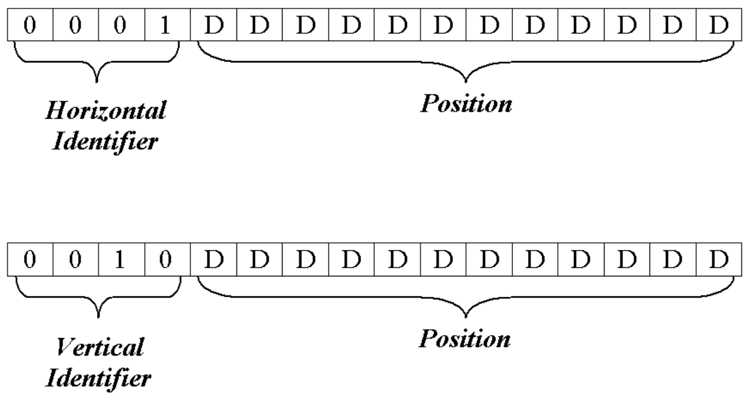
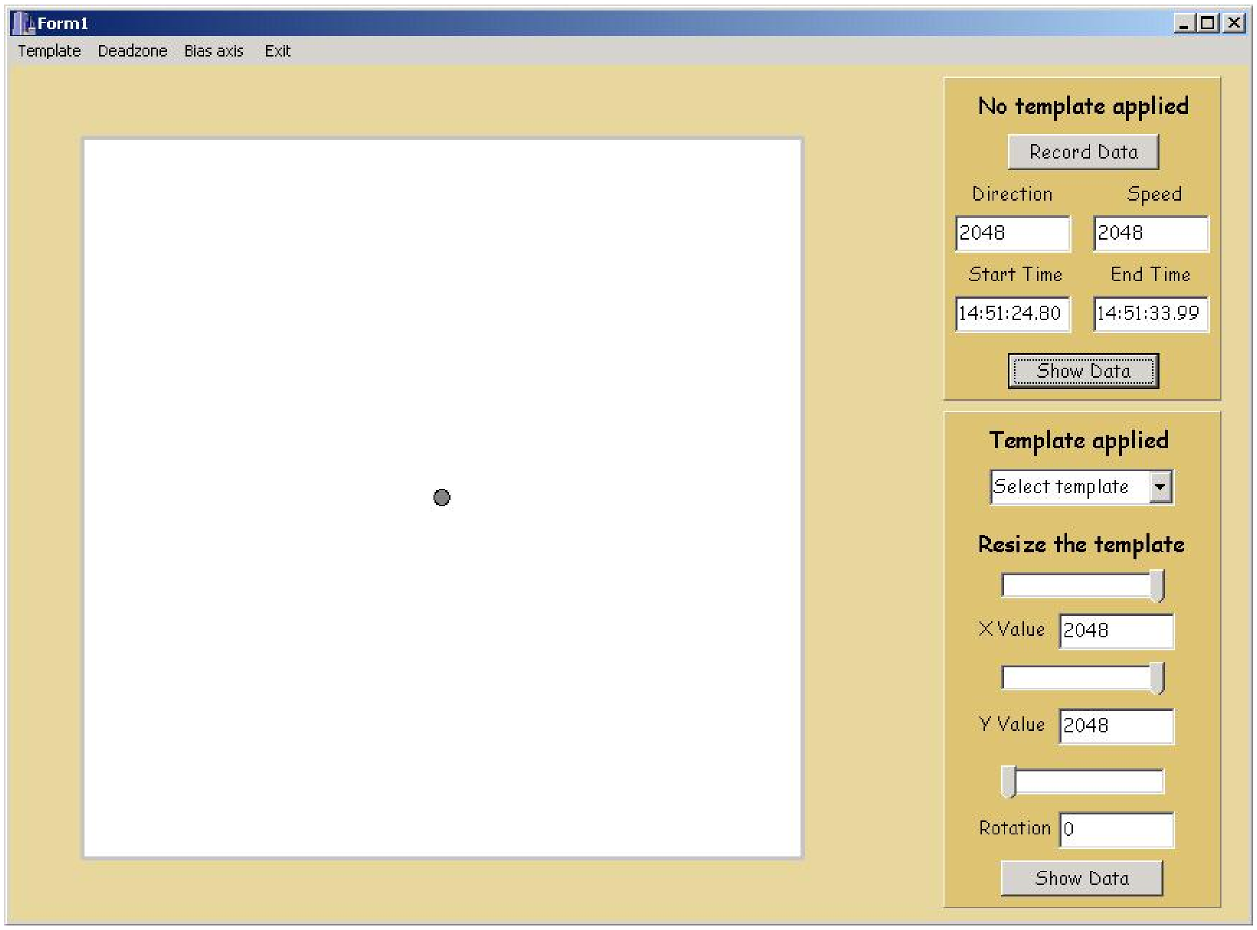

0 0 0 1 D D D D D D D D D D D D
Horizontal Identifier
Position
0 0 1 0 D D D D D D D D D D D D
Vertical Identifier
Position

Figure 37: Serial Communication Frame for Consistency Testing

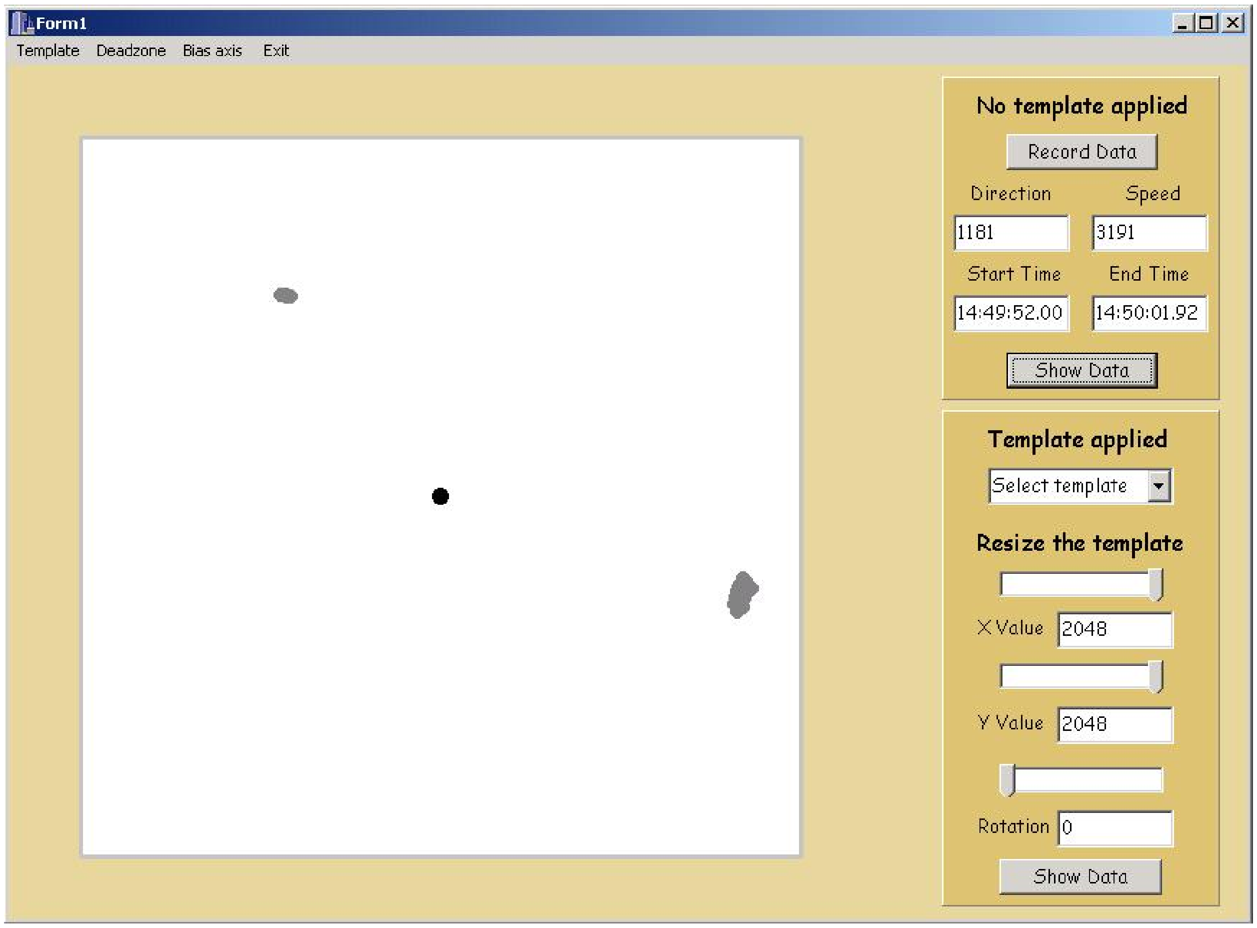
There are actually two separate frames shown in Figure 37, but they are actually sent abutted together. The first frame with a *horizontal identifier* is transmitted, then followed by the frame with the *vertical identifier*. This software application plots the vertical (z) versus horizon (x) components, where the 12 bits in each frame marked by a *D* represents a binary bit. With 12 bits, the largest number that can be represented is *4,906*, which is the extreme top for the vertical component and extreme right for the horizontal component. A zero (*0)* for the vertical and horizontal bits would correspond to the extreme bottom and left respectively. Intrinsically, a value of *2,048* for both components is the middle of the plot as shown in Figure 38.



Direction	2048
Speed	2048
Start Time	14:51:24.80
End Time	14:51:33.99
X Value	2048
Y Value	2048
Rotation	0

Figure 38: Raw Data Plotting Software (Center)

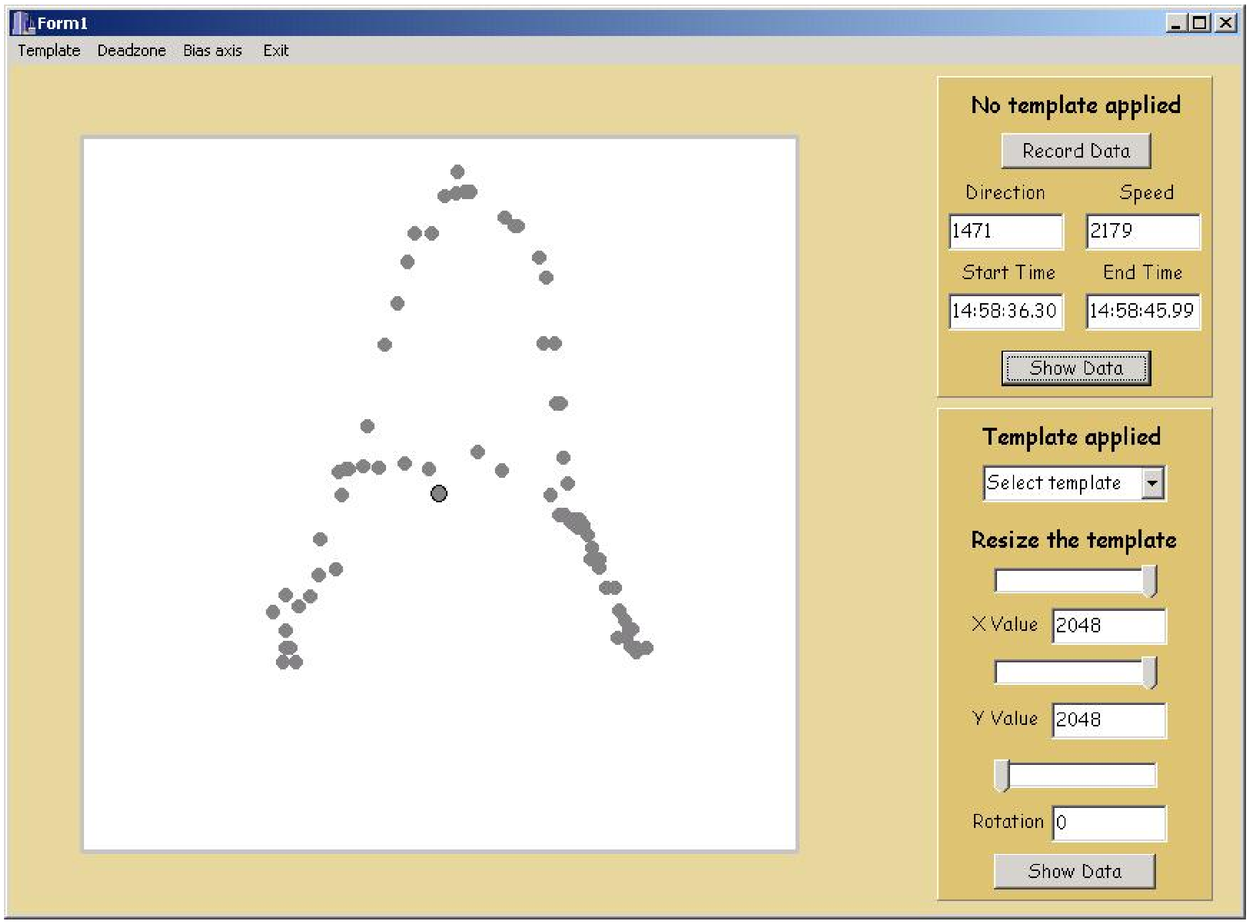
The center plot of Figure 38 is when there is no data for the HPM to send. This is to test how stable the HPM stays idle, and filters spurious or erroneous signals when not intended for operation. The second test was to determine how much the HPM coordinate vector deviated with a fixed magnet position in the HPM shell. In Figure 39, three separate magnet positions were enforced in the HPM shell, where the two small *gray* clusters of data points are a result of detection away from center.



Direction: 1181
Speed: 3191
Start Time: 14:49:52.00
End Time: 14:50:01.92
X Value: 2048
Y Value: 2048
Rotation: 0

Figure 39: Raw Data Plotting Software (3 Points)

It is apparent from Figure 39 that reasonable stability in the HPM result coordinates can be achieved to track general head orientation of the user. The last test done with the plotting program was to attempt a pattern. This may become useful in future development of the *TBI System* to assert certain commands by the person using the HPM. Figure 40 illustrates the letter "A" being drawn through the HPM shell interface.



Direction: 1471
Speed: 2179
Start Time: 14:58:36.30
End Time: 14:58:45.99
X Value: 2048
Y Value: 2048
Rotation: 0

Figure 40: Raw Data Plotting Software (Pattern Tracking)

For general head position, the test conducted through the plotting software shows that the HPM has reasonable stability. It further shows that it has the capability to assert commands through predetermined patterns. Given these functional proficiencies, the HPM should be characterized for its accuracy.

9.0 CHARACTERIZATION OF ACCURACY

To establish a quantifiable degree of accuracy, tests on a mill were performed to measure exact coordinates of the HPM. The procedure compared the actual magnet coordinates configured by the mill with those reported by the microcontroller of the HPM. In achieving this, each test point on the HPM needs to be rotated such that it would be orthogonal to the mill's quill that holds the magnet. This characterization has cooperatively involved mechanical, electrical, and software instrumentations.

Mechanical means for rotating the HPM at specific angles were needed, as well as generation of test points and calculation of associated test point rotation angles using linear (or vector) algebra. Given these are separate problems, they will be explained in different sections of this chapter.

9.1 RANDOM TEST POINT GENERATION

As opposed to manually selecting test points for the characterization, a set of random test points were generated with a MATLAB script. The script produced coordinates in the *x* (horizontal) and *z* (vertical) axes. The *y* (depth) axis was not generated because it could be retrieved from the SolidWorks CAD software where the HPM design was created. The MATLAB script to generate random test points was intended to give a list of the test points, and also a graphical representation to confirm that they were distributed well for a characterization.

A list of the test points were written to a file for analysis of characterization results. The test points used for the characterization are found in Table 3. The script only generated the *x* and *z* coordinates, and the *y* coordinates were retrieved from the CAD software.

Table 3: Test Points

X (horizontal)	Y (depth)	Z (vertical)
0.3993	0.3907	1.6913
1.2284	0.4257	1.4530
0.1212	0.2202	1.8897
2.3334	-0.2017	1.3533
1.7650	0.1537	1.4440
2.4610	0.0131	0.8772
3.1762	-0.7218	0.7655
3.5786	-1.1261	0.2917
3.4414	-0.8447	0.2191
2.3160	0.5958	0.0637
1.9755	0.5600	0.6266
1.7481	0.6141	0.7954
0.9191	1.0899	1.6022
1.0704	1.0973	0.5231
1.2346	0.5493	1.2862
1.0245	1.4109	-0.0690
1.3134	1.3977	-0.2644
0.1122	1.5573	-0.1587
0.7736	1.8572	-1.0783
1.1468	1.9551	-1.7150
1.0989	1.8975	-1.4446
1.7031	1.5706	-1.2991
1.4803	1.3655	-0.3754
1.5382	1.5761	-1.0171
2.5184	0.4939	-0.1141
2.4827	0.6414	-0.3924
1.9628	1.0107	-0.2610
2.6957	0.7761	-1.3476
2.9665	0.2939	-0.8218
3.1765	0.0102	-0.8023
3.3986	-0.4846	-0.4367
3.0096	0.0007	-0.2029
3.3806	-0.3352	-0.7468

To affirm that the test points were distributed well, the script also plots the test points in a *z* (vertical) vs. *x* (horizontal) fashion. The plotted test points are shown in Figure 41. It should be noted that test points for only half of the HPM shell were generated because its design is mirrored about the *z* axis. Therefore, only one side is necessary for characterization. The vertices of the formed triangles in Figure 41 are the locations of sensors with respect to the *right* side of the illustrated HPM shell. The *blue* lines connecting the sensors are to show the test points in each triplet of the shell. Accordingly, the *red* circles represent the test points.

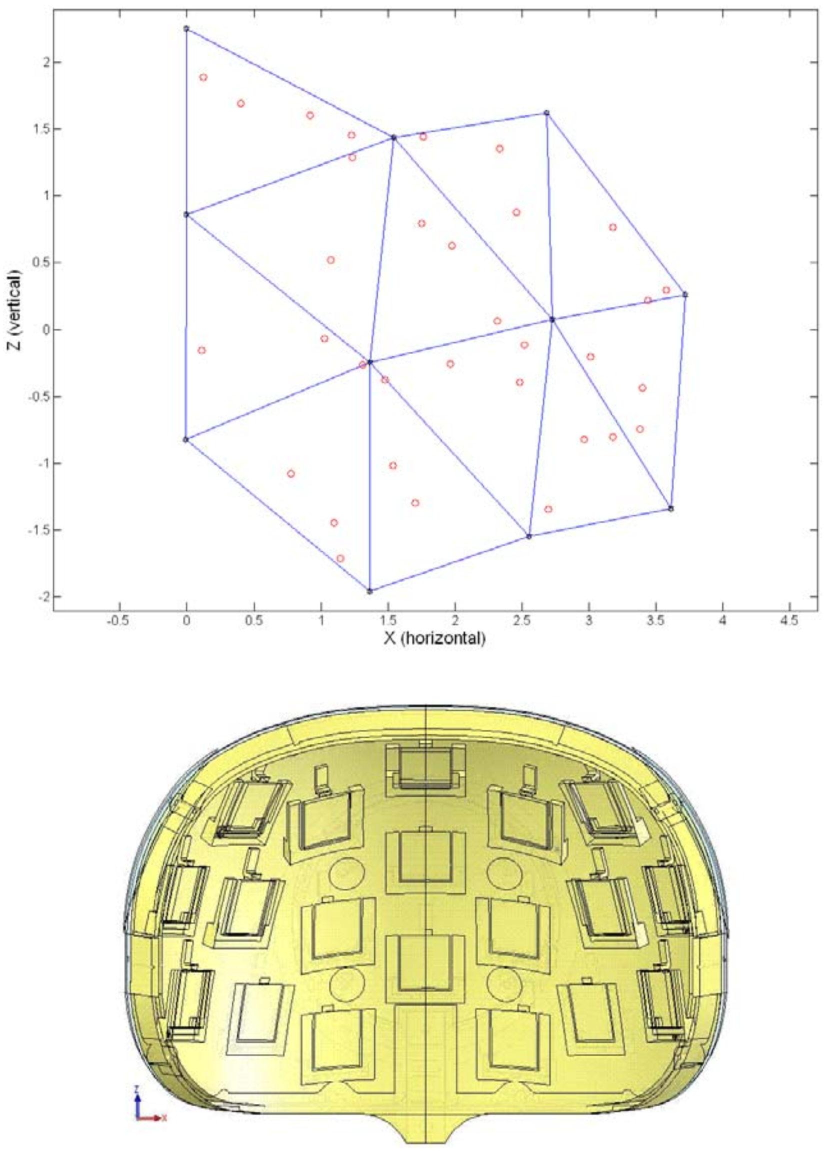


Figure 41: HPM Test Points

The MATLAB script used to generate the test points and graphs can be found in Appendix D.1.

9.2 ANGLE CALCULATIONS

Once the test points have been confirmed, the next step is to determine the associated horizontal and vertical angles that will make each test point vertically aligned with the quill of the manual mill. Finding these angles involves modeling the HPM shell's vertical and horizontal curves to determine the angles to rotate on each of these axes.

9.2.1 Cubic Spline Interpolation

Using arbitrary data points along the vertical and horizontal curves of the HPM shell, a *piecewise polynomial approximation* can be formed. This approximation is a *cubic spline interpolation* that ensures continuous differentiability on each interval, and also a continuous second derivative on the interval [21]. This type of interpolation provides better "smoothness" for the approximation function, rather than a piecewise linear or polynomial regression model [21]. Using the approximation gives a curve which can be used to determine the angles needed to make the normal of a test point vertical. This relationship is illustrated in Figure 42.

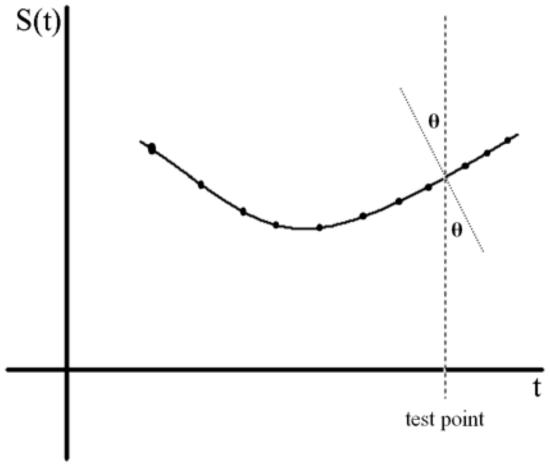


Figure 42: Spline Curve and Test Point Angle

Once cubic spline approximations are formed each for the vertical and horizontal curves, the angle of rotation can then be determined by the angle between a tangent and the curve at each test point. This is mathematically consistent with taking the *arctangent* of the approximation function's derivative at a test point. Considering the approximation function to be $S(t)$ in Figure 42 and t to be a horizontal (x) or vertical (z) component of a test point, then Equation 11 determines the angle needed to rotate on the respective axis.

$$\theta = \tan^{-1}\left(S'(t)\right)$$

Equation 11: Angle of Tangent to a Curve

Given each interval of a cubic spline interpolation is realized by a cubic polynomial function [21], a test point taking on the value of t in Equation 11 is *irrationally* infinite in the numerical sense. This allows for testing of any desired test point within the numeric bounds of the approximation curve and the precision of the computing hardware.

9.3 ROTATING THE HPM

One of the major concerns in rotating the HPM is to ensure that the rotation maintains the same origin throughout the entire process. While rotating on both the vertical and horizontal axes, maintaining one origin of rotation is realized with a ball joint. When the ball joint is assembled tight enough, it can lock onto an angle, and hold it for testing. Figure 43 is a SolidWorks CAD realization detailing the HPM with a level plate between it and the ball joint.

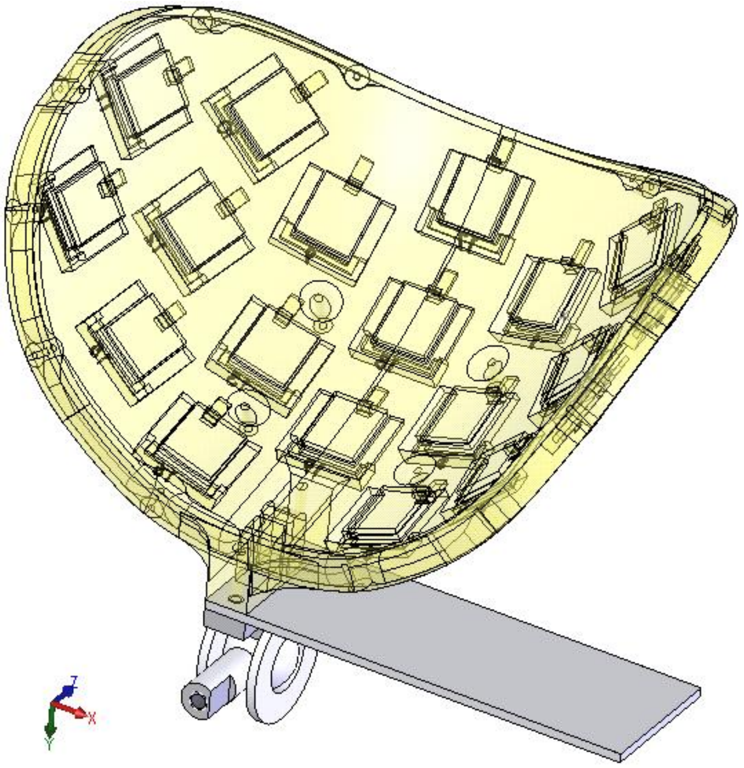


Figure 43: HPM with Level Plate for Angle Measurement

The level plate depicted in Figure 43 is used to measure the angles at which the HPM is tilted. When tilting the HPM, the angles in the horizontal and vertical directions must be precisely measured. This was accomplished using a digital protractor that assesses angles to a *tenth of a degree*. The digital protractor used for the characterization is shown in Figure 44.

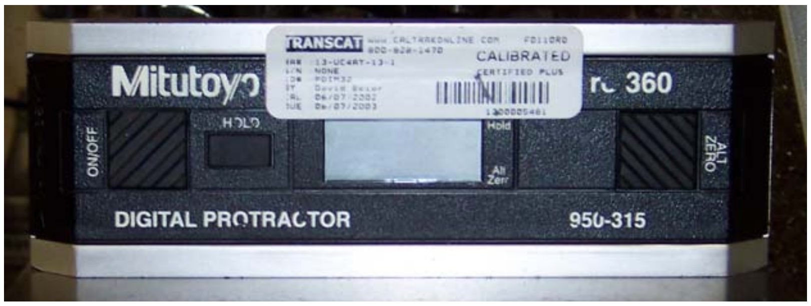


Figure 44: Digital Protractor

9.4 MILL COORDINATES

After the HPM shell has been rotated for a particular test point, the vertical and horizontal distances to each location has been displaced. To accurately position the mill for the displaced test points, new coordinates must be determined after rotating about the ball joint. To attain the new coordinates, two parts of information needs to be known. First is the position of the ball joint with respect to the origin of the HPM subspace. Second is the rotation matrices to convert the test points to there new positions after rotation.

The coordinates of the center of the ball joint in the HPM subspace is shown in Equation 12.

$$\underline{\text{BallJoint}} = \begin{bmatrix} 0.0 & 4.2750 & -3.4820 \end{bmatrix}$$

Equation 12: Ball Joint Coordinates

Given the rotation about the ball joint, it behaves as another origin. This leads to expressing the test point vectors in terms of a relationship to the ball joint. This relationship is detailed in Equation 13.

$$\underline{\text{TestPoint}_{\text{wrt - BallJoint}}} = \underline{\text{TestPoint}} - \underline{\text{BallJoint}}$$

Equation 13: Test Point with respect to Ball Joint

With the information from Equation 12 and Equation 13, the mill coordinates can then be determined in the vertical and horizontal directions. Calculations for the vertical and horizontal mill coordinates are shown in Equation 14 and Equation 15 respectively.

$$\underline{\text{Mill}_{\text{vertical}}} = \underline{\text{TestPoint}_{\text{wrt-BallJoint}}} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(phi) & \sin(phi) \\ 0 & -\sin(phi) & \cos(phi) \end{bmatrix}$$

Equation 14: Mill Coordinate in Vertical Direction

$$\underline{\text{Mill}_{\text{horizontal}}} = \underline{\text{Mill}_{\text{vertical}}} \cdot \begin{bmatrix} \cos(phi) & \sin(phi) & 0 \\ -\sin(phi) & \cos(phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Equation 15: Mill Coordinate in Horizontal Direction

Examination of Equation 14 and Equation 15 shows that they must be performed in the presented chronological sequence. The matrices in Equation 14 and Equation 15 rotate about the *x* and *z* axes respectively [20]. The operation to determine the mill coordinates is simply to add one (1) to all the vertical mill coordinates. This is done because the *reference point* is exactly one (1) inch away from the ball joint in the vertical (z) direction. More accuracy is upheld by referencing the mill from the edge of the HPM, rather than obscure references from the ball joint. The reference point is detailed in Figure 45.

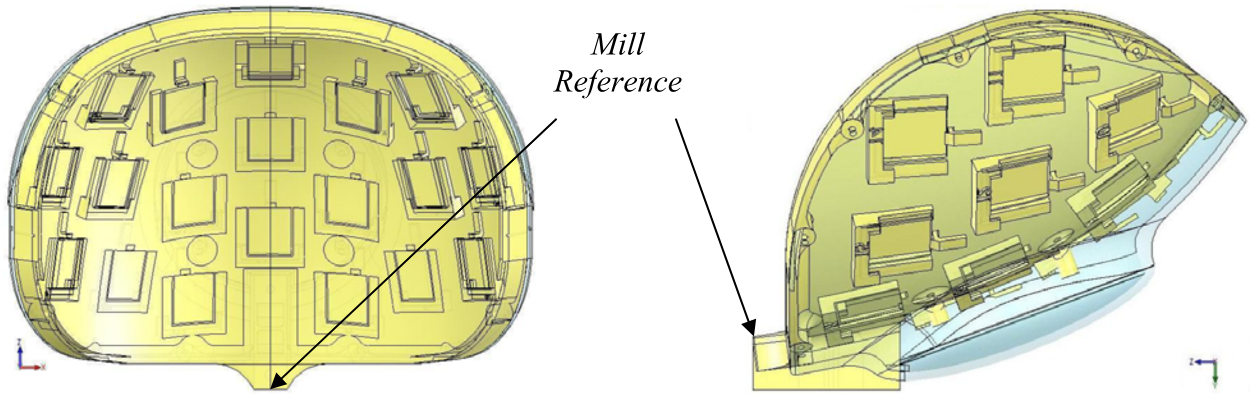


Figure 45: Mill Reference

The mill coordinates pertain to moving in the forward (up) and lateral (right) directions with respect to the left perspective of Figure 45. Given the associated angles for each test point are done in conjunction with the mill coordinates, both are shown in Table 4.

Table 4: Mill Coordinates & Rotation Angles

Mill $_{vertical}$	Mill $_{horizontal}$	Angle $_{vertical}$	Angle $_{horizontal}$
6.6953	0.4117	40.3357	2.8576
6.4616	1.365	37.1954	15.0891
6.9625	0.1211	43.2424	0.1774
6.7352	2.6972	36.0455	33.7975
6.6171	1.9904	37.0872	27.1921
6.1125	3.0605	32.1653	36.9123
6.3739	4.0298	31.3685	48.8445
5.9375	4.9048	27.2267	57.4081
5.7002	4.7195	26.474	56.3511
4.8525	3.2162	24.7587	36.6672
5.5579	2.5389	30.2863	31.1227
5.7094	2.2052	31.5881	28.5127
6.1783	0.8889	39.1042	9.5789
5.1783	1.3386	29.4109	13.277
6.2288	1.4201	35.3357	15.5688
4.3526	1.3769	23.4216	13.1317
4.1316	1.8543	21.925	19.5726
4.1911	0.1159	22.6625	0.1312
3.1119	1.0546	18.9233	9.1348
2.3047	1.6801	13.9787	16.2847
2.6504	1.5837	15.9382	15.1499
2.932	2.6018	17.2594	30.5909
4.0209	2.1521	21.3392	24.4599
3.2742	2.2974	19.2899	26.5043
4.6582	3.571	23.025	40.0582
4.2653	3.5905	21.2668	39.9039
4.28	2.841	21.946	32.8455
3.0985	4.035	16.8223	43.5259
3.9327	4.3365	20.1983	46.665
4.053	4.6531	20.2673	51.5034
4.6216	4.9606	21.1004	56.7875
4.7323	4.3105	22.3331	47.1125
4.2361	4.9616	20.4427	56.5858

9.5 TESTING PARAMETERS

With the HPM positioned in a mill and a magnet connected (via adhesive) to the quill, the magnet was positioned using the mill's lead screws for each test point. The calculated position from the HPM was then recorded by a laptop computer on the periphery. The testing configuration is illustrated in Figure 46.

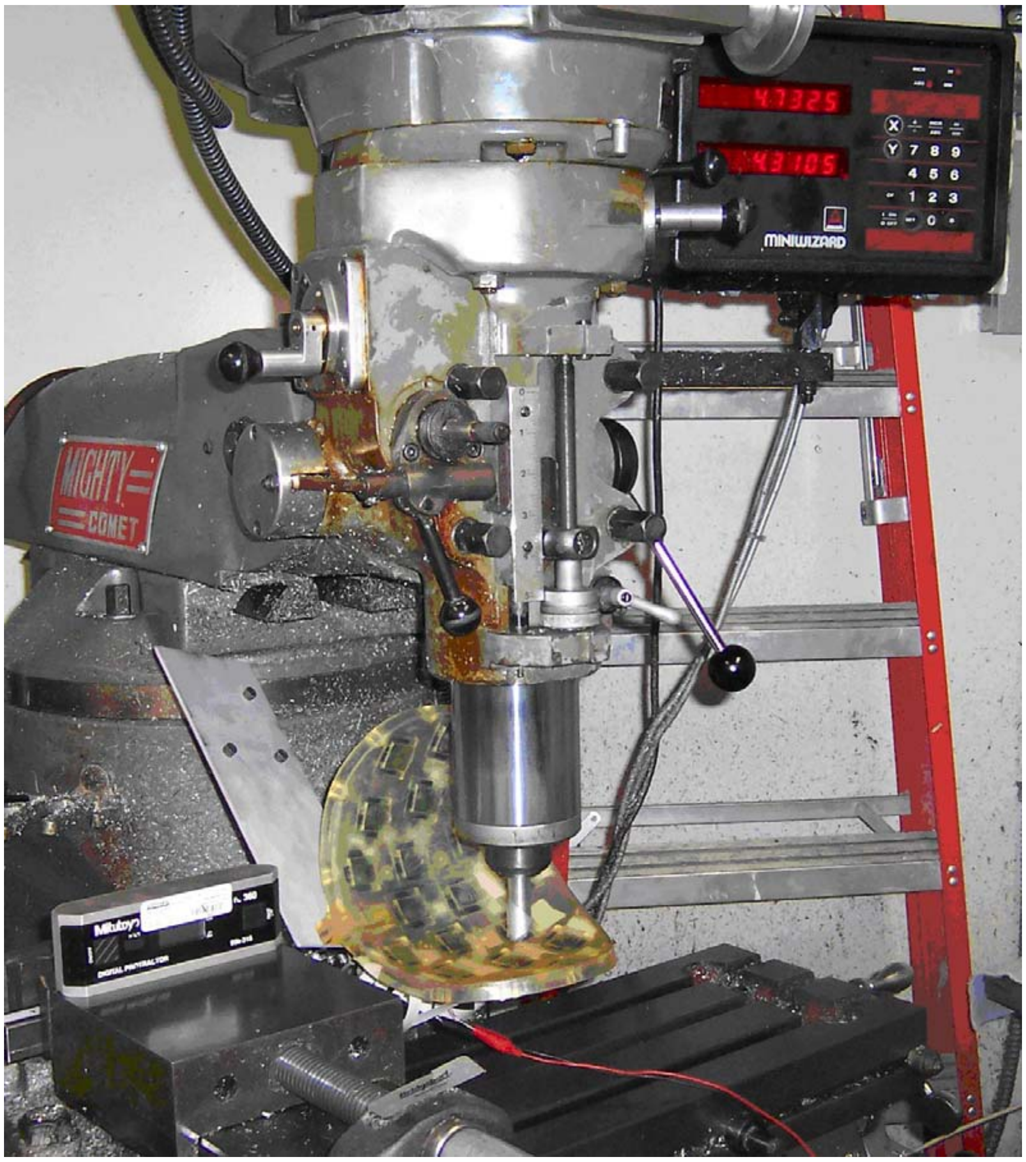


Figure 46: HPM Mounted in Mill

As a measure of robustness, different angles of tilt for the magnet were considered. A zero (0) degree angle means that the North Pole of the magnet is parallel to the test point. The angles tested were zero (0), 30, 45, and 60 degrees of tilt. One of the angles is shown being tested in Figure 47.

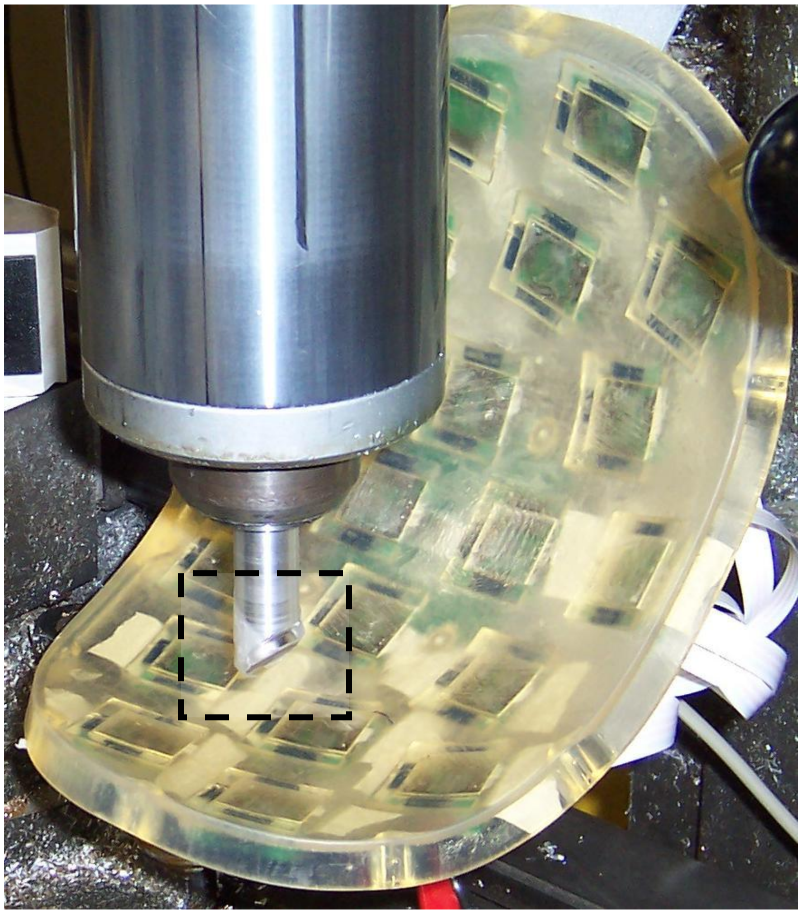


Figure 47: Angle Tested on HPM

To position different angles from the quill of the mill, aluminum stock was cut given its relatively high tolerance to form shapes, and its non-magnetic characteristics that could potentially have adverse affects on the characterization results. Figure 48 shows the angled aluminum stock in zero (0), 30, 45, and 60 degrees from left to right respectively. Simple inspection shows that each piece of stock is a different length. To compensate for this when measuring at fixed displacements from the HPM shell, the mill's quill displacement can be measured with its incorporated ruler. For each measurement, the magnet was lowered to make slight contact with the surface of the shell, and then displaced upwards vertically by the fixed distances of the characterization measured by the quill's incorporated ruler.

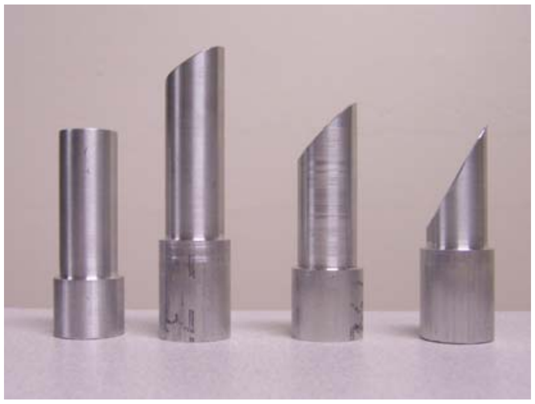


Figure 48: Angled Aluminum Stock

9.6 CHARACTERIZATION RESULTS

There are a few aspects to consider when interpreting the results of this characterization. Human error was introduced into this characterization in positioning the HPM for each test point. To achieve the correct test point position, the rotations were manually performed in both the vertical and horizontal directions to attain the appropriate angles respective to the x and z axes.

Besides human error, the method for securing the analog Hall sensors was not finalized, therefore, were positioned using a Velcro bonding agent. Although these inaccuracies were fundamentally imposed, it is believed that the characterization will yield convincing results demonstrating its capability to track head orientation.

9.6.1 Deviation Angle of Zero Degrees

The average error of the HPM was only *0.1527* inches away from the true position when tested with no deviation angle. This part of the characterization was performed at *1.25* inches away from the surface of the HPM shell. This is seen in Table 5 where the 33 test points, with the associated error of the HPM, are shown.

Table 5: Characterization Deviation for Zero Degrees

Test Point	**Test X**	**Test Z**	**Measured X**	**Measured Z**	**Distance from *Test* (average = 0.1527)**
1	0.3993	1.6913	0.5449	1.6888	0.1456
2	1.2284	1.4530	1.4661	1.3920	0.2454
3	0.1212	1.8897	0.0369	1.7113	0.1973
4	2.3334	1.3533	2.3598	1.4377	0.0884
5	1.7650	1.4440	1.8272	1.4634	0.0652
6	2.4610	0.8772	2.3042	0.7725	0.1885
7	3.1762	0.7655	3.2271	0.8044	0.0641
8	3.5786	0.2917	3.6582	0.2720	0.0820
9	3.4414	0.2191	3.2825	0.2475	0.1614
10	2.3160	0.0637	2.4627	-0.0757	0.2024
11	1.9755	0.6266	1.7842	0.8403	0.2868
12	1.7481	0.7954	1.8831	0.8956	0.1681
13	0.9191	0.6022	0.7705	1.6982	0.1769
14	1.0704	0.5231	0.9003	0.3504	0.2424
15	1.2346	1.2862	1.2725	1.3399	0.0657
16	1.0245	-0.0690	1.0508	0.0522	0.1240
17	1.3134	-0.2644	1.3283	-0.1572	0.1082
18	0.1122	-0.1587	0.0275	-0.2059	0.0970
19	0.7736	-1.0783	0.8842	-0.9119	0.1998
20	1.1468	-1.7150	0.9511	-1.6742	0.1999
21	1.0989	-1.4446	1.0245	-1.5460	0.1258
22	1.7031	-1.2991	1.7834	-1.1497	0.1696
23	1.4803	-0.3754	1.6734	-0.3700	0.1932
24	1.5382	-1.0171	1.6837	-0.9202	0.1748
25	2.5184	-0.1141	2.5998	-0.2130	0.1281
26	2.4827	-0.3924	2.2887	-0.2079	0.2677
27	1.9628	-0.2610	1.9170	-0.2898	0.0541
28	2.6957	-1.3476	2.7616	-1.3262	0.0693
29	2.9665	-0.8218	2.8204	-0.9386	0.1870
30	3.1765	-0.8023	3.1146	-0.6081	0.2038
31	3.3986	-0.4367	3.4253	-0.5710	0.1369
32	3.0096	-0.2029	3.1388	-0.1308	0.1480
33	3.3806	-0.7468	3.4502	-0.7682	0.0728

9.6.2 Deviation Angle of Thirty Degrees

The average error at $30°$ was not much different than with no deviation angle. The error slightly increased to *0.1893 inch* away from the true position of the magnet. This part of the characterization was performed at *1.0* inches away from the surface of the HPM shell. Table 6 shows the 33 test points with the associated error of the HPM, where one (1) measure was outside the precision of *0.30 inch* as denoted with an asterisk (*).

Table 6: Characterization Deviation for Thirty Degrees

Test Point	Test X	Test Z	Measured X	Measured Z	Distance from *Test* (average 0.1893)
1	0.3993	1.6913	0.4928	1.6818	0.0940
2	1.2284	1.4530	1.0792	1.2706	0.2356
3	0.1212	1.8897	-0.0960	1.7019	0.2871
4	2.3334	1.3533	2.3646	1.2702	0.0888
5	1.7650	1.4440	1.8246	1.2645	0.1891
6	2.4610	0.8772	2.4043	1.0465	0.1785
7	3.1762	0.7655	3.3753	0.5397	0.3010 *
8	3.5786	0.2917	3.7059	0.5200	0.2614
9	3.4414	0.2191	3.5870	0.1455	0.1631
10	2.3160	0.0637	2.5033	0.0998	0.1907
11	1.9755	0.6266	2.0605	0.7613	0.1593
12	1.7481	0.7954	1.6477	0.5733	0.2437
13	0.9191	0.6022	0.7826	1.6625	0.1492
14	1.0704	0.5231	1.1335	0.7489	0.2345
15	1.2346	1.2862	1.0304	1.3582	0.2165
16	1.0245	-0.0690	0.9947	-0.1927	0.1272
17	1.3134	-0.2644	1.2689	-0.4408	0.1819
18	0.1122	-0.1587	-0.0235	-0.2526	0.1650
19	0.7736	-1.0783	0.7691	-1.3236	0.2453
20	1.1468	-1.7150	0.9098	-1.7196	0.2370
21	1.0989	-1.4446	1.3034	-1.6764	0.3091
22	1.7031	-1.2991	1.7512	-1.2436	0.0734
23	1.4803	-0.3754	1.4642	-0.2221	0.1541
24	1.5382	-1.0171	1.2991	-1.1742	0.2861
25	2.5184	-0.1141	2.5670	-0.0599	0.0728
26	2.4827	-0.3924	2.4020	-0.4799	0.1190
27	1.9628	-0.2610	2.1275	-0.2005	0.1755
28	2.6957	-1.3476	2.5761	-1.2030	0.1877
29	2.9665	-0.8218	3.0143	-0.7781	0.0648
30	3.1765	-0.8023	3.4220	-0.6841	0.2725
31	3.3986	-0.4367	3.5547	-0.6691	0.2800
32	3.0096	-0.2029	3.0647	-0.1822	0.0589
33	3.3806	-0.7468	3.4813	-0.5256	0.0940

9.6.3 Deviation Angle of Forty-Five Degrees

At 45° of deviation, the accuracy of the HPM significantly degraded to *0.3712 inch* away from the true position of the magnet. This part of the characterization was performed at *0.75 inch* away from the surface of the HPM shell, and the results are in Table 7. Fifteen (15) measures that failed to meet the precision of *0.30 inch* are denoted with an asterisk (*).

Table 7: Characterization Deviation for Forty-Five Degrees

Test Point	**Test X**	**Test Z**	**Measured X**	**Measured Z**	**Distance from *Test* (average 0.3712)**
1	0.3993	1.6913	0.3502	1.8147	0.1328
2	1.2284	1.4530	1.1340	0.7814	0.6782 *
3	0.1212	1.8897	-0.1233	1.9748	0.2589
4	2.3334	1.3533	2.0887	1.5156	0.2936
5	1.7650	1.4440	1.7334	1.6707	0.2289
6	2.4610	0.8772	2.7834	1.6164	0.8064 *
7	3.1762	0.7655	3.0862	1.0247	0.2744
8	3.5786	0.2917	3.3480	0.3059	0.2310
9	3.4414	0.2191	3.2548	0.5186	0.3529 *
10	2.3160	0.0637	2.2149	0.0850	0.1033
11	1.9755	0.6266	2.2135	0.7748	0.2804
12	1.7481	0.7954	1.5545	1.4488	0.6815 *
13	0.9191	0.6022	0.8409	1.6443	0.0888
14	1.0704	0.5231	0.8239	0.0782	0.5086 *
15	1.2346	1.2862	0.9947	1.4090	0.2695
16	1.0245	-0.0690	1.0278	-0.2513	0.1823
17	1.3134	-0.2644	1.2118	-0.6205	0.3703 *
18	0.1122	-0.1587	0.1270	0.2655	0.4245 *
19	0.7736	-1.0783	0.8583	-0.8433	0.2498
20	1.1468	-1.7150	1.1371	-1.7341	0.0214
21	1.0989	-1.4446	1.3013	-0.4365	1.0282 *
22	1.7031	-1.2991	1.8852	-1.9355	0.6619 *
23	1.4803	-0.3754	1.5990	0.1915	0.5792 *
24	1.5382	-1.0171	1.2611	-1.4121	0.4825 *
25	2.5184	-0.1141	2.4688	-0.2051	0.1036
26	2.4827	-0.3924	2.7794	-0.1604	0.3766 *
27	1.9628	-0.2610	1.8470	0.4162	0.6870 *
28	2.6957	-1.3476	2.4285	-0.7563	0.6489 *
29	2.9665	-0.8218	2.9587	-0.7361	0.0861
30	3.1765	-0.8023	2.9913	-0.5890	0.2825
31	3.3986	-0.4367	3.6238	-0.0549	0.4433 *
32	3.0096	-0.2029	3.0488	0.0263	0.2325
33	3.3806	-0.7468	3.1807	-0.7483	0.1999

9.6.4 Deviation Angle of Sixty Degrees

The average error of the HPM was *0.5117 inches* away from the true position when tested with a $60°$ angle of deviation. This part of the characterization was performed at *0.50 inch* away from the surface of the HPM shell. Even at this close distance to the shell, the HPM was not able to consistently detect the magnet with a $60°$ tilt. Table 8 shows these results with an asterisk (*) denoting failed precision while experiencing intermittent detection.

Table 8: Characterization Deviation for Sixty Degrees

Test Point	Test X	Test Z	Measured X	Measured Z	Distance from *Test* (average 0.5117)
1	0.3993	1.6913	0.6548	2.2204	0.5876 *
2	1.2284	1.4530	0.9300	1.6314	0.3477 *
3	0.1212	1.8897	0.6508	2.2007	0.6142 *
4	2.3334	1.3533	1.9376	2.0091	0.7660 *
5	1.7650	1.4440	2.0104	1.2261	0.3282 *
6	2.4610	0.8772	2.5245	1.5143	0.6403 *
7	3.1762	0.7655	3.2741	0.3267	0.4496 *
8	3.5786	0.2917	4.0348	0.2506	0.4580 *
9	3.4414	0.2191	3.7409	-0.2816	0.5834 *
10	2.3160	0.0637	1.8605	0.0256	0.4571 *
11	1.9755	0.6266	1.6349	1.2324	0.6950 *
12	1.7481	0.7954	1.5527	1.4943	0.7257 *
13	0.9191	0.6022	1.4665	1.4479	0.5687 *
14	1.0704	0.5231	0.8349	0.6947	0.2914
15	1.2346	1.2862	1.5660	1.1006	0.3798 *
16	1.0245	-0.0690	1.1605	-0.5030	0.4548 *
17	1.3134	-0.2644	1.3178	-0.0710	0.1935
18	0.1122	-0.1587	0.0749	-0.4714	0.3149 *
19	0.7736	-1.0783	0.4272	-1.6335	0.6544 *
20	1.1468	-1.7150	1.7121	-1.9042	0.5961 *
21	1.0989	-1.4446	1.6395	-1.7821	0.6373 *
22	1.7031	-1.2991	1.4294	-0.7006	0.6581 *
23	1.4803	-0.3754	1.3539	-0.0459	0.3529 *
24	1.5382	-1.0171	1.7540	-1.0013	0.2164
25	2.5184	-0.1141	2.4524	-0.2442	0.1459
26	2.4827	-0.3924	2.6025	-0.4228	0.1236
27	1.9628	-0.2610	1.4910	-0.4843	0.5220 *
28	2.6957	-1.3476	3.1898	-0.7163	0.8017 *
29	2.9665	-0.8218	2.7476	-0.2910	0.5742 *
30	3.1765	-0.8023	3.0844	-1.0641	0.2775
31	3.3986	-0.4367	3.3897	0.2511	0.6879 *
32	3.0096	-0.2029	3.3561	0.0939	0.4562 *
33	3.3806	-0.7468	2.3613	0.1017	1.3262 *

9.6.5 Paired T-Test of Deviation Angles

As the deviation angle increases in the characterization, the average error tends to increase accordingly. In determining whether the average difference of the coordinates are essentially zero, a t-test pairing first the known and calculated *x* coordinates is administered. The same paired t-test is then performed on the known and calculated *z* coordinates. The results of this test for each deviation angle of the characterization are shown in Table 9.

Table 9: Paired T-Test for X and Z Coordinates

Deviation Angle	**Null Hypothesis [X, Z]**	**Significance [X, Z]**	**Confidence Interval [{X_{low}, X_{high}},{ Z_{low}, Z_{high} }]**
0º	[Accept, Accept]	[0.9852, 0.8458]	[{-0.5016, 0.4923}, {-0.5331, 0.4382}]
30º	[Accept, Accept]	[0.9728, 0.9722]	[{-0.5242, 0.5065}, {-0.4969, 0.4798}]
45º	[Accept, Accept]	[0.8542, 0.4881]	[{-0.4519, 0.5439}, {-0.6727, 0.3246}]
60º	[Accept, Accept]	[0.9351, 0.5717]	[{-0.5143, 0.4739}, {-0.6661, 0.3710}]

The *null hypothesis* of the paired t-test in Table 9 is that the means of both known and calculated coordinates are equal at the *0.05* significance level. The *significance* is the probability that the associated value of *T* in the paired t-test is at least as large as observed under the null hypothesis. Furthermore, the *confidence interval* is 95% for the true difference in means. By inspection of Table 9, it can be accepted that the known and calculated coordinates have the same means. One characteristic to note is that the significance of the paired t-test significantly decreases as the deviation angle exceeds 30º.

Although the known and calculated coordinates satisfy the null hypothesis, the significance in the deviation angle cases suggests accurate positioning is not characteristic in the entire 60º range. Accurate positioning is most characteristic up to a 30º deviation angle.

9.6.6 Discussion of Characterization

Under ideal conditions with no deviation angle, the closest and furthest observed HPM calculation from the true position was *0.0652* and *0.2868 inch* respectively. As the deviation angle increases, the distance where the magnet is still detectable decreased. This is exemplified in the transition from a zero (0) to 30º deviation angle. The furthest distance where the magnet was detectable decreased from *1.25* to *1.0* inch. As the deviation angle increased further, the distance detectable to the magnet continued to decrease due to degradation of the magnetic field.

In the cases of deviation angles, a summary of how the HPM met the specification for under *0.30* inch is:

- Zero (0) degree trials at *1.25 inch* displacement met $^{33}\!/_{33}$ test points, yielding *100 %* accuracy.
- 30º trials at *1.0 inch* displacement met $^{32}\!/_{33}$ test points, yielding *96.97 %* accuracy.
- 45º trials at *0.75 inch* displacement met $^{18}\!/_{33}$ test points, yielding *54.55 %* accuracy.
- 60º trials at *0.50 inch* displacement met $^{5}\!/_{33}$ test points, yielding *15.15 %* accuracy.

Analyzing the characterization data demonstrates that head tracking can be provided reliably up to about a 30º deviation angle in the magnet at a distance between *1 - 1.25 inch* away from the HPM shell.

Given 30º of angle deviation, some procedure should be exercised to mount the magnet on a user's head within the deviation constraints. A magnet attached to the rear of a head band or baseball cap should provide adequate angle placement. The last thing to note is that considering the fundamental error imposed on the characterization, it is likely that the performance of the HPM surpasses that recorded with more stringent experimentation measures.

10.0 CONCLUSIONS

The HPM has demonstrated that it can track head orientation within *three tenths of an inch*. Its physical design specification was to follow the contour movement of the head. Given this condition, it is unlikely for a magnet to deviate more than $30°$ from parallel to any respective instantaneous point in the shell while in use. From the data resulting from characterization, the HPM is sufficiently accurate to use in subject trials for persons with a TBI. Furthermore, the tests conducted demonstrate sufficient accuracy to track head orientation. Use of the HPM in the *TBI System* can potentially provide persistent cued guidance for the user with a TBI. This ensures that the user will be attentive to the direction of travel driven by an EPW.

With the appropriate logic at the EPW controller, this type of cued guidance administered through the HPM can be obtained by ensuring a joystick assertion is congruent with the direction of head orientation. Over extended periods of this type of integrated rehabilitation for a person surviving a TBI, they can "re-learn" their remaining motor control to drive an EPW without supervision. The expectations of the HPM in the *TBI System* for this population are to improve upon visual inattentiveness and reduce the risk of collisions while driving an EPW to acceptable levels. This promotes major steps toward realizing independent mobility.

The objective succeeding EPW driving is for the HPM to be a potential gateway to other assistive technologies or areas typically unavailable to those with severe disabilities. Through pattern recognition, the HPM can be used to ascertain commands just as the letter "A" was gestured in section 8.2 in Figure 40. This pattern recognition could be integrated into the functions of the *Algorithm Module* of the *TBI System* shown in Figure 1, or the HPM itself.

The aspect of the HPM is that it is a non-stigmatizing AT. There are no intrusive linkages around the user's face; only a small magnet on the rear of the head. Through this work, the HPM has demonstrated itself to be a non-stigmatizing AT to provide cued guidance to persons with a TBI. Hopes for this technology are to be integral in future developments to better the quality of life for those with severe disabilities.

11.0 FUTURE OBJECTIVES

The HPM is an integral part of the *TBI System*, where the purpose of this research was to develop a device to aid persons with a TBI to drive an EPW. The HPM has been specifically developed to track the head position of the user, and report the coordinates that it determines. There are three foreseen areas to complete the objectives of the *TBI System.* One of the areas would be to further improve upon the accuracy of the HPM. Another area is an *Algorithm Module* to interpret the information received from both the HPM and a direction-sensing joystick. The third area is establishing a communication protocol between the modules of the *TBI System*.

To improve the accuracy of the HPM, there are four obvious places to begin:

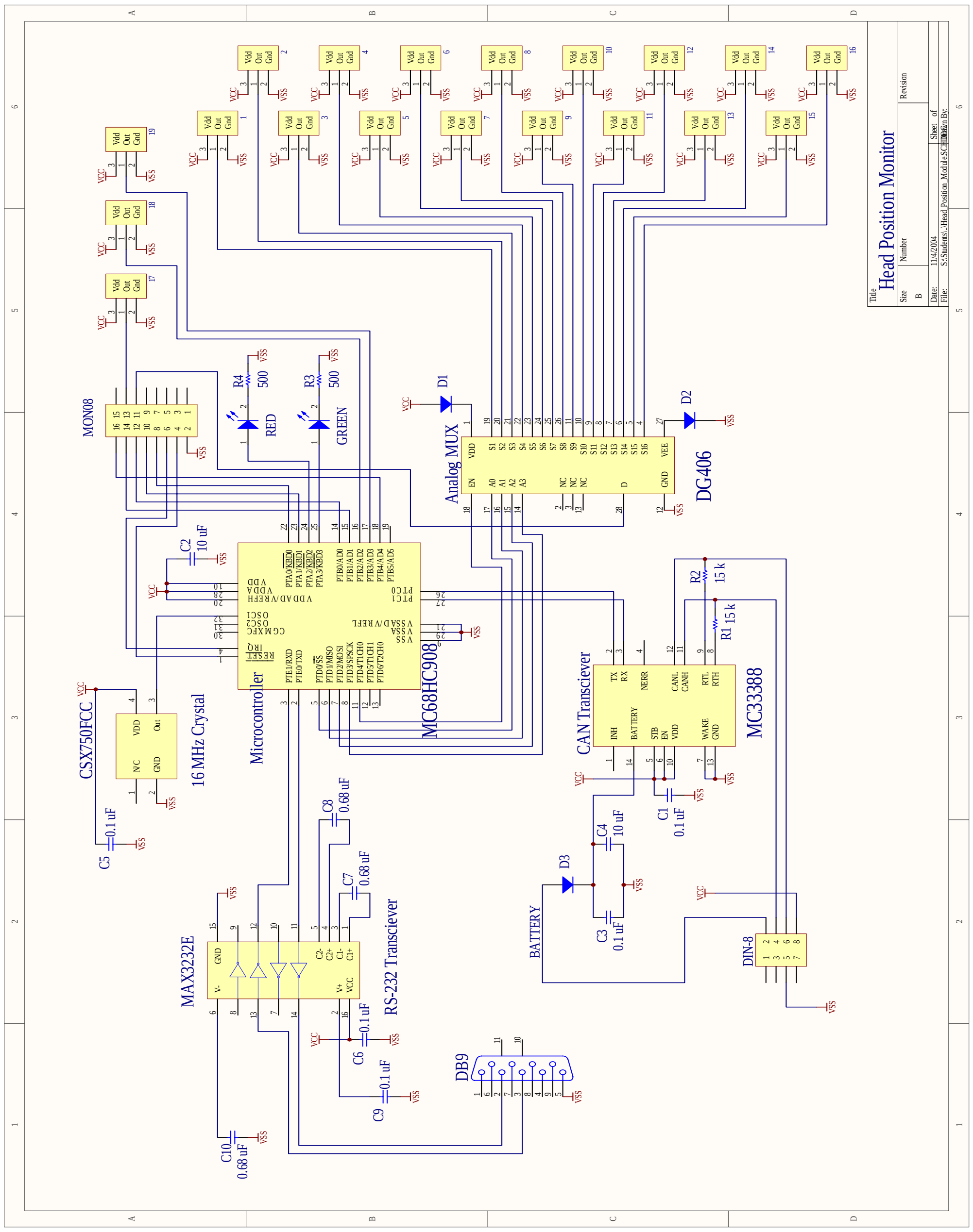
- Establish a more precise mechanical system for placement of sensors around the HPM shell.
- Use a magnet that produces a stronger magnetic field.
- With these programmable sensors, increase the sensitivity to magnetic flux density.
- Determine a more accurate formula for the relative distances (RD_i) used in Equation 10 of the triangulation algorithm.

To provide noise immunity in the *TBI System*, communication between modules (HPM, Algorithm Module, etc.) should be handled through a Controller Area Networks (CAN) bus. The CAN bus can currently be accessed with an IC transceiver through Motorola to interface with the *MC68HC908GZ16* microcontroller currently used in the HPM [18]. However, a network protocol for communication between modules needs to be developed. Once information can be transmitted within the guidelines of a network protocol, the *TBI System* can then realize the task of human testing.

The HPM can potentially be incorporated to operate devices other than an EPW for persons with a TBI. This area of research remains unchallenged. However, the future aims of the *TBI System* research may very well lead into more integrated applications.

APPENDIX A

A.1 HEAD POSITION MONITOR (HPM) FULL SCHEMATIC



APPENDIX B

B.1 MATLAB TRIANGULATION CALCULATION – TRIANGULATION.M

```
%%%Sample coordinate calculation for triangulation of point inside triangle
format long

%%%Coordinates of each sensor in triangulation for calculation%%%
P1 = [ 2.7288 3.8780 0.0733 ];
P2 = [ 3.6127 3.2039 -1.3425 ];
P3 = [ 2.5512 4.6432 -1.5463 ];
%%%Coordinates of each sensor in triangulation for calculation%%%

a = 0.9398;     %Distance 'P1' is from magnet
b = 1.2584;     %Distance 'P2' is from magnet


%Calculate length of side 1-to-2 of triangle
s_1_2  =  sqrt(  (P2(1,1)-P1(1,1))^2  +  (P2(1,2)-P1(1,2))^2  +  (P2(1,3)-
P1(1,3))^2)

%Calculate length of side 1-to-3 of triangle
s_1_3  =  sqrt(  (P3(1,1)-P1(1,1))^2  +  (P3(1,2)-P1(1,2))^2  +  (P3(1,3)-
P1(1,3))^2)

%Calculation of unit vector 1
unit_v1 = (P2 - P1)/s_1_2

%Calculation of unit vector 2
unit_v2 = ((P3-P1) - (1/2)*(P2-P1)) / (sqrt(3)/2 * s_1_3)

%Calculation of unit vector 1 coefficient
unit_v1_coefficient = (s_1_2^2 + a^2 - b^2) / (2 * s_1_2)

%Calculation of unit vector 2 coefficient
unit_v2_coefficient = sqrt( a^2 - unit_v1_coefficient^2 )

Q = P1 + unit_v1_coefficient*unit_v1 + unit_v2_coefficient*unit_v2
```

B.2 MATLAB SENSOR ANGLES SIMULATION -- SENSOR_ANGLES.M

```
% sensor_angles.m

%   >> DESCRIPTION <<
%
%   this program calculates the angles of each sensor relative to center.
%   it takes positions from the file 'sensor positions.txt', translates the
%   origin, and
%   calculates the positions in spherical coordinates.
%
%   >> NOMENCLATURE <<
%
%   mode        method for calculating angles since more than one can exist
%   pos         position matrix for sensors.  column 1 is the order in which
%               data was collected, column 2 is the order on the shell,
columns
%               3 through 5 are x through z, respectively
%   x, y, z     columns of pos which pertain to these coordinates
%   m, n        size of pos
%   pos_tran    transformed position matrix
%   theta       angle of sensor with respect to +x-axis
%   theta2      angle of sensor with respect to +y-axis
%   phi         angle from x-y plane
%   omega       rotation angle
%   trans       rotation matrix
%   TMPpos_rot  temporary rotated position matrix
%   pos_rot     rotated position matrix
%   pos_tran_ex expanded transformation matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

%%% initialization
% clear variables
clear;

% set mode of computation: 0=translate, 1=rotate
% ya, keep this 0
mode = 0;

% define positions of sensors
% Read in data from file, skipping first 6 lines of header
pos = dlmread('sensor positions.txt', '\t', 6, 0);
% define matrix configuration
x = 3;
y = 4;
z = 5;
% get size of pos
[m, n] = size(pos);
```

```
%%% transform the matrix because SolidWorks coordinate system isn't the same
as
%%% the real world.
%%% then get angle
switch mode
    case 0,
        % move x-z plane forward and x-y plane down
        for i = 1:m
            pos_tran(i, :) = pos(i,x:z) + [0.0 3.5 1.0];
        end

        % get vertical angle
        for i=1:m
           theta(i) = atan2(pos_tran(i,2), pos_tran(i,1));
           theta2(i) = atan(pos_tran(i,1)/pos_tran(i,2));
           phi(i) = atan2(pos_tran(i,3), ...
                          sqrt(pos_tran(i,1).^2 + pos_tran(i,2).^2));
        end
    case 1,
        % rotate 62 degrees
        omega = 28 * pi/180;
        trans = [1 0 0; 0 sin(omega) -cos(omega); 0 cos(omega) sin(omega)];
        for i=1:m
            TMPpos_rot = trans * [pos(i,x); pos(i,y); pos(i,z)];
            pos_rot(i,:) = [TMPpos_rot(1) TMPpos_rot(2) TMPpos_rot(3)];
        end

        % translate shell
        for i = 1:m
            pos_tran(i, :) = pos_rot(i,:) + [0 0 1.5];
        end

        % get horizontal angle
        for i=1:m
           theta(i) = atan(pos_tran(i,1)/pos_tran(i,3));
        end
    otherwise,
        disp('something went wrong');
end

%%% add zeros so that the origin is made aparent
for i=2*m:-2:2
    pos_tran_ex(i,:) = pos_tran(i/2,:);
    pos_tran_ex(i-1,:) = [0,0,0];
end

%%% let's do some plotting
% plot original positions
plot3(pos(:,3), pos(:,4), pos(:,5), 'x');
xlabel('x'); ylabel('y'); zlabel('z');
title('original file');
rotate3d on

% plot new positions
figure
plot3(pos_tran(:,1), pos_tran(:,2), pos_tran(:,3), 'o');
```

```
xlabel('x'); ylabel('y'); zlabel('z');
title('transformed matrix');
hold on

% plot lines
plot3(pos_tran_ex(:,1), pos_tran_ex(:,2), pos_tran_ex(:,3), '-');
xlabel('x'); ylabel('y'); zlabel('z');
title('transformed matrix');
rotate3d on

%%% reformat matrix for output
pos_tran(:,2:3+1) = pos_tran(:,:);
pos_tran(:,1) = pos(:,2);
% convert to degrees and save
pos_tran(:,5) = theta'*180/pi;
pos_tran(:,6) = phi'*180/pi;

pos_tran

%%% output sensor angles to screen
fprintf('sensor \tx \t\t\t\ty \t\t\t\t\tz \t\t\t\ttheta \t\t\t\tphi\n');
for i = 1:19
fprintf('%1.0f \t\t\t%5.4f \t\t%5.4f \t\t%5.4f \t\t%6.4f \t\t%6.4f\n',
pos_tran(i,1), ...
    pos_tran(i,2), pos_tran(i,3), pos_tran(i,4), pos_tran(i,5),
pos_tran(i,6));
end
```

B.3 MATLAB SIMULATION TEXT FILE – SENSOR POSITIONS.TXT

* These are the 3d positions in space (inches) for the hall effect sensors as modeled in the SolidWorks file, S:\joystick\Head Position Monitor\su-z\version6\ergo shell small6.SLDPRT.
* The Solidworks column gives the order in which the points appear in the SLDPRT file.
* The Pin Numbering 1 column gives the order of the sensors as defined for use.

SolidWorks	Pin Numbering 1	x	y	z
1	17	0.00000000	0.01056901	1.24796321
2	18	0.00000000	1.15414286	-0.14208674
3	1	1.54128944	0.46090313	0.43455922
4	5	1.36238794	1.50584018	-1.24427034
5	4	2.72882121	0.37799342	-0.92674584
6	2	2.68353575	-0.54508645	0.61788123
7	19	-0.00965783	1.79372472	-1.82461085
8	8	1.36425921	2.06250457	-2.95603003
9	7	2.55120928	1.14322603	-2.54627449
10	6	3.61273534	-0.29609005	-2.34248143
11	3	3.71981532	-1.11298786	-0.73991575
12	9	-1.54128944	0.46090313	0.43455922
13	13	-1.36238794	1.50584018	-1.24427034
14	12	-2.72882121	0.37799342	-0.92674584
15	10	-2.68353575	-0.54508645	0.61788123
16	16	-1.36425921	2.06250457	-2.95603003
17	15	-2.55120928	1.14322603	-2.54627449
18	14	-3.61273534	-0.29609005	-2.34248143
19	11	-3.71981532	-1.11298786	-0.73991575

APPENDIX C

C.1 C MAIN – HPM.C

```
/* Including used modules for compilling procedure */
#include "Cpu.h"
#include "Events.h"
#include "Term1.h"
#include "Inhr1.h"
#include "AD1.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"


#include "HPM_Procedures.h"


void main(void)
{
    char ch;
    unsigned char ctr = 0, i, SB[3];
    unsigned short DS;

  /*** Processor Expert internal initialization. ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization. ***/

    EnableInterrupts          /* Enables Global Interrupt */
    CONFIG1 |= 0x01;          /* Disables COP */
    DDRD = 0x1F;              /* Configures port D[4..0] as output */
    DDRA = 0x0C;

    /*
    ch = 'Z';
    while(ch != 'A') {

        while( (SCS1 & 0x20) == 0) { }
        ch = SCDR;
    } /*while ->The 'A' character has NOT been recieved yet */
    */

    for(;;) {
        PTA = 0x0C;

        /*
        ch = 'Z';
        while(ch != 'A') {

            while( (SCS1 & 0x20) == 0) { }
            ch = SCDR;
        } /*while ->The 'A' character has NOT been recieved yet   */
        */

        proximities[0] = 0;   /*Reset high value of the sensor proximities*/
        sample_HPM_sensors(); /*Sample & sort sensor analog values in
                              /*ascending order        */


        if(in_proximity) {
            /*Determines which sensors make up the triangulation set to begin
              vector calculations */
            i = determine_triangulation_set();

            if(i) {
                  /* Calculate coefficient for 1st unit vector */
                  unit_vector1_coefficient();
                  /* Calculate coefficient for 2nd unit vector */
                  unit_vector2_coefficient();
                  /* Calculate 1st unit vector */
                  unit_vector1();
                  /* Calculate 2nd unit vector */
                  unit_vector2
                  /* Calculate result vector */
                  result_vector();
                  /* Apply “smoothing” filter for coordinates */
                  coordinate_filter();
                  /* Transmit ‘result’ vector via RS-232 */
                  transmit_results();
            } /* if-> can determine triangulation set */
        } /* if */

        else {


            Term1_SendChar('A');  /*Send Preamble character*/
            Term1_SendChar('Z');  /*Send Negative Acknowledgement character
*/

            /*
            Term1_SendChar( ((char)(0x18)) );
            Term1_SendChar( ((char)(0x00)) );
            Term1_SendChar( ((char)(0x28)) );
            Term1_SendChar( ((char)(0x00)) );
            */
        } //else

        in_proximity = 0;
    } //for
```

```
#ifndef PE_OS_OSEK_SUPPORT
  for(;;){}
#else
  StartOS(Mode);                       /* Jump to OSEKturbo OS startup */
  /*DO NOT WRITE CODE BELOW THIS LINE*/
#endif PE_OS_OSEK_SUPPORT
}
```

C.2 C FUNCTIONS – HPM_PROCEDURES.H

```
/****************************************************************************
*
      Head Position Monitor (HPM)
      Procedures/Functions/Methods/Utilities
      Alex J. Bevly III
****************************************************************************/


#include "HPM_Static_Variables.h"

/****       Function Prototypes     ***************************************/

void setup_delay(void);
/* System delay to allow circuitry to stabilize */

short binary_search(float x);
/* Performs a binary search of the 'FP' array and
    returns the index closes to 'x' */

float abs_val(float a);
/* Return absolute value of 'a' */

float mult(float a, float b);
/* Floating-point multiplication using the 'MLT'
    and 'FP' tables to approximate the calculation */

void vector_subtract(float A[], float B[], float C[]);
/* Performs vector subtraction between 2 3-dimensional vectors */

float vector_magnitude(float A[]);
/* Calculates the magnitude of a 3-dimensional vector and returns
    it as a 'float' value */

void scalar_multiply(float t, float A[], float result[]);
/* Multiplies a vector 'A' times a scalar 't' */

//void Insertion_Sort(unsigned char x);   //Sorts 'HPM_temp' array using

void sample_HPM_sensors(void);
/* Sample & sort sensor analog values in ascending order */

int determine_triangulation_set(void);
/* Determines absolute triangulation set */

void unit_vector1_coefficient();
/* Calculate coefficient for 1st unit vector */
```

```
void unit_vector2_coefficient();
/* Calculate coefficient for 2nd unit vector */

void unit_vector1();    /* Calculate the 1st unit vector */

void unit_vector2();    /* Calculate the 2nd unit vector */

void low_pass_filter(); /*Filter's out large sporadic jumps in distance */

void result_vector();   /* Calculate the result vector where the magnet
lies*/

void coordinate_filter();     /*Average current & previous 2 coordinates */

void transmit_results();      /* Transmit results via RS-232 connection */

/****       Function Prototypes     ****************************************/


/****       Sample HPM Sensors      ***************************************
Description: Sample analog values in ascending then assigns the global
pointers P1, P2 & P3 to the sensors sensing the strongest magnetic field. */

void sample_HPM_sensors(void) {

    int i, j;
    unsigned int y;
    unsigned char sub, counter;
    counter = 0x10;     /* Initialize counter the same as 'PTD' */
    PTD = 0x10;
    ADSCR = 0x20;

    for(i=0; i < 16; i++) {

        for(j=0; j < ARB_DELAY; j++) {}   /* Wait small arbitrary delay */
        while( !(ADSCR & 0x80)) { }
        while( !(ADSCR & 0x80)) { }
            /*Ensure currently selected signal is being sampled!! */

        sub = ADRH;     y = sub << 8;
        sub = ADRL;
        y += sub;

        if(i != 10) {

            prev_proxy[i][1] = prev_proxy[i][0];
            prev_proxy[i][0] = proximities[i+1];
            proximities[i+1] = (y > 512) ? (((y>512)+prev_proxy[i][1]+prev_proxy[i][0])/3):0;

            if(proximities[i+1] > proximities[proximities[0]])  {
                proximities[0] = i + 1;
            } /* if */

            if(proximities[i+1] > THRESHOLD)
                  in_proximity = 1;
        } /* if ->signal is above the threshold */
        counter++;
        PTD = counter;
    } /* for */

    counter = 0x21;     /* Initialize counter the same as 'ADSCR' */
    ADSCR = counter;

    for(i=0; i < 3; i++) {

        for(j=0; j < ARB_DELAY; j++) {}   /* Wait small arbitrary delay */
        while( !(ADSCR & 0x80)) { }
        while( !(ADSCR & 0x80)) { }
            /*Ensure currently selected signal is being sampled!! */

        sub = ADRH;     y = sub << 8;
        sub = ADRL;
        y += sub;

        prev_proxy[i][1] = prev_proxy[i][0];
        prev_proxy[i][0] = proximities[i+1];
        proximities[i+1] = (y > 512) ? (((y>512)+prev_proxy[i][1]+prev_proxy[i][0])/3):0;


            if(proximities[17+i] > proximities[proximities[0]])  {
                proximities[0] = 17 + i;
            } /* if */

            if(proximities[17+i] > THRESHOLD) in_proximity = 1;

        counter++;
        ADSCR = counter;
    } /* for ->3 sensors connected directly to ADC */
} /* Sample HPM Sensors */



/****       Determine Triangulation Set   ******************************
Description: Determines the sensors in the absolute triangulation set */

int determine_triangulation_set(void) {

    int i, n, place, sum = 0, high = 0;


      HPM_temp[0].proximity = proximities[proximities[0]]; /* Store 1st prox &
      HPM_temp[0].ID_Number = proximities[0];                 ID in the
                                                              triang. set */

    n = proximities[0] - 1;

    for(i=1; i <= TC[n][0]; i++) {

        /***   Sum the highest proximity with that of 2 candidates.  Each
               group of 2 candidates will be summed in a circular queue
               fashion upon each iteration. */

sum = HPM_temp[0].proximity + proximities[ TC[n][i] ] + proximities[ TC[n][((i+1)%TC[n][0])] ];
```

```
        if(sum > high) {

            high = sum;     /* Make previous high equal to current sum */
            place = i;      /* Save place where high summation occurred */
        } /*if ->the current current sum is greater than the current high */
    } /* for */


      HPM_temp[1].proximity = proximities[TC[n][place]];/*Store 2nd proxy &*/
      HPM_temp[1].ID_Number = TC[n][place];             /* ID in tri. set */

    if(TC[n][place] != TC[n][0]) {
      HPM_temp[1].proximity=proximities[TC[n][place+1]];/*Store 2nd proxy &*/
      HPM_temp[1].ID_Number = TC[n][place+1];           /* ID in tri. set */
    } /* if */

    else {
      HPM_temp[1].proximity=proximities[TC[n][place-1]];/*Store 3rd proxy &*/
      HPM_temp[1].ID_Number = TC[n][place-1];            /*ID in tri. set */
    } /* else */

    return 1;
} /* Determine Triangulation Set */


/****       Binary Search     *********************************************
Description: Performs a binary search of the 'FP' array and returns the index
             closes to 'x'. */

short binary_search(float x) {

    unsigned char low = 0;
    unsigned char high = 19;
    unsigned char mid;
    float tmp;

    while(low <= high) {

        mid = (low + high) / 2;
        tmp = FP[mid];

        if ((tmp - x < 0.09) && (tmp - x > -0.09)) { return mid; }
            /* found!!! */

        else if(tmp < x) { low = mid + 1; }
            /* if ->current element is less than target */

        else if(tmp > x) { high = mid - 1; }
            /*else if ->current element is more than target */

        else { return mid; } /*else -> target found!  */

    } /* while ->'low' index is actually lower than the 'high' index */

    return -1;
```

```
} /* Binary Search */







/****       Absolute Value    *******************************************
Description: Returns the absolute value of a floating-point number. */

float abs_val(float a) {

    a = (a < 0) ? (-a):a;
    return a;
} /* Absolute Value */



/****       Multiplication    *******************************************
Description: Floating-point multiplication using the 'MLT' and 'FP' tables to
             approximate the calculation. */

float mult(float a, float b) {

    if((a < 0) && (b < 0))
      return MLT[(binary_search(abs_val(a)))][(binary_search(abs_val(b)))];

    else if(a < 0)
      return (-
MLT[(binary_search(abs_val(a)))][(binary_search(abs_val(b)))]);

    else if(b < 0)
      return (-
MLT[(binary_search(abs_val(a)))][(binary_search(abs_val(b)))]);

    else
      return MLT[(binary_search(abs_val(a)))][(binary_search(abs_val(b)))];

} /* Multiplication */


/****       Vector Subtract   **********************************************
Description: Performs vector subtraction between 2 3-dimensional vectors,
             where C = B - A */

void vector_subtract(float A[], float B[], float C[]) {

    int i;

    for(i=0; i < N; i++) {  C[i] = B[i] - A[i]; }

} /* Vector Subtract */


/****       Vector Magnitude   ********************************************
```

```
Description: Calculates the magnitude of a 3-dimensional vector and returns
             it as a 'float' value. */

float vector_magnitude(float A[]) {

    int i;
    float accumulator;

    for(i=0; i<N; i++) { accumulator += mult(A[i], A[i]); }

    return sqrt(accumulator);  /*Return square-root of the 'sum of squares'*/
} /* Vector Magnitude */



/****       Unit Vector-1 Coefficient     ***********************************
Description: Calculate coefficient for 1st unit vector */

void unit_vector1_coefficient() {

    float VM;    /* variable to store vector magnitude */
    float p1, p2, p3;

    p1 = (float) HPM_temp[0].proximity;     /*Store 1st sensor proximity*/
    p2 = (float) HPM_temp[1].proximity;     /*Store 2nd sensor proximity*/
    p3 = (float) HPM_temp[2].proximity;     /*Store 3rd sensor proximity*/

    /* Determine the distances associated with the observed proximities */
    D1 = TRIANGULATION - (((float)(p1 / (p1 + p2 + p3))) * TRIANGULATION);
    D2 = TRIANGULATION - (((float)(p2 / (p1 + p2 + p3))) * TRIANGULATION);

    /***   Subtract vector P1 from P2, P1_to_P2 = P2-P1    ***/
    vector_subtract(SC[HPM_temp[0].ID_Number-1], SC[HPM_temp[1].ID_Number-1], P1_to_P2);

    /* Calculate the magnitude of vector */
    VM = vector_magnitude(P1_to_P2);

    /*Calculate 1st unit vector coefficient w/ prior determined variables. */
    (float)( (mult(VM,VM) + mult(D1,D1) - mult(D2,D2)) / mult(2,VM) );
} /* Unit Vector-1 Coefficient */



/****       Unit Vector-2 Coefficient     *******************************
Description: Calculate coefficient for 1st unit vector */

void unit_vector2_coefficient() {

    uv2c = sqrt( abs_val(mult(D1,D1) - mult(uv1c,uv1c)) );

} /* Unit Vector-2 Coefficient */



/****       Scalar Multiply   *******************************************
Description: Multiplies a vector 'A' times a scalar 't'.
Where: result = t*A */
```

```
void scalar_multiply(float t, float A[], float SM[]) {

    int i;

    for(i=0; i < N; i++) {

      SM[i] = t * A[i]; /* Multiply each element of 'A' by 't' */

} /* Scalar Multiply */


/****       Unit Vector-1   ***********************************************
Description: Calculate the coordinates for the 1st unit vector (uv1) */

void unit_vector1() {

    int a;

    scalar_multiply(1/vector_magnitude(P1_to_P2), P1_to_P2, uv1 );

} /* Unit Vector-1 */


/****       Unit Vector-2   **********************************************
Description: Calculate the coordinates for the 2nd unit vector (uv2) */

void unit_vector2() {

    int a;
    float tmp1[3], tmp2[3];

    vector_subtract(SC[HPM_temp[0].ID_Number-1], SC[HPM_temp[2].ID_Number-1], P1_to_P3);

    /*Multiply 'P1_to_P2' by 0.5 */
    scalar_multiply(0.5, P1_to_P2, tmp1);

    /* Subtract [0.5*P1_to_P2] from P1_to_P3 */
    vector_subtract(tmp1, P1_to_P3, tmp2);

    /*Scalar Multiply */
   scalar_multiply( (float)(1/mult(vector_magnitude(P1_to_P3),(SQRT_3_DIV_2))), tmp2, uv2);

} /* Unit Vector-2 */


/****       Result Vector   *********************************************
Description: Calculate the coordinates for the result vector.  This vector is
             the magnet lies in 3-dimensional space */

void result_vector() {

    int i;
    int a;
    float tmp2[3], tmp3[3];



    /* Construct term with 1st unit vector with coefficient */
    scalar_multiply(uv1c, uv1, tmp2);

    /* Construct term with 2nd unit vector with coefficient */
    scalar_multiply(uv2c, uv2, tmp3);


    for(i=0; i < 3; i++) {

      result[i] = SC[HPM_temp[0].ID_Number-1][i] + tmp2[i] + tmp3[i];
    } /* Construct result vector */

} /* Result Vector */



/****       Coordinate Filter   *******************************************
Description: 3-point filter averaging the current and previous 2 coordinates
             calculated. */

void coordinate_filter(void) {

    int a;

    result[0] = (result[0] + X[1] + X[0])/3;
    result[1] = (result[1] + Y[1] + Y[0])/3;
    result[2] = (result[2] + Z[1] + Z[0])/3;

    X[1] = X[0];    X[0] = result[0];
    Y[1] = Y[0];    Y[0] = result[1];
    Z[1] = Z[0];    Z[0] = result[2];

} /* Coordinate Filter */



/****       Transmit Results   *******************************************
Description: Transmit results via RS-232 connection */

void transmit_results() {

    int i;

    Term1_SendChar('A');      /* Send Preamble character */

        for(i=0; i < 3; i++) {

            Term1_SendChar('F');    /* Send "Begin Float" character */
            Term1_SendFloatNum(result[i]); /* Send result component */
        } /* for */

        //***   Transmit sensor numbers ***
        Term1_SendChar('N');      /* Send "ID Number" character */
        Term1_SendChar( HPM_temp[0].ID_Number ); /* Send ID Number */
        Term1_SendChar( HPM_temp[1].ID_Number ); /* Send ID Number */
        Term1_SendChar( HPM_temp[2].ID_Number ); /* Send ID Number */
} /* Transmit Results */

/*
void transmit_results() {

    int i;
    unsigned short D_S;     /* Direction & Speed of joystick simulation */
    unsigned char SB[2];    /*Buffer for bytes to send during transmission*/

    for(i=0; i < 2; i++) {

        D_S = (int)(((result[0] + 3.7198) / 7.4396) * 4096);

        SB[0] = (D_S >> 8) & (0xFF);
        SB[0] |= (0x10);
        SB[1] = D_S & (0xFF);
        Term1_SendChar(SB[0]);
        Term1_SendChar(SB[1]);

        D_S = (int)(((result[2] + 2.2480) / 4.4960) * 4096);

        SB[0] = (D_S >> 8) & (0xFF);
        SB[0] |= (0x20);
        SB[1] = D_S & (0xFF);
        Term1_SendChar(SB[0]);
        Term1_SendChar(SB[1]);
    } /* for */

} /* Transmit Results */
*/
```

C.3 C HPM CONFIGURATION – HPM_STATIC_VARIABLES.C

```
/****************************************************************************
    Head Position Monitor (HPM)
    Static Variables & Structure Definitions / Library (header file) includes
    Alex J. Bevly III
****************************************************************************/

//***   Includes   **********************************************************
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "HPM_Lookup.h"
//***   Includes   **********************************************************

//***   Definitions   *******************************************************
#define HEAD_ARRAY_SIZE 19    /*Number of sensors in HPM Head array */
#define N 3                   /*Number of dimensions calculations are based*/
#define TRIANGULATION 1.8     /*Approximate distance sensor stops detecting*/
#define ARB_DELAY 4           /*Arbitrary delay for sampling sensors
#define THRESHOLD 25          /*Minimum ADC value to track head position
typedef struct sensor_sample HPM_sort;
//***   Definitions   *******************************************************

//***   Variables   *********************************************************
struct sensor_sample {        //Structure to store temporary sensor
information
    unsigned char ID_Number;    //unique number identifying sensor
    unsigned int proximity;
};

HPM_sort HPM_temp[3] = { {0,0}, {0,0}, {0,0} }; /*Sensor structure array */

unsigned short proximities[HEAD_ARRAY_SIZE+1];  /*ADC values for sensors*/
unsigned short prev_proxy[HEAD_ARRAY_SIZE][2];  /*Filter for sensor values*/

float uv1[N] = {0.0, 0.0, 0.0};     /* 1st Unit Vector */
float uv1c = 1.0f;                  /* 1st Unit Vector Coefficient */
float uv2[N];                       /* 2nd Unit Vector */
float uv2c = 5.45f;                 /* 2nd Unit Vector Coefficient */
float result[N] = {0.0, 0.0, 0.0};  /* Result Vector of */

float P1_to_P2[N];   /* Vector between P2 & P1 where: {P1_to_P2 = P2 - P1} */
float P1_to_P3[N];   /* Vector between P3 & P1 where: {P1_to_P3 = P3 - P1} */
float D1 = 1.62f;                   /* Magnet distance w.r.t. P1 */
float D2 = 0.45f;                   /* Magnet distance w.r.t. P2 */
float X[2];                         /* buffer to average X coordinates */
float Y[2];                         /* buffer to average Y coordinates */
float Z[2];                         /* buffer to average Z coordinates */

unsigned char initialized = 0;      /* flag to denote initialization */
unsigned char in_proximity = 0;     /* flag to denote magnet in proximity */
```

C.4 C HPM ROM VARIABLES – HPM_LOOKUP.C

```
#pragma INTO_ROM
unsigned char const TC[19][7] = {
      { 5, 2, 4, 5, 18, 17 },
      { 3, 3, 4, 1 },
      { 3, 6, 4, 2 },
      { 6, 1, 2, 3, 6, 7, 5 },
      { 6, 1, 4, 7, 8, 19, 18 },
      { 3, 7, 4, 3 },
      { 4, 8, 5, 4, 6 },
      { 3, 19, 5, 7 },
      { 5, 17, 18, 13, 12, 10 },
      { 3, 9, 12, 11 },
      { 3, 10, 12, 14 },
      { 6, 9, 13, 15, 14, 11, 10 },
      { 6, 18, 19, 16, 15, 12, 9 },
      { 3, 11, 12, 15 },
      { 4, 14, 12, 13, 16 },
      { 3, 15, 13, 19 },
      { 3, 1, 18, 9 },
      { 6, 1, 5, 19, 13, 9, 17 },
      { 5, 16, 13, 18, 5, 8 }
}; //Triangulation Candidates



#pragma INTO_ROM
float const SC[19][3] = {

      { 1.5413,  3.9609, 1.4346  },
      { 2.6835,  2.9549, 1.6179  },
      { 3.7198,  2.3870, 0.2601  },
      { 2.7288,  3.8780, 0.0733  },
      { 1.3624,  5.0058, -0.2443 },
      { 3.6127,  3.2039, -1.3425 },
      { 2.5512,  4.6432, -1.5463 },
      { 1.3643,  5.5625, -1.9560 },
      { -1.5413, 3.9609, 1.4346  },
      { -2.6835, 2.9549, 1.6179  },
      { -3.7198, 2.3870, 0.2601  },
      { -2.7288, 3.8780, 0.0733  },
      { -1.3624, 5.0058, -0.2443 },
      { -3.6127, 3.2039, -1.3425 },
      { -2.5512, 4.6432, -1.5463 },
      { -1.3643, 5.5625, -1.956  },
      { 0.0000,  3.5106, 2.2480  },
      { 0.0000,  4.6541, 0.8579  },
      { -0.0097, 5.2937, -0.8246 }
```

```
}; //Sensor Coordinates



#pragma INTO_ROM
float const FP[21] = {
0.0000, 0.1000, 0.2000, 0.3000, 0.4000, 0.5000, 0.6000, 0.7000, 0.8000, 0.9000,
1.0000, 1.1000, 1.2000, 1.3000, 1.4000, 1.5000, 1.6000, 1.7000, 1.8000, 1.9000, 2.0000
};



#pragma INTO_ROM
float const MLT[21][21] = {
{ 0.0000,  0.0000,  0.0000,  0.0000,  0.0000,  0.0000,  0.0000,  0.0000,  0.0000,  0.0000,
  0.0000,  0.0000,  0.0000,  0.0000,  0.0000,  0.0000,  0.0000,  0.0000,  0.0000,  0.0000,
  0.0000, },

{ 0.0000,  0.0100,  0.0200,  0.0300,  0.0400,  0.0500,  0.0600,  0.0700,  0.0800,  0.0900,
  0.1000,  0.1100,  0.1200,  0.1300,  0.1400,  0.1500,  0.1600,  0.1700,  0.1800,  0.1900,
  0.2000, },

{ 0.0000,  0.0200,  0.0400,  0.0600,  0.0800,  0.1000,  0.1200,  0.1400,  0.1600,  0.1800,
  0.2000,  0.2200,  0.2400,  0.2600,  0.2800,  0.3000,  0.3200,  0.3400,  0.3600,  0.3800,
  0.4000, },

{ 0.0000,  0.0300,  0.0600,  0.0900,  0.1200,  0.1500,  0.1800,  0.2100,  0.2400,  0.2700,
  0.3000,  0.3300,  0.3600,  0.3900,  0.4200,  0.4500,  0.4800,  0.5100,  0.5400,  0.5700,
  0.6000, },

{ 0.0000,  0.0400,  0.0800,  0.1200,  0.1600,  0.2000,  0.2400,  0.2800,  0.3200,  0.3600,
  0.4000,  0.4400,  0.4800,  0.5200,  0.5600,  0.6000,  0.6400,  0.6800,  0.7200,  0.7600,
  0.8000, },

{ 0.0000,  0.0500,  0.1000,  0.1500,  0.2000,  0.2500,  0.3000,  0.3500,  0.4000,  0.4500,
  0.5000,  0.5500,  0.6000,  0.6500,  0.7000,  0.7500,  0.8000,  0.8500,  0.9000,  0.9500,
  1.0000, },


{ 0.0000,  0.0600,  0.1200,  0.1800,  0.2400,  0.3000,  0.3600,  0.4200,  0.4800,  0.5400,
  0.6000,  0.6600,  0.7200,  0.7800,  0.8400,  0.9000,  0.9600,  1.0200,  1.0800,  1.1400,
  1.2000, },

{ 0.0000,  0.0700,  0.1400,  0.2100,  0.2800,  0.3500,  0.4200,  0.4900,  0.5600,  0.6300,
  0.7000,  0.7700,  0.8400,  0.9100,  0.9800,  1.0500,  1.1200,  1.1900,  1.2600,  1.3300,
  1.4000, },

{ 0.0000,  0.0800,  0.1600,  0.2400,  0.3200,  0.4000,  0.4800,  0.5600,  0.6400,  0.7200,
  0.8000,  0.8800,  0.9600,  1.0400,  1.1200,  1.2000,  1.2800,  1.3600,  1.4400,  1.5200,
  1.6000, },

{ 0.0000,  0.0900,  0.1800,  0.2700,  0.3600,  0.4500,  0.5400,  0.6300,  0.7200,  0.8100,
  0.9000,  0.9900,  1.0800,  1.1700,  1.2600,  1.3500,  1.4400,  1.5300,  1.6200,  1.7100,
  1.8000, },

{ 0.0000,  0.1000,  0.2000,  0.3000,  0.4000,  0.5000,  0.6000,  0.7000,  0.8000,  0.9000,
  1.0000,  1.1000,  1.2000,  1.3000,  1.4000,  1.5000,  1.6000,  1.7000,  1.8000,  1.9000,
  2.0000, },

{ 0.0000,  0.1100,  0.2200,  0.3300,  0.4400,  0.5500,  0.6600,  0.7700,  0.8800,  0.9900,
  1.1000,  1.2100,  1.3200,  1.4300,  1.5400,  1.6500,  1.7600,  1.8700,  1.9800,  2.0900,
  2.2000, },

{ 0.0000,  0.1200,  0.2400,  0.3600,  0.4800,  0.6000,  0.7200,  0.8400,  0.9600,  1.0800,
  1.2000,  1.3200,  1.4400,  1.5600,  1.6800,  1.8000,  1.9200,  2.0400,  2.1600,  2.2800,
  2.4000, },

{ 0.0000,  0.1300,  0.2600,  0.3900,  0.5200,  0.6500,  0.7800,  0.9100,  1.0400,  1.1700,
  1.3000,  1.4300,  1.5600,  1.6900,  1.8200,  1.9500,  2.0800,  2.2100,  2.3400,  2.4700,
  2.6000, },
    { 0.0000,  0.1400,  0.2800,  0.4200,  0.5600,  0.7000,  0.8400,  0.9800,  1.1200,  1.2600,
      1.4000,  1.5400,  1.6800,  1.8200,  1.9600,  2.1000,  2.2400,  2.3800,  2.5200,  2.6600,
      2.8000, },

    { 0.0000,  0.1500,  0.3000,  0.4500,  0.6000,  0.7500,  0.9000,  1.0500,  1.2000,  1.3500,
      1.5000,  1.6500,  1.8000,  1.9500,  2.1000,  2.2500,  2.4000,  2.5500,  2.7000,  2.8500,
      3.0000, },

    { 0.0000,  0.1600,  0.3200,  0.4800,  0.6400,  0.8000,  0.9600,  1.1200,  1.2800,  1.4400,
      1.6000,  1.7600,  1.9200,  2.0800,  2.2400,  2.4000,  2.5600,  2.7200,  2.8800,  3.0400,
      3.2000, },

    { 0.0000,  0.1700,  0.3400,  0.5100,  0.6800,  0.8500,  1.0200,  1.1900,  1.3600,  1.5300,
      1.7000,  1.8700,  2.0400,  2.2100,  2.3800,  2.5500,  2.7200,  2.8900,  3.0600,  3.2300,
      3.4000, },

    { 0.0000,  0.1800,  0.3600,  0.5400,  0.7200,  0.9000,  1.0800,  1.2600,  1.4400,  1.6200,
      1.8000,  1.9800,  2.1600,  2.3400,  2.5200,  2.7000,  2.8800,  3.0600,  3.2400,  3.4200,
      3.6000, },

    { 0.0000,  0.1900,  0.3800,  0.5700,  0.7600,  0.9500,  1.1400,  1.3300,  1.5200,  1.7100,
      1.9000,  2.0900,  2.2800,  2.4700,  2.6600,  2.8500,  3.0400,  3.2300,  3.4200,  3.6100,
      3.8000, },

    { 0.0000,  0.2000,  0.4000,  0.6000,  0.8000,  1.0000,  1.2000,  1.4000,  1.6000,  1.8000,
      2.0000,  2.2000,  2.4000,  2.6000,  2.8000,  3.0000,  3.2000,  3.4000,  3.6000,  3.8000,
      4.0000, }
};
```

C.5 C HPM PROGRAM FILE – HPM.PRM

```
NAMES

END

SECTIONS
      ROM  =  READ_ONLY                  0xC000 SIZE 0x3E00;
      ZPAGE  =  READ_WRITE               0x0040 SIZE 0x00C0;
      RAM  =  READ_WRITE                 0x0100 SIZE 0x0340;
END

PLACEMENT
      DEFAULT_RAM                        INTO  RAM;
      DEFAULT_ROM, ROM_VAR, STRINGS      INTO  ROM;
      _DATA_ZEROPAGE                     INTO  ZPAGE;
END

INIT _EntryPoint   /* The entry point of the application. This function is
                  generated into the CPU module. */

STACKSIZE 0x0246  /* Size of the system stack. Value can be changed on the
"Build options" tab */
```

APPENDIX D

D.1 MATLAB RANDOM TEST POINT GENERATION – TEST_POINTS.M

```
%Generation of test points inside each triplet using Barycentric
%coordinates to confirm the points are in fact inside the triangle.


clear all
clf

x = [   1.5413  0.0000  0.0000
        1.5413  2.6835  2.7288
        2.6835  3.7198  2.7288
        1.5413  2.7288  1.3624
        1.5413  1.3624  0.0000
        1.3624  0.0000  -0.0097
        1.3624  1.3643  -0.0097
        1.3624  2.5512  1.3643
        2.7288  1.3624  2.5512
        2.7288  3.6127  2.5512
        3.7198  2.7288  3.6127  ]; %% x coordinates of the triangle vertices.


z = [   1.4346  2.2480  0.8579
        1.4346  1.6179  0.0733
        1.6179  0.2601  0.0733
        1.4346  0.0733  -0.2443
        1.4346  -0.2443  0.8579
        -0.2443  0.8579  -0.8246
        -0.2443  -1.9560  -0.8246
        -0.2443  -1.5463  -1.9560
        0.0733  -0.2443  -1.5463
        0.0733  -1.3425  -1.5463
        0.2601  0.0733  -1.3425 ]; %% z coordinates of the triangle vertices.

points = 1;

for i = 1:11

    yesno = 0;  %Initialize Barycentric condition
    testPointCount = 0;
    i
    sortX = sort( x(1,:), 2 );
    sortZ = sort( z(1,:), 2 );
    while (yesno ~= 1) || (testPointCount ~= 3)

        randX = rand(1) * 4;
        randZ = (rand(1) * 4.5) - 2;
        point = [ randX randZ ]    %% The random point

        yesno = inside_Tri(point, x(i,:), z(i,:))%% Is the point inside the
                                                  %% triangle?

        if yesno == 1
            pointX(points) = point(1);
            pointY(points) = point(2);
            points = points + 1;
            testPointCount = testPointCount + 1;
        end %if

    end %while

end %for

      %%  Plot the triangles
      for i = 1:11
            plot( [ x(i,:), x(i, 1)], [ z(i,:) z(i, 1) ], 'b' )
            hold on
      end %for

      %%  Plot the test points
      for i = 1:33
            %plot( pointX(i), pointY(i), 'ro' )
            plot( TP(i,1), TP(i,3)+1, 'ro');
        hold on
      end %for

      %%  Plot the sensor points
      for i = 1:11
            plot( x(i,1), z(i,1), 'kh' )
            hold on
            plot( x(i,2), z(i,2), 'kh' )
            hold on
            plot( x(i,3), z(i,3), 'kh' )
            hold on
      end %for

    axis equal


% ***   Write Random Test Points to 'TestPoints.txt'    ***
fid = fopen('TestPoints.txt', 'w');

fprintf(fid, '\tRANDOM TEST POINTS FOR HPM\n');
fprintf(fid, '\t==========================\n\n');

fprintf(fid, '\t   X\t\t   Z\n');
fprintf(fid, '\t==========================\n');

      for i = 1:33
            fprintf(fid, '%d)\t%7.4f\t%7.4f\n', i, pointX(i), pointY(i) );
      end %for
fclose(fid);
```

D.2 MATLAB TEST POINT ANGLE CALCULATION – TP_MILL_&_ANGLES.M

```
%Script to generate the angles to rotate the HPM in its horizontal and
%vertical directions.  Lastly, the coordinates for the manual mill will
%also be determined for positioning after the HPM has been rotated.

clear all
%clf
format long

N_h = 2;    %Degree of horizontal polynomial
N_v = 2;    %Degree of vertical polynomial

h = 0.0001   %Differentiation factor

horizontal = [   1.06299213         0.00000000
                 1.05185039         0.43673228
                 0.94488189         1.07503937
                 0.78740157         1.49606299
                 0.54952756         1.88862205
                 0.20610236         2.34846457
                 -0.24858268        2.82992126
                 -0.80708661        3.28460630
                 -1.41208661        3.64570866
                 -1.98811024        3.95330709
                 -2.52208661        4.18062992 ]; %Horiz [y,x] coordinates



vertical = [    2.55905512      -4.72440945
                2.48031496      -3.93700787
                2.32283465      -3.1496063
                2.12598425      -2.36220472
                1.8503937       -1.57480315
                1.57480315      -0.90551181
                1.06299213       0.00000000
                0.47244094       0.78740157
                -0.39370079      1.57480315
                -0.78740157      1.8503937    ]; %Vertical [y,z] coordinates


Test_Points = [         0.3993   0.3907     0.6913
                        1.2284   0.4257     0.4530
                        0.1212   0.2202     0.8897
                        2.3334  -0.2017     0.3533
                        1.7650   0.1537     0.4440
                        2.4610   0.0131    -0.1228
                        3.1762  -0.7218    -0.2345
                        3.5786  -1.1261    -0.7083
```

```
                                3.4414  -0.8447    -0.7809
                                2.3160   0.5958    -0.9363
                                1.9755   0.5600    -0.3734
                                1.7481   0.6141    -0.2046
                                0.9191   1.0899     0.6022
                                1.0704   1.0973    -0.4769
                                1.2346   0.5493     0.2862
                                1.0245   1.4109  -1.0690
                                1.3134   1.3977  -1.2644
                                0.1122   1.5573  -1.1587
                                0.7736   1.8572  -2.0783
                                1.1468   1.9551  -2.7150
                                1.0989   1.8975  -2.4446
                                1.7031   1.5706  -2.2991
                                1.4803   1.3655  -1.3754
                                1.5382   1.5761  -2.0171
                                2.5184   0.4939  -1.1141
                                2.4827   0.6414  -1.3924
                                1.9628   1.0107  -1.2610
                                2.6957   0.7761  -2.3476
                                2.9665   0.2939  -1.8218
                                3.1765   0.0102  -1.8023
                                3.3986  -0.4846  -1.4367
                                3.0096   0.0007  -1.2029
                                3.3806  -0.3352  -1.7468 ];%test points coordinates


%Calculate the angle of rotation along Vertical (z) & Horizontal (x) axes of
%HPM for Test Points

dimensions = size(Test_Points);     %Get dimensions of of Test_Points matrix
num_TP = dimensions(1);             %Get number of test points in HPM

%Next statement performs multiple operations, ultimately to determine angle
%of rotation in
%Vertical (z) direction:
      %1) Create boundaries for vertical spline with segments of ‘h’.
      %2) Create a vertical spline with boundary created in [1].
      %3) Differentiate between each segment on the vertical boundary.
      %4) Determine tangent angle on every segment of vertical spline.

      z = floor(min(vertical(:,2)) ) : h : ceil( max(vertical(:,2)));   %(1)
      y_z = spline(vertical(:,2), vertical(:,1), z);  %(2)
      y_z_Diff = diff(y_z) / h;     %(3)
      Angle_V = atan(y_z_Diff);     %(4)


      [r c] = size(horizontal);
      before_rot = [ horizontal(:,2)  horizontal(:,1)    zeros(r, 1)  ]

      Origin_wrt_BJ = [ 0.0   -4.275  3.482 ];  % SolidWorks origin with
                                                %  respect to Ball Joint
      Ref_wrt_BJ = [ 0.0 -1.8403 -1.0 ];        % Reference for Mill with
                                                &  respect to Ball Joint

        for i = 1:num_TP
        foo = find(z<=(Test_Points(i,3)));
        phi = Angle_V( foo(end) );

        R_x = [ 1       0          0
                0       cos(phi)  sin(phi)
                0      -sin(phi)  cos(phi)  ];  %Rotation matrix about
                                                % x-axis by angle 'phi'

        after_rot = R_x * before_rot';      %Rotate in vertical
        after_rot = after_rot';             %Transpose matrix

        x = floor(min(after_rot(:,1))):h:ceil(max(after_rot(:,1)));  %(1)
        y_x = spline(after_rot(:,1), after_rot(:,2), x);  %(2)
        y_x_Diff = diff(y_x) / h;   %(3)
        Angle_H = atan(y_x_Diff);   %(4)

        foo = find( x<=(Test_Points(i,1)) );      %Retrieve test point &
        theta = -Angle_H( foo(end-1) );           % math with angle

        TP_Angles(i,:) = [ theta, phi ]; % [ horizontal, vertical ]


        %Finding test point coordinates with respect to Ball Joint
        TP_wrt_BJ(i,:) = Test_Points(i, :) + Origin_wrt_BJ;


            R_x = [ 1       0          0
                    0       cos(phi)  sin(phi)
                    0      -sin(phi)  cos(phi)  ]; %Rotation matrix
                                                   % about x-axis by
                                                   % angle 'phi'

            R_z = [  cos(theta)     sin(theta)  0
                    -sin(theta)     cos(theta)  0
                     0              0           1  ]; %Rotation matrix
                                                      % about z-axis by
                                                      % angle 'theta'


        %Rotate test point in vertical direction by the ball joint, then
        % rotate in the horizontal direction along the vertical axis.
        TP_rot_BJ_vertical(i,:) = TP_wrt_BJ(i,:) * R_x;
        TP_rot_BJ_horizontal(i,:) = TP_rot_BJ_vertical(i,:) * R_z;

        %Assign mill coordinates as the ‘x’ horizontal component & the
        % ‘z’ vertical component.
        Mill_Coord(i,:)=[TP_rot_BJ_horizontal(i,1),(TP_rot_BJ_vertical(i,3)+1)];
    end


%Create file with test point coordinates
fid = fopen('Test_Point_Order.txt', 'w');

  fprintf(fid, '\t\t\tHPM TEST POINTS\n');
  fprintf(fid, '\t\t\t===============\n\n\n');

for i = 1:num_TP
```

```
  fprintf(fid, '%d)\tX: %7.4f\t\tY: %7.4f\t\tZ: %7.4f\n',i, Test_Points(i,1),
        Test_Points(i,2), (Test_Points(i,3)+1) );
  fprintf(fid, '-------------------------------------------------------\n');
end

fclose(fid);


%Create file to reference mill coordinates & angles for characterization
  fid = fopen('HPM_Rotation_&_Mill_Coordinates.txt', 'w');

  fprintf(fid, '\t\tHPM TESTPOINT ANGLES & MILL COORDINATES\n');
  fprintf(fid, '\t\t=======================================\n\n\n');

  fprintf(fid, '\tHORIZONTAL\tVERTICAL\t(mill -->)\t LATERAL\t\tFORWARD\n');
  fprintf(fid, '\t==========\t========\t          \t =======\t\t=======\n\n');


for i = 1:num_TP

    theta = TP_Angles(i,1) * (180/pi);
    phi = -TP_Angles(i,2) * (180/pi);
    lat = Mill_Coord(i,1);
    up = Mill_Coord(i,2);

   fprintf(fid, '%d)\t%7.2f\t%7.2f\t\t\t%7.4f\t\t\t%7.4f\n',i, theta, phi, lat, up);
fprintf(fid, '------------------------------------------------------------------------\n');
end

fclose(fid);
```

BIBLIOGRAPHY

1. Centre for Neuro Skills (CNS), Retrieved November 4, 2004, from: http://www.neuroskills.com/tbi/epidemiology.shtml.

2. Center for Disease Control and Prevention (CDC), Retrieved November 4, 2004, from http://www.cdc.gov/doc.do/id/0900f3ec8001012b.

3. Defense and Veterans Brain Injury Center (DVBIC), Retrieved May 20, 2005, from http://www.dvbic.org/pdfs/DVBIC_Fact_Sheet_2003.pdf.

4. Keren O.; Reznik J.; Groswasser Z.; *Combined motor disturbances following severe traumatic brain injury: an integrative long-term treatment approach*. [Case Reports. Journal Article] Brain Injury. 15(7):633-8, July 2001.

5. Niemeier J, *The Lighthouse Strategy: use of a visual imagery technique to treat visual inattention in stroke patients*, *Brain Injury*, vol.12, no. 5, pp. 399-406, 1998.

6. Karnath H, Neimeier, and Dichgans, *Space exploration in neglect*, *Brain*, vol. 121, no. Dec, pp. 2357-2367, 1998.

7. Schindler I and Kerkhoff G, *Head and trunk orientation modulate visual neglect*, *Neuroreport*, vol. 18, no. 8, pp. 2681-2685, 1997.

8. Niemann, Hendrik 1; Ruff, Ronald M. 1,2; Baser, Christine A. 1, *Computer-Assisted Attention Retraining in Head-Injured Individuals: A Controlled Efficacy Study of an Outpatient Program.* Journal of Consulting & Clinical Psychology. 58(6):811-817, December 1990.

9. Kirsch, Ned L. 1,5; Shenton, Michelle 1; Spirl, Erin 1; Rowan, James 1; Simpson, Rich 2; Schreckenghost, Debra 3; LoPresti, Edmund F. 4, *Web-Based Assistive Technology Interventions for Cognitive Impairments After Traumatic Brain Injury: A Selective Review and Two Case Studies.* Rehabilitation Psychology. 49(3):200-212, August 2004.

10. Brienza, D.M., Angelo, J, *A force feedback joystick and control algorithm for wheelchair obstacle avoidance*. Disability & Rehabilitation. 18(3):123-9, 1996 Mar.

11. LoPresti, E.F., Brienza, D.M., Angelo, J., Gilbertson, L., *Neck range of motion and use of computer head controls*. Journal of Rehabilitation Research and Development. Volume 40, No. 3, May/June 2003, pp. 199-212.

12. Melexis Microelectronic Integrated Systems, Retrieved October 25, 2004, from http://www.melexis.com/prodfiles/MLX90215_Rev007.pdf.

13. C.J. Georgopoulos, *Suppressing background-light interference in an in-house infrared communication system by optical filtering*, Internat. J. Optoelectronics 3(3), 1988.

14. Halliday David; Resnick, Robert; Walker, Jearl; *Fundamentals of Physics, Fifth Edition*, Volume 1, pp. 428-431.

15. Force Field, Retrieved October 25, 2004, from http://www.wondermagnets.com/cgi-bin/edatcat/WMSstore.pl?user_action=detail&catalogno=0063.

16. Stewart, James; *Calculus: Concepts and Contexts*, pp. A31.

17. Maxim Integrated Products, Retrieved October 25, 2004, from: http://pdfserv.maxim-ic.com/en/ds/DG406-DG407.pdf.

18. Motorola Inc. subsidiary Freescale Semiconductor Inc., Retrieved October 25, 2004, from http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC68HC908GZ16.pdf.

19. Maxim Integrated Products, Retrieved October 25, 2004, from http://pdfserv.maxim-ic.com/en/ds/MAX3222-MAX3241.pdf.

20. Anton, Howard; Busby, Robert C.; *Contemporary Linear Algebra*, pp. 3, 16-17, 289-291.

21. Burden, Richard L.; Faires, J. Douglas; *Numerical Analysis, Sixth Edition*, pp.143-150.