TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technology

Nika Mukbaniani 165622

EARTH MAGNETIC FIELD SIMULATOR

Master's Thesis

Supervisor: Ants Koel

PhD

Eiko Priidel

MCs

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Nika Mukbaniani 165622

MAA MAGNETVÄLJA SIMULAATOR

Magistritöö

Juhendaja: Ants Koel
Doktorikraad

Eiko Priidel

Magistrikraad

Tallinn 2019

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Nika Mukbaniani

05.05.2019

Abstract

The aim of this task was to develop an earth magnetic field simulator for testing a cube satellite. Simulating earth magnetic field is needed to test and develop the satellites control algorithm and to test the functionality of hardware and software. The work describes the process of calculating necessary parameters for creating earth magnetic field by Helmholtz Cage according to satellites dimensions and magnetic field in Tallinn. It also describes the analysis for determining the best solution to control the coils. With the Helmholtz Cage, the electromagnetic field can be cancelled. Therefore, in-orbit magnetic field magnitudes and direction of the magnetic field in the orbit can be simulated. The major goals of this research are to design and build a Helmholtz cage capable of generating a magnetic field to test an attitude determination algorithm for the nanosatellite. Minor objectives represent dynamically simulating magnetic field which satellite would experience in the orbit by utilizing predicted geomagnetic field data from the IGRF model as well as analysing the performance of the ADCS of nanosatellite. The magnetic field simulation will be provided by MATLAB script. which the geomagnetic field report from the sketch, checks the ambient magnetic field in the cage with the truth magnetometer, calculates the current required in each coil to obtain the geomagnetic field from the sketch, commands will give to coils as a current and then verifies that the desired magnetic field is achieved before proceeding to the next desired magnetic field. The algorithm will be implemented on the Arduino open source platform.

This thesis is written in English and is 58 pages long, including 11 chapters, 17 figures and 17 tables.

Annotation

Lõputöö eesmärgiks oli arendada maa magnetvälja simulaator, millega oleks võimalik testida kuubik-satelliiti. Maa magnetvälja simuleerimine on vajalik, et testida ja arendada satelliitide juhtimiseks vajaliku algoritmi, mis aitab testida riistvara ja tarkvara funktsionaalsust. Täpsemalt kirjeldatakse Helmholtz kasti poolt magnetvälja loomiseks vajalike parameetrite arvutamise protsessi vastavalt satelliidide mõõtmetele ja magnetväljale Tallinnas. Ühtlasi analüüsiti antud töö raames, kuidas hinnata parimat lahendust poolide jaoks. Helmholtz kastiga saab saavutada olukorda, kus elektromagneetiline väli on tühistatud, mistõttu saab simuleerida magnetvälja suurusi ja magnetvälja suunda orbiidil. Selle töö põhieesmärk on kujundada ja ehitada Helmholtzi puur, mis suudab tekitada magnetvälja suvalises suunas ja katsetada nanosatelliidi asukoha määramise algoritmi. Sekundaarsed eesmärgid sisaldavad dünaamiliselt simuleerivat geomagnetilist välja, mis satelliidi orbiidil tekiks, kasutades prognoositud geomagnetilised välja IGRF-mudeit. Sellega ühtlasi analüüsitakse nanosatelliidi ADCS toimivust. Magneetilise välja simulatsiooni tegemiseks kasutati Matlabi skripti, mis kontrollis kasti juures ümbritsevat magnetvälja magnetomeetriga ning arvutas igal joonisel nõutava voolu, et saada graafik geomagnetiline väli. Käsud, mida on koodis kasutatud annavad poolide voolu, ja seejärel kontrollivad, et enne soovitud magnetvälja jätkamist saavutatakse soovitud magnetväli. Algoritm implementeeri Arduino avatud lähtekoodiga platvormil.

Lõputöö on kirjutatud Inglise keeles ning sisaldab teksti 58 leheküljel, 11 peatükki, 17 joonist, 17 tabelit.

List of abbreviations and terms

ADCS	Attitude determination control system
nT	Nanotesla
IDE	Integrated development environment
IGRF	International geomagnetic reference field
SPDT	Single pole, double throw
SWG	Standard wire gauge
I2C	Inter-Integrated communication
LCD	Liquid crystal display
USB	Universal serial bus
GUI	Graphical user interface
PCB	Printed circuit board
AWG	American wire gauge

Table of contents

List of figures

List of tables

1 Introduction

The term nanosatellite stands for a 10x10x10 cm cube-satellite with a weight not exceeding 1kg. Besides size and weight, there might be structural, electrical and operational requirements. Two and three times more satellites with double and triple configurations could also be used. It's very important to know the satellite's orientation in space, to accomplish most of the satellite missions. For instance, a camera, sun sensor, or some other kind of orientation-specific equipment, which helps to definite orientation. As well as, knowing entire orientation is necessary to reduce the spinning of the satellite after launching into space. In order, to achieve the desired orientation and reduce spinning, the satellite control algorithm needs to know the orientation of the satellite. It is necessary to apply some amount of force to change the orientation. The name of the whole system, which involves perceiving and changing the orientation, is the Attitude Determination Control System, an abbreviation is ADCS. [1]

The satellites use different methods to change their orientation:

- Jet engine,
- Ion motor, [2]
- Flywheel,
- Magnetic coils.

The reactive motor achieves thrust through fuel combustion. The disadvantage of a reactive engine is that it needs fuel and respectively affecting on satellite's dimensions. The ion motor generates thrust-induced ion acceleration. The flywheel turns the satellite on the basis, of the momentum restraint law. In the framework of this project, to generate a magnetic field of the satellite are used magnetic coils, and due to the magnetic interaction, the satellite acts as a diaphragm in the earth's magnetic field and allows the satellite to change its orientation. The satellite needs to simulate the magnetic field in orbit to develop and test control algorithms and test hardware and software. The magnetic field is most easily simulated using Helmholtz cage. With three Helmholtz coils, it is

possible to create a three-dimensional controlled magnetic field. The advantage of simulating a three-dimensional magnetic field is the ability to leave the test equipment stationary and only change the magnetic field through the coil driver. The main purpose of this project is to create a magnetic calibration device, as is a Helmholtz Cage, that can generate a magnetic field and is used to calibrate the magnetic sensors on satellites. It should be able to counteract the Earth's local magnetic fields and create a magnetic environment which will like space orbit. This will allow the satellite to calibrate and verify magnetometers and Attitude Determination Control System (ADCS). The device should also be easy to transport between different locations.

2 Earth Magnetic Field

IGRF - International Geomagnetic Reference Field (International Geomagnetic Model) [3] IGRF is a standard mathematical description of the magnetic field of the earth and its changes after a while. The model information comes from observatories, ships, airplanes and satellites all over the globe. According to the IGRF model, it is possible to calculate the magnetic vectors and extract the total strength on the ground (sea level) and on the orbit. Example of total field intensity is shown in figure 1.

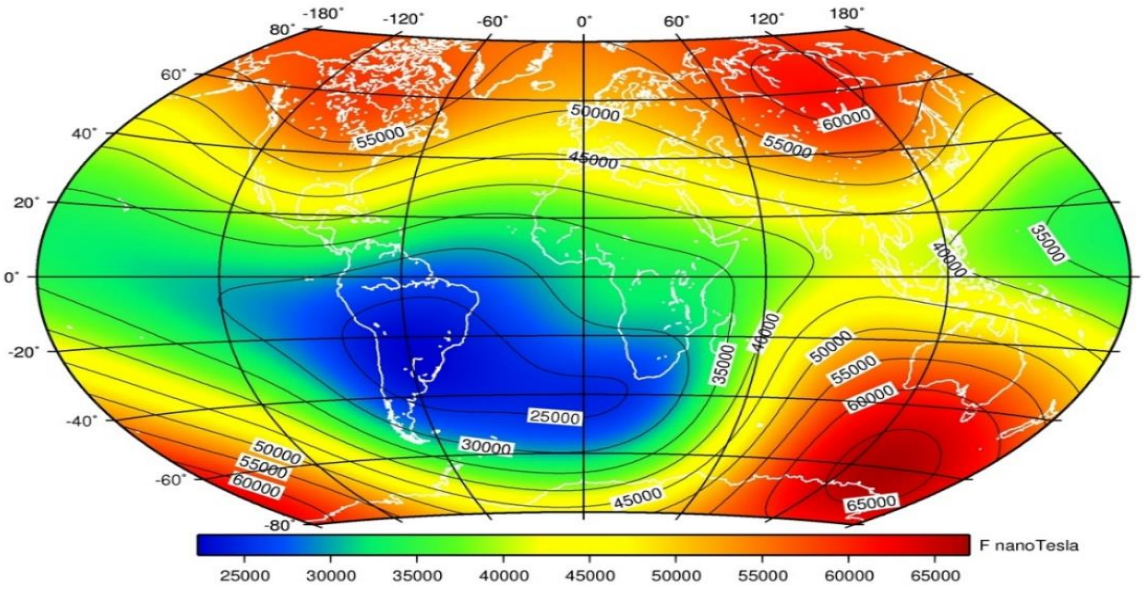


Figure 1 Total Field Intensity [4]

The Cube satellites will generally be launched at orbital heights 450 km and 650 km. Because it would be too expensive to send Satellites alone to orbit and usually small satellites are launched together with one launcher into space. For example, the International Space Station ISS [5] is 450km high and can provide cube satellites to supply ISS and Earth observation satellites are 650km above orbit, where large cube satellites are sent along with major satellites. The orbital altitude also determines the life of the satellite, the lower the orbit, the lower the lifetime, because of gravitational vibrations, space winds, and solar radiation.

As the magnetic field is changing over time and for the future it must be considered, it is imperative to know how big deviations are because the simulator should be capable to

generate respectively to those changes. The maximum magnetic vectors identified in the range from 04.05.2016 to 31.12.2019. The necessary information for this was obtained from a calculator based on the NOAA website. [6]

The magnetic field test site must be kept in mind. Testing is planned to be done in Tallinn, so the device must be able to generate a magnetic field that can compensate the already existing magnetic field in Tallinn. In the 2016-2019 range, the maximum vertical strength of the magnetic field is 49781.8 nT and orbit at 450 km has a maximum magnitude of 44325 nT in the same years. By joining the magnetic fields in Tallinn and 450 km altitude, we get a magnetic field of 94106.8 nT,

Table 1 shows the maximum magnetic component vectors found between 04.05.2016 and 31.12.2019 at 450km in orbit and in the sea at the sea level in Tallinn.

Axis	Sea level	Orbit 450 KM
X	15012,14	300080,62
Y	2358,21	2759,1
Z	49781,82	44325,2

Table 1. The magnetic field strength in Nano Tesla

3 Helmholtz Cage

Hermann Ludwig Ferdinand von Helmholtz (August 31, 1821 – September 8, 1894) was a German physician and physicist who made significant contributions to several widely varied areas of modern science. [7]

A **Helmholtz coil** is a device for producing a region of nearly uniform magnetic field, named after the German physicist Hermann von Helmholtz. It consists of two solenoid electromagnets on the same axis. Besides creating magnetic fields,

Helmholtz coils are also used in scientific apparatus to cancel external magnetic fields, such as the Earth's magnetic field.

The main parts of the Helmholtz Coils are the two coils of wire. Since the coils are identical in every way, (wire material, wire gauge, resistance, radius R), feeding the same current to both coils will create two identical magnetic fields. The key characteristic of the Helmholtz Coils is how the two coils are placed. Each coil lies along the same axis, and the separation between the coils is equal to each coil's radius.

When current passes through the coils, the ensuing magnetic fields then interact in important ways in the space between the two coils. Applying the right-hand law, one sees that the fields perpendicular to the common axis run into each other and cancel each other out. The same rule also shows that the fields parallel to the common axis run in the same direction, reinforcing each other. All the cancelling and reinforcing leaves a uniform field in a cylindrical volume between the two coils. Specifically, the volume has a radius (r) equal to 25% of coil radius (R) and a length equal to 50% of the distance between the two coils shown in Figure 2.

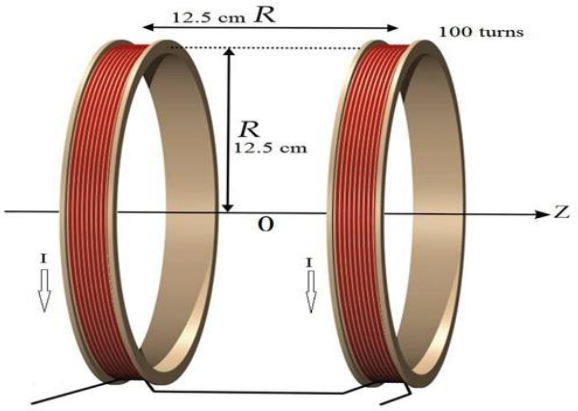


Figure 2. Helmholtz's Coils [8]

Helmholtz coil midpoint of the magnetic induction is possible to find with Equation [1]:

$$B = (\frac{4}{5})^{\frac{3}{2}} * \frac{\mu_0 NI}{R}$$

Where:

B is the magnetic field strength in Tesla;

$\mu_0 = 4\pi * 10^{-7} * T * \frac{m}{A}$ is the magnetic permeability constant of the vacuum;

I - the current through the coils, in amperes;

R - the radius of the coil, in meters;

N - the number of turns in the winding.

4 Pre-requirements

The size of the test equipment depends on the dimensions of the desired satellite. For smaller satellites, it's necessary to build smaller size test equipment to create a homogenous magnetic field. The cube satellite dimension is 10cm3. Considering that the test equipment should be as small as possible and could be portable, the coils should be made with minimal dimensions, with a small power supply. The starting parameter is a medium radius 30 cm ring with the number of windings 34, which is capable, of generating a magnitude of 101906.6 nT with a current flow of 1 ampere. It is planned to use a 12V power supply because there is a large selection of components. If the voltage is too high, the choice of control electronics components will be narrowed. Knowing all the necessary parameters, the over-controlled assertion that a homogeneous magnetic field occurs in one-third of the windings between the windings has become moreover verified.

To precisely test the attitude determination control system of the satellite, the Helmholtz cage design should be able to generate a magnetic field two times stronger than Earth's magnetic field. There are some couple of reasons:

- We need to cancel out the magnetic field, in ideal environment it must be equivalent to 0 nT.
- Cage must generate a magnetic field equivalent to the field of the orbit at 450km attitude.

4.1 Concept selection

Concept selection allowed to compare ideas one to one another at different aspects, were defined three aspects mechanical, electrical and software.

4.1.1 Mechanical decision

The general shape of the cage was selected to be circular. This decision will result in a cage that is easier to manufacture than a square at the size we need. These conditions indicate that the cage is capable to produce required magnetic field to test a satellite. A Helmholtz cage must be completely made from non-ferromagnetic materials because ferrous materials affect the magnetic fields, which would disturb produced magnetic field. Therefore, the bottom plate, frame, coils, and screws should be made from:

- Plastic [9]
- Aluminum [10]
- Wood [11]

It is also important to remove any ferrous materials inside the cage during testing to ensure a good uniform magnetic field.

Helmholtz Cage consists of three pairs of coils, in our case were chosen three optimal sizes of coils: 28cm,30cm and 32cm, which could provide a necessary magnetic field.

- Rings used: Hoop, P/N 11586-08, Diameter 65CM.
- The optimal radius of coils in cm R: 28,30,32.
- Number of turns N: 32,34,36.

4.1.2 Electrical decision

I Referred to COTS power supply because it would be more reliable to use, which has been fully tested and commercially approved. Designing, testing, and verifying the functionality of customized power supply would take extra time that could be used to progress the design of the controller board. For 3 pair of coils could be used multiple power supplies, one power supply to each but it would increase the number of extra wires and cost of power supplies. Accordingly, was decided to use a power supply with current 10A and voltage 12 volts

The coils will connect to coil driver through connector sockets. The Arduino will connect to the PCB through a pin connector (socket). The data from the user will be communicated to the coils through a universal communication bus (USB) and then through the chosen microcontroller. Was originally decided to use the Arduino Uno microcontroller [12], which is widely used, it is a free source and libraries are available. It has required basic functionality and parameters to accomplish what was needed to be done, as well as I did

all coding on it, then debugged the same codes to another Arduino microcontrollers, for coil driver was using Arduino Pro Micro, which is smaller and does the same function as Uno, For magnetic field reading, was needed separate microcontroller because of MATLAB GUI could not transmit and receive data on the same serial PORT at the same time, to perform data reading was used Arduino Nano.

4.1.3 Software decision

To implement the software part, would need an operational system, which could be Windows, Linux or MacOS. I had more experience with Windows and was aware that needed software might not be available for other operating systems, I chose Windows 10 as an operational system to work on.

For programming Arduino, was used C/C++ language, programming environment was Arduino IDE, which is a user-friendly and free source, such environment has a serial monitor which allows read/write data from/in the microcontroller. Install new libraries is also available, the compiler is easy to use, gives messages about errors and wrong syntaxes.

For 3D designing was used AutoCAD and Fusion360, both are powerful CAD tools, AutoCAD gives the possibility to make 2D drawing and then convert it to 3D, drawings can be done directly in the 3D environment as well, it has a high variety of options and features. For 3D rendering is used Fusion 360, which allows making 3D designs, has a nice interface and has plenty of features.

For simulations was used MATLAB, as far as in future would need simulations and implementing the IGRF model, it is the right tool to use. Recently data input/output is implemented through MATLAB. Was created GUI and data can be sent from GUI to the microcontroller using a special syntax to change magnetic field strength.

5 Calculations

5.1 Coil Calculation

To calculate the required parameters was used Equation [2]:

$$B = \left(\frac{4}{5}\right)^{\frac{3}{2}} * \frac{\mu_0 NI}{R} = \left(\frac{4}{5}\right)^{\frac{3}{2}} * \frac{4\pi * 10^{-7} * T * \frac{m}{A} * N * I}{R}$$

Coil	Radius, Cm	Turns, N	Current, I	Nano Tesla
1	28	32	1	102763,1
2	30	34	1	101906,6
3	32	36	1	101157,3

Table 2. Magnetic field strength according to the coil parameters

After defining the radius of the coils and number of windings, was needed to know the circumference as a linear distance around the ring (coil), to know how many meters of copper wire is necessary to prepare coils and to calculate the resistance of each coil.

To calculate the circumference was used Equation [3]:

$$C = \pi * d$$

Where:

C – circumference

π - ratio of a circle's circumference

d - diameter of the circle

Coil	Radius, Cm	Diameter, Cm	Circumference, M	Wire length. M	Turns, N	Wire length. M
1	28	56	1.759292	56,297344	32	56,297344
2	30	60	1.884956	64,088504	34	64,088504
3	32	64	2.01062	72,38232	36	72,38232

Table 3. Circumference

As a reference was taken 30 cm radius coil. All calculations were performed only for single coil. Table 4 below shows:

- Wire diameter, the resistance per meter (Ω per meter), unit Ohm;
- The radius of the coil, unit Centimeters;
- Wire length according to the radius of the coil (Length, M), unit Meter;
- Resistance per length (Ω*Length, R), centimeters
- Voltage drop in case of 1-ampere current flow (Vdrop), unit Volt;

The optimal diameter of the wire would be about 0.8 mm if the maximum voltage flow is 10 volts. The table is for comparison of different standard wire gauge.

To calculate voltage drop, for each coil was used Equation [4]:

$$Vdrop = I * R$$

where:

I - the current through the wires, measured in amperes;

R - the resistance of the wires, measured in ohms;

To calculate resistance per total length of the wire was used Equation [5]:

Ω*Length= Ω per meter * Length

Wire	Diameter, mm	Ω per meter	Radius	Length, M	Ω*Length, R	Vdrop
SWG 21	0,813	0,035	30	64,088	2,24308	2,24
AWG 22	0,6426	0,05241	30	64,088	3,35888	3,36
AWG 23	0,574	0,06569	30	64,088	4,2100	4,21
AWG 24	0,2405	0,08304	30	64,088	5,32191	5,32
AWG 25	0,4547	0,1047	30	64,088	6,71006	6,71

Table 4. Standard wire gauge

As was mentioned for Cage were chosen three optimal diameters of rings and were calculated magnetic field strength. It was necessary to calculate wire length and resistance for each coil to create the required magnetic field.

To reduce heat was chosen wire gauge SWG21 with diameter 0,813 mm and resistance per meter (Ω per meter) is 0,035Ohms. All calculations were performed only for single coil. Table 5 below:

Wire	Diameter, mm	Ω per meter	Radius	Length, M	Ω*Length, R	Vdrop
SWG21	0,813	0,035	28	56,297344	1,97040704	1,97
			30	64,088504	2,24309764	2,24
			32	72,38232	2,5333812	2,53

Table 5. Wire gauge

5.2 Magnetic field uniformity

After all theoretical calculations, that number of windings, amount of current and ring radius can produce magnetic field equal to ~100 000 nT, it is necessary to check magnetic field uniformity, it is affected by misalignments. This undesired effect regularly occurs in any practical tri-axial Helmholtz coils system. To prove mathematical calculations, the magnetic field has been measured under different conditions.

Firstly, was necessary to have a preliminary picture, how calculations above are optimal and if they are suitable to create a magnetic field as well as if magnetic field uniformity looks fine. Uniformity of the coil was checked by MATLAB code. On the horizontal axis is represented uniformity width, the frame is from -50 centimetres to 50 centimetres, the middle point 0 represents a middle point of the cage where the magnetic field is the

highest, on the vertical axis, is represented magnetic field strength. Show in figure 3:

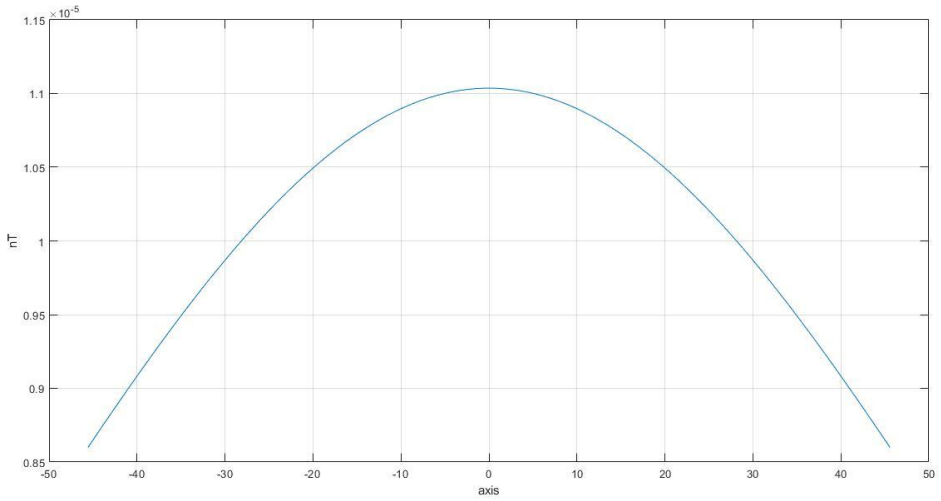


Figure 3. Theoretical magnetic uniformity

6 Magnetic sensor

In order, to measure actual field magnitudes, the magnetic sensor is needed, in my case I had two options:

1 Magnetometer HMC5883L [13]

2 Magnetometer MAG3110 [14]

A three axes Magnetometer HMC5883L was purchased. This device is equipped with a small sensor by which can be measured magnetic field strength.it will locate in the centre of the cage and measure magnetic field from 3 Axes (X, Y, Z). For its characteristics, this device satisfies our requirements to implement testing. The sensor is shown in Figure 4

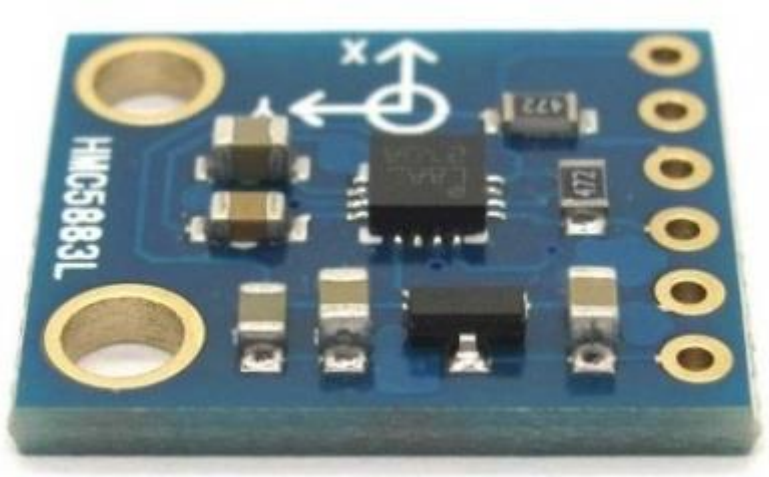


Figure 4. Magnetometer HMC5883L

The main parameters of the magnetometers are shown in table 6:

	HMC5883L	MAG3110
Interface	I2C	I2C
Output Data Rates	160Hz	80Hz
Range +/-	8 Gauss	10 Gauss
Noise	2 Milligauss	2.5 Milligauss

Table 6. Magnetometer main parameters

6.1 Sensor validation

Measurements were done in open field, far from buildings, to verify magnetic sensor, measured data is presented in table 7 and must be compared to table 1, the values are almost similar, which means that the magnetic sensor is accurate with small deviation and deviation can be explained by the different location where measurements are done.

X earth, nT	**Y earth, nT**	**Z earth, nT**
15000.00	2650.18	56020.41
15272.73	2545.45	55918.37
15090.91	2545.45	55918.37
15090.91	2636.36	55918.37

15000.00	2545.45	55816.33

Table 7. Measurements on open field

7 Cage Design

7.1 Cage Design Implementation

In the previous chapter were performed calculations to determine the number of windings per coil and the sizes of coil pairs. To design cage has been used AutoCAD 2016, shown in Figure 5 and for 3D modelling Fusion360. I had two options of CAD environments one was SolidWorks and the second one was AutoCAD. The main reason why the desired CAD environment has been chosen is that I had working experience with AutoCAD from previous university and work, as well as AutoCAD and Fusion A360 for a while is free for students.

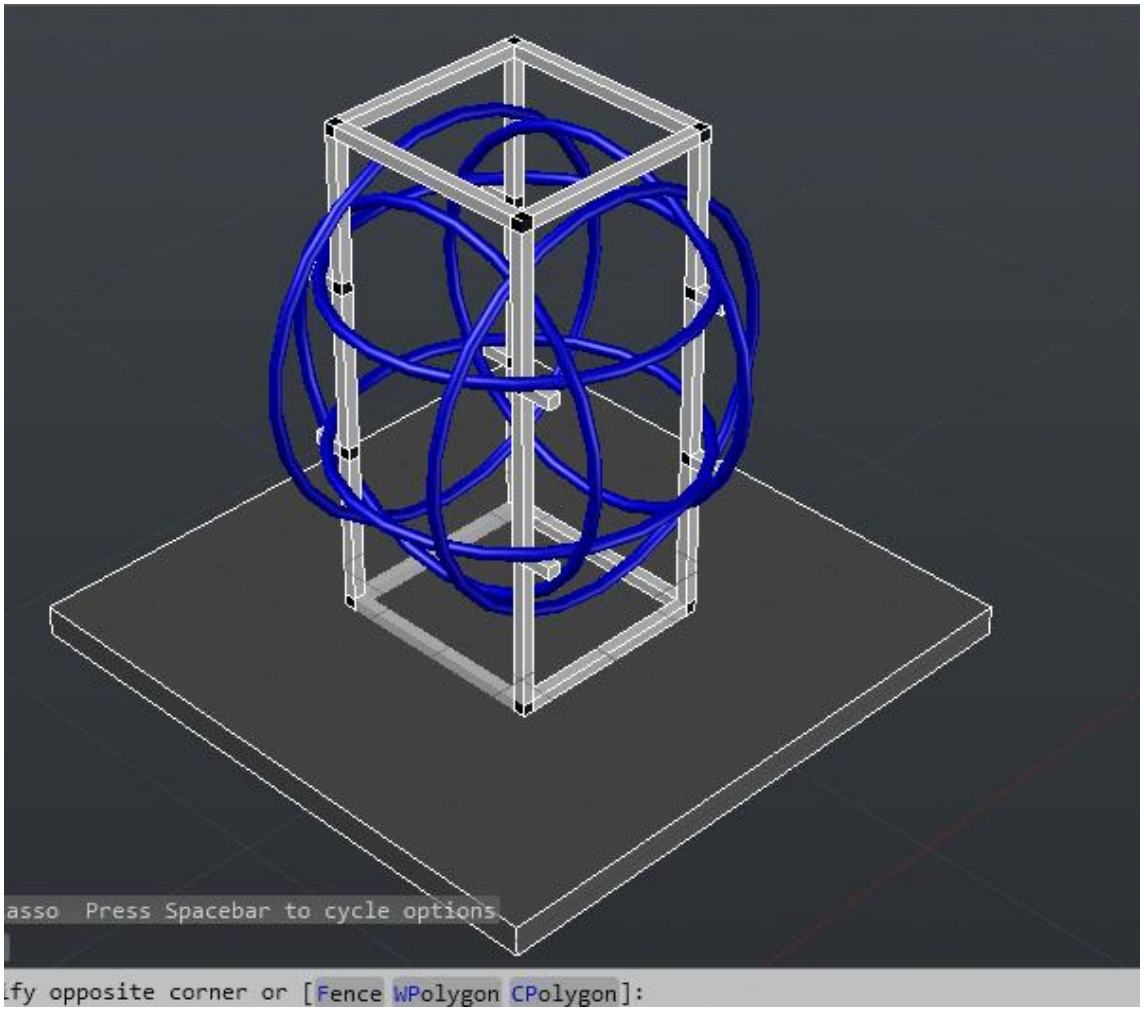


Figure 5. 3D model of the cage in AutoCAD 2016

It was necessary to represent AutoCAD design to a more realistic environment to see the clear picture of the design and appearance of the cage, to implement such task Fusion A360 has been used. The result is shown in Figure 6.

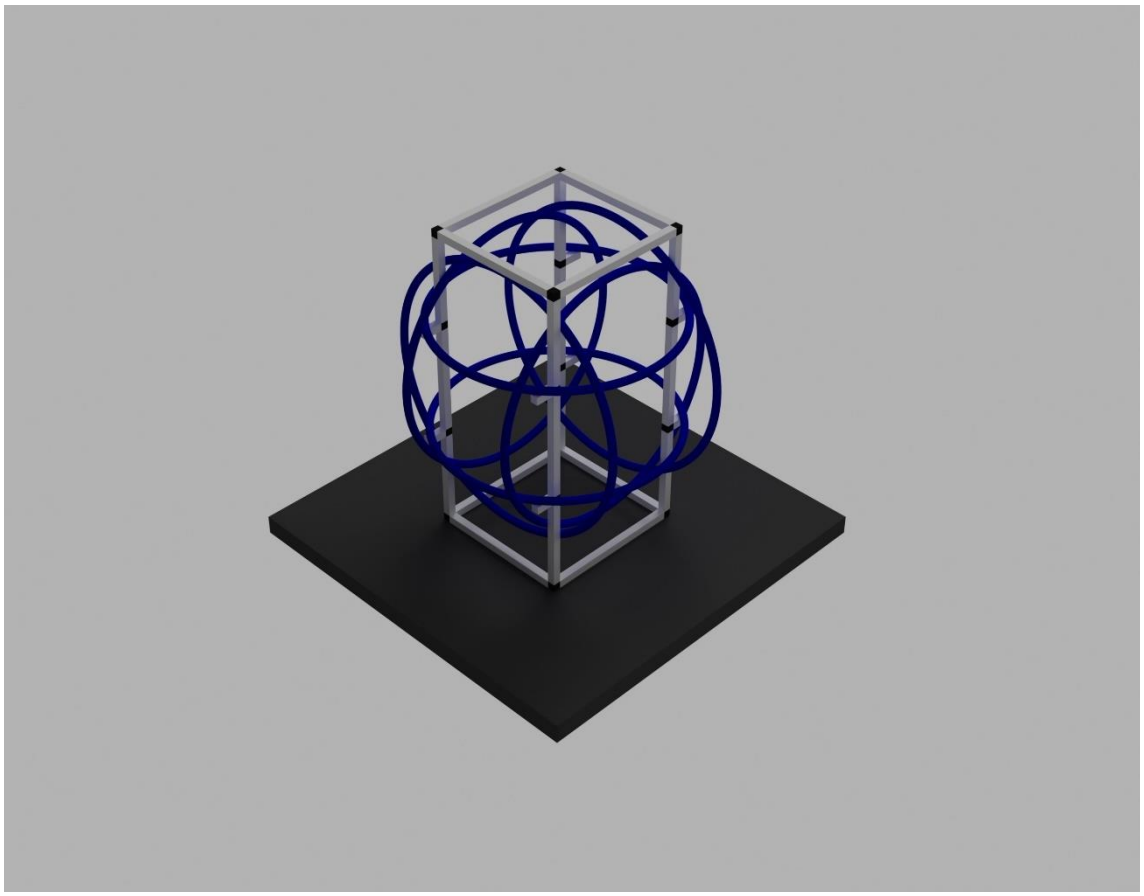


Figure 6. Cage 3D design rendered by Fusion A360

7.2 Coil Assembly

To prepare coils were chosen hoop plastic rings because of plastic properties:

- High paramagnetic characteristics
- Resistance to corrosion and chemicals
- Low electrical conductivity
- Resistance to shock
- Good durability and low cost.

Each coil is cut from the side and put windings inside, as shown in Figure 7. Open edges connected with plastic extender, plastic holders were used to attaching coils to the frame.

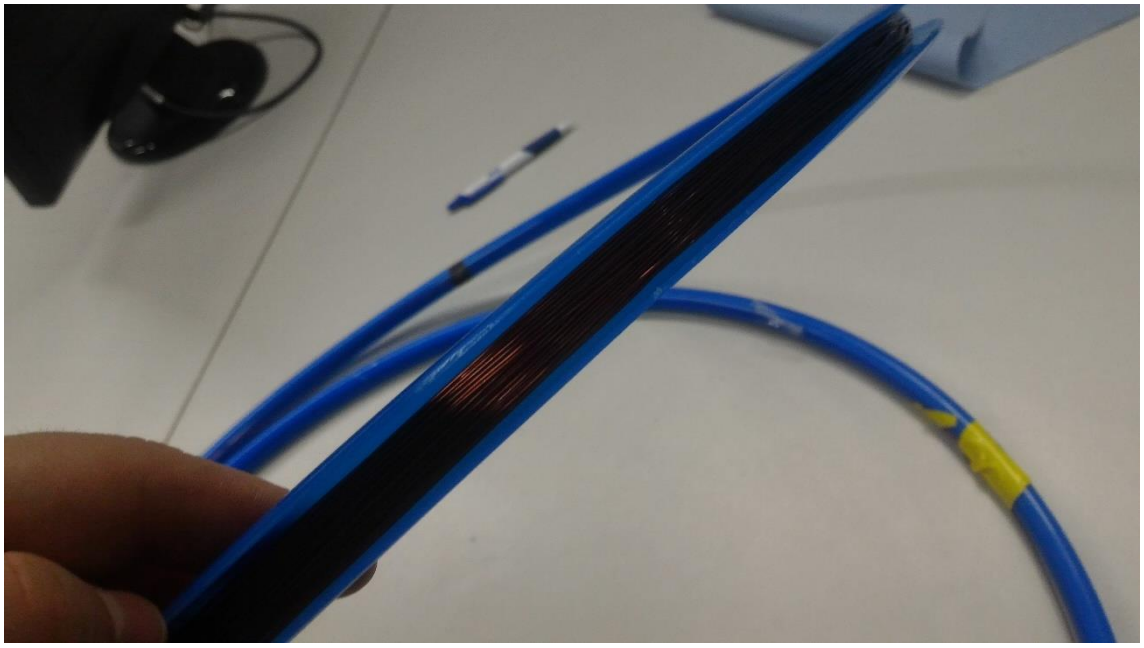


Figure 7. Coil assembly

7.3 Frame Assembly

The frame was made from Aluminum, because of its properties, according to the temperature of the working environment prevents aluminum to remain paramagnetic:

- Reliability
- Paramagnetic
- Durability
- Capable of being shaped or bent
- Ease of construction.
- Availability

All coils should be attached in a specific configuration. As well as, the centers of the coil pairs must be concentric, and all three axes must meet in the center of the cage. To connect aluminum sticks to each other were used T-shaped plastic connectors, which are ideal to make corner connections, the frame is shown in Figure 8.

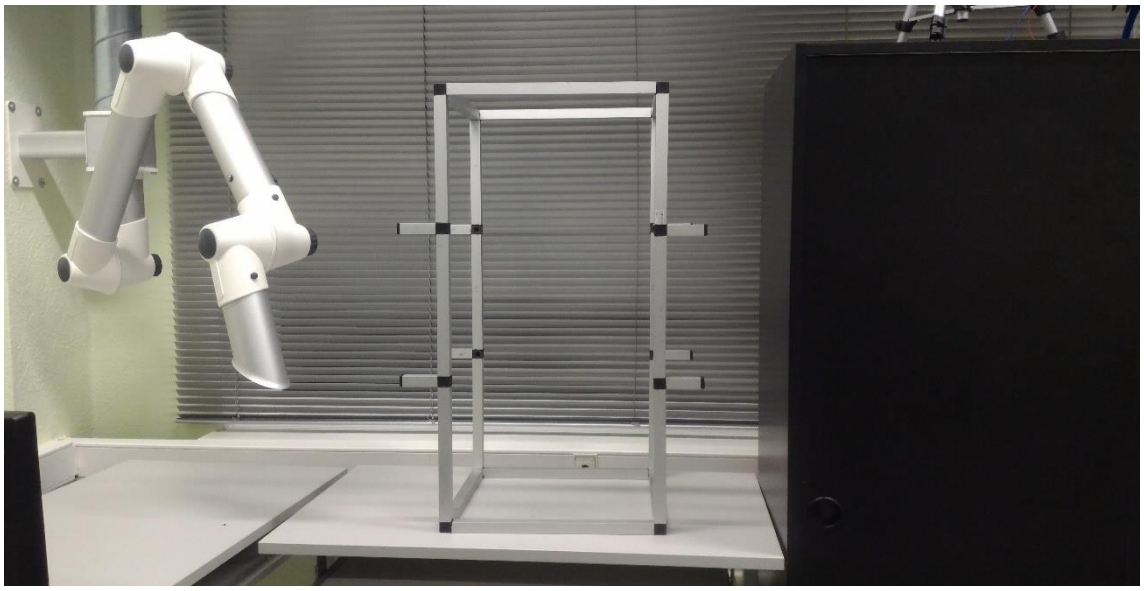


Figure 8. Final frame assembly

7.4 Final Assembly

All coils were attached to the frame as shown in Figure 9. The X and Y axis coils were fixed vertically on the frame and the Z-axis coils were fixed horizontally. The coils were positioned within ¼” of their design position. Extra care was taken to ensure that the distance between the coils matched the design specifications. Helmholtz cage was fully assembled, including coil plastic holders, power supply unit, a wooden bottom plate with rudder feet. the wooden stand was necessary to point the magnetic sensor to the centric point, in order, to measure the magnetic field generated by coils.

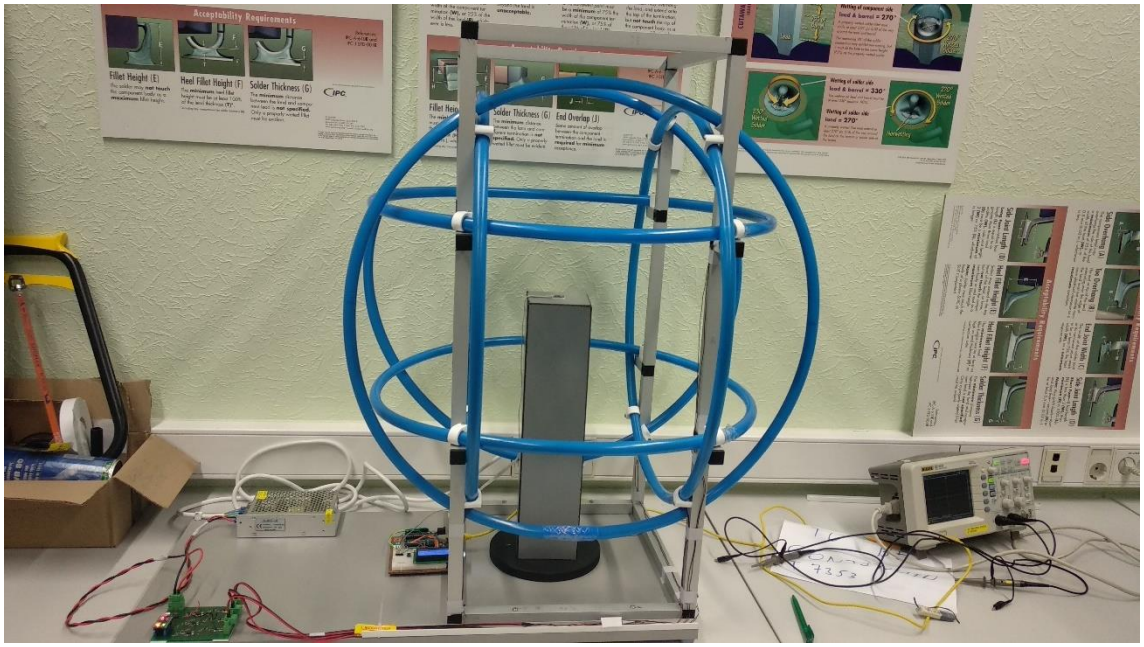


Figure 9. Cage final assembly

7.5 Coil Validation

The Helmholtz coils must now be validated to prove that they can create a magnetic field. The coils were validated with a magnetic sensor. Firstly, the sensor was placed in the center of the cage to read the earth magnet field, in this case, measurements were done in Mektory building and values will differ from the measurements which were done in the open field, due to building surrounding, shown in Table 8. A pair of coils was turned on, the magnetic sensor is placed into the center of the cage, shown in Figure 10. As was mentioned in Chapter 4, the coils should generate magnetic field strength equal to ~ 100 000 Nanotesla, during the cage testing with current ~1 ampere, I was able to gain magnetic field strength equal to nominal values shown in Table 9. Because of the Y and Z axes have a negative sign, after passing the current through the coils they will generate a magnetic field which is equal to ~100 000 Nanotesla, what means that negative signs will change to positive and compensates. The principle of comparing is to compare measured data which was generated by earth (X, Y, Z earth) and the data which was generated by cage (X, Y, Z cage) and subtract them, the result will show difference (X, Y, Z Δ) using Equation [6], because of values are different, need to understand the real magnetic field strength generated by coils, results are shown in Table 10.

Figure 10. Magnetic field validation

X earth, nT	**Y earth, nT**	**Z earth, nT**
11272,73	-30545,46	-54183,67
11727,27	-30545,46	-53877,55
11363,64	-30636,36	-53877,55
11363,64	-30181,82	-53877,55
11636,36	-30272,73	-54081,63
Current flow Ampere		
0,00	0,00	0,00

Table 8. Magnetic field strength generated by the earth in building

X cage, nT	**Y cage, nT**	**Z cage, nT**
109545,46	84000,00	65714,29
109272,73	84000,00	65408,16
109636,37	84454,56	65612,25
109090,92	83909,09	65612,25
109545,46	84181,82	65612,25
Current flow, Ampere		
0.950	1.083	1.093

Table 9. Magnetic field strength generated by cage

Equation [6]:

$$X\ cage - X\ earth = X\ \Delta$$
$$Y\ cage - Y\ earth = Y\ \Delta$$
$$Z\ cage - Z\ earth = Z\ \Delta$$

X Δ, **nT**	Y Δ, **nT**	Z Δ, **nT**
98272,73	114545,46	119897,96
97545,46	114545,46	119285,71
98272,73	115090,92	119489,80
97727,28	114090,91	119489,80
97909,10	114454,55	119693,88
Current flow Ampere		
0.950	1.083	1.0935

Table 10. The difference between generated magnetic field by earth and the cage

7.6 Magnetic field uniformity validation

After cage assembly and implementing all necessary measurements, was necessary to measure magnetic field uniformity, to validate theoretical meanings and to ensure that cage physical assembly meets the design requirements.

Was set the maximum current to the coil and sensor was relocating linearly among the desired axis by step of 1 centimetre.

Uniformity of the coil was verified by MATLAB code according to the collected data from magnetic field reader. On the horizontal axis is represented uniformity width, the frame is from -50 centimetres to 50 centimetres, the middle point 0 represents a middle point of the cage where the magnetic field is the highest, on the vertical axis, is represented magnetic field strength. Shown in figure 11:

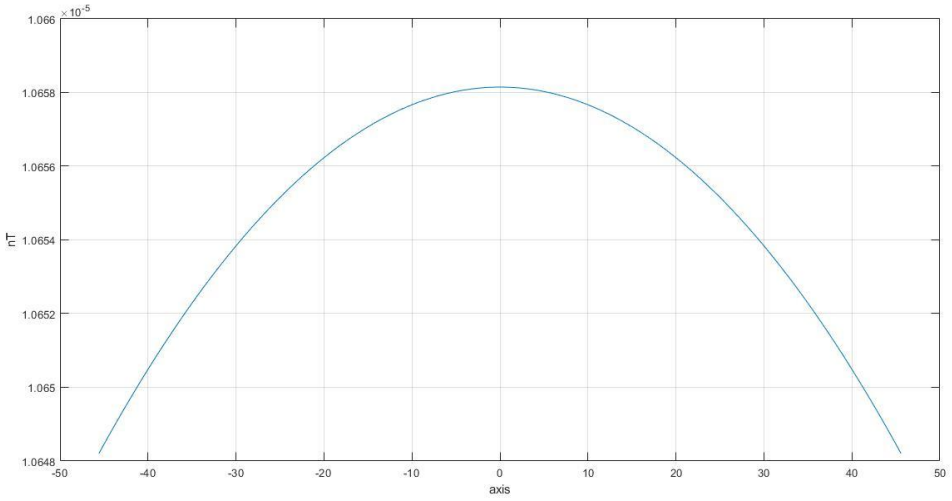


Figure 11. Practical magnetic uniformity

8 Magnetic Field Read and Control System

8.1 Magnetic Field Reader

8.2 For measuring magnetic field is using separate Arduino microcontroller. Magnetic sensor connected through an I2C communication protocol. The sensor is positioned in

the middle of the cage and measuring the magnetic field, the measuring unit is Nanotesla. Received data can be displayed on LCD, which shows real-time measurements, can be sent to excel document using the special tool[15] or can be read by serial monitor, which is integrated into Arduino IDE environment. [16] The reader block diagram is shown in Figure 12. Code is shown in Appendix 2

Table 11 shows the data which is sent to serial monitor from Arduino while the test procedure was running.

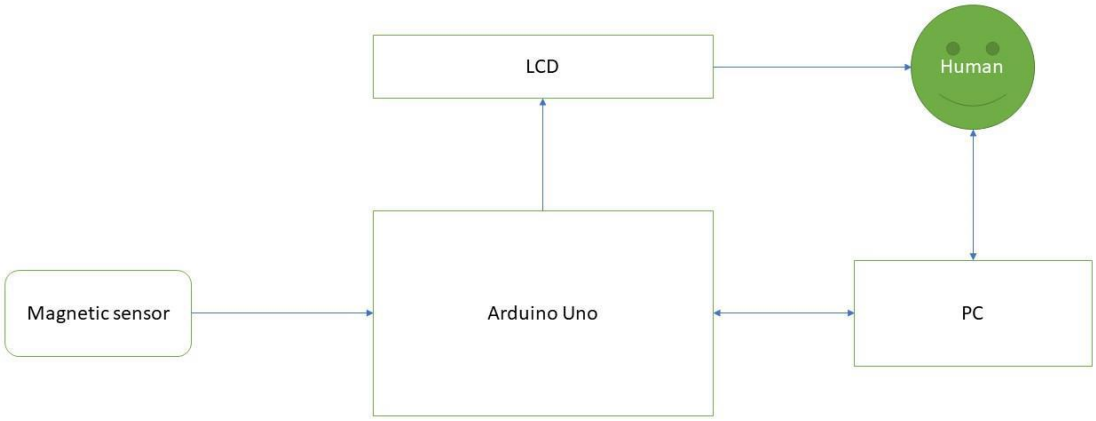


Figure 12. Magnetic field reader block diagram

X, nT	**Y, nT**	**Z, nT**
181,82	181,82	102,04
0,00	-90,91	204,08
272,73	0	-102,04
181,82	-363,64	0
272,73	-363,64	0
272,73	181,82	102,04
0,00	90,91	204,08
454,55	-272,73	102,04
181,82	-90,91	306,12

272,73	-181,82	102,04
181,82	-90,91	306,12
181,82	-181,82	0

Table 11. Data were taken from Serial Monitor

7.2 Magnetic Field Controller

To create magnetic field is used coil driver which is consisting of Arduino micro pro and DAC, the working principle is to give digital value to DAC in numbers and then convert it to Analog signal, which will set the current and manipulate relevant magnetic field inside the cage. From beginning coils' current was driven by the laboratory power supply, then by Arduino micro pro which was installed on PCB of coil driver, In Figure 13 is shown the block diagram of the controller. Code is shown in Appendix 3.

Figure 13. Magnetic field controller block diagram

The main idea was to use a cheap and smart solution to develop our test equipment. before moving the second step, it is necessary to start the first step. The controller which is based on Arduino is reliable, but as far as I needed to change values all the time to see and read random measurement points, I could not change numbers every time in the code and debug it over and over, to avoid such inconvenience I made MATLAB GUI [17], which

is shown in Figure 14 and displayed in Appendix 4. The number is given through MATLAB GUI, that number will be converted to Analog signal and give its relevant current to coil(s) and current will create magnetic field, that magnetic field will affect the magnetic sensor and magnetic sensor will read and send the values to Arduino Uno, which will be displayed on LCD, or Serial monitor. Then according to readings, we might want to change magnetic field strength until we don't get satisfactory value on the reader's screen. The overall structure of the Read and control system structure is shown in Figure 15.

Figure 14. MATLAB GUI

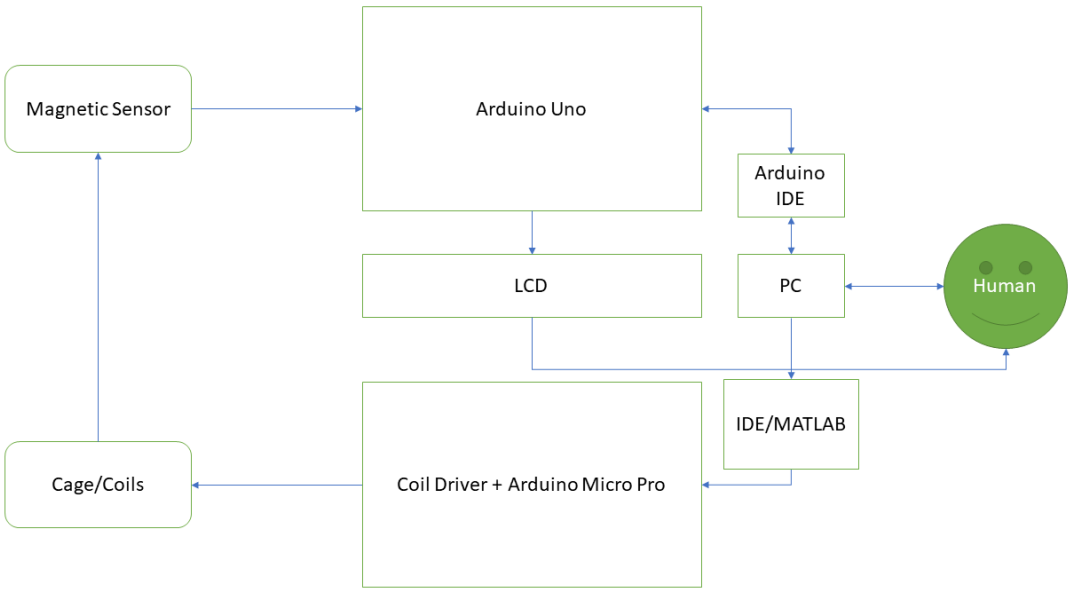


Figure 15. The overall structure of reader and control system

9 Material testing

Each material has own properties and behaves differently in various situations. It is easy to detect ferrous metal or even magnet with the usual magnet, but because of we are using non-ferromagnetic materials and components, they still have some residuals which may occur faulty on heavy loads or if such metals are together they will create a big magnetic bunch. As far as the satellite is under development, I was not able to test the satellite attitude determination control system. Our Nanosatellite is consisting of non-ferromagnetic materials, as are screws, nuts and etc, but it was a good idea to check materials on magnetism, how they behave before and after magnetization. [18] Test procedures have been performed on 16 components which are parts of the satellite. The test procedure is consisting of a few steps:

1. Cancel out the magnetic field into the cage and take samples. Table 12
2. Then put the component inside and take samples. Table 13
3. Set the magnetic field to maximum and take samples. Table 14
4. Nullify magnetic field into the cage and take samples. Table 15
5. Testing results. Table 16

The 1st step is needed to take some data as a reference, in order, to have evidence of initial measurements. 2nd step put component on the top of the magnetometer, if it has own magnetism it will change reading data and could be easily detected, 3rd step will set the maximum current which will guarantee a powerful magnetic field near to 100 000 Nanotesla, it will magnetize the component, on 4th step the magnetic field into the cage will be zeroed, it means that if from beginning the metal was not affecting on readings and everything was looking fine, now it will have leftover, residual magnetic field which will spread for a while. The 5th step is to analyse results, the difference between the 2nd step and the 4th step, were taken first 10 samples.

X, nT	Y, nT	Z, nT
-181,82	181,82	0
-90,91	90,91	90,91
-90,91	0	0
-272,73	0	-102,04
-90,91	90,91	-102,04

Table 12. Nullified Magnetic field into the cage

X, nT	Y, nT	Z, nT
90,91	-181,82	714,29
636,36	-545,45	1122,45
454,55	-636,36	1632,65
90,91	-818,18	2040,82
454,55	-909,09	1122,45

Table 13. Measuring component initial magnetism

X, nT	Y, nT	Z, nT
-88272,73	86272,73	68469,39
-88272,73	87090,92	68877,56
-88636,36	87000	69387,76
-88272,73	86545,46	69489,79
-88272,73	86454,55	69183,67

Table 14. Cage spreading maximum power to magnetize component inside

X, nT	Y, nT	Z, nT
454,55	-1363,64	918,37
272,73	-1272,73	1224,49
454,55	-1363,64	1020,41
454,55	-1090,91	1224,49
545,45	-1181,82	1122,45

Table 15. The magnetic field is nullified again, measuring residual magnetism

2^{nd} step (before)			4^{th} step (after)			difference		
X, nT	Y, nT	Z, nT	X, nT	Y, nT	Z, nT	X, nT	Y, nT	Z, nT
454,55	-636,36	1632,65	454,55	-1363,64	1020,41	0,00	727,28	612,24
90,91	-818,18	2040,82	454,55	-1090,91	1224,49	-363,64	272,73	816,33
454,55	-909,09	1122,45	545,45	-1181,82	1122,45	-90,90	272,73	0

545,45	-1363,64	1020,41	454,55	-1181,82	1224,49	90,90	-181,82	-204,08
545,45	-818,18	1326,53	363,64	-1000	1122,45	181,81	181,82	204,08
545,45	-909,09	1224,49	272,73	-1090,91	1020,41	272,72	181,82	204,08
363,64	-1181,82	1530,61	545,45	-909,09	1326,53	-181,81	-272,73	204,08
545,45	-1000	918,37	363,64	-1090,91	1326,53	181,81	90,91	-408,16
636,36	-1090,91	1224,49	545,45	-1181,82	1224,49	90,91	90,91	0
454,55	-1000	1326,53	363,64	-1181,82	1020,41	90,91	181,82	306,12

Table 16. Testing results

10 Problems and Solutions

The first problem occurred when the cage was completely assembled, and all electronics parts were connected. Coils were generating a magnetic field, but for some reason they were disturbed, the research showed that problem was in H-bridges [19] [20], which were faulty from the factory and they start oscillating on the certain input voltage. H-bridges were removed from the PCB to ensure that the problem was in them and it has been proved. H-bridge is a device which changes polarity when needed, for our requirements cage should be able to produce as positive, as negative signed magnetic field. The temporary solution was implemented and on PCB H-bridges were removed and pins on PCB were soldered directly, I was changing polarity manually by replacing negative and positive wires on coils' connector.

To implement H-bridge on relays I referred to the circuitry shown in Figure 16. When the switches S1-S4 are closed and S2-S3 are open a positive voltage will be applied across the coil. By opening S1-S4 switches and closing S2-S3 switches, this voltage is reversed and changes polarity to negative sign.

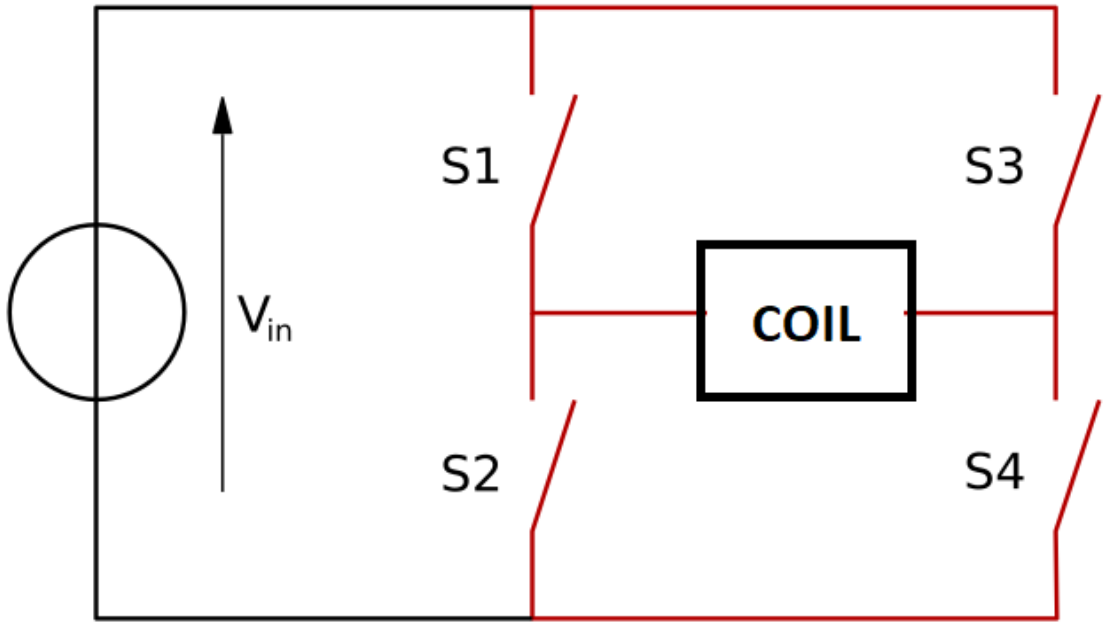


Figure 16. Principal circuit of H-bridge [21]

S1	S2	S3	S4	RESULT
1	0	0	1	positive
0	1	1	0	negative

Table 17. Relay switch table

There were defined two solutions:

- Solid state relays [22]
- Electromechanical relays [23]

For our purposes was chosen electromechanical SPDT relays, SPDTs have three terminals: one common pin and two pins which vie for connection to the common. SPDTs are great for selecting between two power sources, swapping inputs. To implement H-bridge for 3 axes, were needed 6 relays, 2 relays per each axis. was chosen 8 relay modules[24].

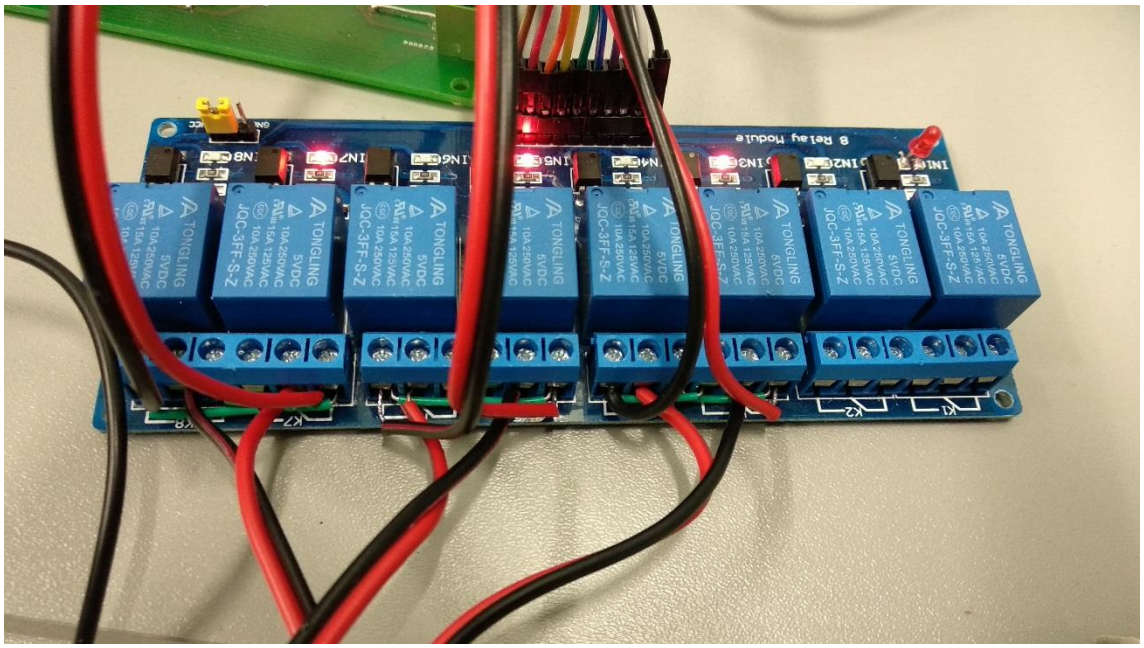


Figure 17. 8 Relay Module

11 Summary

This thesis had the aim to create a simulator based on the Helmholtz coils to test the cube-satellite. The work showed that Helmholtz's cage is able, to test the cube-satellite on Earth. Have learnt orbit properties and satellite behaviour in the orbit, what are the main challenges for it, how we orientate into the space using special on-board sensing devices and of course, IGRF mathematical model which is loaded in the on-board computer of the satellite as a reference. During the studies have made theoretical and practical measurements, learnt how to calculate, design and built such test equipment as Helmholtz's Cage. This test equipment has wide usage and I was able to generate the similar magnetic field which is in the orbit. The result is that cage is capable to test ADCS as well as will be used to develop cube-satellite hardware and design. In summary, the project was successful and my part in space mission is accomplished.

11.1 Future Development

It is necessary to run IGRF mathematical model in MATLAB, this model will simulate earth magnetic vectors by Helmholtz's cage, recently the system is only able to run simulations manually but IGRF is the big list of numbers which needs to be set in MATLAB and will send to Arduino controller.

After IGRF implementation is done, need to create orbit propagator, it determines the position of satellite at any instance of time, with given acceleration and initial velocity. We cannot assume that only gravitational field is affecting the satellite motion, there are other factors such as earth oblateness, gravitational force from moon and sun, atmospheric drag and solar pressure.

References

[1] M. R. Brewer, “CUBESAT ATTITUDE DETERMINATION,” march 2012. [Online]. Available: https://apps.dtic.mil/dtic/tr/fulltext/u2/a557488.pdf.

[2] N. A. a. S. Administration, “https://www.nasa.gov,” Glenn Research Center, 21 11 2004. [Online]. Available: https://www.nasa.gov/centers/glenn/pdf/105819main_FS-2004-11-021.pdf.

[3] C. F. (Chair), “https://www.ngdc.noaa.gov,” 2009. [Online]. Available: https://www.ngdc.noaa.gov/IAGA/vmod/IGRF-11_announce.pdf.

[4] B. E. Survey, “http://www.geomag.bgs.ac.uk/index.html,” [Online]. Available: http://www.geomag.bgs.ac.uk/research/modelling/IGRF.html.

[5] S. S. a. E. C. resources. [Online]. Available: https://www.ssec.wisc.edu/airportexhibit/files/side3.pdf.

[6] N. C. F. E. sources. [Online]. Available: https://www.ngdc.noaa.gov/geomag/calculators/magcalc.shtml#igrfgrid.

[7] K. Yang, “https://www.edn.com,” 20 01 2016. [Online]. Available: https://www.edn.com/design/analog/4441240/High-Frequency-Helmholtz-Coils-Generate-Magnetic-Fields.

[8] H. A.Haghnegahdar, “https://www.ncbi.nlm.nih.gov,” 01 09 2014 . [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4258865/.

[9] Wiley-VCH. [Online]. Available: https://application.wiley-vch.de/books/sample/3527409726_c01.pdf.

[10] A. B. Ron Cobden. [Online]. Available: http://core.materials.ac.uk/repository/eaa/talat/1501.pdf.

[11] U.-F. S. F. P. L. JERROLD E. WINANDY. [Online]. Available: https://www.fpl.fs.fed.us/documnts/pdf1994/winan94a.pdf.

[12] A. compare, “www.arduino.com,” [Online]. Available: https://www.arduino.cc/en/products.compare.

[13] H. i. source, “https://cdn-shop.adafruit.com,” [Online]. Available: https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf.

[14] PDF. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MAG3110.pdf.

[15] CrtSuznik, “https://www.instructables.com,” [Online]. Available: https://www.instructables.com/id/Sending-data-from-Arduino-to-Excel-and-plotting-it/.

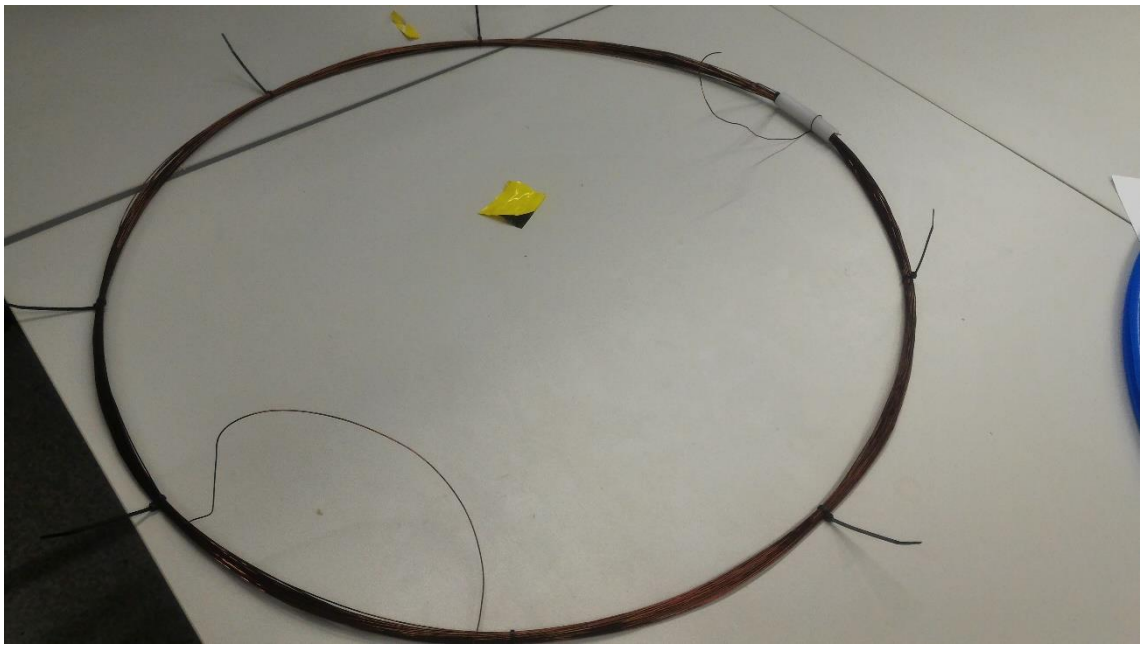
[16] A. source, “https://playground.arduino.cc,” [Online]. Available: https://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf.

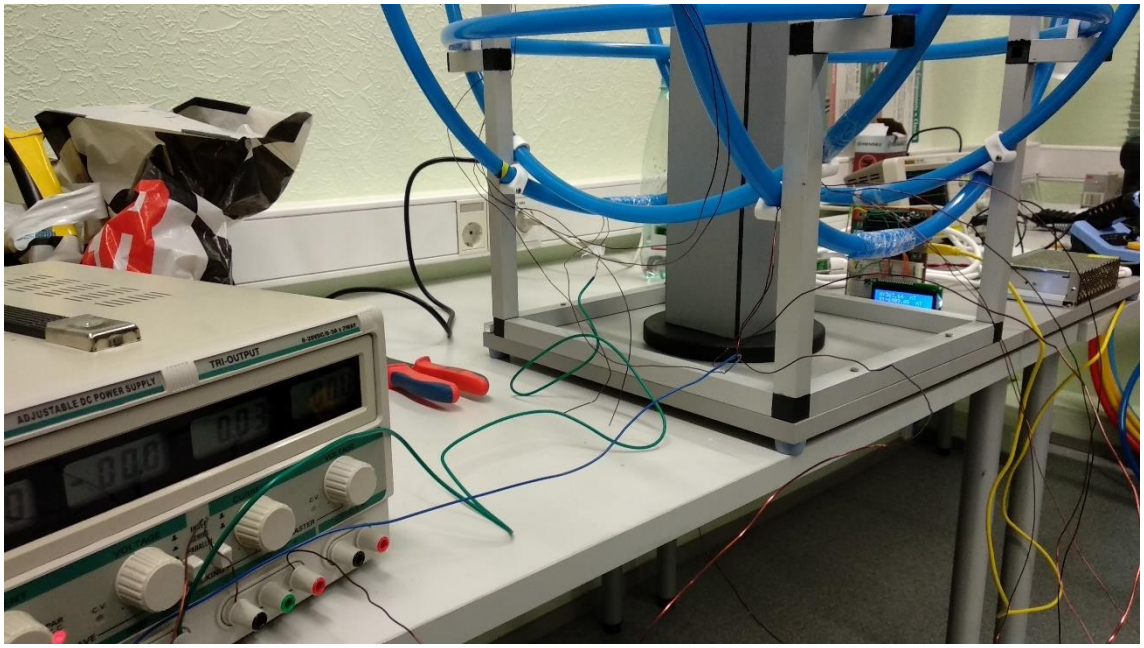
[17] J. Park, “http://www.ee.bgu.ac.il,” [Online]. Available: http://www.ee.bgu.ac.il/~adcomplab/Tutorial%20MATLAB%20GUI.pdf.

[18] U. o. a. source, “https://www.geo.arizona.edu,” [Online]. Available: https://www.geo.arizona.edu/Paleomag/chap03.pdf.

[19] H.-b. picture. [Online]. Available: http://www.wikiwand.com/en/H_bridge.
[20] S. S. Relay. [Online]. Available: https://www.electronics-tutorials.ws/power/solid-state-relay.html.
[21] E. Relay. [Online]. Available: https://www.electronicshub.org/electromechanical-relay-basics/.
[22] 8. c. r. module. [Online]. Available: https://www.elecrow.com/download/8%20channel%20SCH.pdf.
[23] H-bridge\]. [Online]. Available: https://www.rohm.com/documents/11308/12928/100260.H-BRDG_WP_Jan09.pdf.
[24] H.-b. basics. [Online]. Available: http://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics/.

Appendix 1 - Cage assembly




TRI-OUTPUT
ADJUSTABLE DC POWER SUPPLY

Appendix 2 – Data Reader Code

```
/**************************************************************************
*****
  Nika Mukbaniani Edition, Rev 10, some rights reserved
**************************************************************************
*****/

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_HMC5883_U.h>
#include <LiquidCrystal.h>
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
const int buttonPin = 6;
int buttonState = 0;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7); // LCD PINS

/* Assign a unique ID to this sensor at the same time */
Adafruit_HMC5883_Unified mag = Adafruit_HMC5883_Unified(12345);

void displaySensorDetails(void)
{
  sensor_t sensor;
  mag.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println(" nT");
```

```
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println(" nT");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" nT");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}

void setup(void)
{
  // Serial.begin(9600);
  //Serial.println("HMC5883 Magnetometer Test"); Serial.println("");

  pinMode(buttonPin, INPUT);

  /* Initialise the sensor */
  if (!mag.begin())
  {
    /* There was a problem detecting the HMC5883 ... check your connections */
    Serial.println("Ooops, no HMC5883 detected ... Check your wiring!");
    while (1);
  }

  /* Display some basic information on this sensor */
  // displaySensorDetails();
}

void loop(void)
{
  // multiplication - convert micro to Nano
  int a = 1000;

  /* Get a new sensor event */
  Serial.begin(9600);
  sensors_event_t event;
```

```
  mag.getEvent(&event);
  lcd.begin(16, 2);                    //Sets LCD to start
  /* Display the results (magnetic vector values are in micro-Tesla (uT)) */
  Serial.print("X: "); Serial.print(event.magnetic.x * a); Serial.print("  ");
  Serial.print("Y: "); Serial.print(event.magnetic.y * a); Serial.print("  ");
  Serial.print("Z: "); Serial.print(event.magnetic.z * a); Serial.print("  ");
Serial.println("nT");

  //averaging part is here
  float averagex, averagey, averagez;
  float Sum = 0 , Sumx, Sumy, Sumz;
  for (int i = 0; i < 5; i++) {
    Sumx = Sum + event.magnetic.x;
    Sumy = Sum + event.magnetic.y;
    Sumz = Sum + event.magnetic.z;
  }
  averagex = Sumx / 10;
  averagey = Sumy / 10;
  averagez = Sumz / 10;

  //Print on LCD
  lcd.print("X:"); lcd.print(averagex * a ); lcd.print("  ");
  lcd.println("nT      ");
  lcd.setCursor(0, 1);
  lcd.print("Y:"); lcd.print(averagey * a ); lcd.print("  ");
  lcd.println("nT      ");

  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // clear LCD and appear Z
    lcd.clear();
```

```
   lcd.print("Z:"); lcd.print(averagez * a ); lcd.print("  ");
   lcd.println("nT      ");
  }
  delay(1000);
}
```

Appendix 3 – Field Controller Code

```
//Nika Mukbaniani edition, Rev 42, Some right reserved.
//change list by revisions
/*-added 3 channels X,Y,Z revision 10
  -added changing from positive to negative and vice versa revision 15
  -LTC enable implemented revision 20
  -all DRV defined revision 30
  -DAC set to 1V revision 34
  -infinite loop implemented,then disabled revision 37
  -implemented string separation and input/output conversion 42
*/
#include <SPI.h>
#include <DAC7565.h>


// DAC connection, If you do not have all these connected
// (or hardwired), just set unconnected values to -1
// On dac shield board https://github.com/hallard/DAC-Shield
// we have Enable=D7, Sync=D4, LDAC=D8 but we use soft SPI
// (D11 MOSI used by PWM so Data=D12, Clock=D13
// Instantiate DAC with correct wiring
DAC dac(12, 2, 13, 4, 3);

int16_t pos, neg, rx_byte ;  //stores 0 to MAX_SCALE
```

```
String serialstring;

//DRV set which drivers to use numbers are PINs of arduino
int DRV21 = 7;
int DRV22 = 6;

int DRV31 = 8;
int DRV32 = 9;

int DRV11 = 10;
int DRV12 = 16;

int xvalue = 0;
int yvalue = 0;
int zvalue = 0;
//LTC
int LTC_RUN = 5;//RUN of LTC chip

//DAC
int stepSize = 127;
void setup()
{

  //drv output pins of arduino
  pinMode(7, OUTPUT);
  pinMode(6, OUTPUT);
  //drv
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  //drv
  pinMode(10, OUTPUT);
  pinMode(16, OUTPUT);

  //LTC
```

```
pinMode(LTC_RUN, OUTPUT);

//DAC


// Initialize the DAC
dac.init();

// Set DAC reference to be powered up, VREF = 2.5V
// this mean that vout will be able to go from 0V to 2.5V (full scale)
dac.setReference(DAC_REFERENCE_ALWAYS_POWERED_UP);

// Set all outputs to 1V
dac.writeChannel(DAC_CHANNEL_A, 0);
dac.writeChannel(DAC_CHANNEL_B, 0);
dac.writeChannel(DAC_CHANNEL_C, 0);

Serial.begin(9600);


// We start at mid scale (VREF/2)
pos = 0;


//DRV
digitalWrite(DRV11, HIGH);
digitalWrite(DRV12, LOW);
digitalWrite(DRV21, HIGH);
digitalWrite(DRV22, LOW);
digitalWrite(DRV31, HIGH);
digitalWrite(DRV32, LOW);
//need to add another line
//LTC
digitalWrite(LTC_RUN, HIGH);
```

```
  Serial.println("initial delay 5000");
  delay(1000);
  Serial.println("start");
  pos = 510;

}

void loop()
{

  //DAC_MAX_SCALE on 4096
  //   Serial.println(pos);
  /* delay(50);
    for(int i = 2000; i<=3500; i+=100){
    dac.writeChannel(DAC_CHANNEL_A, i);
    }
    delay(50);
    for(int i = 3500; i>=2000; i-=100){
    dac.writeChannel(DAC_CHANNEL_A, i);
    }
  */
  // Activate DAC ,abs-absolute value
  delay(50);

  dac.writeChannel(DAC_CHANNEL_A, abs(xvalue));
  dac.writeChannel(DAC_CHANNEL_B, abs(yvalue));
  dac.writeChannel(DAC_CHANNEL_C, abs(zvalue));

  //infinite loop,increase and dicrease,disabled because of  external inputs

  // pos = pos + stepSize;
  //if (pos < 500 || pos > 4096) {
  //    stepSize = -stepSize;
  //    pos = pos + stepSize;
```

```
if (xvalue > 0 ) {
  digitalWrite(DRV11, HIGH);
  digitalWrite(DRV12, LOW);
} else
{
  digitalWrite(DRV11, LOW);
  digitalWrite(DRV12, HIGH);
}
if (yvalue > 0 ) {
  digitalWrite(DRV21, HIGH);
  digitalWrite(DRV22, LOW);
} else
{
  digitalWrite(DRV21, LOW);
  digitalWrite(DRV22, HIGH);
}
if (zvalue > 0 ) {
  digitalWrite(DRV31, HIGH);
  digitalWrite(DRV32, LOW);
} else
{
  digitalWrite(DRV31, LOW);
  digitalWrite(DRV32, HIGH);
}

if (Serial.available()) {
  // get the byte from the software serial port and implemented spearation as is ';'
  serialstring = Serial.readString();
  String x = getValue(serialstring, ';', 0);
  String y = getValue(serialstring, ';', 1);
  String z = getValue(serialstring, ';', 2);
  xvalue = x.toInt();
  yvalue = y.toInt();
```

```
    zvalue = z.toInt();
    // int yvalue = stringToNumber(y);
    // int zvalue = stringToNumber(z);
    Serial.println(xvalue);
    Serial.println(yvalue);
    Serial.println(zvalue);
    Serial.println();

  }
}

String getValue(String data, char separator, int index)
{
  int found = 0;
  int strIndex[] = { 0, -1 };
  int maxIndex = data.length() - 1;

  for (int i = 0; i <= maxIndex && found <= index; i++) {
    if (data.charAt(i) == separator || i == maxIndex) {
      found++;
      strIndex[0] = strIndex[1] + 1;
      strIndex[1] = (i == maxIndex) ? i + 1 : i;
    }
  }
  return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```

Appendix 4 – MATLAB GUI

```
%NIKA MUKBANIANI, REV 7, script 3, All rights reserved.
%change list is in the same folder, in txt format
```

```
%I have generated the code based on MATLAB examples and MATLAB generates
own syntax, including comments, which will be similar for everybody.
%was defined all channels X Y Z together under one button, sending data with one
button.

%was implemented error dialog for character a, b, c and etc
%was implemented error dialog if input was bigger than -4000 or +4000
%=============================================================
==============================

%=============================================================
==============================
%added buttons for all channels X Y Z to set MIN MAX values. separately for each
%added button to DROP VALUES, one button works for all channels%\
function varargout = script3(varargin)
% SCRIPT3 MATLAB code for script3.fig
%      SCRIPT3, by itself, creates a new SCRIPT3 or raises the existing
%      singleton*.
%
%      H = SCRIPT3 returns the handle to a new SCRIPT3 or the handle to
%      the existing singleton*.
%
%      SCRIPT3('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SCRIPT3.M with the given input arguments.
%
%      SCRIPT3('Property','Value',...) creates a new SCRIPT3 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before script3_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to script3_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help script3

% Last Modified by GUIDE v2.5 20-Apr-2018 21:17:19

% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @script3_OpeningFcn, ...
                   'gui_OutputFcn',  @script3_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
```

```
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before script3 is made visible.
function script3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to script3 (see VARARGIN)

% Choose default command line output for script3
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes script3 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = script3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double

X = str2double(get(hObject, 'String'));
if isnan(X)
```

```
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif X > 4001
    set(hObject, 'String', 0);
    errordlg('MAX value should be + 4000','Error');
    elseif X < -4000
    set(hObject, 'String', 0);
    errordlg('MAX value should be -4000','Error');

else
        updateXaxis (handles)
end

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function updateXaxis (handles)
 X = get(handles.edit1,'string');
X = str2double(X);




function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Y = str2double(get(hObject, 'String'));
if isnan(Y)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif Y >= 4001
    set(hObject, 'String', 0);
    errordlg('MAX value should be + 4000','Error');
    elseif Y <= -4000
    set(hObject, 'String', 0);
    errordlg('MAX value should be -4000','Error');
```

```
else
        updateYaxis (handles)
end
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function updateYaxis (handles)
 Y = get(handles.edit2,'string');
Y = str2double(Y);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Z = str2double(get(hObject, 'String'));
if isnan(Z)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif Z > 4001
    set(hObject, 'String', 0);
    errordlg('MAX value should be + 4000','Error');
    elseif Z < -4000
    set(hObject, 'String', 0);
    errordlg('MAX value should be -4000','Error');

else
        updateZaxis (handles)
end
% Hints: get(hObject,'String') returns contents of edit3 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function updateZaxis (handles)
  Z = get(handles.edit3,'string');
%Z = str2double(Z);


% --- Executes on button press in BT.
function BT_Callback(hObject, eventdata, handles)
% hObject    handle to BT (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
X = get(handles.edit1,'string');
Y = get(handles.edit2,'string');
Z = get(handles.edit3,'string');
X = str2double(X);

s = serial('COM6');

set(s,'BaudRate',9600);
fopen(s);
fprintf(s,' %d;',X);
fprintf(s,'%s',Y);
fprintf(s,';%s',Z);
disp(s)
fclose(s);
updateXaxis (handles)
updateYaxis (handles)
updateZaxis (handles)
 %%%%%%%%%%%%%%%%%%%%%%%%


% --- Executes on button press in BT2.
```

```
function BT2_Callback(hObject, eventdata, handles)
% sets to zeroes
% hObject    handle to BT2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
X = get(handles.edit1,'string');
Y = get(handles.edit2,'string');
Z = get(handles.edit3,'string');
X = str2double(X);

s = serial('COM6');
set(s,'BaudRate',9600);
fopen(s);
fprintf(s,'0;0;0');
disp(s)
fclose(s);
updateXaxis (handles)
updateYaxis (handles)
updateZaxis (handles)


% --- Executes on button press in BT3.
function BT3_Callback(hObject, eventdata, handles)
%sets to MIN
% hObject    handle to BT3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

s = serial('COM6');
set(s,'BaudRate',9600);
fopen(s);
fprintf(s,'-4000;-4000;-4000');
disp(s)
fclose(s);


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% sets to MAX
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

s = serial('COM6');
set(s,'BaudRate',9600);
fopen(s);
fprintf(s,'4000;4000;4000');
disp(s)
fclose(s);
```