NAIT
Edmonton, Alberta

AUTOMATED CHESS BOARD

As a submission to

Mr. Kelly Shepherd

Instructor, English and Communications Department,

Mr. Ross Taylor

Instructor, CNT Department

Submitted by

Reeshav Kumar

Student ID: 200388774

Computer Engineering Technology

April 19, 2018

2924 - 109st NW

Edmonton, AB T6J3E6

April 19, 2018

Mr. Kelly Shepherd

Instructor, English and Communications Department,

Mr. Ross Taylor

Instructor, CNT Department

NAIT

11762 - 106 St NW

Edmonton, AB T5G 2R1

Dear Mr. Shepherd and Mr. Tylor,

I am submitting the report Automated Chess Board, as you requested for evaluation. This report partially fulfills my obligations towards the requirements of CMPE 2960: Computer Engineering Capstone.

The report details the design and construction of a physical automated chess board that replicates the moves that were made on a Windows chess application. It goes over the details of the designing and workings of the board.

I'd like to thank all my instructors at NAIT for their dedication to work for the success of their students. Their passion for what they teach is a great motivator in the classroom

Sincerely,

Reeshav Kumar

CNT Student

Table of Contents

List of Figures

Abstract

Our project “Automated Chess Board” is a physical rendition of the virtual chess app (SharpChess) on the PC. The board maps the moves made in the game automatically. It workes by using an X-Y table moving an electomagnet. After the initial setup and calibration, once the move is made on the pc it is sent over to the MX9S12XDP512 microboard where the move is decoded by the decoder library (Decoder_Lib.c) and sent over to the Movement.c library. The NEMA 17 stepper motors and the electromagnet are controlled by this library using Pulse Width Modulation and logical outputs.

The motors are conneted to a driver IC (DRV8825) that controls them. The whole circuit is powerd by using a 12V 30A power supply. The PC and the microboard are cummunicating using a serial port that transfer data as strings.

1.0 Introduction

The purpose of this report is to walk the reader through the designing and building phase of our capstone project Automated Chess Board. This project does exactly what its name suggests. It is a game of chess controlled using the ShapChess application (Hughes, n.d.) on windows. The user plays the game on the computer which is mapped to the chess board, so every single move made in the game will be executed on the board. This is being done using a combination of X-Y table and an electromagnet. The X-Y table was the biggest hurdle that we overcame during our time working on this project. We used our microboard to process data from the desktop app and control the stepper motors in the X-Y table.

The mechanical portion of the project is a custom-built variant of an X-Y table controlling an actuator. We used stepper motors as drivers for the arms of the table which is discussed in detail in the Mechanical section of the project.

The software portion of the project is handled by the chess app and microboard. As the move is made in the game, it is sent to our microboard where the data is used to drive the motors and activate/deactivate the electromagnet whenever required. The details of this are discussed in the Software section where the integration of the chess app and the microboard is extensively explained.

We tried to integrate most of what we learnt during our program at NAIT into the project and are really pleased with the results as we achieved everything that we initially planned.

2.0 Overview

After extensive research, my capstone partner Carter Kozakevich and I decided to pick this project for our capstone. This project highlights everything we have learned during our time at NAIT and gave us a chance to learn new things on the go. Our project has three main components - Mechanical, Electrical and Software. This report explains these components in their specific sections in detail. These components work together to move the pieces on the chess board.

As the move is made in the chess app, the data is extracted from the app using serial port. This is done using a serial communication class that we built. The data extracted from the app is parsed by our Decoder library on the microboard. Once the start and the end position has been determined these are sent to the Movement library. In the movement library the time required to make the move is calculated and the required wave is sent to the driver IC of the motor. The motor which is waiting at its home position on the X-Y table moves to the start position. Once at the start position the electromagnet is activated and the chess piece is moved to its destination and the electromagnet is deactivated to let go of the piece. After that the magnet is moved back to its home position.

3.0 THE MECHANICS

The mechanics of our project is based on the workings of an X-Y table. We custom built the X-Y table ourselves from scratch and it's safe to say that this was the hardest part of our project. We used stepper motors as the driver for the arms of our table. A combination of wooden blocks with aluminium rods make the skeleton of our X-Y table and two motors are mounted on it as shown in the image below.

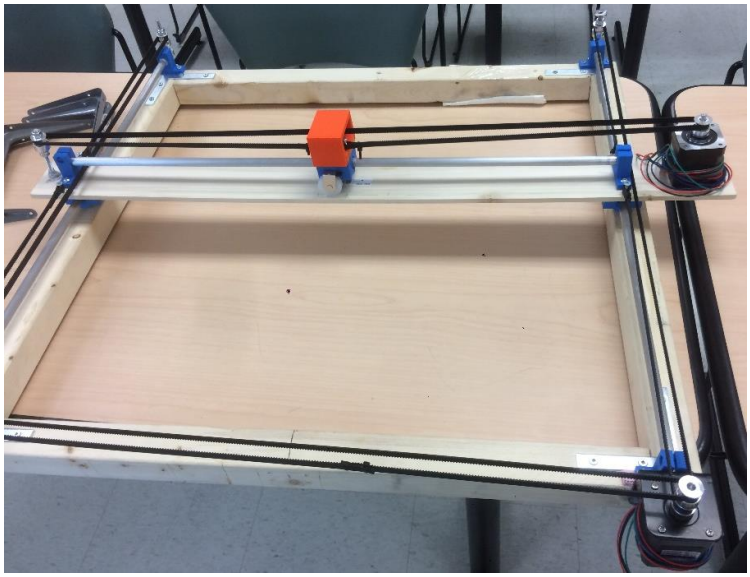

Figure 3-1: Skeleton of the X-Y table

3.1 X-Y Table

The X-Y table are built and configured to provide high-performance positioning along multiple axis. They offer precision controlled automated movement and have a lot of applications. They provide two-dimensional motion for a lot of automated machinery especially in manufacturing. These are widely used as pen plotters. (Intellidrivers, n.d.) Add another dimension to them and these act as a skeleton for a 3-d printer.

3.1.1 Design

During our research we realized that there are a few of variants to an X-Y table. The most popular type is the cross-section type in which the rails are attached to each other. The other type is the one in which the small intersection between the rails is the only moving piece. (Intellidrives, n.d.) This is the type we thought would best serve our purposes. We built our custom X-Y table after modelling it according to the skeleton of the RobotShop model (MakeBlock, 2017). The frame of the table is four 2X4 pieces of wood attached together to give us a 2' by 2' working area. The rails of the table are aluminium rods attached at the opposite ends of the wooden frame. The rods are held in place by the 3-D printed shaft supports. A wooden plank with linear bearings acts as a connecting cross-section between the rails. As soon as we finished the skeleton of the table we realized our bearings bind on the rails easily. To fix this issue we had to sand down the rods to make the movement smoother. The first motor is connected to the linear bearings that move the plank and takes care of one dimension. The second motor sits on the plank and moves that platform that holds the electromagnet - the second dimensions (see Figure 3-1). There are corner brackets in the four corners of the table to hold the chess frame which sits on top them.

3.2 Stepper Motor

A stepper motor is a brushless DC electric motor that move in discrete steps. They have multiple coils that are organized in groups called phases. By energizing each phase, the motor moves in steps. Since Stepper Motors are so precise they are used in 3D printers, CNC and X-Y plotters. (Earl, n.d.) We used a two NEMA 17 bipolar motors for our project.

3.2.1 NEMA 17 Stepper Motors

NEMA 17 is a stepper motor with a 1.7 X 1.7-inch faceplate. This motor 1.8° step angle (200 steps/revolution). Each phase draws 1.7 A at 2.8 V, allowing for a holding torque of 3.7 kg-cm (Polou Robotics and electronics, n.d.). This stepper motor is controlled using a driver IC (the details of which are explained in the electronics section of the report). The following figure is the image of a NEMA 17 stepper motor that we used for our project.

Figure 3-2 NEMA 17 Stepper Motor

4.0 THE ELECTRONICS

A bipolar stepper motor is high current consuming motor. We use a 12V 30A power supply to power up the motors. The voltage is supplied to the controller IC of the motor. Driving a stepper requires two full H-Bridges so it can reverse the current to the phases. These H-brides are hard to build from scratch, so we decided to use a driver IC DRV8825 for our purposes. Our breadboard contains all our driver ICs. The MX9S12XDP512 microboard is connected to the breadboard circuit and provides the required pulse and digital output to the driver ICs to drive the motors. The electromagnet is powered through the Field Effect transistor through the microboard. The image below is the fully connected circuit between the microboard and the driver ICs.

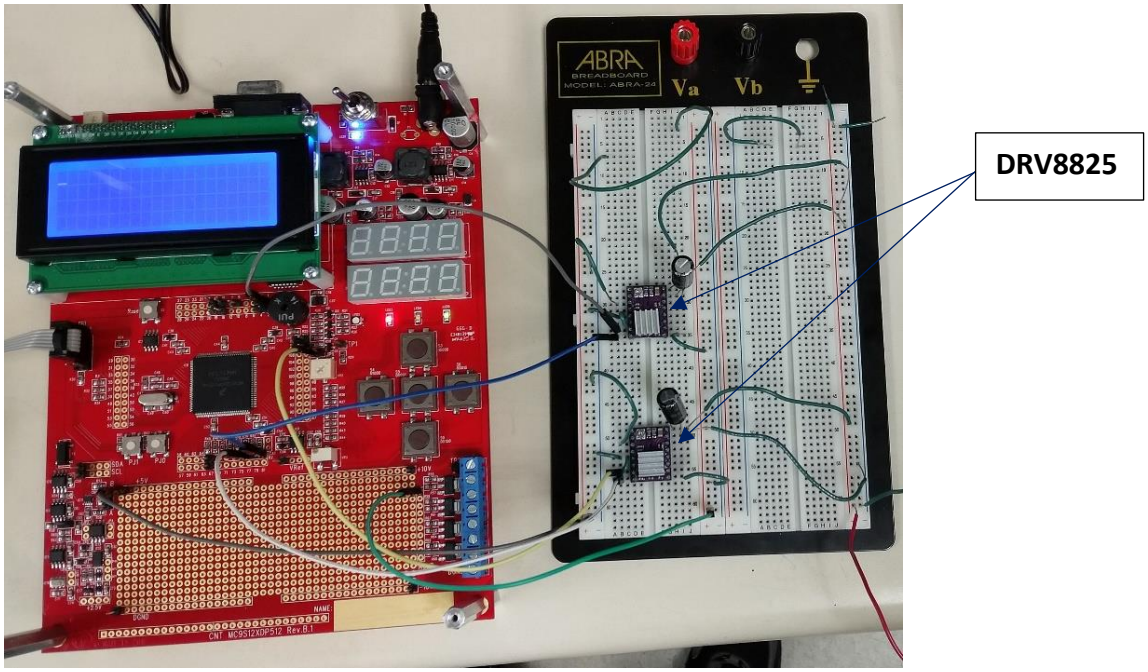

Figure 4-1: Breadboard circuit connected to the microboard

4.1 Breadboard Circuit

The circuit of our project is fairly simple. It consists of two driver ICs DRV8825 (more on these later...) which are essentially an H-Bridge connected to the two motors. The power supply is connected to these ICs. The pulse output from the microboard is also supplied to the IC.

4.1.1 DRIVER IC - DRV8825

A driver IC is an integral part of running a stepper motor. This microstepping bipolar stepper motor driver features adjustable current limiting, over-current and over-temperature protection, and six microstep resolutions (down to 1/32-step). It operates from 8.2 V to 45 V and can deliver up to approximately 1.5 A per phase. The IC has a small potentiometer on it for calibration. We calibrated our ICs to have a voltage difference of 1.7V. When the power is connected to the IC along with the motors, the whole circuit pulls a total current of 1A. supply

Fig 4-2 is the schematic diagram of the driver IC describing its circuit.

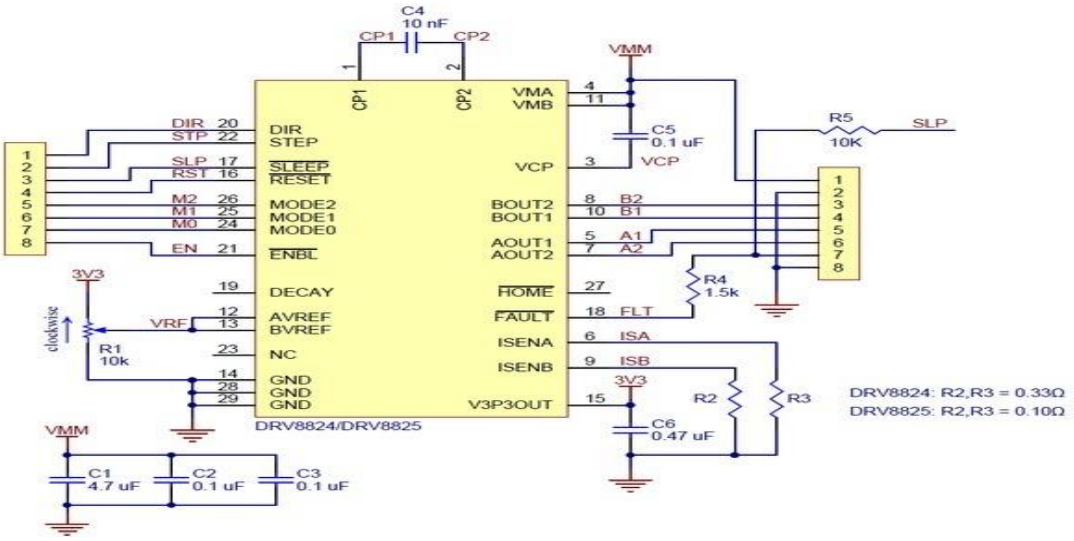

Figure 4-2 Schematic for the DRV8825 (Polou Robotics and electronics, n.d.)

Fig 4-3 is the wiring diagram for connecting the components to the IC.

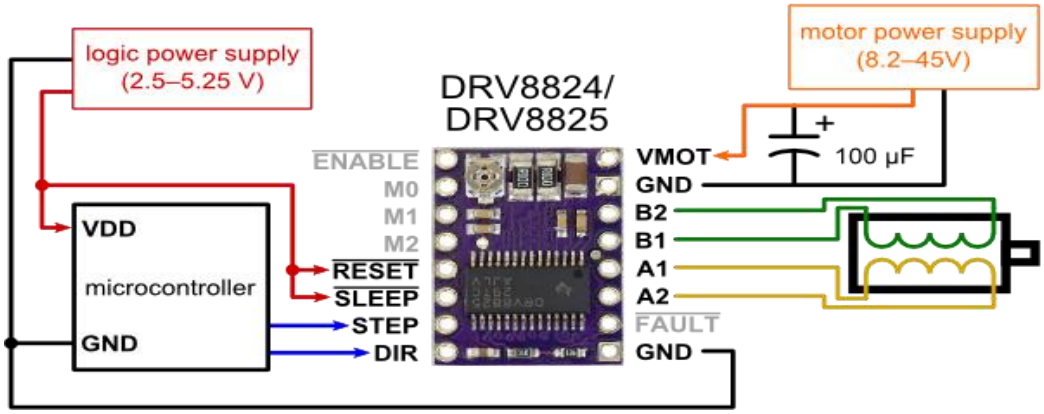

Figure 4-3 Pinouts and circuit diagram

4.2 Electromagnet

To move the pieces on the chess board we are using an electromagnet that is moved by the arms of the X-Y table. When the magnet reaches the start position it is activated and the piece is moved along the board. We use a Uxcell 20-pound electromagnet (Fig: 4-3). It is rated 12V and uses 300mA current. We supply the input voltage from the power supply to the FET. A diode is connected across the magnet to protect the microboard from kickback. The magnet and the FET are connected in series. We control the circuit programmatically to activate/deactivate the magnet.

Figure 4-4 : Uxcell Electromagnet

4.3 MX9S12XDP512 Microboard

The MX9S12XDP512 is the microboard that we are required to use throughout our program at NAIT and were very familiar with. Initially we were using an Arduino uno as our microcontroller but quickly shifted to the 9S12X because it allowed us to do a lot more than the Arduino. We use two PWM (Pulse Width Modulation) pins to serve as input waves for the stepper motor and two digital high/low switching outputs to control the direction. These outputs are supplied to the IC. To control the electromagnet, we are using the FETs available on the microboard. This is because the main board circuit is not able to handle high current, but FETs are designed to handle 10A. Since our magnets needs 200mA to work with it was safer to use the FET circuit to avoid burning up the microboard (we were told that's bad for some reason).

4.3.1 Wave Output

The stepper motor requires a wave to operate. The wave is supplied to the step pin of the IC. For our motors we are sending a 1KHz square wave from the PWM pins of the microboard. We use the speaker to get the wave for the first motor and the LCD pins for the second motor. We calculate the amount of time for which the motors should be activated and send the wave for that amount of time.

4.3.2 Logical Output

The direction of the motor is controlled by supplying a logic high or low to the DIR pin (See Figure 4-3). We use two LED pins from the micro. Turning on the LED supplies 5V that rotates the motor clockwise and turning it off rotates the motor anti-clockwise.

5.0 THE SOFTWARE

The software, as I consider it is the soul of our project. It is divided in two sections - The SharpChess App and The Chess_Capstone microboard project. These two communicate with each other. The chess app sends the move to the microboard where it is decoded by the microboard project. Once the move is decoded, it is executed using the libraries that we created for the movement.

5.1 The Chess App

We used the open source SharpChess App as our primary game. It allows the user to play against a computer or a human opponent. (Hughes, n.d.) We added some of our own code to the app to extract the move that was made in the game. Figure 5-1 depicts the UI of the game.

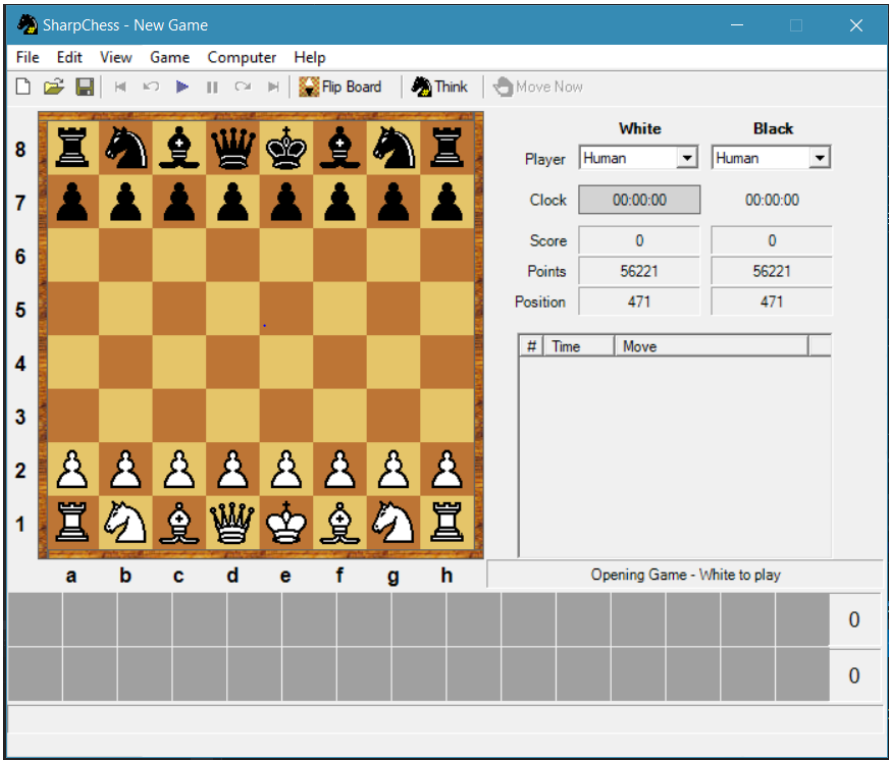

Figure 5-1: SharpChess UI

5.2 Communication

After extracting the move from the main game, we use our SerialComm class to send the data to the microboard. This is being done using a serial connection. We open the serial port (COM1) and set the baud rate to 19200. The move is sent in the following format.

“startPosition separator endPosition endCharacter”
Ex: “a5-d6y”

We send the string using the serial port write function, which takes in a string and sends it over.

5.3 Microboard Project

The microboard project named Chess_Capstone consist of the following libraries:

- **SCI0_Lb.c**

  This library initializes and controls the serial communication on the microboard. We initialize the com port at the baud rate of 19200. The string written in to the serial port by the chess app is read in by using the method from this library, character by character. After terminating the incoming string at the end character, the string library is sent over to the decoder library to extract the move to be made.

- **Decoder_Lib.c**

  The decoder library checks the string built from the serial port. We defined a structure named Point that stores two integers for the X dimension and Y dimension. Library methods checks the separator for what kind of move was made and extracts the start position and the end position. The positions are returned as Points to be sent over to the movement library where the move is executed.

- **Movement.c**

  This Library executes the actual movement of the magnet based on the points received. For a simple move from a start point to the end point, the magnet moves from home position to the start position. It picks the piece and then moves to the end position and drops the piece. The time for which the motor is supposed to move is calculated according to the following equation:

```
int distance = msHalfSquare * stepSize * howLong;
```

  In the above equation msHalfSquare is the time it takes to move half a square on the board, the stepSize is the number of steps the motor is working with and howLong is the distance between one dimension of the start point and the end point. The following lines are an example of activating the X direction motor for the calculated amount of time. We do the same with the Y direction by changing the fifth bit instead.

```
PWME |= 0b01000000;          //Turn on motor
Sleep_ms(distance); //for the calculated time
PWME &= 0b10111111;          //Turn off motor
```

  For picking the piece the electromagnet is activated by turning on bit 7 on the FET address (PTM) and turning it off for dropping the piece. The direction is controlled using the LEDs. Green LED controls the X direction and Red LED controls Y direction. This is done by setting/clearing specific bits on PT1AD1.

- **SwLeds_Lib.c**

  This library initializes the LED controls so that they can be used as required. The address PT1AD1 controls the three LEDs by setting or clearing the bit for the specific one. The

LED outputs 5V when on and 0V when off, which we supplied to the DIR pin of the driver IC.

- **LCD_Lib.c**

  This Library controls the LCD display on the microboard. We use the LCD_String method to display relevant information on the LCD during the game.

Out of the five libraries above we created the Decoder.c and Movement.c specifically for this project. The remaining three helper libraries are the ones we created as a part of our Embedded systems course in the third semester.

5.4 Special Moves

Special movements on the chess board is also taken care of by the software. This includes death move and the Castling move.

5.4.1 Death Move

When a piece is killed in the chess game it is moved to the death area and then the piece that killed it is moved to its's destination. It executed using the DeathMove method.

```
struct Point DeathMove(struct Point startPos, struct Point endPos, struct Point deathArea)
{
	//save the death area into a temporary point
	struct Point temp = deathArea;
	Move(homePos,endPos);		//move the magnet to the start point

	ToCenter('p');
	Move(endPos,deathArea); //move the piece to the death area
	//check where to drop the piece
	if(deathAlt == 0)
```

```
	{
		PTM &= 0b01111111; //drop the piece
		deathAlt = 1;		//alternate the death variable
	}
	else if(deathAlt == 1)
	{
		ToCenter('d'); // drop the piece at the center
		deathAlt = 0;  //alternate the death variable
		temp.Y++;	   //increment the Y component
	}
	//move the other piece to its destination
	Move(deathArea, startPos);
	ToCenter('p');
	Move(startPos,endPos);
	ToCenter('d');
	Move(endPos,homePos);
	//return the new death area to be saved for the next move
	return temp;
}
```

5.4.2 Castling

Castling is the only time when more than one chess piece moves in the game. It only occurs if there is no piece between the king and rook and they have not been displaced from their original position. (ZYjacklin, n.d.) In the chess app the move is displayed as O-O (kingside) or O-O-O (queenside). Since there is no relevant information about the chess pieces, we manually built the movement strings required to execute this move. We build the king move in SerialComm class and send it to the micro with a separator specifying that it was a castle move. Then it gets checked by the decoder library on the micro which build up the rook move and then both moves are executed in order.

6.0 Project Result

We finished our project with all the planned features and requirements. The X-Y table works exactly as we expected it to be. There are some inefficiencies as of now the biggest one being the speed with which the operation is executed. If we increase the speed the electromagnet, we increase the chance of missing the pieces during the movement phase. Due to the time constraints we had to ditch the idea of wireless communication, but serial communication worked way better than expected as there is next to no lag.

The SharpChess App works seamlessly with our board. We managed to utilize the X-Y table in a unique and unorthodox way for this project. All things considered the project turned out to be a success in the end.

7.0 Conclusion

This report focused on construction of the Automated Chess Board. Divided in three main section, it first went over the mechanics of the project that explained the designing of the X-Y table and details how the stepper motors were used. Next the Electronics phase details all the electrical components and the circuity used to make the project work. It details how the electromagnet and the microboard were used with the X-Y table. The Software section explains the programming required to communicate between the app and the micro. It also goes into details of the microboard programming required to move the motors and use the electromagnet. With our project we managed to utilize most of what we learned at NAIT and create a unique product.

8.0 Reflection

- Going into capstone, I was unsure about what project I wanted to pick but I am happy to say that my capstone partner and I decided to settle on “Automated Chess Board”. Completing this project was a roller coaster of a ride. We didn’t help ourselves by deciding to build the X-Y table from scratch. Since initially our knowledge about X-Y tables or stepper motors was next to none, we relied extensively on research and once we started building we found our way as we go. I can proudly say that anything that could go wrong during our design phase did went wrong. We improvised a lot to fix the issues with our project and since we were on a budget, we used a lot of components that ideally shouldn’t be a part of an X-Y table. It took a lot of reading and experimentation to make the stepper motors work. It did get frustrating at times but once it was working there was no looking back. Once things started coming together, we were at the point where we could test the movement and start finalizing. I would also like to point out that I paired up with a great partner and the project would not have turned out the way it did if it wasn’t for that.
- The thing that I enjoyed the most while working on this project was working with the stepper motor and finding out about their mechanics. I had never worked with them before and found their operation very interesting. I learned that they work with a square wave and the frequency of the wave decides the speed of the motor.
- The part of the project that was the most difficult in my opinion was building the skeleton of the X-Y table. Our rails were not smooth enough for the platform to move freely and it took us a long time before we realized that. Due to this problem the

platform would bind on movement. We did take care of this problem eventually by sanding down the rails and filing the hole in the linear bearing. Another problem that we encountered was figuring out what combination of magnet and metal pieces would work best without interfering with other pieces during the movement phase.

- The part of the project that I am the proudest of is the programming portion. We compartmentalized our program in a very effective manner. Since there was no way for us to check our program while we were writing it, so we had to write the whole thing in one go and hope for the best. I am proud of the fact that about 80% of our program worked on the first go and we didn't have to spend a lot of time to debug our program.
- Though I am really happy with the outcome of our project, there are few changes I would defiantly make if I get another chance at this project. I would like to plan a little more next time around so that I don't get caught off-guard by any issues that arise. I have a list of features that I would like to add. I would want to make the game fully independent so that the user doesn't need the computer to play the game. In this format, the game will be running directly off the microboard, so the user can play against the chess algorithm in real time.

9.0 References

Earl, B. (n.d.). *What is a Stepper Motor?* Retrieved from Adafruit: https://learn.adafruit.com/all-about-stepper-motors?view=all

Hughes, P. (n.d.). *Github*. Retrieved from Git/Sharpchess: https://github.com/PeterHughes/SharpChess

Intellidrivers. (n.d.). *XY Tables*. Retrieved from Intellidrives : Linear Actuators, Rotary Tables and XYZ Systems: http://www.intellidrives.com/Art-XY-Tables

Intellidrives. (n.d.). *XY TABLE: MEANING AND USAGE*. Retrieved from Intellidrives: Linear Actuators, Rotary Tables and XYZ Systems: http://www.intellidrives.com/Art-XY-Table-Meaning-Usage

MakeBlock. (2017). *MakeBlock XY-Plotter Robot Kit v2.0 (With Electronics)*. Retrieved from Robot Shop: https://www.robotshop.com/ca/en/makeblock-xy-plotter-robot-kit-v20-electronics.html?gclid=Cj0KCQjw7Z3VBRC-ARIsAEQifZSKv6tJYiSHKUADaGLwdF4Dx21nqLHdsnS1sGfktM27fl0uItXN67AaAnkjEALw_wcB

Polou Robotics and electronics. (n.d.). *Polou*. Retrieved from Stepper Motor: Bipolar, 200 Steps/Rev, 42×38mm, 2.8V, 1.7 A/Phase: https://www.pololu.com/product/2267

ZYjacklin. (n.d.). *Casting*. Retrieved from Chess.com: https://www.chess.com/chessopedia/view/castling