

Assignment 2 SGEE6003 Major Project

by Michael Lewis

FILE	5293_MICHAEL_LEWIS_ASSIGNMENT_2_SGEE6003_MAJOR_PROJECT_1 65697_1005943820.PDF (1.43M)		
TIME SUBMITTED	29-APR-2019 04:13PM (UTC+0100)	WORD COUNT	7847
SUBMISSION ID	105716753	CHARACTER COUNT	44546



BSc (Hons) Electrical and Electronic Engineering
SGEE6003 Major Project
Assignment 2 - Project Design, Implementation and Evaluation
Author: Mike Lewis
Submission Date: April 29th 2019
Word Count: 5991
[Word count does not include: Front page, Contents, table of figures, citation list & appendix]

mike lewis

Contents

Table of figures	2
1. Executive Summary	3
2. Introduction	4
3. Design	5
Microcontroller	5
Sample and Hold circuit	6
Motor Driver	7
Control scheme for axis.....	8
Screen	9
4. Implementation	10
Stepper motor driver	10
Sample-and-hold circuit.....	11
Microcontroller C code explanation	12
C code print-out.....	13
5. Testing.....	16
Completed testing table.....	16
Test results explanation	18
6. Evaluation	21
7. Conclusion and recommendations	22
Recommendations	22
8. Citation List.....	23
Works Cited	23
9. Appendix.....	24
Photos.....	24
Testing evidence	29
C code functions	32
Timing diagram	37

Table of figures

Figure 1 - Mechanical gantry design	8
Figure 2 - Menu screen on PC terminal.....	9
Figure 3 - Driver circuit, Microcontroller & stepper motor.....	10
Figure 4 - Sample and hold Veroboard circuit	11
Figure 5- 'TrigOut' breakout box, 8ms 'TrigOut' pulse.....	24
Figure 6- 2 Tesla measured on gaussmeter (reading 2.039T).....	24
Figure 7 - 2T scope output from search coil	25
Figure 8 - Search coil measuring magnetic field with scope output	25
Figure 9- Search coil: Closeup of X, Y and Z windings.....	26
Figure 10- sample and hold circuit diagram.....	26
Figure 11 - Nema 17 stepper motor specifications, motor torque speed curve	26
Figure 12- Motor bridge operation	27
Figure 13 – full- step sequence	27
Figure 14- Stepper Motor Driver circuit	28
Figure 15 – Test 1&5 - Testing sample-and-hold - Channel 1	29
Figure 16 - Test 1&5 - Testing sample-and-hold - Channel 2	29
Figure 17 - Test 1&5 - Testing sample-and-hold - Channel 3	30
Figure 18 - Test 7 - Magnetic field outside machine enclosure	30
Figure 19 - Test 9 - Acoustic measurement test.....	31
Figure 20 - Timing diagra, from assignment 1	37

1. Executive Summary

The project is based around measuring the peak amplitude of a magnetic field that is generated from a medical treatment device. The device in question is a 'Transcranial Magnetic Stimulator' or TMS machine. The TMS uses 300uS to 600uS mono-phasic and bi-phasic magnetic field to stimulate a human's brain. The magnetic pulse is generated inside the TMS machine and a hand-held treatment coil is used to direct the magnetic field to the necessary location on the brain.

This project is based around plotting the TMS output magnetic field in 3D space. This requires a search coil scanning over the top of the treatment coil whilst its firing, taking readings at 1cm intervals. The search coil will scan in a raster fashion covering the X, Y and Z axis as it scans. This data can be fed into computer software to give a 3D map of a treatments coil output, in terms of maximum field amplitude in the treatment area of the TMS coil.

The planning stage of the project has been completed; with the design, implementation and testing yet to be completed. This report contains the full design, implementation, testing and evaluation stages of the project. Once these stages are complete, the 'finished' project will compare with what was predicted and expected during the planning phase of the project.

2. Introduction

A transcranial magnetic stimulator is a medical device that administers high-frequency high-amplitude magnetic field pulses to a human brain with the purpose of curing some mental illnesses. This project was based around measuring the magnetic fields peak amplitude when the stimulator is being fired.

The introduction for assignment 1 will be revisited to state the original aims and objectives to see how it compares with the finished project. The earlier introduction stated that:

- Follow a process of planning the project efficiently and effectively
- Efficient and effective planning requires an understanding of design and management methods
- Show a critical understanding of the project's specifications.
- Demonstrate an in-depth understanding of how theoretical knowledge can be applied to a real-world project where necessary.
- Extensive planning to be carried out to try and ensure that the final project met the initial specification
- Design a product that the customer wanted and ensure the product is delivered on time with all the required features.

(Lewis, 2019)

The original aims and objectives set out in assignment 1 have been modified slightly due to budget and time constraints within the company. The modified aims and objectives are:

- Design and present the project in such a way to allow upper management to release more time and budget to allow the project to be fully implemented.
- Prove that the individual components that will be used in the final implementation are working in the way they were designed and intended, for a cost-effective price.

Only the original aims and objectives are going to be evaluated to see if the final product meets the requirements and plans set out in assignment 1. The modified aims and objectives have been added out of necessity due to the course of the project changing after the planning stage had been completed.

3. Design

Microcontroller

The microcontroller will operate by waiting for a signal from the magnetic stimulator that defines when the machine has fired, this is known as 'TrigOut' signal. The 'TrigOut' signal is accessed through a breakout box on the back of the magnetic stimulator, see Figure 5. This is a 5V pulse that lasts 8ms, which will be fed into an interrupt pin on the microcontroller. The positive-to-negative transition on the interrupt pin will signal when a sample has been taken and needs to be processed by the controller.

The microcontrollers program will operate as follows:

1. Dump capacitors on start-up
2. Wait for the transition on the 'TrigOut' interrupt pin to transition from high to low.
3. each analogue input gets processed one-by-one by the ADC and a digital equivalent is stored on EEPROM
4. Once all three samples have been processed the capacitors are dumped
5. The digital equivalent can be converted to a string to be sent along USART and displayed on a PC terminal, incrementing the storage address on the EEPROM
6. The motor driver is driven with X pulses which relates to a linear positional move in space
7. Return to step 2

Sample and Hold circuit

The overall concept for the sample-and-hold circuit is to charge a capacitor to a voltage corresponding to the peak of a magnetic field generated from a TMS machine. The capacitor voltage can then be fed into an ADC to obtain a digital equivalent of the analogue voltage.

Please note to avoid confusion about the shape of the waveform: The system that produces a 2T magnetic field is a Monophasic system not a biphasic system. The monophasic system does not produce a sinusoidal magnetic field. It instead produces a pulse in one direction, hence the name Monophasic.

The output of the sample and hold circuit must be kept under 5v as that is the maximum voltage input for the ADC on the ATmega328p. The input to the sample and hold circuit must be attenuated to ensure that when the circuit measures 2 teslas the output does not exceed 5v. To find out what attenuation ratio is needed, the magnetic stimulator was fired at a power level sufficient to generate a 2-tesla magnetic field. The magnetic field was first measured with a Gaussmeter giving a peak magnetic field reading of 2.036T, see Figure 6. The search coil built for this project will sit in the 2-tesla magnetic field and the voltage generated will be measured on the scope, as shown in Figure 7. When the search coil measured the 2-Tesla magnetic field the search coil generated a voltage of 10.41V.

Based on the voltage that the search coil generates when measuring a 2-tesla magnetic field, a 1-to-4 attenuation ratio on the input gives an input voltage of $(10.41/4 =) 2.6V$ to the input of the sample and hold circuit. This allows the device to measure above and below the specified limit in the FDS. A simple voltage divider will provide the attenuation and limit the current entering the sample and hold circuit.

The search coil consists of three loops of wire wound around a sphere in the X, Y and Z rotation, see Figure 9. The search coil is kept inside a plastic enclosure to avoid interference with the magnetic field. The circuit diagram for the sample and hold circuit is shown in Figure 10. As there are three coils which make up the search coil, the exact sample and hold circuit will be duplicated, with each circuit being connected to a single coil on the search coil.

The 10nF capacitor stores the peak voltage generated by the search coil. The first op amp used in the circuit compensates for the volt drop of the diode. The second op amp allows the 'Microcontroller sample' pin to be loaded without the capacitor discharging. The transistor allows the microcontroller to dump the capacitors energy to ground. The diode from earth to the input of the first op amp ensures the op amp input doesn't go negative.

Motor Driver

The chosen motor type for this project is bipolar stepper motors. The specific stepper motor used is a Nema 17, the specifications for this motor are shown in Figure 11. The important specs for this part of the project is as follows:

- Phases = 2
- Steps/revolution = 200
- Torque speed curve (see Figure 11)

The steps per revolution details how many individual steps it takes to rotate the motor shaft 360 Degrees. As the full step sequence takes 4 steps, the full step sequence has to run $200/4 = 50$ times to make one full shaft rotation. The phases tell us how many poles the motor has, in this case two. The torque speed curve shows the relationship between the speed at which the motor is stepped and the torque produced. The torque required by the motors depends entirely on the design of the mechanical gantry. As the gantry systems' moving parts will run on bearing, the torque needed to move each axis will be relatively small. As the mechanical side of the project is not being completed due to lack of funding, the exact torque values are not currently known. However, the speed at which the gantry moves can be adjusted to give the motor more torque if needed. The motor bridge is universal and other 2-pole stepper motors can be purchased if the torque output of the NEMA 17 isn't sufficient.

The motor driver circuit consists of 2 motor H-bridges, which allows the direction of current in each winding to be reversed or to stop the current slowing at all. How the bridge changes the direction of current is shown in Figure 14. Each time the current direction of a winding changes, the motor moves one step. The correct sequence for stepping the motor one complete electrical cycle is shown in Figure 13, which is known as full-step sequence.

No positional feedback is going to be used as the stepper motor is accurate if kept within its torque limits. The full step sequence shows how the stepper motors position can be predicted by counting the number of steps delivered.

Control scheme for axis

Unfortunately, the budget for the mechanical gantry for the project has been denied. However, it must still be proven that the motor driver and stepper motor would be able to accurately position the mechanical gantry in space. This is going to be done by driving the stepper motor a particular number of steps and calculating how far the gantry would have moved. The basic operation of the X, Y axis will be pulley and belt driven. The distance the gantry will actually move depends on the circumference of the pulley is being used at the shaft of the motor. A diagram of how the drive motor, pulley and drive belt is coupled to the carriage is shown in Figure 1.

To move the carriage 10mm, the drive belt must advance 10mm, as the drive belt is coupled to the top of the carriage. If the circumference of the pulley is 100mm, the pulley must be rotated $1/10^{\text{th}}$ of a full rotation to advance the drive belt 10mm. 50mm of belt advancement would require half a rotation etc. As the motor requires 200 pulses for a full rotation of the pulley, and a full rotation = 100mm, each pulse of the motor will move the gantry $200\text{pulses}/100\text{mm} = 2/1 = \mathbf{2\text{pulses/mm}}$. This is only true with a pulley with a circumference of 100mm.

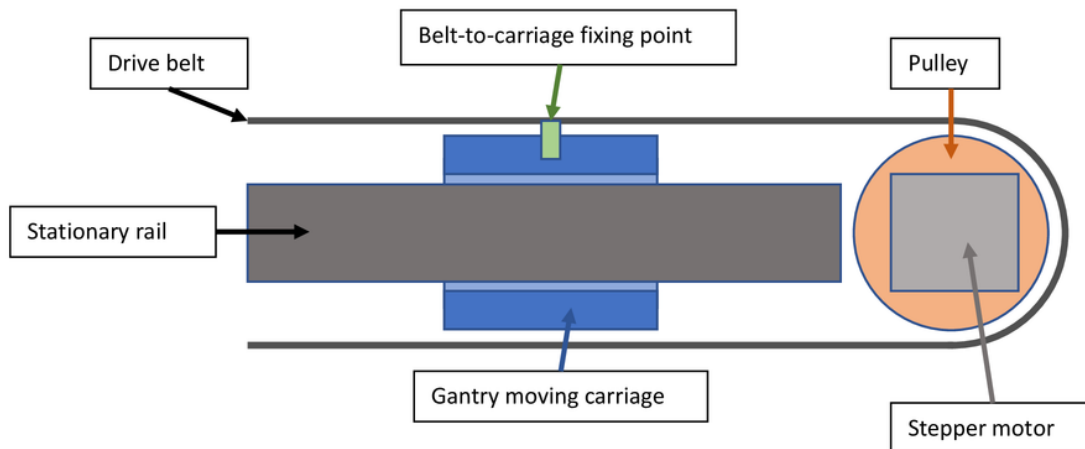


Figure 1 - Mechanical gantry design

Screen

The screen is one part that has not been implemented in the C code. As the design for the menu and screen is completed, exactly how the menu will look on a PC terminal has been recreated in Figure 2. The menu options can be selected by the user by inputting a character that corresponds to one of the on-screen options [A to F]. Once a menu option is selected the main menu is removed and the correct sub-menu is printed on screen with its own selectable options. One crucial aspect of the main menu is to prompt the user to enter a password when the 'service menu' [F] is selected. This is because important machine settings can be changed inside the service menu and a password would stop unauthorised users from changing these critical parameters.

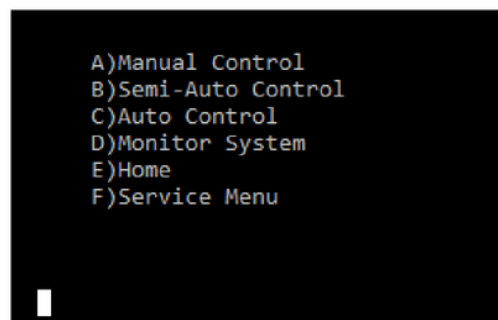


Figure 2 - Menu screen on PC terminal

4. Implementation

Stepper motor driver

The stepper motor driver circuit and code has been fully implemented. The code is setup to run the stepper in half or whole step modes, which is explained in the appropriate section. The code can be used to count the individual pulses that are delivered to the stepper motor to allow the motor to turn to a specific rotation.

The stepper motor driver has only been built onto Veroboard to demonstrate that this method of control is accurate enough for use with a mechanical gantry. Eventually, the driver circuit will be designed onto an actual PCB along with the other circuitry needed for the project. The motor driver, microcontroller and stepper motor are setup and shown in Figure 3. The PCB and Microcontroller will be inside an enclosure when the project is fully implemented but they are exposed in Figure 3 to show the connections and different circuit boards more clearly.

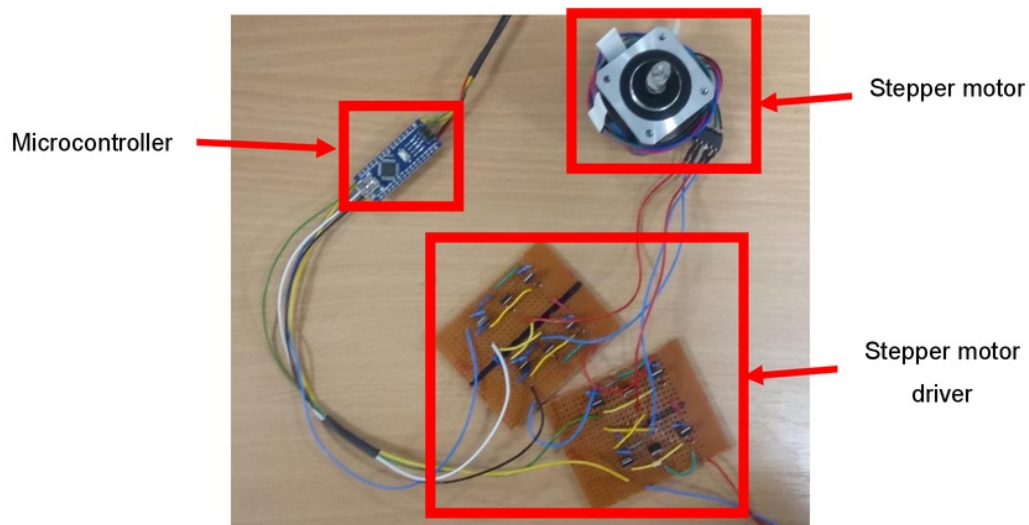


Figure 3 - Driver circuit, Microcontroller & stepper motor

Sample-and-hold circuit

The sample and hold circuit has been implemented and tested on Veroboard. As with the motor driver, in the final implementation of the project the sample and hold circuit will be designed onto a PCB.

The code to automatically control the sample and hold circuit has not yet been completed. However, the circuit still operates exactly as it should and is fully testable without the C code. Figure 4 shows the circuit on Veroboard along with the designated terminals which will be connected to the search coil/microcontroller in the final implementation.

In the final implementation, the connections to the microcontroller and search coil will be very simple: The 3-axis search coil simply plugs into the six-pin header on the Veroboard. One digital output from the microcontroller will be used to dump all the capacitors at once. Three data lines will input into the Microcontrollers ADC pins. The C code that will be used to control the sample and hold circuit has been previously explained, but this is a relatively simple task to implement as the operation of the code has already been designed.

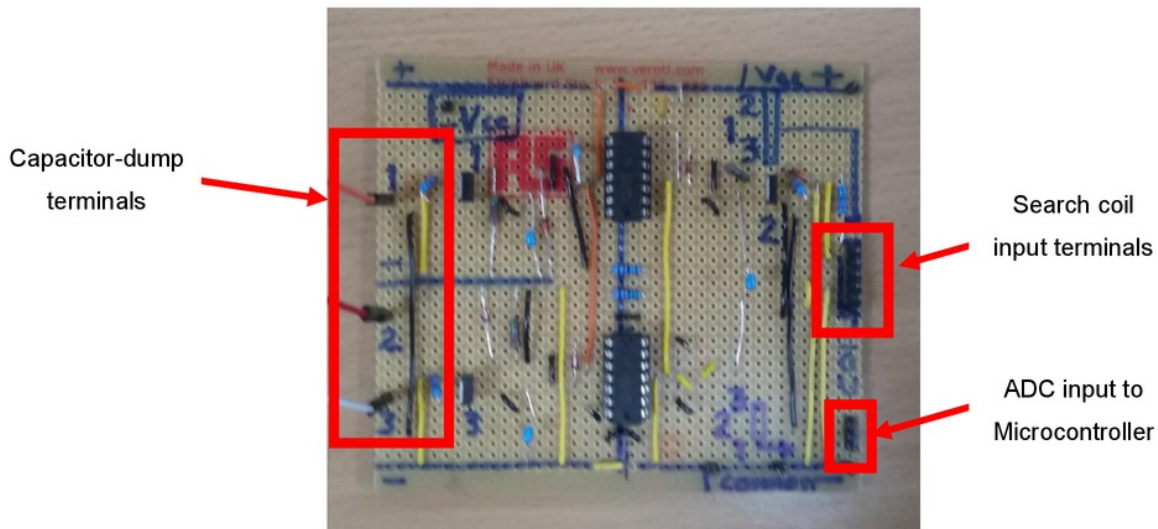


Figure 4 - Sample and hold Veroboard circuit

Microcontroller C code explanation

The C code shown in this section of the report is what was necessary to prove that the implementation of the DC stepper motor driver works as expected. Only the configuration of the Microcontroller and the Main loop are shown in this section. The function prototypes and function definitions are available under the heading 'C code functions' in the Appendix.

The C code under the heading 'C code print-out' shows the working C code that has been implemented onto the microcontroller so far. Although the EEPROM and interrupt have not been fully implemented, the C code clearly shows that the configuration has been complete and these features can easily be added at a later date.

The function prototypes and definitions show that all the functions have already been set up to write and read to and from the EEPROM. The 'LONGNUM UNION' in the C code is used to automatically convert large numbers to a series of chars, which can easily be written to the EEPROM in 'char' sized chunks. The free EEPROM addresses have also been defined as well as the first free addresses that are available.

An empty interrupt has been setup that will activate when TIMER1 overflows. Although this is not the interrupt that will be used in the final implementation, it serves as a placemark to modify with the correct interrupt vector in the future. TIMER1s registers and settings have been set up using a function.

The USART communications bus has been setup using a baud rate of 19200. According to the manufacturers datasheet, this baud rate gives a low percentage error (0.2%) when used with a clock frequency of 16MHz, and with the USART registers correctly set (Microchip, 2015, p. 165). The USART initialisation sequence is contained within a function. The equation for calculating the MYUBRR value that is correct for a specific baud rate, clock frequency and prescaler was taken from the manufacturers datasheet.

Terms have been defined to easily allow each bridge of the stepper motor driver to be turned on and off independently (e.g. MOTOR_Apos_DRIVE). This allows the actual programming of the whole and half step sequence to be easily readable by humans.

The definition of MOTOR_ON_TIME is how long each bridge is energised. Once the bridges have been energised for a particular step in the sequence, a for loop will count down the value that is defined by MOTOR_ON_TIME. This can be used to effectively speed the motor up or down by stepping it through the sequence faster or slower.

The remaining parts of the configuration code simply set up the I/O ports and sets the state of all outputs to 0. The functionality of the actual step sequence is easily explained by examining

Figure 13. The code that turns a particular bridge has already been explained. With the whole step sequence, each individual step requires both bridges to be driven in the direction shown in Figure 13. The bridges are energised for the count defined in MOTOR_ON_TIME. Once all four steps of the sequence are completed, the sequence repeats from step 1. This can continuously rotate the motor indefinitely providing a higher torque than half-step sequencing. The half-step sequence is included in the code but is disabled by commenting the code out. Half-step has twice as many steps in the sequence, provides less torque but runs smoother than the full step sequence.

In the final implementation, control measures will be programmed to count the number of steps the motor is driven, and relate the number of steps to linear distance travelled on a particular axis. This will be used as an alternative to using positional encoders.

C code print-out

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdbool.h>
#include <stdio.h>
#include <math.h>

////////////////////////////////////
//////////////////////////////////// Definitions //////////////////////////////////
////////////////////////////////////

#define F_CPU 16000000UL
#define BAUD 19200
//define baud for USART
#define MYUBRR (((long)(F_CPU))/(long)(BAUD*16UL))-1)
// set baud rate value for UBRR for USART

#define UNION_NO_OF_BYTES          4

#define EEPROM_SIZE                512

#define EE_FREE_ADDR_BYTE4         0x04           //MSB
#define EE_FREE_ADDR_BYTE3         0x05
#define EE_FREE_ADDR_BYTE2         0x06
#define EE_FREE_ADDR_BYTE1         0x07           //LSB

#define ADDR_BYTE4_CHAR             0x00           //MSB
#define ADDR_BYTE3_CHAR             0x01
#define ADDR_BYTE2_CHAR             0x02
#define ADDR_BYTE1_CHAR             0x03           //LSB

#define RXPIN                       DDD0
#define TXPIN                       DDD1

#define MOTOR_Apos_DRIVE             (PORTB |= (1<<DDB0))
#define MOTOR_Apos_STOP              (PORTB &= ~(1<<DDB0))
#define MOTOR_Aneg_DRIVE             (PORTB |= (1<<DDB1))
#define MOTOR_Aneg_STOP              (PORTB &= ~(1<<DDB1))
#define MOTOR_Bpos_DRIVE             (PORTB |= (1<<DDB2))
#define MOTOR_Bpos_STOP              (PORTB &= ~(1<<DDB2))
#define MOTOR_Bneg_DRIVE             (PORTB |= (1<<DDB3))
#define MOTOR_Bneg_STOP              (PORTB &= ~(1<<DDB3))

#define MOTOR_ON_TIME                26000
```

```

typedef union
{
    char str[UNION_NO_OF_BYTES];
    unsigned long num;
}LONGNUM;

////////////////////////////////////
//////////////////////////////////// Global variables ///////////////////////////////////
////////////////////////////////////

char tmpstr[100];
static uint16_t clockOffset = 159;          //Generates an interrupt every 10uS

////////////////////////////////////
//////////////////////////////////// ISR ///////////////////////////////////
////////////////////////////////////
ISR(TIMER1_COMPA_vect)
{
}

////////////////////////////////////
//////////////////////////////////// Main Loop ///////////////////////////////////
////////////////////////////////////

int main(void)
{
    DDRD &= ~(1 << RXPIN);                //Set USART rx as input
    DDRD |= (1 << TXPIN);                  //Set USART TX pin as output

    DDRB = 0b11111111;                    //Set PB3, PB2, PB1 and PB0 as outputs
    PORTB = 0b00000000;                    //Set port B outputs to 0

    USART_Init(MYUBRR);                    //Initialize USART
    CLOCK_init();                          //Initialize clock
    while (1)
    {
        ///////////////////////////////////
        /////////////////////////////////// WHOLE STEP SEQUENCE ///////////////////////////////////
        ///////////////////////////////////

        int x;
        MOTOR_Bpos_STOP;                    //B+ OFF
        MOTOR_Apos_DRIVE;                    //A+
        MOTOR_Bneg_DRIVE;                    //B-
        for (x=0;x<MOTOR_ON_TIME;x++){
            MOTOR_Apos_STOP;                    //A+ OFF
            MOTOR_Aneg_DRIVE;                    //A-
            for (x=0;x<MOTOR_ON_TIME;x++){
                MOTOR_Bneg_STOP;                    //B- OFF
                MOTOR_Bpos_DRIVE;                    //B+
                for (x=0;x<MOTOR_ON_TIME;x++){
                    MOTOR_Aneg_STOP;                    //A- OFF
                    MOTOR_Apos_DRIVE;                    //A+
                    for (x=0;x<MOTOR_ON_TIME;x++){

        ///////////////////////////////////
        /////////////////////////////////// HALF STEP SEQUENCE ///////////////////////////////////
        ///////////////////////////////////
        //int x;
        //
        //MOTOR_Apos_DRIVE; //A+
        //MOTOR_Bneg_DRIVE; //B-
        //for (x=0;x<MOTOR_ON_TIME;x++){
        //
        //MOTOR_Apos_STOP;
        //for (x=0;x<MOTOR_ON_TIME;x++){
        //
        //MOTOR_Aneg_DRIVE;
        //for (x=0;x<MOTOR_ON_TIME;x++){
    }
}

```

```
    //  
    //MOTOR_Bneg_STOP;  
    //for (x=0;x<MOTOR_ON_TIME;x++){  
    //  
    //MOTOR_Bpos_DRIVE;  
    //for (x=0;x<MOTOR_ON_TIME;x++){  
    //  
    //MOTOR_Aneg_STOP;  
    //for (x=0;x<MOTOR_ON_TIME;x++){  
    //  
    //MOTOR_Apos_DRIVE;  
    //for (x=0;x<MOTOR_ON_TIME;x++){  
    //  
    //MOTOR_Bpos_STOP;  
    //for (x=0;x<MOTOR_ON_TIME;x++){  
  
    }  
}
```

5. Testing

The system has been tested to ensure that the criteria set out in the FDS have been met. The system that was built to measure the magnetic field will be referred to as 'field plotter' for this section.

Completed testing table

The following testing table shows a summary of each test, the method used to perform each test, whether the test passed or failed and the evidence of the testing. The test table directly relates to the criteria specified in the FDS in assignment 1 (Lewis, 2019).

No.	Test	Test Method	Pass/Fail	Evidence
1	a magnetic field of at least 2 teslas must be capable of being measured by the field plotter.	A magnetic stimulator was setup and configured to produce a 2-tesla magnetic field at the centre of the treatment coil. The field plotters search coil was placed directly on the treatment coil. The field plotter must be able to measure this magnetic field without clipping up to the supply rails of the op-amps. The signals leading into the ADC must not exceed 5V when measuring the 2T field produced by the magnetic stimulator.	PASS	Figure 15 Figure 16 Figure 17
2	the field plotter must be able to move in three-dimensional space.	On the home screen the manual control option will allow the machine to be moved in the X, Y and Z directions independently. Each axis should move back and forth using this screen.	FAIL	
3	the field plotter must be able to measure a magnetic field in a 20*20*20cm area.	Using the manual control mode on the machine can be used to bring the search coil to each corner of the test area (test area = 20*20*20cm). Produce a magnetic field using a magnetic stimulator and measure the amplitude of the field in each corner. If a reading is successfully taken in each corner of the test jig, it's reasonable to say that the maximum measurable area of the jig is 20*20*20cm.	FAIL	

No.	Test	Test Method	Pass/Fail	Evidence
4	The jig must have three search coils perpendicular to the x, y and z axis.	The 3-axis search coil must be perpendicular to the X, Y and Z axis of movement. This search coil must remain perpendicular to the axis through all X, Y and Z positions the jig moves in.	FAIL	
5	A magnetic field created by a magnetic stimulator must be sample and held to find the peak magnetic field output of the stimulator.	One coil will be wired into the 3 search coil inputs on the sample and hold circuit. The stimulator will be fired and the circuit must charge the capacitors to a corresponding voltage. the value of all three capacitors must be consistent with each other.	PASS	Figure 15 Figure 16 Figure 17
6	The values stored by the capacitors are stored on the EEPROM	The magnetic stimulator is fired and with the search coil placed on the treatment coil. The Microcontrollers ADC processes this analogue voltage and converts it to a digital equivalent. The digital equivalent will then be transferred to be stored on the EEPROM. This can be proved by printing the contents of the EEPROM to the terminal and comparing the actual value to the value that is expected.	FAIL	
7	magnetic field value outside the test jig (20cm x 20cm x 20cm) must be below 500 microTeslas	With the magnetic stimulator coil being placed in front of the jig, the stimulator is fired to produce a magnetic field of 2T. A gaussmeter is used to scan the perimeter of the 20*20*20 jig and the maximum measurement is recorded. To pass this test the maximum measured value on the gaussmeter must be below 500uT.	FAIL	Figure 18
8	The positional accuracy for all axis must be +/- 1mm	This test will have to be performed on each axis individually. Mark the starting point of one axis. Use the manual mode on screen to move that axis a fixed distance (5cm for example). Measure how far the axis actually travelled and deduct this value from the fixed distance the axis was supposed to move. The result from this must be lower/greater than +/- 1mm	FAIL	

No.	Test	Test Method	Pass/Fail	Evidence
9	The noise emissions outside of the field plotter must be below 75db	When the machine is running, use an acoustic measuring device to measure the noise emissions where a user would stand to operate the field plotter. The measured noise must be below 75dB.	PASS	Figure 19
10	The field plotter must automatically power off once the machine has been idle for a short period of time	Turn on the field plotter and ensure that it is not running on automatic mode. Leave the machine idle for 5 minutes without touching any controls and observe that the field plotter automatically turns off.	FAIL	

Test results explanation

Test 1 states that a magnetic field of at least 2 teslas must be capable of being measured. A magnetic field of 2 teslas was generated using a magnetic stimulator operating a monophasic pulse. The field strength was confirmed using a calibrated gaussmeter. The field plotter was then placed in the same strength magnetic field and the reading on screen should show 2 teslas. The test would have failed if the field plotter is not able to measure this strength of field. Test 1 is very important as this is one of the core components that is required to measure be able to create a fully auto magnetic field plotter. Test 1 passed as the search coil/sample-and-hold circuit setup was able to measure a magnetic field with a strength of 2T. The testing evidence (Figure 15, Figure 16 & Figure 17) shows each channel of the sample hold circuit charging the capacitor to the peak value of the inputted magnetic field. The attenuation has been accounted for on the oscilloscope. The scope channel 'CH2' is the search coil output that is the input to the sample and hold circuit.

Test 2 states that the field plotter must be able to move in three-dimensional space. This is not entirely possible as the budget for the mechanical gantry has been denied. This test is therefore a failure. However, the second test could have been modified to involve driving the stepper motor a particular number of steps and calculating how far the gantry would have moved. The basic operation of the X, Y axis would be pulley and belt driven. The distance the gantry would actually move depends on what sized pulley was used at the shaft of the motor. If the test plan was modified the parts of the project that have been fully implemented could have been used to prove that the field plotter could successfully move in 3D space.

Test 3 involves measuring a magnetic field in an area of at least 20cm x20cm x 20cm. As the mechanical side of the project had not been implemented, the results from this test will match test 2. If test 2 is completed, it can be assumed that duplicating the axis will allow the jig to

move in three-dimensional space. Test 3 could also be modified to include counting pulses from the stepper motor and calculating how far the field plotter could move if the gantry was fully implemented.

Test 4 states that the jig must have three search coils perpendicular to the x, y and z axis. As the search coil has already been constructed with 3 coils, it can be assumed that this will be mounted perpendicular to each axis. This test depends on the style of gantry used. As the axes on a 3-axis cartesian gantry stay parallel to each axis, it can be safely assumed that what is mounted to the gantry will also stay parallel. As this is all based on calculations and cannot actually be proven, this test is a failure.

Test 5 needs to show that the three search coil readings are sample and held to find the peak magnetic field output of the stimulator. This can be proven by taking appropriate oscilloscope readings of the magnetic field output of the stimulator and captured waveforms. Figure 15, Figure 16 and Figure 17 shows the same magnetic field being input into each channel of the sample and hold circuit. Each screenshot shows that the sample and hold circuit charges to roughly 2.4v-2.6v. This shows consistency between each of the sample and hold circuits when being fed the same input.

Test 6 involves proving that the values captured by the sample and hold circuits are stored onto the EEPROM. As the projects time and budget has been cut from this point, only the design of the code can be analysed, as this part has not yet been implemented. The C code functions in 'C code functions' section shows that all the code needed to read and write to the EEPROM has already been written. The code for calculating the free EEPROM address has also been written. The code has been tested for functionality but has not been implemented into the project at this time.

Test 7 involves measuring the magnetic field value outside the test jig (20cm x 20cm x 20cm) is kept below 500 microTeslas, and to determine if the jig needs shielding to reduce the magnetic field. As the mechanical side of the project has not been implemented, the stimulator will be fired to produce a 2-tesla magnetic field and a gaussmeter will be used to measure the magnetic field in a radius around the stimulator coil. Figure 18 shows a 20*20cm box around the stimulator coil that simulates the size of the machine's enclosure. A gaussmeter was used 20cm above the corner of the 20*20cm box to measure the magnetic field strength at that point. The point chosen represents the corner of the machine's enclosure. With no shielding, the peak field strength at the chosen point is 0.04T (40mT), which well above the target of <500uT. Although this test failed, the test confirmed that EMF shielding will be required to bring down the magnetic field strength outside the field plotters enclosure.

Test 8 involves measuring the positional accuracy of all axes, which must be ± 1 mm. As the gantry of the project does not exist, this test is a fail. However, the accuracy of the motor could be calculated based on number of steps and the mechanical system used to drive each axis, see Figure 1. How the accuracy of the axis can be calculated is explained in the *Control scheme for axis* section.

This test requires the noise emissions from the jig to be measured, and be under 75dB. As the highest noise emissions will come from the stimulator coil this test could still be performed. An acoustic measuring device was placed 1m from the stimulator coil and was set on 'peak-hold' mode. The stimulator was fired and a maximum noise emission measurement of 63.7dBA was measured, which is under the 75dBA that was specified. See Figure 19 for the test setup and measurement.

Test 10 involves proving the auto-power-off feature of the field plotter. As this has not been fully implemented in the C code yet, this test is a fail.

6. Evaluation

The testing table that was derived from the FDS in assignment 1 clearly shows that the project did not meet the specification and therefore the project failed in that respect. Only 3 out of 10 of the requirements set out in assignment 1 were met; the reasons for these failures has been explained clearly in the report.

The current expenses for the project are: £0. After undertaking the planning stages of the project, upper management decided that the best course of action was to create a proof-of-concept for as little cost as possible. Each sub task within the project was broken down and designed with this new goal in mind. The major parts of the project have been implemented in such a way to show upper management that this is possible for a low price. The component choices were very limited due to this added constraint.

The following components were available in the workplace and did not have to be purchased:

Stepper motor, P & N channel MOSFETS, ATmega 328p NANO, Quad package Op-amps, Capacitors, Resistors and Diodes.

The components listed above were ample to implement the design in such a way to show its potential to upper management. Once the proof of concept has been proven and extra budget is signed off, a full component list can be drawn up based on the components used in the proof-of-concept. This full component list should predict the component cost fairly accurately.

The time management side of the project was very poor. Due to the time constraints being changed after the planning stage of the project was completed, the Gantt chart created in assignment 1 was not followed as it should have been. See Figure 20 for the Gantt chart taken from assignment 1. The Gantt chart shows that only some of the sub-tasks were completed to 100%. This was due to the changing constraints throughout the project's life. The sub-tasks on the Gantt chart show that some tasks have had 0% completion; this is not necessarily the case as some of the tasks have already been shelled out in the C code.

Even though the final product does not match the FDS, the parts that have been implemented serve to prove the concept to upper management with the intention of convincing them to release the budget to implement the remaining parts of the project. The parts of the project that have not been implemented yet have been designed in a way to prove that these techniques would work, as well of many of the features being implemented in the functions of the C code only. More budget and time allocated to the project would allow these designs to be realised and fully implemented to create a fully fledged magnetic field plotter.

7. Conclusion and recommendations

The design phase went fairly well, even though the Gantt chart was not followed exactly. The sample and hold circuit work as intended and has been designed to easily interface with the chosen microcontroller. The motor driver was designed for use with many different stepper motors, as the power requirement for the motor are not known yet due to the mechanical gantry being incomplete. The control scheme for the axes has been designed and in theory should work as intended when implemented in full. The menu screen has been designed to spec showing the user all of the functions that the machine has.

The implementation phase of the project did not go very smoothly. Although most of the design work was complete, the time taken to actually implement the designs took far longer than expected. The stepper motor driver was implemented and tested with different modes of operation selectable from the C code. The driver was tested and found to work with various sized/rated stepper motors. The sample and hold circuits were implemented but the code needed to automatically control it has not been implemented. The sample and hold circuit functions to specification when being tested independently, and successfully holds the peak magnetic field value of the magnetic field being generated from the TMS machine. Although not all of the C code has been fully implemented, many of the functions needed to fully implement all of the code has already been written. The fact that much of the functions have already been written means that less time is needed to implement the C code in full on the finished product.

Recommendations

The overall scope of the project was too large to be completed in the time frame given and too many interdependent systems had to work together for the final product to work as specified. In many cases the sub tasks were assumed to be simple and quick to design and implement, but in reality, these tasks took a lot more time than originally estimated. Simplifying the overall aim of the project would have been beneficial in hindsight.

Although many of the parts that were implemented worked to spec, the time taken to implement them took far longer than expected and led to the final product being relatively basic and not automated.

The time taken to integrate all of the separate parts of the project together was not taken into account when designing the project. This is a very time-consuming part of the project and this oversight led to the illusion of having lots of time spare until it was time to implement this part.

8. Citation List

Works Cited

Hopkins, T., 2012. *AN235 Application note*, s.l.: ST microelectronics.

Lewis, M., 2019. *SGEE6003 Project Definition, Research and Quality Planning*, Llanelli: Coleg Sir Gar Graig.

Microchip, 2015. *ATmega328P Datasheet*. [Online]
Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf#page=89&zoom=100,0,68
[Accessed 15 3 2019].

NXP semiconductors, 2016. *Driving DC and Stepper motors*, Eindhoven: NXP semiconductors.

PCB linear, n.d. *Stepper motor nema 17*, Rockford: PCB linear.

9. Appendix

Photos

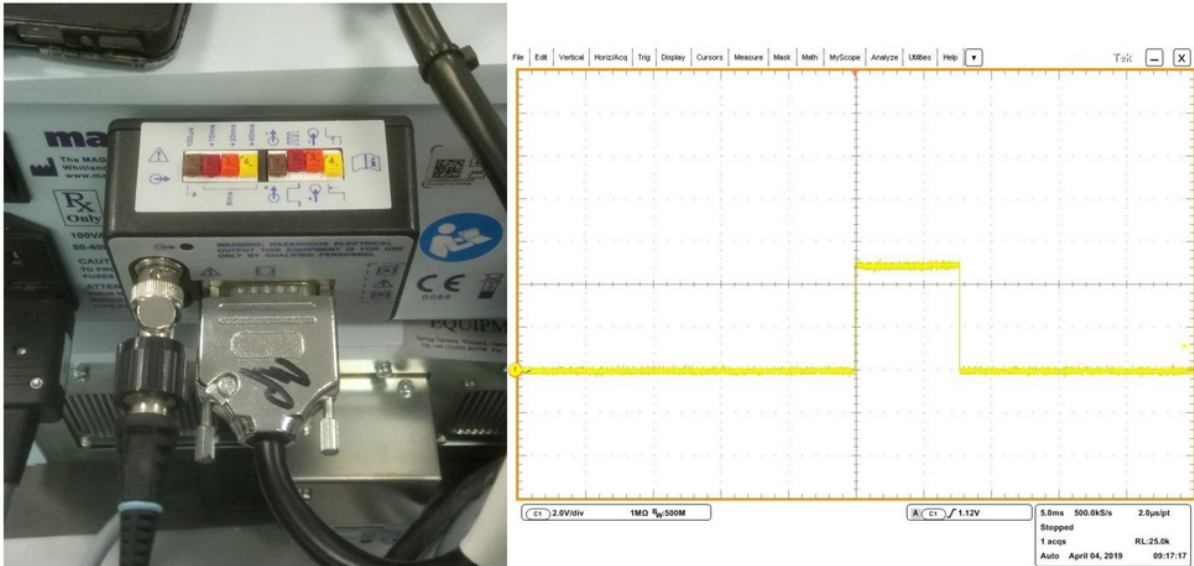


Figure 5- 'TrigOut' breakout box, 8ms 'TrigOut' pulse

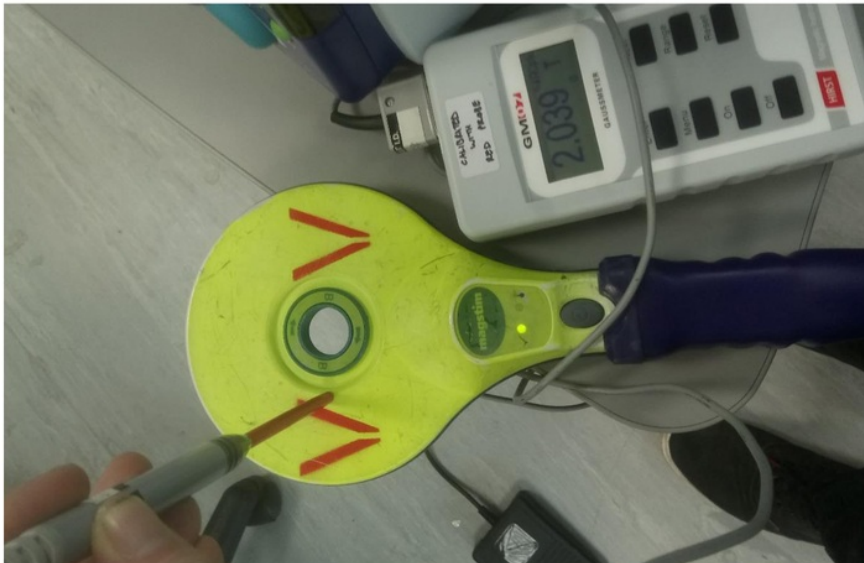


Figure 6- 2 Tesla measured on gaussmeter (reading 2.039T)

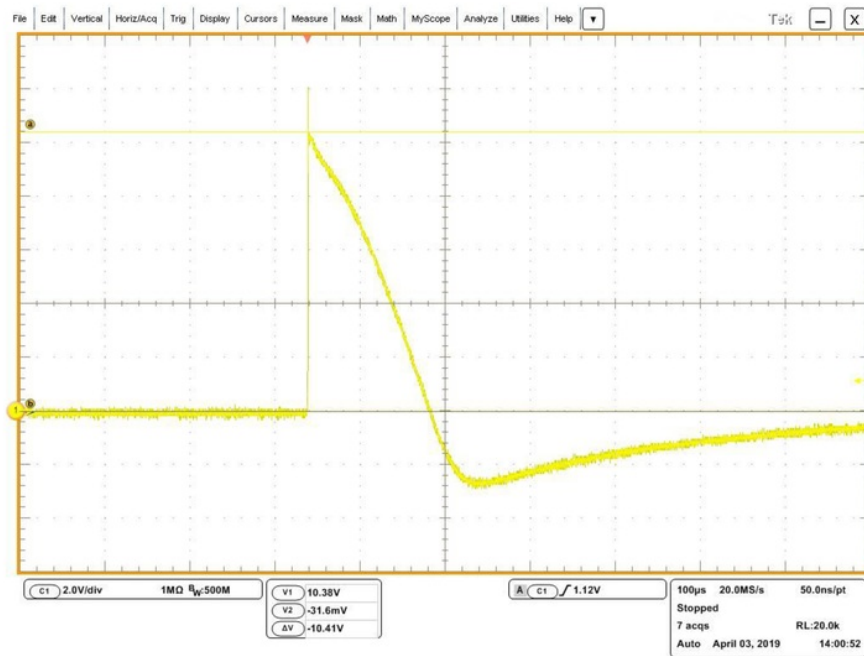


Figure 7 - 2T scope output from search coil



Figure 8 - Search coil measuring magnetic field with scope output

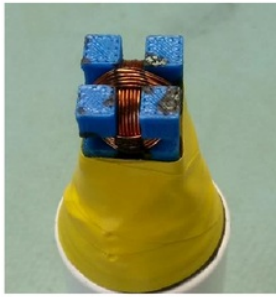


Figure 9- Search coil: Closeup of X, Y and Z windings

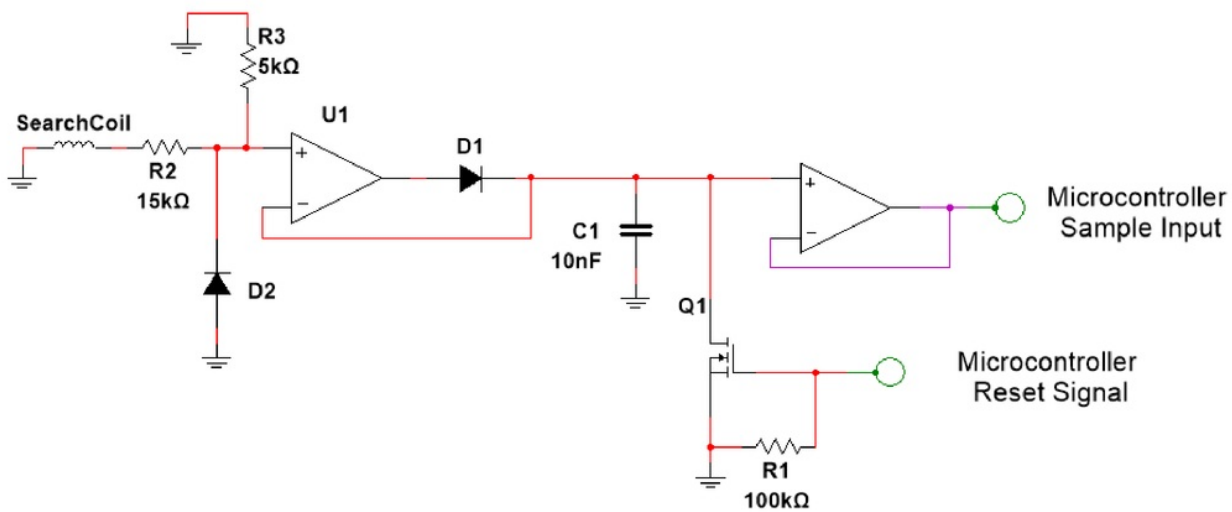


Figure 10- sample and hold circuit diagram

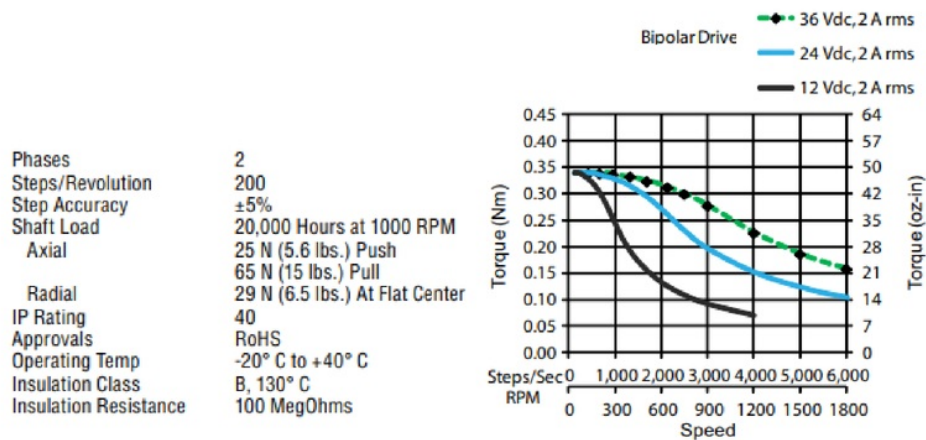


Figure 11 - Nema 17 stepper motor specifications, motor torque speed curve

(PCB linear, n.d.)

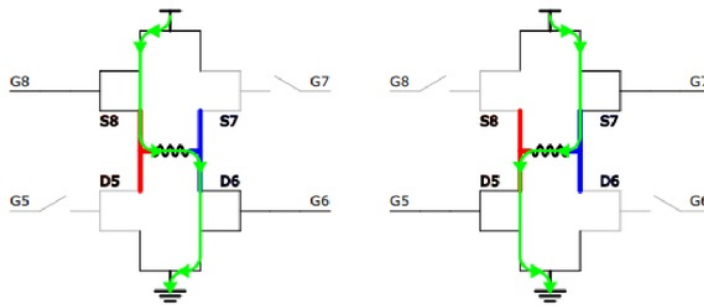


Figure 12- Motor bridge operation
(NXP semiconductors, 2016)

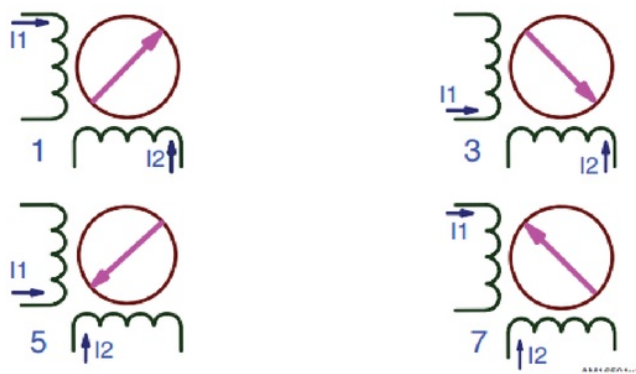


Figure 13 – full- step sequence
(Hopkins, 2012)

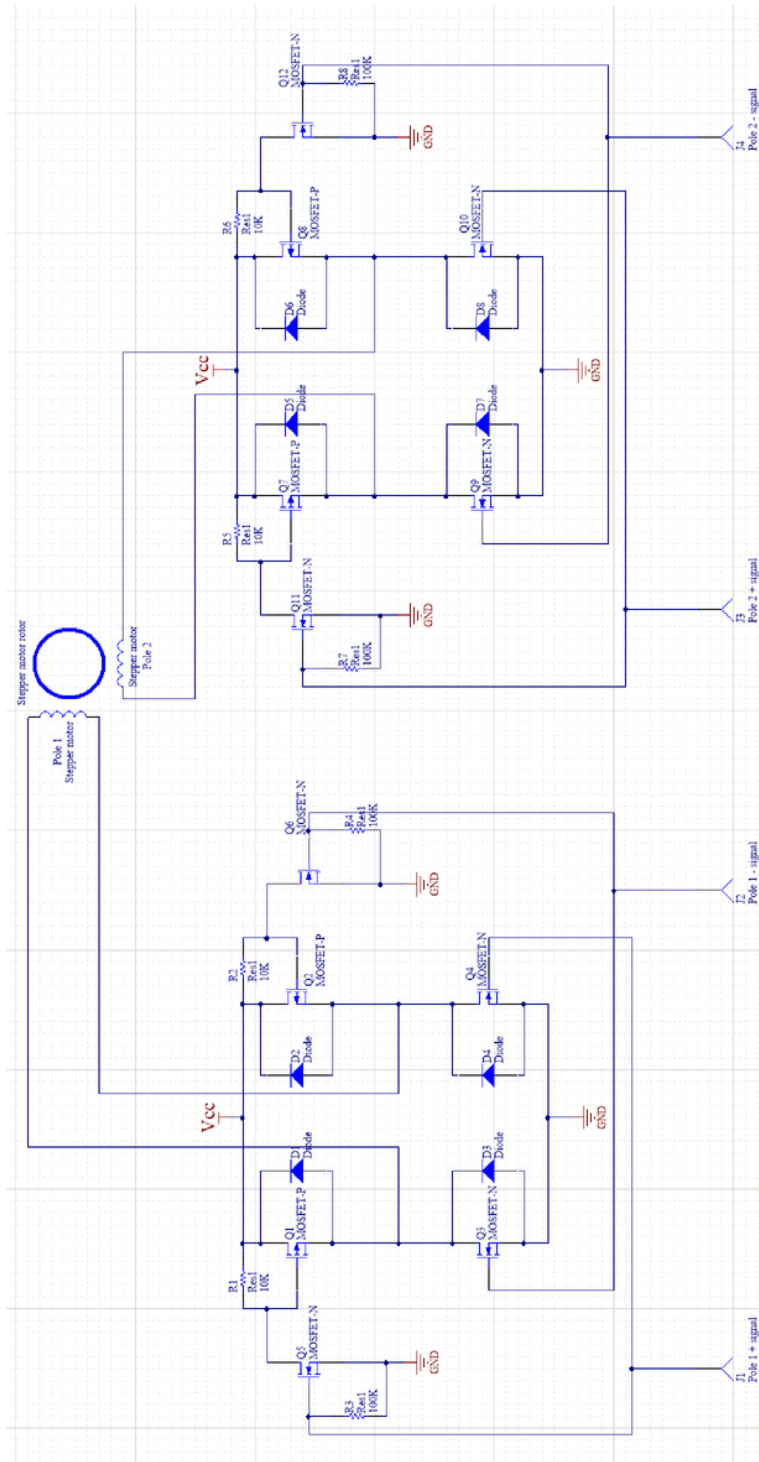


Figure 14- Stepper Motor Driver circuit

Testing evidence

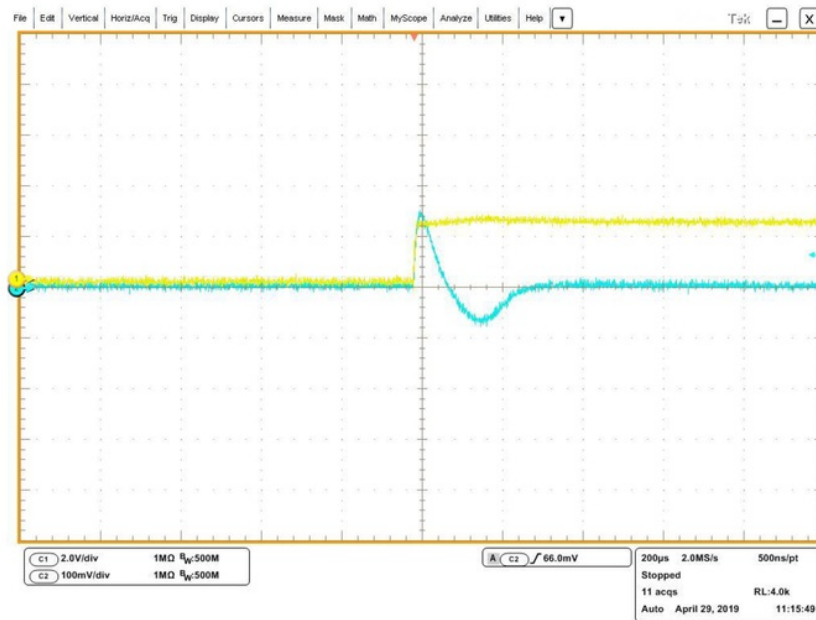


Figure 15 – Test 1&5 - Testing sample-and-hold - Channel 1

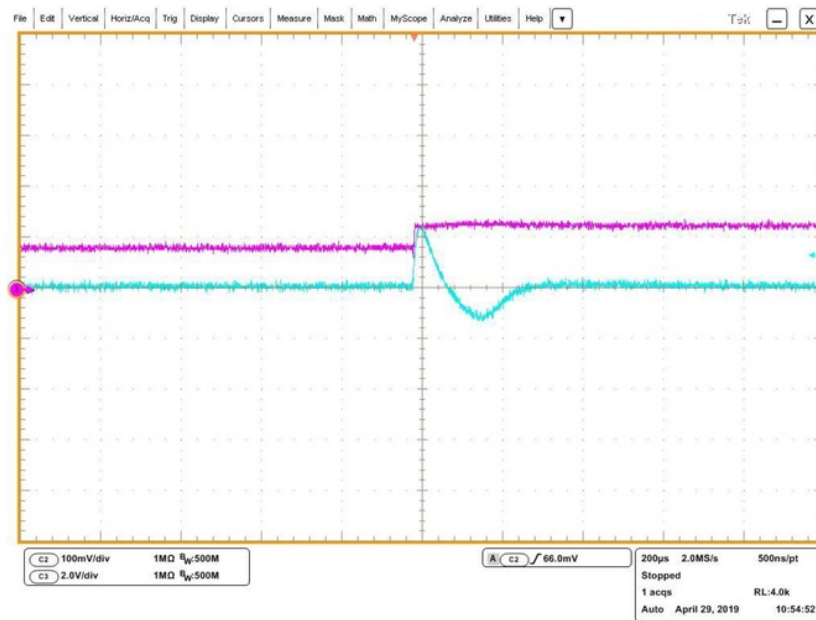


Figure 16 - Test 1&5 - Testing sample-and-hold - Channel 2

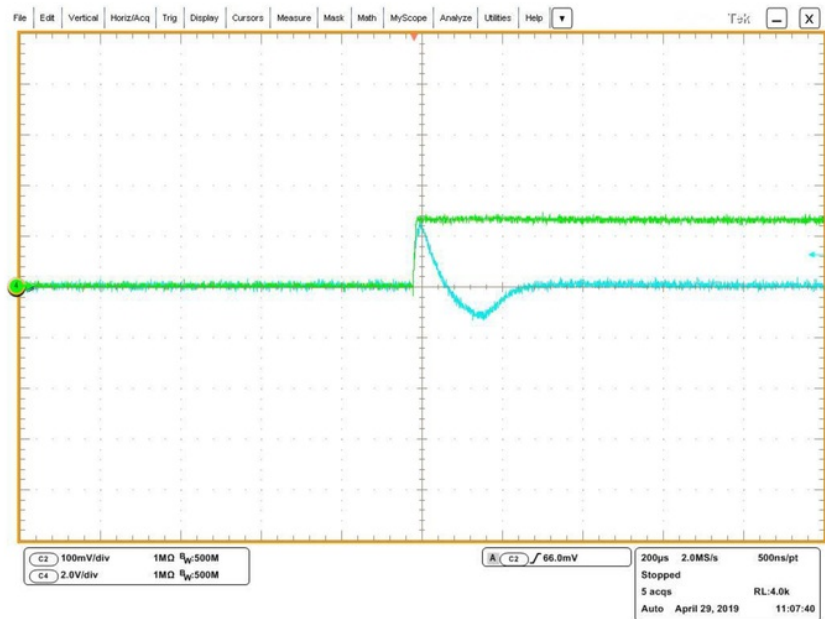


Figure 17 - Test 1&5 - Testing sample-and-hold - Channel 3

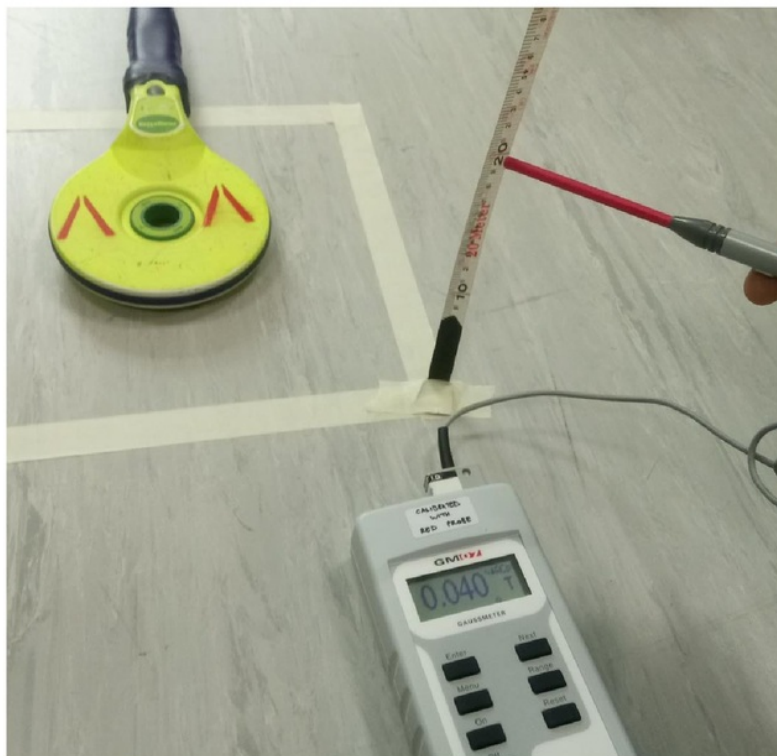


Figure 18 - Test 7 - Magnetic field outside machine enclosure



Figure 19 - Test 9 - Acoustic measurement test

C code functions

```
////////////////////////////////////
//////////////////////////////////// Function Prototypes //////////////////////////////////////
////////////////////////////////////

void CLOCK_init(void);
void USART_send( unsigned char data);
void USART_putstr(char* StringPtr);
void USART_Init(unsigned int ubrr);
unsigned char USART_Receive( void );
void USART_Flush( void );

void EEPROM_write(unsigned int uiAddress, unsigned char ucData);
unsigned char EEPROM_read(unsigned int uiAddress);

void read_EE_count(LONGNUM *total);
void write_EE_count(LONGNUM *total);
void reset_all_EEPROM(LONGNUM *total);
void display_all_EEPROM (void);
void write_junk_EEPROM(void);

void write_free_EE_addr(LONGNUM *free_EE_addr);
void read_free_EE_addr(LONGNUM *free_EE_addr);
void display_next_free_EE_address(LONGNUM *free_EE_addr);

void MAI_Delay(long unsigned int delaytime_ms);
void display_count(LONGNUM *total);
void display_count_short(LONGNUM *total);
void display_count_debug(LONGNUM *total);
void display_menu(void);

////////////////////////////////////
//////////////////////////////////// FUNCTION definitions //////////////////////////////////////
////////////////////////////////////

void record_brief_discontinuity(LONGNUM *free_EE_addr,LONGNUM *total, bool *discontinuity_recorded)
{
    unsigned int temp_addr;
    unsigned char temp_data0, temp_data1, temp_data2, temp_data3;

    cli();
    temp_addr = free_EE_addr->num;
    temp_data3 = total->str[3];
    temp_data2 = total->str[2];
    temp_data1 = total->str[1];
    temp_data0 = total->str[0];
    EEPROM_write(temp_addr, temp_data3);
    EEPROM_write(temp_addr+1, temp_data2);
    EEPROM_write(temp_addr+2, temp_data1);
    EEPROM_write(temp_addr+3, temp_data0);
    free_EE_addr->num = free_EE_addr->num + 4;
    write_free_EE_addr(free_EE_addr);
    *discontinuity_recorded = true;
    USART_putstr("\n\nrecorded\n\n");
    sei();
}

void display_all_EEPROM (void)
{
    cli();
    int x,y = 0,z;
```

```

for(x = 0; x<EEPROM_SIZE; x++)
{
    while(EECR & (1<<EEPE));
    /* Set up address register */
    EEAR = x;
    /* Start eeprom read by writing EERE */
    EECR |= (1<<EERE);
    /* Return data from Data Register */
    //return EEDR;
    z = x + 3;
    sprintf(tmpstr,"Address: 0x%03X to 0x%03X\t=",(unsigned int)x,(unsigned int)z);
    if (y == 0)        USART_putstring(tmpstr);
    sprintf(tmpstr,"%02X\t", (unsigned int)EEDR);
    USART_putstring(tmpstr);
    if(x == 3)        USART_putstring("=Total bend count");
    if(x == 7)        USART_putstring("=Next free EEPROM address");
    if(x == 11)       USART_putstring("=Storage for temp discontinuity");
    y++;
    if (y >= 4)
    {
        y = 0;
        USART_putstring("\n\n");
    }
}
sei();
}

void display_menu(void)
{
    USART_putstring("\n\nWelcome to the MAGSTIM bender-unit\n\n");
    USART_putstring("Press '1' to start bending\n\n");
    USART_putstring("Press '2' to print EEPROM data\n\n");
    USART_putstring("Press '0' to reset ALL EEPROM data 0\n\n");
}

void display_count(LONGNUM *total)                //Displays the total count on the PC terminal
{
    USART_putstring("Count =\t");
    sprintf(tmpstr," %ld\n", (unsigned long)total->num);
    USART_putstring(tmpstr);
    USART_putstring("\n\n");
}

void display_count_debug(LONGNUM *total)           //Displays the total count on the PC terminal
{
    USART_putstring("Count =\t");
    sprintf(tmpstr," %ld\t = \t", (unsigned long)total->num);
    USART_putstring(tmpstr);
    sprintf(tmpstr," %i", (unsigned int)total->str[3]);
    USART_putstring(tmpstr);
    USART_putstring("\t");
    sprintf(tmpstr," %i", (unsigned int)total->str[2]);
    USART_putstring(tmpstr);
    USART_putstring("\t");
    sprintf(tmpstr," %i", (unsigned int)total->str[1]);
    USART_putstring(tmpstr);
    USART_putstring("\t");
    sprintf(tmpstr," %i", (unsigned int)total->str[0]);
    USART_putstring(tmpstr);
    USART_putstring("\n\n");
}

void display_count_short(LONGNUM *total)           //Displays the total count on the PC terminal
{
    sprintf(tmpstr," %ld", (unsigned long)total->num);
    USART_putstring(tmpstr);
}

```

```

}

void write_EE_count(LONGNUM *total)
{
    cli();
    EEPROM_write(ADDR_BYTE4_CHAR, total->str[3]);
    EEPROM_write(ADDR_BYTE3_CHAR, total->str[2]);
    EEPROM_write(ADDR_BYTE2_CHAR, total->str[1]);
    EEPROM_write(ADDR_BYTE1_CHAR, total->str[0]);
    sei();
}

void read_EE_count(LONGNUM *total)           //Reads the stored 'bend' count from EEPROM
{
    cli();
    total->str[3] = EEPROM_read(ADDR_BYTE4_CHAR);
    total->str[2] = EEPROM_read(ADDR_BYTE3_CHAR);
    total->str[1] = EEPROM_read(ADDR_BYTE2_CHAR);
    total->str[0] = EEPROM_read(ADDR_BYTE1_CHAR);
    sei();
}

void write_free_EE_addr(LONGNUM *free_EE_addr)
{
    cli();
    if ((free_EE_addr->num)>=0x1FC)         free_EE_addr->num = 0x1FC;
    EEPROM_write(EF_FREE_ADDR_BYTE4, free_EE_addr->str[3]);
    EEPROM_write(EF_FREE_ADDR_BYTE3, free_EE_addr->str[2]);
    EEPROM_write(EF_FREE_ADDR_BYTE2, free_EE_addr->str[1]);
    EEPROM_write(EF_FREE_ADDR_BYTE1, free_EE_addr->str[0]);
    sei();
}

void read_free_EE_addr(LONGNUM *free_EE_addr)
{
    cli();
    free_EE_addr->str[3] = EEPROM_read(EF_FREE_ADDR_BYTE4);
    free_EE_addr->str[2] = EEPROM_read(EF_FREE_ADDR_BYTE3);
    free_EE_addr->str[1] = EEPROM_read(EF_FREE_ADDR_BYTE2);
    free_EE_addr->str[0] = EEPROM_read(EF_FREE_ADDR_BYTE1);
    if ((free_EE_addr->num)>=0x1FC)         free_EE_addr->num = 0x1FC;
    sei();
}

void display_next_free_EE_address(LONGNUM *free_EE_addr)           //Displays the total count on the
PC terminal
{
    sprintf(tmpstr, "%03X\n\n\r", (unsigned int)free_EE_addr->num);
    USART_putstring(tmpstr);
    USART_putstring("\n\n\r");
}

void reset_all_EEPROM(LONGNUM *total)           //Resets all values on the EEPROM to 0
{
    cli();
    int x;
    for(x = 8; x<EEPROM_SIZE; x++)
    {
        EEPROM_write(x, 0xff);
    }
    sei();
}

void write_junk_EEPROM(void)           //Resets all values on the EEPROM to 0
{
    cli();
    int x;

```

```

        for(x = 0; x<EEPROM_SIZE; x++)
        {
            EEPROM_write(x, 0x69);
        }
    sei();
}

void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
    /* Wait for completion of previous write */
    while(EECR & (1<<EEPE));
    /* Set up address and Data Registers */
    EEAR = uiAddress;
    EEDR = ucData;
    /* Write logical one to EEMPE */
    EECR |= (1<<EEMPE);
    /* Start eeprom write by setting EEPE */
    EECR |= (1<<EEPE);
}

unsigned char EEPROM_read(unsigned int uiAddress)
{
    /* Wait for completion of previous write */
    while(EECR & (1<<EEPE));
    /* Set up address register */
    EEAR = uiAddress;
    /* Start eeprom read by writing EERE */
    EECR |= (1<<EERE);
    /* Return data from Data Register */
    return EEDR;
}

void CLOCK_init(void)
{
    cli(); //Disable interrupts
    //Set waveform generation mode and prescaler value of 1
    TCCR1B |= (1 << WGM12)|(1<<CS10);
    //Enable interrupt for: Output compare A match
    TIMSK1 |= (1<<OCIE1A);
    // initialize counter value
    TCNT1 = 0;
    //OCR1A controls frequency of PWM (PB1), OCR1B is phase shift for second PWM wave (PB2)
    OCR1A = clockOffset;
    //enable interrupts
    sei();
}

void USART_send( unsigned char data)
{
    while(!(UCSR0A & (1<<UDRE0)));
    UDR0 = data;
}

void USART_putstring(char* StringPtr)
{
    while(*StringPtr != 0x00){
        USART_send(*StringPtr);
        StringPtr++;
    }
}

void MAI_Delay(long unsigned int delaytime_ms)
{
    int z;

    for(z = 0; z < delaytime_ms; z++)

```

```

    {
    }

}

void USART_Init(unsigned int ubrr)
{
    /*Set baud rate */
    UBRR0H = (unsigned char)(ubrr>>8);
    UBRR0L = (unsigned char)ubrr;
    /*Enable receiver and transmitter */
    UCSR0B = (1<<RXEN0)|(1<<TXEN0);
    /* Set frame format: 8data, 1stop bit */
    UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);
}

unsigned char USART_Receive( void )
{
    /* Wait for data to be received */
    while ( !(UCSR0A & (1<<RXC0)) );
    /* Get and return received data from buffer */
    return UDR0;
}

void USART_Flush( void )
{
    unsigned char dummy;
    while ( UCSR0A & (1<<RXC0) ) dummy = UDR0;
    while(dummy != dummy);
}

```

Timing diagram

Major Project - Magnetic field measurement jig Project Schedule

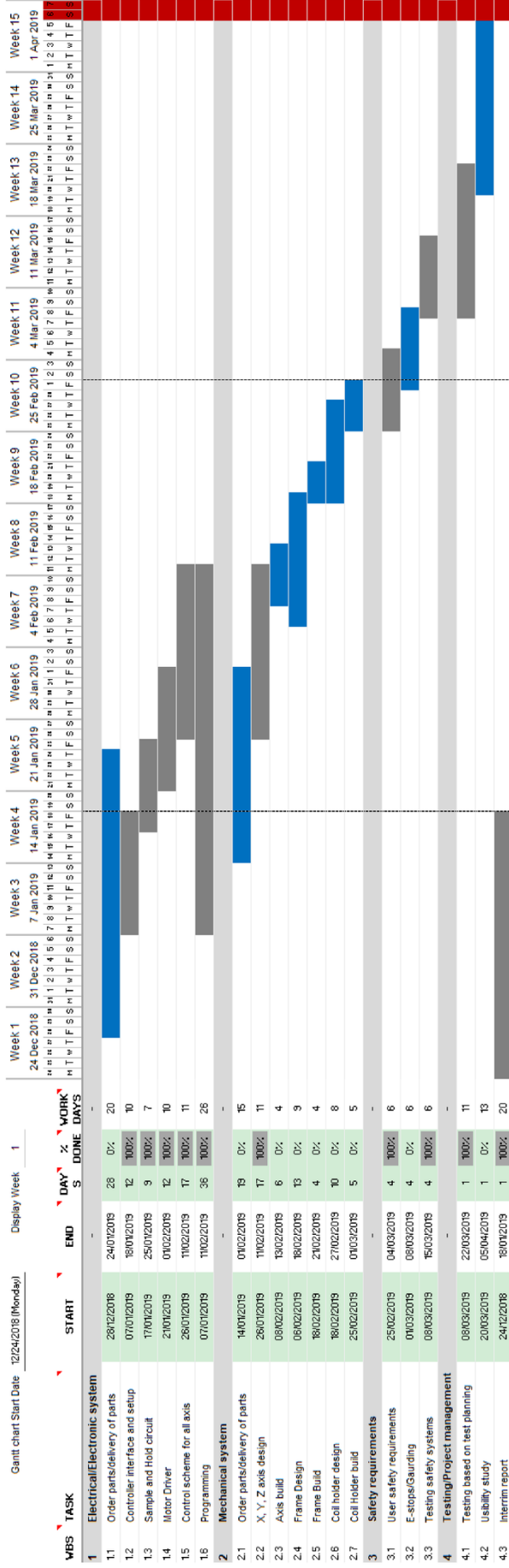


Figure 20 - Timing diagram, from assignment 1

(Lewis, 2019)

Assignment 2 SGEE6003 Major Project

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

46/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

GRADING FORM: NEW CSG-RUBRIC

MICHAEL LEWIS

46

1. ENGAGEMENT WITH LITERATURE

 There is some evidence of use of technical literature in the form of datasheets for the design components, but there was a lack of references to established testing parameters such as safe limits for magnetic exposure. There is evidence of reading from credible sources but the number of sources is limited. Citations used were mainly correct and consistently applied. (5/10)

5

2. KNOWLEDGE AND UNDERSTANDING

 The sample and hold circuit is a good simple design and the complexity of the C code and the explanation demonstrates advanced knowledge and critical understanding of programming techniques. However, not all aspects of the system design were completed and further explanations are needed. (6/10)

6

3. COGNITIVE AND INTELLECTUAL

 Testing is limited to the sample and hold circuit with good results displayed as waveforms showing the input and output of the circuit, and the sound level from the main coil. However, no testing or results were carried out for the ADC, the USART comms link and user interface or the stepper drive circuit. Due to problems in the implementation phase, some of the testing was not possible. The components

11

that were completed could have been tested more vigorously through simulation or the construction of a test bed. There was obvious disappointment about the way the project progressed, however the evaluation should have focused more on what was achieved rather than why things weren't. There is too much blame attributed to upper management. Some conclusion are presented but the recommendations are more of a personal reflection on what went wrong rather than a plan for moving forward. There should have been more on what went right and how the problems could have been solved. (11/30)

4. PRACTICAL APPLICATION



A good simple design of the sample-and-hold circuit and good innovative skills shown in C code for the stepper drive and PC comms. However, not all parts of the design were completed. Good evidence of deployment of correct methods, materials, tools and equipment in the construction of some of the project prototype. Construction was limited to the veroboard circuits with additional work on the software for the microcontroller. Good use of C programming software. Project prototype developed on veroboard is of an acceptable standard. However, there was no final product as such with the individual circuits not being connected together. Some use of test equipment evident. The project seems to have run out of control with much of what was intended not being completed; contingency measures were not evident. Some evidence of planning but the FDS should have been revised due

20

to the change in the project direction and the influences outside your control. (20/40)

5. TRANSFERABLE SKILLS



Executive summary covers some aspects of the project but does not give the outcome or success. No acknowledgements given and introduction contains general information that could be applied to any project – though a revised set of aims are presented. Some good, clear diagrams/photos, pages and figures are numbered, word count has not been exceeded. However, there are a number of spelling, punctuation and grammatical errors that could have been resolved by more careful proof reading. Some diagrams which were included in the annex are small, whereas large chunks of code were in the main report. Report didn't flow well without a block diagram. By your own admission, you have underestimated the amount of work required to complete the project. This is a useful lesson to carry forward into the future. Log book shows good evidence of managing the design process with weekly entries demonstrating the changes that occurred outside your control. (4/10)

4