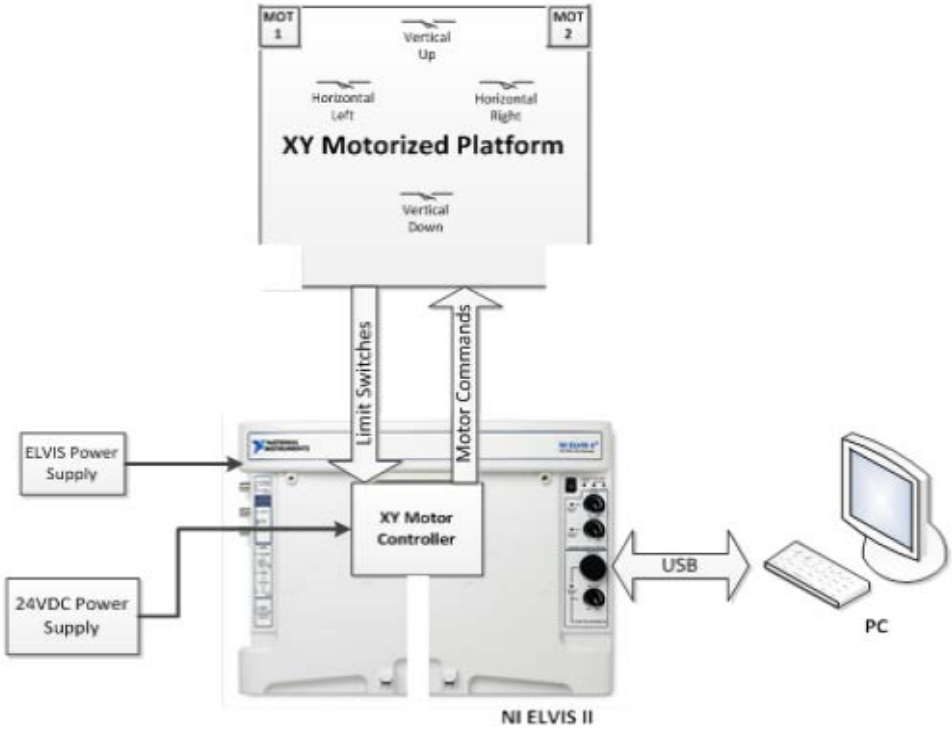
XY PLATFORM PROJECT

MECHENG 312 - Sensors and Actuators



Department of Mechanical Engineering

By Vincent Krishna (ID: 9653710)

12th May 2017

Table of Contents

1.0 Introduction 1
2.0 Position Control 2
2.1 Position Tracking Using Encoders 2
2.2 Manual Position Control (Open-Loop) 2
2.3 Cursor Graph and Coordinate Control (Closed-Loop) 2
2.4 Path Display 3
3.0 Speed Control 4
4.0 Switch Protection 4
5.0 Counter Reading 4
5.1 Pulse Count 4
5.2 Pulse Width 5
6.0 Drawing Implementation 6
6.1 Drawing a Rectangle 6
6.2 Drawing a Circle 6
7.0 Interface Design 7
Results and Conclusions 9

1.0 Introduction

The Slide Rail Mechanism contains a two dimensional motorised platform consisting of two DC motors with encoders, four limit switches, and a belt and pulley mechanism, driven by the PC through the NI ELVIS II platform. These components describe how the slide rail mechanism interact with the real-time software, LabVIEW.

The NI ELVIS II connects the controller board to the PC, providing an interface for real time programming with LabVIEW. It sends two analogue signals from the PC to the controller in order to set the desired speed of the two DC motors. The angular position of the motors can be monitored as the NI ELVIS II system can read and count the encoder outputs of each motor to send to the PC. The NI ELVIS II system can also detect when the limit switches are activated on the motorised platform and send these signals to the PC so that the real-time software application can stop the motors.

The motorised platform moves the pen in a two dimensional Cartesian plane by using two DC motors. The controller which is programmed via LabVIEW is what provides the voltages. The motion of the pen relies on the voltage applied to each motor which sets the speed and direction that it moves along the platform. Higher positive voltages increase the rotational speed of the motors in the anticlockwise direction while lower voltages decrease the rotational speed. For negative voltages, the same idea applies except the motors rotate in the clockwise direction. If no voltage is applied to the motors, no rotation will occur. This can be summarised by the table shown in Appendix A.1.

The displacement of the motors is governed by the direction in which the motors rotate, as these dictate how the two belt drives operate on the platform to move the pen from point to point. From observing the Mechanical Platform Layout (see Appendix A.2), the movement of the pen were represented by equations of motion which are shown in section 2.4.

On the motorised platform, there are four strategically placed limit switches which are used to prevent the pen from reaching the edges of the platform. This prevents damage to the platform itself, the DC motors and the belt drives. When a limit switch is pressed, the motors come to a complete stop so that the pen can no longer move in the direction it was coming from before activating the limit switch, stopping the pen from colliding with the edge of the platform.

2.0 Position Control

The purpose of position control was to be able to move the pen from one point of the platform to another. There were two methods of position control implemented in LabVIEW: Manual Position Control and Cursor Graph and Coordinate Control. Before these methods could be implemented, the measuring capabilities of the motor encoders need to be utilised to determine how far the pen has travelled in both the x and y directions from a specified point.

2.1 Position Tracking Using Encoders

The motor encoders are sensors that track the position of the motor shafts as they rotate, therefore providing closed-loop feedback signals. In order to determine the coordinates of the pen's location on the platform as it moves from one point to another, the number of pulses generated by the encoders need to be converted to distance (in millimetres). From the **Motors POLOLU-2288 datasheet** in A.3, the gear ratio of each motor gearbox was 172:1. This indicated that the shaft of each motor rotated at 172 revolutions per revolution of the belt drive sprocket. The datasheet specified that counting a single edge of one channel would result in 12 pulses per revolution of the motor shaft. Thus the number of pulses generated by the motor shaft was:

No. of pulses = 172 x 12 pulses
No. of pulses per output shaft = 2064 pulses

The distance the belt drive moved per revolution was determined by finding the circumference of the sprocket. The measured diameter was approximately 17mm, therefore:

Circumference = 17 x π
Circumference = 53.41 mm (4 sf)

Therefore, the distance measured by the encoder per pulse is:

Distance measured = Circumference / No. of pulses
Distance measured = 53.41 / 2064
Distance measured = 0.0259 mm / pulse

With the encoders measuring the distance of the pen in both the x and y directions as it moves from one point to another, the feedback enabled.

2.2 Manual Position Control (Open-Loop)

Manual position control allows the user to freely move to a particular point on the platform using four interactive switches: Up, Down, Left and Right. Each switch gives a Boolean output value of either 0 or 1 when off or on respectively. These Boolean values are stored in a build array block which gets converted into a binary number. This binary number dictates which case in the case structure will execute depending on the desired action. For example, if the left switch was pressed and held, the binary number for this case would be 10000 (when no limit switch is activated). The case structure executes the case that contains this binary number, which sets the voltage of both motors to be negative while maintaining the magnitude of its speed set by the speed controller. This causes the pen to move to the left on the platform. Similar implementations were applied for the other three switches, where each case sets the direction of the voltage of each motor.

2.3 Cursor Graph and Coordinate Control (Closed-Loop)

The cursor graph uses an interactive cursor on a set of Cartesian axes to move the pen to any desired point on the platform. As the cursor is continuously moved around the graph, the pen will simultaneously follow the exact same path the cursor is moving towards. Before this can be

implemented, the XY Platform needed to be calibrated to provide a reference point for the pen as it moves from one point to another. A switch was used to execute this task. The approximate dimensions of the platform was measured to be 260 mm in length and 170 mm in width. When the calibration switch is on, the pen moves to the top right corner of the platform until the right and top limit switches are activated. Once this occurs, the pen moves downward by a distance of 85 mm and then left by a distance of 130 mm, positioning the pen in the middle of the platform. The down and left directions were chosen for calibration because this caused less motor drift compared to the other two directions. This position, the origin, was used as the reference point.

The cursor graph was implemented by first considering the error between the desired point on the graph and the actual point, represented by the cursor and the pen's location, using encoder values on the platform respectively. Therefore, it was a closed loop system as shown in figure 1. In order to reduce this error, a proportional controller was used to multiply the error by a gain K and so output a voltage signal. A larger proportional gain will reduce the error quicker, however it will increase overshoot of the trajectory. If the gain is small enough, the overshoot would decrease but the steady state error will increase. A suitable range found was between 0.2 and 1.1 as the main equation was:
((Desired X point - Actual X point) +/- (Desired Y point - Actual Y point)) x K = Voltage
The max voltage was limited at |6|V majority of the time.
This also means that the controller was proportional only for the range between -6V and 6V or whatever max value the user decided to set. It was also noted that motor speed was 0 at 1V.

Through continuous experimentation, it was found that the most suitable proportional gain to minimise the error was 0.25. This was low enough to slow the pen down when difference was less and high enough to have an ideal settling time with large differences. If the user wanted to move to a specific point on the platform, using the cursor to reach the exact coordinates would be difficult. To address this, an interactive coordinate control was implemented, allowing the user to input a specific x and y coordinate for the cursor, and hence the pen, to move to. This setup made use of the equations of motions shown in figure

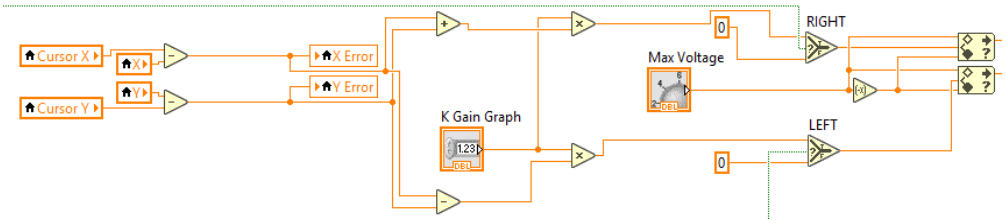


Figure 1 Proportional controller

2.4 Path Display

$\Delta X = (\Delta A + \Delta B) / 2$
$\Delta Y = (\Delta A - \Delta B) / 2$

Where ΔA is the displacement of the belt indicated in pink and ΔB is the displacement of the belt indicated in cyan in figure

The path or trajectory of the pen's position is displayed on a graph using stored x and y coordinates in an array. These x and y coordinates are ΔX from 0 and ΔY from 0. If the number of points stored exceed a user specified amount, the previous points are deleted. All points can also be cleared with a button called "Clear Graph". The x and y coordinates or displacement from origin were also used to provide feedback in the proportional controller as shown in figure 1.

3.0 Speed Control

The speed of the pen was variable in each movement case structure implemented. In manual positioning, the output voltage magnitude was decided by a control that utilized user input. These are fed into the motors with a negate block depending on which direction it is moving or wired directly. Refer to A.1 in the appendix for correct magnitudes and signs. With manual speed control, the magnitude and sign of the voltage are both controlled by the user to determine trajectory.

Using a graph/cursor control, the maximum speed was determined by a coerce value block. The gain determined the acceleration and was what determined the voltage and therefore the speed of motors. These were proportionate to the error or difference between desired point and actual point based on encoders. This was also implemented as the speed control when drawing the rectangle and the circle. However, the circle also consisted of frequency of the sine wave as a method of controlling speed. Care was taken when changing speed as large speeds resulted in overshoots and at times settling errors.

4.0 Switch Protection

As described earlier, the limit switches on the motorised platform were used to prevent damage to the platform itself, by completely stopping the motors once activated. Similar to the switches used for manual position control, each limit switch gives a boolean output value when off or on, which gets stored in the build array block. For example, if the left limit switch is activated, the binary output of the build array will be 1. This executes the case that contains this binary number in the case structure, which sets the voltage of both motors to zero to prevent further movement in the left direction. The pen can still move in all other directions while the left limit switch is active, provided that another limit switch isn't simultaneously active. In addition, all other cases that contain 1 at the end of the binary number can only be executed when the left limit switch is on.

If two limit switches are activated simultaneously, then the binary outputs for both limit switches are added together. For example, if the left and top limit switches are activated, then the binary output of the left limit switch, 1, is added to the binary output of the top limit switch, 1000. This executes the cases containing the binary number 1001 in the case structure, preventing moving in both the left and upward directions but still allowing movement in the downward and right directions. All other cases that contain 1001 at the end of the binary can only be executed when the left and top switches are on simultaneously.
This implementation therefore protects the platform from damage while handling cases adequately when limit switches are reached.

5.0 Counter Reading

The optical encoders were wired such that only one channel could be read. The rising edges of the channel was utilized to determine period. A DAQ Assistant VI was used to read absolute encoder pulses. This works within the while loop but a drawback is having to remake channels when moving to a new board. The left encoder was on pin 3 and the right encoder on pin 8 as mentioned in

5.1 Pulse Count

The relative pulse count was achieved by first calibrating the platform and resetting values to 0. Upon rotation of the encoders, a value is stored in an indicator variable and the difference found from previous stored value. The refresh time for the number of pulses is 0.1 seconds. The difference is then either added or subtracted from a stored value of relative encoder pulse movement depending on direction of rotation of motor. A conditional statement was implemented to check if applied voltages were negative or positive.

Absolute pulses after refreshing - Absolute pulses before refreshing = Change in pulse count
Stored pulse count +/- Change in pulse count = Relative pulse count

The relative pulse count for both motors is set to 0 at the centre of the board after calibration. The conversion factor for pulses/mm which was calculated in 2.1 is multiplied to find linear distance travelled by belt depending on rotation of motor.

Relative pulse count x 0.0259 = linear distance travelled by motor

As mentioned in section 2.5 of the report, the displacement in X and Y dimensions were then calculated from these distances and used as feedback for our position controllers.

5.2 Pulse Width

The frequency of the pulses required finding the period. Knowing the refresh period, the calculations were done to find the frequency.

Pulses/ 0.105 seconds = Pulses/second = Frequency
A square wave was generated with 50% duty cycle, the above frequency and then graphed.
Using frequency of the pulses, linear speed and RPM of motors and therefore, the X and Y dimension speeds of the pen can also be calculated.
Linear speed = frequency x 0.0259
RPM = (Linear speed / circumference) x 60 seconds
X-dimension speed = (Left linear speed + Right linear speed)/2
Y-dimension speed = (Left linear speed - Right linear speed)/2

6.0 Drawing Implementation

The drawings were implemented with the reuse of the proportional controller code and both were closed loop systems with set points generated and actual points derived from encoder positions.

6.1 Drawing a Rectangle

The binary value for the case structure for drawing a rectangle was 100000000. It was made using a nested case structure within this case structure. Each nested case corresponded to which side was being drawn. The position of the top left corner of the rectangle is determined when using cursor or coordinate position control. The lengths and widths are decided by the user and acts as a condition to when the pen starts drawing sequential sides of the rectangle.
Each nested case structure has another nested case structure enabling on or off states for switches to go to another side after the side being drawn is done.

6.2 Drawing a Circle

The binary value for the case structure for drawing a circle was 1000000000. Two sine generator VIs were used to generate set points for the x and y positions for the circle. The sine generator for y position had an offset by 90 degrees (cosine) so the points would plot a circle. The equations for the set points for the circle were:
X desired point = (-Radius x sin(Speed x time) + X origin
Y desired point = (Radius x cos(Speed x time) + Y origin
These were implemented as shown in figure:

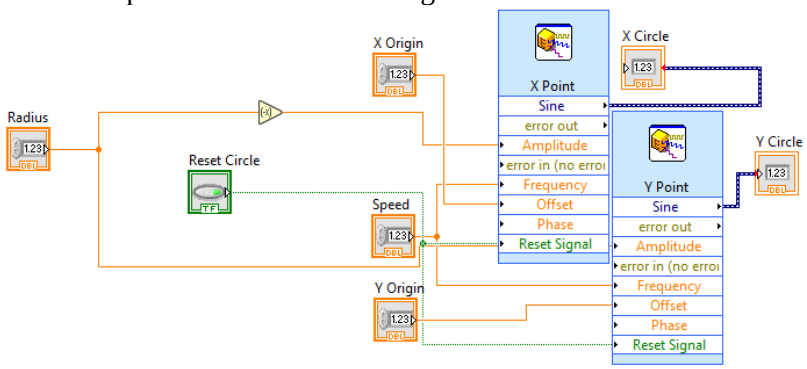


Figure 2 Sine Function Generators

The closed loop system was implemented via the magnitude and direction of motor speed being determined by the difference or error being multiplied by gain. Refer to proportional controller equation in section 2.3 for the general setup. The set points are the outputs shown in figure 2. The gain was higher than that of position control but as before, the voltage was limited to avoid overshoots and the speed changed to provide a smoother circumference.

7.0 Interface Design

Tab Control Containers have been used throughout the interface design as shown in figures 3 and 4, laid out strategically in rows to allow easy access and interaction for the user to the features of the program.

The first row contains the most important features of the program to allow easy access and clear visibility of these features, specifically the limit switches and logistics buttons. The user must be aware of when the pen reaches the edges of the platform and how to stop the entire program in case of emergency. The killswitch (labelled STOP) is coloured red to clearly distinguish itself from the rest of the user interface. The motor on/off button is also essential to stop the motors from moving, but also allow debugging while the program is still running.

The tab control containers in the second row displays the response of the motorised platform as the pen is moving. The user can do this by navigating from one tab to another to witness, for example, the encoding readings, the frequency of the sensors, or the speed of the motors.

The third row, shown in figure 2, consists of components in the tab control containers that the user can interact with and adjust depending on the user's preferences (specifically, the tab control container on the right). The user can select its choice of position control, adjust the speed of the motors, and set the desired dimensions of the rectangle and circle to be drawn. These adjustments can be witnessed using the Cradle Position tab control container, which has a two dimensional graph displaying the motion of the pen as it is being moved on the platform.

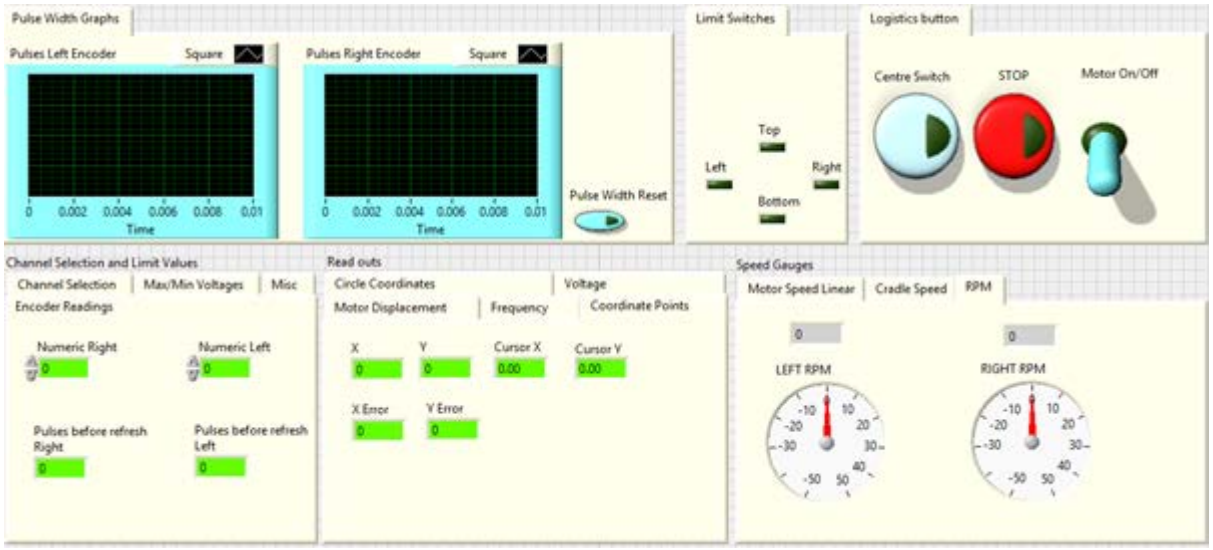


Figure 3 GUI 1

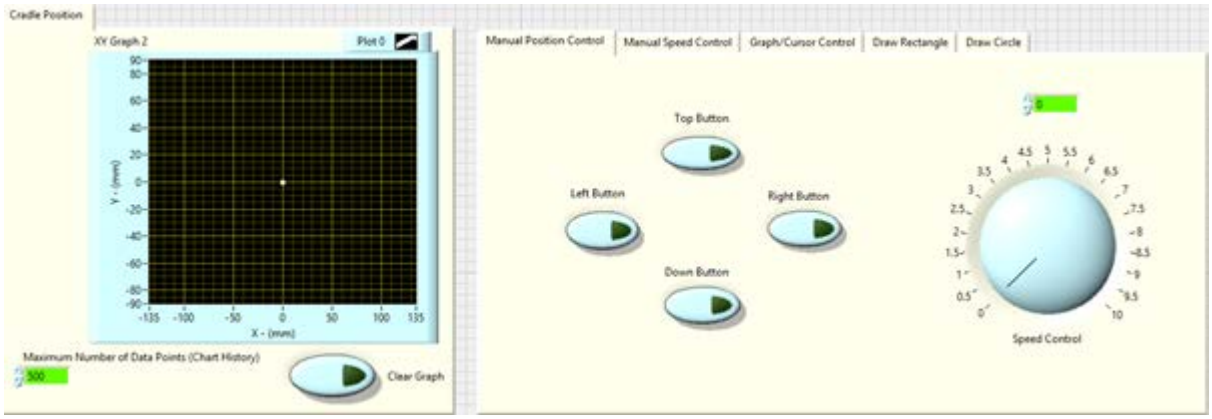


Figure 4 GUI 2

Results and Conclusions

The proportional controller was largely successful allowing omni-directional movement and so making a precise circle as evident from figure 5. Closed loop allowed more accurate movement of the pen and enabled feedback display such as that mentioned in section 2.5. Using a sine function generator was a successful method in controlling both speed, smoothness, radius and origin.

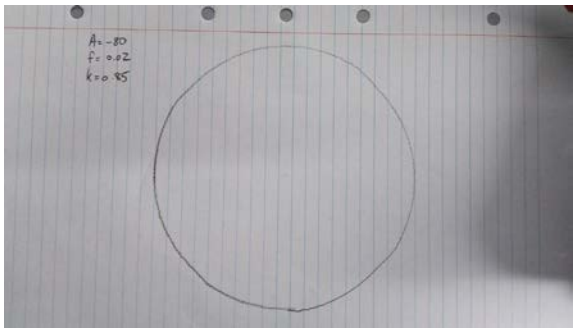


Figure 5
Circle

The rectangle was also successful with the implementation of nested case structures. A closed loop system was again used for this shape with a proportional gain up until a certain voltage limit. The rectangle was largely successful as well.

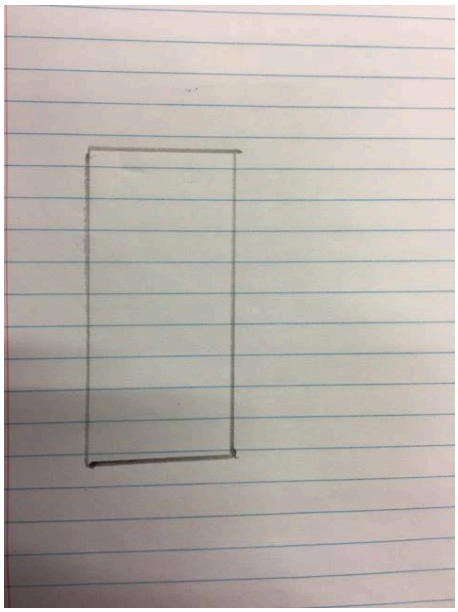


Figure 6
Rectangle

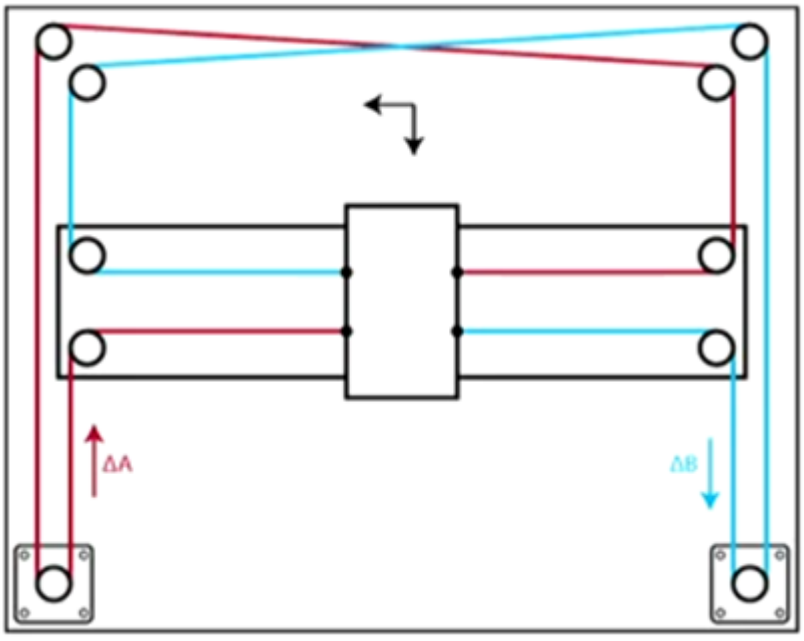
In a definite conclusion, a proportional controller was accurate and had minimum errors. Position control using proportional controller and closed loops to draw shapes were the most precise and efficient method. Given more time, a PID controller might have shown more promise but a P controller with the used gains were successful.

Appendix A.1 - Mechanical Platform Layout

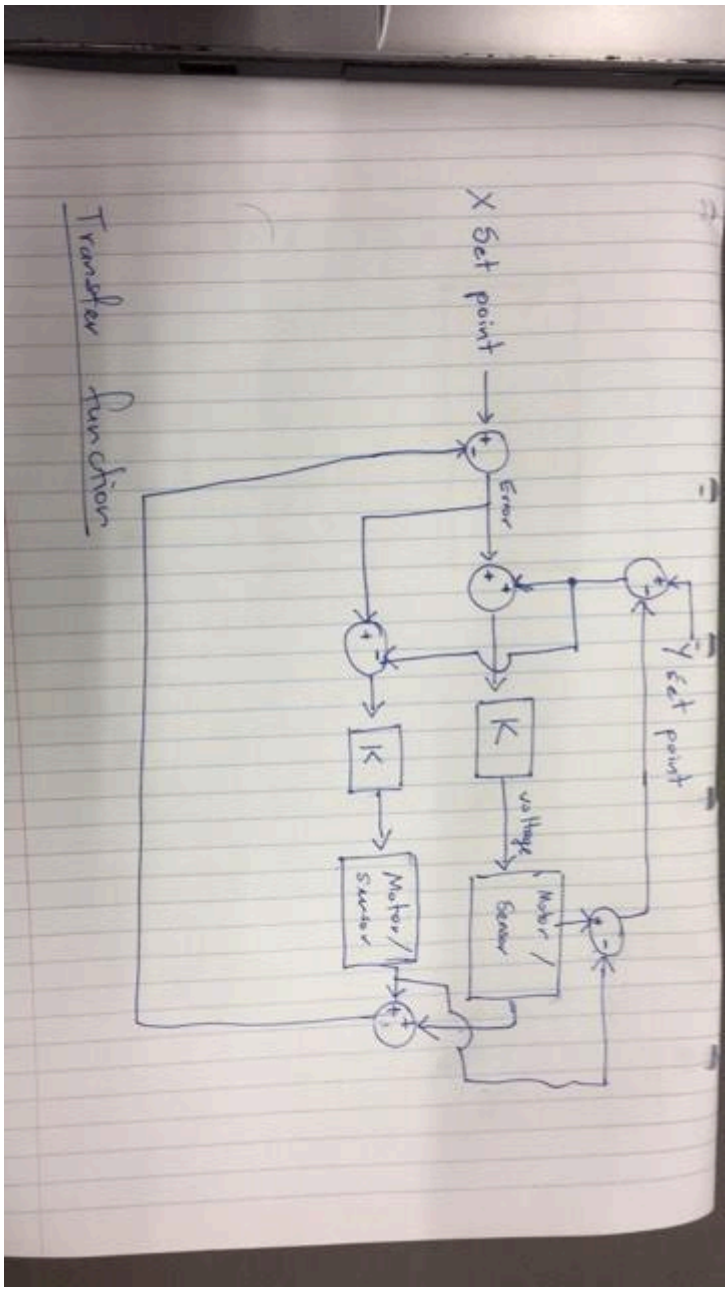
Direction of Pen	Mot-LX	Mot-RX
H-R	+	+
H-L	-	-
V-U	-	+
V-D	+	-
Dg-RU	0	+
Dg-RD	+	0
Dg-LU	-	0
Dg-LD	0	-

Appendix A.2 - Mechanical Platform Layout

Pink and Blue lines represent the two belt-and-pulley mechanisms.



Appendix B


Transfer function
X Set point
Error
K
K
Motor / Sensor
Motor / Sensor
Y set point