

*Palestine Polytechnic University
College of Engineering
Mechanical Engineering Department*



Ball and Plate Balancing System

Team:

Anas Qasrawi
Yazeed Natsheh

Supervisors:

Dr . Jasem Tamimi

Submitted to the College of Engineering
in partial fulfillment of the requirements for the
Bachelor degree in Mechatronics Engineering

May 28, 2018

Ball and Plate Balancing System

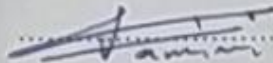
Project Team

Anas Qasrawi

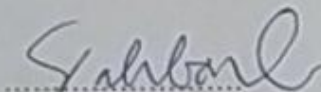
Yazeed Natsheh

Submitted to the College of Engineering in partial fulfilment of the requirements for
Bachelor degree in Mechatronics Engineering

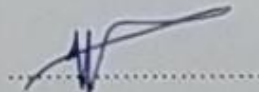
Supervisor Signature



Testing Committee Signature



Chair of the Departure Signature



May 2018

Dedication

We dedicate our research project to our beloved parents for their continuous invaluable support and encouragement all through the years and to our dear siblings for providing us with a comfortable environment for study and research.

Acknowledgement

We would like to express our gratitude to our supervisor, Dr Jasim Tamimi, for his full support and guidance and remarkable suggestions. We would also like to thank our teachers for all the efforts they have exerted to make us qualified engineers who can assume-with confidence-our role in building our community. Thanks are also due to our classmates and friends for their cooperation and encouragement.

We would also like to thank the Deanship of Graduate Studies and Scientific Research at Palestine Polytechnic University for their financial support.

Abstract

This report presents the design and implementation of ball and plate system. The system consists of a plate, a touch screen and servo motors. The Touch screen is placed over the plate and the plate is moving by the servo motors. The objective is to balance a rolling ball in a specific position with the least possible error and smallest settling time achieved for the real system. Linear and non linear mathematical model of the system is derived for simulation purpose. MATLAB is used to evaluate the closed loop system response and to determine the PID parameters. A arduino is used as the controller in which the PID control algorithm is implemented.

Keywords- Ball and Plate, balancing systems, PID controller, touch screen .

المخلص

هذا المشروع يقدم التصميم والتنفيذ لنظام الكرة و اللوحة , هذا النظام يحتوي على لوحة وشاشة لمس ومحركات سيرفو , الشاشة للمسية توضع على اللوحة ويتم تحريك اللوحة بواسطة محركات السيرفو .يهدف هذا المشروع الى تحقيق توازن الكرة الدائرية في مكان معين على اللوحة باقل مقدار من الخطأ و اقل وقت ممكن .التمثيل الخطي للنظام اشتق من اجل ايجاد العلاقة بين مدخلات النظام ومخرجات النظام .برنامج الماتلاب استخدم من اجل تمثيل اساليب التحكم المختلفة و ايجاد استجابة النظام المغلق و من اجل الحصول على متغيرات المتحكم المتحكم التناسبي التكاملي التفاضلي .المتحكم الدقيق اردوينوا استخدم من اجل تحقيق التحكم في النظام وتطبيق المتحكم التناسبي التكاملي التفاضلي (بي اي دي) .

الكلمات المفاتيحة – الكرة واللوحة ,اتزان الانظمة ,المتحكم التناسبي التكاملي التفاضلي(بي اي دي),الشاشات للمسية .

Contents

Abstract	i
List of Figures	vii
List of Tables	x
1 Introduction	1
1.1 Problem definition	1
1.2 The importance and motivation of the project	2
1.3 Project aim and methodology	2
1.4 Literature review	3
1.5 work plane	6
1.6 Budget	8

1.7	Outline	9
2	Mathematical Model	10
2.1	Introduction to modeling	10
2.2	Mathematical Model	12
2.3	Non-linear model	16
2.4	Linearized model	17
2.5	Torque calcalations	18
3	Prototype design	21
3.1	Introduction to prototype design	21
3.2	The structure base	22
3.3	Plate holder of the touch screen	23
3.4	The servo motor holder	24
3.5	The servo motor arm	24
3.6	The linkage threaded rod	25
3.7	Servo rod bearing	26

3.8	Plate holder slider	26
3.9	The central shaft	27
3.10	Univrsal Joint	27
3.11	Prototype design	28
4	Electrical Design	29
4.1	Introduction to electrical design	29
4.2	Resistive touch screen	30
4.3	Servo motor	32
4.4	Microcontroller	33
5	Control Design	34
5.1	Introduction to control design	34
5.2	State space model	35
5.3	Transfer Function	36
5.4	Simulation	37
5.4.1	Non-linear Simulation	38

5.4.2	Result from Non-linear	40
5.4.3	state feedback	40
5.4.4	Optimal control LQR	42
5.4.5	PID	44
6	Experimental Setup	46
6.1	Hardware	46
6.1.1	Prototype	47
6.1.2	Real system response	48
6.2	User Interface	50
6.2.1	Introduction to User interface	50
6.2.2	Graphical user interface	51
7	Conclusions and Future Work	52
7.1	Conclusions	52
7.2	Future Work	53
8	References	54

Appendix A Datasheets	56
Appendix B MATLAB Code	76
Appendix C S-function	79
Appendix D Arduino codes	84
Appendix E visual studio code	100

List of Figures

1.1	Wang ball and plate system	3
1.2	zeeshan ball and plate system	4
1.3	Fan,Xingzhe ball and plate system	5
2.1	schematic diagram of ball and plate	11
3.1	the structure base	22
3.2	plate holder of touch screen	23
3.3	servo motor holder	24
3.4	Servo motor arm	24
3.5	Linkage threaded rod	25
3.6	Servo rod bearing	26

3.7	Plate holder slider	26
3.8	central shaft	27
3.9	Univrsal Joint	27
3.10	Prototype	28
4.1	system wiring	29
4.2	4-wire resistive touch screen	31
4.3	Servo Motor	32
4.4	Microcontroller	33
5.1	Non-linear model	38
5.2	s-function block	38
5.3	X-System response	39
5.4	y-System response	41
5.5	State feedback Model	41
5.6	State feedback Response	42
5.7	Optimal control in Simulink	43

5.8	Optimal controller response in x-axis	43
5.9	Optimal controller response in y-axis	43
5.10	Close loop of the system in x-axis	44
5.11	close loop system in y-axis	44
5.12	x-axis response	45
5.13	y-axis response	45
6.1	Real model	47
6.2	Real model	48
6.3	x-cordinate response	49
6.4	y-cordinate response	49
6.5	Graphical user interface	51

List of Tables

1.1	Work Package List	7
1.2	Cost Table	8

Chapter 1

Introduction

1.1 Problem definition

One of the most challenging problems in control system is balancing a system. Many platforms for this field like double and multiple inverted pendulums, ball-beam system, magnetic levitation , the last platforms is developed based on mechatronics design principles. The challenge here is to balance these systems as desired. Ball and plate system is a promoted version of ball and beam system. The challenge in ball and plate system is balancing the ball on the plate.

The system of the ball and plate is a two-dimensional, non-linear, multivariable as well as unstable system in the open loop system. That is the ball; will run away if the plate is not on the horizontal state unless the plate correct it is angle to the horizontal plane. The ball and plate is a electromechanical system include a rigid plate with a ball rolling freely on the plate, several linear sensors for locating ball position and detecting plate deflection angle, torque generation facilities such as stepping motor, servo motor

1.2 The importance and motivation of the project

The ball and plate system it is a problem under continuous study for applications from robotics to transportation, it is extensions of the ball and beam project. Therefore, the system can present many challenges and opportunities as an educational tool to laboratory experiments with different proceeding of the control.

1.3 Project aim and methodology

The main aim of this project is to design and implement a ball and plate balancing system that can maintain a static ball position on the plate, rejecting position disturbances. The initially horizontal plate will be lean along of two horizontal axes in order to control the position of the ball. Each axis will be operated by an electric motor independently. Each motor will be controlled using control algorithm. The position of the ball on the plate will be measured with resistive touch screen. The scope of this project is to explain the design of the hardware and software in a ball and plate balancing platform, the requirements, the process of operation, as well as the Methods of control implementation. This project is meant to be used to understand principles of functionality and hardware generalities. by using mechatronics approach as in following :

- 1- Choose the subject of the project.
- 2- Collecting sources and references
- 3- Collect the scientific material needed for project
- 4- Physical Systems Modeling.
- 5- 3D drawing of the system.
- 6- sizing of each component sensor, motors and hardware that is needed to use.
- 7- Chose a suitable control system strategy.
- 8- Simulation and optimization result on software.
- 9- Buy the project compount
- 10- Do hand work and lath work
- 11- Building a prototype for the project
- 12- Test the prototype until having desired result

1.4 Literature review

Many studies were presented in the literature about the ball and plate system. These studies differ finite type of controls as well as the way of reading the ball position. However, Some of the recent available researches on the ball and plate system are reviewed in this section..

In the article presented by Wang, Y. in 2014 [1]. A digital camera is placed above the plate to measure the relative position of a copper ball about the center of the plate. The control objective, by rotating the plate, is to stabilize the ball on a specific position or make it go along a certain trajectory. In this paper, active disturbance rejection control (ADRC) strategy is used. (ADRC) algorithm based on the feedback linearization philosophy. ADRC is a disturbance observer based control method and has demonstrated its effectiveness in many fields. Utilized to reject the friction effect at low velocity. A reduced-order disturbance observer is proposed to attenuate the aftereffects caused by the friction on the output of the traditional PD-type control scheme instead of compensating the primary friction mechanics. In other words, a pseudo reference command is produced to counteract the output error during the steady-state phase. Moreover, the describing function technique is used to test whether the proposed method can generate limit cycles or not. Figure 1.1 shows Wang prototype of ball and plate.

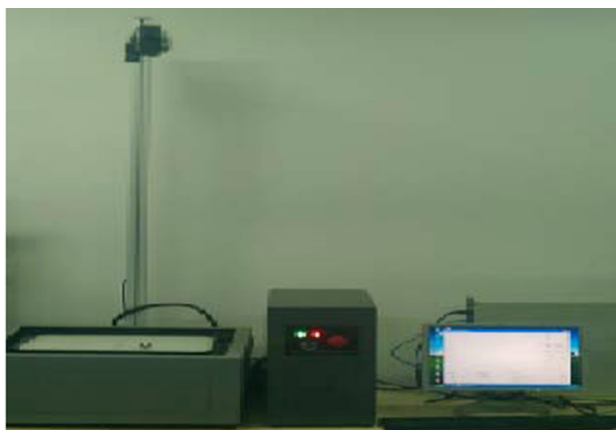


Figure 1.1: Wang ball and plate system

In the article presented by Zeeshan, A. in 2012 [2]. The position of the ball is being measured by using a grid of 11 x 11 phototransistors, the distance between consecutive sensors being 1 inch. Each phototransistor is provided with a monochromatic beam of laser light. Whenever the ball passes in front of a particular phototransistor, its supply of light is interrupted and the voltage level varies. This variation is interpreted by the controller corresponding to each axes as x and y coordinates.. In order to control the position and velocity of the motors, PID algorithm is applied on each motor. The gains of Proportional, integral and derivative compensators were tuned separately. Figure 1.2 shows Zeeshan prototype of ball and plate system.

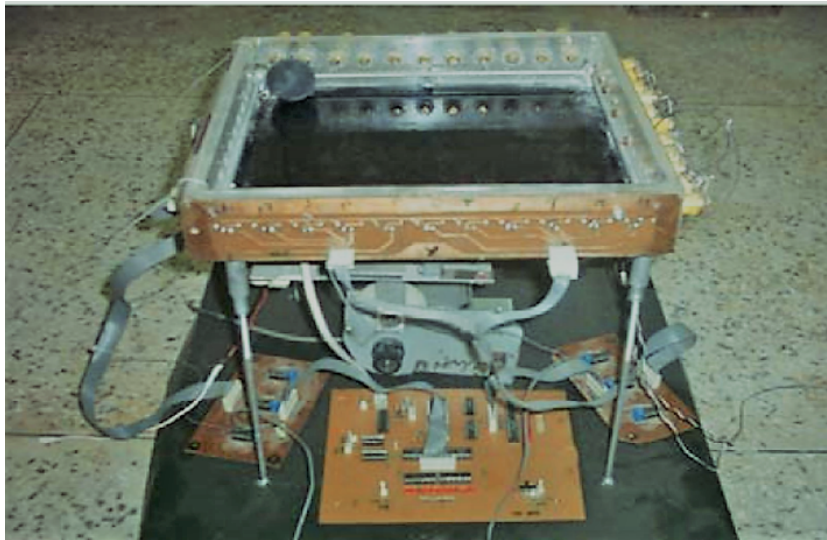


Figure 1.2: zeeshan ball and plate system

In the article presented by Fan, Xingzhe. in 2004 [3]. A digital CCD camera is used in this system to measure the ball position. For the control algorithm they used hierarchical fuzzy control scheme, which is composed of three levels. The lowest level is a TS type fuzzy tracking controller; the middle level is a fuzzy supervision controller that takes actions in extreme situations; and the top level is a fuzzy planning controller that determines the desired trajectory. In order to optimize the ball's moving trajectory. Figure 1.3 shows Fan, Xingzhe prototype of ball and plate system.



Figure 1.3: Fan,Xingzhe ball and plate system

1.5 work plane

The project Schedule outlines the tasks and activities of the project; the duration; start and end dates for each individual task and the project as a whole; and the resources and effort required. Work packages where an activity duration can be reliably estimated and managed.

A work package details a level of work to be completed. It would contain a description of the work, details of constraints, and agreement between the PM and the team or individual doing the work, that the work can be done within the constraints. Assumptions would be defined in the work package depending on the needs of the project. In this project we used work package to develop the project Schedule for the two semester introduction of graduation project and graduation project as shown below in table 1.1.

In the Work packages 1: Identification of Project Idea. we begin with identify the problem in stabilizing the system and how it related to our project and the solution that our system offer. In the Work packages 2: Main Design Concepts and 3D drawings. we start to select our own design for the project. By using drawing software we build our prototype. In Work packages 3: Derive the mathematical model and simulation. We use our math knowledge to obtain a mathematical model for the system to be controlled which is a closest approximation of its true behavior. In Work packages 4: Mechanical Design and Electrical Design. we begin to define mechanical system parts and physical structure which consists of the plate, joints, linkage... etc. and electrical system components consist of the microcontroller, touch screen, motors. In Work packages 5: choose a best control method. We begin to design the control strategy. In WP6: Define the required equipment. In this WP we choose the equipment we are going to use in our project. In WP7: Buy the equipment. After we define the required equipment we start to look for the best deal to buy the equipment. In WP7: Build the prototype after we buy the necessary equipment we begin to assemble these equipment to build the project prototype. In WP9: Testing the prototype. we begin to test our prototype in different ways to find the best working way. In WP10: Compare simulation result with the real result. After the prototype work correctly we begin to collect the result and compare it to the result we have from simulation. In WP11: Submit graduation book to department of mechanical engineering. The final step in this project is submit it to department of mechanical engineering.

Table 1.1: Work Package List

Work package number	Work Package Title	Work package type	Start Work	End Work	Duration of the Work Package(Week)
WP1	Identification of Project Idea and calculate needed Information	Search	1	2	2
WP2	Main Design Concepts and 3D Drawings	Software Work	3	4	2
WP3	Mathematical Model	Hand Work	4	6	3
WP4	Mechanical Design and Electrical Design	Design Work	7	9	3
WP5	Control Design	Design Work	10	14	5
WP6	Define the Require Equipment	Search	15	16	2
WP7	Buy the Equipment	Buying	17	18	2
WP8	Build the Prototype	Hand Work	19	23	5
WP9	Testing the Prototype	Testing	24	32	9
WP10	Compare Simulation Result with the Real Result	Software	33	33	1
WP11	Submit Graduation Book to Department of Mechanical Engineering	Reporting	34	35	2

1.6 Budget

The estimated cost of the project is around 2500 NIS , the Table 1.2 shows the project component with there prices .

Table 1.2: Cost Table

Item Name	Quantity	Cost of each Item (NIS)	Total Cost (NIS)	Source
Microcontroller	1	80	80	Market
Servo Motor	2	120	240	Market
Servo Motor rod	2	50	100	Market
Servo arm	2	30	60	Market
Universal Joint	1	50	50	Market
Resistive touch screen	1	500	500	Market
Lath Work :- Central Shaft Servo Motor Base Plate Holder The Structure Base		2500	2500	
Total			3530 NIS	

1.7 Outline

The remainder is arranged as follows: Chapter 2 describes the Mathematical Model of the ball and plate system. Chapter 3 present a prototype design and 3D drawing of the project. Chapter 4 contains details about electrical components. Chapter 5 proposes a control algorithm, presents a set of containability experiments using this algorithm and compares the results with theoretical predictions. Chapter 6 is a discussion of the experimental apparatus used and the communication constraints that govern its operation. Chapter 7, presents the conclusions of the project and ideas for future work

Chapter 2

Mathematical Model

2.1 Introduction to modeling

In order to design a controller that will meet the required performance of the system, it is important to obtain a mathematical model for the system to be controlled which is a closest approximation of its true behavior. The system can then be analyzed, and controlled.

In this chapter we present a mathematical model for the ball and plate balancing system. In order to control the system approximate to its true behavior. The equation of motion for any system can be deriving either by Newton laws or Lagrange's equation. In the following section we derive the motion equation of the system based on the Lagrange's-Euler method, that is equivalent to Newton's laws of motion, but it has the advantage that it takes the same form in any system of generalized coordinates, and it is better suited to generalizations. The freebody digram of ball and plate is shown in 5.11

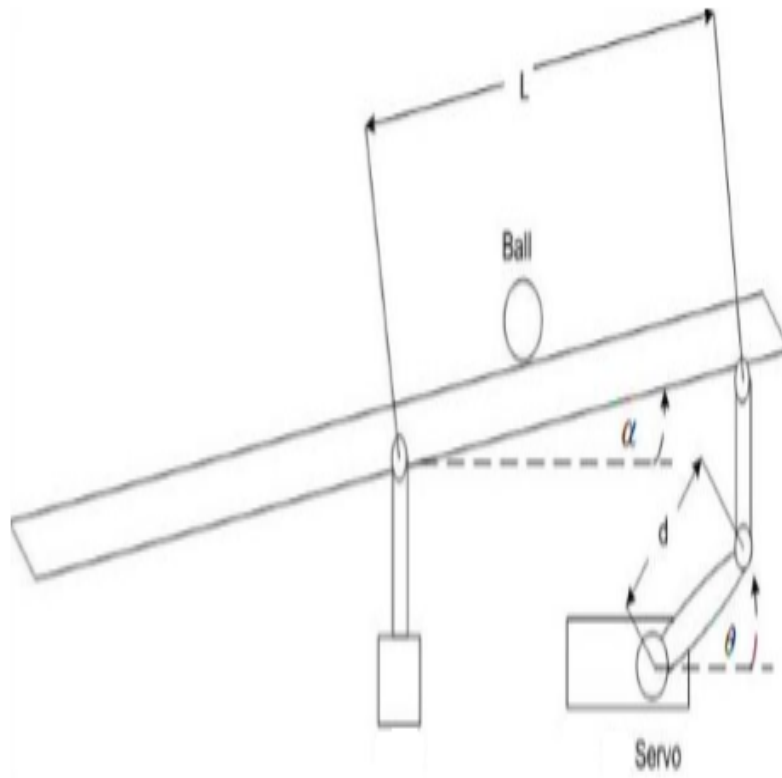


Figure 2.1: schematic diagram of ball and plate

2.2 Mathematical Model

The Euler-Lagrange equation of ball and plate system is as followings :

$$\frac{d}{dt} \frac{\partial T}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} + \frac{\partial v}{\partial q_i} = Q_i \quad (2-1)$$

Where q_i stands for i-direction coordinate, T is Kinetic energy of the system, V is potential energy of system and Q is composite force . The system has 4 degree of freedom ; two in ball motion and two in inclination of plate . Assume the generalized coordinates of system to be (xb and yb) ball's positions on plate and (α and β) the inclination of the plate .It is important to assume the center of x-y coordinates be at center of plate . The Kinetic energy of ball consists of its both rotational with respect to its center of mass and translational energy :

$$T_b = \frac{1}{2} m_b (\dot{x}_b^2 + \dot{y}_b^2) + \frac{1}{2} I_b (w_x^2 + w_y^2) \quad (2-2)$$

Where m_b is mass of the ball and I_b is moment of inertia of the ball; \dot{x}_b and \dot{y}_b are ball's translational velocities along x-axis and y-axis, w_x and w_y are ball' rotational velocities along x-axis and y-axis. The following relations between translational velocities and rotational velocities:

$$\dot{x}_b = r_b w_y \quad , \quad \dot{y}_b = r_b w_x \quad (2-3)$$

In which r_b denotes ball's radius. By substituting equations 3 into equations 4 :

$$T_b = \frac{1}{2} \left(m_b (\dot{x}_b^2 + \dot{y}_b^2) + \frac{I_b}{r_b^2} (\dot{x}_b^2 + \dot{y}_b^2) \right) = \frac{1}{2} \left(m_b + \frac{I_b}{r_b^2} \right) (\dot{x}_b^2 + \dot{y}_b^2) \quad (2-4)$$

The kinetic energy of the plate, by considering ball as a point mass which is placed in (xb, yb), , consists of its rotational energy with respect to its center of mass:

$$T_P = \frac{1}{2}(I_P + I_b)(\dot{\alpha}^2 + \dot{\beta}^2) + \frac{1}{2}m_b(x_b\dot{\alpha} + y_b\dot{\beta})^2$$

$$T_P = \frac{1}{2}(I_P + I_b)(\dot{\alpha}^2 + \dot{\beta}^2) + \frac{1}{2}m_b(x_b^2\dot{\alpha}^2 + 2x_b\dot{\alpha}y_b\dot{\beta} + y_b^2\dot{\beta}^2) \quad (2-5)$$

Where α and β are plate's angle of inclination along x-axis and y-axis, respectively. Therefore $\dot{\alpha}$ and $\dot{\beta}$ are plate's rotational velocity. Here we can calculate the kinetic energy of system as followings:

$$T = T_b + T_P$$

$$T = \frac{1}{2}\left(m_b + \frac{I_b}{r_b^2}\right)(\dot{x}_b^2 + \dot{y}_b^2) + \frac{1}{2}(I_P + I_b)(\dot{\alpha}^2 + \dot{\beta}^2) + \frac{1}{2}m_b(x_b^2\dot{\alpha}^2 + 2x_b\dot{\alpha}y_b\dot{\beta} + y_b^2\dot{\beta}^2) \quad (2-6)$$

The potential energy of the ball relative (the relative potential) to horizontal plane in the center of the inclined plate can be calculated as:

$$V_b = m_bgh = m_bg(x_b \sin \alpha + y_b \sin \beta) \quad (2-7)$$

Here we can derive the system's equation by Lagrangian and equations:

$$L = T_b + T_P - V_b \quad (2-8)$$

$$\frac{\partial T}{\partial \dot{\alpha}} = (I_P + I_b)\dot{\alpha}_x + m_b x_b (x_b \dot{\alpha} + y_b \dot{\beta}) , \quad \frac{\partial L}{\partial \alpha} = mg \cos \alpha \quad (2-9)$$

$$\frac{\partial T}{\partial \dot{\beta}} = (I_P + I_b)\dot{\beta}_x + m_b y_b (y_b \dot{\beta} + x_b \dot{\alpha}) , \quad \frac{\partial L}{\partial \beta} = mg \cos \beta \quad (2-10)$$

$$\frac{\partial T}{\partial \dot{x}_b} = \left(m_b + \frac{I_b}{r_b^2}\right)\dot{x}_b , \quad \frac{\partial L}{\partial x_b} = m_b(x_b \dot{\alpha} + y_b \dot{\beta})\dot{\alpha} \quad (2-11)$$

$$\frac{\partial T}{\partial \dot{y}_b} = \left(m_b + \frac{I_b}{r_b^2} \right) \dot{y}_b, \quad \frac{\partial L}{\partial y_b} = m_b (x_b \dot{\alpha} + y_b \dot{\beta}) \dot{\beta} \quad (2-12)$$

Assume generalized toques as τ_x and τ_y which are exerted torques on the plate. From Lagrange's Euler equation we can write:

$$\begin{aligned} \frac{d}{dt} \frac{\partial T}{\partial \dot{\alpha}} - \frac{\partial L}{\partial \alpha} &= (I_P + I_b) \ddot{\alpha} + m_b x_b^2 \ddot{\alpha} + 2m_b x_b \dot{x}_b \dot{\alpha} + m_b x_b y_b \ddot{\alpha} \\ &+ m_b \dot{x}_b y_b \dot{\beta} + m_b x_b \dot{y}_b \dot{\beta} - m_b g \cos \alpha = \tau_x \end{aligned} \quad (2-13)$$

$$\begin{aligned} \frac{d}{dt} \frac{\partial T}{\partial \dot{\beta}} - \frac{\partial L}{\partial \beta} &= (I_P + I_b) \ddot{\beta} + m_b y_b^2 \ddot{\beta} + 2m_b y_b \dot{y}_b \dot{\beta} + m_b x_b y_b \ddot{\beta} \\ &+ m_b \dot{y}_b x_b \dot{\alpha} + m_b y_b \dot{x}_b \dot{\alpha} - m_b g \cos \beta = \tau_y \end{aligned} \quad (2-14)$$

$$\frac{d}{dt} \frac{\partial T}{\partial \dot{x}_b} - \frac{\partial L}{\partial x_b} = \left(m_b + \frac{I_b}{r_b^2} \right) \ddot{x}_b - m_b (x_b \dot{\alpha} + y_b \dot{\beta}) \dot{\alpha} + m_b g \sin \alpha = 0 \quad (2-15)$$

$$\frac{d}{dt} \frac{\partial T}{\partial \dot{y}_b} - \frac{\partial L}{\partial y_b} = \left(m_b + \frac{I_b}{r_b^2} \right) \ddot{y}_b - m_b (y_b \dot{\beta} + x_b \dot{\alpha}) \dot{\alpha} + m_b g \sin \beta = 0 \quad (2-16)$$

Substituting this expression into (1) and evaluating the derivatives and rearranging terms, we obtain four differential equations as follows:-

$$\left(m_b + \frac{I_b}{r_b^2} \right) \ddot{x}_b - m_b (x_b \dot{\alpha}^2 + y_b \dot{\alpha} \dot{\beta}) + m_b g \sin \alpha = 0 \quad (2-17)$$

$$\left(m_b + \frac{I_b}{r_b^2}\right)\ddot{y}_b - m_b \left(y_b \dot{\beta}^2 + x_b \dot{\alpha} \dot{\beta}\right) + m_b g \sin \beta = 0 \quad (2-18)$$

$$\begin{aligned} \tau_x = & (I_{Px} + I_b + m_b x_b^2) \ddot{\alpha} + 2m_b x_b \dot{x}_b \dot{\alpha} + m_b x_b y_b \ddot{\beta} \\ & + m_b \dot{y}_b x_b \dot{\beta} + m_b y_b \dot{x}_b \dot{\beta} + m_b g x_b \cos \alpha \end{aligned} \quad (2-19)$$

$$\begin{aligned} \tau_y = & (I_{Py} + I_b + m_b y_b^2) \ddot{\beta} + 2m_b y_b \dot{y}_b \dot{\beta} + m_b x_b y_b \ddot{\alpha} \\ & + m_b \dot{y}_b x_b \dot{\alpha} + m_b y_b \dot{x}_b \dot{\alpha} + m_b g y_b \cos \beta \end{aligned} \quad (2-20)$$

In these equations, $m(\text{kg})$ is the mass of the ball; $R(\text{m})$ is the radius of the ball; x (m), \dot{x} (m/s), and \ddot{x} (m/s²) are the ball's position, velocity and acceleration respectively along X-axis; y (m), \dot{y} (m/s), and \ddot{y} (m/s²) are the ball's position, velocity and acceleration respectively along Y-axis; θ_x (rad), $\dot{\theta}_x$ (rad/sec) are respectively the plate's deflection angle and angular velocity about X-axis. θ_y (rad), $\dot{\theta}_y$ (rad/sec) are respectively the plate's deflection angle and angular velocity about Y-axis; $\tau_{\theta x}$ (Nm) and $\tau_{\theta y}$ (Nm) are respectively the torque exerted on the plate along X-axis and Y-axis [4].

Non-linear model

2.3 Non-linear model

From equations (2-17) ,(2-18),(2-19) and (2-20) by considering theses Equations. As mentioned τ_x and τ_y are toques exerted on the plane or on the ball from the motor in y-axis and x-axis direction respectively. By defining state variables $= [x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ x_8] = [x_b, \dot{x}_b, \alpha, \dot{\alpha}, y_b, \dot{y}_b, \beta, \dot{\beta}]^T, U = [u_x, u_y]^T = [\alpha, \beta]^T$

$$x_1 = x$$

$$x_2 = \dot{x}$$

$$x_3 = \alpha$$

$$x_4 = \dot{\alpha}$$

$$x_5 = y$$

$$x_6 = \dot{y}$$

$$x_7 = \beta$$

$$x_8 = \dot{\beta}$$

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \frac{5}{7} (x \dot{\alpha}^2 - g \sin \alpha) = \frac{5}{7} (x_1 x_4^2 - g \sin x_3)$$

$$\dot{x}_3 = x_4$$

$$\dot{x}_4 = v_x$$

$$\dot{x}_5 = x_6$$

$$\dot{x}_6 = \frac{5}{7} (y \dot{\beta}^2 - g \sin \beta) = \frac{5}{7} (x_5 x_8^2 - g \sin x_7)$$

$$\dot{x}_7 = x_8$$

$$\dot{x}_8 = v_y$$

2.4 Linearized model

If one assumes that the angle do not change much, $\pm 30^\circ$, the sine function can be replaced by its argument [5]. In equations (3) and (4), the velocities $\dot{\theta}_x$ and $\dot{\theta}_y$ are small and hence have negligible effects when squared or multiplied together [6],[7]. Furthermore, the system inputs are considered as θ_x and θ_y and not the torque moments $\tau\theta_x$ and $\tau\theta_y$. Due to this reason, (5) and (6) could be dropped out in the consecutive study of the system. As a result of these assumptions, linearized, simplified and uncoupled ordinary differential equations are obtained. Substituting for the moment of inertia of the ball, $J = 25 (mR^2)$ we get:

$$\frac{7}{5}\ddot{x} = g\theta_x$$

$$\frac{7}{5}\ddot{y} = g\theta_y$$

2.5 Torque calculations

$$F_{m1} \frac{(2*10^{-2})}{2*10^{-2}} = \frac{\tau_{m1}}{2*10^{-2}} \quad (2-21)$$

$$F_{m1} \frac{(Lx)}{2} = \tau_x \quad (2-22)$$

$$\tau_x = \tau_{m1} \frac{19}{2*2} \quad (2-23)$$

$$\tau_x = 4.75 * \tau_{m1} \quad (2-24)$$

$$\tau_y = \tau_{m2} \frac{24.8}{2*2} \quad (2-25)$$

$$\tau_y = 6.2 * \tau_{m2} \quad (2-26)$$

You can see in figure below the diminution of our plate and diminution the motor arm

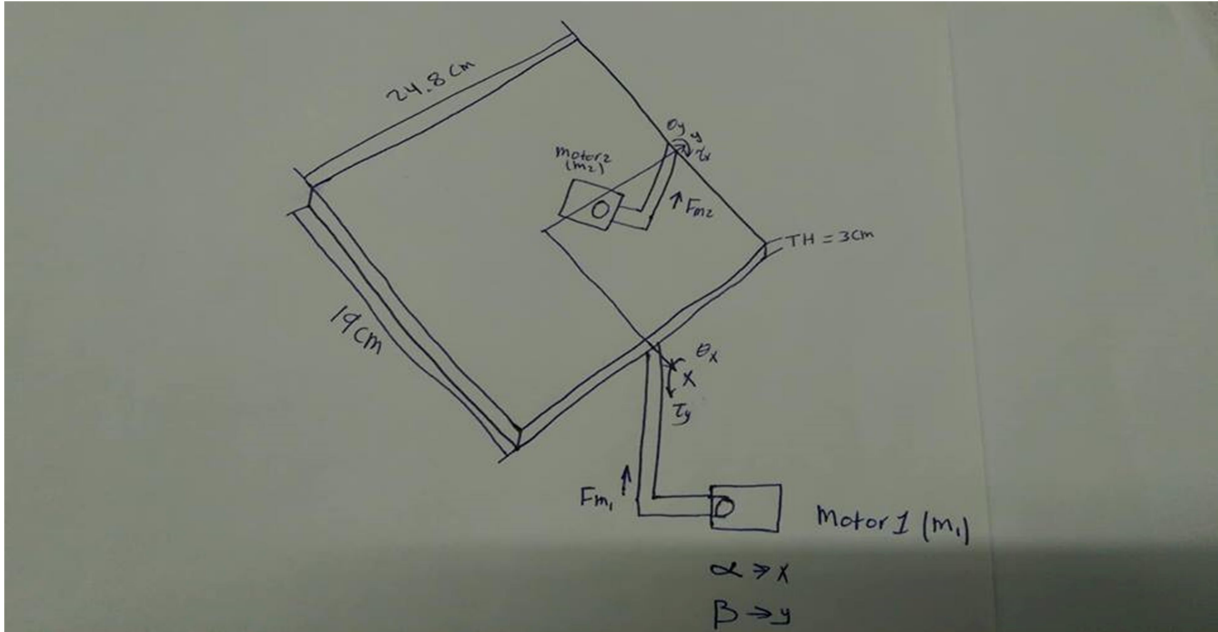


Figure 5.2 plate diminutions

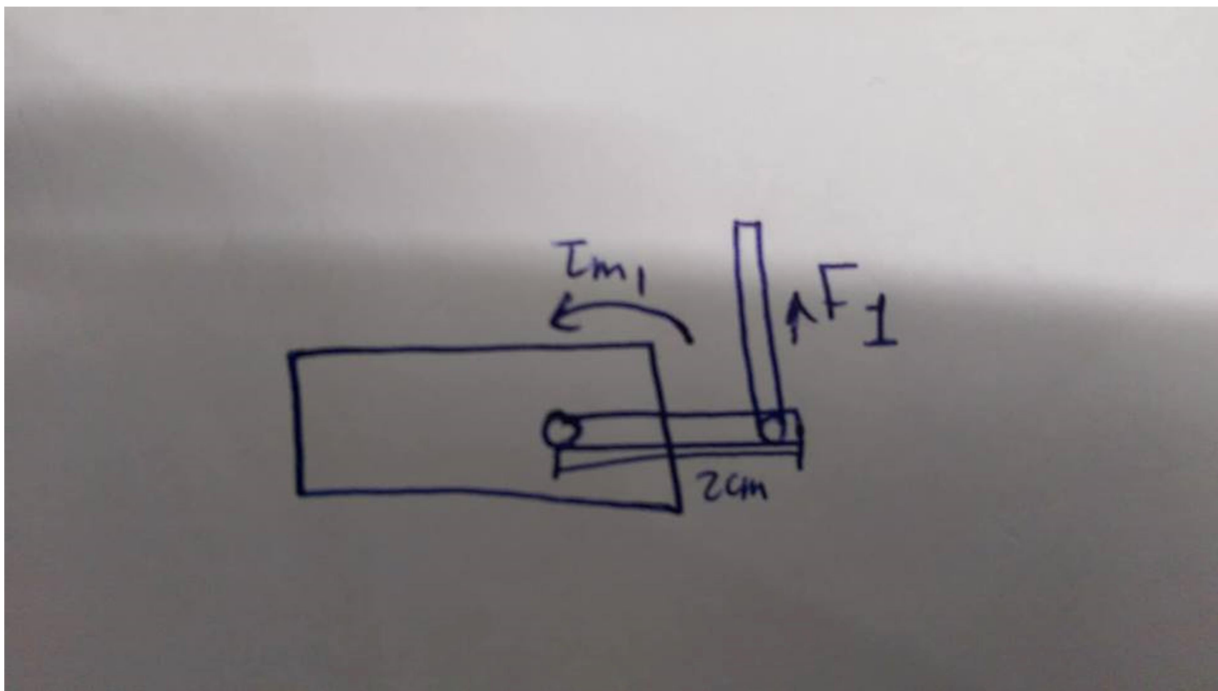


Figure 5.3 motor arm diminutions

By submitte these calculations into torque equation we get

$$\tau_{m1} = \frac{(I_{Px} + I_b + m_b x_b^2) \ddot{\alpha} + 2m_b x_b \dot{x}_b \dot{\alpha} + m_b x_b y_b \ddot{\beta} + m_b \dot{y}_b x_b \dot{\beta} + m_b y_b \dot{x}_b \dot{\beta} + m_b g x_b \cos \alpha}{4.75}$$

$$\tau_{m2} = \frac{(I_{Py} + I_b + m_b y_b^2) \ddot{\beta} + 2m_b y_b \dot{y}_b \dot{\beta} + m_b x_b y_b \ddot{\alpha} + m_b \dot{y}_b x_b \dot{\alpha} + m_b y_b \dot{x}_b \dot{\alpha} + m_b g y_b \cos \beta}{6.2}$$

Chapter 3

Prototype design

3.1 Introduction to prototype design

In this chapter a three-dimensional structure is presented. In this structure a set of requirements is determined. After we determined requirements and generate some solutions by studying similar projects. We use the knowledge of mechanical and Computer-aided design (CAD) software to specify design concepts that meet project requirements. We use Catia v5 software to draw the parts and Solidworks software to assemble the parts and produce the final product. SolidThinking software was used to process the project picture. Dimensions of all the parts mentioned in the text above the figure.

3.2 The structure base

The structure base carries all the other components of the system, servo motors, servo motors linking, servo motors arm, center shaft, plate hold of touch screen, and touch screen. Figure 3.1 show a 3D drawing to the structure base. The structure base made from aluminum with 300mm length and 250mm width and 9mm thickness.



Figure 3.1: the structure base

3.3 Plate holder of the touch screen

The plate holder made from acrylic the acrylic was chose to make it light as passiole as we can .It should be wider that the resistive touch screen with 3mm.Figure3.2 shows a 3D drawing of the plate holder.plate holder dimension is 240mm length and 182mm

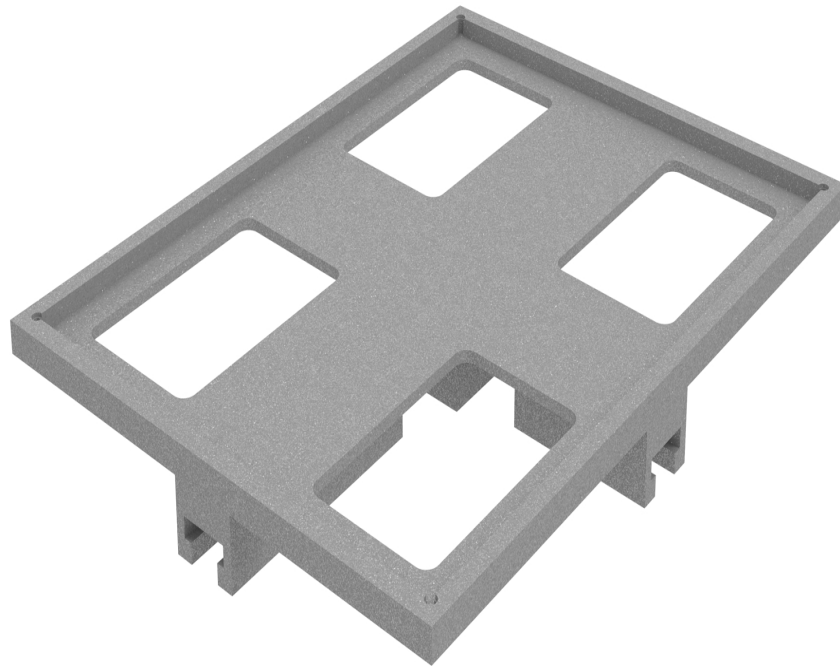


Figure 3.2: plate holder of touch screen

3.4 The servo motor holder

In this project we need two servo motor. So we have to design two servo motor holders. The holders will be fits in the main base, and it will joint tighter by nuts and screw. Figure 3.3 shows a 3D drawing of the servo base.

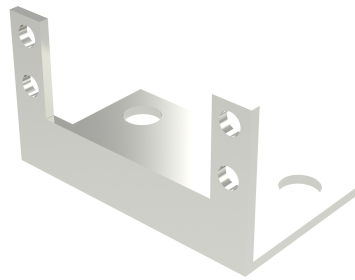


Figure 3.3: servo motor holder

3.5 The servo motor arm

Servo motor arm is use to link between servo motor and threaded rod. 3.4 shows a 3D drawing of the servo motor arm.



Figure 3.4: Servo motor arm

3.6 The linkage threaded rod

It is used to convert rotation motion from servo motor to linear motion in the touch screen. In this project we have two servo motor so we need two linkage. Figure 3.5 shows a 3D drawing of The linkage threaded rod. The linkage threaded rod dimension is 110mm length and 5mm diameter



Figure 3.5: Linkage threaded rod

3.7 Servo rod bearing

small bearing are used between the servo rod and plate holder to reduce twist of the plate holder during the moving of the holder. Figure 3.6 shows 3D drawing of servo rod bearing.



Figure 3.6: Servo rod bearing

3.8 Plate holder slider

this slider used to connect the servo motor rod to plate holder and you can move this slider in x-axis and y-axis. Figure 3.7 shows 3D drawing of servo rod bearing.



Figure 3.7: Plate holder slider

3.9 The central shaft

The central shaft used between structure base and middle of touch screen, it helps to hold the touch screen. Figure 3.8 shows a 3D drawing of the central shaft. The central shaft dimension is 120mm length and 10mm



Figure 3.8: central shaft

3.10 Univrsl Joint

the Univrsl joint used to reduce the twisted of the touch screen and to allow the touch screen to move freely in spce. Figure 3.9 shows 3D drawing of Univrsl joint



Figure 3.9: Univrsl Joint

3.11 Prototype design

Figure 3.10 show a complete system component. Draw usign catia v5 software and assembly using solidworks and the final picture taken by solidthinking software.

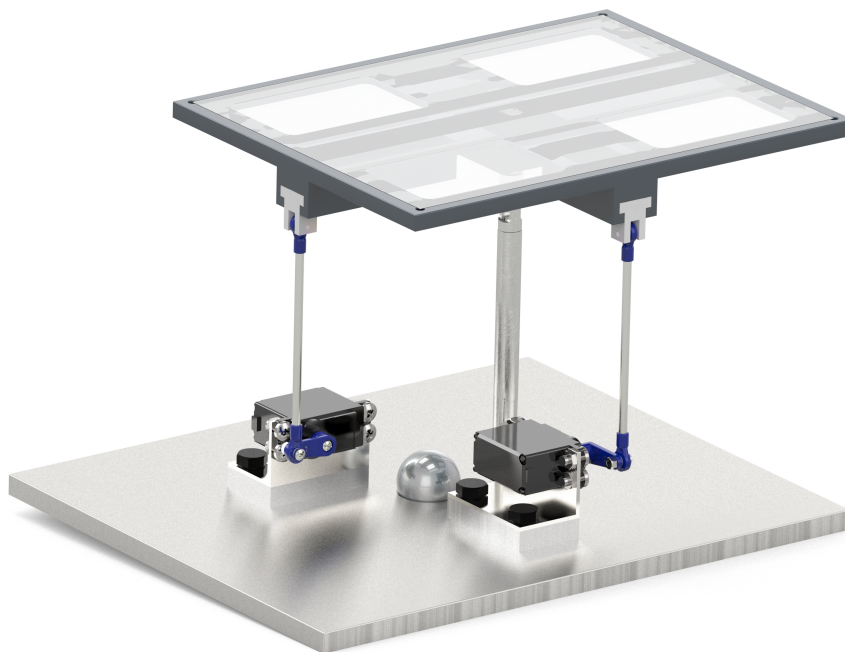


Figure 3.10: Prototype

Chapter 4

Electrical Design

4.1 Introduction to electrical design

This chapter contains details about electrical components that used in the project. And the electrical hardware that used in the project. The electrical circuit diagram is shown in The touch screen is connected to Arduino which is represented by pins A0-A4. The servo Motor X is connected to digital pin, and the servo Motor Y is connected to digital pin. 4.1 show wiring diagram of the project .

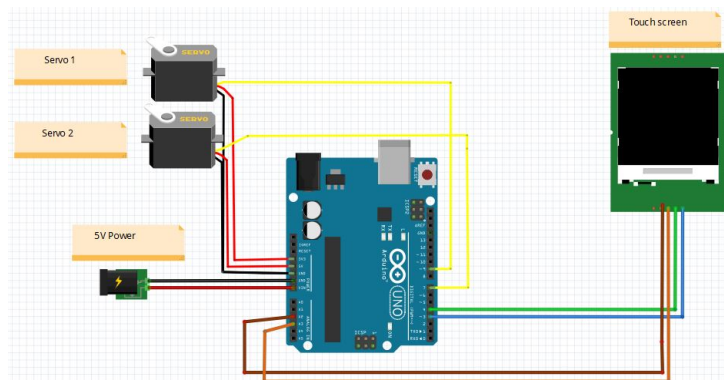


Figure 4.1: system wiring

4.2 Resistive touch screen

Ball position feedback is accomplished through the usage of an analog resistive touch screen element. As the feedback in this system, the touch screen will read the application of pressure by a steel ball and track it as it moves within the active portion of the screen. An analog resistive touch screen works using a simple design. Two layers of material are coated with indium tin oxide (ITO) to give them a known resistance, something between 100 and 500 ohm/sq. depending on the manufacturer and the application. The two layers are then placed 90 degree offset from each other on top of a glass or polycarbonate substrate, with a grid of transparent insulating dots separating the layers. Silver bus bars are placed at the edges to allow the application of voltage across the layers. To generate a touch position, the user or an object must exceed the activation force of the screen, pressing the two layers together. This creates an electrical connection between the layers. However, both X and Y axes cannot be read simultaneously, because the concept works by applying a voltage across one layer, and looking for the voltage that appears on the other layer. Therefore, the system works by alternately applying a voltage to one layer and reading off of the other. The voltages are then read in by an analog-to-digital (A/D) converter to be used as a coordinate value[1].

There are, however, several types of analog resistive touch screens, notably 4-wire, 5-wire and 8-wire. These designations refer to the number of wires between the screen and the screen controller. 4-wire screens operate by applying voltage across the two wires for one layer, and reading a voltage from the ground wire of the other layer. The voltage application and read operation then swaps around for the other layer. The 5-wire screens use the bottom layer for both axes measurements, applying voltage across the top layer only. This simplifies measurement and voltage application duties over the 4-wire screen. The 8-wire screen is an extension of the 4-wire screen, using the additional 2-wires per layer to provide a stable voltage gradient. This makes the 8-wire screen immune to voltage and resistance fluctuations due to ambient heat and humidity or aging of the screen. however in this project we use a 4-wire resistive touch screen ,you can see more detalise on resistive touch screen in appendices.4 wire touch screen is shown in 4.2 below.

SKYLARPU

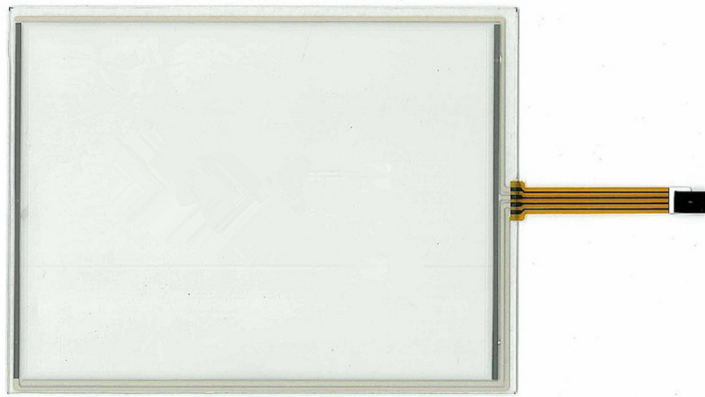


Figure 4.2: 4-wire resistive touch screen

4.3 Servo motor

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration it is a suitable motor coupled to an encoder sensor for position feedback. It also has a relatively customized controller, often a dedicated module designed specifically for the use with servomotors. Here we use HS-422 servo motor as shown in 4.3. you can find more details about the servo in appendices.



Figure 4.3: Servo Motor

4.4 Microcontroller

In this project we used Arduino Uno microcontroller. Arduino Uno is a microcontroller board . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started (See 4.4).you can find more details about the arduino uno in appendices.

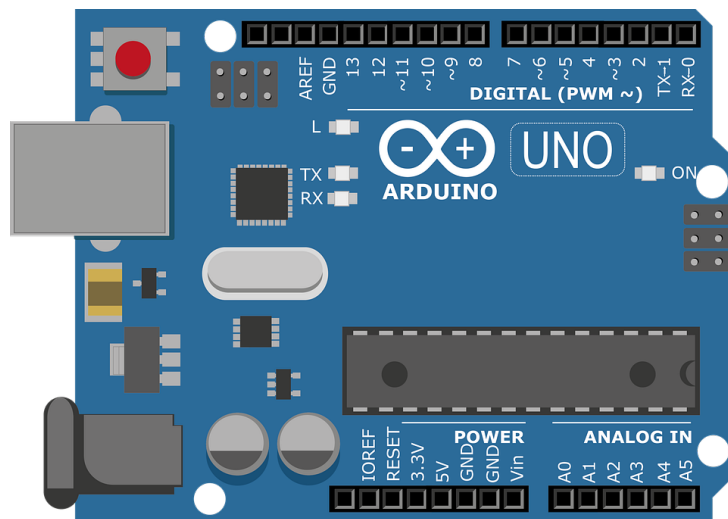


Figure 4.4: Microcontroller

Chapter 5

Control Design

5.1 Introduction to control design

The system we have here is unstable system, so we need to use control methods in order to stabilize the system. There are two common classes of control systems, open loop control systems and closed loop control systems. In open loop control systems output is generated based on inputs. In closed loop control systems current output is taken into consideration and corrections are made based on feedback. A closed loop system is also called a feedback control system. Linear: A linear system is a mathematical model of a system based on the use of a linear operator. Linear systems typically exhibit features and properties that are much simpler than the general, nonlinear case. After we derive mathematical model for the system now we can analyzed, and a controller can be designed to meet the required performance. In the following sections we will present a state-space representation using matlab software for the system. And the control methods we intend to use.

5.2 State space model

We can write state equations of ball-on-plate system by considering Equations. As mentioned τ_x and τ_y are torques exerted on the plane or on the ball from the motor in y-axis and x-axis direction respectively. By defining state variables $= [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8]^T = [x_b, \dot{x}_b, \alpha, \dot{\alpha}, y_b, \dot{y}_b, \beta, \dot{\beta}]^T$, $U = [u_x, u_y]^T = [\alpha, \beta]^T$ and constant value $b = \frac{m}{m + \frac{I_b}{r_b}}$. Because of the low velocity and acceleration of the plate rotation ($|\dot{\alpha}| \ll 1$ and $|\dot{\beta}| \ll 1$) we can omit the coupling term in $f(x, u)$ and divide the system into two sub-systems and control them independently.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -b/g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\frac{mg}{(\frac{j}{2} + m)r} & 0 \\ 0 & -\frac{mg}{(\frac{j}{2} + m)r} \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix}$$

$$c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}$$

5.3 Transfer Function

By assuming $\alpha(s)$ and $\beta(s)$ as inputs to ball and plate system, we find the transfer functions:

$$G_X(s) = \frac{X_b}{\alpha(s)} = \frac{g}{\frac{5}{7}s^2}$$

$$G_Y(s) = \frac{Y_b}{\beta(s)} = \frac{g}{\frac{5}{7}s^2}$$

5.4 Simulation

In this section we will show how the ball and plate system control problem is solved going from the non-linear equations to the linear model and simulation using Matlab and Simulink toolbox. We divide the system in two parts x and y because it was the easiest way to have a full control. In the following section we will present a matlab model and their response matlab m.file can find in the appendices.

5.4.1 Non-linear Simulation

In this subsection we will show how non-linear model is build.S-function for the whole non-linear system is build you can see s-function in appendices .figure below show the simulink model.

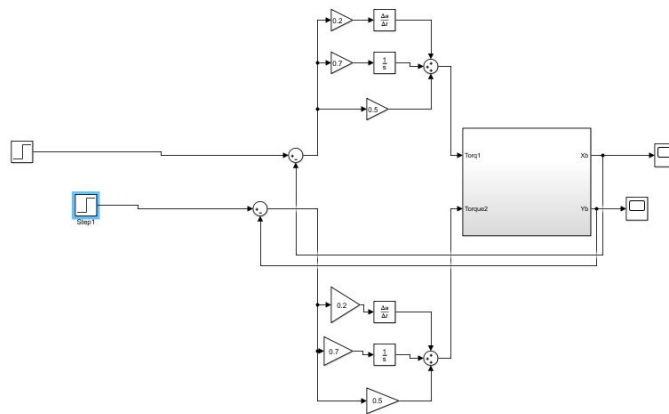


Figure 5.1: Non-linear model

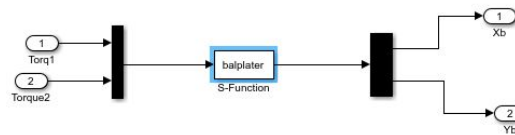


Figure 5.2: s-function block

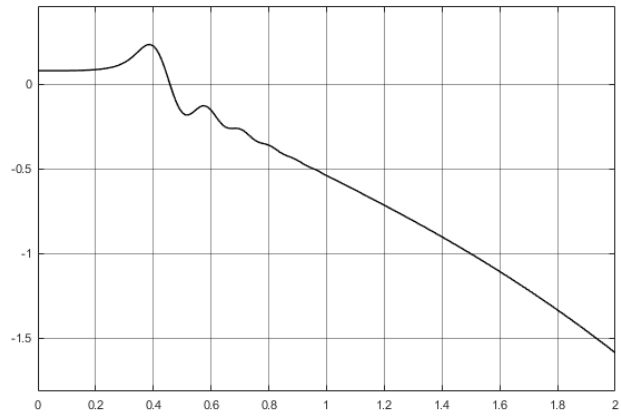


Figure 5.3: X-System response

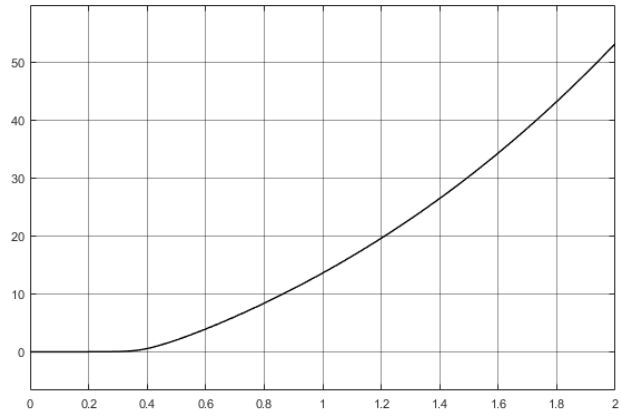


Figure 5.4: y-System response

5.4.2 Result from Non-linear

As we can see in figures above the response from non-linear system is unstable system that because the PID control can't deal with such complex system in order to make the system stable you have to design another controller like optimal control.

5.4.3 state feedback

We use state feedback control as regulator to achive point stabilization regardless disturbance force on the ball. The regulator is to design a controller that, while guaranteeing the stability of the closed-loop system, drives the tracking error to zero. when the reference and disturbance signals are produced the regulator controller drive it to zero. Figure 5.5 shown a state feedback model.

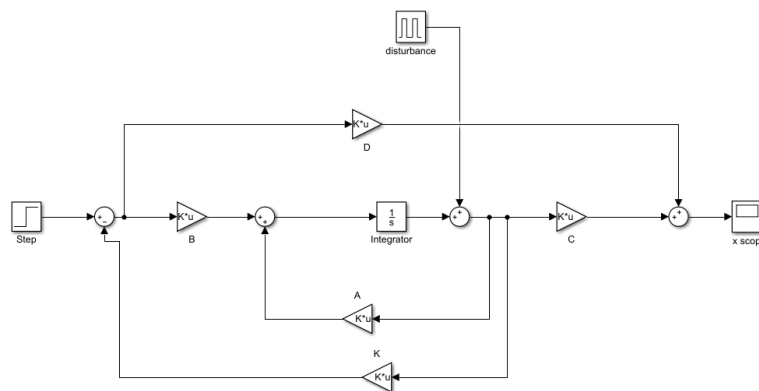


Figure 5.5: State feedback Model

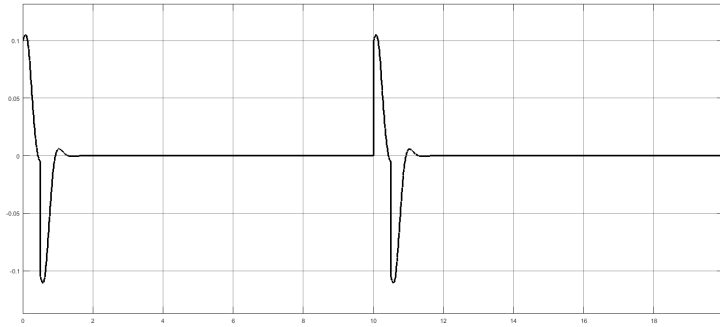


Figure 5.6: State feedback Response

as we can see in Figure 5.6 the system was able to regulate the disturbance signals and drive it to zero .

5.4.4 Optimal control LQR

In this section we present the theory of a Linear Quadratic Regulator and the matlab simulation optimal control depends on system states and the weighting matrix Q and R ,Since the important state is position of the ball, and care less about the velocity, we choose the Q matrix to have a high weighting on the position states.Adjustment of the R matrix will either promote or penalize the control effort. By altering these values, we can further adjust the time-domain response, while making sure not to operate in the saturation region of our motors. For this reason, it is important to take the motor saturation into account when designing the simulation.The matlab m.file can be found in appendices.

$$Q = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

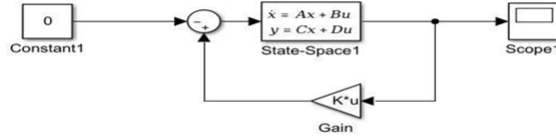


Figure 5.7: Optimal control in Simulink

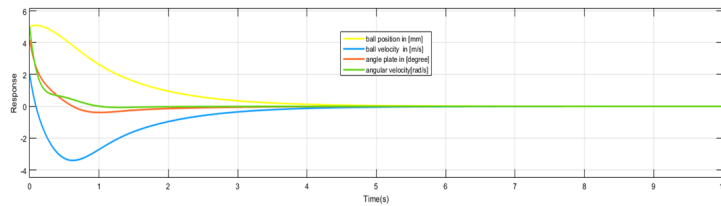


Figure 5.8: Optimal controller response in x-axis

With these state penalty matrices, the state gain matrix was calculated with MATLAB. $K = \begin{bmatrix} 3.162 & 5.0464 & -16.75 & 4.0334 \\ 3.162 & 5.0464 & -16.75 & 4.0334 \end{bmatrix}$

Results from the simulation. Scope1: x; scope2: x ; scope3: ; scope4: As we can see each graphic tends to different values in a stable way, we get a stable system. can see in 5.85.9.

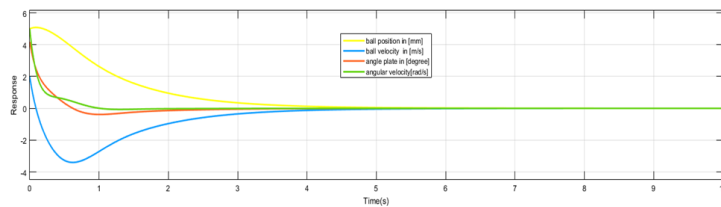


Figure 5.9: Optimal controller response in y-axis

5.4.5 PID

In order to obtain easier control algorithm the MIMO system is simplified to a simpler combination of 2 SISO systems. As per the SISO model of the plant in X-axis and Y-axis separately the PID control.[13]

In order to provide a system that is more resistant to disturbances the parameters derived by tuning the settling time was much faster. The final PID parameters is $K_p = 1.01$; $K_i = 0.038$ and $K_d = -0.105$.

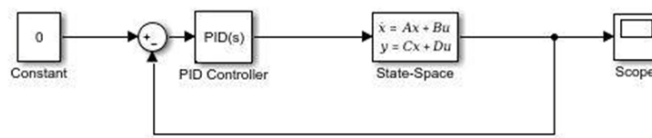


Figure 5.10: Close loop of the system in x-axis

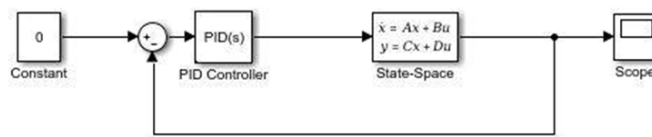


Figure 5.11: close loop system in y-axis

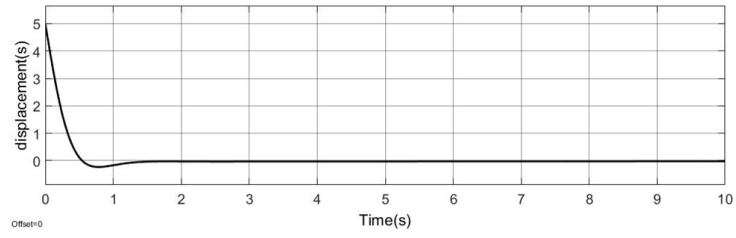


Figure 5.12: x-axis response

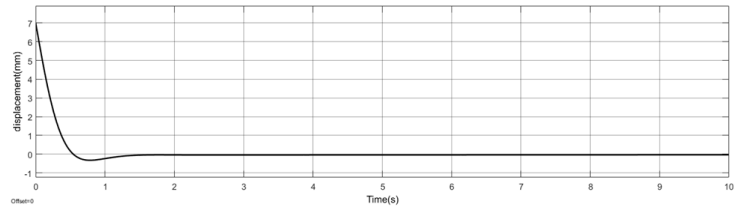


Figure 5.13: y-axis response

In Figure 5.13 and Figure 5.12, the simulation result for the tracking of a desired is presented where we obtained an error less than 3 mm using these parameters $K_p = 1.01$; $K_i = 0.038$ and $K_d = -0.105$.

Chapter 6

Experimental Setup

6.1 Hardware

Several physical components were selected to build the prototype servo motors, servo motors base, central shafts, touch screen, belts, universal joint were necessary in order to realize the construction of the system. These custom made parts. Each of these parts of the system were machined out of aluminum, steel and plastic and their full dimensions are included in Chapter 3 (prototype design). The lengths and widths of each component were designed to fit the size and shape of the touch screen that was used in the project with some slight clearance along the long side of the universal joint to allow for wiring to move freely. The special shaft was designed to support the holder of touch screen. The motor mount bracket was created to connect the tilt axis motor to the system. It was designed to place the motor directly on the base.

6.1.1 Prototype

Figure 6 shows the developed prototype. The movement of the prototype operates the same as the drawings model; in such the resistive touch panel detects and sends data to the controller and microcontroller, commanding the two servos to produce torque to move the arm links and panel. There are few alterations made to the dimensions of the arm links and mounts for the servos. Inclusively, the ball-and-plate system is created in relation to the model simulation.

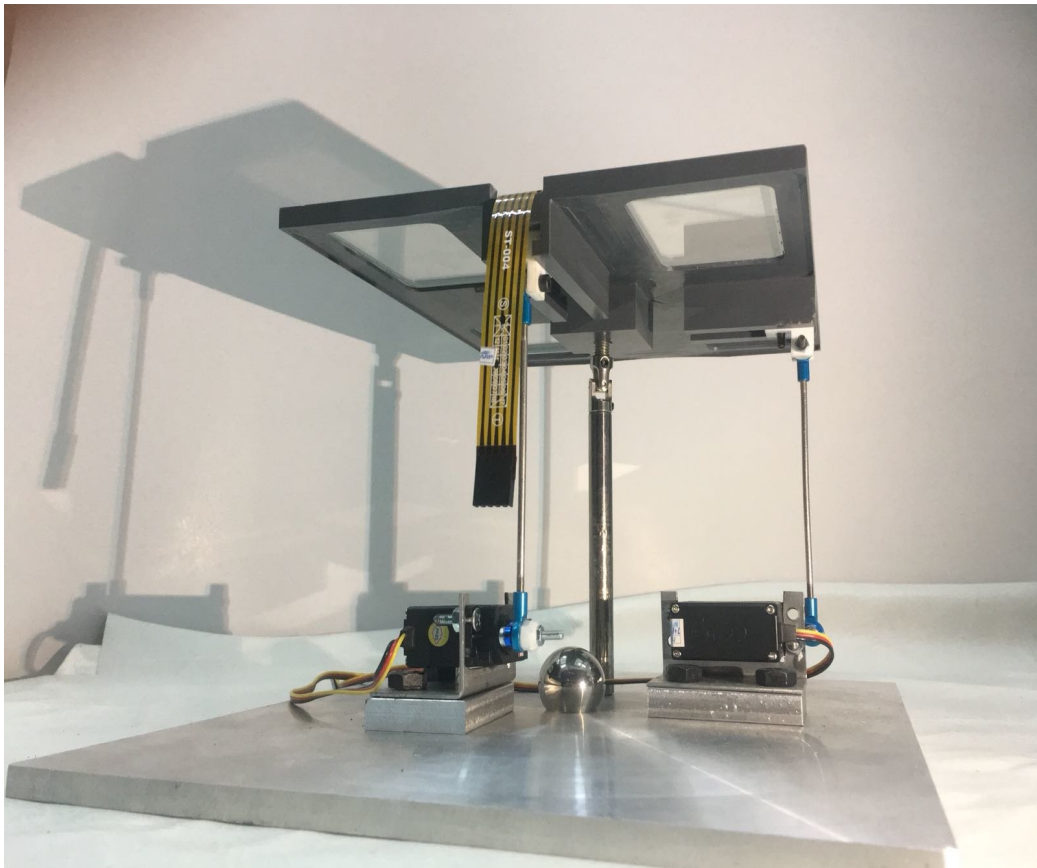


Figure 6.1: Real model

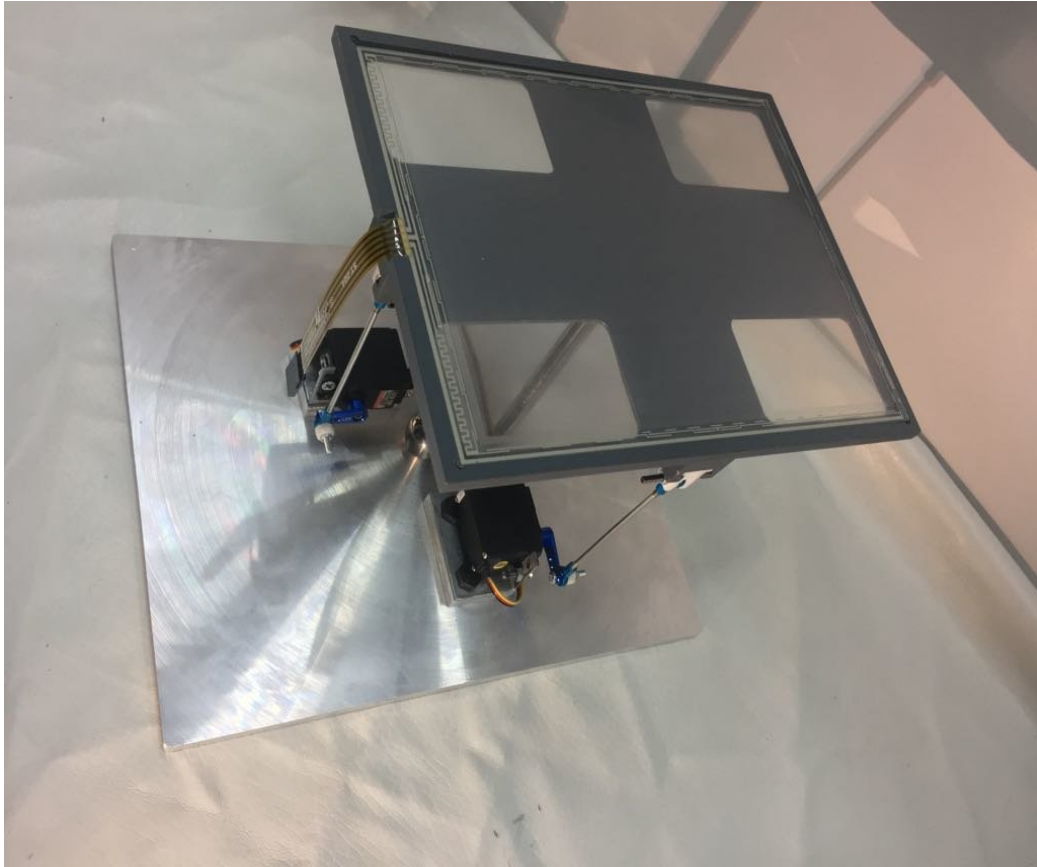


Figure 6.2: Real model

6.1.2 Real system response

In the Figure below you can see a response from the real model as you can see the system is able to drive the ball position with disturbance rejection.

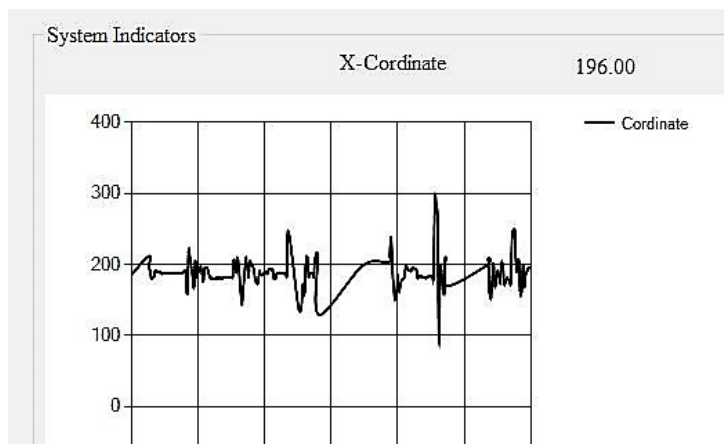


Figure 6.3: x-cordinate response

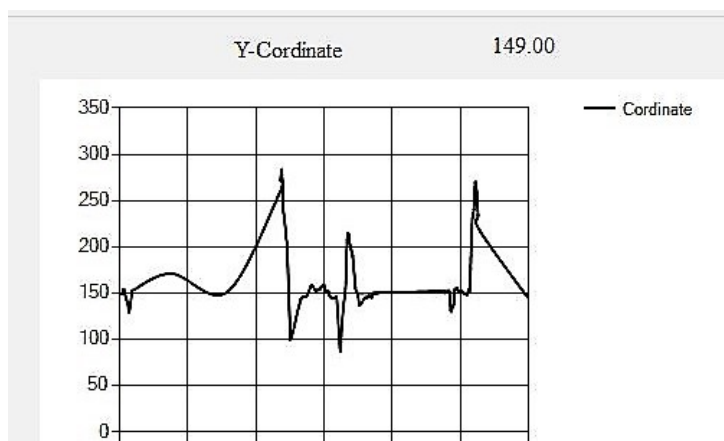


Figure 6.4: y-cordinate response

6.2 User Interface

6.2.1 Introduction to User interface

The user interface (UI), in design field of human–computer interaction, is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators’ decision-making process. Examples of this broad concept of user interfaces include the interactive aspects of computer operating systems, hand tools, heavy machinery operator controls, and process controls. The design considerations applicable when creating user interfaces are related to or involve such disciplines as ergonomics and psychology. In this project we use Visual studio software to design out user interface. Visual Studio provides a rich, integrated development environment for creating stunning applications for Windows, Android, and iOS, as well as modern web applications and cloud services. Visual Studio 2017 also provides a comprehensive, highly flexible set of application lifecycle management (ALM) tools. Visual Studio subscriptions offer customers high-value subscriber benefits such as development/test use rights for Microsoft platform software like SQL Server/Windows/Windows Server, monthly Microsoft Azure credits, a developer account for publishing apps to the Windows Store and an Office 365 Developer subscription.

6.2.2 Graphical user interface

The graphical user interface (GUI) was designed using Visual studio design environment to control the real model. The designed GUI implements 14 sources of the reference value SetpointX,SetpointY,System states,X-servo angle,Y servo angle,X-cordinate,Y-cordinates,X-servo response,Y-servo response as you can see in 6.5.

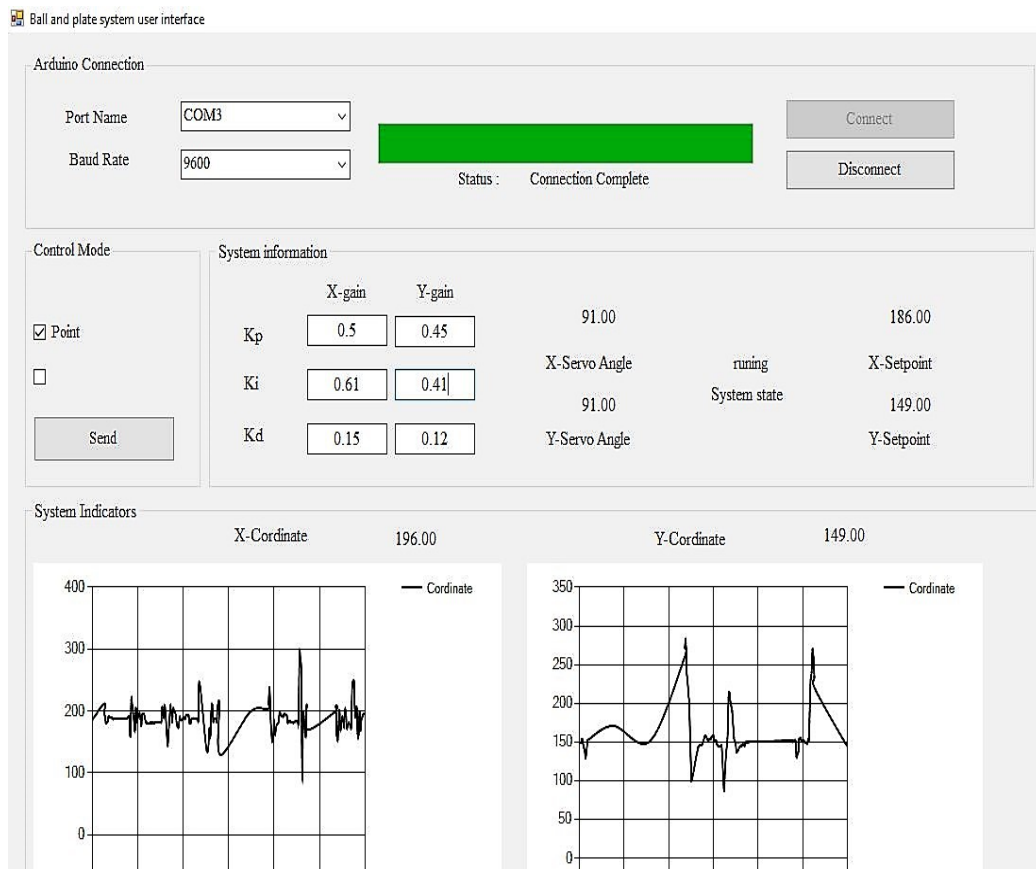


Figure 6.5: Graphical user interface

Chapter 7

Conclusions and Future Work

7.1 Conclusions

We described the design and construction of a two dimensional four degree of freedom ball and plate system. The objective of the project was to control the position of the ball on the plate for static positions, as well as have the system capable of correcting for disturbances in ball position these goals were accomplished, static position balancing and disturbance rejection. The ball can be centered on the plate or another coordinate location, and will quickly correct for even large disturbances to the ball position with fast response time, small overshoot and a very minimal steady state error. The final results showed that the designed controllers had almost acceptable performances.

In this project we used PID controller algorithms to control the ball position. The resulting controller was able to reject disturbances. This solution was robust enough to keep the ball in same position with disturbance rejection. The other control algorithms was design for simulation purpose only.

The graphical user interface (GUI) was designed, for control the real model. This proved to be a valuable tool to quickly estimate reference values and see the final outcome of the model.

In non-linear simulation the pid control can't make the system to be stable but in real system the pid control success to make the system stable

7.2 Future Work

After we built and test the prototype, it can be used as an excellent test-bed for testing various other control algorithm like optimal control full-state feedback . Although controllers based on the linear model perform well but it will interesting to apply of non-linear controls and have further improvements in the system performance.Design a control algorithm to make the ball trace any desired path on the plate, for example a circle.

Chapter 8

References

1. Radi, K., A. Faraj, and Y. Amsad, *Theoretical design of a ball balancing on plate controller*. Al-Mustansiriya University, 2008.
2. Zeeshan, A., N. Nauman, and M.J. Khan. *Design, control and implementation of a ball on plate balancing system*. in *Applied Sciences and Technology (IBCAST), 2012 9th International Bhurban Conference on*. 2012. IEEE.
3. Negash, A. and N.P. Singh. *Position Control and Tracking of Ball and Plate System Using Fuzzy Sliding Mode Controller*. in *Afro-European Conference for Industrial Advancement*. 2015. Springer.
4. Wang, Y., et al., *A novel disturbance-observer based friction compensation scheme for ball and plate system*. *ISA Trans*, 2014. **53**(2): p. 671-8.
5. Fan, X., N. Zhang, and S. Teng, *Trajectory planning and tracking of ball and plate system using hierarchical fuzzy control scheme*. *Fuzzy Sets and Systems*, 2004. **144**(2): p. 297-312.

6. Wellstead, P.E., *Introduction to physical system modelling*. 1979: Academic Press London.
7. Dušek, F., D. Honc, and K.R. Sharma. *Modelling of ball and plate system based on first principle model and optimal control*. in *Process Control (PC), 2017 21st International Conference on*. 2017. IEEE.
8. Zamani, N.G., *CATIA V5 FEA tutorials: release 19*. 2010: SDC Publications.
9. Walker, G., *A review of technologies for sensing contact location on the surface of a display*. *Journal of the Society for Information Display*, 2012. **20**(8): p. 413-440.
10. Badamasi, Y.A. *The working principle of an Arduino*. in *Electronics, Computer and Computation (ICECCO), 2014 11th International Conference on*. 2014. IEEE.
11. Rodriguez, R.R., D.A.R. Diaz, and L.E.R. Vargas, *BALL AND PLATE*.
12. Kabil, A.M., et al., *Ball and Plate Modeling and Control Hardware Approach*.
13. Haugen, F., *The Good Gain method for PI (D) controller tuning*. Tech Teach, 2010: p. 1-7.
14. Lewis, F.L., D. Vrabie, and V.L. Syrmos, *Optimal control*. 2012: John Wiley & Sons.

Appendix A

Datasheets

Resistive Touch Screens

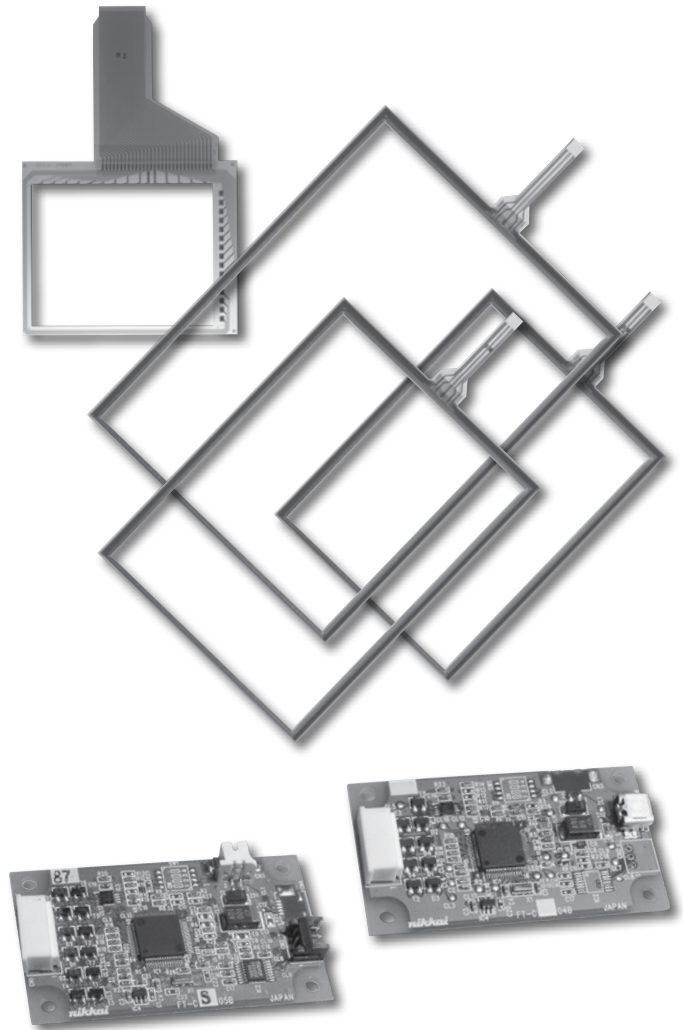
5-Wire, 4-Wire & Digital Solutions

NKK's transparent touch screens are engineered to complement the application of choice while offering superior durability and flexibility. With options in multiple sizes, and choices of input by finger, gloved finger or stylus, we maintain a consistent focus on impeccable quality and value added solutions with the diverse needs of our customers at the forefront.

Whether an application requires the 5- or 4-wire technology, the features include metal tails (analog), contact reliability with a connector, and ANR film, eliminating many of the typical visual artifacts. The film surface is non-glare and hard coated for ease of use and integrity of the surface.

Additional benefits of NKK's 5-wire touch screens include:

- Screens highly resistant to static electricity and noise pollution
- Drift-free operation despite any temperature fluctuation
- Greater touch point density translating to more precision and reduction of false actuations
- Quicker response time



DISTINCTIVE CHARACTERISTICS

- Wide Range of Available Sizes
- Custom Solutions a Specialty
- Digital and Analog Solutions
- Controllers Available
- Anti-Newton Ring (ANR) Technology
- Design Minimizes Visual Artifacts
- RoHS Compliant

APPLICATIONS

- Information Kiosks
- Industrial Automation
- Banking, Exchange Management Systems
- Broadcast
- Office Automation
- Medical Equipment
- Hand-held Devices
- Hospitality and Restaurant
- Gaming

Customization Options

Parameter	Notes & Options
Resistive Analog Touch Screen	4-Wire or 5-Wire
Integrate LCD & Touchscreen	Yes - No
5-Wire - Screen Size Diagonal Inches	Standard 10.4, 12.1, 15 (min 10.4, max 19)
4-Wire - Screen Size Diagonal Inches	Standard 5.7, 6.5, 8.4, 10.4, 12.1, 15 (min 2.5, max 19)
Data Entry Area mm x mm	Same as the dimensions of the display area of the LCD
Viewable Area mm x mm	Same as the dimensions of the bezel opening of the LCD
Perimeter Dimensions mm x mm	
Tail Type	PCB standard, FPC option
Tail Pitch	1.25mm is standard
Tail Pins	8 is standard, 4 is option
Tail Length	2 standard options: 65mm or 80mm
Tail Base Width	28.2mm is standard
Tail Location	Left side is standard, non-std options top or bottom
Tail Material	Carbon-coated silver is standard, option is copper-gold
Glass Thickness	1.1 and 1.8mm are standard (note: total thickness = glass thickness + 0.3mm)
Hardcoat Treatment	Standard
Anti-Newton Ring Treatment	Standard
Optical Transmission Factor	80% is standard
Controller	Yes - No (no = customer will supply)
Communication	USB or RS232
Operating System	Windows 7 or Windows XP

Toggle

Rockers

Pushbuttons

PB

Illuminated

Keylocks

Rotaries

Slides

Tactiles

Tilt

Touch

Indicators

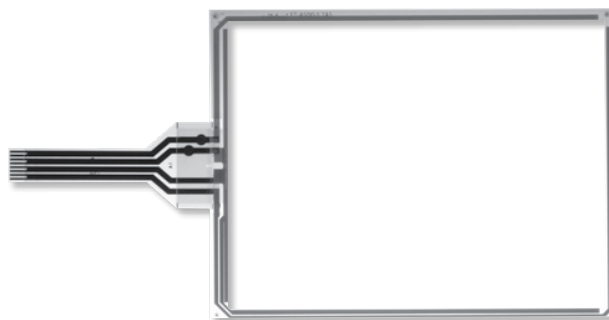
Accessories

Supplement

GENERAL SPECIFICATIONS FOR 4-WIRE

4-Wire Analog Resistive Touch Screens

Optical	
Light Transmission	Analog: 80% standard Digital: 78% standard
Film Options	Anti-glare, anti-Newton ring standard
Electrical	
Power Level	1mA @ 5V DC (resistive load)
Insulation Impedance	10MΩ minimum @ 25V DC
Linearity	3% maximum (analog)
Chattering Time	10 milliseconds maximum
Mechanical	
Touch Activation Force	1.4N maximum
Available Sizes	5.7" ~ 15" standard
Durability	
Surface Hardness	2H (JIS K5600)
Expected Operational Life	1,000,000 operations minimum
Environmental	
Operating Temperature Range	-10°C ~ +60°C (+14°F ~ +140°F)
Storage Temperature Range	-20°C ~ +70°C (-4°F ~ +158°F)
Relative Humidity	+60°C (+140°F), humidity 90%, 240 hours



**Analog
FTAS00-57AS4**

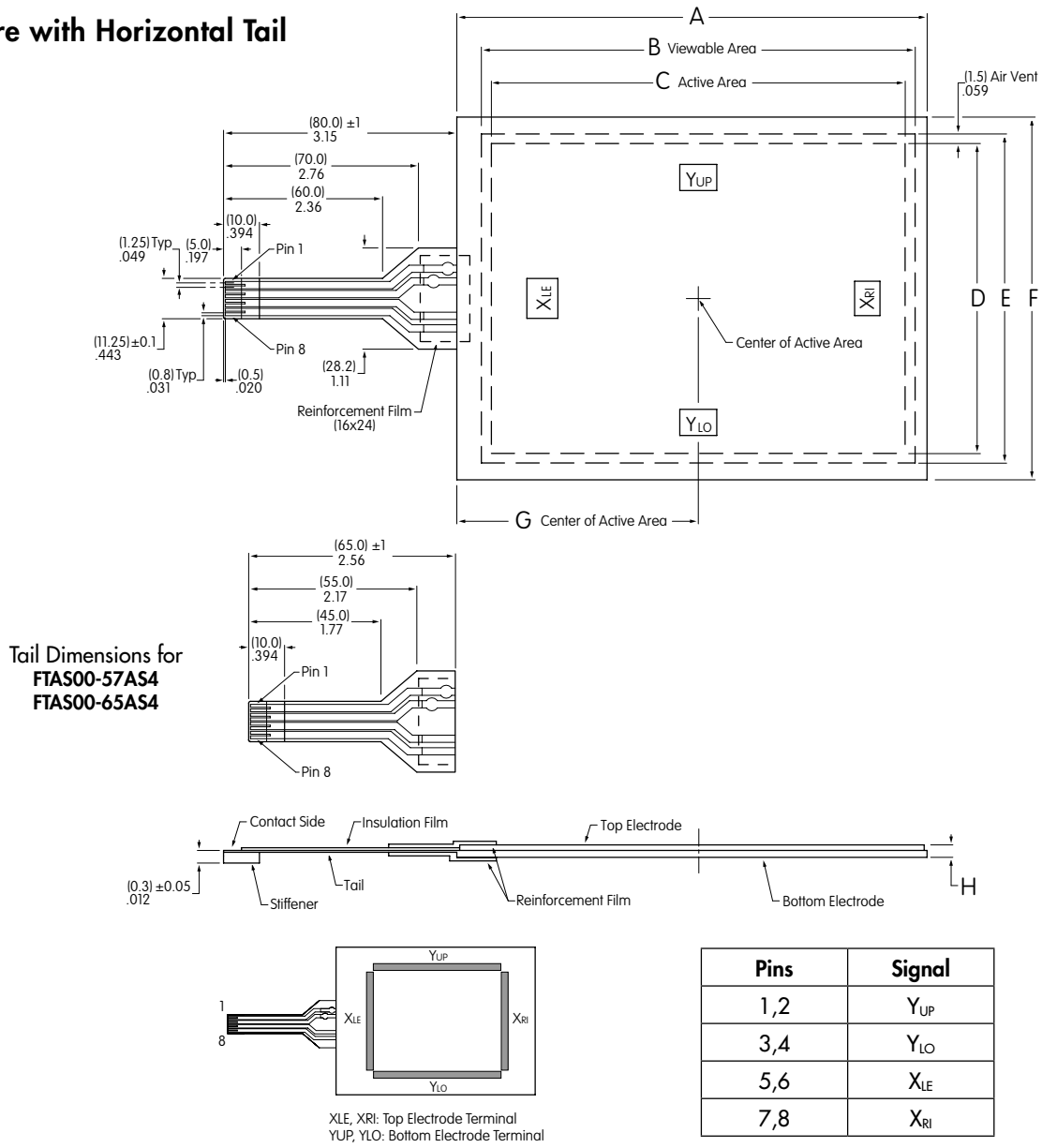
PART NUMBERS & DESCRIPTIONS FOR 4-WIRE

4-Wire Analog Touch Screens						
Part Number	Screen Size in Inches	Key Area Dimensions	Viewing Area Dimensions	External Dimensions	Panel Thickness	* Terminal Detail 8 Pin .049" (1.25mm) Pitch
FTAS00-57AS4	5.7	4.54" x 3.40" (115.2mm x 86.4mm)	4.76" x 3.61" (121.0mm x 91.6mm)	5.16" x 3.98" (131.0mm x 101.0mm)	.055" (1.4mm)	Length 2.56" (65.0mm)
FTAS00-65AS4	6.5	5.20" x 3.90" (132.0mm x 99.0mm)	5.43" x 4.13" (138.0mm x 105.0mm)	5.91" x 4.57" (150.0mm x 116.0mm)		Length 2.56" (65.0mm)
FTAS00-84AS4	8.4	6.73" x 5.10" (170.9mm x 129.6mm)	6.95" x 5.33" (176.5mm x 135.4mm)	7.34" x 5.69" (186.5mm x 144.4mm)	.083" (2.1mm)	Length 3.15" (80.0mm)
FTAS00-104AS4	10.4	8.32" x 6.24" (211.2mm x 158.4mm)	8.47" x 6.39" (215.0mm x 162.4mm)	8.88" x 6.75" (225.6mm x 171.4mm)		Length 3.15" (80.0mm)
FTAS00-104AV4	10.4	8.35" x 6.28" (212.2mm x 159.4mm)	8.52" x 6.43" (216.4mm x 163.4mm)	8.92" x 7.21" (226.5mm x 183.0mm)		Length 3.15" (80.0mm)
FTAS00-121A4	12.1	9.72" x 7.30" (246.76mm x 185.32mm)	10.04" x 7.53" (255.0mm x 191.32mm)	10.67" x 8.07" (271.0mm x 205.0mm)		Length 3.15" (80.0mm)
FTAS00-121AS4	12.1	9.69" x 7.26" (246.0mm x 184.5mm)	9.84" x 7.42" (250.0mm x 188.5mm)	10.28" x 7.80" (261.0mm x 198.0mm)		Length 3.15" (80.0mm)
FTAS00-150A4	15.0	12.05" x 9.06" (306.1mm x 230.1mm)	12.21" x 9.25" (310.0mm x 235.0mm)	12.91" x 9.84" (328.0mm x 250.0mm)	Length 3.15" (80.0mm)	

Note: Input methods are finger or stylus.

* 4 pin available with 1.0mm or 1.25mm pitch. Contact factory for details.

4-Wire with Horizontal Tail



Tail Dimensions for FTAS00-57AS4 FTAS00-65AS4

X_{LE}, X_{RI}: Top Electrode Terminal
Y_{UP}, Y_{LO}: Bottom Electrode Terminal

4-Wire Analog Touch Screen Dimensions									
Part Number	Screen Size in Inches	Dim A	Dim B Viewable Area	Dim C Active Area	Dim D	Dim E Viewable Area	Dim F Active Area	Dim G Center of Active Area	Dim H
FTAS00-57AS4	5.7	5.16" (131±0.3mm)	4.76" (121mm)	4.54" (115.2mm)	3.40" (86.4mm)	3.61" (91.6mm)	3.98" (101±0.3mm)	2.65" (67.25mm)	.055" (1.4mm)
FTAS00-65AS4	6.5	5.91" (150±0.3mm)	5.43" (138mm)	5.20" (132mm)	3.90" (99mm)	4.13" (105mm)	4.57" (116±0.3mm)	3.03" (77mm)	.055" (1.4mm)
FTAS00-84AS4	8.4	7.34" (186.5±0.3mm)	6.95" (176.5mm)	6.73" (170.9mm)	5.10" (129.6mm)	5.33" (135.4mm)	5.69" (144.4±0.3mm)	3.73" (94.85mm)	.083" (2.1mm)
FTAS00-104AS4	10.4	8.88" (225.6±0.3mm)	8.46" (215mm)	8.31" (211.2mm)	6.24" (158.4mm)	6.39" (162.4mm)	6.75" (171.4±0.3mm)	4.49" (114.1mm)	.083" (2.1mm)
FTAS00-121AS4	12.1	10.28" (261±0.3mm)	9.84" (250mm)	9.69" (246mm)	7.26" (184.5mm)	7.42" (188.5mm)	7.80" (198±0.3mm)	5.18" (131.6mm)	.083" (2.1mm)
FTAS00-150A4	15.0	12.91" (328±0.3mm)	12.20" (310mm)	12.05" (306.1mm)	9.06" (230.1mm)	9.25" (235mm)	9.84" (250±0.3mm)	6.52" (165.6mm)	.083" (2.1mm)

Toggles

Rockers

Pushbuttons

Illuminated PB

Programmable

Keylocks

Rotaries

Slides

Tactiles

Tilt

Touch L

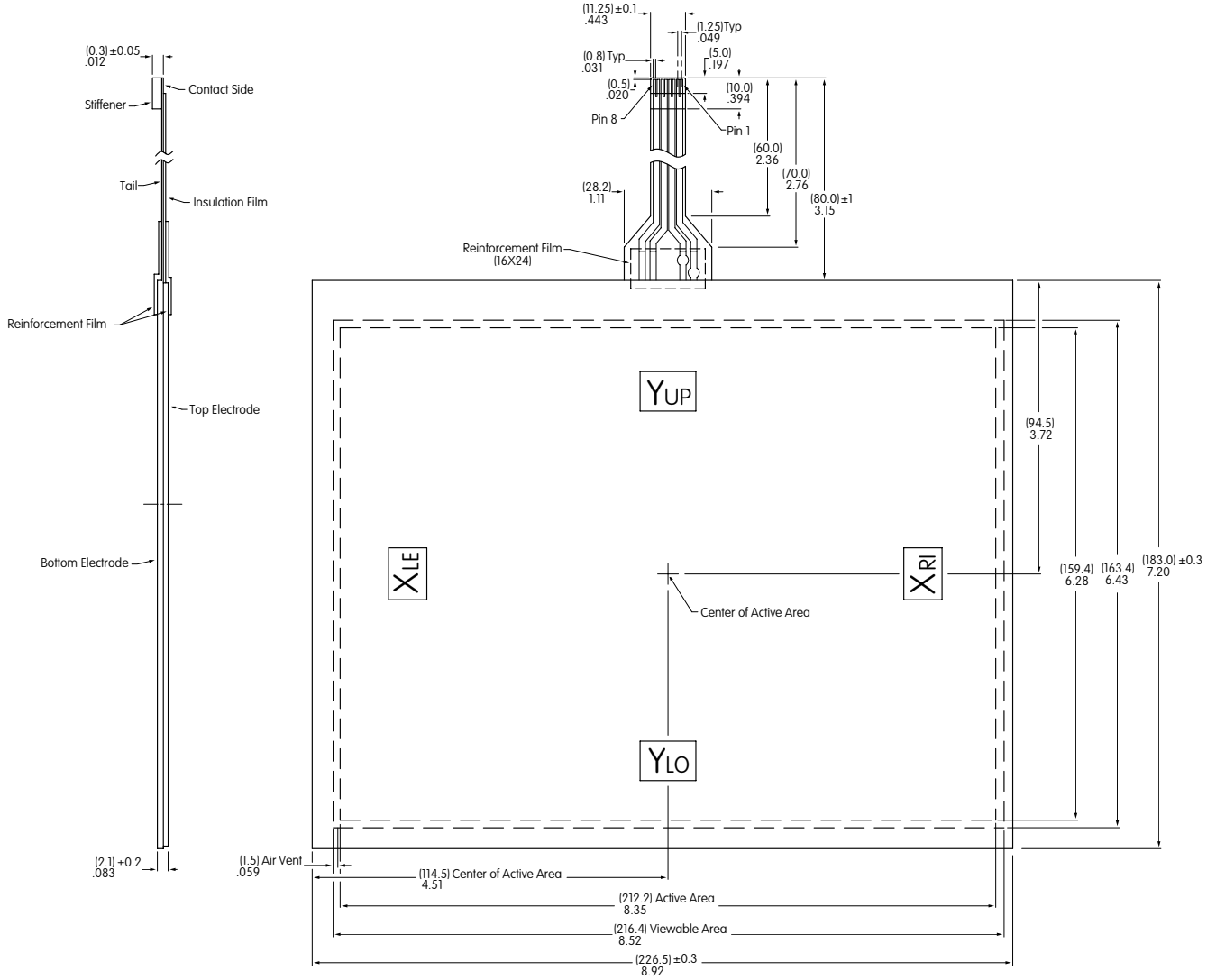
Indicators

Accessories

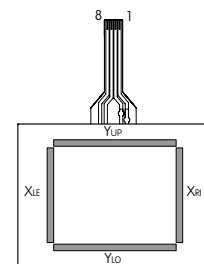
Supplement

4-Wire with Vertical Tail

FTAS00-104AV4



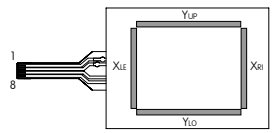
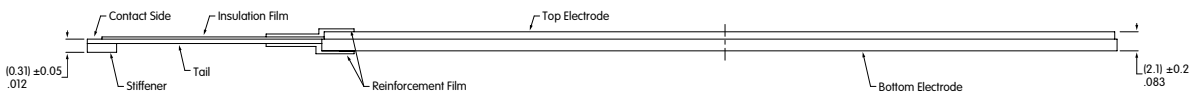
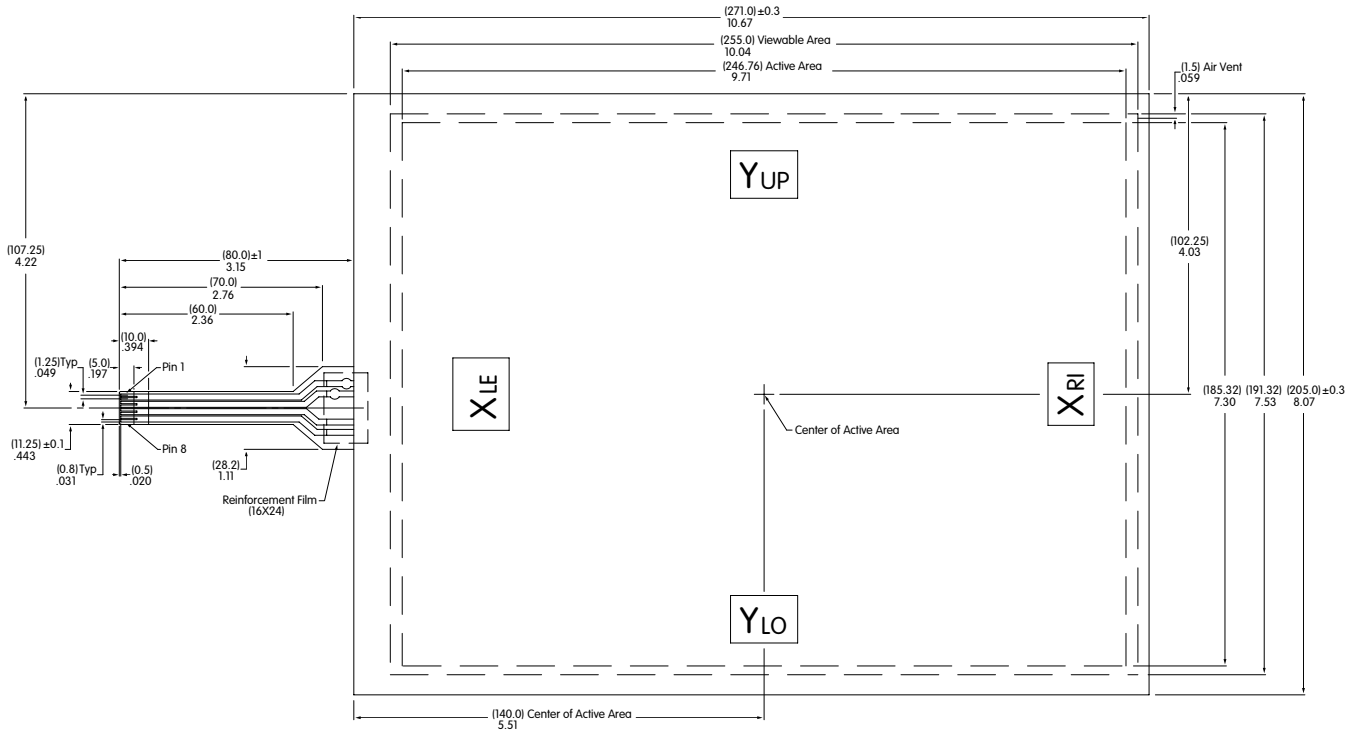
Pins	Signal
1,2	Y _{UP}
3,4	Y _{LO}
5,6	X _{RI}
7,8	X _{LE}



XLE, XRI: Top Electrode Terminal
YUP, YLO: Bottom Electrode Terminal

4-Wire with Horizontal Tail (Off-Center)

FTAS00-121A4



XLE, XRI: Top Electrode Terminal
YUP, YLO: Bottom Electrode Terminal

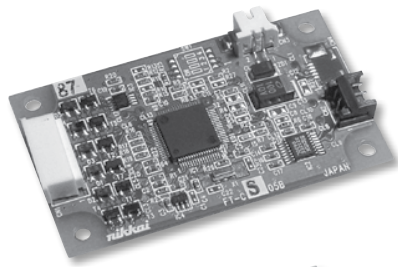
Pins	Signal
1,2	Y _{UP}
3,4	Y _{LO}
5,6	X _{LE}
7,8	X _{RI}

4-Wire Touch Screen Controller Boards & Drivers

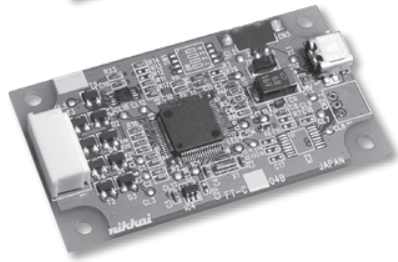
DISTINCTIVE CHARACTERISTICS

- High Quality and Reliability
- Easy Integration Replacing Mouse Functionality
- Compatible with Control Board USB/RS2
- Device Driver Compatible with Vista and Windows XP Operating Systems

Controller Boards Available for RS232C



Controller Boards Available for USB



NKK offers controller boards compatible with USB or with RS232C. See web site or contact factory for specifications and technical data.

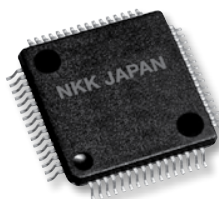
Controller Boards		
Type	Part No.	Communication Protocol
4-Wire	FTCS04B	RS232C
4-Wire	FTCU04B	USB

See web site for dimensioned drawings or technical data for any of the controller boards and drivers.

IC & Accessories

DISTINCTIVE CHARACTERISTICS

- Interface: USB and RS232C
- High Speed and Accuracy
- Built-in Calibration Function
- Data Function Removal Built In to Eliminate Noise



IC FTCSU564

The IC is for use with the 5- and 4-wire transparent touch screens, and is available for those who prefer to design their own controller boards. When the screen is touched, it recognizes the position of the touch by the level of the analog voltage detected by the A/D. The A/D converter receives the value and sends a set of coordinate values as serial data or USB.

See web site or contact factory for IC specifications.

OPTIONAL ACCESSORIES

AT713 Receptacle Connector

This Receptacle Connector with code connects to RS232C communication of the controller boards. It is compatible with FTCS04A and FTCS04A2.



AT714 Receptacle Connector

AT714 is a Receptacle Connector with code to connect to power source of the control boards.



For more details and dimensioned drawings of the accessories, go to the web site or call our engineering support personnel.

Storage, Handling & Installation

Below are some general precautions for the 5-wire & 4-wire touch screen devices. Please check web site for complete documentation.

Installation (4-wire, 5-wire)

- Do not pull on the tail. Do not apply stress to the tail area.
- Avoid vibration or shock.
- The touch screen mounting should not be loose.
- Ensure there are no burrs around the edges of the case or housing that can cause false actuation. The edges of the case or housing should not enter the keying area.
- The case or housing and upper electrode should have a space of about 0.5 mm to accommodate expansion or shrinkage due to humidity variances. If a shock barrier is used do not press hard on the upper electrode area. Any shock barrier should be installed more than 0.6 mm above the screen.
- To secure the touch screen, secure the lower portion with an item such as the LCD display panel. Do not secure the upper electrode with double-sided tape or similar items to avoid stress that can damage the upper or lower electrode.
- In order to balance upper and lower pressure, an air vent may be installed. Ensure that no liquid or oil will enter into the device.

Handling Precautions (4-wire, 5-wire)

- When opening product, take precaution with up/down and front/back directions. Glass edges are not chamfered, corners or edges can be sharp. Wear gloves when handling the product.
- Do not pick up the product by the tail or pull the tail area.
- Use gloves or finger cots to prevent fingerprints on surface.
- When handling the product, hold it outside of the viewing area.
- Avoid stacking multiple products or placing other items on the product.
- Remove protective film after installation is completed.

Operating Precautions (4-wire, 5-wire)

- Only operate with fingers or a touch screen stylus.
- Do not press hard with pen or similar objects between viewing area and key area.

Design Precautions (4-wire, 5-wire)

- With analog type, resistive value change can dislocate the input area. Input area can be calibrated with software.
- When installing on top of a LCD, noise from the display device can create misoperation. To avoid noise, implement actions such as grounding the display device frame.
- Do not create software for two point touch as analog type will read the center point between two touch points.
- When used to draw a line, analog type will have a break at dot spacer. Compensate for this with software.

Other Precautions (4-wire, 5-wire)

- Clean with a soft cloth and ethanol. Do not use any cleaning agents other than ethanol.
- Store product in original package and store at the temperature and humidity range specified.
- Do not store in an environment with acids or other corrosive gases or dew.
- Not suited for use in critical control systems such as nuclear power, aerospace, medical or transportation equipment, without proper failsafe design consideration.

Handling Precautions (5-wire)

- NKK warrants the 5-wire touch panel when it is used with the NKK control board and driver. Do not use third party control boards.
- 5-wire devices can misalign cathode position or touch position even after calibration. See web site for details.
- Create a larger input area. If you have the input button at the edge of a screen, it might not operate properly.
- Complete 9 point calibration with NKK driver. If more precision is desired, 25 point calibration is recommended.

Handling of Controller Board (5-wire)

- Warranty for one year after delivery. We do not warrantee the controller board unless used with NKK touch panel.
- Use arc prevention to protect device from static electricity.
- Power source should be activated after host and touch panel are connected.
- When inserting connector CN1 and touch panel tail, be sure the slider of connector CN1 is pulled. Do not pull more than 10 times.
- Do not alter the product.
- Do not use any commands other than the ones specified in specifications.
- Place the product away from noise source (such as inverter from LCD operation) since tail can be affected by noise.
- If device driver (USB) does not work after installation, reboot the host computer while connected to the controller board.
- This product does not support suspended mode (USB).
- Protocol of USB transmission is one frame per one transaction.
- Contact factory if not using protocol above.



Hitec HS422 Servo

SKU:SER0002



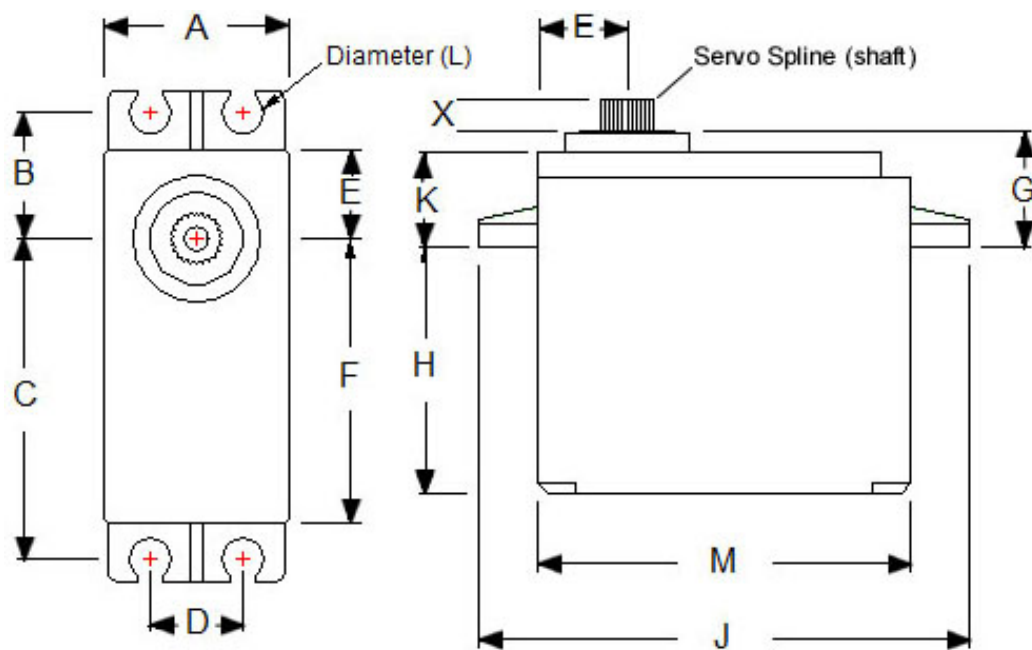
INTRODUCTION

The HS-422 has been around for a while and for a good reason, durability and the ability to be easily modified. For inexpensive robotic applications this is the perfect servo. It can easily be modified for wheeled applications since the output shaft is supported on the bottom and top with bronze bushings and the potentiometer is indirect drive. The circuit board is also separate from the motor so the potentiometer can be easily accessed. Simply a great servo at a great price.

SPECIFICATION

- Control System: +Pulse Width Control 1500usec Neutral
- Required Pulse: 3-5 Volt Peak to Peak Square Wave
- Operating Voltage: 4.8-6.0 Volts
- Operating Temperature Range: -20 to +60 Degree C
- Operating Speed (4.8V): 0.21sec/60° at no load
- Operating Speed (6.0V): 0.16sec/60° at no load
- Stall Torque (4.8V): 45.82 oz/in. (3.3kg.cm)
- Stall Torque (6.0V): 56.93 oz/in. (4.1kg.cm)
- Operating Angle: 45 Deg. one side pulse traveling 400usec
- 360 Modifiable: Yes
- Direction: Clockwise/Pulse Traveling 1500 to 1900usec
- Current Drain (4.8V): 8mA/idle and 150mA no load operating
- Current Drain (6.0V): 8.8mA/idle and 180mA no load operating
- Dead Band Width: 8usec
- Motor Type: 3 Pole Ferrite
- Potentiometer Drive: Indirect Drive
- Bearing Type: Dual Oilite Bushing
- Gear Type: Nylon
- Connector Wire Length: 11.81" (300mm=11.81in)
- Dimensions: See Schematics
- Weight: 1.6oz (45.5g)



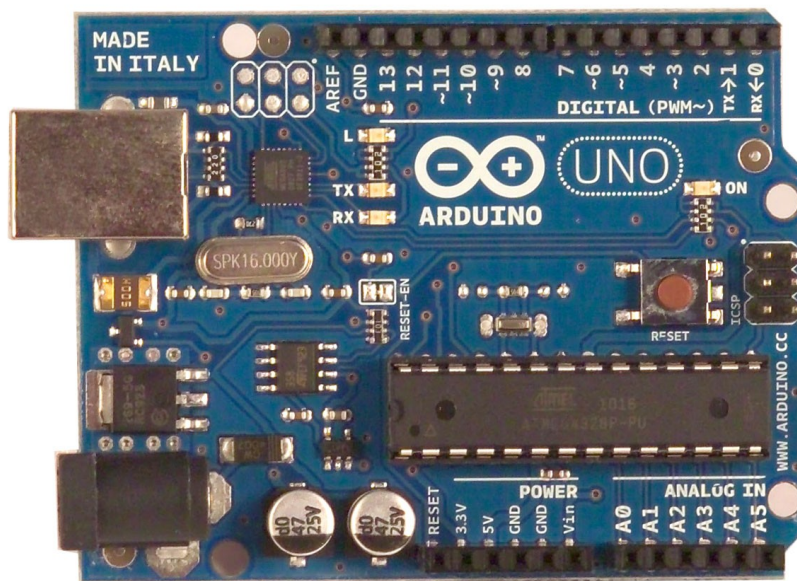


- A = .780" (19.82mm=0.78in)
- B = .530" (13.47mm=0.53in)
- C = 1.33" (33.79mm=1.33in)
- D = .400" (10.17mm=0.4in)
- E = .380" (9.66mm=0.38in)
- F = 1.19" (30.22mm=1.19in)
- G = .460" (11.68mm=0.46in)
- H = 1.05" (26.67mm=1.05in)
- J = 2.08" (52.84mm=2.08in)
- K = .368" (9.35mm=0.37in)
- L = .172" (4.38mm=0.17in)
- M = 1.57" (39.88mm=1.57in)
- X = .120" (3.05mm=0.12in)

SHIPPING LIST

- Hitec HS422 Servo x1

Arduino UNO



Product Overview

The Arduino Uno is a microcontroller board based on the ATmega328 ([datasheet](#)). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the [index of Arduino boards](#).

Index

Technical Specifications

Page 2

How to use Arduino
Programming Enviroment, Basic Tutorials

Page 6

Terms & Conditions

Page 7

Enviromental Policies
half sqm of green via Impatto Zero®

Page 7



radiospares

RADIONICS



Technical Specification

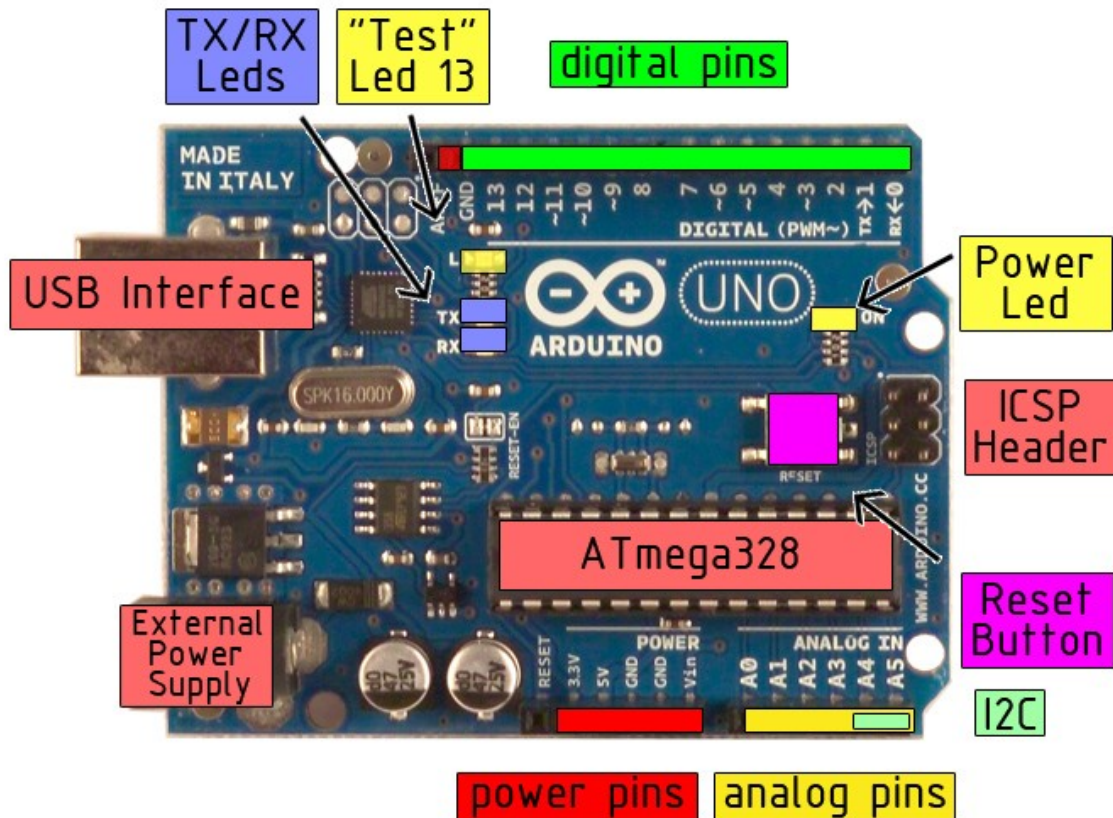


EAGLE files: [arduino-duemilanove-uno-design.zip](#) Schematic: [arduino-uno-schematic.pdf](#)

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

the board



radiospares

RADIONICS



Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the [EEPROM library](#)).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using [pinMode\(\)](#), [digitalWrite\(\)](#), and [digitalRead\(\)](#) functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the [attachInterrupt\(\)](#) function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the [analogWrite\(\)](#) function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.



radiospares

RADIONICS



The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and the [analogReference\(\)](#) function. Additionally, some pins have specialized functionality:

- **I²C: 4 (SDA) and 5 (SCL).** Support I²C (TWI) communication using the [Wire library](#).

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with [analogReference\(\)](#).
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the [mapping between Arduino pins and Atmega328 ports](#).

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required..

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [SoftwareSerial library](#) allows for serial communication on any of the Uno's digital pins.

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the [documentation](#) for details. To use the SPI communication, please see the ATmega328 datasheet.

Programming

The Arduino Uno can be programmed with the Arduino software ([download](#)). Select "Arduino Uno w/ ATmega328" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the [reference](#) and [tutorials](#).

The ATmega328 on the Arduino Uno comes preburned with a [bootloader](#) that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol ([reference](#), [C header files](#)).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see [these instructions](#) for details.

The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use [Atmel's FLIP software](#) (Windows) or the [DFU programmer](#) (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader).



RADIOSPARES

RADIONICS



Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

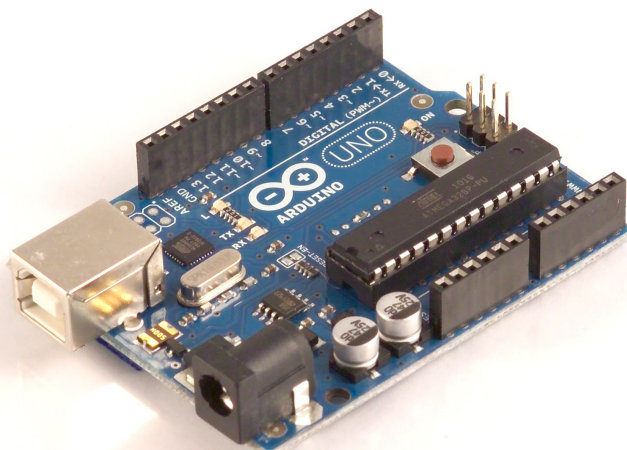
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see [this forum thread](#) for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.



Radiospares

RADIONICS



How to use Arduino



Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the [Arduino programming language](#) (based on [Wiring](#)) and the Arduino development environment (based on [Processing](#)). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platform program. You'll have to follow different instructions for your personal OS. Check on the [Arduino site](http://arduino.cc/en/Guide/HomePage) for the latest instructions. <http://arduino.cc/en/Guide/HomePage>

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

**File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink**

Once you have your sketch you'll see something very close to the screenshot on the right.

In **Tools>Board** select

Now you have to go to **Tools>SerialPort** and select the right serial port, the one arduino is attached to.

```
int ledPin = 13; // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts

void setup() {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}

// the loop() method runs over and over again,
// as long as the Arduino has power

void loop()
{
  digitalWrite(ledPin, HIGH); // set the LED on
  delay(1000);                // wait for a second
  digitalWrite(ledPin, LOW);  // set the LED off
  delay(1000);                // wait for a second
}
```



Done compiling.

Press Compile button
(to check for errors)



Upload



TX RX Flashing



Blinking Led!

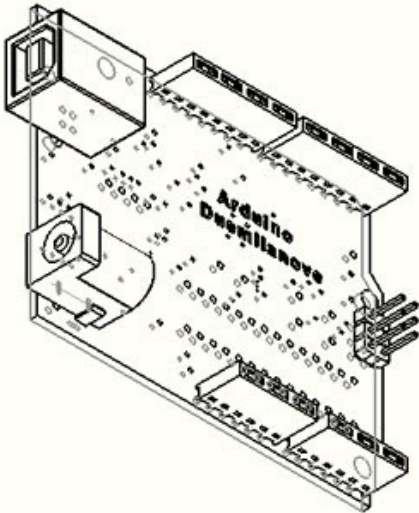
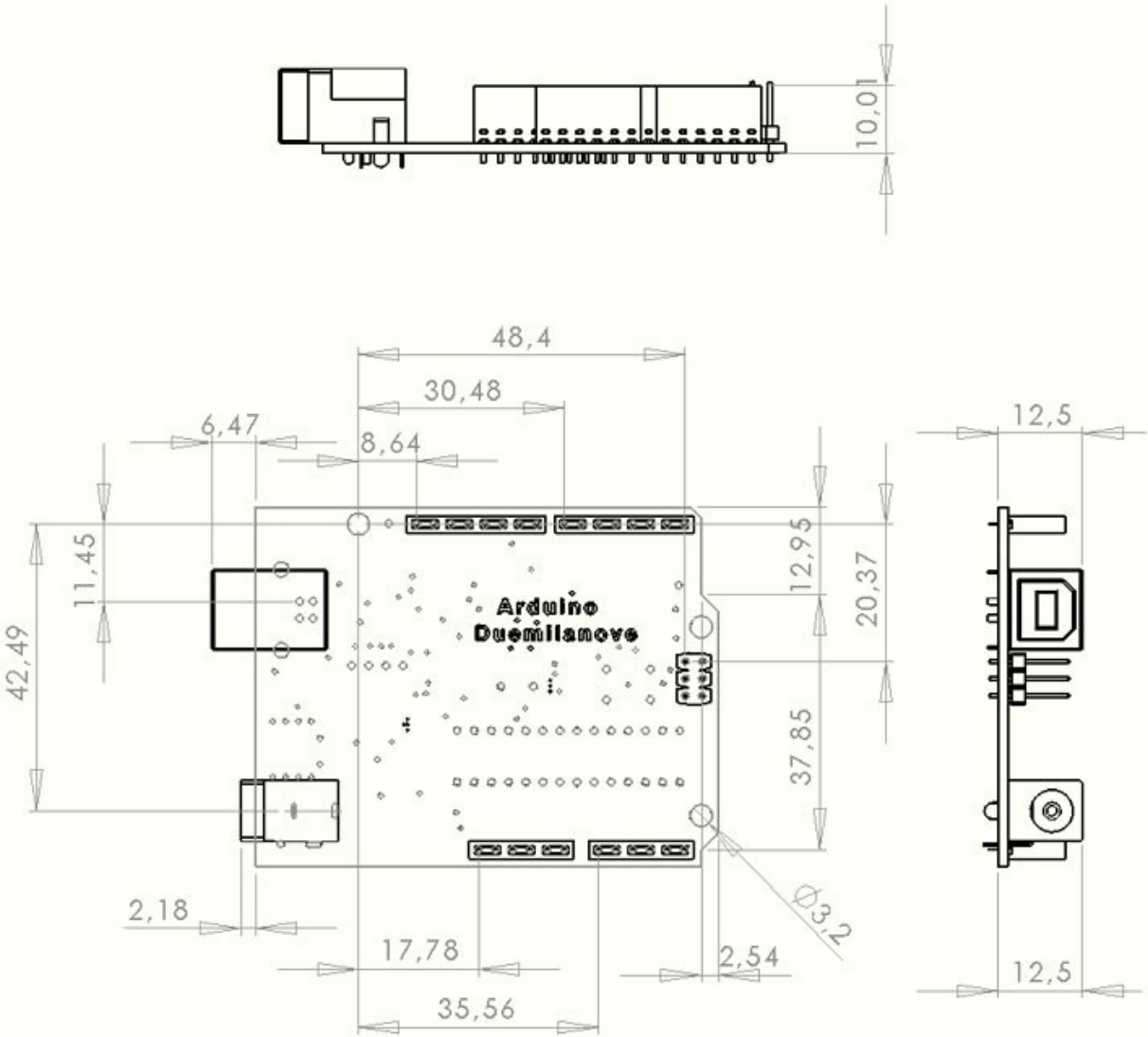


radiospares

RADIONICS



Dimensioned Drawing



radiospares

RADIONICS



Terms & Conditions



1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, or for any products that have been altered or modified in any way by a Customer. Moreover, The producer shall not be liable for any defects that result from Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producer shall have a reasonable time to replacements. Replaced products shall be warranted for a new full warranty period.

1.3 EXCEPT AS SET FORTH ABOVE, PRODUCTS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." THE PRODUCER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

1.4 Customer agrees that prior to using any systems that include the producer products, Customer will test such systems and the functionality of the products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above, and no additional obligations or liabilities shall arise from the producer providing such services.

1.5 The Arduino™ products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™ products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™ products which is solely at the Customer's risk, and that Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

1.6 Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products and any use of Arduino™ products in Customer's applications, notwithstanding any applications-related information or support that may be provided by the producer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilities and expenses it incurs to the extent directly caused by: (i) an actual breach by a Customer of the representation and warranties made under this terms and conditions or (ii) the gross negligence or willful misconduct by the Customer.

3. Consequential Damages Waiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

4. Changes to specifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.



Environmental Policies



The producer of Arduino™ has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.



radiospares

RADIONICS



Appendix B

MATLAB Code

```
%LQR optimal control m.file
```

```
Ts=0.01;%sampling time  
M=0.025; %ball mass  
Ib=4.3e-6; %ball moment of inertia  
R=0.02; %ball's radiu  
g=9.8 %gravity  
Tm=0.187;  
b=(M/(M+(Ib/R^2)));
```

```
% system state space
```

```
Ax=[0 1 0 0;0 0 -b*g 0;0 0 0 -1;0 0 0 0];  
Bx=[0 0 0 1/Tm]';  
Cx=eye(4);  
Dx=zeros(4,1);
```

```
Ay=[0 1 0 0;0 0 -b*g 0;0 0 0 -1;0 0 0 0];  
By=[0 0 1 0]';  
Cy=eye(4);  
Dy=zeros(4,1);
```

```
Q=[10 0 0 0;0 10 0 0;0 0 10 0;0 0 0 10];  
R=1;
```

```
%controller LQR
```

```
[K, P, E]=lqr(Ax,Bx,Q,R);
```

```

%state feedback mfile
Ts=0.01;%sampling time
M=0.025; %ball mass
Ib=4.3e-6; %ball moment of inertia
R=0.02; %ball's radiu
g=9.8 %gravity
Tm=0.187;
b=(M/(M+(Ib/R^2)));

% system state space

Ax=[0 1 0 0;0 0 -b*g 0;0 0 0 -1/Tm;0 0 0 0];
Bx=[0 0 0 1]';
Cx=eye(4);
Dx=zeros(4,1);

Ay=[0 1 0 0;0 0 -b*g 0;0 0 0 -1/Tm;0 0 0 0];
By=[0 0 0 1]';
Cy=eye(4);
Dy=zeros(4,1);

%controller

K=place(Ax,Bx,[-1.0 -0.99 -0.98 -0.999])
Ki=-K(:,4);

```


Appendix C

S-function

```

%The standard S-Function template written by MathWorks team

% the function rewritten to represent the Ball and Plate system
%for the sake of simulation & controller design
% the input for the system (torque1, torque2)
%the output for the system (xb,yb) respectively

function [sys,x0,str,ts,simStateCompliance] =Ball_Plate(t,x,u,flag)

switch flag,

    %%%%%%%%%%%
    % Initialization %
    %%%%%%%%%%%
    case 0,
        [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes;

    %%%%%%%%%%%
    % Derivatives %
    %%%%%%%%%%%
    case 1,
        sys=mdlDerivatives(t,x,u);

    %%%%%%%%%%%
    % Update %
    %%%%%%%%%%%
    case 2,
        sys=mdlUpdate(t,x,u);

    %%%%%%%%%%%
    % Outputs %
    %%%%%%%%%%%
    case 3,
        sys=mdlOutputs(t,x,u);

    %%%%%%%%%%%
    % GetTimeOfNextVarHit %
    %%%%%%%%%%%
    case 4,
        sys=mdlGetTimeOfNextVarHit(t,x,u);

    %%%%%%%%%%%
    % Terminate %
    %%%%%%%%%%%
    case 9,
        sys=mdlTerminate(t,x,u);

    %%%%%%%%%%%
    % Unexpected flags %
    %%%%%%%%%%%
    otherwise
        DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end

% end sfuntmpl

```

```

%
%=====
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=====
%
function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes

%
% call simsizes for a sizes structure, fill it in and convert it to a
% sizes array.
%
% Note that in this example, the values are hard coded. This is not a
% recommended practice as the characteristics of the block are typically
% defined by the S-function parameters.
%
sizes = simsizes;

sizes.NumContStates = 8;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1; % at least one sample time is needed

sys = simsizes(sizes);

%
% initialize the initial conditions
%
x0 = [0.08 0 0.08 0 0 0 0 0]; %xb,xb_d,yb,yb_d,alpha,alpha_d,beta,beta_d

%
% str is always an empty matrix
%
str = [];

%
% initialize the array of sample times
%
ts = [0 0];

% Specify the block simStateCompliance. The allowed values are:
% 'UnknownSimState', < The default setting; warn and assume DefaultSimState
% 'DefaultSimState', < Same sim state as a built-in block
% 'HasNoSimState', < No sim state
% 'DisallowSimState' < Error out when saving or restoring the model sim state
simStateCompliance = 'UnknownSimState';

% end mdlInitializeSizes

%
%=====
% mdlDerivatives
% Return the derivatives for the continuous states.

```

```

%=====
%
function sys=mdlDerivatives(t,x,u)

sys = [];
%states
%x(1)=xb;          x(3)=yb;          x(5)=alpha;   x(7)=beta
%x(2)=xb_d;        x(4)=yb_d;          x(6)=alpha_d; x(8)=beta_d;

%Model parameters
sa=sin(x(5)); % sin(alpha)
sb=sin(x(7)); %sin(bata)
ca=cos(x(5)); % cos(alpha)
cb=cos(x(7)); %cos(bata)
%ball parameters
mb=112*10^-3;
rb=1.5*10^-2;
Ib=(2/3)*mb*rb^2;
%plate parameters
mp=428*10^-3;
Ly=24.8*10^-2;
Lx=19*10^-2;
th=3*10^-2;
Ipx=(mp/12)*(Ly^2+th^2); %mass moment of inertia
Ipy=(mp/12)*(Lx^2+th^2); %mass moment of inertia

%the computation of alpha_dd & beta_dd (x6dot & x8dot)
%mass matrix
M=[(Ipx+Ib+mb*x(1)^2)/4.7 (mb*x(1)*x(3))/4.75; (mb*x(1)*x(3))/6.2 (Ipy+Ib+mb*x(3)^2)/6.2
];
C=[(2*mb*x(1)*x(2)*x(6)+mb*x(4)*x(1)*x(8)+mb*x(3)*x(2)*x(8)+mb*9.8*x(1)*ca)/4.75;(2*mb*x
(3)*x(4)*x(8)+mb*x(4)*x(1)*x(6)+mb*x(3)*x(2)*x(6)+mb*9.8*x(3)*cb)/6.2 ] ;
% end mdlDerivatives
A=inv(M)*(u-(C));
x1dot=x(2);
x2dot=(mb*(x(1)*x(6)^2+x(3)*x(6)*x(8))-(mb*9.8*sa))/(mb+(Ib/rb^2));
x3dot=x(4);
x4dot=(mb*(x(3)*x(8)^2+x(1)*x(6)*x(8))-(mb*9.8*sb))/(mb+(Ib/rb^2));
x5dot=x(6);
x6dot=A(1);
x7dot=x(8);
x8dot=A(2);
sys=[x1dot x2dot x3dot x4dot x5dot x6dot x7dot x8dot];

%
%=====
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time step
% requirements.
%=====
%
function sys=mdlUpdate(t,x,u)

sys = [];

% end mdlUpdate

```

```
%S
%=====
% mdlOutputs
% Return the block outputs.
%=====
%
function sys=mdlOutputs(t,x,u)

sys = [x(1) x(3)];

% end mdlOutputs

%
%=====
% mdlGetTimeOfNextVarHit
% Return the time of the next hit for this block. Note that the result is
% absolute time. Note that this function is only used when you specify a
% variable discrete-time sample time [-2 0] in the sample time array in
% mdlInitializeSizes.
%=====
%
function sys=mdlGetTimeOfNextVarHit(t,x,u)

sampleTime = 1; % Example, set the next hit to be one second later.
sys = t + sampleTime;

% end mdlGetTimeOfNextVarHit

%
%=====
% mdlTerminate
% Perform any end of simulation tasks.
%=====
%
function sys=mdlTerminate(t,x,u)

sys = [];

% end mdlTerminate
```

Appendix D

Arduino codes

```

// Servo motor test cod
#include <Servo.h>
#define XServoPin 1 // x-servo pin
#define YServoPin 2 // y-servo pin

Servo xServo; // Define x servo
Servo yServo; // Define y servo

int xservo_angle; // Desire xServo angle
int yservo_angle; // Desire yServo angle

void setup(void) {

xServo.attach(XServoPin); //attach x-servo
yServo.attach(YServoPin); //attach y-servo

Serial.begin(9600); }

void loop(void) {

xservo_angle=90; // Desire xServo angle
yservo_angle=90; // Desire yServo angle
xServo.write(xservo_angle); // servo angle
yServo.write(yservo_angle); //servo angle

delay(100);

Serial.print(xservo_angle); Serial.println(xservo_angle);
Serial.print("yservo_angle"); Serial.
println(yservo_angle);
}

```

```
// Touchscreen Test cod
```

```
#include "TouchScreen.h"
```

```
#define YP A1      // must be an analog pin,  
#define XM A2      // must be an analog pin,  
#define YM 1       // can be a digital pin or analog pin  
#define XP 2       // can be a digital pin or analog pin
```

```
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 20);
```

```
void setup(void) {
```

```
    delay(2000);  
    Serial.begin(9600);
```

```
}
```

```
void loop(void) {
```

```
    TSPoint p = ts.getPoint();
```

```
    InputX = p.x;  
    InputY = p.y;
```

```
    Serial.print("InputX"); Serial.print(InputX);  
    Serial.print("InputY"); Serial.print(InputY);
```

```
}
```



```
// PID test code
#include <PID_v1.h>

double SetpointX = 0, InputX = 0 , OutputX = 0;
double xKp = 0 , xKd = 0 , xKi = 0;

double SetpointY =0, InputY = 0 , OutputY = 0;
double yKp =0, yKd = 0 , yKi = 0;

PID xPID(&InputX, &OutputX, &SetpointX, xKp, xKi, xKd,
DIRECT);
PID yPID(&InputY, &OutputY, &SetpointY, yKp, yKi, yKd,
DIRECT);

void setup(void) {

  xPID.SetMode(AUTOMATIC);
  xPID.SetSampleTime(ts);

  yPID.SetMode(AUTOMATIC);
  yPID.SetSampleTime(ts);
  delay(2000);
  Serial.begin(9600);
}

void loop(void) {

  xPID.Compute();
  yPID.Compute();

}
```

```
// PID Autotune code
#include <Servo.h>
#include "TouchScreen.h"
#include <PID_v1.h>
#include <PID_AutoTune_v0.h>

#define YP A1 // must be an analog pin,
#define XM A2 // must be an analog pin,
#define YM 1 // can be a digital pin
#define XP 2 // can be a digital pin
#define XServoPin 3 //
#define YServoPin 4 //
#define sen 25

TouchScreen ts = TouchScreen(XP, YP, XM, YM, 10);

byte ATuneModeRemember=2;

double kpmodel=1.5, taup=100, theta[50];
double outputStart=5;
double aTuneStep=50, aTuneNoise=1, aTuneStartValue=100;
unsigned int aTuneLookBack=20;

boolean tuning = false;
unsigned long modelTime, serialTime;

Servo xServo;
Servo yServo;
unsigned long time;
unsigned long stable = 0;
unsigned int noTouchCount = 0;

double SetpointX = 200, InputX = 200 , OutputX = 80;
```

```

double xKp = 0 , xKd = 0 , xKi = 0;

double SetpointY = 150, InputY = 150 , OutputY = 90;
//double yKp =0.29, yKd = 0.055 , yKi = 4*yKd;
double yKp =0, yKd = 0 , yKi = 0;

PID xPID(&InputX, &OutputX, &SetpointX, xKp, xKi, xKd, DIRECT
PID yPID(&InputY, &OutputY, &SetpointY, yKp, yKi, yKd, DIRECT
PID_ATune aTune(&InputX, &OutputX);
boolean useSimulation = true;
void setup(void) {

if(useSimulation)
{
    for(byte i=0;i<50;i++)
    {
        theta[i]=outputStart;
    }
    modelTime = 0;
}
//Setup the pid
xPID.SetMode(AUTOMATIC);

if(tuning)
{
    tuning=false;
    changeAutoTune();
    tuning=true;
}

serialTime = 0;

xServo.attach(XServoPin);

```

```

xServo.write(80);
xPID.SetMode(AUTOMATIC);
xPID.SetSampleTime(40);

yServo.attach(YServoPin);
yServo.write(90);
yPID.SetMode(AUTOMATIC);
yPID.SetSampleTime(30);
delay(2000);
Serial.begin(9600);
}

void loop(void) {

    unsigned long now = millis();

    while (stable < 125)
    {
        TSPoint p = ts.getPoint();

        xServo.attach(XServoPin);
        yServo.attach(YServoPin);

        if(!useSimulation)
        {
            InputX =p.x;
            InputY =p.y;//
        }

        if(tuning)
        {
            byte val = (aTune.Runtime());
            if (val!=0)
            {
                tuning = false;
            }
        }
    }
}

```

```

}
if(!tuning)
{ //we're done, set the tuning parameters
    xKp = aTune.GetKp();
    xKi = aTune.GetKi();
    xKd = aTune.GetKd();
    xPID.SetTunings(xKp,xKi, xKd);
    AutoTuneHelper(false);
}
} else xPID.Compute();

if (val!=0)
{
    tuning = false;
}
if(!tuning)
{ //we're done, set the tuning parameters
    xKp = aTune.GetKp();
    xKi = aTune.GetKi();
    xKd = aTune.GetKd();
    xPID.SetTunings(xKp,xKi, xKd);
    AutoTuneHelper(false);
}
} else xPID.Compute();

if(useSimulation)
{
    theta[30]=OutputX;
    if(now>=modelTime)
    {
        modelTime +=100;
        DoModel();
    }
}
else
{

```

```

        xServo.write (OutputX) ;
    }
    if (millis () > serialTime)
    {
        SerialReceive ();
        SerialSend ();
        serialTime += 500;
    }

    xPID.Compute ();
    yPID.Compute ();

    Serial.print ("xki"); Serial.println (noTouchCount);
    Serial.print ("xkd"); Serial.println (noTouchCount);
    Serial.print ("xkd"); Serial.println (noTouchCount);

}

void changeAutoTune ()
{
    if (!tuning)
    {
        //Set the output to the desired starting frequency.
        OutputX = aTuneStartValue;
        aTune.SetNoiseBand (aTuneNoise);
        aTune.SetOutputStep (aTuneStep);
        aTune.SetLookbackSec ((int) aTuneLookBack);
        AutoTuneHelper (true);
    }
}

```

```

    tuning = true;
}
else
{ //cancel autotune
    aTune.Cancel();
    tuning = false;
    AutoTuneHelper(false);
}
}

void AutoTuneHelper(boolean start)
{
    if(start)
        ATuneModeRemember = xPID.GetMode();
    else
        xPID.SetMode(ATuneModeRemember);
}

void SerialSend()
{
    Serial.print("setpoint: ");Serial.print(SetpointX); Serial.
print(" ");
    Serial.print("input: ");Serial.print(InputX); Serial.print(
");
    Serial.print("output: ");Serial.print(OutputX); Serial.
print(" ");
    if(tuning){
        Serial.println("tuning mode");
    } else {
        Serial.print("kp: ");Serial.print(xPID.GetKp());Serial.
print(" ");
        Serial.print("ki: ");Serial.print(xPID.GetKi());Serial.
print(" ");
        Serial.print("kd: ");Serial.print(xPID.GetKd());Serial.
println();

```

```

        delay(20 );
    }
}

void SerialReceive()
{
    if(Serial.available())
    {
        char b = Serial.read();
        Serial.flush();
        if((b=='1' && !tuning) || (b!='1' &&
tuning))changeAutoTune();
    }
}

void DoModel()
{
    //cycle the dead time
    for(byte i=0;i<49;i++)
    {
        theta[i] = theta[i+1];
    }
    //compute the input
    InputX = (kpmodel / taup) *(theta[0]-outputStart) +
InputX*(1-1/taup) + ((float)random(-10,10))/100;
}

```



```
#include <Servo.h>
#include "TouchScreen.h"
#include <PID_v1.h>

#define YP A2 // must be an analog pin,
use "An" notation!
#define XM A3 // must be an analog pin,
use "An" notation!
#define YM A4 // can be a digital pin
#define XP A5 // can be a digital pin
#define XServoPin 3 //
#define YServoPin 4 //
#define near 35

Servo xServo;
Servo yServo;
unsigned long time;
unsigned long stable = 0;
unsigned int noTouchCount = 0;

double SetpointX = 185;
double InputX = 200;
double OutputX = 90;
double xKp = 0.6 ;
double xKd = 0.18 ;
double xKi = 0.71;

double SetpointY = 135;
```

```

double InputY = 150 ;
double OutputY = 90;
double yKp =0.6;
double yKd = 0.18 ;
double yKi = 0.71;

PID xPID(&InputX, &OutputX, &SetpointX,
xKp, xKi, xKd, DIRECT);
PID yPID(&InputY, &OutputY, &SetpointY,
yKp, yKi, yKd, DIRECT);
    TouchScreen ts = TouchScreen(XP, YP,
XM, YM, 20);

    void setup(void) {

        xServo.attach(XServoPin);
xServo.write(90);
        xPID.SetMode(AUTOMATIC);
xPID.SetSampleTime(50);

        yServo.attach(YServoPin);
yServo.write(90);
        yPID.SetMode(AUTOMATIC);
yPID.SetSampleTime(40);
delay(2000);
        Serial.begin(9600);
    }

    void loop(void) {

```

```

while (stable < 100)
{
    TSPoint p = ts.getPoint();
    if ( p.x != 0 && p.y != 1023 ) // Ball is
on plate
    {
        xServo.attach(XServoPin);
        yServo.attach(YServoPin);

        InputX = map(p.x, 125, 965, 0,400);
        InputY = map(p.y, 130, 910, 300, 0);//

        noTouchCount = 0;
        if ( (InputX > SetpointX - near && InputX
< SetpointX + near && InputY > SetpointY -
near && InputY < SetpointY + near ) )
        {
            stable++;
        }

        xPID.Compute();
        yPID.Compute();
        OutputY = map(OutputY,0,255,60,110);
        OutputX = map(OutputX,0,255,55,105);
        serialprint();
    } else
    {
        Serial.print(" No touch "); Serial.

```

```

println(noTouchCount);
    noTouchCount++;
    if (noTouchCount == 75)
    {
        noTouchCount++;
        OutputX = 90;
        OutputY = 90;
        xServo.write(OutputX);
        yServo.write(OutputY);
        delay(100);
    }
    if (noTouchCount == 150) //if there is no
ball on plate longer
    {
        xServo.detach();
        yServo.detach();
    }
}

xServo.write(OutputX);
yServo.write(OutputY);
}

xServo.detach();
yServo.detach();

while (stable == 100)
{
    Serial.print("Stable");
}

```

```

        TSPoint p = ts.getPoint();

    InputX = map(p.x, 125, 965, 0, 400);
    InputY = map(p.y, 130, 910, 300, 0);

    if ( ( InputX > SetpointX -near&& InputX <
SetpointX + near  && InputY > SetpointY - near
&& InputY < SetpointY + near) )
    {
        xServo.attach(XServoPin);
        yServo.attach(YServoPin);
    }else stable = 0;
}
}

void serialprint ()

{

    Serial.print("InputX"); Serial.
print(InputX);
    Serial.print("InputY"); Serial.
print(InputY);
    Serial.print("Xoutput");Serial.
print(OutputX);
    Serial.print("Youtput");Serial.
println(OutputY);

}

```

Appendix E

visual studio code

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
using System.Diagnostics;

namespace test1
{
    public partial class Form1 : Form
    {
        private String serData = "";
        private String speed;
        private String position;
        private int i;
        private String InputX;
        private String OutputX;
        private String InputY;
        private String OutputY;
        private String SetpointY;
        private String SetpointX;
        private String Notouch;
        private String xKp;
        private String xKi;
        private String xKd;
        private String yKp;
        private String yKi;
        private String yKd;

        public Form1()
        {
            InitializeComponent();
            SearchPorts();
            timer1.Start();
            serData= "0";
            i = 300;
        }
        void SearchPorts()
        {
            string[] port = SerialPort.GetPortNames();
            com_box.Items.AddRange(port);
        }
        private void connect_btn_Click(object sender, EventArgs e)
        {
            try
            {
                if (com_box.Text == "" || baud_box.Text == "")
                {
                    label13.Text = "No Connection Available";
                }
                else
                {
                    progressBar1.Value = 33;
                    serialPort1.PortName = com_box.Text;
                    progressBar1.Value = 66;
                    serialPort1.BaudRate = Convert.ToInt32(baud_box.Text);
                    serialPort1.Open();
                    connect_btn.Enabled = false;
                    circle_cont.Enabled = true;

                    point_cont.Enabled = true;
                    disconnect_btn.Enabled = true;
                }
            }
            catch { }
        }
    }
}

```

```

        send_btn.Enabled = true;
        xkp_text.Enabled = true;
        xki_text.Enabled = true;
        xkd_text.Enabled = true;
        ykp_text.Enabled = true;
        yki_text.Enabled = true;
        ykd_text.Enabled = true;
        label13.Text = "Connection Complete";
        progressBar1.Value = 100;
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
}

private void disconnect_btn_Click(object sender, EventArgs e)
{
    progressBar1.Value = 33;
    serialPort1.Close();
    connect_btn.Enabled = true;
    progressBar1.Value = 66;
    disconnect_btn.Enabled = false;
    send_btn.Enabled = false;

    point_cont.Enabled = false;
    circle_cont.Enabled = false;

    xkp_text.Enabled = false;
    xki_text.Enabled = false;
    xkd_text.Enabled = false;
    label13.Text = "Communication Disconncted";
    progressBar1.Value = 100;
}

private void serData_Rec(object sender, SerialDataReceivedEventArgs e)
{
    serData = serialPort1.ReadLine();
    this.Invoke(new EventHandler(data_display));
}

private void data_display(object sender, EventArgs e)
{
    serData = serialPort1.ReadLine();
    String length;
    length = serData.Length.ToString();
    if (System.Convert.ToInt32(length) > 5)
    {
        if (serData.Substring(0, 9) == "SetpointX")
        {
            SetpointX = serData.Substring(9, System.Convert.ToInt32(length) - 10);
            serData = "";
            SetpointX_lab.Text = SetpointX;
        }
    }
}

```



```

else if (serData.Substring(0, 9) == "SetpointY")
{
    SetpointY = serData.Substring(9, System.Convert.ToInt32(length) - 10);
    serData = "";
    SetpointY_lab.Text = SetpointY;
}
else if (serData.Substring(0, 7) == "Notouch")
{
    Notouch = serData.Substring(7, System.Convert.ToInt32(length) - 8);
    serData = "";
    System_lab.Text = ("Notouch");
}
else if (serData.Substring(0, 7) == "OutputX")
{
    OutputX = serData.Substring(7, System.Convert.ToInt32(length) - 8);
    serData = "";
    OutputX_lab.Text = OutputX;
    System_lab.Text = ("running");
}
else if (serData.Substring(0, 3) == "xKp")
{
    xKp = serData.Substring(3, System.Convert.ToInt32(length) - 4);
    serData = "";
    xkp_text.Text = xKp;
}
else if (serData.Substring(0, 3) == "xKd")
{
    xKd = serData.Substring(3, System.Convert.ToInt32(length) - 4);
    serData = "";
    xkd_text.Text = xKd;
}
else if (serData.Substring(0, 3) == "xKi")
{
    xKi = serData.Substring(3, System.Convert.ToInt32(length) - 4);
    serData = "";
    xki_text.Text = xKi;
}
else if (serData.Substring(0, 3) == "yKi")
{
    yKi = serData.Substring(3, System.Convert.ToInt32(length) - 4);
    serData = "";
    yki_text.Text = yKi;
}
else if (serData.Substring(0, 3) == "yKp")
{
    yKp = serData.Substring(3, System.Convert.ToInt32(length) - 4);
    serData = "";
    ykp_text.Text = yKp;
}
else if (serData.Substring(0, 3) == "yKd")
{
    yKd = serData.Substring(3, System.Convert.ToInt32(length) - 4);
    serData = "";
    ykd_text.Text = yKd;
}
}

```

```

}

private void send_btn_Click(object sender, EventArgs e)
{
    if (point_cont.Checked == true && circle_cont.Checked == false) { serialPort1.WriteLine("SM"); }
    else if (circle_cont.Checked == true && point_cont.Checked == false) { serialPort1.WriteLine("circle"); }
}

private void timer1_tick(object sender, EventArgs e)
{
    String length;
    length = serData.Length.ToString();
    if (System.Convert.ToInt32(length) > 5)
    {
        if (serData.Substring(0, 6) == "InputX")
        {
            InputX = serData.Substring(6, System.Convert.ToInt32(length) - 7);
            serData = "";
            InputX_lab.Text = InputX;
            this.chart1.Series["Cordinate"].Points.AddXY(i, System.Convert.ToDouble(InputX));
            i++;
            this.chart1.ChartAreas["ChartArea1"].AxisX.Minimum = i - 300;
        }
        else if (serData.Substring(0, 6) == "InputY")
        {
            InputY = serData.Substring(6, System.Convert.ToInt32(length) - 7);
            serData = "";
            InputY_lab.Text = InputY;
            this.chart2.Series["Cordinate"].Points.AddXY(i, System.Convert.ToDouble(InputY));
            i++;
            this.chart2.ChartAreas["ChartArea1"].AxisX.Minimum = i - 300;
        }
        else if (serData.Substring(0, 7) == "OutputX")
        {
            OutputX = serData.Substring(7, System.Convert.ToInt32(length) - 8);
            serData = "";
            OutputX_lab.Text = OutputX;
        }
        else if (serData.Substring(0, 7) == "OutputY")
        {
            OutputY = serData.Substring(7, System.Convert.ToInt32(length) - 8);
            serData = "";
            OutputY_lab.Text = OutputX;
        }
    }
}

private void chart1_Click(object sender, EventArgs e)
{
}

```

```
private void groupBox3_Enter(object sender, EventArgs e)
{
}

private void groupBox4_Enter(object sender, EventArgs e)
{
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
}

private void Form1_Load(object sender, EventArgs e)
{
}

private void ykd_text_TextChanged(object sender, EventArgs e)
{
}

private void OutputX_lab_Click(object sender, EventArgs e)
{
}

private void label19_Click(object sender, EventArgs e)
{
}

private void groupBox1_Enter(object sender, EventArgs e)
{
}

}
}
```