RMIT University

School of Electrical and Computer Engineering

Trajectory Tracking Control of Ball and Plate System

This project thesis is submitted in partial fulfilment of the requirements for the course of

EEET 2268 (Engineering Design 4B)

Student Name: John Lee

Student Number: (s3273319)

Submission Date: 17/10/2013

Supervisor: Prof. Liuping Wang

Acknowledgement

I would like to acknowledge and thank Professor Liuping Wang for her guidance, mentoring, and support throughout the year. Professor Wang is very generous with her time and her assistance with the project was very much appreciated.

Executive Summary

A cascaded I-PD control structure with reference pre-compensation is used to track various trajectories on the ball and plate system. The I-PD controller was determined through pole assignment. Analysis of the closed loop response of the system to sinusoidal reference inputs was conducted against a non-linear model to determine the magnitude and phase response of the closed loop system. Reference pre-compensation was applied to negate the magnitude and phase response. The resulting system tracks sinusoidal inputs with no steady state error. This departs from traditional control methods in which the frequency of the reference signal is embedded into the controller structure. The control scheme was implemented on a ball and plate system prototype. The trajectory tracking scheme was successfully applied to a circular path, a figure 8 path, a square path and a maze path.

Table of Contents

Table of Figures

Table 1 - List of notations and abbreviations

Symbol	Description	Unit
$\boldsymbol{m}$	Mass of the ball	kg
$\boldsymbol{R}$	Radius of the ball	m
$\boldsymbol{x, y}$	Position of the ball on the x-axis and y-axis	m
$\boldsymbol{\dot{x}, \dot{y}}$	Velocity of the ball on the x-axis and y-axis	m/s
$\boldsymbol{\ddot{x}, \ddot{y}}$	Acceleration of the ball on the x-axis and y-axis	m/s^2
$\boldsymbol{\theta_x, \theta_y}$	Angle of the plate from the x-axis and y-axis	$rads$
$\boldsymbol{\tau_x, \tau_y}$	Torque applied to the plate in the x-axis and y axis	$kg \times m^2/s^2$
$\boldsymbol{J_P}$	Mass moment of inertia of the plate	$kg \times m^2$
$\boldsymbol{J_b}$	Mass moment of inertia of the ball	$kg \times m^2$
$\boldsymbol{\theta_S}$	Angle of the servo control arm	°
$\boldsymbol{kc}$	Proportional gain	
$\boldsymbol{\tau_I}$	Integral time constant	
$\boldsymbol{\tau_D}$	Derivative time constant	
$\boldsymbol{PID}$	Proportional plus integral plus derivative controller	

Chapter 1 - Introduction

The ball and plate balancing system poses a challenging design problem. The system is an example of a nonlinear under-actuated system seen in many real world cases including underwater vehicles, space vehicles, balancing robots, vertical take-off and landing aircraft, exothermic chemical reactions and power generation[1, 2]. It is used to study dynamic systems and as a tool to develop, implement and test various control schemes[3]. Its complex and nonlinear nature makes it a suitable control target to test many modern control methods.

The ball and plate balancing system consists of a ball balanced on top of a rigid plate. The position of the ball is manipulated by changing the inclination of the plate about its x and y axes. This project has three main objectives. The first objective is to develop a ball and plate balancing system prototype. The second objective is to develop a control scheme which is capable of stabilising the ball at a specific position on the plate and to reject any position disturbances to the ball. The final objective is to develop a trajectory planning and control scheme which allows the ball to follow set trajectories.

This report presents a design for a ball and plate system prototype. The prototype uses a resistive touchscreen for ball position feedback and digital servos for plate angle positioning. A cascaded I-PD control structure is presented as a solution to the ball stabilisation and trajectory tracking problem. A reference pre-compensator is proposed in order to allow the I-PD controller to track sinusoidal inputs.

1.1 Literature Review

The ball and plate system is a two dimensional electromechanical system in which a ball is balanced on top of a rigid plate. The balls position is able to be manipulated by altering the inclination of the plate about its x and y axis as shown in Figure 1[4-12].

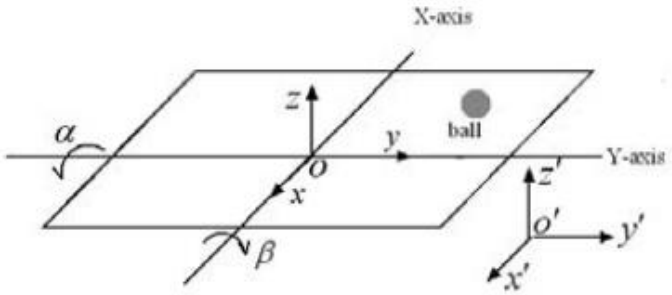

Figure 1 - Ball and Plate System[4]

The system has four degrees of freedom. Two degrees of freedom are due to the inclination of the plate in the x-axis and y-axis. The inclination of the plate is actuated by the use of either a stepper motor or a DC motor. The other two degrees of freedom are the position of the ball in the x-axis and y-axis. This position is a function of the angle of the plate. As the system has four degrees of freedom and only two actuators it is referred to as an under actuated system.

1.1.1 Ball and Plate System Model

Many control methods rely on the availability of an accurate model of the system they are trying to control. Several sources have derived a mathematical model of the ball and plate systems dynamics using the Euler-Lagrange differential equations[13, 14]. The Lagrange equations enable a relatively easy derivation of equations of motion of complex dynamic systems by analysing the kinetic energy in a system[15]. In [4] the mathematical model of the ball and plate system is shown as:

$$\left(m + \frac{J_b}{R^2}\right)\ddot{x} - mx\dot{\theta}_x^2 - my\dot{\theta}_x\dot{\theta}_y = -mgsin\theta_x \quad (1)$$

$$\left(m + \frac{J_b}{R^2}\right)\ddot{y} - my\dot{\theta}_y^2 - mx\dot{\theta}_x\dot{\theta}_y = -mgsin\theta_y \quad (2)$$

$$\tau_x = (J_P + J_b + mx^2)\ddot{\theta_x} + 2mx\dot{x}\dot{\theta}_x + mxy\ddot{\theta}_y + m\dot{x}y\dot{\theta}_y + mx\dot{y}\dot{\theta}_y + mgxcos\theta_x \quad (3)$$

$$\tau_y = (J_P + J_b + my^2)\ddot{\theta_y} + 2my\dot{y}\dot{\theta}_y + mxy\ddot{\theta}_x + m\dot{x}y\dot{\theta_x} + mx\dot{y}\dot{\theta}_x + mgycos\theta_y \quad (4)$$

Equations (1) and (2) describe the movement of the ball on the plate and shows that the acceleration of the ball is dependent on the plate angle and angular velocity. Equations (3) and (4) describe the torque forces on the external plate positioning drive due to the load moment and position of the ball on the plate.

1.1.2 Physical Design

The general physical design of previous works is of two distinct types. In the most common design a plate is placed upon a central pivot point [13, 16-18]. The central pivot consists of either a ball joint or universal joint. The angle of the plate is actuated by two sets of mechanical linkages attached to the sides of the plate by another set of ball or universal joints. The second design consists of two rotating frames, with one placed inside the other as shown in Figure 2. The angle of each frame is actuated by a motor attached to a shaft at each pivot point [5, 19].

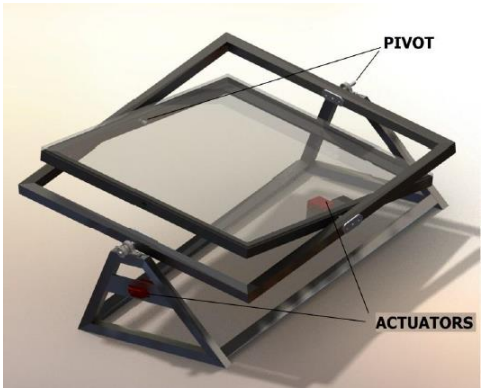

Figure 2 - Rotating frame design[20]

1.1.3 Sensors

A requirement to designing a ball and plate balancing system is a method to determine the position of the ball. Each successful implementation has used either a resistive touch screen sensor or a video camera sensor

1.1.3.1 Resistive Touchscreen

Several ball and plate system implementations have used a resistive touch screen to detect the position of the ball [21-24]. Resistive touch screen devices are used in many applications including Personal Data Assistants, industrial instrumentation, and medical devices. Resistive touch screens consist of a glass or acrylic panel that is coated with electrically conductive and resistive layers made with indium tin oxide and a flexible outer membrane[25]. Figure 3 shows the general arrangement of a resistive touch screen. The screen consists of three layers: a glass sheet, a conductive coating on top of the glass sheet, and another conductive layer on top. An external voltage is applied to electrodes spread out across the glass causing a uniform electric field. When pressure is exerted on the top conductive layer, it comes into contact with the conductive coating on the glass. This causes a voltage which is proportional to the distance to the edge of the coating. This voltage is measured at several points and filtered in order to obtain the coordinates of the touch [25, 26].

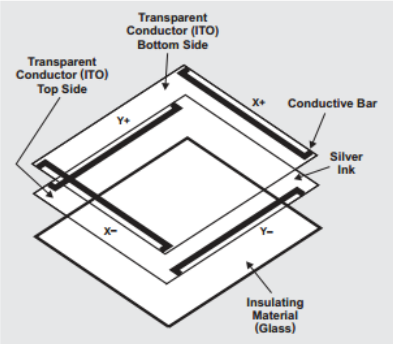

Figure 3 - Resistive Touch Screen[25]

The two most common types of resistive touch screen are the four-wire resistive touch screen and the five-wire resistive touch screen. A four-wire resistive touch screen uses both the upper and lower layers to determine the x and y coordinates[26]. The main drawback with using a four-wire configuration is that one coordinate axis uses the flexible membrane as the surface over which the electric voltage spreads out. Over time the membrane develops microscopic cracks which change the electrical characteristics of the sensor. In a five-wire design the voltage is applied to the four corners of the glass back layer. The fifth wire is attached to the flexible membrane as the voltage probe. Microscopic cracks can still occur but these do not introduce nonlinearities into the measurements. This means that the five-wire touch screen is a much more durable design than the four-wire touch screen[26].

1.1.3.2 Video Camera

The most common sensor used in previous ball and plate system designs has been a CCD camera with a frame grabber and computer vision software [10, 13, 17, 18, 27]. This method requires that each frame is processed and scanned to determine the location of the ball[27].

Several steps are involved in the process. The initial step is to separate the image of the ball from the background. In each implementation a highly contrasting plate colour was chosen to assist in this process. The image was converted to binary data and compared with a threshold table to separate the ball from the background. The centre of the ball was then found using various algorithms. The main disadvantage of this method is that it is susceptible to interference from foreign objects entering the frame.

1.1.4 Actuators

The solution to the actuation problem is dependent on the physical design of the system. In the rotating frame design the actuator is attached to a shaft connected directly to the pivot point. In [5] a closed loop servo controller was used to drive two stepper motors attached to the pivot points of the frame. In [19] a DC motor was used to drive the frames inclination. The central pivot point design requires additional mechanical linkages for the actuation problem. Most implementations have used closed loop DC motor or stepper motor positioning with a spatial linkage mechanism as shown in Figure 4. In [18] a pulley system was used with a servo motor and in [12, 28] an oscillating pneumatic cylinder was used.

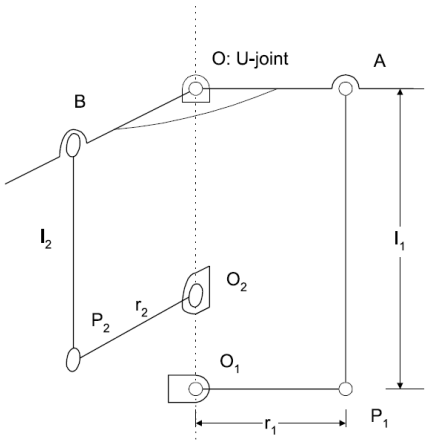

Figure 4 - Spatial Linkage Mechanism[16]

1.1.5 Control Structure

The majority of successful ball and plate balancing systems have implemented some variation of a double loop cascaded structure as shown in Figure 5[5, 10, 16, 24, 29, 30]. The outer control loop controls the position of the ball and the inner loop controls the position of the plate. The outer loop controller calculates the plate angle required in order to position the ball according balls reference position. The inner loop controller generates the required control signal to drive the plate to the correct position using whatever actuation method the design uses[19]. The inner loop has much faster dynamics than the outer loop meaning that it can reject disturbances to the position of the plate quickly before the disturbance can affect the outer loop[31]. This greatly improves the performance over a single loop structure and can simplify the outer loop controller design. If the outer loop dynamics are sufficiently slower than the inner loop then the inner loop dynamics can be ignored in the design while still ensuring outer loop stability[31, 32]. While the majority of ball and plate systems use this

structure, the chosen inner and outer loop controllers vary between each design. The outer loop controller design is the most important as it determines the overall performance of the system.

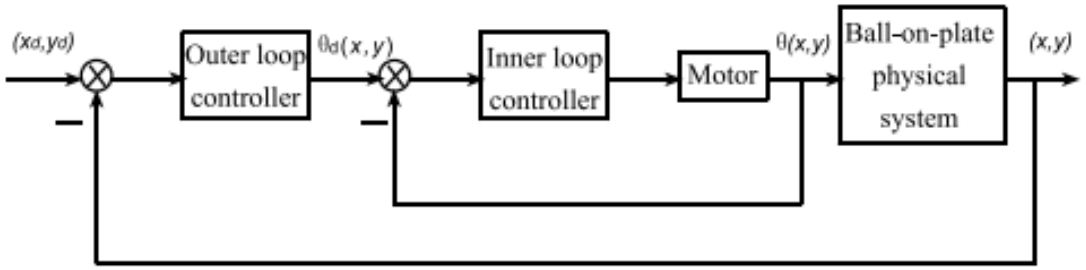

Figure 5 – Double Loop Cascaded Control Structure [19]

1.1.6 Control Scheme

Due to the complex nonlinear nature of the ball and plate system there have been many approaches to the overall control scheme. In [16] a double loop control structure was used for both stabilisation and trajectory control problems. The inner loop was a PID controller and the outer loop used a lead compensator. In [10] a switching control scheme was proposed. At each sampling instant the system determined which of the two controllers to use based on the error of the balls position from the reference position. When the error was large a PID with neural network tuning scheme was selected. The neural network is used to adjust the weighting of a feed-forward network. When the error was small a standard PID controller was used. It was claimed that the use of two controllers reduced the settling time in the presence of a large error while also reducing oscillations when the error was small. In [5] a single input rule module (SIRM) fuzzy logic controller was used in the outer loop. The SIRM fuzzy controller was used to generate the plate angle control signal based on the positional and velocity error of the ball. Trajectory tracking control was achieved for a circular trajectory. In [3] a three level hierarchal fuzzy controller was used. At the lowest level a Takagi-Sugeno fuzzy tracking controller was used to generate the plate angle control signal. A Mamdani type fuzzy controller was used to plan the desired trajectory and a supervisory Mamdani fuzzy controller was used to safeguard the system in extreme situations. The scheme was validated by simulation against a nonlinear system model.

1.1.7 Controller Design

The controller design of previous works is of three main types: fuzzy logic controllers, lead compensators and PID controllers. The essence of fuzzy control is to build a model of human expert who is capable of controlling the plant without thinking in terms of mathematical model[33]. The input variables are mapped into membership functions in a fuzzy set. A decision is then made based on a series of linguistic rules to determine the fuzzy result. The specific controller output value is determined by defuzzification of the output fuzzy set. Mamdani and Takagi-Sugeno methods are the most common methods for developing fuzzy models[34, 35]. Lead compensation involves introducing a pole zero pair into the open loop system which changes the locations of the closed loop poles and zeros. The introduced pole zero pair is chosen such that the closed loop performance of the system matches the desired transient and steady state response. The compensator parameters can be identified through root

locus or bode plot design techniques [36, 37] . PID controllers are the most widely used controllers in industry due to their simplicity, good control performance and excellent robustness to uncertainties[31]. The proportional part of the controller is a gain applied to the control error which is proportional to the size of the error. The integral action of the controller is applied to the integral of the control error and the derivative action is proportional to the derivative of the error. The presence of the integral action ensures that the controller can reduce the steady-state error to zero when a step reference signal is applied to the system[38, 39]. One disadvantage of using a PID controller is that the integrator action can increase without bounds in the presence of saturation of the control variable. This is known as integrator windup and is a problem that must be solved in order to use this type of controller[40].

1.2 Design Specifications

The aim of the project was to design and build a ball and plate system prototype and to implement controllers to satisfy the following specifications:

a. Ball Stabilisation – Given a reference position of (0,0) the system shall stabilise the ball at the centre of the plate. The position error shall be < 5mm and it shall take < 3 seconds of settling time.

b. Disturbance Rejection – With the ball stabilised at the centre of the plate the system shall reject disturbances to the position of the ball. The disturbance will be a pulse disturbance provided by perturbing the ball with a finger. The system shall return the ball back to its position at the centre of the plate in < 5 seconds.

c. Tracking step changes to the reference position – Given a step change to the reference position in either the x-axis or y-axis the position of the ball should track the reference position. For a step change of 7.5cm the settling time shall be < 5 seconds with < 1% overshoot.

d. Square Tracking – As an extension to tracking step changes the system shall provide the capability to follow the square trajectory shown in Figure 6 – Ball trajectory printout. The ball shall follow the trajectory with < 10mm distance between the ball and the plotted trajectory and shall not overshoot the corners of the square.

e. Maze tracking – As a further extension to tracking step changes the system shall provide the capability to track the maze path shown in Figure 6.

f. Sinusoidal Referencing Tracking – Given a sinusoidal reference position the system shall track the sinusoidal input with no tracking error.

g. Circle tracking – As an extension to sinusoidal tracking the system shall provide the capability to follow the circular trajectory shown in Figure 6. The circle has a radius of 7.5cm and the ball shall complete 1 revolution of the circle every 2.5 seconds.

h. Figure 8 – As a further extension to sinusoidal tracking the system shall provide the capability to follow a figure-eight trajectory on the plate.

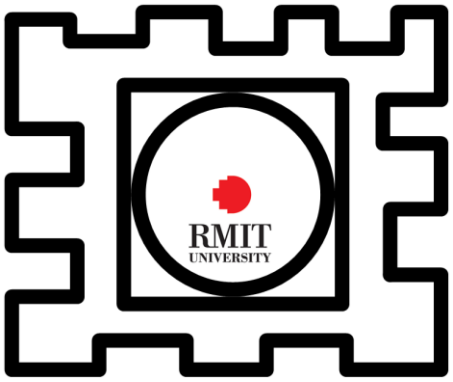

Figure 6 – Ball trajectory printout

1.3 Summary of Contributions

In this section the main contributions of the thesis are described. In chapter 2 the design for a ball and plate system prototype is presented. The prototype uses a five-wire resistive touchscreen for ball position feedback and two digital servos to regulate the angle of the plate. The prototype is based on a central pivot design and uses three brass and plastic universal joints at each of the three pivot points. The controller is implemented on a BeagleBoard XM Linux board in C code. Access to the touchscreen coordinate data is through the linux event driver. The servos are controlled through an external servo driver board that interfaces with the BeagleBoard XM through a serial interface. A user interface allows the ball and plate system prototype to be switched between various trajectory modes. The UI system uses a Nintendo wiimote controller as the input device and uses auditory output for the output device.

In chapter 4 a double loop cascaded I-PD control structure was proposed as a solution to the ball and plate stabilisation and trajectory problem. The inner loop contains the servo drive and the outer loop calculates the desired plate angle to achieve the given control objective. An I-PD controller was proposed for both loops however the inner loop was actually implemented using an external servo controller. The I-PD controller was shown to be able to quickly track a step position reference change with negligible overshoot. The outer loop controller design was extended to include a reference pre-compensator to allow the system to track a sinusoidal reference with the same I-PD controller. The compensator values were determined by analysing the magnitude and phase response of the closed loop ball and plate system when subjected to an uncompensated sinusoidal input. The sinusoidal reference was compensated by applying a gain which was the inverse of the closed loop magnitude response, and by applying a phase shift which was of equal but opposite magnitude to the closed loop phase response. The resulting pre-compensated controller was able to track a sinusoidal reference without steady state error. Pre-compensation values were calculated for a range of input frequency by simulating the closed loop controller against a nonlinear model. The designed continuous time I-PD controller was discretised by the backwards difference method. It was

shown that this method was superior to the bilinear transform in this application due to the ability to maintain the sample period as a variable in the controller equation. This simplifies the microprocessor implementation as it makes the discrete controller tolerable to variations in the sample period. This is advantageous in a Linux operating environment where some variations to the sample period are to be expected.

1.4 Report Outline

The outline of this thesis is as follows. Chapter 1 provides an introduction to the control problem and some background into previous. The specifications of the final product design are also presented. In chapter 2, the physical design of the ball and plate system prototype is discussed. In chapter 3 a detailed design of the ball and plate system controller is presented. The designed controllers are simulated to determine the appropriateness for achieving the stated specifications of the project. Chapter 6 presents the software configuration and implementation of the ball and plate system prototype. In chapter 7 the experiment results are presented. Chapter 8 identifies the conclusions of the project.

Chapter 2 - Ball and Plate System Prototype Design

2.1 Introduction

The ball and plate balancing system prototype shown in Figure 7 is based on a central pivot design. The main components of the system are the Beagleboard XM development board, two Hitec servos, three Huco universal joints, a perspex touchscreen plate holder and a 17 inch resistive touchscreen. This chapter describes the important components and physical specifications of the prototype.

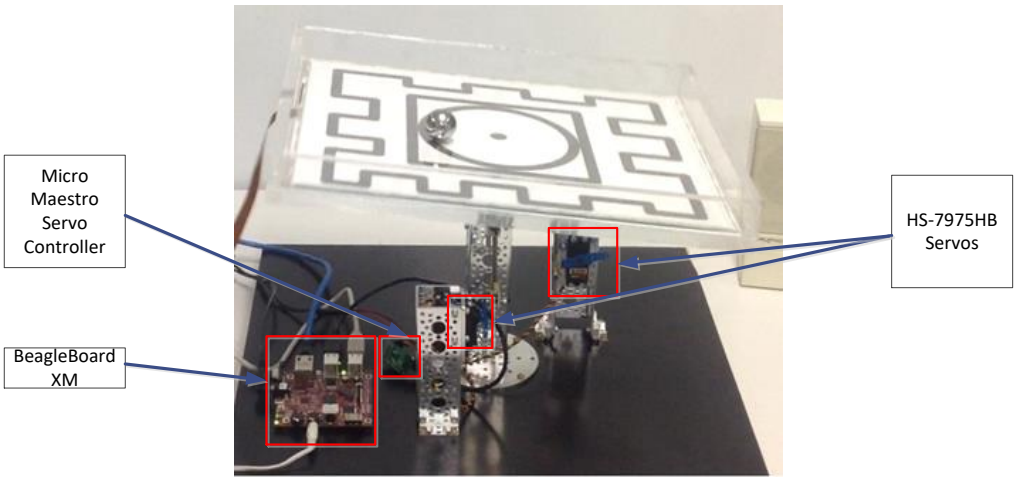

Figure 7 – Ball and plate balancing system prototype

2.2 Prototype Components

2.2.1 Perspex touchscreen holder

The Perspex touchscreen holder was required to house the resistive touchscreen and to attach the universal joints to interface with the servo linkage. The holder was fabricated from 10mm Perspex which provides a solid mounting point for the universal joints and is able to withstand severe impacts from the steel ball. The Perspex holder design, shown in Figure 8 was sent out to a local plastic fabricator to be constructed. A clamping hub for a 3/16 shaft was screwed to the base after drilling and countersinking the Perspex. The clamping hub was attached to a 3/16 shaft connected to the universal joint. The design included a rectangular cut-out at the side for the touchscreen ribbon cable to exit the holder. The ball trajectory printout in Figure 6 was inserted between the touchscreen and the Perspex. The overall weight of the touch screen holder was 2.5kg.

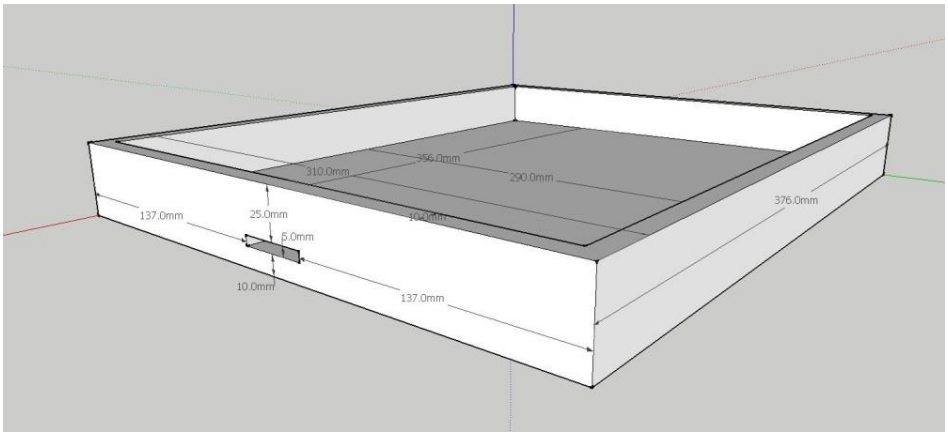

Figure 8 - Perspex touchscreen holder

2.2.2 Resistive touchscreen

A five-wire resistive touch screen shown in Figure 9 was chosen as the ball position sensor. It was identified that the viable options for the ball position sensor were either a resistive touch screen or a CCD video camera. The advantage of using a resistive touch screen over a camera is that it is easier to implement in code, requires much less computing power and does not require a bulky stand to hold it in place. The main disadvantages of using the touch screen are the purchase cost and the durability. The touch screen chosen was a 5 wire 17 inch 4:3 touchscreen. Using the supplied USB touch screen controller the measurement resolution was found to be less than 1mm and the time between measurements for each axis were approximately 6ms. Better results could have been obtained with a custom touch screen controller; however the results were adequate for this project.

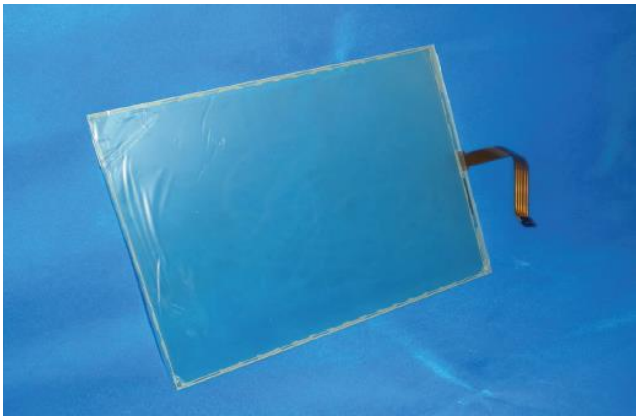

Figure 9 - Resistive Touch Screen

2.2.3 Digital Servos

Two Hitec HS-7975HB digital servos shown in Figure 10 were used to control the angle of the plate in both the x-axis and y-axis. Each servo has a maximum torque rating of 10.4kgcm and has a speed of 0.1secs/60° at 6 Volts[41]. The maximum current draw of the two servos during operation was measured at 1.2 amps which meant that that they were able to be powered from a 6V 2A plug power pack. The torque rating was found to be adequate only after the speed and acceleration was limited during implementation. It was found that if the servos were used

without limitation then the torque rating could be exceeded when the direction of angle change of the plate was quickly reversed as is expected to occur during the circle mode of operation. Initial testing at maximum servo acceleration tracking a sinusoidal input with $w_0 = \frac{2\pi}{1.5}\ rads/s$ caused the servo to fail. The failure was caused by the Karbonite gearing teeth shearing when the rotation of the servo shaft quickly changed direction. Following this failure the sinusoidal tracking target was revised down to $w0 = \frac{2\pi}{2.5}\ rads/s$ and the maximum acceleration of the servo was limited. Digital servos are controlled using a 50Hz pulse with the duty cycle proportional to the desired angle of the servo. These pulses were generated using a Pololu Micro Maestro Servo Controller which is discussed later in this chapter.

Figure 10 - Hitec 7975HB Digital Servo[41]

2.2.4 Servo Linkage

The servo linkage components are used to translate the rotary movement of the servo shaft to up and down movement and to allow lateral movement at the plate connection due to the angle change of the opposing axis. Most of the components used are components typically used in model aeroplane construction. An aluminium servo arm connects to the servo shaft. This attaches to a 10-32 threaded pushrod by a 10-32 clevis. The pushrod is inserted into a 3/16-3/16 universal joint and secured using grub screws. The most difficult component to select was the universal joint. Universal joints of this size are rare and only one supplier was able to be found. Initially a Huco plastic yoke 3/16-3/16 universal joint was used. The main issue with universal joints is that they are typically used for rotary movement with angular misalignment between the two shafts. In this project the main use of the universal joint is for the up and down movement of the shaft. The plastic yoke universal joint was not rated for any end loading and the result was that the plastic yoke pins sheared after about one day of use. The final design uses Huco brass yoke 3/16-3/16 universal joints which are rated to 38N of end loading. The central pivot also uses this universal joint and typically it supports the majority of the 2.5kg weight of the plate. This translates to $2.5kg \times 9.8ms^{-2} = 24.5N$ of end loading which is well within the rated 38N value. The joints work from 0 to 30° which is sufficient angle for the prototype.

2.2.5 Beagleboard XM

The Beagleboard XM development board (Figure 11) was used to implement the controller and to interface with the resistive touchscreen and servos. The board has a 1GHz Arm Cortex A8 processor with 512MB of RAM[42]. It was flashed with a Ubuntu 13.04 (Raring) image. The board has four USB 2.0 ports which were used to connect with the Pololu Micro Maestro Servo

controller (Figure 12), the resistive touchscreen and a Bluetooth dongle to connect to the Nintendo Wiimote. The wiimote was used as the human interface device to change the mode of the ball and plate system demonstrator.

Figure 11 - Beagleboard XM development board

2.2.6 Pololu Micro Maestro Servo Controller

The Pololu Micro Maestro Servo Controller shown in Figure 12 was used to control the two servos. It connects to the BeagleBoard XM by a USB cable and uses a virtual serial port to communicate. The servo power cable from the 6V plug power pack is connected directly to this controller which then provides power to the servos. It provides the ability to limit the rate of change of servo angle as well as the acceleration of the servo. These options were used to implement the soft start function of the demonstrator and are discussed further in the implementation chapter.

Figure 12 - Pololu Micro Maestro Servo controller

Chapter 3 - Model of the Ball and Plate Balancing System

3.1 Introduction

This paper proposes an I-PD cascaded control structure to achieve the given control objectives. In order to find the control parameters a linear model of both the plate angle positioning system and the balls dynamics is required. The dynamics of the ball are nonlinear so in order to obtain a linear model the nonlinear model of the dynamics must be linearised. This chapter introduces the nonlinear model of the balls dynamics, the linearised model of the balls dynamics and the linear model of the plate angle actuator used for the prototype development.

3.2 Nonlinear model of the ball's dynamics

Several papers have derived the mathematical equation of the ball and plate system dynamics based on Lagrange equations[2-5, 10, 16]. The equations assume that the ball remains in contact with the plate at all times and that there is no slippage between the ball and plate. The nonlinear system dynamic equations are shown below,

$$\left(m + \frac{J_b}{R^2}\right)\ddot{x} - mx\dot{\theta}_x^2 - my\dot{\theta}_x\dot{\theta}_y = -mgsin\theta_x \tag{5}$$

$$\left(m + \frac{J_b}{R^2}\right)\ddot{y} - my\dot{\theta}_y^2 - mx\dot{\theta}_x\dot{\theta}_y = -mgsin\theta_y \tag{6}$$

Due to the use of a touch screen as the balls position sensor it was required to use a heavy ball. The moment of inertia for a solid sphere is[43]:

$$J_b = \frac{2}{5}mR^2 \tag{7}$$

Subbing Equation 3 into Equation 1 & 2 Yields:

$$\left(\frac{7m}{5}\right)\ddot{x} - mx\dot{\theta}_x^2 - my\dot{\theta}_x\dot{\theta}_y = -mgsin\theta_x \tag{8}$$

$$\left(\frac{7m}{5}\right)\ddot{y} - my\dot{\theta}_y^2 - mx\dot{\theta}_x\dot{\theta}_y = -mgsin\theta_y \tag{9}$$

These equations describe the balls dynamics in relation to the forces due to the angle of the plate with gravity, friction and centripetal forces due to the plate changing angle. Simulation of the control design was conducted in Simulink. In order to determine the performance of the design a nonlinear model was built into Simulink using the nonlinear equations shown in Eqn. (8) and Eqn. (9). The Simulink model is shown below in Figure 13.

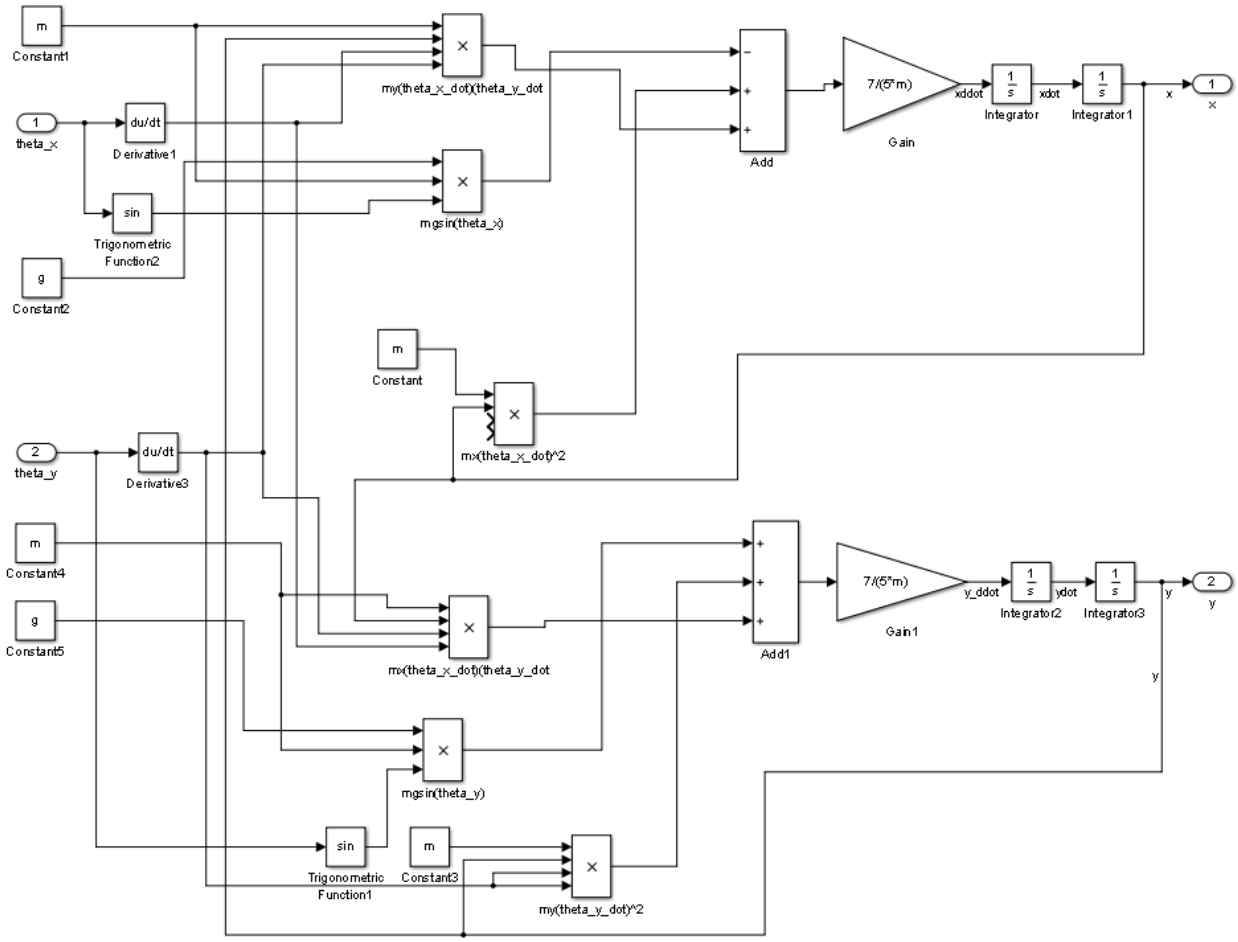

Figure 13 - Nonlinear Simulink Model

3.3 Ball and Plate System Linearized Model

To linearize the equations an approximation of the nonlinear components of the equations was taken using the first two terms of a Taylor Series Expansion. The approximation was taken around the equilibrium or operating point of the system. This operating point is defined as being when the ball is stable at the centre of the plate, the angle of the plate is zero in both the x and y axis and the angle of the plate is not changing. i.e.

$$x^0 = \dot{x}^0 = \ddot{x}^0 = 0 \; and \;\; \theta_x^0 = \dot{\theta}_x^0 = \ddot{\theta}_x^0 = 0$$
$$and \; y^0 = \dot{y}^0 = \ddot{y}^0 = 0 \; and \;\; \theta_y^0 = \dot{\theta}_y^0 = \ddot{\theta}_y^0 = 0$$

The Taylors Series Approximation of Equation 4 is shown below. The nonlinear parts of the equation are highlighted:

$$\left(\frac{7m}{5}\right)\ddot{x} - mx\dot{\theta}_x^2 - my\dot{\theta}_x\dot{\theta}_y = -mgsin\theta_x$$

$$x\dot{\theta}_x^2 \approx x^0\left(\dot{\theta}_x^0\right)^2 + \frac{\partial x\dot{\theta}_x^2}{\partial x}\Big|_{x=x^0,\dot{\theta}_x=\dot{\theta}_x^0}(x - x^0) + \frac{\partial x\dot{\theta}_x^2}{\partial \dot{\theta}}\Big|_{x=x^0,\dot{\theta}=\dot{\theta}^0}\left(\dot{\theta} - \dot{\theta}^0\right) = 0$$

$$y\dot{\theta}_x\dot{\theta}_y \approx y^0\theta_x^0\dot{\theta}_y^0 + \left.\frac{\partial y\dot{\theta}_x\dot{\theta}_y}{\partial y}\right|_{y=y^0,\dot{\theta}_x=\theta_x^0,\dot{\theta}_y=\dot{\theta}_y^0}(y-y_0) + \left.\frac{\partial y\dot{\theta}_x\dot{\theta}_y}{\partial \dot{\theta}_x}\right|_{y=y^0,\dot{\theta}_x=\theta_x^0,\dot{\theta}_y=\dot{\theta}_y^0}\left(\dot{\theta}_x-\theta_x^0\right) + \left.\frac{\partial y\dot{\theta}_x\dot{\theta}_y}{\partial \dot{\theta}_y}\right|_{y=y^0,\dot{\theta}_x=\theta_x^0,\dot{\theta}_y=\dot{\theta}_y^0}\left(\dot{\theta}_y-\theta_y^0\right) = 0$$

$$sin\theta_x \approx sin\theta_x^0 + \left.\frac{dsin\theta_x}{\partial\theta_x}\right|_{\theta_x=\theta_x^0}(\theta_x-\theta^0) = \cos(0)*(\theta_x-0) = \theta_x$$

Therefore, the linearised differential equation for the ball and plate system dynamics in the x-axis is:

$$\ddot{\mathrm{x}} = \frac{-mg\theta_x}{\frac{7}{5}m} \tag{10}$$

The Laplace transform model was found by taking the Laplace transform of the linearised differential equation model:

$$L\left\{\ddot{x} = \frac{-mg\theta_x}{\frac{7}{5}m}\right\} \xrightarrow{yields} s^2X(s) = -\frac{5}{7}g\theta_x(s) \tag{11}$$

Rearranging Eqn. 11 gives the Laplace domain transfer function of the ball and plate system for the x-axis as:

$$\therefore G_{\mathrm{x}}(s) = \frac{X(s)}{\theta_x(s)} = \frac{-\frac{5}{7}g}{s^2} \tag{12}$$

And due to equation symmetry the y-axis transfer function is:

$$G_{\mathrm{y}}(s) = \frac{Y(s)}{\theta_{y(s)}} = \frac{-\frac{5}{7}g}{s^2} \tag{13}$$

The linearised equations show that the linearized model in the x and y plane are equal and are now decoupled from each other. This means that two identical controllers can be used to control each of these systems. The controller design from now on will only be shown for the x-axis and the y-axis will be identical.

3.4 Servo Model

The ball and plate system prototype uses two Hitec servos to manipulate the angle of the plate. These servos use an internal controller in order to drive the output angle of the servo shaft. It is not possible to obtain a transfer function for the servos however a transfer function is still required in order to simulate the overall system. This was achieved by modelling a similar sized DC motor and using a PID controller to mimic the performance of the servo. A lab DC motors transfer function was approximated by doing a step response test of the motor which had a tacho generator connect to a data acquisition card of a windows xPC target machine. From this a first order approximation of the motor was found as:

$$G_m(s) = \frac{37.5}{s + 12.5} \tag{14}$$

Chapter 4 - Controller Design and Simulation

4.1 Introduction

The controller design process was undertaken in order to satisfy the given control objectives. These were:

a. Disturbance Rejection – Balance the ball at the centre of the plate and account for disturbances to the position of the ball.
b. Circle tracking – The ball is able to follow a circular path on the plate.
c. Figure 8 – The ball is able to follow a figure 8 path on the plate.
d. Square Tracking – The ball is able to follow a square path on the plate.
e. Maze tracking – The ball is able to follow a predetermined random path on the plate.

The controller for each of these control objectives was designed using pole assignment design method. This method uses a linearised model of the system to find the Proportional, Integral and Derivative controller parameters based on a designated performance specification. The control design process was conducted using the following steps:

a. Choose the controller structure
b. Use the pole assignment method to find the control parameters based on the linearised model of the plant.
c. Simulate the controller against the nonlinear model of the plant.

This chapter describes each of these design steps in detail.

4.2 Cascaded Control Structure

A cascaded control structure was chosen to achieve each control objective. The structure was identical for each objective; however there are differences in the implementation for each case. The structure shown in Figure 14 has two separate control loops for each axis. The inner control loop is for the servo positioning system. In simulation this was achieved using an I-PD controller and its performance was intended to mimic the performance of the built in controller contained within the Hitec HS-7975HB servos used in the prototype. The outer loop calculates the required plate angle in order to achieve the desired position of the ball. The desired position of the ball is indicated in the figure by “X*” and “Y*” with “X*” being the desired position of the ball in the x-axis and “Y*” being the angle of the ball in the y-axis.

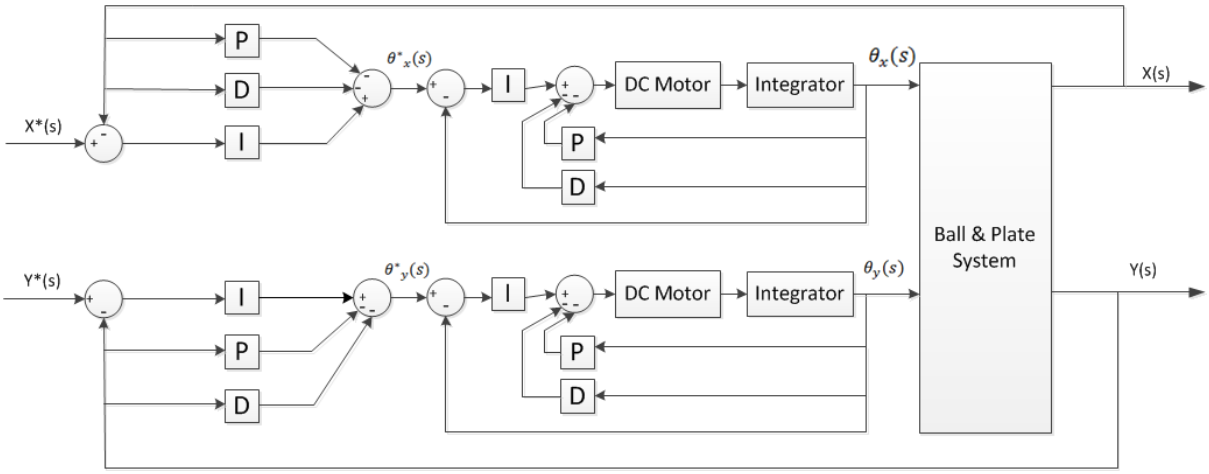

Figure 14 - Cascaded I-PD Control Structure

4.3 Inner Loop Controller Design

The inner loop controller was designed in order to mimic the performance of the built in controller contained within the Hitec Hs-7975HB servo. The performance specifications of the servo states that it has a speed of $0.1secs/60°$. Figure 15 shows the configuration of the inner loop controller. The controller is highlighted in red. Typically a PID controller is implemented on the signal $R(s) - Y(s)$, which is the reference signal minus the feedback signal. In this design the proportional and derivative component of the PID controller is implemented on the feedback only. This ensures that there is no overshoot as there is no proportional control on the reference signal θ^*. A derivative filter l_0 was used to avoid amplification of measurement noise.

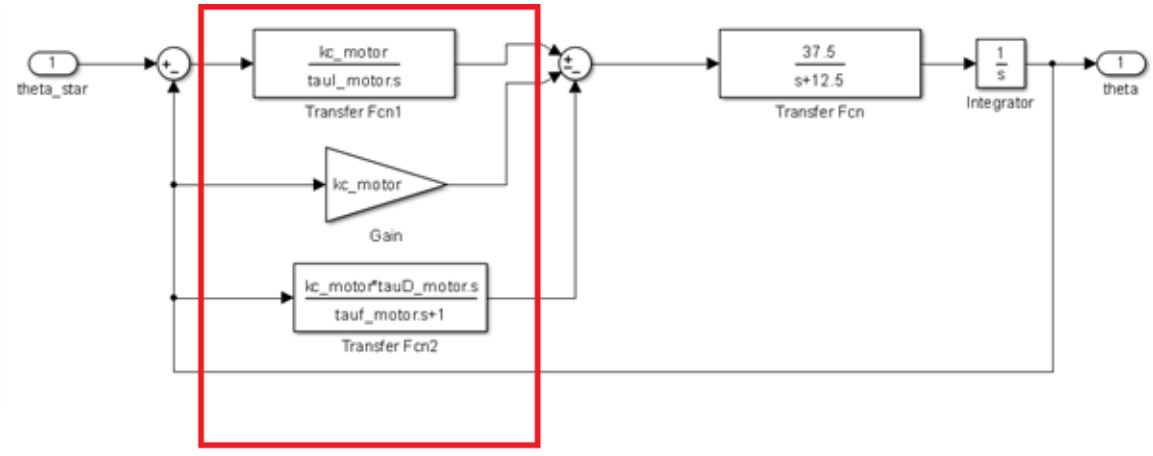

Figure 15 - Inner loop controller

Calculation of the controller parameters is as follows:

$$G(s) = \frac{37.5}{s + 12.5} \times \frac{1}{s} = \frac{37.5}{s(s + 12.5)} = \frac{B(s)}{A(s)} \quad (15)$$

$$C(s) = \frac{C_2 s^2 + C_1 s + C_0}{s(s + l_0)} = \frac{P(S)}{L(S)} \tag{16}$$

The performance specification was chosen as:

$$AC_l(s) = (s^2 + 2\zeta\omega_n s + \omega_n^2)(s + \lambda_1)^2 \tag{17}$$

Where $\zeta, \omega_n \; and \; \lambda_1$ sets the location of the closed-loop poles such that ζ is the damping factor and $\omega_n \; and \; \lambda_1$ sets the frequency of the response. ζ was chosen to be 0.707 which is critically damped. This gives the system the fastest response without resonance. The other parameters chosen were $\omega_n = \lambda_1 = 50$. The PID parameters were found by letting the actual system closed loop poles be equal to the desired closed loop poles as shown in the performance specification:

$$A(s) \times L(s) + B(s) \times P(s) = AC_l(s) \tag{18}$$

The control parameters $C_2, C_1, C_0 \; and \; l_o$ were then calculated by equating the coefficients of both sides of the equation. Finally the PID parameters were found by rearranging the equation:

$$C(s) = \frac{C_2 s^2 + C_1 s + C_0}{s(s + l_o)} = Kc\left(1 + \frac{1}{\tau_I s} + \frac{\tau_D s}{\tau_f s + 1}\right) \tag{19}$$

$$\therefore Kc = 65.2748, \qquad \tau_I = 0.0620\,,\ \tau_D = 0.0197\,,\ \tau_f = 0.0063$$

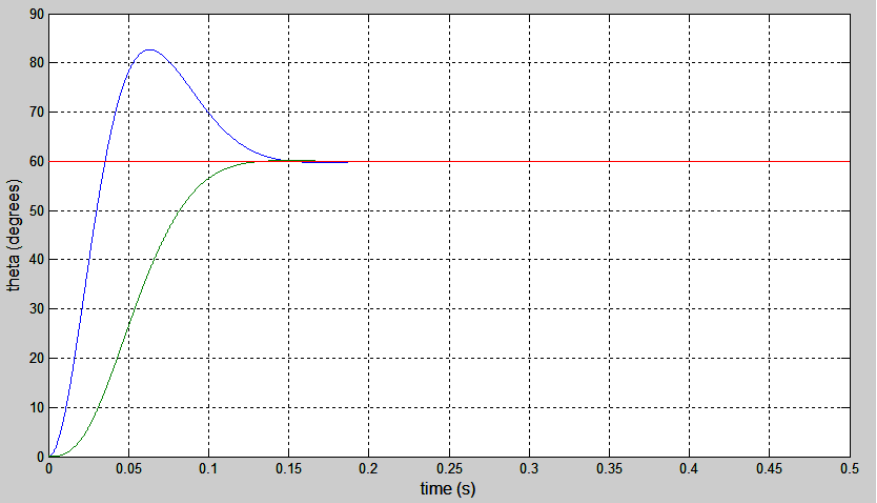

Figure 16 - Inner Loop Controller Simulation

The inner loop was simulated using the calculated parameters. Figure 16 shows the output of the inner loop simulation for a step change of the reference signal from 0 to 60 degrees. The plot shown in green is the response of the inner loop with an I-PD controller. It took 0.13 seconds for the output angle to match the reference. The Hitec servo data sheet shows that the HS-7975HB servo has a speed of $0.1 secs/60\,°$. This means that the designed controller is slightly slower than the rated performance of the servo it is intended to mimic. This is desired

as the rated speed is with no load and also because a slower performance is more detrimental to the outer loop performance. This means that there is a margin for error built in to the design. In order to demonstrate the effect of implementing an I-PD controller instead of a more traditional PID controller, the inner loop was simulated with a PID controller using the same control parameters. The plot shown in blue shows the response of the system with a PID controller. There was a 32% overshoot of the desired servo angle. It also took 1.53 seconds to be stable on the desired angle which is significantly slower than the I-PD controller.

4.4 Outer Loop Controller

The outer loop controller was designed in the same manner as the inner loop controller. The natural frequency ω_n was chosen to be a minimum of 10 times slower than the natural frequency chosen for the inner loop controller. This allows the inner loop dynamics to be ignored in the outer loop design process while still ensuring outer loop stability. The performance specification for the outer loop varies for each of the control objectives so the controller solution shown below is using pronumerals only. The actual controller parameters used for each control objective are shown later in this chapter. A derivation of the I-PD controller parameters is as follows:

$$G(s) = \frac{b}{s^2} = \frac{B(s)}{A(s)} \qquad \text{where b} = \frac{-5}{7} g\theta \tag{20}$$

$$C(s) = \frac{C_2 s^2 + C_1 s + C_0}{s(s + l_0)} = \frac{P(s)}{L(s)} \tag{21}$$

The performance specification was chosen as:

$$AC_l(s) = (s^2 + 2\zeta s + \omega_n^2)(s + \lambda_1)^2$$

The control parameters are identified by letting the desired closed loop polynomial equal the actual closed loop polynomial:

$$B(s)P(s) + A(s)L(s) = Acl(s)$$

$$s^4 + s^3 l_0 + bC_2 s^2 + bC_1 s + bC_0 = s^4 + (2\zeta + 2\lambda_1)s^3 + (2\lambda_1\omega_n^2 + 2\zeta\omega_n\lambda_1^2)s + \lambda_1^2\omega_n^2$$

By inspecting the equality of the equations:

$$l_o = 2\zeta\omega_n + 2\lambda_1 \tag{22}$$

$$C_2 = \frac{\omega_n^2 + 4\lambda_1\zeta\omega_n + \lambda_1^2}{b} \tag{23}$$

$$C_1 = \frac{2\lambda_1\omega_n^2 + 2\zeta\omega_n^2}{b} \tag{24}$$

$$C_0 = \frac{\lambda_1^2\omega_n^2}{b} \tag{25}$$

The form for a practical PID controller is:

$$C(s) = K_C\left(1 + \frac{1}{\tau_I s} + \frac{K_C \tau_D s}{\tau_F s + 1}\right) \tag{26}$$

Rearranging Eqn. (26) into the same form as Eqn. (21) gives:

$$C(s) = \frac{\frac{K_c \tau_I(\tau_F + \tau_D)}{\tau_I \tau_F} s^2 + \frac{K_c(\tau_I + \tau_F)}{\tau_I\, \tau_F} s + \frac{K_C}{\tau_I \tau_F}}{s\left(s + \frac{1}{\tau_F}\right)} \tag{27}$$

Therefore:

$$C(s) = \frac{\frac{K_c \tau_I(\tau_F + \tau_D)}{\tau_I \tau_F} s^2 + \frac{K_c(\tau_I + \tau_F)}{\tau_I\, \tau_F} s + \frac{K_C}{\tau_I \tau_F}}{s\left(s + \frac{1}{\tau_F}\right)} = \frac{C_2 s^2 + C_1 s + C_0}{s(s + l_0)} \tag{28}$$

By inspecting the equality of the equations:

$$l_0 = \frac{1}{\tau_F}\ , \quad C_2 = \frac{K_c \tau_I(\tau_F + \tau_D)}{\tau_I \tau_F}, \quad C_1 = \frac{K_c(\tau_I + \tau_F)}{\tau_I\, \tau_F} \ \ and\ C_0 = \frac{K_C}{\tau_I \tau_F} \tag{29}$$

By evaluating Eqns. (22-25) and Eqn. (29) and solving for all variables simultaneously, the PID control parameters $\tau_I, \tau_F, \tau_D\ and\ K_c$ are identified as:

$$\tau_F = \frac{1}{2\zeta\omega_n + 2\lambda_1} \tag{30}$$

$$\tau_I = \frac{2\lambda_1\omega_n^2 + 2\zeta\omega_n\lambda_1^2}{\lambda_1^2\omega_n} - \frac{1}{2\zeta\omega_n + 2\lambda_1} \tag{31}$$

$$K_C = \frac{\frac{\lambda_1^2\omega_n^2}{2\zeta\omega_n + 2\lambda_1}\left(\frac{2\lambda_1\omega_n^2 + 2\zeta\omega_n\lambda_1^2}{\lambda_1^2\omega_n} - \frac{1}{2\zeta\omega_n + 2\lambda_1}\right)}{b} \tag{32}$$

$$\tau_D = \frac{\omega_n^2 + 4\lambda_1\zeta\omega_n + \lambda_1^2}{(2\zeta\omega_n + 2\lambda_1)\frac{\lambda_1^2\omega_n^2}{2\zeta\omega_n + 2\lambda_1}\left(\frac{2\lambda_1\omega_n^2 + 2\zeta\omega_n\lambda_1^2}{\lambda_1^2\omega_n} - \frac{1}{2\zeta\omega_n + 2\lambda_1}\right)} \tag{33}$$

While the deriving the above equations using pronumerals is tedious it is becomes advantageous when tuning the performance parameters. Once the derivations were entered

into MATLAB, it was possible to quickly obtain a new numerical calculation of the control parameters after every change of performance specification. This makes it an easy process to find the optimum values for τ_I, τ_F, τ_D *and* K_c.

4.5 Sinusoidal Tracking

I-PD controllers are unable to track a sinusoidal input without tracking error. Traditionally a sinusoidal input can be tracked with a resonant controller in which the reference frequency is embedded into the controller design. The resulting controller has a very high gain centred around the reference frequency[44]. A resonant controller was designed for the ball and plate system and the simulated results were very positive. A problem arose when it came time to implement the design on the BeagleBoard XM development board. Using the BeagleBoard XM with a Linux operating system meant that at times the sample time of the controller was erratic. It was noticed that due to background processes occurring in the operating system, occasionally the desired sampling period was exceeded by as much as 30%. Due to the controller complexity the designed resonant controller was discretised using the bilinear transform method. The bilinear transform method maps the continuous time controller in the s-domain to a discrete time controller in the z-domain at a specific sampling frequency. This resulted in a discrete controller that was not tolerable to variations in the sampling period. A new method of tracking a sinusoidal input was needed. The I-PD controller used previously was discretised using the backward difference method. This method maintains the sampling period as a variable in the control equation. By measuring each sample period and inserting the measured value into the control equation, the problem of variable sample time is negated.

4.5.1 I-PD Controller with Reference Pre-compensator

A sinusoidal reference was applied to the I-PD controller designed in chapter 4.4 and simulated against the nonlinear ball and plate model. The results shown in Figure 17 below indicate that the controller was unable to track the sinusoidal reference as was expected.

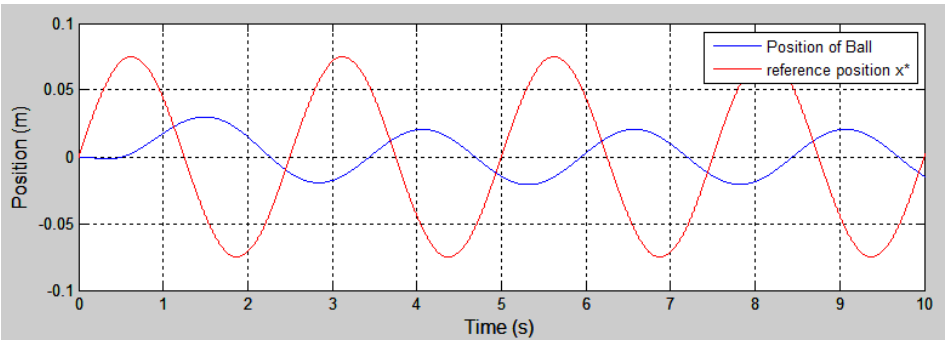

Figure 17 - I-PD control scheme with sinusoidal input

In Figure 17 above the reference signal was $R_x = 0.075 \sin\left(\frac{2\pi}{2.5}\right)$. The magnitude and phase response of the closed loop system was then found by measuring the peaks of the input and output waveforms. The magnitude and phase response was found as:

$$gain_x = \frac{y(t)}{r(t)} = 0.2566 \tag{34}$$

$$phase\_shift_x = 3.8973rads \tag{35}$$

By finding these values a pre-compensator was able to be designed to negate the negative phase and gain response. The input signal was amplified by the inverse of the gain and phase shifted by the negative of the phase response as shown below:

Given the reference signal:

$$R_x = Asin(w0t) \tag{36}$$

The pre-compensated reference signal is found as:

$$\overline{R_x} = \frac{\mathrm{A}}{\mathrm{gain_x}} \sin(\mathrm{w0t} - \mathrm{phase_x}) \tag{37}$$

Therefore for the reference in Figure 17 the pre-compensated reference signal was:

$$\overline{R_x} = \frac{0.075}{0.2566} \sin(\mathrm{w0t} - \mathrm{phase_x}) \tag{38}$$

The pre-compensated reference signal was applied to the system and the resulting output is shown below in Figure 18. The plot shown in red is the initial reference signal and the plot in blue is the simulated position of the ball. It was seen that the system was able to track the input sinusoidal reference signal perfectly with no tracking error.

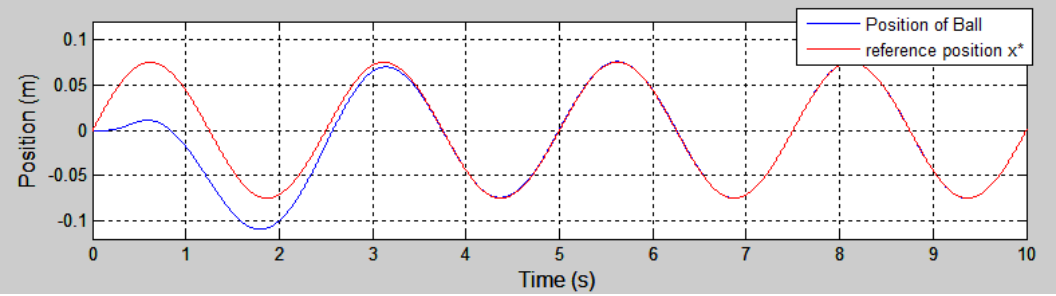

Figure 18 - Sinusoidal reference with pre-compensator

In order to use the pre-compensation method the gain and phase response of the closed loop system to the specific input frequency is required. A MATLAB script was generated to simulate the system against a range of input frequencies from $w0$=0.01rads/s to 10 rads/s 0.01 rads/s increments. At each input frequency the magnitude and phase response of the system was found. The magnitude and phase plots of the system are shown below in Figure 19. Any sinusoidal reference with $\omega_0$between 0.01 rads/s and 10 rads/s is able to be tracked by interpolating the magnitude and phase response from the obtained values.

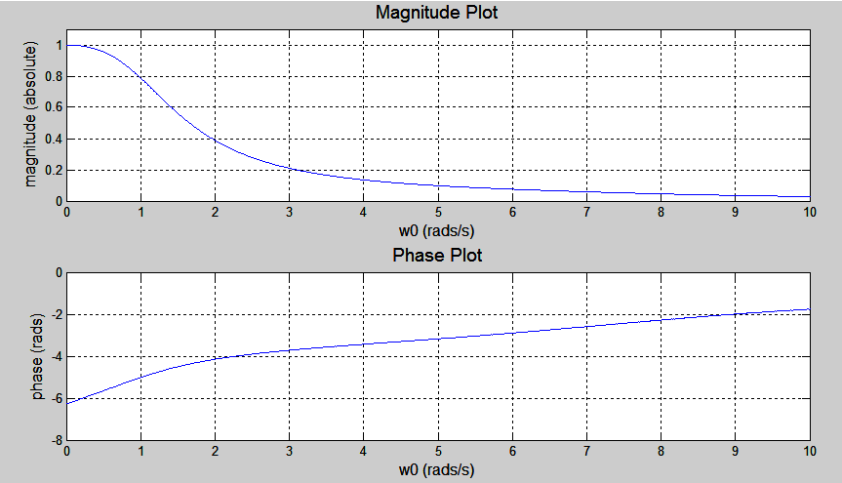

Figure 19 – Closed loop magnitude and phase response of the system

4.6 Ball Stabilisation and Disturbance Rejection

The disturbance rejection control objective requires the ball to be stabilised at the centre of the plate and to reject disturbances to the ball's position. In the ball and plate system prototype this disturbance comes in the form of a person providing a pulse disturbance to the balls position by hitting it away from the centre of the plate. To set the position of the ball at the centre of the plate the input reference position to the plate in the x-axis and y-axis is set to zero. Choosing the values for ω_n and λ_1 is a trade-off between speed and stability. If these values are too high then the system response can become too fast and become unstable. If the values are chosen too low then the response becomes very slow. After tuning the performance specification the chosen parameters were $\omega_n = \lambda_1 = 3$. Figure 20 below shows the response of the system in the x-axis. Only one axis was shown as the responses are identical for both axes. In the test the reference position was set at R(s)=0 and at t=30 seconds a 0.1 second pulse disturbance of 0.05m was applied to the feedback position. The results show that the system was stable and that the disturbance was able to be rejected within 0.8 seconds.

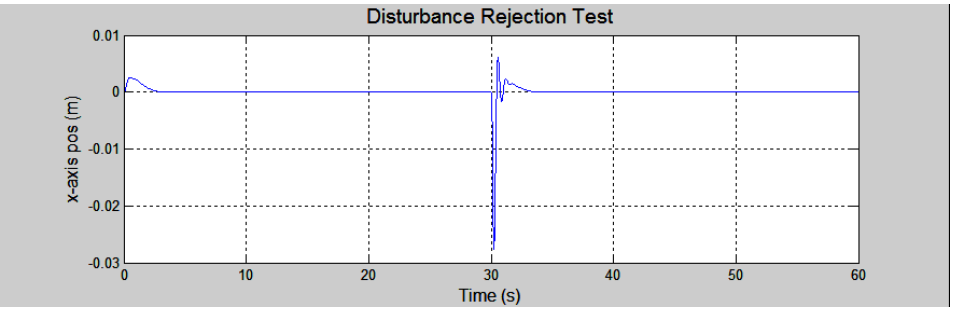

Figure 20 - Disturbance Rejection Test Simulation

4.7 Square Tracking

The square tracking control objective was to follow a predefined square trajectory on the plate. For this objective the I-PD control parameters were chosen as $\omega_n = \lambda_1 = 3.3$. The square trajectory was generated by stepping the reference x,y position pairs through the coordinates of a square. For a square of $side = 15cm$ the reference signal used was:

$$R_{x,y} = \{-0.075, 0.075\}, \{0.075, 0.075\}, \{0.075, -0.075\}, \{-0.075, -0.075\}.$$

The resulting simulation, shown in Figure 21, indicates that the system was able to step to each position in 2.6 seconds with almost zero overshoot and zero settling time. This particular control objective really highlights the reason why the I-PD controller is more effective than a PID controller for this system. Because the PID controller has overshoot, designing the trajectory for the square would be much more difficult. The reference trajectory would need to take the overshoot into account. Once the ball has reached each position the reference position can be changed to the next position in the desired trajectory.

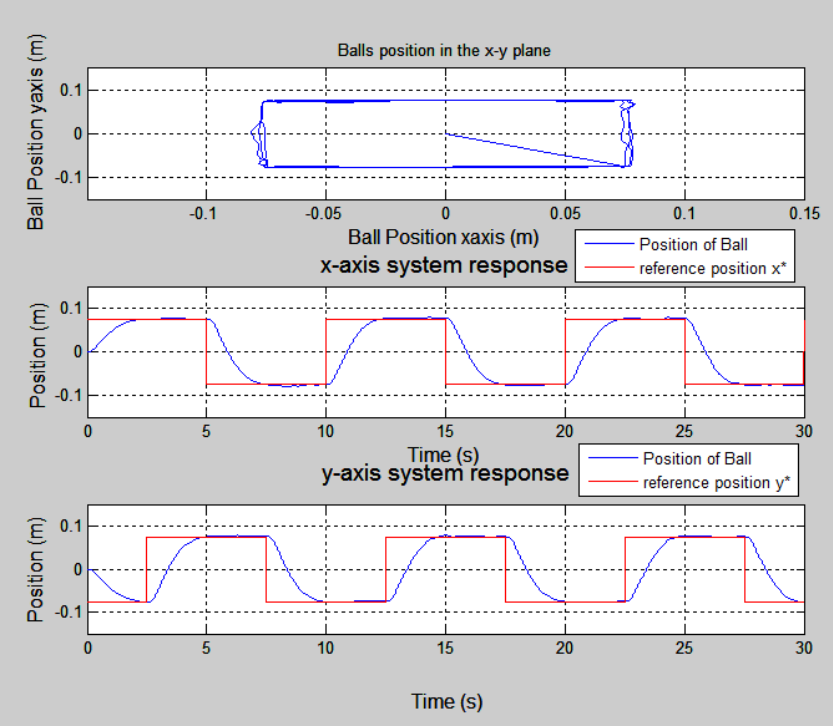

Figure 21 - Square Tracking Simulation

4.8 Maze Tracking

The maze tracking specification stated that the system shall provide the capability to track the maze path shown in Figure 6. The trajectory was generated by mapping the coordinates of each corner of the maze. The reference position of the system was then stepped through each coordinate pair in turn to allow the ball to follow the maze path. Rather than stepping the system at a predetermined time period, the reference position was stepped when the ball position error was less than 1mm. This meant that each reference position was held only as

long as it was needed for the ball the move to the next corner. Figure 22 below shows the x and y-axis response plots.

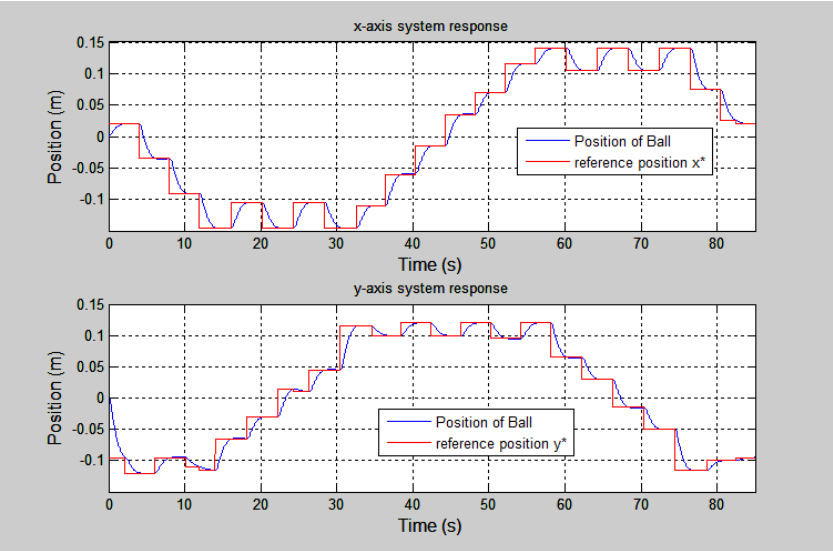

Figure 22 – Simulated Maze Tracking x and y-axis Response

The simulated trajectory shown below in Figure 23 matches the trajectory printout sheet in Figure 6.

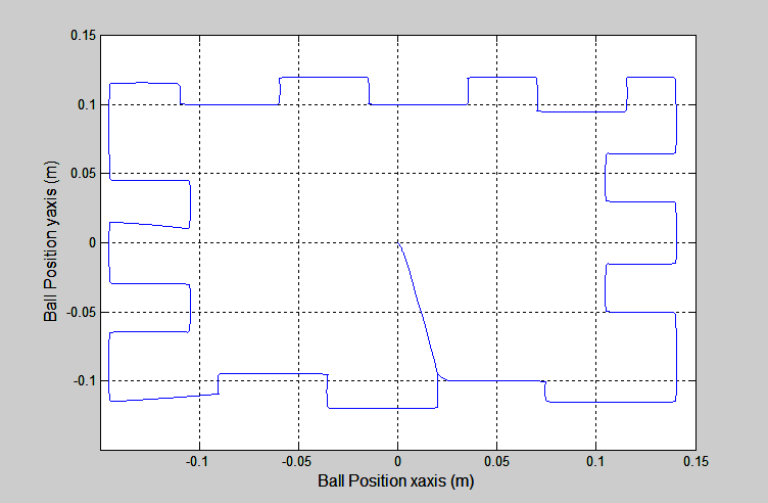

Figure 23 – Simulated Maze Tracking Trajectory

4.9 Circle Tracking

In accordance with the design specification in chapter 1.2 the system must track a circular trajectory of radius of 7.5cm with one revolution of the circle taking 2.5 seconds. It was found that a circular trajectory was able to be generated by tracking a sinusoidal signal in the x and y-axis that is 90 degrees out of phase. The input frequency was chosen as $w0 = \frac{2\pi}{2.5}$ rads/s which

translates to one cycles every 2.5 seconds. Therefore the reference signals for the circular trajectory were chosen as:

$$R_x = 0.075 \sin\left(\frac{2\pi}{2.5}t + \frac{\pi}{2}\right) \tag{39}$$

$$R_y = 0.075 \sin\left(\frac{2\pi}{2.5}t\right) \tag{40}$$

In order to again demonstrate the action of the pre-compensator Figure 24 shows the response of the system to the reference signals in Eqn. (39) and Eqn. (40) without pre-compensation applied. The output waveform was unable to track the input.

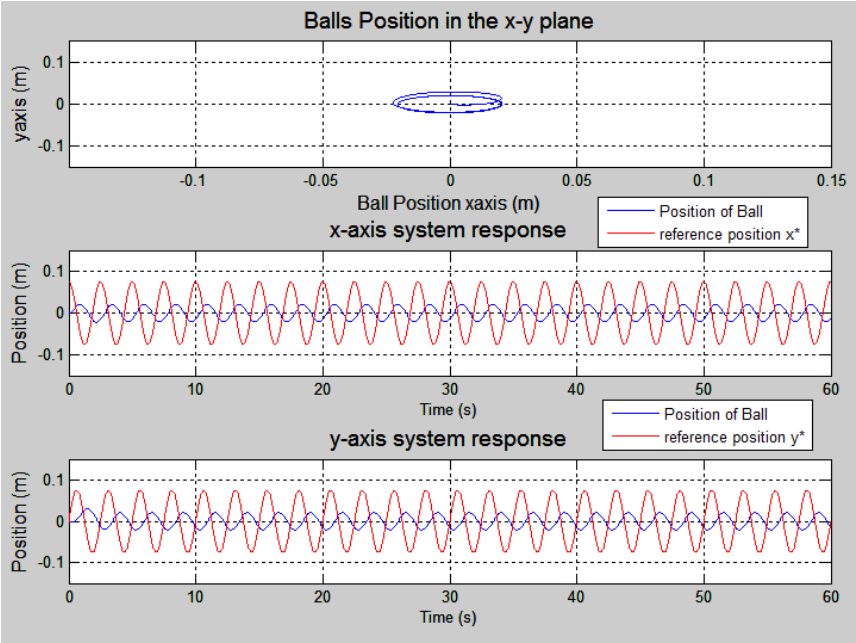

Figure 24 - Circle Tracking without Pre-Compensator

The pre-compensation values were obtained by interpolating magnitude and phase data for the chosen input frequency of $\omega_0 = \frac{2\pi}{2.5}$ rads/s. Applying pre-compensation formula in Eqn. (37) and Eqn. (38) resulted in the following reference signals:

$$R_x = \frac{0.075}{0.2566} \sin\left(\frac{2\pi}{2.5}t + \frac{\pi}{2} - 3.8973\right) \tag{41}$$

$$R_y = \frac{0.075}{0.2566} \sin\left(\frac{2\pi}{2.5}t - 3.8973\right) \tag{42}$$

Figure 25 shows the results of the circle tracking simulation with reference signal pre-compensation. In the simulation the ball starts at the centre of the plate and within 1.7 seconds is tracking the reference signal exactly. This simulation achieves the control objective of tracking a circle with a radius of 7.5cm in 2.5 seconds per revolution.

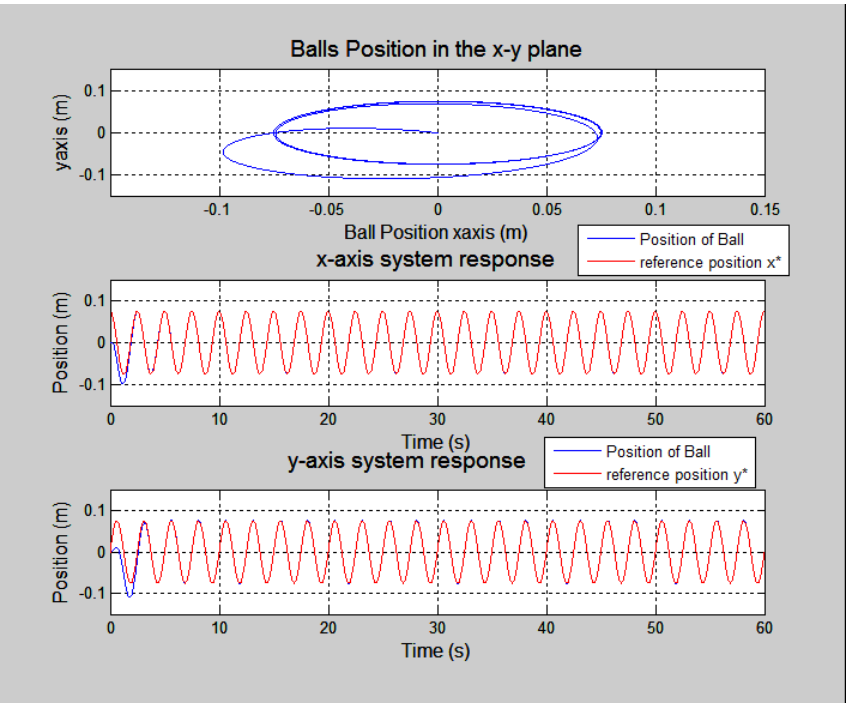

Figure 25 - Circle tracking with reference pre-compensator

4.10 Figure-eight Tracking

As a further extension of the sinusoidal tracking function of the system there is a specification that states that the system shall provide the capability to follow a figure-eight trajectory on the plate. To achieve the desired trajectory the reference signals are sinusoidal signals in which the x-axis reference signal is half the frequency of the y-axis reference signal. A reference signal pre-compensation is still required to properly track the reference signal. The pre-compensation values are different because the system gain and phase shift vary with reference frequency. In the simulation shown in Figure 26 the pre-compensated reference signals chosen are:

$$R_X = \frac{0.12}{0.098}\sin\left(\frac{2\pi}{5}t - 3.1750\right) \tag{43}$$

$$R_Y = \frac{0.075}{0.02566}\sin\left(\frac{2\pi}{2.5}t - 3.9873\right) \tag{44}$$

At t=0 the ball is at the centre of the plate and within 3.08 seconds the position of the ball follows the reference position exactly. This results in the figure 8 pattern shown in the first subplot. This satisfies the figure 8 control objective.

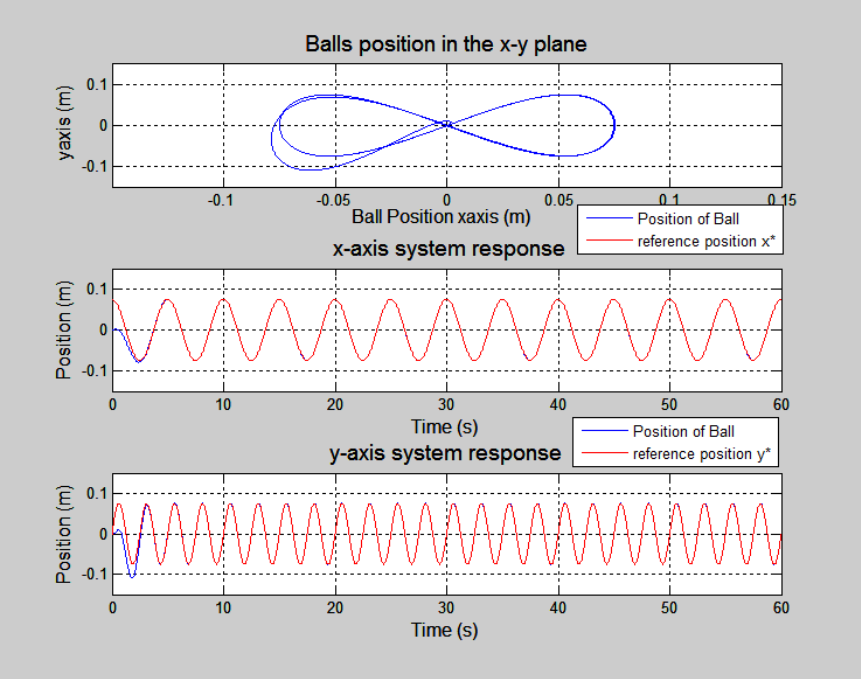

Figure 26 - Figure 8 tracking simulation

4.11 Discretisation of the Controller

The controller that was designed and simulated was a continuous time controller. To implement the controller on a microprocessor it is necessary to convert the continuous time controller to a discretised form. The bilinear transform method and backward difference method were identified as possible solutions to this discretisation. The discrete time controller found using bilinear transform is only valid for a specific sampling period Δt. In initial testing of the BeagleBoard XM it was found that regardless of the timing method used there was always small variations in Δt. In order to get accurate enough timings for use with the bilinear transform controller a complex process of applying real time patches to the operating system would be required. Using the backward difference method allows for Δt to remain as a variable of the control equation. This means that the variations of Δt can be measured and input into the control equation. This makes the discrete controller tolerable to variations in Δt. The backward difference method of discretisation was chosen for this reason. The discrete time controller was derived as follows:

Control signal $U(s)$ of an I-PD controller:

$$U(s) = \frac{K_C}{\tau_I s}\big(R(s) - Y(s)\big) - K_C Y(s) - \frac{K_C \tau_D s}{\tau_F s + 1} Y(s) \qquad (45)$$

Let:

$$U(s) = U_{PI}(s) + U_D(s) \tag{46}$$

where: $U_{PI}(s) = \frac{K_C}{\tau_I s}\big(R(s) - Y(s)\big) - K_C Y(s)$ and $U_D(s) = -\frac{K_C \tau_D s}{\tau_F s + 1} Y(s)$

Looking at the Proportional and integral parts only:

$$sU_{PI}(s) = \frac{K_C}{\tau_I}\big(R(s) - Y(s)\big) - K_C s Y(s) \tag{47}$$

Taking the inverse Laplace transform of both sides gives:

$$\dot{u}(t) = \frac{K_C}{\tau_I}\big(r(t) - y(t)\big) - K_C \dot{y}(t) \tag{48}$$

Using the backward difference method to approximate $\dot{u}(t)$ and $\dot{y}(t)$ at sample time t_i gives:

$$\dot{u}(t_i) \approx \frac{u(t_i) - u(t_i - \Delta t)}{\Delta t} \tag{49}$$

$$\dot{y}(t_i) \approx \frac{y(t_i) - y(t_i - \Delta t)}{\Delta t} \tag{50}$$

Subbing (31) and (32) into (30) gives:

$$\frac{u(t_i) - u(t_i - \Delta t)}{\Delta t} = \frac{K_C}{\tau_I}\big(r(t) - y(t)\big) - K_C\left(\frac{y(t_i) - y(t_i - \Delta t)}{\Delta t}\right) \tag{51}$$

Solving for $u(t_i)$ gives the proportional and integral part of the control signal:

$$u(t_i) = u(t_i - \Delta t) + \frac{K_C \Delta t}{\tau_I}\big(r(t) - y(t)\big) - K_C\big(y(t_i) - y(t_i - \Delta t)\big) \tag{52}$$

The derivative part of the control signal was then found:

$$U_D(s) = -\frac{KC\tau_D s}{\tau_F s + 1} Y(s) \tag{53}$$

$$\tau_F s U_D(s) + U_D(s) = -K_C \tau_D s y(s) \tag{54}$$

Taking the Laplace transform of both sides gives:

$$\tau_F \dot{u}(t) + u(t) = -K_C \tau_D \dot{y}(t) \tag{55}$$

Using the backward difference approximation for $\dot{u}(t)$and $\dot{y}(t)$ as in Eqn. (49) and Eqn. (50) at I-PD sample time t_i gives:

$$\tau_F\left(\frac{u(t_i)-u(t_i-\Delta t)}{\Delta t}\right)+u(t_i)=-K_C\tau_D\left(\frac{y(t_i)-y(t_i-\Delta t)}{\Delta t}\right) \quad (56)$$

Solving for $u(t_i)$ gives the derivative part of the control signal:

$$u(t_i)=\frac{\tau_F}{\tau_f+\Delta t}u(t_i-\Delta t)-\frac{K_C\tau_D}{\tau_f+\Delta t}\big(y(t_i)-y(t_i-\Delta t)\big) \quad (57)$$

Combining Eqn. (52) and Eqn. (57) gives the total control signal as:

$$u(t_i)=u(t_i-\Delta t)+\frac{K_C\Delta t}{\tau_I}\big(r(t)-y(t)\big)-K_C\big(y(t_i)-y(t_i-\Delta t)\big)+\frac{\tau_F}{\tau_f+\Delta t}u(t_i-\Delta t) -\frac{K_C\tau_D}{\tau_f+\Delta t}\big(y(t_i)-y(t_i-\Delta t)\big) \quad (58)$$

Eqn. (58) represents the velocity form of the discrete I-PD controller for one axis. The other axis has an identical controller. The equation is in velocity form as it relies on the previous control signal plus the change in control signal. This equation was very easy to convert into code with each variable saved as a double data type.

4.12 Selection of Δt

When discretising a continuous time controller the general rule is to choose Δt such that the I-PD sample rate is 10 times faster than the natural frequency. i.e. $\Delta t=\frac{1}{10\omega_n}$. In order to have a better understanding of the effect of different I-PD sample rates the discrete controller was simulated in matlab against a discrete plant model. Initially Δt was chosen as $\Delta t=25ms\ or\ f_s=40Hz$. The first subplot in Figure 27 shows that this sampling rate was too slow and the result was oscillations about the reference. In the second simulation the sample rate was chosen as 100Hz or $\Delta t=10ms$. With the faster value of sample rate the response was smooth and very closely matched the response from the continuous time controller. As there was no technical impediment to choosing a sample rate this fast, 10ms was chosen as the value for Δt.

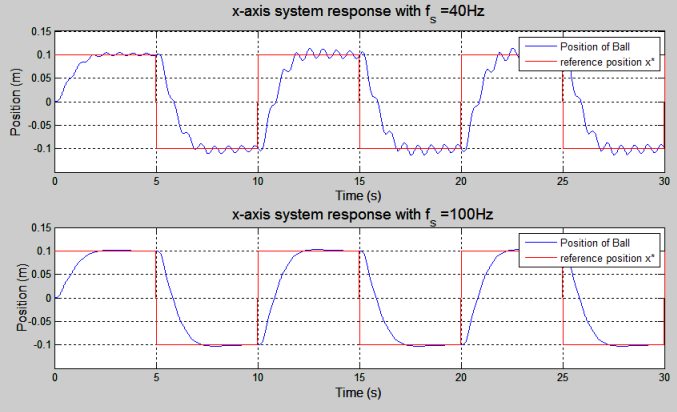

Figure 27 - Effect of using different I-PD sampling rates

4.12.1 Anti-Windup

Integrator windup occurs when the amplitude of the control signal is constrained and the calculated control signal exceeds the constraints. Because the controller calculated in Eqn. (58) is in velocity form, the integral action of the controller can be stopped simply by applying the constraint. At the next controller sample time the controller is aware of the previous saturation value as it is used in the controller equation as $u(t_i - \Delta t)$. Three levels of anti-windup were implemented. The control signal was constrained to $\pm 15°$ or $\pm 0.261\ rads$. This was to limit the maximum angle of the plate to the mechanical limits of the design. The derivative of the control signal was constrained to $7\ rads/s$. This was to ensure the velocity of the plate wasn't fast enough to cause the ball to be thrown from the surface of the plate. The double derivative of the control signal was constrained to $\pm 10\ rads/s^2$. This was to limit the acceleration of the plate in order to protect the servo components from damage due to excessive torque. The anti-windup was implemented as follows:

Let the limits of the double derivative of the control signal equal $DDU_{max}\ and\ DDU_{min}$. The limits of the derivative of the control signal equal DU_{max} and DU_{min}. The limits of the control signal equal U_{max} and U_{min}.

At sample time the control signal $u(t_i)$ is calculated using Eqn. (58) and the derivative of the control signal was estimated using the backward difference method:

$$\dot{u}(t_i) = \frac{u(t_i) - u(t_i - \Delta t)}{\Delta t} \tag{59}$$

Then each of the constraints was applied for the derivative of the control signal and the new control signal calculated based on the new constraint:

$$\text{If } \dot{u}(t_i) < DU_{max} \text{ then } u(t_i) = u(t_i - \Delta t) + DU_{max}\Delta t \tag{60}$$

$$\text{If } \dot{u}(t_i) < DU_{min} \text{then } u(t_i) = u(t_i - \Delta t) + DU_{min}\Delta t \tag{61}$$

The double derivative of the control signal was then estimated using the backward difference method. The double derivative constraints were applied and the new control signal was calculated:

$$\ddot{u}(t_i) = \frac{\dot{u}(t_i) - \dot{u}(t_i - \Delta t)}{\Delta t} \quad (62)$$

If $\ddot{u}(t_i) > \text{DDU}_{max}$ then $u(t_i) = u(t_i - \Delta t) + \dot{u}(t_i - \Delta t)\Delta t + \text{DDU}_{max}(\Delta t)^2$ (63)

If $\dot{u}(t_i) < \text{DDU}_{min}$ then $u(t_i) = u(t_i - \Delta t) + \dot{u}(t_i - \Delta t) + \text{DDU}_{min}(\Delta t)^2$ (64)

Finally the control signal constraints were applied:

$$if\ u(t_i) > \text{U}_{max} \text{ then } u(t_i) = \text{U}_{max} \quad (65)$$

$$if\ u(t_i) < \text{U}_{min} \text{ then } u(t_i) = \text{U}_{min} \quad (66)$$

Chapter 5 - Software Implementation

5.1 Introduction

The software implementation stage of the project involved generating the software code to operate the ball and plate system prototype. The software implementation process was conducted using the following steps:

a) Design the Software configuration
b) Implement and test the touchscreen code.
c) Implement and test the servo code.
d) Implement and test the Human Machine Interface.
e) Implement the controller code.
f) Implement the code to obtain the experimental testing results.

This chapter describes how each of the software implementation steps was conducted.

5.2 Software Configuration

The BeagleBoard XM development board was flashed with Ubuntu 13.04 (Raring) and all of the software was coded in C. Using a Ubuntu distribution meant that there were drivers available for the hardware used. It did make it more difficult to get accurate sample rates for the I-PD controller due to having the background operating system but overall it was much easier to implement than using a micro-controller based design. Software version control was maintained using a git database and the project git repository is available online at https://github.com/john-s-lee/Ball_Plate. Development was conducted on feature branches and once a feature was fully implemented it was merged back into the master branch. The experimental results collection code was conducted in its own branch to avoid contaminating the code from the ball and plate demonstrator. The git commit tagged v1.0 was the final version of the code used on the ball and plate system prototype.

5.2.1 Disaster Recovery and rapid deployment

A method of rapid deployment of the ball and plate software was developed. The reason for this was so that in the event of a corrupted image, the software could be quickly replicated. This also means that if this product were ever to be produced it is an easy process to install the software. The most up to date fully working version of the code was always kept as the master git branch. This means that with a working image a git clone will provide the best version of the software. In the scripts folder of the git is a ball and plate install script. This script installs all necessary debian packages, downloads all of the sound files off dropbox and installs them in the correct directory. In addition to this there is a startup option in the projects makefile. Typing the command “sudo make startup” copies the relevant script into the etc/init.d folder and makes it execute on startup.

5.3 Resistive Touchscreen Code

The analogue resistive touchscreen was shipped with a compatible USB controller. The USB controller was recognised by Ubuntu as an evdev device. This meant that a /dev/input/eventx event file was created for the touchscreen. A device rules file was created so that a touchscreen symlink file was linked to the relevant eventx file when the USB controller was plugged in. This was due to the fact that the event file number changes depending on the order that USB devices are recognised. The event data structure was available through the linux/input.h header file. A touchscreen event handler program was coded that ran off a seperate thread to the main code. The touchscreen events were read with a blocking read function in a never ending loop. This ensured that the latest touchscreen events were detected immediately and that no touchscreen events were missed.

5.3.1 Touchscreen Calibration

Each touchscreen reading was an integer between 0 and 4000. On the x-axis 0 was on the left of the screen and 4000 was on the right. On the y-axis 0 was on the bottom and 4000 on the top. Using a simple program to output the current touched position, the centre of the touchscreen was found to be at (2018, 2070). The active touchscreen area was 0.35m wide and 0.28m high. This meant that in the x-axis there were $\frac{4000}{0.35} = 11428.5$ steps/m and in the y-axis there were $\frac{4000}{0.28} = 14285.7\ \text{steps/m}$. From this a formula for calculating the position in metres from a touchscreen reading was found as:

$$x_{pos} = \frac{readvalue - 2018}{11428.5} \tag{67}$$

$$y_{pos} = \frac{readvalue - 2070}{14285.5} \tag{68}$$

A simple touchscreen test was conducted in which a pattern was drawn on the touchscreen and each measured coordinate saved to a text file. This text file was imported into matlab and plotted. Figure 28 below shows the results of the test.

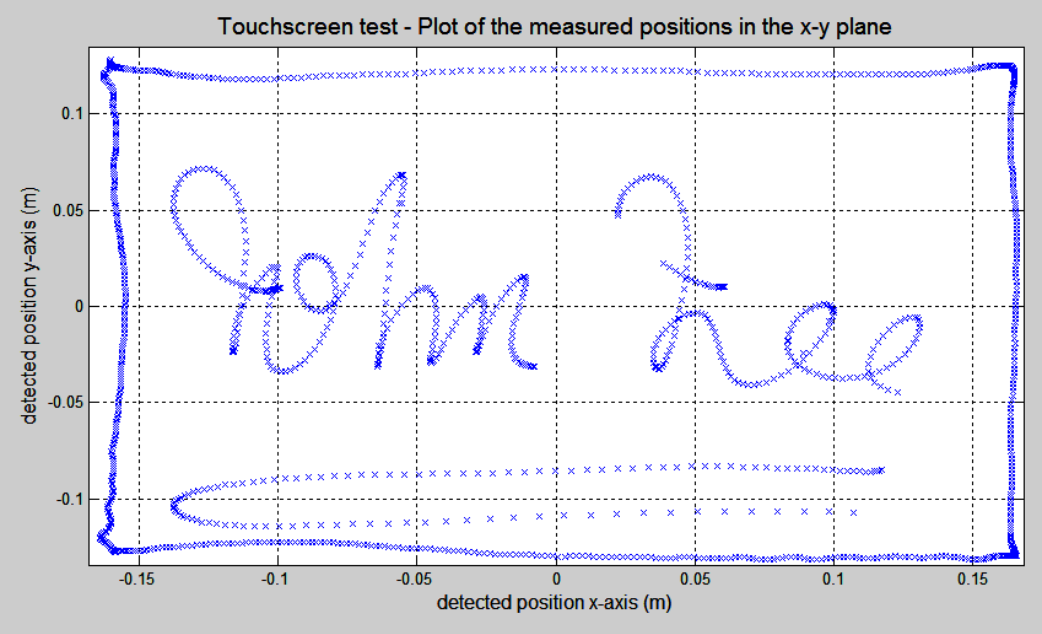

Figure 28 - Touchscreen Test

5.3.2 Ball Position estimator

One problem with using an external touchscreen controller as an event device is that there is no way to control the sample time of the touched coordinates. Monitoring the touch events showed that typically that there was 6ms between readings for each axis, however this time fluctuated enormously. Moving a finger across the touchscreen quickly resulted in a reading approximately every 6ms but if the finger was slowed then the time between readings increased. With the finger held in a stationary position the readings were as far apart as 100ms. The problem with this is that the I-PD controller has a regular sample time and a position is needed to be known for that specific time. If the last read position is used and a significant amount of time has passed since that reading was taken then there could be large discrepancy between the actual position of the ball and the measured position. In order to improve the accuracy of the ball position feedback a position estimator was used. The position estimator uses an approximation of the balls velocity and acceleration at measurement time and the difference in time between the measurement time and the I-PD sample time to calculate the approximate position of the ball at the I-PD sample time. This approximation is based on the elementary kinematic equation for displacement shown in Eqn. (69).

$$s = ut + \frac{1}{2}at^2 \tag{69}$$

Where s is displacement in metres, t is time in seconds, u is initial velocity in m/s and a is acceleration in m/s^2.

$x(t_t)$ is the measured position of the ball in the x-axis at the touchscreen sample time t_t, $\dot{x}(t_t)$ is the velocity of the ball and $\ddot{x}(t_t)$ is the acceleration of the ball.

The velocity and acceleration of the ball are estimated at touchscreen sample time (t_t) using the backward difference method:

$$\dot{x(t_t)} = \frac{x(t_t) - x(t_t - \Delta t_t)}{\Delta t_t} \tag{70}$$

$$\ddot{x}(t_t) = \frac{\dot{x}(t_t) - \dot{x}(t_t - \Delta t_t)}{\Delta t_t} \tag{71}$$

Based on Eqn. (69) the estimated balls position at I-PD sample time t_i is calculated as:

$$x(t_i) = x(t_t) + \dot{x}(t_t)(t_i - t_t) + 0.5\ddot{x}(t_t)(t_i - t_t)^2 \tag{72}$$

The position of the ball in the y-axis is estimated in the same way.

5.4 Plate Angle Positioning

The mechanical configuration of the servos and plate are shown in Figure 29. The configuration shown in the diagram is for the x-axis but the y-axis is identical. The 6cm distance from the centre of the plate to the servo pivot d_x was chosen by setting some constraints on the system. To ensure proper operation of the servo a maximum servo shaft angle $\theta_S max = 45°$ was required. Analysing the control simulations showed that a maximum plate angle of $\theta_x max = 18°$ was sufficient for all modes of operation. The servo control arm is 2.5cm from the centre of the servo shaft to the pushrod pivot point. To reduce the servo torque required it was necessary to have the longest possible d_x. Therefore by analysing the relationship between the angle of the servo and the plate and the two distances d_m and $d_{x,}$ a value for d_x was found by:

$$\theta_x \approx \frac{d_x}{d_m}\theta_S \tag{73}$$

Applying the constraints:

$$45° = \frac{d_x}{2.5cm} \times 18° \tag{74}$$

Solving for d_x gives $d_x = 6.25cm$. The distance d_x was rounded to 6cm for ease of measurement and drilling for the universal joint mounting hubs.

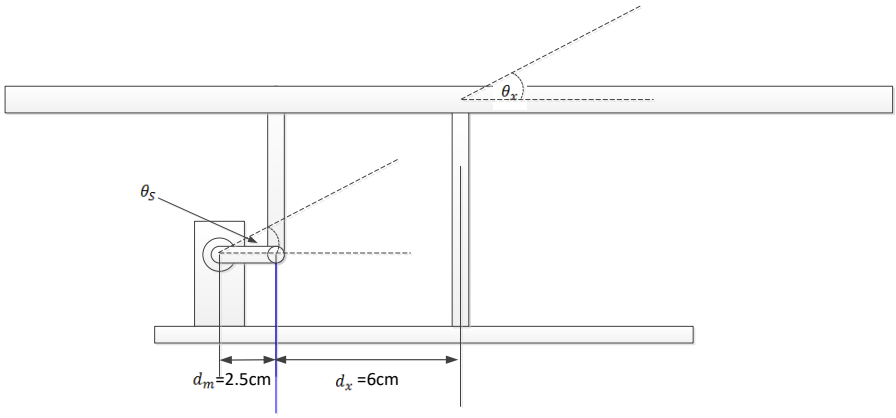

Figure 29 - Ball and Plate prototype pivot points

5.4.1 Micro Maestro Servo Controller

The Micro Maestro servo controller connects to the Beagleboard XM by USB and is recognised by the operating system as a virtual serial device. There are several servo commands that can be sent to the device. These are set maximum speed, set maximum acceleration and set servo position. In most cases the acceleration and speed were set to unlimited as these are set by the output of the I-PD controller. The angle of the servo is set by sending a servo set position command with the position being a pulse width value in quarter microsecond increments. The Hitec servos are at their centre position when they receive a pulse width of 1500 μs and move 1° for every 10μs of change. The output of the controller is a desired plate angle in rads. The required servo angle also takes into account the relationship between the length of the servo arm θ_S and the length between the centre of the plateθ_x. The value to send to the servo controller is calculated as:

$$target = desired\_plate\_angle \times \frac{d_x}{d_S} \times \frac{180}{pi} \times 40 + 4 \times 1500 \quad (75)$$

Substituting for values of d_x and d_S gives:

$$target = desired\_plate\_angle \times \frac{17280}{pi} + 1500 \quad (76)$$

Testing was conducted using a mobile phone accelerometer to confirm that the correct angles were being achieved. The measured angles were very close to what was ordered. It wasn't possible to refine using this method as the accuracy of the mobile phone accelerometer is unknown.

5.4.2 Soft Start

When the ball and plate system prototype is initialised the angle of the plate could be at any angle up to 18°. At this position the ball will be resting against the buffer in one of the corners of the plate. If the controller was allowed to act at full speed from the start then it would quickly change the angle of the plate to 0°. This rapid angle change causes the ball to be flicked into the air which can cause damage to the glass touchscreen. To safeguard against damage a soft start feature is implemented. During the first second of operation the rate of change of

plate angle is limited to a speed which doesn't cause the ball to leave the surface of the plate. This also ensures proper operation of the system as slippage and bouncing were not modelled for in the initial controller design.

5.5 User Interface

The ball and plate balancing prototype software runs all of the different modes of operation from one executable file. To change between the different modes of operation a User Interface (UI) was required. A Nintendo Wiimote controller was selected as the input device and a pair of attached speakers was selected as the output device. These devices were selected as they provide an easy to understand interface, ability to be used while not standing within the vicinity of the moving plate and because they were able to be sourced immediately for free.

5.5.1 Nintendo Wiimote

The Nintendo Wiimote is the controller provided with the Nintendo Wii console system. The Wiimote uses standard Bluetooth without encryption and there are various public libraries available to provide an API. The controller provides access to eight buttons, a digital keypad and IR sensor, accelerometer and four LEDs. Figure 30 shows the features of the Wiimote that are utilised for the ball and plate system prototype. The controller is intended to be held with the digital keypad on the left as shown in the picture.

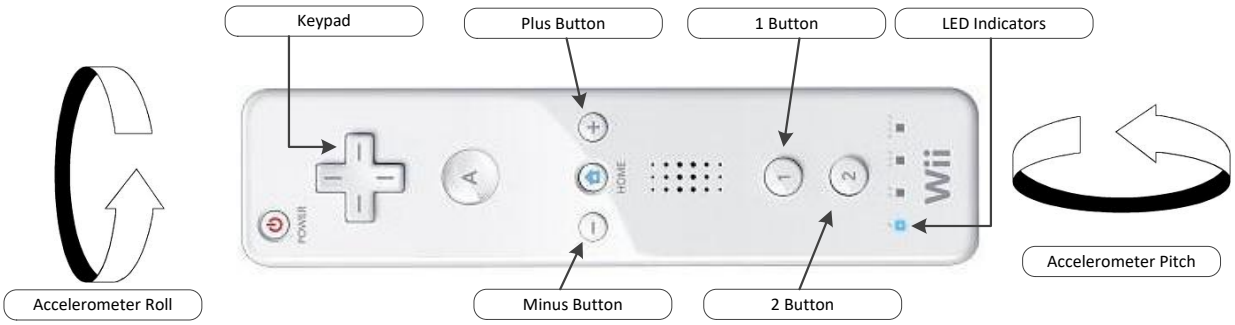

Figure 30 - Nintendo Wiimote used for user input

Communication to the Wiimote is executed through a USB Bluetooth dongle. The libcwiid API https://github.com/abstrakraft/cwiid is used to interface with the device. Button press and accelerometer events are handled though callback functions and the cwiid API has various functions to control access to the LEDs.

5.5.2 Audio Based Feedback

Audio based feedback was used as an alternative to a LCD display as a pair of speakers is much cheaper than an LCD display. The spoken menu phrases were generated as .wav files through the use of AT&Ts text to speech demonstration website at http://www2.research.att.com/~ttsweb/tts/demo.php. Using the website for this project is

acceptable under the Limited Use policy of the website. The .wav files were played at the proper time through the use of the aplay command line .wav player. As the player is a blocking program, the sound player was accessed in a separate thread. If a menu phrase was already being spoken when a new audio play request was generated, then the aplay process is killed so that the new audio can be played immediately.

5.5.3 UI Design

The ball and plate prototype can be in one of five different modes. Initially the system starts in mode 0 which is the initialisation mode. Access to the other modes is not available until this mode has complete. This is where the communication between the program and the Wiimote is initialised. Figure 31 shows the MODE 0 operation.

Mode 0: Initialisation mode
LEDS: all LEDs initially off. All LEDs will flash when Wiimote is in pair mode

Operation: Prior to operating the ball and plate control functions the wiimote needs to be paired to the BeagleBoard XM. Pressing buttons 1 and 2 simultaneously starts the bluetooth pair mode. This mode finishes when the wiimote has been successfully paired.

Figure 31 - UI - Mode 0: Initialisation Mode

On completion of the initialisation mode the prototype can be in 1 of four modes. Changing between these modes is done through the use of the "plus" and "minus" buttons on the Wiimote. The current selected mode is indicated through the four LEDs on the Wiimote as well as from the audio feedback. The operation of the four modes is described below in Figure 32 to Figure 35.

Mode 1: Disturbance Rejection and Square Tracking Mode
LEDS: LED1 on, all other LEDs Off

Operation: Initially the prototype is in disturbance rejection mode. The ball moves to the centre of the plate. Pressing button 1 or 2 toggles between disturbance rejection mode and square tracking mode.

Figure 32 - UI Mode 1: Disturbance Rejection and Square Tracking Mode

Mode 2: Manual Mode
LEDS: LED2 on, all other LEDs Off

Operation: Manual mode allows the user to change the desired position of the ball. The position of the ball is changed by changing the orientation of the Wiimote (acceleromoter mode) or pressing the keypad directions (keypad mode). Buttons 1 & 2 toggle between the two modes.

Figure 33 - UI Mode 2: Manual Mode

Mode 3: Circle Mode and Figure 8 Mode
LEDS: LED3 on, all other LEDs Off

Operation: Mode initially starts in Circle Mode. Toggling between Circle mode and figure 8 mode is by pressing button 1. The direction of the ball in the circle or figure 8 can be reversed by pressing the 2 button.

Audio Feedback:

Initial: "Circle Mode. Press the 1 button to toggle between circle mode and figure 8 mode. Press the 2 button to reverse direction."

When circle mode is entered: "Circle Mode"

When figure 8 mode is entered: "Figure 8 Mode"

Figure 34 - UI Mode 3 - Circle Mode and Figure 8 Mode

Mode 4: Maze Mode
LEDS: LED4 on, all other LEDs Off

Operation: Prototype is in maze mode.

Audio Feedback: "Maze Mode."

Figure 35 - UI Mode 4 - Maze Mode

5.5.4 Timing

One disadvantage of using a linux operating system based development board over a bare metal board is that it is much more difficult to get accurate timing. The I-PD sample time was set to 10ms. Various methods were tested to determine the best method of getting the 10ms timing. The sample time is a variable of the control equation which does reduce the effect of timing variance however the best solution is to get as accurate timing as possible. The final solution used the monotonic clock and a clock_nanosleep() function to generate the required 10ms. Monitoring the sample time found that this resulted in a variance of around 0.05ms of sample time. The main advantage of using the clock_nanosleep() function is that it allows the use of an absolute sleep. During other timing methods a relative sleep is calculated by subtracting the current time from the last I-PD sample time to compensate for calculation time. The problem with this is that if an external signal stops the sleep process then there can be an error in the actual sleep time. By using an absolute sleep it is assured that the sleep will be until the correct sample time.

5.6 Experimental Results Data Collection

A series of experiments was conducted to ensure that each of the control objectives was achieved. All of the relevant variables were saved to a text file at each I-PD sample time. These text files were then imported into matlab to be analysed. The results data collection method was tested at the same time as the touchscreen and the results are shown in Figure 28. In this test a signature was drawn on the touchscreen and the measured positions were saved to a text file. The text files were imported into matlab and plotted into matlab. The results clearly show that the signature was replicated very well.

Chapter 6 - Experimental Testing Results

6.1 Introduction

This chapter presents the results for the ball and plate system prototype experiments. Each of the control objectives is presented as a separate experiment. A video of the working prototype is available online at http://www.youtube.com/watch?v=YwtRWhdnid8. Figure 6 shows the pattern used for each of the experiments. The printout was placed under the resistive touchscreen so that it was possible to visually determine that the ball was following the specified trajectory or tracking mode.

6.2 Ball Stabilisation

The aim of the ball stabilisation experiment was to stabilise the ball at the centre of the plate. Prior to switching on the controller the plate was positioned at $\theta_y = 5°$ and $\theta_x = 5°$. This meant that at the start of the experiment the ball was positioned against the Perspex buffer on the lower right hand corner of the plate. Figure 36 shows the results of the experiment. The plot shown in blue was from the measured coordinates of the ball as received form the resistive touchscreen. The red plot shows the reference position that was input into the controller. Once the controller was switched on it took 1.75 seconds to stabilise at the centre of the plate in the y-axis and 1.98 seconds to stabilise at the centre of the plate in the x-axis. It was expected that it would take longer in the x-axis because of the rectangular shape of the plate. In the printout shown in Figure 6 the centre of the plate corresponds to the red RMIT logo.

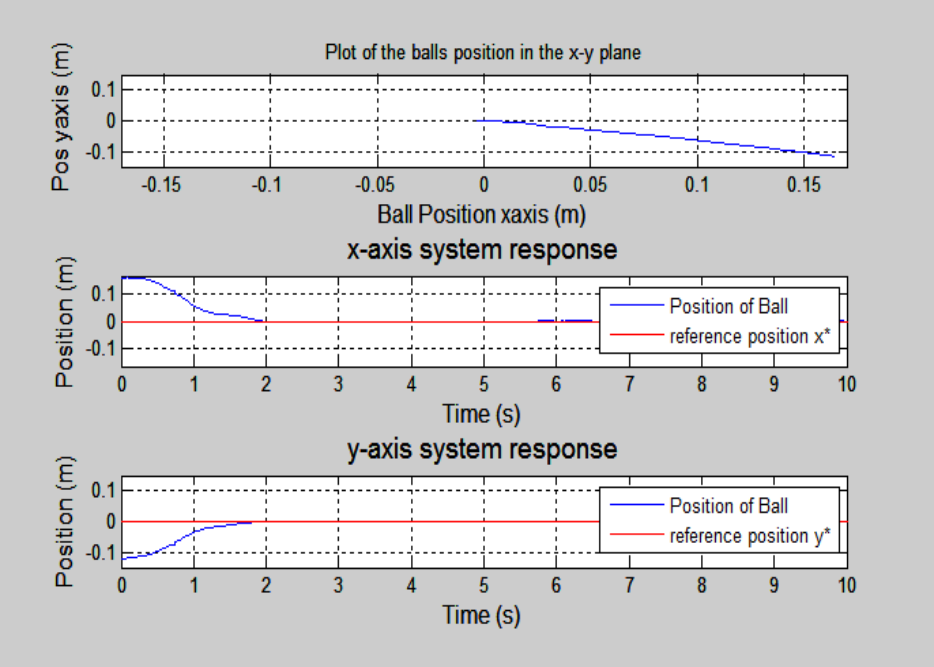

Figure 36 - Ball Stabilisation Experiment

6.3 Disturbance Rejection

In the disturbance rejection experiment the ball is stabilised at the centre of the plate by setting the input positions in the x and y-axis to zero. With the ball stabilised at the centre of the plate an external pulse disturbance was applied to the ball by using a finger to perturb the position of the ball. Figure 37 below shows the results of this experiment. The ball was pushed from its start position at the centre of the plate towards the bottom left hand corner of the plate. The second subplot shows that the ball took 1.39 seconds to return and stabilise at the centre of the plate in the x-axis; and its furthest point from the centre was 5.29cm. The third subplot shows that the ball took 2.4 seconds to stabilise in the y-axis; and it furthest point from the centre was 3.84cm. The y-axis took significantly longer to settle than the x-axis even though it maximum error was smaller. It is also seen that the y-axis had a small amount of oscillations prior to settling. Overall this anomaly was not repeatable so at times the x-axis would settle faster while at other times the y-axis would settle faster. Sometimes the oscillation would occur and other times it wouldn't. The most likely cause of this anomaly was the random time of the balls position measurements. If a new measurement was received too late then the response of the controller could be delayed causing this small variation in the results. There steady state error was measured as zero.

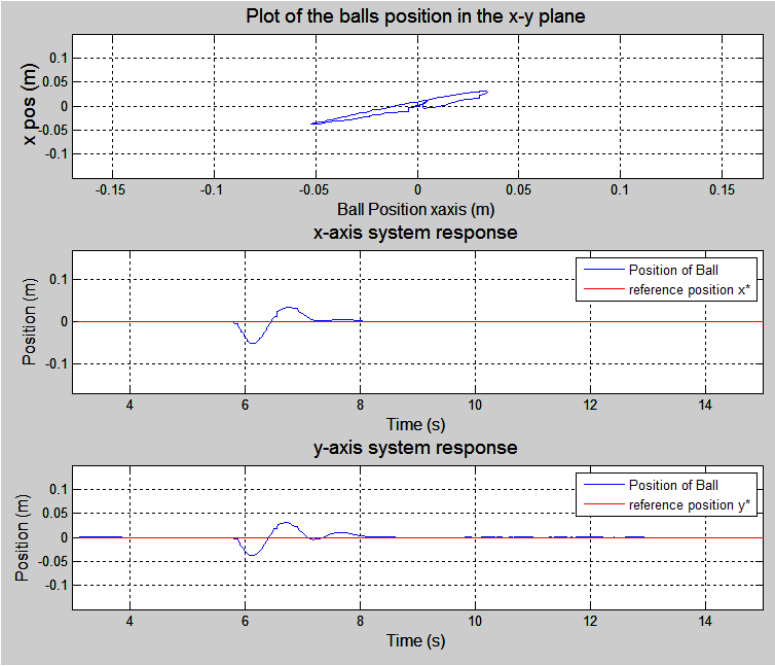

Figure 37 - Disturbance Rejection experiment results

6.4 Circle Tracking

The aim of the circle tracking experiment was for the ball to follow a circular trajectory with radius of 7.5cm at a rate of 2.5 seconds per circle. The ball followed the circle shown on the plate trajectory printout in Figure 6. The input reference position of the ball to achieve the desired trajectory was $R_x = 0.075 \sin\left(\frac{2\pi}{2.5}t + \frac{\pi}{2}\right)$ and $R_y = 0.075 \sin\left(\frac{2\pi}{2.5}\right)$. The red plot shown in Figure 38 indicates the desired trajectory of the ball. As discussed in the controller

design chapter, to achieve the desired trajectory the reference positions of both axis must be pre-compensated to account for the phase and magnitude gains of the system. The resulting trajectory shown in blue indicates that the control scheme was successful in tracking the circle trajectory. The required pre-compensation of the phase and magnitude gains were slightly different to the values calculated in the simulations. The magnitude and phase gains of the system were remeasured from experiment results with no pre-compensation in order to determine the new values. The fact that the phase and magnitude gains were different from the simulation indicates that there is some error in the model used. Some of the possible causes of this model uncertainty are:

- Inaccuracy of the plate angle.
- Delays cause by random measurement sampling rate.
- Inaccurate modelling of the servo.
- Small amount of twisting motion in the plate that was unaccounted for.

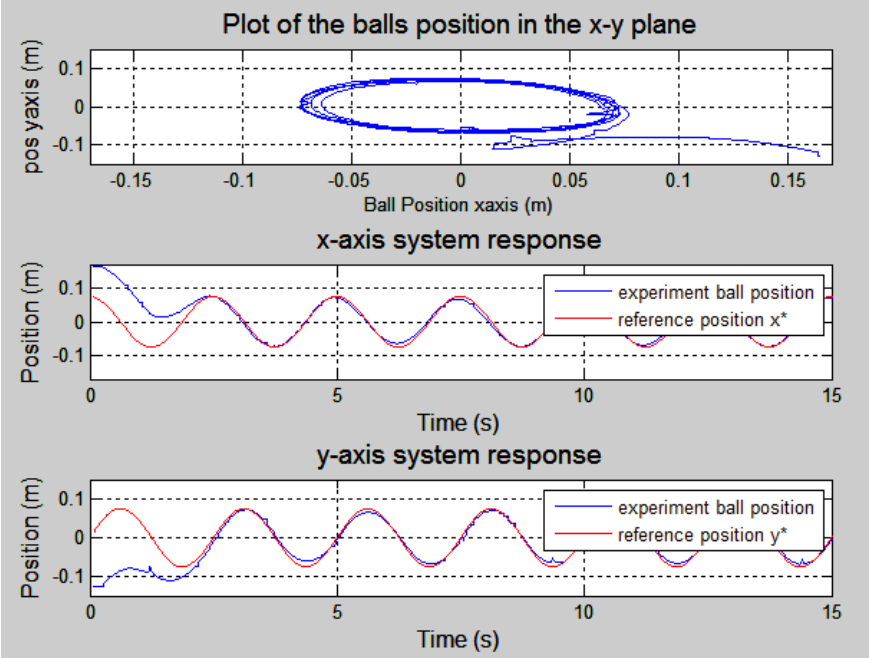

Figure 38 - Circle tracking experiment results

The measured tracking error for the circular trajectory experiment is shown below in Figure 39. It was observed that the trajectory of the ball took 3 seconds to converge with the desired trajectory in both axes. Once the trajectory had converged the maximum tracking error was found to be 1.6cm. The initial specifications stated a minimal tracking error was required. A specification metric was unable to be applied because there was no precedence for this type of controller. The results exceeded initial expectations of the experiment.

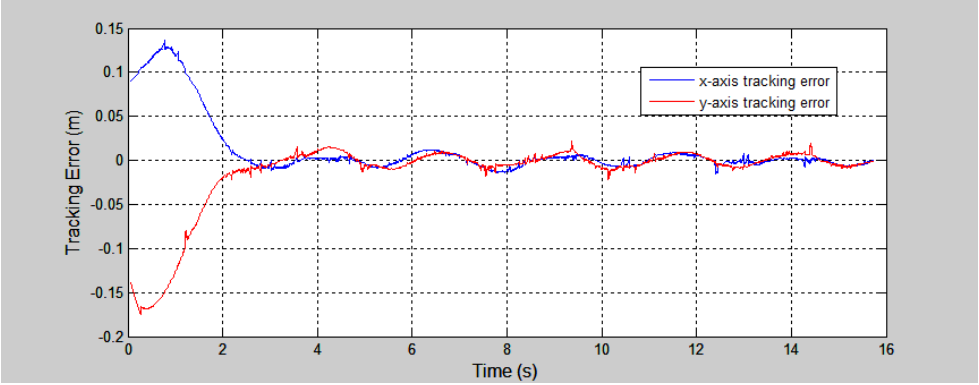

Figure 39 - Tracking error plot for circular trajectory experiment

6.5 Figure 8

The figure 8 experiment is an extension of the circle tracking experiment. The reference input to the controller is again two sinusoids, this time the x-axis sinusoid is half the frequency of the y-axis sinusoid. Figure 40 shows the final results of the figure 8 experiment once pre-compensator values were tuned. The ball was successfully able to track the figure-eight trajectory.

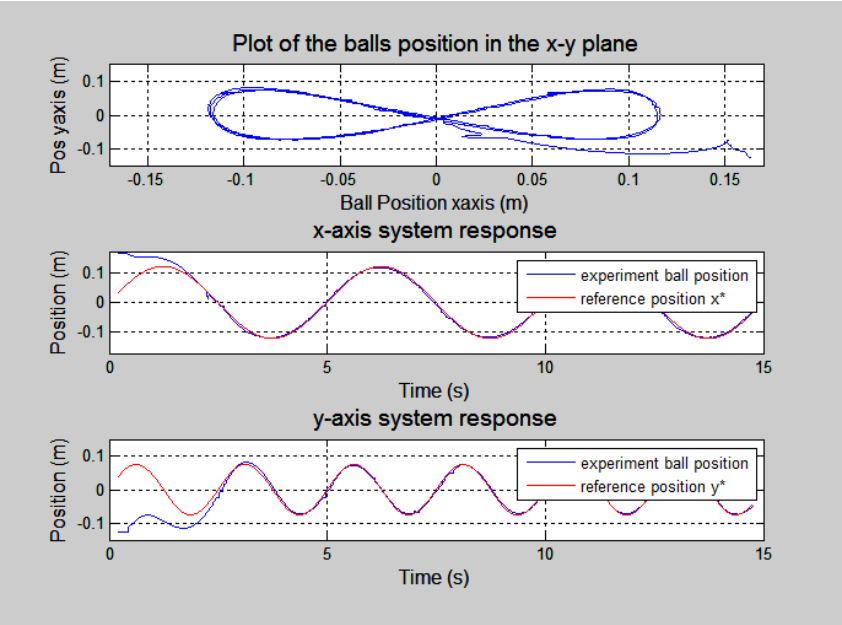

Figure 40 - Figure 8 experiment with pre-compensation of the reference signal

Figure 41 below shows the measured tracking error for the figure-eight experiment. It took 2.84 seconds for the trajectory of the ball to converge with the desire trajectory. Once converged the maximum tracking error was measure at 1.84cm.

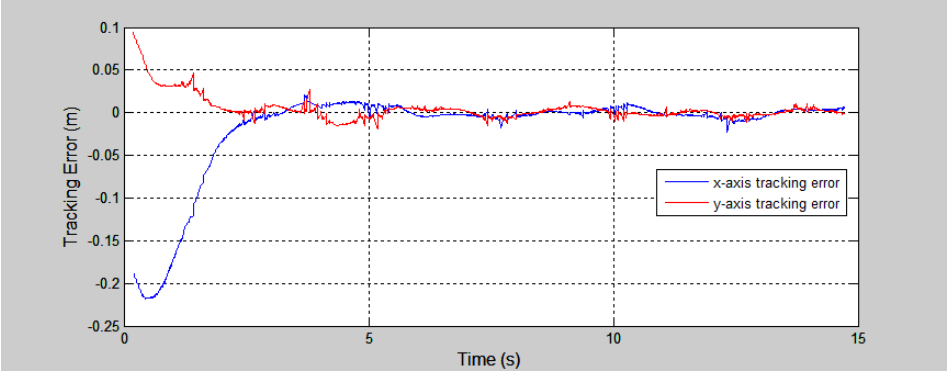

Figure 41 - Figure-eight experiment tracking error

6.6 Square Tracking

The aim of the square tracking experiment was for the ball to follow the square path on the trajectory printout sheet in Figure 42. The square trajectory was achieved by setting the reference position to each of the four corners of the square in turn. This was only possible because the I-PD controller used meant that there was no overshoot that would have resulted in the ball overshooting the corner of the square. The first subplot shows the position of the ball in the x-y plane. The ball started in the top right corner of the plate and initially moved to the bottom left corner of the square. From this position it moved in a clockwise direction around the square with the reference position changing every 2.5 seconds. The ball took approximately 2.3 seconds to stabilise at each corner and was stationary for 0.2 seconds before the reference position was stepped. In the time plots the reference signal is shown in red, the measured ball position is shown in blue and the simulated position of the ball is shown in green. The balls response to a step change of the reference signal is the best way to analyse the performance of the controller and this is why the simulated response is also shown. It is seen that there was a small delay after the reference step before the measured position of the ball started moving. This delay was not present in the simulations and was most likely caused by the measurement delay of the touchscreen. Another possible reason for the delay could be that a small amount of time passes between issuing a servo angle change command to the Micro Maestro servo controller and the change actually occurring. Additionally the delay might not actually exist as the results show only the measured position of the ball as received from the touchscreen and not the actual position. Despite the initial delay the ball is stable at the desired position at the same time as the simulated result. This most likely indicates that the delay which is seen is actually caused by a delay in the measurement and is not actually present. Once the trajectory of the ball converged to the square trajectory the maximum position error was measured at 6.2mm. This was one measurement which showed the error to be 1.53cm however this looked to be a spurious touchscreen reading. This met the design specifications which stated an error of < 10mm.

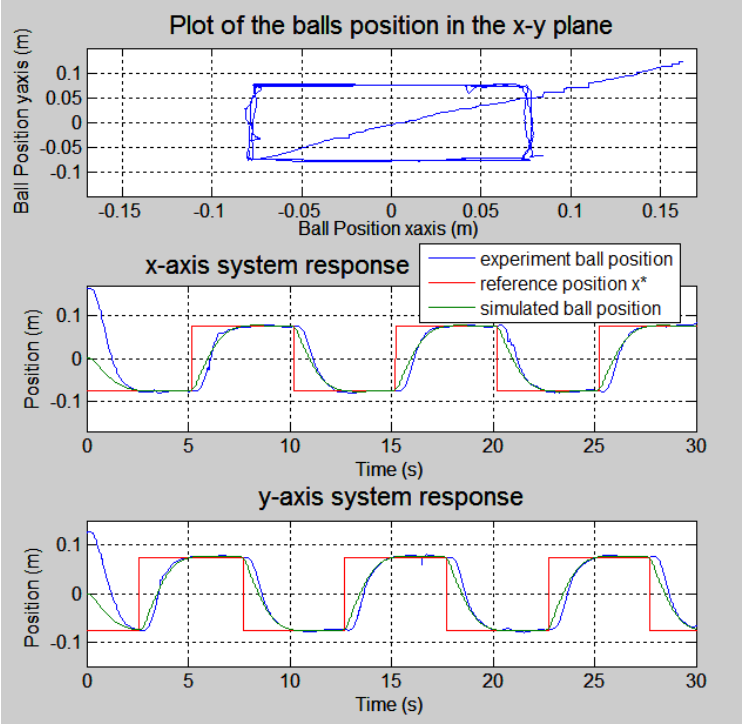

Figure 42 - Square Tracking Experiment Results

6.7 Maze Tracking

The aim of the maze tracking experiment was for the ball to follow the maze shaped path on the outside edges of the trajectory printout sheet in Figure 6. This trajectory was achieved by stepping the reference signal between each corner of the maze path. Once the ball was within 0.5cm of the corner the reference signal was shifted to the next corner. This stepping process is shown below in Figure 44. The control scheme was successful in following the maze path trajectory as can be seen from the measured trajectory in the x-y plot in Figure 43.

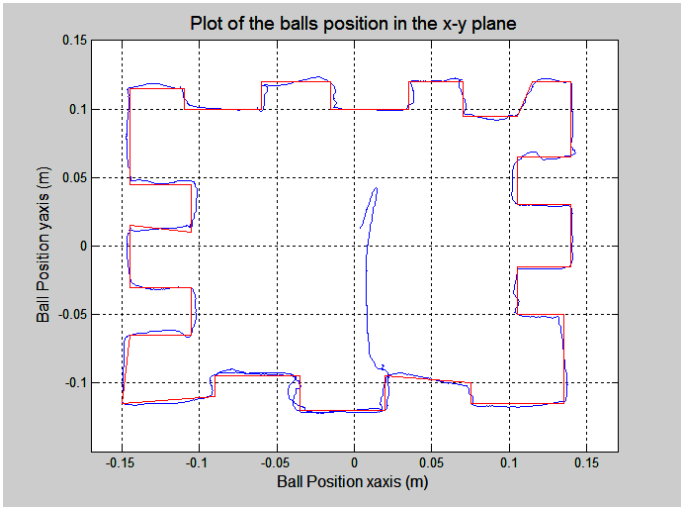

Figure 43 - Maze Tracking Experiment x-y plot

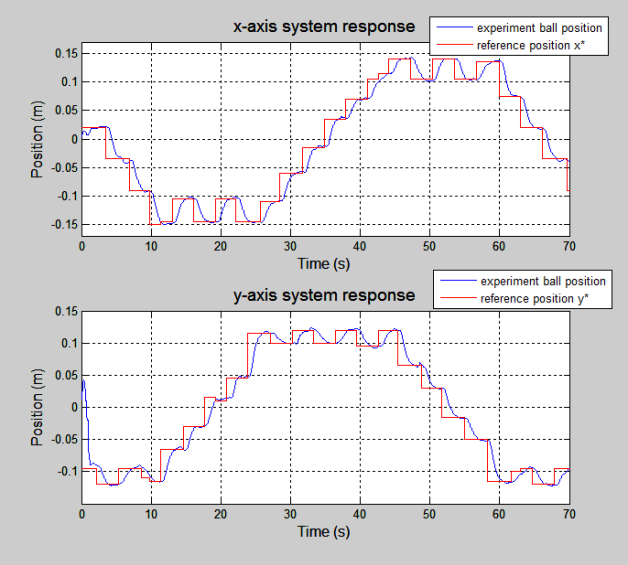

Figure 44 - Maze Tracking Experiment Time plots

6.8 Analysis of Results

Table 2 shows the target metrics as defined by the design specifications in Ch. 1.2. The experiment results showed that each of the stated design specifications was achieved.

Table 2 - Specifications Table

Specification	Metric	Target Value	Measured Value	Achieved?
Stabilisation	Steady State Error	< 5mm	Nil	yes
	Settling Time	< 3 secs	1.98 secs	yes
Disturbance Rejection	Settling Time	< 5 secs	2.4 secs	yes
Reference Step Tracking	Settling Time	< 5 secs	2.3 secs	yes
	Percentage Overshoot	< 1%	negligible	Yes
Square Tracking	Maximum Position Error	< 10mm	6.2mm	yes
	Overshoot	minimal	negligible	yes
Maze Tracking	Tracks Maze Trajectory	yes	yes	yes
Sinusoidal Tracking	Tracking Error	minimal	1.84cm	yes
Circle Tracking	Tracks Circle Trajectory	yes	yes	yes
	Steady State Error	minimal	1.6cm	yes
Figure-eight Tracking	Tracks figure-8 trajectory	yes	yes	yes
	Tracking Error	minimal	1.84cm	yes

Chapter 7 - Conclusion

7.1 Conclusion

This paper presented a design for a ball and plate balancing system prototype. The control objectives were defined as: being able to stabilise the ball at the centre of the plate and reject disturbances to the position of the ball, be able to track a circle trajectory, be able to track a square trajectory, be able to track a figure 8 trajectory and be able to track a predetermined maze shaped trajectory. A cascaded I-PD control design was proposed as the solution to each of these control objectives. Pre-compensation of the input was conducted so that the I-PD controller could effectively track a sinusoidal input. A nonlinear model of the system was constructed in Simulink so that the controller design could be refined and the ideal control parameters determined. The final design was implemented on a BeagleBoard XM development board utilising a resistive touch screen for ball position feedback and two hobby servos to set the angle of the plate. The final prototype and control scheme was able to meet the stated design specifications.

Chapter 8 - References

[1] B. Krishna, S. Gangogpadhyay, and J. George, "Design and Simulation of Gain Scheduling PID Controller for Ball and Beam System," presented at the Internation Conference on Systems, Signal Processing and Electronics Engineering (ICSSEE'2012), Dubai (UAE), 2012.

[2] A. Manan Khan, A. Iqbal Bhatti, D. Sami ud, and Q. Khan, "Static & dynamic sliding mode control of ball and beam system," in *Applied Sciences and Technology (IBCAST), 2012 9th International Bhurban Conference on*, 2012, pp. 32-36.

[3] X. Fan, N. Zhang, and S. Teng, "Trajectory planning and tracking of ball and plate system using hierarchical fuzzy control scheme," *Fuzzy Sets and Systems*, vol. 144, pp. 297-312, 6/1/ 2004.

[4] D. Huida, T. Yantao, and W. Guangbin, "Trajectory tracking control of ball and plate system based on auto-disturbance rejection controller," in *Asian Control Conference, 2009. ASCC 2009. 7th*, 2009, pp. 471-476.

[5] W. Hongrui, T. Yantao, S. Zhen, Z. Xuefei, and D. Ce, "Tracking Control of Ball and Plate System with a Double Feedback Loop Structure," in *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, 2007, pp. 1114-1119.

[6] H. Kyongwon, T. Yantao, K. Yongsu, L. Jinsong, and Z. Yinghui, "Tracking control of ball and plate system using a improved PSO on-line training PID neural network," in *Mechatronics and Automation (ICMA), 2012 International Conference on*, 2012, pp. 2297-2302.

[7] H. Date, M. Sampei, M. Ishikawa, and M. Koga, "Simultaneous control of position and orientation for ball-plate manipulation problem based on time-State control form," *Robotics and Automation, IEEE Transactions on*, vol. 20, pp. 465-480, 2004.

[8] S. Jintao, T. Yantao, and B. Ming, "Research on Trajectory Tracking of Ball-and-Plate System Based on Supervisory Fuzzy Control," in *Control Conference, 2006. CCC 2006. Chinese*, 2006, pp. 1528-1532.

[9] N. Mohajerin, M. B. Menhaj, and A. Doustmohammadi, "A Reinforcement Learning Fuzzy Controller for the Ball and Plate system," in *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, 2010, pp. 1-8.

[10] Z. Fei, L. Xiaoli, W. Shangjun, and D. Dawei, "Position control of ball and plate system based on switching mechanism," in *Automation and Logistics (ICAL), 2011 IEEE International Conference on*, 2011, pp. 233-237.

[11] B. Ming, L. Huiqiu, S. Jintao, and T. Yantao, "Motion Control of Ball and Plate System Using Supervisory Fuzzy Controller," in *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, 2006, pp. 8127-8131.

[12] D. Yuan and Z. Zhang, "Modelling and control scheme of the ball-plate trajectory-tracking pneumatic system with a touch screen and a rotary cylinder," *Control Theory & Applications, IET*, vol. 4, pp. 573-589, 2010.

[13] F. Zheng, X. Qian, X. Li, and S. Wang, "Modeling and PID neural network research for the ball and plate system," in *Electronics, Communications and Control (ICECC), 2011 International Conference on*, 2011, pp. 331-334.

[14] W. Yao, L. Xiaoli, L. Yang, and Z. Baoyong, "Identification of ball and plate system using multiple neural network models," in *System Science and Engineering (ICSSE), 2012 International Conference on*, 2012, pp. 229-233.

[15] D. Y. Gao and R. W. Ogden, *Advances in Mechanics and Mathematics*: Springer, 2003.

[16] S. Awtar, C. Bernard, N. Boklund, A. Master, D. Ueda, and K. Craig, "Mechatronic design of a ball-on-plate balancing system," *Mechatronics,* vol. 12, pp. 217-228, 3// 2002.
[17] A. Knuplez, A. Chowdhury, and R. Svecko, "Modeling and control design for the ball and plate system," in *Industrial Technology, 2003 IEEE International Conference on*, 2003, pp. 1064-1067 Vol.2.
[18] A. Moreno, x, M. A. riz, E. Rubio, Pe, x, *et al.*, "Design and Implementation of a Visual Fuzzy Control in FPGA for the Ball and Plate System," in *Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on*, 2010, pp. 85-90.
[19] L. Hongwei and L. Yanyang, "Trajectory tracking sliding mode control of ball and plate system," in *Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on*, 2010, pp. 142-145.
[20] K. Bruce, Rodriguez, "Four Degrees of Freedom Control System Using Ball and Plate," Southern Polytechnic State University2011.
[21] J. Bruce, C. Keeling, and R. Rodriguez, "Four Degrees of Freedom Control System Using Ball and Plate," Southern Polytechnic State University2011.
[22] D.-h. Yuan, "Pneumatic servo ball & plate system based on touch screen and oscillating cylinder," in *Intelligent Systems and Applications, 2009. ISA 2009. International Workshop on*, 2009, pp. 1-4.
[23] W. G. S. Zhengshun, "Algorithmic Research on PD Direct Fuzzy Control Based on Ball & Plate Apparatus [J]," *Electric Drive,* vol. 4, p. 005, 2004.
[24] G. Andrews, C. Colasuonno, and A. Herrmann, "Ball on Plate Balancing System," ed: Internet, 2004.
[25] R. Downs, "Using resistive touch screens for human/machine interface," *Analog Applications Journal, Texas Instruments,* 2005.
[26] E. T. Solutions. (2013, October). *Compare resistive touch technologies*. Available: http://www.elotouch.com/Technologies/compare_resist.asp
[27] J. H. Park and Y. Jong Lee, "Robust visual servoing for motion control of the ball on a plate," *Mechatronics,* vol. 13, pp. 723-738, 2003.
[28] Y. De-hu, "Pneumatic Servo Ball & Plate System Based on Touch Screen and Oscillating Cylinder," in *Intelligent Systems and Applications, 2009. ISA 2009. International Workshop on*, 2009, pp. 1-4.
[29] M. Nokhbeh and D. Khashabi, "Modelling and Control of Ball-Plate System," Amirkabir University of Technology2011.
[30] M. Moarref, M. Saadat, and G. Vossoughi, "Mechatronic design and position control of a novel ball and plate system," in *Control and Automation, 2008 16th Mediterranean Conference on*, 2008, pp. 1071-1076.
[31] S. W. Sung, J. Lee, and I.-B. Lee, *Process identification and PID control*. Singapore ; Hoboken, N.J.: John Wiley, 2009.
[32] A. Visioli, *Practical PID control*: Springer, 2006.
[33] K. Arshdeep and K. Amrit, "Comparison of Mamdani-Type and Sugeno-Type Fuzzy Inference Systems for Air Conditioning System," *International Journal of Soft Computing & Engineering,* vol. 2, p. 323, 2012.
[34] J. Ragot and M. Lamotte, "Fuzzy logic control," *International Journal of Systems Science,* vol. 24, pp. 1825-1848, 1993/10/01 1993.
[35] R. Johnston, "Fuzzy logic control," *Microelectronics Journal,* vol. 26, pp. 481-495, 7// 1995.
[36] G. Beale, "Phase lead compensator design using bode plots," *Course Literature,* 2008.
[37] R. T. O'Brien, Jr. and J. M. Watkins, "A unified approach for teaching root locus and bode compensator design," vol. 1, ed, pp. 645-649.

[38] L. Wang, "Advanced Control Systems Lecture Notes," RMIT University, Melbourne2013.
[39] S. Clara, "Advances in Industrial Control."
[40] R. C. Dorf and R. H. Bishop, *Modern control systems*, 12th ed., International ed. ed. Boston ; London: Pearson, 2011.
[41] Servocity.com. (2013, September). *HS-7975HB Servo*. Available: http://www.servocity.com/html/hs-7975hb_servo.html
[42] circuitco.com. (2013). *BeagleBoard XM*. Available: http://circuitco.com/support/index.php?title=BeagleBoard-xM
[43] R. D. Knight, *Physics for scientists and engineers : a strategic approach : with modern physics*, 2nd [extended] ed. ed. San Fransisco: Pearson Addison Wesley, 2008.
[44] R. Teodorescu and F. Blaabjerg, "Proportional-resonant controllers. A new breed of controllers suitable for grid-connected voltage-source converters," 2004.

Chapter 9 - List of Appendices

The following appendices are included at http://goo.gl/l1mzn1 or in the attached CDROM:

Appendix A – Matlab Simulations

Appendix B – Project Code

Appendix C – Experiment Results

Appendix D – Experiment Video

Appendix E – Datasheets

Appendix F – Fortnightly Summaries