Acknowledgment

My sincerest gratitude is offered to my supervisor, Dr Hak-Keung Lam, he has been very supportive, patience from the outset of this project, I appreciate all his advice and encouragement throughout this project. Many thanks are also extended to the support of, Dr Chuang Liu for his guidance and support in the dynamic model of the balancing robot .I would like to acknowledge my family , friends and university colleagues for their support and encouragement throughout this project . Last but not least I would like to thanks the King's College Electronic laboratory assistants for facilitating me with the required equipment's on time.

Abstract

This paper will describe the implementation and design required for construction of two-wheeled self-balancing robot .the dynamic behaviour of this system is similar to the non-linear, unstable classical control problem of the inverted pendulum. This paper will investigate the use of sensor fusion technology by using Kalman filter to measure tilt angle of the robot by combining the accelerometer and gyroscope signal to provide accurate estimation by eliminating the drift of the gyroscope and noise of the accelerometer .this paper delve into the performance and suitability of the Proportional Integral Derivative (PID) controller, and there will be various approaches and techniques applied to improve the stability .the robot will utilize an linear state space controllers namely full state feedback controller using a mathematical model to derive the parameters of the robot. The robot will use an Arduino board which holds an Atmega328 microcontroller which requires various communication protocol such as SPI , I2C and PWM to communicate with the electronic devices .the control algorithm will be implemented in C++ programming language and data will be simulated using different software platform such PLX-Excel and processing , Matlab will be used to evaluate the states of the state feedback controller .The Proportional Integral Derivative (PID) controller will be used to stabilise the robot in its equilibrium position ,the gains of the controller will be tuned manually by Ziegler-Nicholas technique ,the gains of the controller can be varied using a potentiometer and the real time data can be monitored on LCD screen .An android voice recondition application was designed and developed for the trajectory control of the robot.

Statement of contribusion

Various sensor fusion techniques have been implemented to address the problem of angle measurement. A Digital Motion Processor (DMP) and a linear Kalman filter were successfully implemented to eliminate the noise of the accelerometer and the drift of the gyroscope sensor. To ensure the reliability and effectiveness of the angle estimation, the signal information was plotted using real time PLX-Exect graphs and a processing cube test to observe the reaction of the MPU6050 based on the calculations.

Two linear control strategies were implemented to address the problem of the robots' stabilization and robustness. A control strategy was developed by using a state space feedback model. The dynamics were derived based on the Newton's second law of motion and the controller was implemented based on the linearized model derived from mathematical equations around the operating point. The linearized model uses the Matlab function to obtain the gains of the full state feedback controller.

A Proportional Integral Derivative (PID) algorithm was implemented to balance the robot in its equilibrium position. The parameters of the PID were obtained based on the manual tuning and the Ziegler-Nicholas techniques. The PID feedback controller was improved by introducing a position control element to the feedback loop. This was done by adjusting the set-point based on the position of the robot from its balancing position.

An Android MIT application inventor was designed and developed in order to send voice commands to the robot. This feature has been successfully implemented and tested to control various electronic devices such as displaying the required data on LCD and being able to control the movement of the robot by voice commands. However to achieve a desirable trajectory control , the current motors should be replaced with higher torque and speed motors.

Nomenclature

ADC	Analog to Digital Converter
DMP	Digital Motion Processer
EMG30	Encoder Motor Gearbox 30:1
I/O	Input Output
I^2 C/IIC	Inter Integral Circuit
IMU	Inertia Measurement Unit
LCD	Liquid Crystal Display
LQR	Linear Quadratic Regulator
LiPo	Lithium Polymer
PID	Proportional Integral Derivative
PWM	Pulse Width Modulation
SCL	Serial Data Clock
SDA	Serial Data Line
SPI	Serial Peripheral Interface
SBR	Self-balancing robot
$\ddot{\emptyset}$	Wheel angular acceleration
$\dot{\emptyset}$	Wheel angular velocity
$\mathrm{F_a}$	Force acting on the x-axis of the pendulum attached to the wheels
$\mathrm{F_p}$	Force acting on the y-axis of the pendulum attached to the wheel
$\mathrm{I_p}$	Inertia of pendulum
$\mathrm{m_p}$	Mass of pendulum
$\mathrm{r_p}$	Distance between centre of gravity and middle of wheels
$\mathrm{T_m}$	Torque applied by the motors
F_f	Friction force applied to the wheel from the ground
F_w	Normal force applied to the wheel
I_w	Inertia of the wheel
m_w	Mass of the wheel
r_w	Radius of wheel
$\ddot{\theta}$	Pendulum angular acceleration
$\dot{\theta}$	Pendulum angular velocity
$\emptyset$	Wheel angle
g	Gravity acceleration
θ	Pendulum angle

Contents

List of Figures

List of Tables

1 Introduction

1.1 Project Context

Research on mobile inverted pendulums has gained momentum over the past few years. This is due to the unstable dynamic behaviour of a self-balancing robot, which comes from the classic behaviour of an inverted pendulum. The systematic behaviour of a mobile inverted pendulum can be approached in different ways, which can either be used in complex or simple applications. These robots have the ability to balance on two wheels and navigate through different directions, traverse small ramps and spin around on a spot. This manoeuvrability can solve a number of challenges both in society and industry, such as mobility wheelchairs, balancing scooters. Utilising this technology would allow an alternative approach for human transportation systems.

This project investigates the use of sensor fusion technology by using a Kalman filter to measure the tilt angle of the robot by combining the signals of the accelerometer and gyroscope sensors to provide accurate angle estimation. This project will delve into the performance and suitability of the Proportional Integral Derivative (PID) controller, and there will be various approaches and techniques applied to improve the stability of the Self Balancing Robot (SBR). The robot will utilise linear state space controllers, namely a full state feedback controller using a mathematical model to derive the parameters of the robot. An Inertia Measurement Unit (IMU) sensor will be used to measure the inclination angle and an encoder sensor will be used to measure the position of the robot. The robot will be interfaced with an android application receiving a voice commands.

1.2 Project motivation

The motivation behind this project firstly stems from the public interest in balancing vehicles such as the Segway and the balancing unicycle, while also considering the impact that these vehicles can have on the environment. For instance, the high demand of the European Union regulations for the reduction of CO_2 emission can provide a use for these vehicles. For populated cities such as London with a population of 8.6 million, these environmentally friendly transport devices can be considered as an alternative way of transportation, which releases very little or no carbon dioxide. Secondly having looked at various stability projects such as nBot, Balanduino and the balancing unicycle, the dynamic behaviour of these robots fascinated my interested from a personal perspective. More specifically, the usefulness of the algorithm and the control theory required for the design and construction of the balancing robot to control its stability and movement can be applied in many other applications in industry.

1.3 Literature review

Conducting and reviewing related work prior to the outset of the project is critical as this will provide researchers with an in depth knowledge of the available technology required and methodology of other researchers on the area. This section will provide brief summary of the related work on key topics related to self-balancing robot.

1.3.1 Balancing Robots

There have been many contributions in the design and development of the inverted pendulum, which is one of the well-known topics in the field of control theory. The uniqueness of this unstable system has drawn much interest from robotics enthusiasts and researchers around the world. There has been a wide range of applications designed and developed by researchers which apply the concept of inverted pendulum to solve problems such as humanoid robots, personal transport vehicles and balancing wheel chairs.

In July 2001, Dean Kamen introduced a new approach to personal transport called the Segway, which adapts the dynamic behaviour of inverted pendulum. The Segway consists of a platform attached to two wheels, providing a space for the user to stand on. The passenger act as the inverted pendulum while the platform acts as the cart allowing the passenger to navigate the Segway by leaning towards the direction in which they wish to travel.[1]

Two years later, in October 2003, another form of transport system was introduced by Dean Kamen known as the iBot. This is a balancing wheelchair designed specifically for disabled people. This mobility device has the ability to climb up curbs and stairs while balancing. Since Dean Kamen introduced his solution to the field of inverted pendulums, the areas has gained momentum both in the research into balancing personal transporters and the public interest. [3]

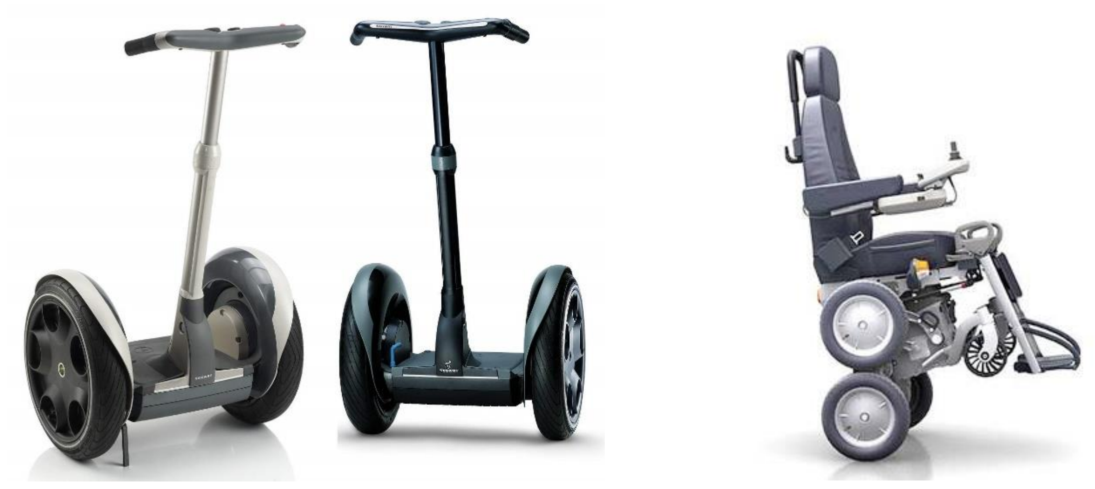


Figure 1 Segway and iBot mobility system [2][3]

Researchers at the Swiss Federal Institute of Technology, have developed a prototype of a two wheel balancing vehicle using a Digital Signal Processor (DSP) controller board for its implementation. This robot was developed based on the non-linear concept of the inverted pendulum. As shown in figure 2, a weight is attached on the top of the robot to simulate the human driver. The system adapts the linear space controller which uses the sensor measurement from the encoders and gyroscopes to control its stability while moving. The robot uses a radio receiver in order to communicate with the microcontroller [2].

Another well-known contribution to self-balancing robots is the ‘nBot.’ The research and development of this prototype was carried out by David Anderson at the University of Southern Methodist, which uses various sensors such as accelerometers, gyroscopes, Electro-Optical sensors and Proximity Detectors. These are used to preform angle estimation which uses an odometer technique to improve the stability of the robot while the robot is moving on low friction surfaces.[3]

There have been many contributions in the dynamic behaviour of the inverted pendulum. On a higher level, these models are used on humanoid robots, which capture the human motion using a balance controller which uses two strategies to perform stability. Firstly, the ankle strategy, where the robot performs balancing based on the inverted pendulum. Secondly, the Hip strategy uses gravitational force to maintain stability. One of these humanoid robot was designed by Sugihara, which stabilises by controlling the centre of mass of the robot by an in-direct manipulation of the Zero Moment Point using the Hip strategy technique, which is based on the real time generation model. [4]

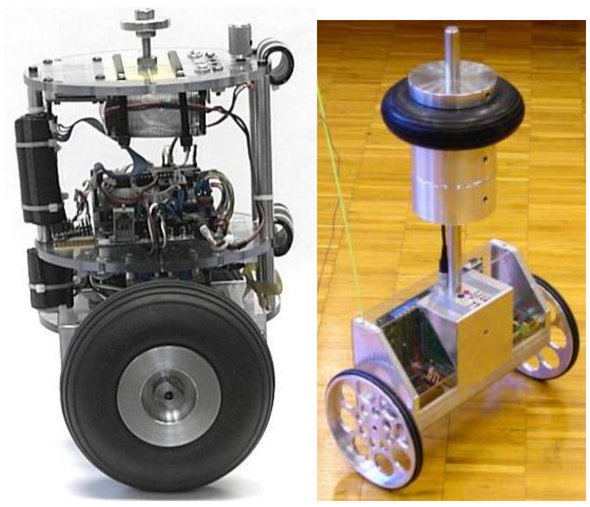


Figure 2 nBOT and JOE [1], [2]

1.3.2 Control systems

It is crucial to implement an appropriate control system to maintain the stability of the robot. While there are numerous control system strategies available that can be adopted to perform this task, this project aims to use a control model to achieve stability, without sacrificing the reliability and robustness of other control strategies. The main difference in the implementation of the control system algorithm, depends on two main factors: the system model derivation and obtaining information based states of the model. There are two types of control strategy used in such an application, which are classified as linear and non-linear control models. In the linear model, the system dynamic is linearized about an operating point. The linear model on its own is sufficient to balance the robot in the equilibrium position. The nonlinear model requires the unscathed dynamic model in order to design the system controller. The nonlinear model has the advantage of providing a more stable and robust system. However, researchers tend to use linear model when designing a controller due to the complexity and difficulties of the implementation required for the nonlinear model.

A literature review found that most of the controller implementations are nonlinear for stability control problems such as rotatory inverted pendulums and inverted pendulums on a cart. In regards to [5], a model of an inverted pendulum was developed to demonstrate the ability of the non-linear controller for an unstable system. Paper [6] describes the design and development of an inverted pendulum attached to a cart using a Fuzzy Logic control strategy based on the knowledge of control and approximate reasoning. The results obtained from the simulation prove that the system can maintain stability using both control models. However, there is no evidence of practical implementation in order to verify the results and findings. The researchers in paper [9] used the concept of an inverted pendulum to design a robotic arm which can pick and place using a multi I/O sliding controller. Most research, including the ones mentioned previously are predominately built and non-mobile, as the non-linear controller requires high computational power.

Linear model controllers are widely used among researchers when implementing similar applications such as JOE. Proportional Integral Derivative (PID), Linear Quadratic Regulators (LQR) and Linear state space control are the well-known control mechanisms used in balancing robots. The implementation and design for these controllers can be seen in [7-10].

Research paper [7], provides a comparison for the design of a generic algorithm PID controller and conventional PID controller. The result provided by this paper shows that both controllers can serve as an effective and valuable controller for an inverted pendulum application. However there is improvement in the result of the generic algorithm PID controller when the rise time and settling time is considered in the implementation. Another research paper [10], demonstrated the improvement in the result of combining the PID and LQR controllers.

1.3.3 Kalman Filter using Sensor Fusion

A paper published by Rudolf Kalman provided new approach to data prediction and filtering [11]. This research intended to provide a solution to the limitation of Weiner Hop filters for problems of a statistical nature. A Kalman filter is a combination various mathematical equations that provide efficient results in data acquisition. The advantage of using this filter compared to other filters is that it provides a prediction process, which estimates the states for present, past and future, without requiring knowledge of the optimum states of the system model.

It is crucial to provide reliability and accuracy when measuring the inclination angle of the balancing robot. A sensor fusion technique is used to combine the data collected from various sensors and provide an accurate estimation. To improve the reliability of the system with multiple sensors, a Kalman filter is designed based on the sensor fusion technique. The accuracy and reliability of this approach has inspired many to adapt sensor fusion for better performance.

A paper published by Durrant and Barshan [12], describes the performance of navigation systems in robots. It improves the performance by using the sensor fusion technique to combine the signal of multiple sensors and outlines the benefits of adding a Kalman filter in similar applications. Paper [13], describes the design of 'Gyro-odometery' which combines the data from odometers and gyroscopes in an autonomous robot. This technique reduced the errors occurring in the robot when the robot slips on low friction surfaces.

1.3.4 Conclusion

Based on the review litrture we conclude that it is curcial to use an appropriate control strategy to maintain the stability of the robot. While there are numerous control system strategies available that can be adopted to perform this task, this project aims to use a control model to achieve stability, without sacrificing the reliability and robustness of other control strategies. This project will use linear control model due to complexity high computation power required for non-linear control models .

It is important to consider the reliability and accuracy when measuring the inclination angle of the balancing robot. A sensor fusion technique will be used to combine the data collected from various sensors and provide an accurate estimation. To improve the reliability of the system with multiple sensors, a Kalman filter is designed based on the sensor fusion technique. This technique is preferred due to high level of accuracy and the result observed based on the published papers .

1.3 Aims and objective

The main aim of this project is to construct and design a mobile inverted pendulum that can balance on its two wheels, using the concept of sensor fusion to measure the inclination angle, and then design a closed loop feedback controller to control the movement and direction of the robot. This project has number of objectives to achieve its aims, which are listed as follows:

- Make use of appropriate hardware components to measure the states of the controller. This includes measuring the angle, angular velocity and position of the robot.
- Research and investigate the behaviour of the PID controller based on a manual tuning approach. This will include various simulations and testing in order to monitor the behaviour of each parameter.
- Make use of block diagrams to design the controller software implementation required for the feedback controller.

- Provide white-box testing for the software implementation, including various white box testing techniques such as unit testing.
- Provide Black-box testing, in order to observe if the robot is functioning as expected.

1.4 Report structure

Chapter One : This chapter will outline the aims and object and technical problem and the challegs to faces these problem will be descried in this chapter .adding to this the litreture review is also presented in this chapter .

Chapter Two : This chapter will contain the background knowledge for the design and implementation of the stability robot.

Chapter Three : This chapter describe the mechanical and electronic design and implementation required to construct the self-balancing robot.

Chapter Four : This chapter use the Newton's second law of motion to drive the mathematical model in order to design linear state feedback controller .

Chapter Five : This chapter describe the control strategies used in this project using block diagram.

Chapter Six : This chapter will describe the main implemation required for the balancing robot.

Chapter Seven : this chapter describes the result and exeriments carried out in this project , proving a graphical representation of data.

Chapter Eight : This section will provide summary of the project contributions and achievements and will discuss the possible future work that can be carried out to improve the robot.

2 Background knowledge

2.1 Introduction to self-balancing robot

The concept of stability robot arises from the non-linear behaviour of the inverted pendulum. As it can be seen in figure 3, the robot will maintain stability as wheels remain under the point where the centre of gravity of the robot is positioned, therefore the wheels apply torque in the direction where the centre of graving is moving .hence to preform stability there are two main approach that has to be considered , firstly we require to know the inclination angle ,secondly we require to measure the required torque the motors need to adjust it position.to achieve this we require to implement an algorithm to combine the values of various sensors by using the concept of sensor fusion .having measured the angle an controller feedback system will be designed to control the direction and speed of the wheels. This section will contain the background knowledge for the design and implementation of the stability robot.

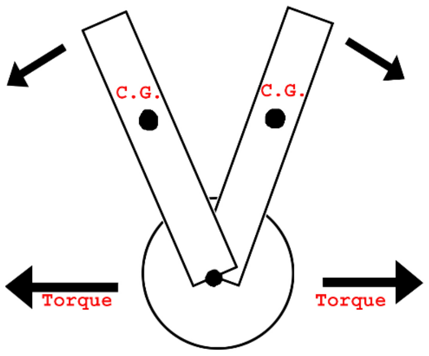


Figure 3 wheels should be under the centre of gravity

2.2 Estimating the angle of inclination by an Accelerometer

As mentioned previously we require to measure the angle in order to control the robot in its equilibrium position. This section will describe the calculation required for the accelerometer sensor to estimate the angle of the robot. Figure 4 represent the accelerometer sensor reading while the robot is in a stationary position assuming the accelerometer sensor is placed on the top of the robot.

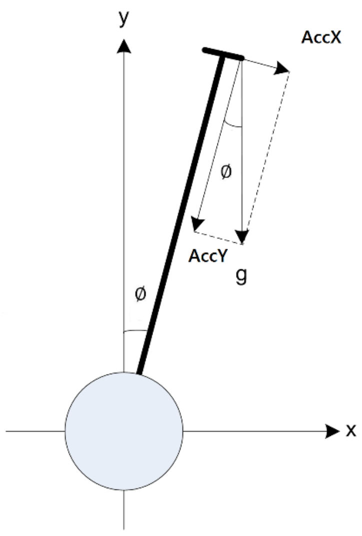


Figure 4 Accelerometer inclination angle data

Equations (2.1) and (2.2) represent the calculation required to measure the angle of inclination using the accelerometer sensor, where $AccX$ represent the x-axis value of the accelerometer and $AccY$ represent y-axis value of the accelerometer. When the robot is placed in its stationary position, the gravitational constant (g), is converted into acceleration constant form. Equation (2.1) calculates the inclination angle of the robot, when robot is in a stationary position.

$$\emptyset = \tan^{-1}\frac{AccX}{AccY} = \sin^{-1}\frac{AccX}{\sqrt{AccX^2+AccY^2}} = \sin^{-1}\frac{AccX}{g} \qquad (2.1)$$

In the case of balancing robot, we are interested in angle estimation when the inclination angle is small, therefore we require to ensure there is minimal deviation from the equilibrium position. Therefor we further simplify equation (2.1) which will result in equation (2.2):

$$\emptyset \approx \sin\emptyset = \frac{AccX}{g} \qquad (2.2)$$

In (2.2), the inclination angle can be estimated using the x-axis of the accelerometer sensor reading, this angle estimation is based on stationary movement of the robot. In reality the balancing robot accelerates towards the direction it tends to fall, this causes x-axis of the accelerometer value will be slightly more than $AccX$ on the other hand when the robot is in its equilibrium position the y-axis of the accelerometer value would be less than $AccY$.In conclusion when the accelerometer is placed on the centre of the gravity and the robot is in its equilibrium position, the x-axis acceleration is negligible. Therefore measuring the angle based on (2.2) will contain some error, however it is a good approach to approximate the angle of inclination.

In a practical level, the accelerometer sensor is considered to be very sensitive when it gets effected by vibration and movement. This will result in an error in the final angle estimation, which cannot simply be eliminated. To overcome this issue we require to use another type sensor, to compensate from the weakness of accelerometer sensor to provide more reliable angle estimation.

2.3 Estimating the angle of inclination by a Gyroscope

Gyroscope is another tilt sensor used to measure angular velocity taking place over a period of time. In (2.3), $\emptyset(t)$ represent the angular velocity and $Gyr(t)$ represent the gyroscope reading. The value of $Gyr(t)$ can be considered as constant when the time interval is small. (2.3) can be approximated as (2.4), when the reading of $Gyr(t)$ is measured in a small time interval.

$$\emptyset(t) = \int_{t0}^{t1} Gyr(t)dt \tag{2.3}$$

$$\emptyset(t) = \emptyset(t0) + Gyr(t)(t1 - t0) = \emptyset(t1) + Gyr(t)\Delta t \tag{2.4}$$

The gyroscope sensor measurement is based on angular position and is less susceptible to acceleration and vibration, which effect the accelerometers reading. However the gyroscope angular velocity measurement is accumulative and the error produced by this sensor deviates from the actual angle estimation .This effect of the gyroscope sensor is also known as the drifting effect .As a result the gyroscope sensor cannot be a reliable sensor on its own when measuring the angle of inclination.

2.4 Sensor fusion

As mentioned previously the accelerometer sensor gets affected by vibration and noise and the gyroscope sensor drift over time. To address this issues we require to combine the measured value of both these sensor in a meaningful way by using the strength of each sensor in order to compensate from the weaknesses of each sensor. Therefore sensor fusion can result in accurate angle estimation than using individual sensor to estimate the angle.

In a classical sensor fusion approach, the complimentary traits of the sensor fusion can provide long term accuracy by the accelerometer sensor when there is no vibration or acceleration to the sensor and short term accuracy provided by the gyroscope sensor as drift accumulates over time. These are the fundamental consideration, when combining the values of both sensor.

Another technique to combine the reading of two sensors are use of data filtering, whereas data filtering is a different approach to sensor fusion. However sensor fusion can be done independent or in conjunction to filtering approach. Filters commonly consider the added white Gaussian noise (AWGN) in there measurement.

2.5 Kalman filter

Kalman filter is an adaptive approach, which can be used to filter accelerometer and gyroscope sensor's readings. In order to achieve an optimal Kalman for an accurate measurement we require to consider the system model and also require to measure the noise covariance matrices. It is possible to use the filter without measuring the Kalman filter parameters, however without the system model the optimal sense will not be evaluated.

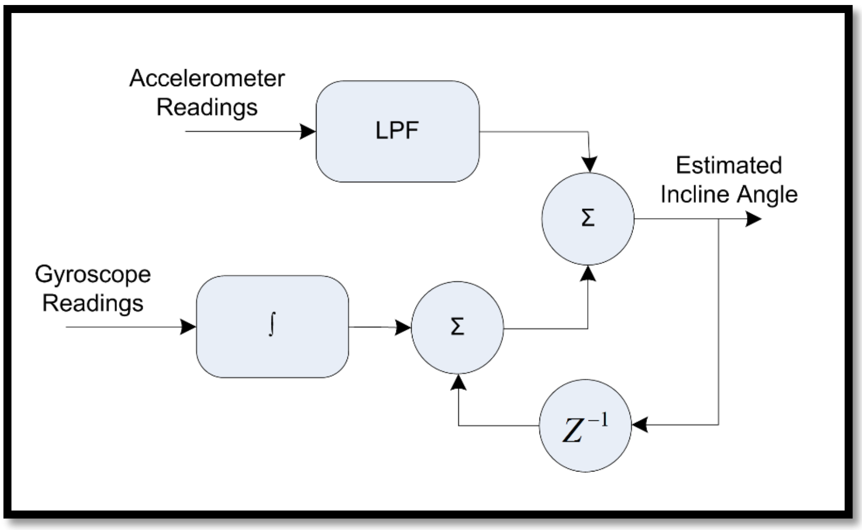


Figure 5 Sensor Fusion Block Diagram

Figure 4 ,is the block diagram for Kalman filter , which uses a low pass filter for the accelerometer data in order to reduces the noise produces by averaging data over period of time and the readings of the gyroscope sensor are integrated and combined to the previous gyroscope measurement ,to evaluate the current angle .

$$\begin{cases} \emptyset_{kal|t1} = X.\left(\emptyset_{est}\middle|t_0 + Gyr(\Delta t)\right) + Y.\emptyset_{AccX} \\ X + Y = 1 \end{cases} \tag{2.5}$$

In (2.5), $\emptyset_{kal}$ represents the calculation required for Kalman filter, where $Gyr(t)$ is the gyroscope sensor reading and $AccX$ is the accelerometer sensor reading, which is measured in equation (2.2) and (2.3). X and Y are the parameters of the Kalman filter , the parameters should be selected in way that the gyroscope drift can be reduces so the noise produced by the accelerometer sensor does not cause the inclination angle estimation to be varied significantly over each iteration. When selecting the weight values for the parameter X and Y, the gyroscope reading should have a higher weight than the accelerometer sensor this is due to the noise produced by the accelerometer sensor which will be recovered by the gyroscope drift. Kalman Filter linearization form is also known as complementary filter, which is used for various sensor fusion application.

2.6 Introduction to closed loop feedback controller

To provide accuracy and certainty in a control application, we require to adapt a precise control level to adjust the state of the system. It is possible to control the movement of the actuator directly by the sensor using an open loop technique. However to the adjust the condition of the system to perform an ideal reaction, we require to use a closed loop control approach .figure 4 represent the block diagram for an feedback controller . Where the error is difference between the measured set point and measured signal. There error produced by the system will be reduced and eliminated by the controller signal which will be amplified and sent to the actuator in order to make the correction.

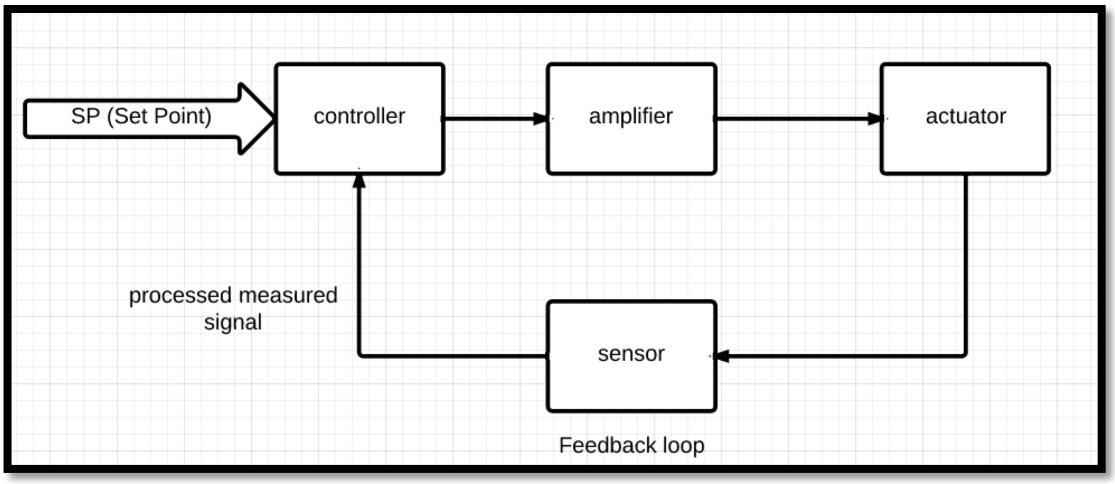


Figure 6 closed loop feedback controller

2.6.1 PID Feedback Controller

The Proportional, Integral, Derivative (PID) controller is feedback closed loop mechanism wildly used in many industrial control applications. The PID feedback controller measures the error signal and processes it into a descried set-point (SP). The controller process the input signal to reduce the error using the manipulation variable adjustment. Figure 7, represent the block diagram for the PID control loop.

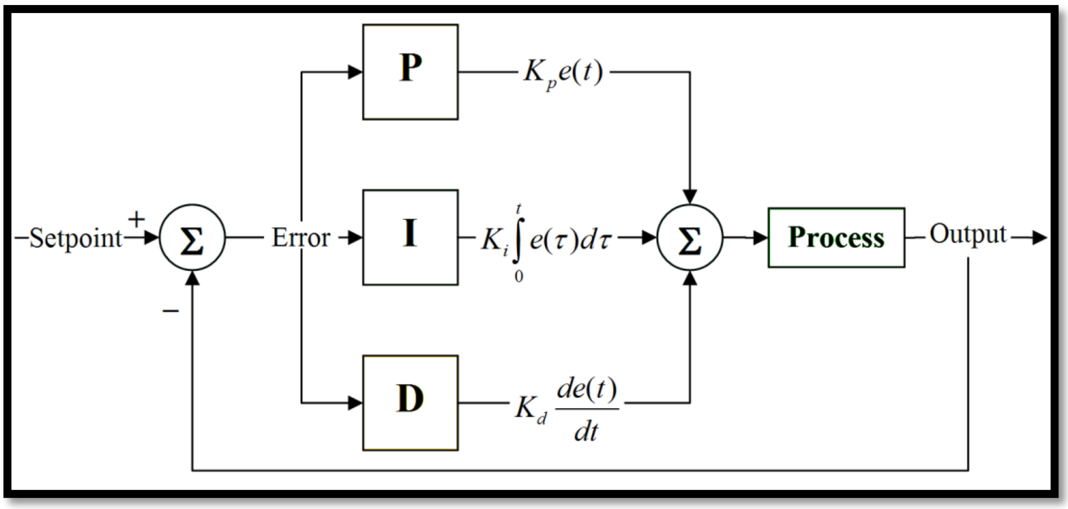


Figure 7 PID feedback controller

PID algorithm consist of three constant parameter which are referred as k_p, k_i, k_d also as constant gains, the three terms are referred as P , I and D. The proportional term is proportional to the current error, the integral term is proportional to the accumulation of error over period of time which is also known as past error, and the derivative term is proportional to the future error based on change of rate. Equation (2.6), (2.7) and (2.8) represent the calculation required to derive the P , I and D term of the PID algorithm.

$$P = k_p \times e(\tau) \qquad (2.6)$$

$$I = k_i \int_0^t e(\tau)d\tau \qquad (2.7)$$

$$D = k_d \frac{de(t)}{d(t)} \qquad (2.8)$$

Combining these three equation will be used to adjust the system process by a control element, which provides the stability of the pendulum. The PID controller equation is represented in equation (2.9).however in some application one or two of these terms will be used in order to provide a system control, this will be achieved by setting the gains of other terms to zero. A PID controller can be used as PD, PI, I or P controller depending on the application.

$$PID = k_p \times e(\tau) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de(t)}{dt} \qquad (2.9)$$

3 Mechatronic system

This section describe the mechanical and electronic design and implementation required to construct the self-balancing robot. There will be a brief description of each hardware component used in this project providing details of the schematics and communication protocol required to interface each component to the Atmega328 microcontroller.

3.1 Mechanical system

The mechanical construction of the robot is divided into two part the upper body which act as the inverted pendulum and the wheels which maintains the robots stability. The general design of the robot is rectangular prism which is attached to two wheels placed in parallel position to each other .the main frame was printed using a laser cutter and the design was carried out using an InkScape software providing a specific place for wire connections and components , which can be viewed in appendix [E]. Figure 8, is the final mechanical design of the balancing robot.

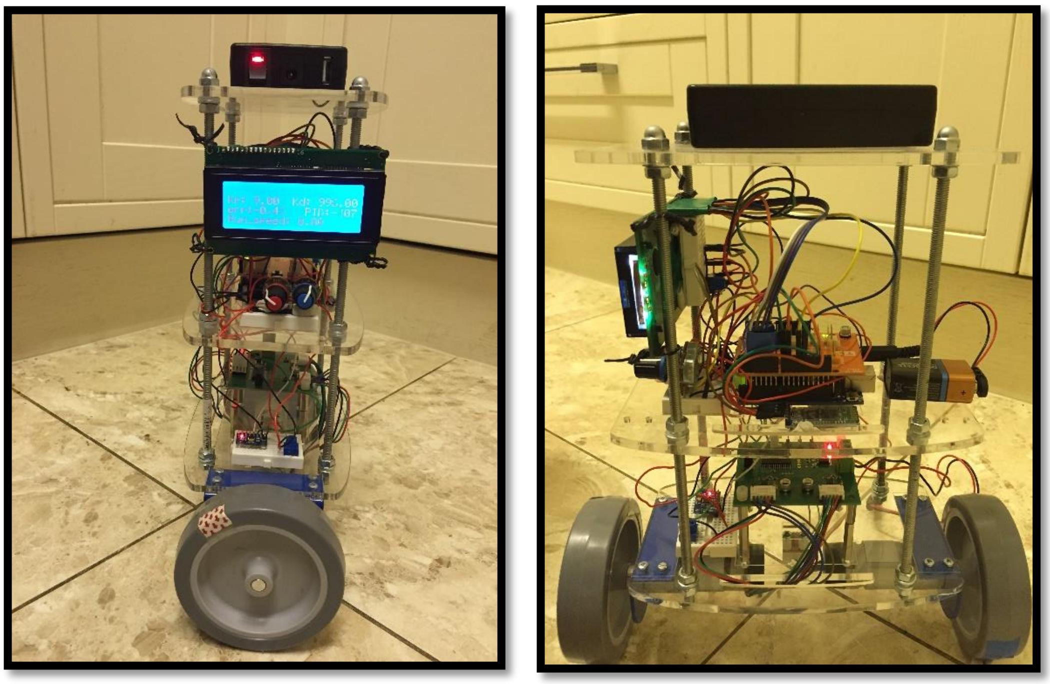


Figure 8 Front and Side view of the balancing robot

3.1.1 Upper body –Frame

the body consist of four 5mm metal rods which attach the layers of the robot together, the layers are 4mm clear plastic acrylic so the wires and component could be viewed from different direction .the reason for selecting 4mm thick plastic is to reduce the vibration effect on the whole body as this could affect the angle measurement as accelerometer sensor gets effected by vibration. There is position specified for each component, in order to improve the effectiveness and reliability of the system. The MPU6050 sensor is placed bellow the first layer near the axis of rotation as this improves the tilt measurement provided by the IMU. The battery is placed on the third layer in order to increase the height of the centre of gravity which is desirable. The microcontroller and the LCD is placed on the second layer, and the MD25 is placed above the first layer. From the equation derived in section 4 the acceleration is proportional to mass and inversely proportional to height. Therefore by increasing the height of the robot the angular acceleration of the pendulum will reduce and increasing the weight of the pendulum will increase the angular acceleration. Therefore to achieve a disable angular acceleration the mass and height of the robot was considered in the mechanical construction.

3.1.2 Lower body –Wheels

The lower body consist of two 'robot-electronics' wheels with diameter of 100mm.the reason for selecting these wheels is due to the light weight and the size of the radius which is suitable to provide a descried speed and angular velocity. The wheel is made out of 26mm wide rubber tread and consist of 5mm hub ,which can easily attach the wheels to the EMG30.the EMG30 is attached to a bracket the EMG30 to the plastic frame of the robot.

3.2 Electronic system

The following section will describe the electronic design and implementation carried out for the stability robot. Which will be divided into two section main section:

a) The hardware component.
b) The schematic of electronic circuit.

3.2.1 Microcontroller

The Arduino is prototype board which provides hardware and software implementation. The board is designed to ease the hardware connection for the user. The microcontroller used on the Arduino Uno is a 32-bit Amega328 processor which consist of various serial communication ports, consisting of 14 digital I/O which are used for PWM and SPI and Serial communication on the digital pin and 6 Analog pins with a SDA/SCL proving an IIC connection. The Arduino software platform or Integrated Development Environment (IDE) supports programming languages such as Java, C/C++.The readings of the sensors and the single values can be observed by Arduino serial monitor .The communication port can be interfaced with various software's such as PLX-excel, Processing and Matlab to plot the real time output data.

The Arduino was used in this project due to ease of connection and portability which can be simply interfaced with the computer by a USB cable and provides simple hardware connection and an appropriate communication, where the sensors and devices can be powered up by the Arduino without requiring an external power supply to each device .

3.2.2 MPU-6050 Accelerometer and Gyroscope

The MPU60650 contains a MEMS gyroscope and accelerometer sensor on a single chip, which provides 16 bit Analog to Digital Converter (ADC) for each hardware channel, which captures the X, Y and Z signals at the same time. This sensor requires the IIC port of the Atmega328.The board contains an on board processor chip "Digital Motion Processor" where the software implementation is carried out with a firmware and can manage complex algorithm with the sensor output signals without requiring to use the Arduino processors speed. Figure 9 represent the semantic required for the MPU6050.

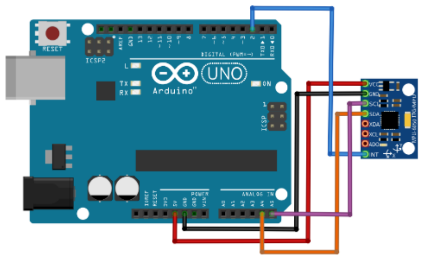


Figure 9 Schematics MPU6050

3.2.3 Inertial Measurement Unit Digital combo board

The Sparkfun digital combo board, with 6 degree of freedom consist of ITG3200 gyroscope and ADXL345 accelerometer, which is combination of 3 axis gyroscope and 3 axis accelerometer. The sensor is interfaced with the ATmega328 over the I2C communication protocol. The IMU combo board will be used like the MPU-6050 to estimate the angle of inclination using the sensor fusion approach, which uses the library provided Varesano laboratory.

3.2.4 Motor Controller

The Arduino motor shield is used to provide a communication link between the microcontroller and motors of the balancing robot in order to control the speed and direction of the movement of the robot depending on the inclination angle. In this project Arduino motor deriver shield and MD25 dual H-bridge was used to control the EMG30 DC motors. The motor driver requires 12volt 2.8Amp power the module and 5volts to its logic board. The dual motor driver provide two way of communication, which can be connected to microcontroller b I2C or serial communication , where the speed of the hardware baud rate can be varied by removing the headers on the board .figure 10 represent the schematic of the MD25.The reason for selecting MD25 is due to multi functionality and the feature provided by the board which is listed below:

- Reads and counts the pulse of the encoder to measure the position of the robot.
- The motor can be set to be controlled independently or used as a combined control
- Providing a steering feature, this is done by selecting the appropriate mode in the software implementation.
- Has the ability to control the speed, acceleration and the direction of the motors.
- Has the ability to monitor the voltage and current used by the battery.
- Providing two wire communication protocol, I2C and serial communication.

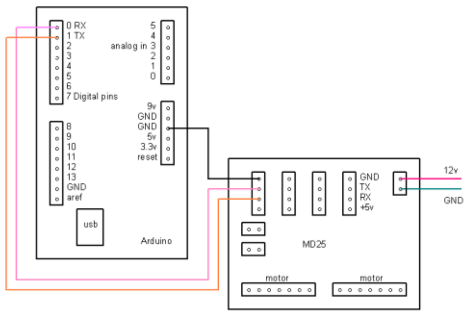


Figure 10 MD25 dual h-bridge schematics

3.2.5 Bluetooth module

HC-05 Bluetooth module provides wireless serial interface to communicate with the serial interface of the Arduino. In this project HC-05 is used to communicate the Arduino by a computer or a smart phone in order to monitor real time data on LCD and using the google voice from the smart phone device to send and transmit data wirelessly. The Bluetooth module was preferred over the Infrared receiver sensor (IR) because it does not get affected by low range and blockage as it provides range of approximately 100 meters communication where the IR sensor communicates in a range of approximately 1 meters .the schematic of the HC-05 can be seen in figure 11.

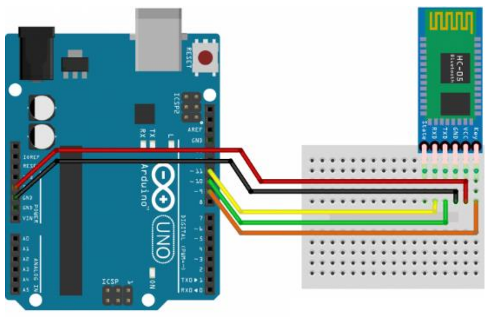


Figure 11 Bluetooth schematics

3.2.6 Liquid Crystal Display

In this project a Liquid Crystal Display (LCD) was used in order to monitor, sensor values and measured data without requiring to connect Arduino board to the computer. This LCD consist of 20 columns and 4 rows which provides sufficient space to display data on the screen such as, position, angle current , voltage and a controller output signal . To connect the LCD with the microcontroller 6 digital pin of the microcontroller was required .however another approach to reduce the number of pin was to use a SPP/I2C serial daughter board which allows a two wire connection by saving 4 pins of the microcontroller. A potentiometer was used in order to change the contrast backlight of the display.

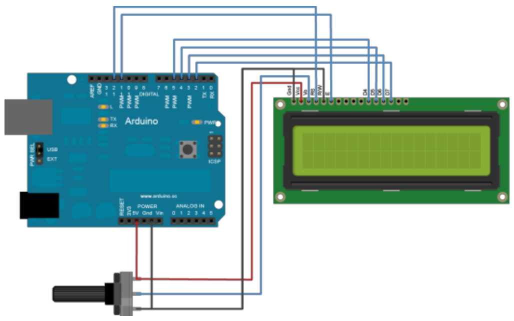


Figure 12 LCD connection

3.2.7 Encoder

An encoder is a sensor that generates digital signal as the motors shaft moves. As an electronic motion detector device it provides an ability for its users information such as speed, position, direction and velocity .the encoder are classified as rotatory and linear. Where liner encoder is used to calculate the position and speed, whereas rotary encoder respond to the motion of the motor shaft.

The encoder used in this project is magnetic rotary encoder, which consist of two part of sensor and rotor .the rotor is connected to the shaft of the motor and contains south and North Pole which are spaced equally as show in figure 3. The sensor count the number of magnetic field changes as the shaft rotates, which the preciseness is varied based on the system application, the encoder used in this project counts 360 per each rotation.

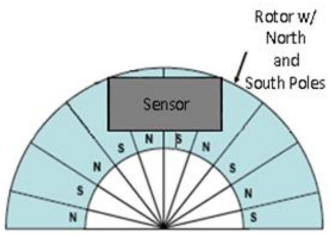


Figure 13 rotor with a north south poles [5]

3.2.8 Battery

Selecting an appropriate power supply for the stability robot was an important factor of this project as the motors and the electronic devices used require sufficient power to perform the stability of the robot.   In this project a 12Volts lithium-ion battery with a capacity of 3.8Ah was selected due to the low weight and the high capacity of this battery which provides sufficient time to keep the robot running. This battery has an USB port than can be connected to the Arduino directly   without any use of voltage regulator. Equation (3.1) and (3.2) is used to estimate the minimum time the motors can run with the selected battery, where maximum current used by the motors is 2.8Amp and batteries capacity is 3.8Ah.

$$Minimum\ time = \frac{capacity\ of\ the\ battery}{maximumm\ current} \tag{3.1}$$

$$Minumum\ time = \frac{3800mAh}{2800A} = 1.357h = 81\ min \tag{3.2}$$

The minimum time the battery can operate with a maximum current required by the motors is approximately 81 minutes. However to measure the precise time for the robot to operating is to divide the capacity by combination of the current drained by all devices and sensor.

3.2.9 Motors

In this project an Encoder Motor Gearbox 30:1 (EMG30) was used to maintain the stability of the robot by moment of the motors .EMG30 was selected because it provides sufficient torque and speed to main the stability of the robot, the robot has a speed of 200rpm without load and the speed reduces to 170 rpm when load is applied .each motor can handle a torque of 1.5kg/cm which is suitable for the designed stability robot with a weight of 1.1kg .The motor and the encoder are connected to the MD25 directly and requires 12volt power supply. To reduce the noise suppression capacitors has been added across the motor windings. The following list is the specification of the EMG30 :

Specification

➢ Voltage rate	12v
➢ Torque rate	1.5kg/cm
➢ Speed rate	170rpm
➢ Current rate	530mA
➢ Without load	200rpm
➢ Stall current	2.5A
➢ Power rate	4.2W

3.2.9 Annotation for balancing robot

Figure 14 , represents the annotation of the self-balancing robot , by providing the name and the position of the component .

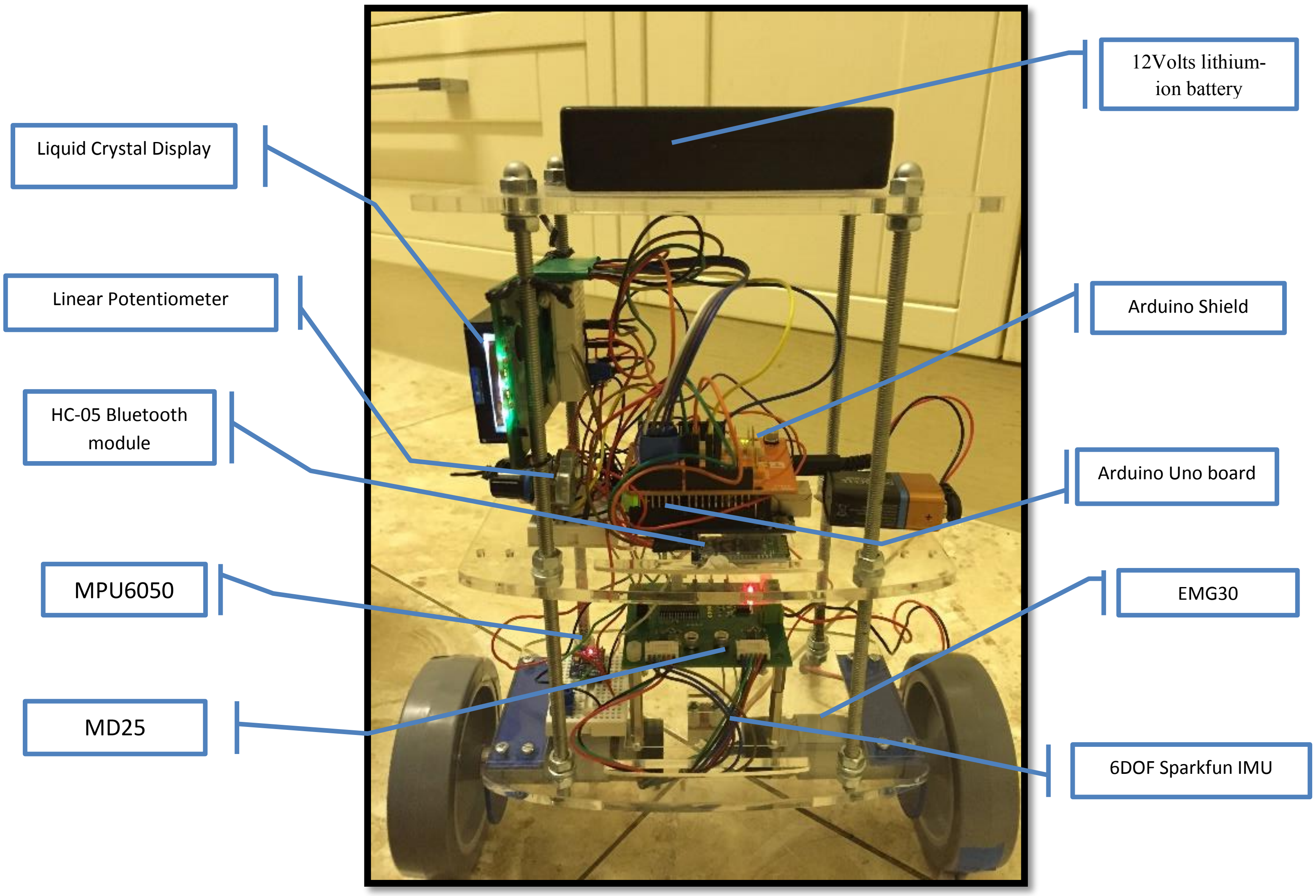


Figure 14 Anotation of self balancing robot

3.3 Communication protocol

This section will describe the communication protocol, to interface the sensors and actuators with Arduino controller.

3.3.1 Pulse Width Modulation

Pulse Width Modulation (PWM) is an approach to get analog reading from digital ports , the pulse produced by the digital port is square wave which produces 0 or 1 states , where this approach is used to switch the pins of the devices connected to the digital pin on-off by providing a voltage of 5 Volt and 0 Volts . In this project the DC power supply delivers a constant voltage to the motors, so in order to control the speed and direction of the motors we require to control the source voltage. The power source was drawn by a 12Volts battery and was used to switch states of the voltage delivered to the motor-drive connected to the PWM port, due to the electrical inertia the voltage will not suddenly change from 0Volts to 5Volts, the change in the voltage in PWM will depend on the duty-cycle, the change of voltage level by the PWM is calculated as follow:

$$V = T^{-1} \int_0^T f(t)\Delta t \qquad (3.3)$$

$f(t)$: represent the pulse-wave, were its constrain is described as follow:

$$f(t) = \begin{cases} V_{max}, & for\ 0 < t < T.D \\ 0, & for\ D.T < t < T \end{cases} \qquad (3.4)$$

V: represents the voltage mean, which is measured as follow:

$$V = D.V_{max} \qquad (3.5)$$

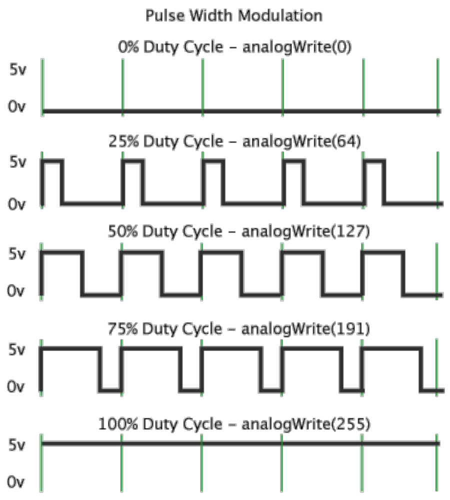


Figure 15 Pulse width Modulation [15]

In Figure 14, the green lines represent the time period which is equal to the inverse of frequency of the PWM where in the Arduino controller is approximately 500Hz (or 2ms) per unit pulse. It can be noticed the pulse width increases as the value if the analogWrite() increases from 0-255.

3.3.2 Serial Communication

Serial interfaces transfer data one bit per a single clock pulse, there are various serial communication, such as IIC, SPI. Serial communication transfer data asynchronous or synchronously .where in a synchronous serial communication data line is paired with a clock to the devices connected to the bus share a same clock, where the clock link provides faster data transfer than asynchronous data. where SPI and I2C both provides synchronous communication .where as an asynchronous those not provide clock pulse in order to assist the data transfer ,this approach is ideal to reduce the use of input output pins .

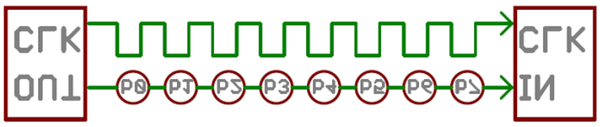


Figure 16 Serial Communication [15]

3.3.3 Inter Integrated Circuits

“Inter integrated circuit “or I^2C is also known as two wire connection is a protocol that allows multiple number of digital slave devices to interact with one or more master digital circuit devices figure 16, represent how and I^2C connection can be set , it consist of two bi-directional wires known as, Clock Line SCL ,and Data Line SDA, which they are usually linked with a pull up resistor .the I^2C in the Arduino controller uses an address of 7-bits with a speed of 100kHz. As shown in figure16, all the devices are connected to the SDA and SCL. The SCL is the clock line used to synchronise all the data than are transferring within the IIC bus. The devices connected to the IIC bus is either master or slave. The master is the device that issues the clock, and the slave is the device that responds to master devices, however both master and slave devices are able to transfer data through the I2C bus.

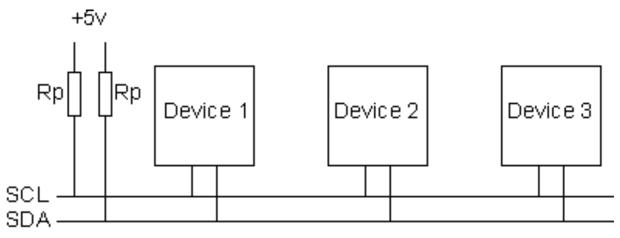


Figure 17 I2C Circuit

3.3.4 Final circuit communication port

Figure 17, represent the final electronic components indicating the communication protocol between the Atmega328 microcontroller and the devices. The MPU6050 uses the SDA/SCL pins of the Arduino, which interfaces with Arduino using I2C port. The EMG30 is connected to the MD25 board providing a serial interface with the Arduino which uses the RX/TX pins .LCD03 is interfaced with controller using the digital ports which uses six pins of the Arduino. The balancing robot is powered with a 12volts DC battery.

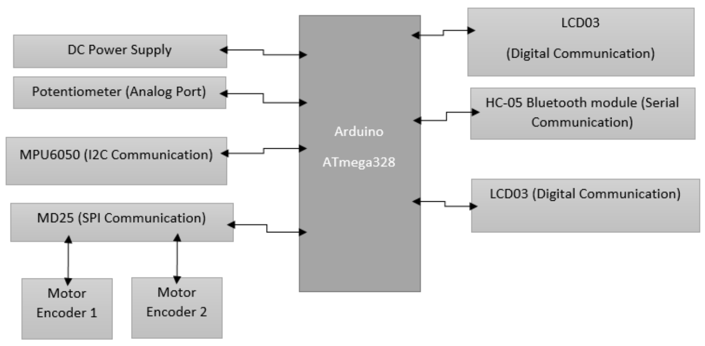

DC Power Supply
Potentiometer (Analog Port)
MPU6050 (I2C Communication)
MD25 (SPI Communication)
Motor Encoder 1
Motor Encoder 2
Arduino ATmega328
LCD03 (Digital Communication)
HC-05 Bluetooth module (Serial Communication)
LCD03 (Digital Communication)

Figure 18 block diagram of electronic components communication ports

4 Mathematical model

The mathematical model of the robot is divided into two main section of pendulum and wheels. Equation (4.4) to (4.6) represent the forces acting on the pendulum based on the clockwise rotation of pendulum on x and y direction. Equation (4.7) and (4.8) represents the forces acting on the wheels while moving in a clockwise direction,the forces acting on the wheel are can be seen in figure 16, the angle , angular velocity and angular acceleration of are denoted as $\emptyset, \dot{\emptyset}, \ddot{\emptyset}$ respectively and the angle , angular velocity and angular acceleration of the pendulum are denoted as $\theta, \dot{\theta}, \ddot{\theta}$ respectively .the parameters used to drive the mathematical model for inverted pendulum and wheels are represented in table 1.

Parameters	Name	Units
$\boldsymbol{g}$	Gravity acceleration	$9.8m/s^2$
$\boldsymbol{m_p}$	Mass of pendulum	$1.267kg$
$\boldsymbol{r_p}$	Distance between centre of gravity and middle of wheels	$0.09m$
$\boldsymbol{I_p}$	Inertia of pendulum	kgm^2
$\boldsymbol{F_p}$	Force acting on the y-axis of the pendulum attached to the wheel	N
$\boldsymbol{F_a}$	Force acting on the x-axis of the pendulum attached to the wheels	N
$\boldsymbol{T_m}$	Torque applied by the motors	Nm
$\boldsymbol{\theta}$	Pendulum Angle	rad
$\dot{\boldsymbol{\theta}}$	Pendulum angular velocity	rad/s
$\ddot{\boldsymbol{\theta}}$	Pendulum angular acceleration	rad/s^2
$\emptyset$	Wheel angle	rad
$\dot{\emptyset}$	Wheel angular velocity	rad/s
$\ddot{\emptyset}$	Wheel angular acceleration	rad/s^2
$\boldsymbol{r_w}$	Radius of wheel	$0.05m$
$\boldsymbol{F_f}$	Friction force applied to the wheel from the ground	N
$\boldsymbol{m_w}$	Mass of the wheel	$0.131kg$
$\boldsymbol{I_w}$	Inertia of the wheel	kgm^2
$\boldsymbol{F_w}$	Normal force applied to the wheel	N

Table 1 parameters of the pendulum

In order to identify the mathematical model we use the Newton's law of motion and Euler's second law. The Newton's second law state that the force applied to an object is proportional to the change of velocity of the object, in the direction that the force is applied. In the case of inverted pendulum the applied force to the rotational pendulum will result in applied torque from the motors which is proportion to the change of velocity of the pendulum.

Newton's second law:

$$m\ddot{x} = \sum F \tag{4.1}$$

Euler's second law:

$$I\ddot{\theta} = \sum T \tag{4.2}$$

4.1 Pendulum model

The torques and forces acting on the pendulum body can be seen in figure 15.

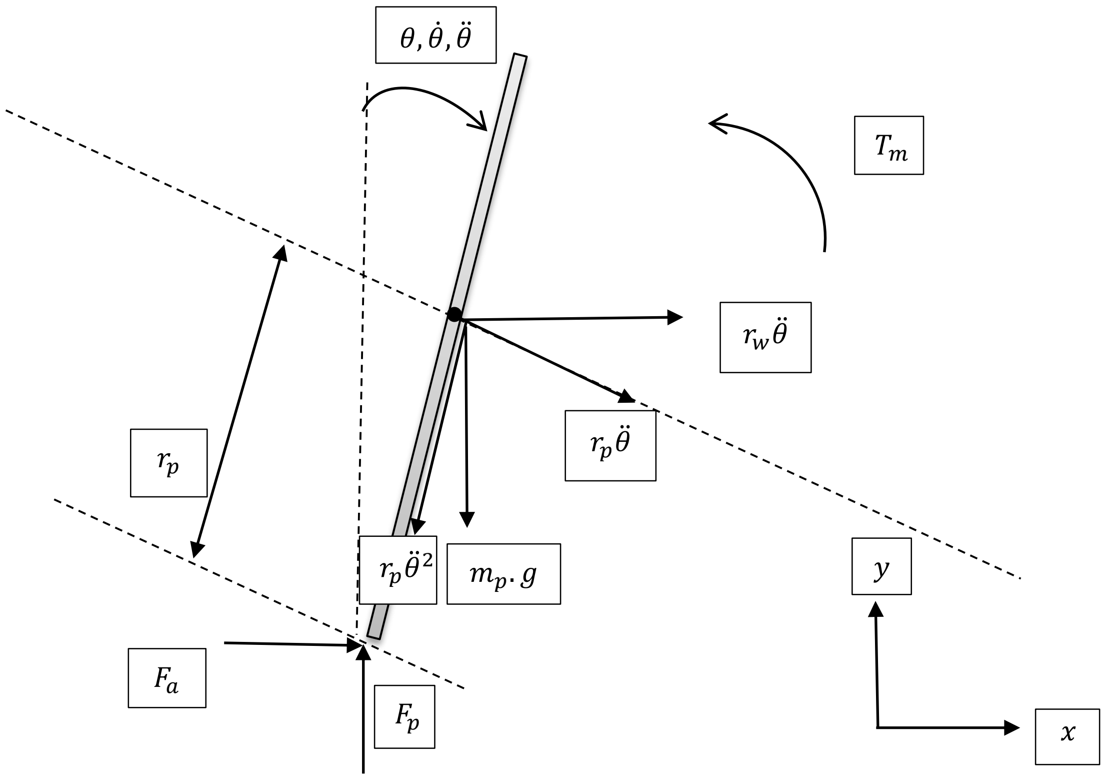


Figure 19 Forces acting on pendulum body

The following equation are derived from the pendulum body .where equation (4.3) and (4.4) are forces acting on the dashed line ,which is perpendicular to the pendulum and (4.5) is the momentum acting on the centre of gravity of the pendulum .

$$m_p r_w \ddot{\emptyset} + m_p r_p \ddot{\theta} \cos\theta - m_p r_p \dot{\theta}^2 \sin\theta = F_a \tag{4.3}$$

$$m_p r_w \ddot{\emptyset} \cos\theta + m_p r_p \ddot{\theta} = F_a \cos\theta - F_p \sin\theta + m_p g \sin\theta \tag{4.4}$$

$$I_p \ddot{\theta} = -T_m - F_a r_p \cos\theta + F_p r_p \sin\theta \tag{4.5}$$

4.2 Wheel model

The torque and forces acting on the wheels can be seen in figure 5.

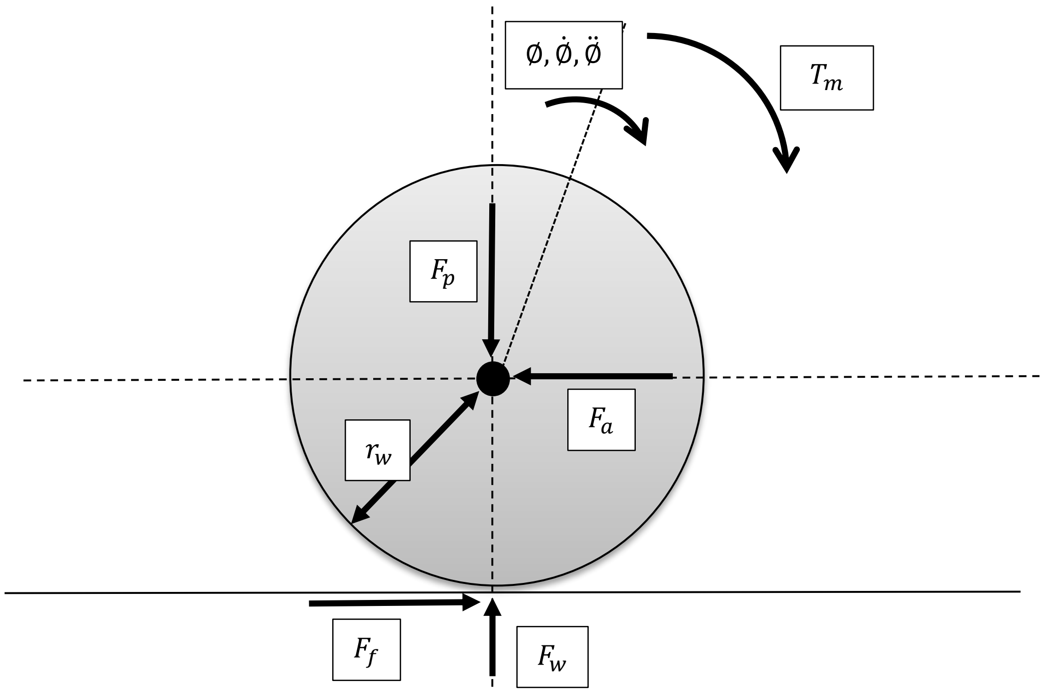


Figure 20 torque and forces applied to wheels

The following equations are derived from wheels body, where equation (4.6) is the momentum applied around the wheel in a clockwise direction and equation (4.7) is the forces applied to the x-axis of the wheels.

$$m_w r_w \ddot{\emptyset} = F_f - F_a \tag{4.6}$$

$$\ddot{\theta} I_w = T_m - r_w F_f \tag{4.7}$$

By substituting equation to (4.5) to (4.4) equation (4.8) is evaluated which by arranging equation (4.8) the angular acceleration of the wheel, equation (4.9) is derived.

$$m_p r_w r_p \ddot{\theta} \cos\theta + m_p r_p^2 \ddot{\theta} = -T_m - I_p \ddot{\theta} + m_p g r_p \sin\theta \tag{4.8}$$

$$\ddot{\theta} = \frac{m_p g r_p \sin\theta - m_p r_w r_p \ddot{\emptyset} \cos\theta - T_m}{m_p r_p^2 + I_p} \tag{4.9}$$

To evaluate the angular acceleration of the wheels, we substitute equation (4.4) and (4.6) into equation (4.7), which results to equation (4.10) and the angular acceleration equation (4.11) is derived by rearranging equation (4.10).

$$I_w\ddot{\emptyset} = T_m - (m_p r_w\ddot{\emptyset} + m_p r_p\ddot{\theta}\cos\theta - m_p r_p\dot{\theta}^2\sin\theta + m_w r_w\ddot{\emptyset})r_w \tag{4.10}$$

$$\ddot{\emptyset} = \frac{m_p r_w r_p\dot{\theta}^2\sin\theta - m_p r_w r_p\ddot{\theta}\cos\theta + T_m}{m_p r_w^2 + m_w r_w^2 + I_w} \tag{4.11}$$

To simply the equations, we let α, β, γ to be the following equations:

$$\alpha = m_p r_p^2 + I_p \tag{4.12}$$

$$\beta = m_p r_w^2 + m_w r_w^2 + I_w \tag{4.13}$$

$$\gamma = m_p r_w r_p \tag{4.14}$$

Equation (4.15) and (4.16) is derived by simplifying (4.9) and (4.11) in the form ofα, β, γ.

$$\boldsymbol{\ddot{\theta}} = \frac{\boldsymbol{m_p g r_p \sin\theta - \gamma\emptyset\cos\ddot{\theta} - T_m}}{\boldsymbol{\alpha}} \tag{4.15}$$

$$\ddot{\emptyset} = \frac{\boldsymbol{\gamma\dot{\theta}^2\sin\theta - \gamma\theta\cos\ddot{\theta} + T_m}}{\boldsymbol{\beta}} \tag{4.16}$$

By combing equation (4.15) and (4.16). Angular acceleration of the wheel and pendulum are represent as (4.17) and (4.18).

$$\ddot{\theta} = \frac{m_p g r_p}{\alpha}\sin\theta - \frac{1}{\alpha}T_m - \frac{\gamma^2}{\alpha\beta}\dot{\theta}^2\sin\theta\cos\theta + \frac{\gamma^2}{\alpha\beta}\ddot{\theta}\cos^2\theta - \frac{\gamma}{\alpha\beta}\cos\theta\, T_m \tag{4.17}$$

$$\ddot{\emptyset} = \frac{\gamma}{\beta}\dot{\theta}^2\sin\theta + \frac{1}{\beta}T_m - \frac{\gamma m_p g r_p}{\alpha\beta}\sin\theta\cos\theta + \frac{\gamma^2}{\alpha\beta}\ddot{\emptyset}\cos^2\theta + \frac{\gamma}{\alpha\beta}\cos\theta\, T_m \tag{4.18}$$

4.3 Nonlinear model

The nonlinear model for pendulum and the wheels can be expressed as equation (4.19) and (4.20) providing nonlinear equation for the angular velocity of the wheel and pendulum, which is derived by combining equation (17) and (18).

$$\ddot{\theta} = \frac{m_p g r_p \beta \sin\theta - \gamma^2 \dot{\theta}^2 \sin\theta\cos\theta - (\beta + \gamma\cos\theta)T_m}{\alpha\beta - \gamma^2\cos^2\theta} \tag{4.19}$$

$$\ddot{\emptyset} = \frac{\alpha\gamma\dot{\theta}^2 \sin\theta - \gamma g m_p r_p \sin\theta\cos\theta + (\alpha + \gamma\cos\theta)T_m}{\alpha\beta - \gamma^2\cos^2\theta} \tag{4.20}$$

4.4 Linearized model

We define the states of the system as follow:

$$x_1 = \theta, x_2 = \dot{\theta}, x_3 = \emptyset, x_4 = \dot{\emptyset} \tag{4.21}$$

In order to design linearized quadratic controller for the robot, equations (4.19) and (4.20) requires to be linearized, where $\theta, \dot{\theta}, \emptyset, \dot{\emptyset}$ is considered to be zero .Equation (4.22) represent the linearized model for state space controller form:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{m_p g r_p}{\alpha\beta - \gamma^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{\gamma m_p g r_p}{\alpha\beta - \gamma^2} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{\beta + \gamma}{\alpha\beta - \gamma^2} \\ 0 \\ \frac{\alpha + \beta}{\alpha\beta - \gamma^2} \end{bmatrix} T_m \tag{4.22}$$

4.5 Linear control strategy

We will analyse the linear model to design a state feedback controller and provide a comparison between PID controller and the linear control strategy. This will be done by observing the behaviour of the balancing robot and providing an evaluation based on the controller response. In the linear control strategy the constant parameter of the full state feedback controller is designed based on the mathematical model whereas the constant parameters of the PID are estimated based on the Ziegler-Nicholas model where controller gains are tuned by observation and testing.

In order to design a closed loop system, we adapt the state feedback control strategy, with an applied torque of $T_m = G \times x$. Consequently to achieve this we require to evaluate the gains of the controllers by using the Matlab software. Which is evaluated by the measured parameters of the robot and the linearized equation derived in section 4.4. The constant gains of the state feedback controller is as follow:

G = [4.9007e+00 4.2323e-01 6.5664e-02 1.0260e-01].

5 Control Design

5.1 Complementary Filter design

In this project in order to combine the singals of the gyroscope and accelerometer a complementary filter was implemented to produce accurate angle estimation. This approach is computationally efficient and as results produces an accurate angle estimation by eliminating the accelerometer noise and long term drift of the gyroscope. The technique used to implement the complementary filter, is to use the gyroscope signal for short term angle estimation by applying numerical integration and the accelerometer signal is used for long term angle estimation by taking average of its measured reading.

Figure 21, represents a block diagram for a complementary filter . The accelerometer raw signals are used measure the angle and the gyroscope raw signals are used to measure the angular velocity. The accelerometer measured angle is low passed filtered to eliminate the short term effect of the accelerometer noise by maintaining the long term measured angle estimating by averaging .

The gyroscope signal is initially converted into angular velocity and integrated in order to get the angle estimation based on the equation drived in section 2.5, the estimated angle will be sent to a high pass filter to eliminate the long term drift produced by the gyroscope. Finally the accelerometer, low pass filtered signal is combined with the gyroscope in order to produce accurate angle estimation, the graphical repetition of the angle estimation is represented in section 7.2.

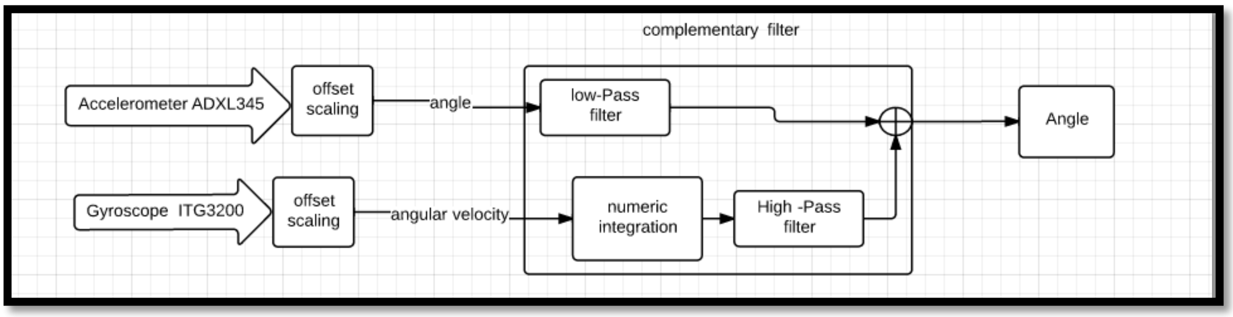


Figure 21 Complementary Filter block diagram

5.3 PID controller design

The fundamental concept of stability robot is based on the non-linear behaviour of the inverted pendulum. In order to make a stability robot we require to use a control algorithm. The control algorithm used in this project is call proportional Integral Derivative (PID). In order to maintain stability we require to keep the robot's wheel move to the position where the centre of gravity of the robot tends to move.To estimate the position centre of gravity an IMU sensor is used to measure the angle of the robot. the filtered IMU signals are then transmited to the PID algorithm to reduce the error produced by the robot.

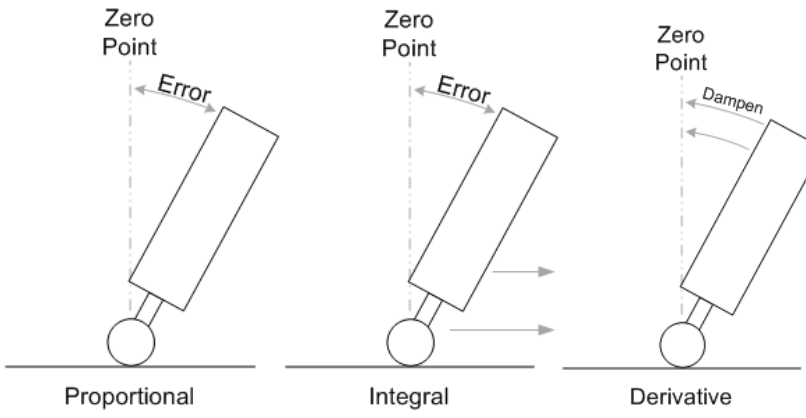


Figure 22 System reaction of PID algorithm

- **Proportional Coefficient:**The proportion term represent the current error appearing in the system, the error, e_x of the system is value of the angle from the set point or zero position which is shown in figure 22, in order to calculate the proportional term, the calculated error requires to be multiplied by the proportion gain, Kp .

$$e_x = \text{Setpoint} - \text{angle} \quad (5.1)$$

$$P.Term = e_x \times Kp \quad (5.2)$$

- **Integral Coefficient:** The integral term generates an output which is equivalent to the accumulation of the past error appearing in the system over constant period of time .for instance if the error is e_1 at time t_1 and e_2 at time t_2 , the integral error would be calculated as (5.3) and (5.4) represent the integral term calculation.

$$I = (\frac{e_1}{t_2} + \frac{e_2}{t_2}) \times k_i \quad (5.3)$$

$$I.Term = k_i \times \sum_{i=0}^{n} e_x \; \Delta T \quad (5.4)$$

- **Derivative Coefficient**: The derivative term generates an output which reduces rate change produced by the feedback controller, by adjusting the constant value of the integral gain the overshoot and oscillation appearing in the system will be reduced and eliminated.

$$D.Term = \frac{e_{x+1} - e_X}{\Delta T} \times K_d \tag{5.5}$$

- **Output generated by the PID controller :**the output generated by the PID controller is equivalent to combination of the Proportional , integral and derivative values , where e_X represent the current error and $K_{(P , I , D)}$ are classified as the constant gain of the controller the following equation is the output produced by the PID controller .

$$PID.OUT = P.Term + I.Terrm + D.Term \tag{5.6}$$

The block diagram shown in Figure 23, represent the design for a linear PID feedback Controller, where the input of the controller is the pitch angle, measured by the Complementary and the set-point .the combination of angle and set-point is processed by the PID controller and the output signal is amplified and send to the motor controller in order for the correction to take place.

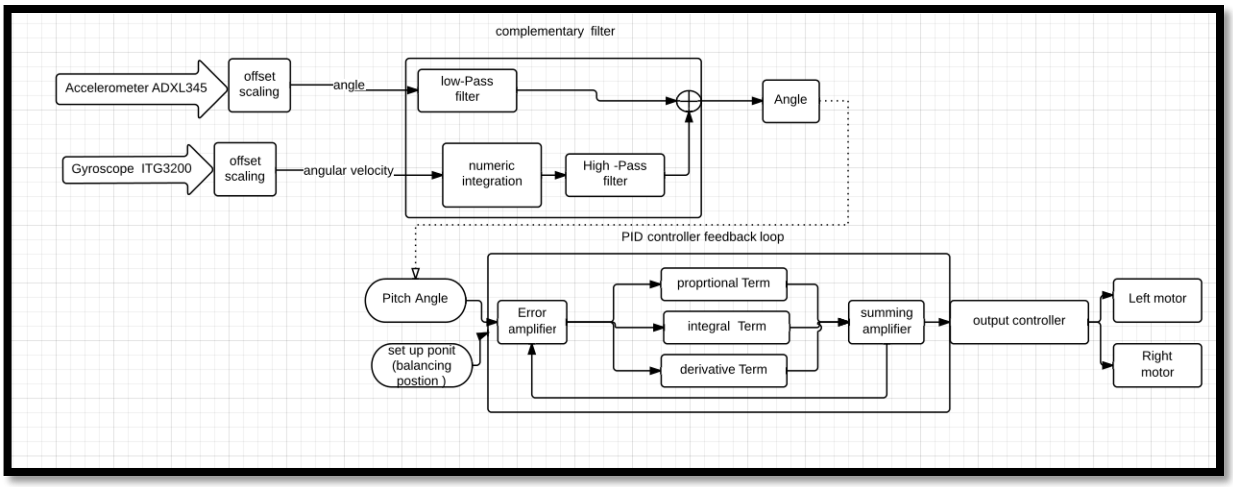


Figure 23 stability system PID block diagram

5.4 Position control design

Initially the set point of the PID controller was set to zero to balance the robot in a vertical position, therefore by applying a disturbance to the robot, the position will remain undetected. In order improve the PID controller an position control algorithm was implemented to prevent the robot from moving in an continuous direction when disturbance was applied.

The encoder sensor was used to measure the postion of the robot based on pulse recived as the motors shaft rotates . the set-point of the robot is calculated based on equation (5.7), and where set-point value is derived based on the zone and position of the robot. Where when the position increases from the inner zone the calculated rest angle increase to prevent the robot from moving further away. Figure 24, is the combination of, PID controller and the position control approach.

The encoder produces 360 pulses per rotation, the following constrains determines the set-point angle of the robot based on its position.

ZoneA: $\frac{0}{1000} < Angle < \frac{2000}{1000}$

ZoneB: $\frac{2000}{500} < Angle < \frac{4000}{500}$

ZoneC: $\frac{4000}{250} < Angle < \frac{\infty}{250}$

Equation (5.7) is used to calculate the set point of the robot, when it’s moving forward and backward.

$$SetPoint = \frac{wheelPosition - TargetAngle}{PositionScale} \tag{5.7}$$

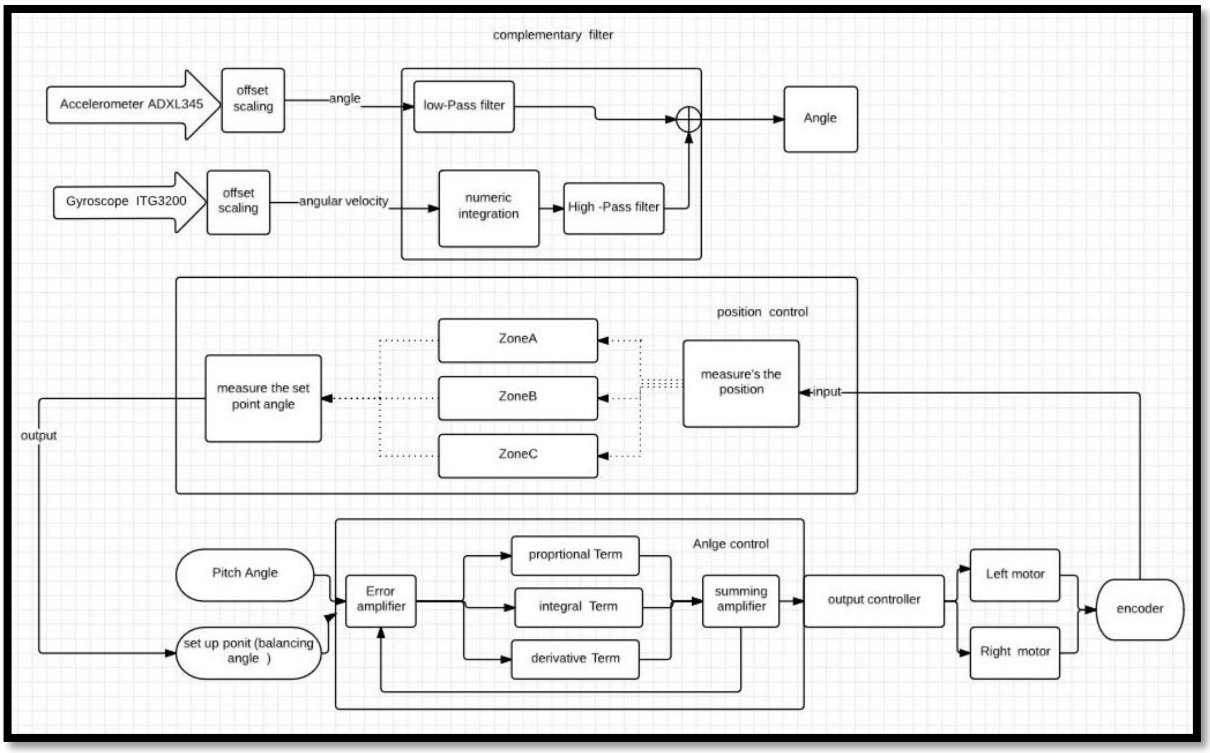


Figure 24 position control block diagram

5.5 Full state feedback controller design

Figure 25, represent the block diagram of a state feedback controller, where input of the controller is based on angle and angular velocity of the pendulum and position and angular velocity of the wheels, to measure the values and IMU and encoder will be used and the constant gains of the state feedback controller is evaluated using the Matlab ,software based on the mathematical equation described in section 4.5. Equation (5.7) represent the output value generated by the state feedback controller, where the value is feed into the motor controller in order to perform the stability.

$$output = k_1\theta + k_2\emptyset + k_3\dot{\theta} + k_4\dot{\emptyset} \qquad (5.8)$$

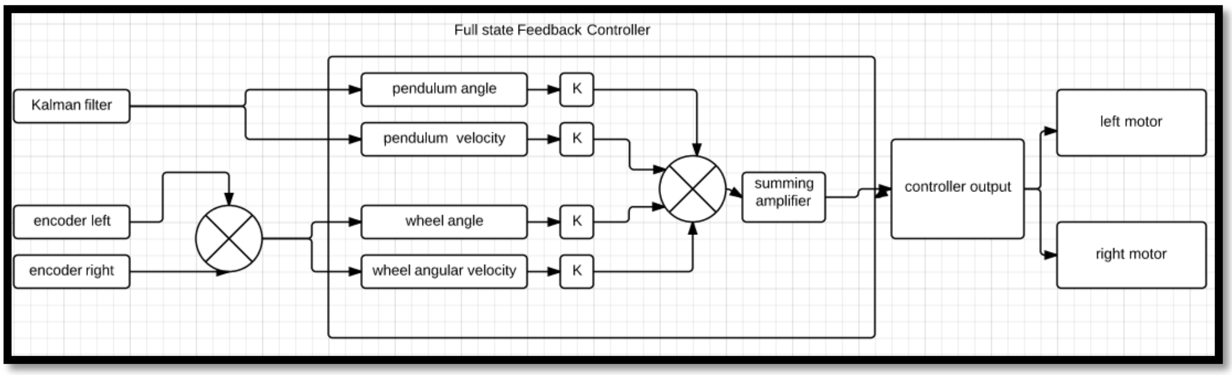


Figure 25 Full State Feedback controller

5.6 Trajectory control design

In order to make to robot move in the desired direction a constant offset is sent to output signal of the controller, this is done so robot maintain its stability while moving around. The speed and direction of the robot is direction is control by the offset value sent to the control. This project use IR-remote control, voice activation and android application to control the value of the offset to control the movement of the robot.it was possible to navigate the robot back and forth, however the robot was unable to maintain stability when stopped this is due to the overshoot and oscillation during the balancing process . Figure 26, represent the remote control operation, where K is constant offset sent to the controller to navigate the robot.

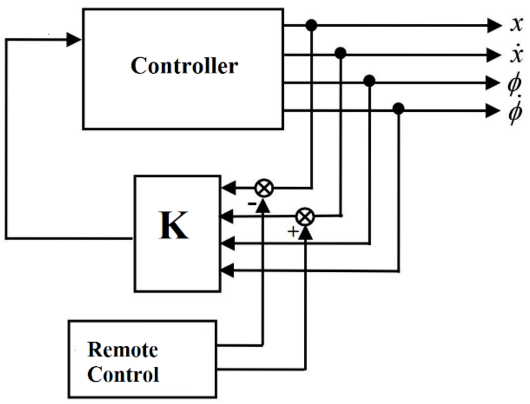


Figure 26 Remote control operation

6 Software Implementation

6.1 Introduction to software implementation Design

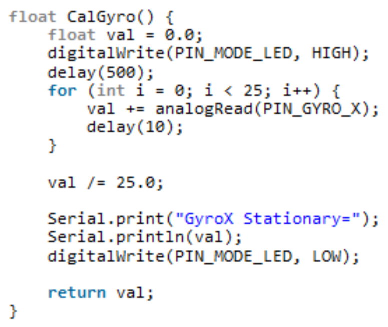
To achieve stability we require to make the actuators to communicate with the sensor in order to control the direction and speed of the wheels, to achieve this there are number of steps that are required to be taken. Initially the incline position of the robot was measured by applying the sensor fusion technique using Kalman filter to provide an accurate angle estimation .Then to improve the effectiveness and reliability of the control system an position control will be added.

The programming language platform for this project is C++ .This ATmega328 controller has various communication port such as digital and analog port which provide I2C and SPI communication. The Arduino IDE provides a serial monitor in order the monitor any necessary measurements and readings. This section will describe the main implemation required for the balancing robot .To improve the software quality , functions were used to avoid repetition of the code and to save memory space . For the I2C and SPI communication , the commands and addresses used in the software implementation can be seen in appendix [C] and [D].

6.2 Calibration of the gyroscope signal

ITG3200 and MEMS are 16bits analog to digital converter , which can be interface with the I2C port of the ATmega328 . The initial step carried out to measure the inclination angle was to calibrate the gyroscope sensor in order to estimate the zero position of the robot, to do this a gyroscope calibration function was created and called as the angle measurement initiates to process .

In the calibration process the robot should be kept in a stationary position to avoid any error in the measurement . The gyroscope raw data under goes 25 iteration, which takes an average of 25 readings to reduce the error in the main angle measurement. There will be an LED and serial print indication when the gyroscope is calibrating .



```
float CalGyro() {
    float val = 0.0;
    digitalWrite(PIN_MODE_LED, HIGH);
    delay(500);
    for (int i = 0; i < 25; i++) {
        val += analogRead(PIN_GYRO_X);
        delay(10);
    }

    val /= 25.0;

    Serial.print("GyroX Stationary=");
    Serial.println(val);
    digitalWrite(PIN_MODE_LED, LOW);

    return val;
}
```

6.3 Estimating the angle

In this project an ADXL345 and MEMS accelerometer will be used, In order to measure the angle of inclination, to do this we require to use the following equation which is derived in section 2.3 , where g is the gravitational force and $A_{x.y.z}$ is the pitch ,raw ,yaw axis of the accelerometer .

$$\partial = arcSin(\partial) = \frac{A_{x.y.z}}{g} \tag{6.1}$$

The following implementation is the angle estimation carried using the accelerometer raw data:

```
float GetAccAngle() {
    getAccXYZ(&VxAcc, &VzAcc, &VyAcc);

    float r = sqrt((float) VzAcc * (float) VzAcc + (float) VyAcc * (float) VyAcc);
    accAngle = (float) VzAcc / r; //approximates sine

    return accAngle;
}
```

To combine the value of the accelerometer and the gyroscope, we will use (6.2) and (6.3) Kalman filter, which was derived in section 2.5, to estimate the accurate angle of inclination, the following equation is used to convert IMU signals into radians

$$\partial(t) = \partial(t1) + G(t).(t2 - t1) = \partial(t1) + G(t).\Delta t \tag{6.2}$$

$$\begin{cases} \partial est|t2 = X.(\partial est|t1 + G\Delta t) + Y.\partial_{X.Y.Z} \\ X + Y = 1 \end{cases} \tag{6.3}$$

The following implementation represent the Kalman Filter equation, which is written in a function form and called in the void loop () of the programme.

```
void Filter()
{
// Raw datas from MPU6050
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  Angle_Raw = (atan2(ay, az) * 180 / pi + Angle_offset);
  omega =  gx / Gyr_Gain + Gyr_offset;
  // Filters datas to get the real gesture
  unsigned long now = millis();
  float dt = (now - preTime) / 1000.00;
  preTime = now;
  float K = 0.9;
  float A = K / (K + dt);
  pitch = A * (pitch + omega * dt) + (1 - A) * Angle_Raw;


 }
```

The omega represent the angular velocity measured by the gyroscope in a period of time and since the ATmega328 is using 10bits for the ADC and the gyroscope receives a voltage of 3300mV with a sensitivity of 2mV/degree, the radians measurement of the gyroscope is approached as follow:

$$\theta = \frac{2 * 3300 * \ Vgyro}{(2^{10}) * 512 * 180} * \ \Delta t \tag{6.4}$$

The following implementation is carried out to calculate the average filtered angle estimation by using a buffer, where the variable angleBuffer is a circular buffer and the values of the buffer are constantly updated in the void loop.

```
float GetAvgAngle() {
    int count = 0;
    float a = 0;

    for (int i = 0; i < ANGLE_BUFFER_LENGTH; i++) {
        if (angleBuffer[i] != 0) {
            count++;
            a += angleBuffer[i];
        }
    }

    if (count > 0)
        return a / float(count);
    else
        return 0.0;
}
```

6.4 PID feedback loop implementation

As mentioned previously the PID algorithm will be implemented in order to control the movement of the robot. To implement the PID algorithm an myPID() function has been created and called in void loop in order to continuously correct the position of the robot , the error in the PID algorithm is the position of the robot from its equilibrium position , to evalue the error the pitch angle of Kalman filter is selected . In order to produce the appropriate output for the controller ,we require to tune the parameters of the PID where the proportional gain and the derivative are adjusted by a potentiometer connected to analog pin of the atmega328.this is done as the potentiometer maps the voltage produced by the analog pin which has a resolution of 0-1024.

```
void myPID()
{
  kp =analogRead(A0)/100;    //5.5//7
  ki =0 ;    // 0.025//0.001
  kd =analogRead(A1);  //375//350
int PIDOutput;
  /*this is for updating the PID */
  // Calculating the output values using the PID values.
  unsigned long now = millis();
  timeChange = (now - lastTime);
  lastTime = now;
  error = (-pitch);  // Proportion
  errSum += error * timeChange;  // Integration
  dErr = (error - lastErr) / timeChange;  // Differentiation
  float PTerm =kp * error;
  float iTerm =ki * errSum;
  float DTerm =kd * dErr;

    if (iTerm > 50) { iTerm = 50; }
  if (iTerm < -50) { iTerm = -50; }

  Output = kp * error + iTerm + kd * dErr;
  lastErr = error;
```

By calling the function myPID() in the voild loop , the function under goes infinit interation to calcultate the output signal of the controller to make the correction .The output signal of the controller is transimted to the MD25 board to control the movment of the robot. The implementation required for the classical PID controller is divided into three different levels , which is described as follow :

Once the sensor calibration takes placed, the inclination angle of the robot which is measured by the complementary filter will be set to zero either manually or by a potentiometer, then the pitch angle of the filter will be used as the error value, the range of the angle is constrains in the range of ± 90 degrees. The PID loop uses the pitch angle in order to measure the angle.

To implement the proportional term of the PID controller, we multiply the current error driven by the angle estimation by the proportional gain. As shown in the equation below:

$$P = k_p \times e(\tau) \tag{6.5}$$

The following code represent the proportional term, where kp represent the proportional gain:

```
error = (-pitch);  // Proportion
float PTerm =kp * error;
```

The integral term of the PID controller is implemented by multiplying the integral gain by the accumulation of error over a period of time   .The value of the integral term is constraint by ±50 to prevent integral wind up. Equation (6.6), represent calculation for integral .

$$I = k_i \int_0^t e(\tau)d\tau \tag{6.6}$$

The following code represent the implementation carried out to calculate integration output were k_i represents integral gain and errSum is the accumulation of error over period of time.

```
unsigned long now = millis();
timeChange = (now - lastTime);
lastTime = now;
errSum += error * timeChange;  // Integration
float iTerm =ki * errSum;
  if (iTerm > 50) { iTerm = 50; }
if (iTerm < -50) { iTerm = -50; }
```

In order to implement the differentiation term of the PID we require to multiply derivative constant gain by the calculated future error over period of time. As shown in equation (6.7).

$$D = k_d \frac{de(t)}{dt} \tag{6.7}$$

The following code represent the implementation carried out to calculate the derivative term, were k_d represents the derivative constant gain and dErr is the future error over period of time.

```
dErr = (error - lastErr) / timeChange;  // Differentiation
float DTerm =kd * dErr;
```

Finally in order to generate the output signal of the PID controller, we require combine all measured parameter of the PID, as shown in (6.7).

$$PID = k_p \times e(\tau) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de(t)}{dt} \tag{6.8}$$

The following code represent the implementation of the PID output, which will be sent to the motor driver to control the robots movement.

```
Output = kp * error + ki * errSum + kd * dErr;
```

6.5 Full state feedback controller implementation

In order to implement the state feedback controller, the encoder sensor was used to measure the position and angular velocity of the wheels and the IMU sensor was used to measure the angle and angular velocity of the pendulum .the controller is design based on the linear model described in section 4.5 and the parameters were evaluated using the Matlab function. Equation (6.9) represent the output of the state feedback controller.

$$output = x_1\theta + x_2\emptyset + x_3\dot{\theta} + x_4\dot{\emptyset} \tag{6.9}$$

```
void FullStateFeedback{
  //pendulum
  sixDOF.getYawPitchRoll(ypr);
  Pitch = ypr[1];
  error = BalancePoint - Pitch;
  pTerm = k1p * error;
  integrated_error += error;
  if (integrated_error > 1) { integrated_error = 1; }
  if (integrated_error < -1) { integrated_error = -1; }
  iTerm = Ki * integrated_error;
  dTerm = k2d * (error - last_error);
  last_error = error;
 //Wheels
 error2 = BalancePostion - encValue;
 EpTerm =k2p*error2;
 EdTerm=k2d*(error2-Last_error2);
 Last_error2=error2;
```

6.6 Loop Time time control

To improve the reliabity and effectiveness of the system an loop time control was implemented. To do this we required to measure the time for each interation in the void loop(). To measure the loop time ,we either can use oscilloscope to measure the output running time signal as shown in figure 27, or make use software implementation to calculate the loop time and monitor the timing on the serial monitor .In the following code , SerialOutTiming() function is exexuted in the void loop () , to measure the run time of the loop .

```
  void serialOut_timing() {
static int skip=0;
  if(skip++==5) {                          // display every 500 ms (at 100 Hz)
    skip = 0;
    Serial.print(lastLoopUsefulTime);     Serial.print(",");
    Serial.print(lastLoopTime);           Serial.print("\n");
  }
```

To control the running time of the loop , the following program is implemented :

```
  lastLoopUsefulTime = millis()-loopStartTime;
   if(lastLoopUsefulTime<STD_LOOP_TIME)         delay(STD_LOOP_TIME-lastLoopUsefulTime);
 lastLoopTime = millis() - loopStartTime;
   loopStartTime = millis();}
```

Based on the serial monitor implementation the loop runs 99.94 interation per second , or 99.94Hz.

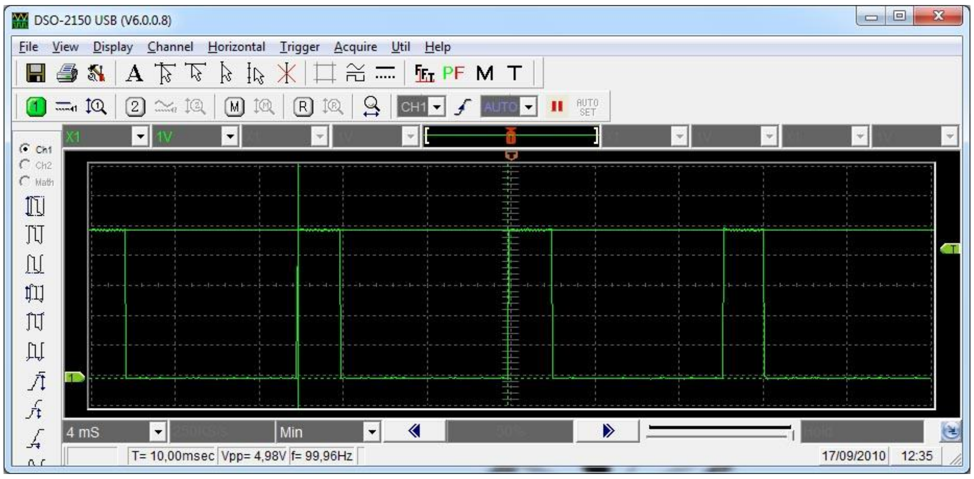


Figure 27 Osciloscope Time measurment (19)

6.7 Voice activation implementation

In order to control the microcontroller by using voice commands, an HC-05 Bluetooth module was used to create connection between the Arduino and the smartphone to do this an android application was designed and implemented by MIT App inventor .this application uses the google voice recognition of the android device and convert voice to string .the string generated by the voice command will be transmitted to the HC-05 and data will be decoded and processed as required , the implementation carried out for front end and back end design of the android application can be seen in appendix [A] and [B]. The following code is implemented to decode the data received by the Bluetooth module which communication with the serial port of the Arduino.

```
while (BT.available()){  //Check if ther
delay(10); //Delay added to make thing s
char c = BT.read(); //Conduct a serial r
if (c == '#') {break;} //Exit the loop w
voice += c; //Shorthand for voice = voic
}
if (voice.length() > 0) {
  Serial.println(voice);
```

7 Experiment, Results & Data Acquisition

7.1 Accelerometer and Gyroscope

In this project Accelerometer ADXL345 and Gyroscope ITG3200 where used as a tilt sensor to measure the angle of inclination, to achieve this the raw values of the sensors were converted into degrees and a practical test was carried out to glow the LEDs depending on angle and axis of rotation .It can be seen in figure 28 the LED glows depending on the axis of rotation.

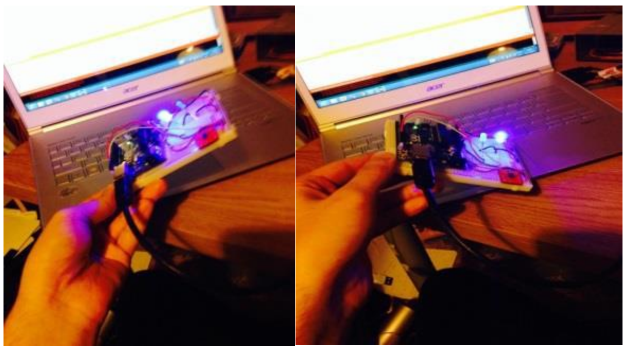


Figure 28 Accelerometer LED test

Angle (Degree)	0	10	20	30	40	50	60	70	80	90
Gyroscope (Raw data)	234	245	267	289	340	367	390	430	460	487
Acceleration (Raw data)	511	528	549	586	611	634	645	676	678	694

Table 2 Raw data convertion to degree

Is was noticed by Arduino serial monitor the initial raw reading of the both the accelerometer and gyroscope is not useful on its own.therefore the raw singal of the accelerometer and gyroscope were conveted into a meaningful value based on the calculation described in section 2.3, by using inverse tangent of x-axis and y-axis to evalucted the angles in radians and then converting the radians to degree using the atan(Xang , -Zang) command .

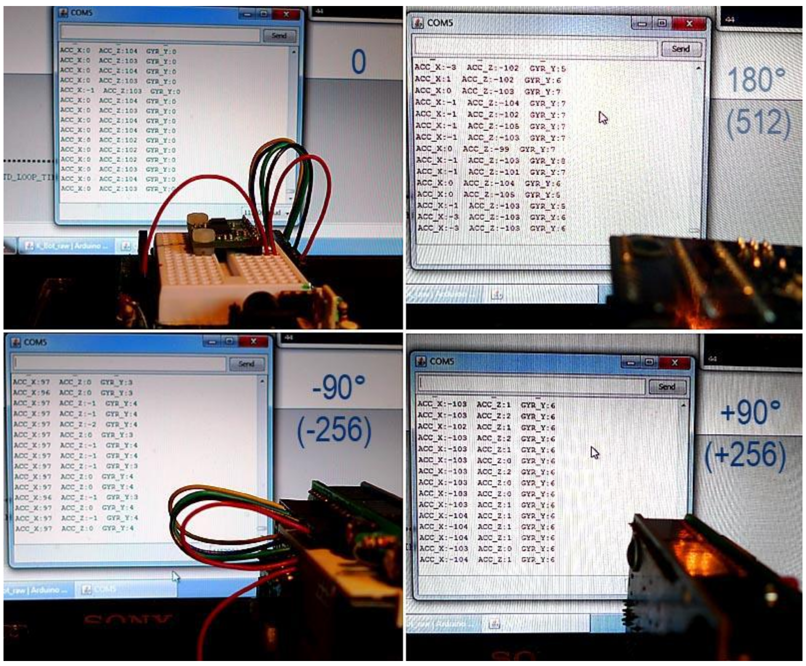
```
x = RAD_TO_DEG * (atan2(-yAng, -zAng) + PI);
y = RAD_TO_DEG * (atan2(-xAng, -zAng) + PI);
z = RAD_TO_DEG * (atan2(-yAng, -xAng) + PI);
```

Once the sensor reading was converted into degrees the values where then mapped in the range of -90 to 90 , where the sensor is zero in its vertical position.

```
int xAng = map(xRead, minVal, maxVal, -90, 90);
int yAng = map(yRead, minVal, maxVal, -90, 90);
int zAng = map(zRead, minVal, maxVal, -90, 90);
```

Test Description	Pass/fail	Comment
Converting the Accelerometer and Gyroscope raw data in degrees.	Pass	Both sensors where used as a tilt sensor ,the raw values where converted in to degree , where the sensor stationary position of the sensor indicates the sensors is in its stationary position and the values changes depending on what axis the sensor is rotating .
Testing the sensors using an LED on one axis of rotation	pass	There was two LED's placed on the X-axis of the tilt Sensor .it was noticed by rotating the sensor the LED glows depending on the angle.
Testing the sensors on multiple axis of rotation	Pass	In this testing three LED's where placed on each axis X, Y , Z and it was possible to glow the LED's based on which axis the sensor is tilting .

Table 3 Accelerometer LED



```
ACC_X:0   ACC_Z:104   GYR_Y:0
ACC_X:0   ACC_Z:103   GYR_Y:0
ACC_X:0   ACC_Z:104   GYR_Y:0
ACC_X:0   ACC_Z:103   GYR_Y:0
ACC_X:-1   ACC_Z:103   GYR_Y:0
ACC_X:0   ACC_Z:104   GYR_Y:0
ACC_X:0   ACC_Z:103   GYR_Y:0
ACC_X:0   ACC_Z:104   GYR_Y:0
ACC_X:0   ACC_Z:104   GYR_Y:0
ACC_X:0   ACC_Z:102   GYR_Y:0
ACC_X:0   ACC_Z:102   GYR_Y:0
ACC_X:0   ACC_Z:102   GYR_Y:0
ACC_X:0   ACC_Z:103   GYR_Y:0
ACC_X:0   ACC_Z:104   GYR_Y:0
ACC_X:0   ACC_Z:103   GYR_Y:0
```

```
ACC_X:-3   ACC_Z:-102   GYR_Y:5
ACC_X:1   ACC_Z:-102   GYR_Y:6
ACC_X:0   ACC_Z:-103   GYR_Y:7
ACC_X:-1   ACC_Z:-104   GYR_Y:7
ACC_X:-1   ACC_Z:-102   GYR_Y:7
ACC_X:-1   ACC_Z:-105   GYR_Y:7
ACC_X:-1   ACC_Z:-103   GYR_Y:7
ACC_X:0   ACC_Z:-99   GYR_Y:7
ACC_X:-1   ACC_Z:-103   GYR_Y:8
ACC_X:-1   ACC_Z:-101   GYR_Y:7
ACC_X:0   ACC_Z:-104   GYR_Y:6
ACC_X:0   ACC_Z:-105   GYR_Y:5
ACC_X:-1   ACC_Z:-103   GYR_Y:5
ACC_X:-3   ACC_Z:-103   GYR_Y:6
ACC_X:-3   ACC_Z:-103   GYR_Y:6
```

```
ACC_X:97   ACC_Z:0   GYR_Y:3
ACC_X:96   ACC_Z:0   GYR_Y:3
ACC_X:97   ACC_Z:-1   GYR_Y:4
ACC_X:97   ACC_Z:-1   GYR_Y:4
ACC_X:97   ACC_Z:-2   GYR_Y:4
ACC_X:97   ACC_Z:0   GYR_Y:3
ACC_X:97   ACC_Z:-1   GYR_Y:4
ACC_X:97   ACC_Z:-1   GYR_Y:4
ACC_X:97   ACC_Z:-1   GYR_Y:3
ACC_X:97   ACC_Z:0   GYR_Y:4
ACC_X:97   ACC_Z:0   GYR_Y:4
ACC_X:96   ACC_Z:-1   GYR_Y:3
ACC_X:97   ACC_Z:0   GYR_Y:4
ACC_X:97   ACC_Z:-1   GYR_Y:4
ACC_X:97   ACC_Z:0   GYR_Y:4
```

```
ACC_X:-103   ACC_Z:1   GYR_Y:6
ACC_X:-103   ACC_Z:2   GYR_Y:6
ACC_X:-102   ACC_Z:1   GYR_Y:6
ACC_X:-103   ACC_Z:2   GYR_Y:6
ACC_X:-103   ACC_Z:1   GYR_Y:6
ACC_X:-103   ACC_Z:0   GYR_Y:6
ACC_X:-103   ACC_Z:2   GYR_Y:6
ACC_X:-103   ACC_Z:0   GYR_Y:6
ACC_X:-103   ACC_Z:0   GYR_Y:6
ACC_X:-103   ACC_Z:1   GYR_Y:6
ACC_X:-104   ACC_Z:1   GYR_Y:6
ACC_X:-104   ACC_Z:1   GYR_Y:6
ACC_X:-104   ACC_Z:1   GYR_Y:6
ACC_X:-103   ACC_Z:0   GYR_Y:6
ACC_X:-104   ACC_Z:1   GYR_Y:6
```

Figure 29 Serial reading or the Accelerometer and Gyro Sensors

7.1 Data fusion

Once the accelerometer and gyroscope raw data were converted to a meaningful values in degree, it was useful to observe the reaction of the measured data using real time sketch, to do this various software such as PLX and processing were used in order monitor the output signals. Data fusion testing describes the reaction of the calculated data for accelerometer and gyroscope based on the processing real time data and then compare it to the measured filtered output. Finally various approaches were used to compare the output value of the sensor fusion, such as Kalman filter, complementary filter and DMP.

7.2 Comparison of tilt angle measured signal on a real time graph

Figure 30 ,was sketched using the PLX software to monitor the real time data of each angle measurement and to make a practical comparison upon the reaction of the measurement based on the movement of the IMU. In the following graphs the x- axis represent the current time and the y-axis represent the pitch angle of the IMU sensor, there are various signal plotted where the gyroscope is coloured in green, the accelerometer is coloured in red and the Kalman filter is coloured in blue.It was noticed from the graph when the IMU sensor is moving steading both accelerometer and gyroscope measured signal is similar to the Kalman signal. Therefore these sensor can be used as tilt sensors without the need of filtering in steady motion application.

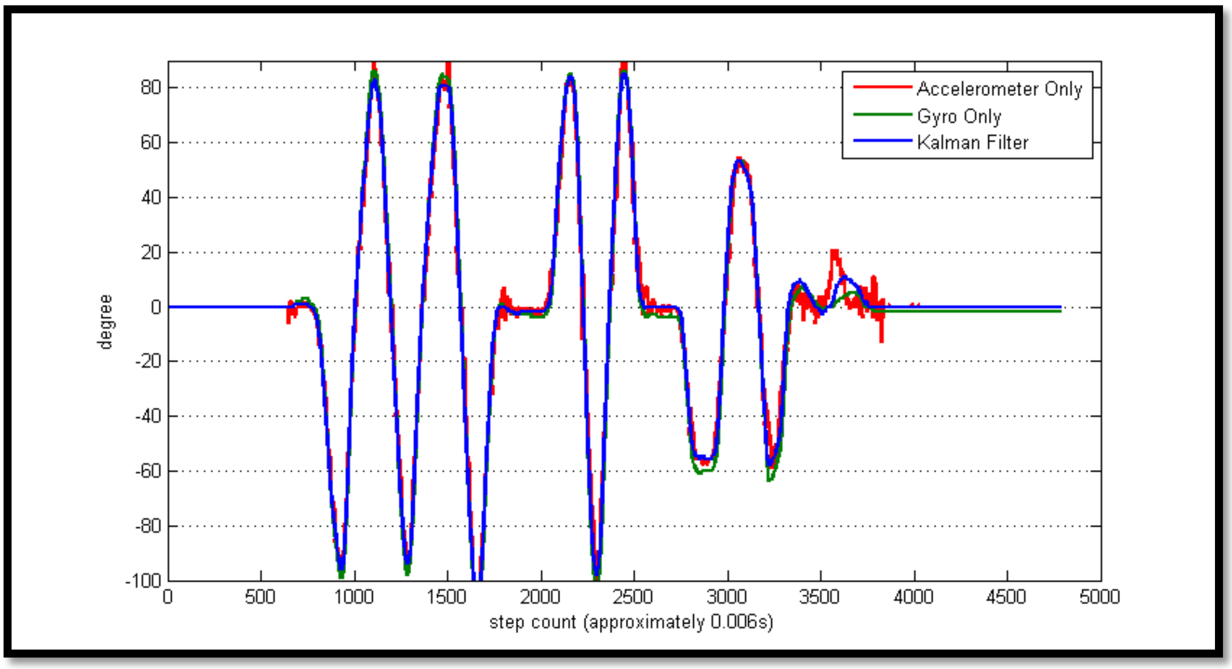


Figure 30 steady motion graphical representation of Kalman filter

Figure 31, represented the unsteady motion signal respond of the IMU sensor, it was noticed by shaking and moving the sensor, the measured signal produced by the accelerometer and the gyroscope differ from the filtered measured signal. The gyroscope measured signal is more stable than the accelerometer measured signal as there is a significant change appearing in the accelerometer signal than the gyroscope .

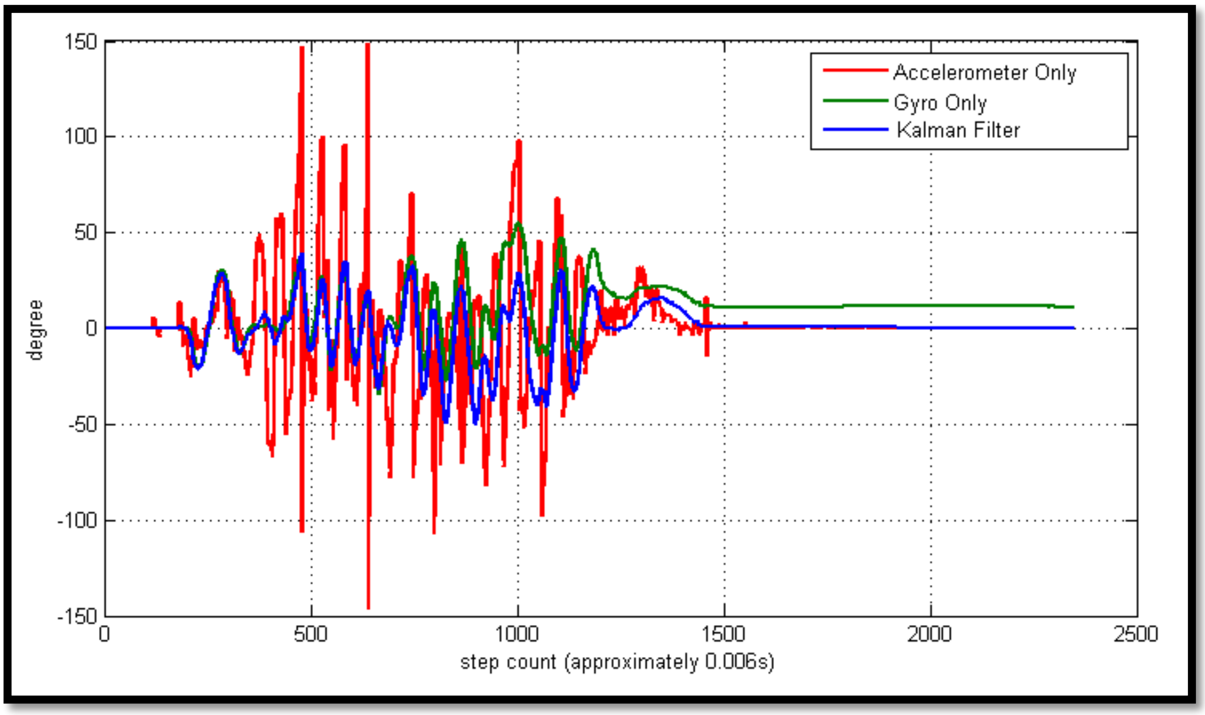


Figure 31 unsteady motion graphical representation of Kalman filter

7.3 Processing cube test

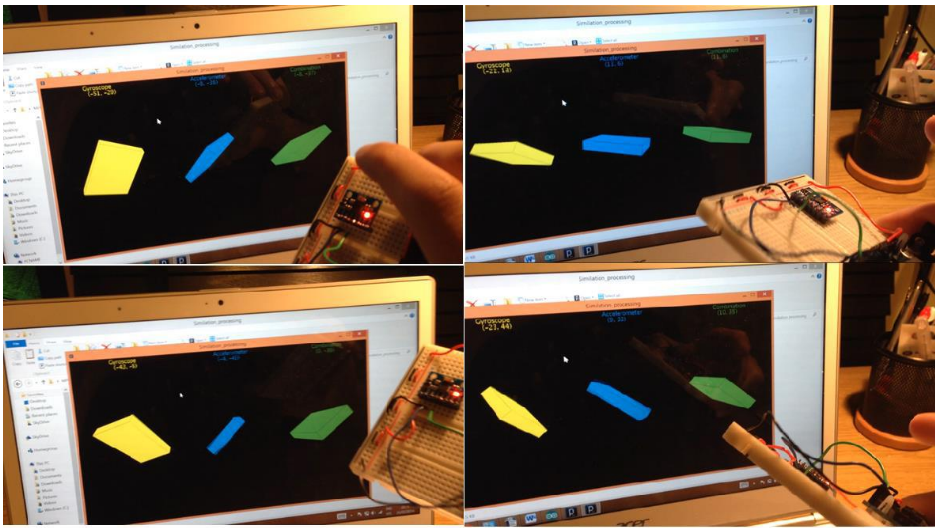


Figure 32 MPU6050 Cube test

Figure 32 ,is the processing cube simulation which was carried out to observe the reaction of each tilt sensor based on the movement of the IMU sensor , the complementary filter represent the green cube , the accelerometer represents the blue cube and the gyroscope represent the yellow cube. It was observed by moving the gyroscope senor the yellow cube drifted over time. This can be also noticed in left pictures shown in figure 32, where there IMU sensor is in a different position to the to the gyroscope cube . It was noticed by shaking and fast motion of the IMU sensor the accelerometer cube would get effected and the cube position was differed from its original position, this was due to the nosiness of the accelerometer sensor. The complementary filter cube or the green cube was moving constantly in the same position as sensor accordingly. This indicates that the algorithm used to calculate the complementary filter is reliable based on simulaton cube testing .

7.4 Test phase III - Controlling motors

In this project MD25 motor driver was used to control the speed and direction of the motors using serial communication, the frequency baud for data transmission can be varied from 9600 – 115200 for serial data. The initial test carried out to control the motors was to control the speed, direction and position of the motors, then the encoders where interfaces with the motors ito count the pulse received as shaft rotates.

Experiment Description	Pass fail	Comment
Speed Control	Passed	Initially the speed of the motors where varied by sending different values to the MD25
Direction Control	Passed	It was possible to control the direction of rotation of the shaft in various speed , either in forward direction or in backward direction
Encoder data	Passed	It was possible to read the encoder pulse using the serial monitor.
Controlling the motors based on the encoder data	Passed	A control loop was implemented to control the speed and direction of the motors based on the encoder pulse reading.
Monitoring the voltage and current of the battery	Passed	It was possible to monitor the voltage and current of the DC battery connected to the motor driver.

Table 4 MD25 test

7.5 Open loop test

This experiment describe the reaction of the bslsncing robot without feedback loop by controlling the motors based on the inclination angle , where the motors change direction based on the sign of the angle there as the MD25 recives a negative angle the robot moves in backward direction with a full speed vice versa receiving a positive angle the robot would move in the forward direction . The pseudo code representation of the control loop is as follow:

```
When Angle = 0
        Robot will stop
When Angle < 0
```

Robot will move forward

When Angle > 0

Robot will move backward

Experiment Description	Pass or Fail	Description
Interfacing the IMU sensor with the motors	Pass	The pitch value of the IMU sensors where used to drive the direction of movement of the motors
Controlling the movement of the by an open loop test	Pass	In was possible to control the motors direction of rotation based on IMU reading
Balancing the robot based on the open loop experiment	Fail	The robot was not able to balance due to the fast movement when receiving a negative or a positive value and producing high level of overshoot.

Table 5 Open loop experiment

7.6 PID feedback controller testing and simulation by LabVIEW

The initial test carried out to understand the characteristic and behaviour of the PID feedback controller was to visualise the real time graph of the measured output of the PID based on the NI LabVIEW graphical system design software. To do this the Quanser Engineering Trainer (QNET) DC motor was connected to the NI ElVIS board in order to control the speed and position of the DC encoder motor based on measured values with the provided LabVIEW GUI. In this experiment the PID parameters will be adjusted based on the reference point and the signals reaction will be plotted on a graph. Figure 33, is the NI ELVIS board interfaced with the QNET DC motor.

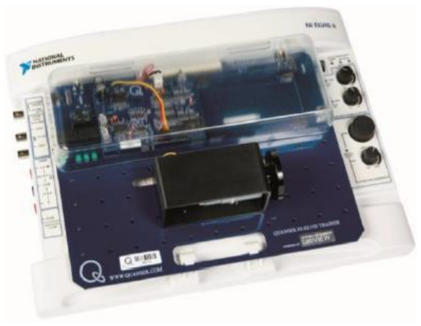


Figure 33 Quanser Engineering trainer board interfaced with the NI ELVIS board

7.6.1 PID feedback controller design

In the LabVIEW experiment every parameter of the PID controller where set to zero and tested individually, initially a PD control test was carried by setting the integral gain to zero and tuning proportional and derivative values individually to observe the output reaction , provided a graphical representation which be described in the following section .

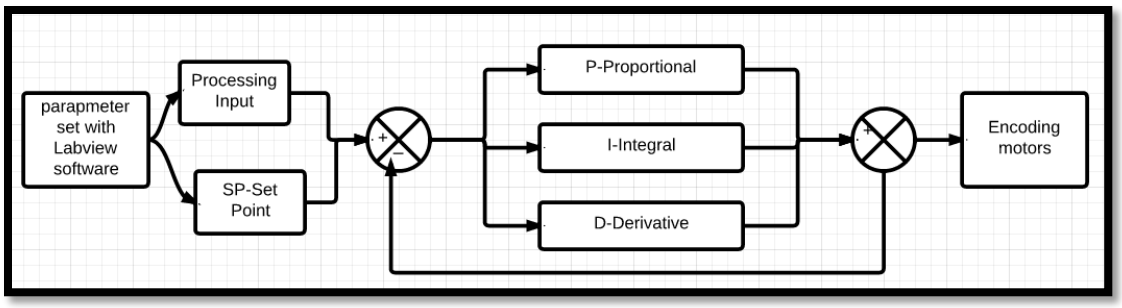


Figure 34 LabVIEW design, Block diagram representation of PID feedback loop

7.6.2 Proportional parameter adjustment

In order to adjust and observe the behaviour of the proportional gain, all the parameter gain were set to zero and the value of proportional test was increased in order to adjust its measured valued according to its reference position, where it can be seen in figure 35, the blue line represent the reference position or thee set-point and the red line represent the measured value or the output signal of the PID.

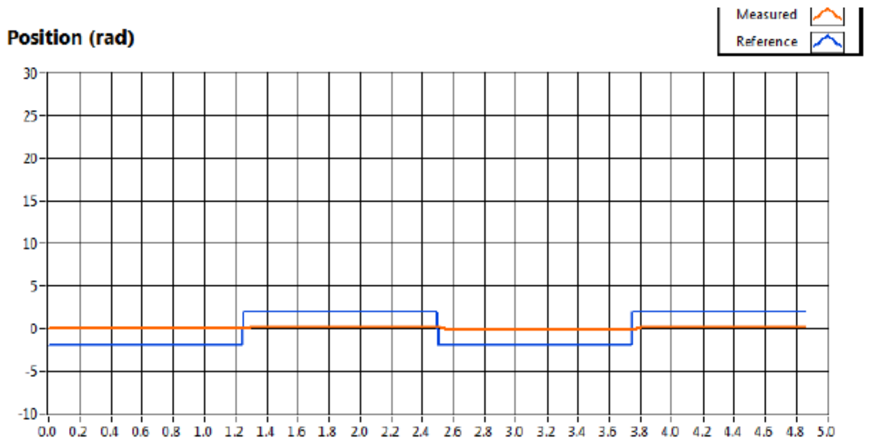


Figure 35 Kp = 0 , Ki = 0 , Kd = 0

It was observed by setting all the parameter of the proportion, integral and derivative to zero the reference line remains on constant zero position and the motors remain in a stationary position as there is no torque applied to the motors.

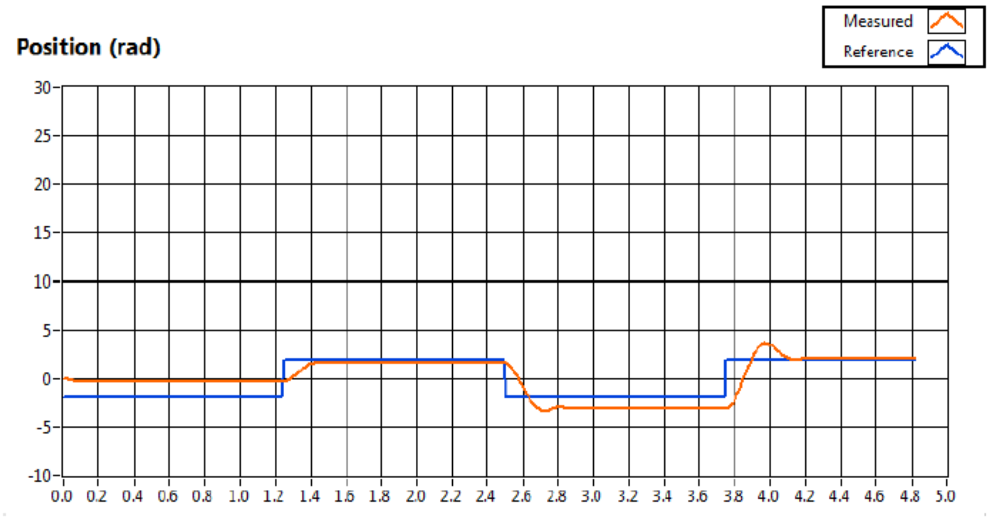


Figure 36 Kp = 0.5 , Ki = 0 , Kd = 0

By increasing the proportional coefficient to 0.5, the measured position gets closer to the set point, however the measured proportional output is not ideal to the original reference position which can be noticed in figure 36 , adding to this there is a slight overshoot appearing in the PID measured signal as the motor changes position.

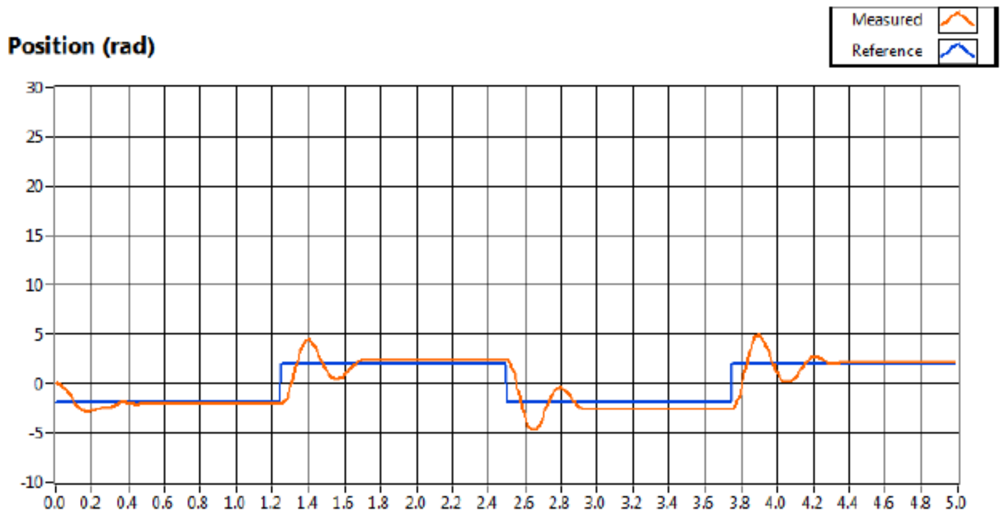


Figure 37 Kp = 1.5 , Ki = 0 , Kd = 0

By increasing the value of proportional gain to 1.5 it can be noticed , there is improvement in change of position of the graph as it can be observed in figure 8 , the measured value get closer to the reference position , however this is not ideal as the measured value is not position perfectly as the reference position ,this can be noticed in the graph at 3.2 to 3.6 seconds the overshoot increases as the motor changes position.

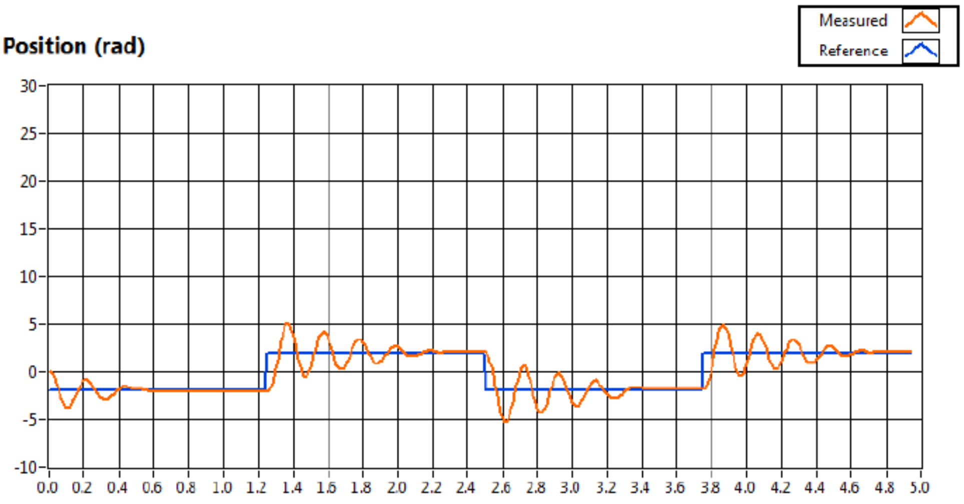


Figure 38 Kp = 4.5 , Ki = 0 , Kd = 0

Continuing to increasing the value of proportional gain to 4.5 it can be noticed, there is a oscillation appearing in the system as the motor changes position and the oscillation increase as Kp increase from 1.5, where the steady state error increases as Kp increases.

As a result it was noticed as the proportional gain further increases from 1.5, the oscillation tends to increase and the error of the system increases significantly. Decreasing the value of proportional gain from 1.5 the steading state error increases where the measured value does not represent the set point position. Therefore the proportional gain will be set at 1.5 because at this point measured value is closer to the reference position with minimal oscillation and steady state error.

7.6.3 Derivative parameter adjustment

Having adjusted the value of the proportional gain to 1.5, the derivative coefficient will be adjusted in order to observe the reaction of the system by varying the derivative constant and monitoring its reaction on real time graph. In this experiment proportional gain is set to 1.5 which is derived in the previous experiment and the value of the derivative constant is increased gradually from zero.

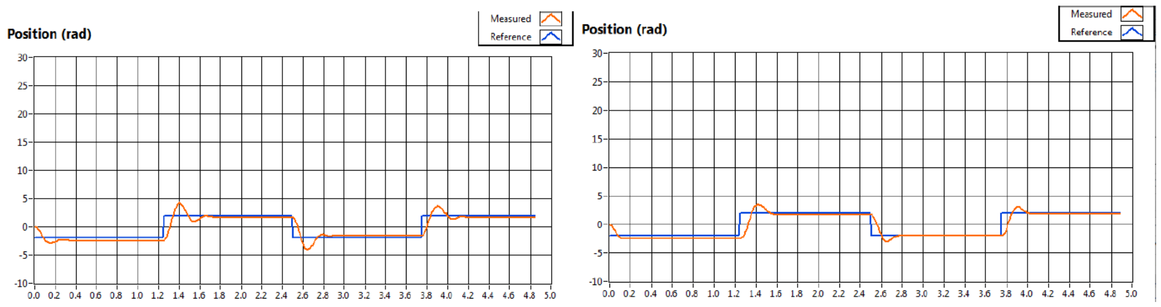


Figure 39 increasing the value of Kd from 0 to 0.4

By increasing the value of derivative constant from 0 to 0.4 it can be noticed in figure 39, the overshoot appearing in system is reduces, therefore the derivative gain is increased until the overshoot gets eliminated , which can be seen in figure40, by setting Kd to 0.6 it can be observed the overshoot is eliminated .

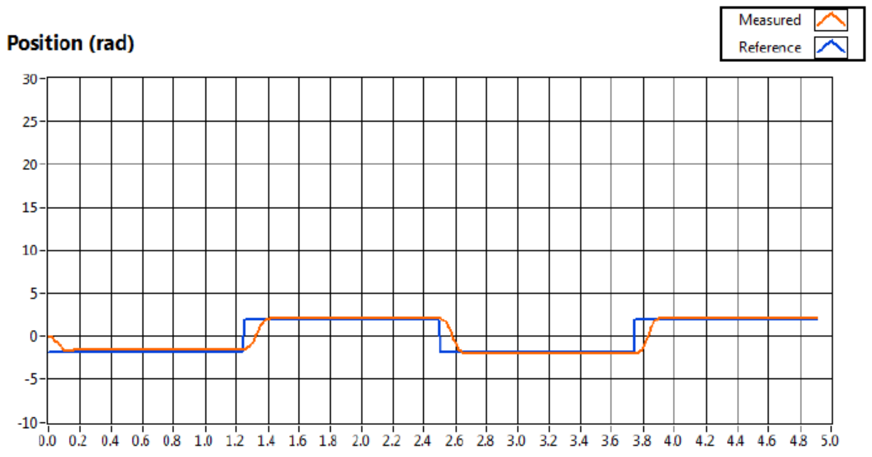


Figure 40 Kp = 1.5 Ki =0 Kd = 0.6

In conclusion by using a PD feedback controller we can adjust the measured position to represent the reference position, where the proportion term is used to reduce the steady state error appearing in the system and the derivative term is used to reduce and eliminate the oscillation and overshoot of the system.

7.7 PID Tuning manually and by Ziegler-Nicholas model

In this experiment the stability robot was tuned manually and by Ziegler-Nicola approach [14] was used to estimate the parameter gains using Ziegler equation. In order to demonstrate the graphical signal of the PID output the PLX software was used to observe the reaction of the robot in the process of tuning the PID parameters.to change the gains a potentiometer was used , while monitoring the values on the LCD display. This section will describe calculation used to Ziegler-Nicholas manual tuning approach, every term of the proportional , integral and derivative gain are referred as Kp , Ki and Kd and ultimate gain is referred as Ku. The following calculation represent the Zigler-Nichols equations to approximate the parameters of the PID gain, where Pu represent the oscillation which can be derived from the graph, after making approximation of the PID values the, PID will be tuned manually based on the observation of reaction of each parameter which was carried out using the LabVIEW simulation .

$$Kp = Ku \times 0.6 \tag{7.1}$$

$$Ki = \frac{2 \times Kp}{Pu} \tag{7.2}$$

$$Kd = \frac{kp \times Pu}{8} \tag{7.3}$$

In this approach every parameter of PID gain is set to zero and the reference position was set to zero .initially proportional gain is increased unit the robot starts to oscillates and reaches the ultimate gain value of Ku , where at this point the robot initiate to oscillation , this can observed in figure 41 , where the robot oscillates in a range of -10 to 10 degrees and two oscillation scores in a period of 5 seconds .further increasing the value of Kp the number of oscillation occurring a period of time, which result in overshoot in the system ,therefore the robot does not remain stable for any longer than 20 seconds and reducing the Kp value , the motors respond slower to change of angle and does not perform an oscillation in order balance the robot.

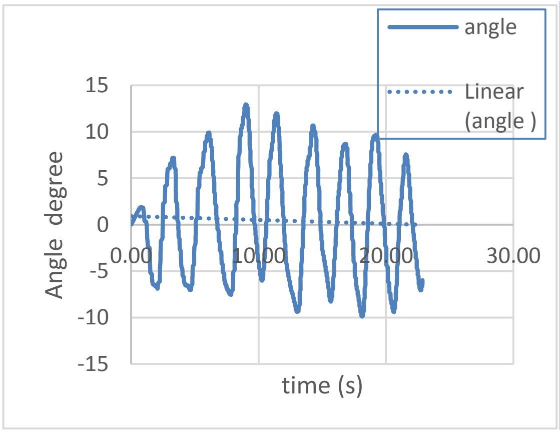


Figure 41 $Kp = 300, Ki = 0, Kd = 0$

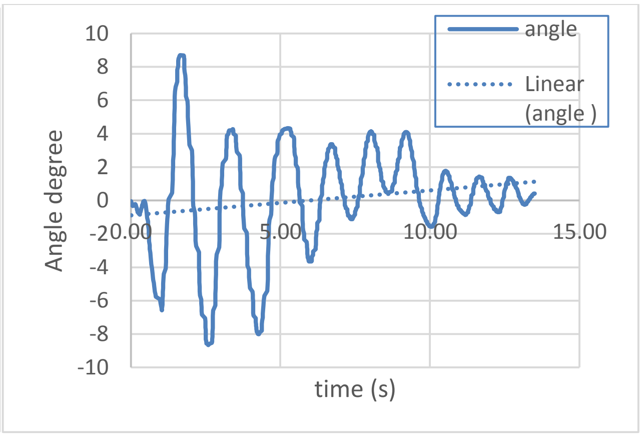
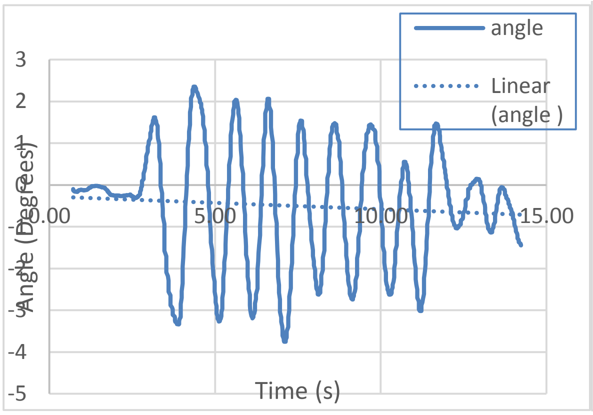


Figure 42 $Kp = 300, Ki = 0.01, Kd = 0$

Onces the proportional gain value is set to a value that initiate the oscillation and reaches the ultimate gain value, the derivative gain is increased gradually from zero, until the oscillation is removed. It can be observed by figure 42, that the oscillation reduces when disturbance is applied to the robot and the PID output signal gets closer to zero reference position over period of time. We continue tuning Kp and Kd unit the robot is perfectly balances.



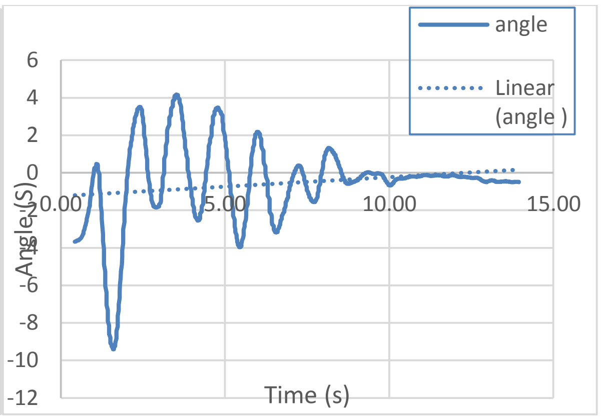
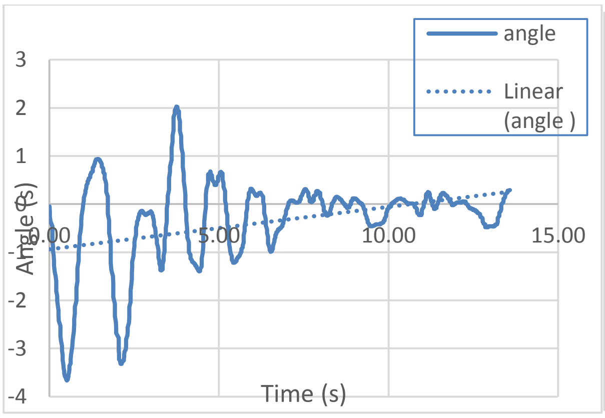


Figure 43 $Kp = 350, Ki = 0.5, Kd = 15$

Figure 44 $Kp = 370, Ki = 0.015, Kd = 17$

Having tuned the values of Kp and Kd , the integral gain Ki is increased in order to improve the stability of the robot .by adding the integral gain to feedback algorithm it was observed that the output signal of the PID increases significantly which result in integral winding effect , in order to overcome the integral winding the out value of the integrate was constrained to prevent the PID output signal from significant changes .it was observed by increasing the value of Ki the robot oscillates to its equilibrium position with faster speed ,this can be observed in figure 43 . Further increasing the value of Ki it can be noted in figure 44 the overshoot increases which causes the robot to be unstable.

Once all the parameter of proportional, integral and derivative gain is tuned the robot should able to balance with a minimal oscillation and return to its reference position, with a faster speed and in low period of time. The ideal balancing can be viewed in figure 45 and 46 , where the PID signal return to a steady state position in period of approximately 6seconds .



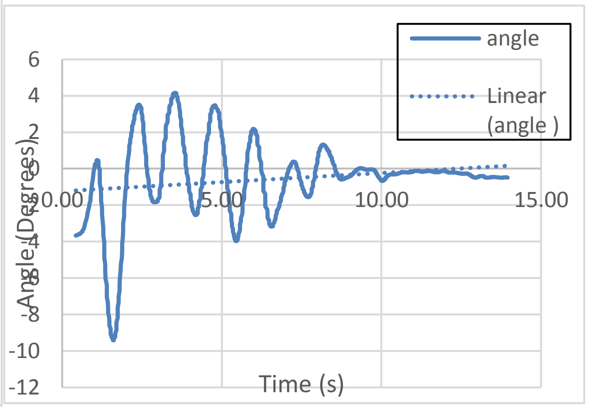


Figure 45 $Kp = 350, Ki = 0.01, Kd = 20$

Figure 46 $Kp = 370, Ki = 0.015, Kd = 17$

8 Discussion

This project has been successful in achieving it aim of stabilising a two-wheel autonomous robot in its vertical position based on the inverted pendulum model. This section will provide summary of the project contributions and achievements and will discuss the possible future work that can be carried out to improve the robot.

8.1 Project review

Various sensor fusion techniques have been implemented to address the problem of angle measurement. A Digital Motion Processor (DMP) and a linear Kalman filter were successfully implemented to eliminate the noise of the accelerometer and the drift of the gyroscope sensor. To ensure the reliability and effectiveness of the angle estimation, the signal information was plotted using real time PLX-Exect graphs and a processing cube test to observe the reaction of the MPU6050 based on the calculations.

Two linear control strategies were implemented to address the problem of the robots' stabilization and robustness. A control strategy was developed by using a state space feedback model. The dynamics were derived based on the Newton's second law of motion and the controller was implemented based on the linearized model derived from mathematical equations around the operating point. The linearized model uses the Matlab function to obtain the gains of the full state feedback controller.

A Proportional Integral Derivative (PID) algorithm was implemented to balance the robot in its equilibrium position. The parameters of the PID were obtained based on the manual tuning and the Ziegler-Nicholas techniques. The PID feedback controller was improved by introducing a position control element to the feedback loop. This was done by adjusting the set-point based on the position of the robot from its balancing position.

An Android MIT application inventor was designed and developed in order to send voice commands to the robot. This feature has been successfully implemented and tested to control various electronic devices such as displaying the required data on LCD and being able to control the movement of the robot by voice commands. However to achieve perfect navigation a faster motors with better torque are required.

8.2 Future work recommendations

To further improve this project the following recommendation can be suggested. Improvement of the angle estimation can be done by measuring the optimal values of the Kalman filter gains. Based on the dynamic model of the system and using an advance sensor fusion approach to combine multiple sensors, such as a magnetometer and an ultra-sonic sensor for more reliable and accurate angle estimation. Improving the control strategy by adapting an advance controller such as Fuzzy logic, LQR and LQG and obtaining the states of the controller based on dynamic model of the systems. Making use of the 'Gyrometry' as this approach improves the stability of the robot by reducing the odometer error caused by the robot, when slipping on low friction surfaces.

References

[1] David-Anderson (27th.January.2003).”nBot-Balancing-Robot”. united state: Fred Kaufman. Availble at:www.geology.smu.edu/dpa , Last access :20th.August.2015

[2] Felix Grasser,Silvio Colomb, Aldo D’Arrigo, i, and Alfred C. Rufer,. (10th.July.2002). JOE. A Mobile, Inverted Pendulum. IEEE.pp1-5.

[3] Mary-Bellis. (8th.October.2010). “Segway Human Transporter”. Available: http://inventors.about.com/. Last accessed : 26.Agust.2015

[4] Shuuji Kajita, Hirohisa Hirukawa, Kensuke Harada, Kazuhito Yokoi (31st.Oct.2005). “Introduction to Humanoid Robotics”. Place: Japan: ohmsha . pp35-40.

[5] Williams, V. Kyush Matsuoka, K. (18-21st.Nov.1991). “Learning to balance the inverted pendulum using neural networks” , IEEE International Joint Conference . pp1-6.

[6] Nizar Al-Holou, Tarek Lahdhiri, Dae Sung Joo, Jonathan Weaver, and Faysal Al-Abbas (17th.March.2012) , “Sliding Mode Neural Network Inference Fuzzy Logic Control for Active Suspension Systems” : IEEE. pp1-13.

[7] Nuraddeen Magaji, Lukman A. Yusuf (18th.Mar.2013) “GA-PID Controller for Position Control of Inverted Pendulum” Electrical Engineering Bayero University : IEEE. pp1-2

[8] Pathak, K. ; Dept. of Mech. Eng., Delaware Univ., Newark, DE, USA ; Franch, J. ; Agrawal, S.K. (14-17th.Dec. 2004). “Velocity control of a wheeled inverted pendulum by partial feedback linearization. "IEEE Conference on Decision and Control, 2004. CDC. 43rd " : IEEE. pp1-6.

[9] Doskocz, E (10th.Mar.1998). “ Help Working with Abstracts MIMO sliding mode control of a robotic pick and place" , Proceedings of the Thirtieth Southeastern Symposium on: IEEE. pp1-5.

[10] Liang Sun ‚Jiafei Gan (28thSep.2011). “Researching Of Two-Wheeled Self-Balancing Robot Base On LQR Combined With PID”. University of Technology Institute of Artificial Intelligence and Robotics Beijing: IEEE. pp1-2

[11] R.E.Kalman (15th.July.1960). “A New Approach to Linear Filtering and Prediction Problems1”, Research Institute for Advanced Study,2 Baltimore, Md: CS. P1.

[12] Billur Barshan , Hugh F. Durrant-Whyt (3rd.June.1995). “Inertial Navigation Systems for Mobile Robots”: IEEE. pp1.

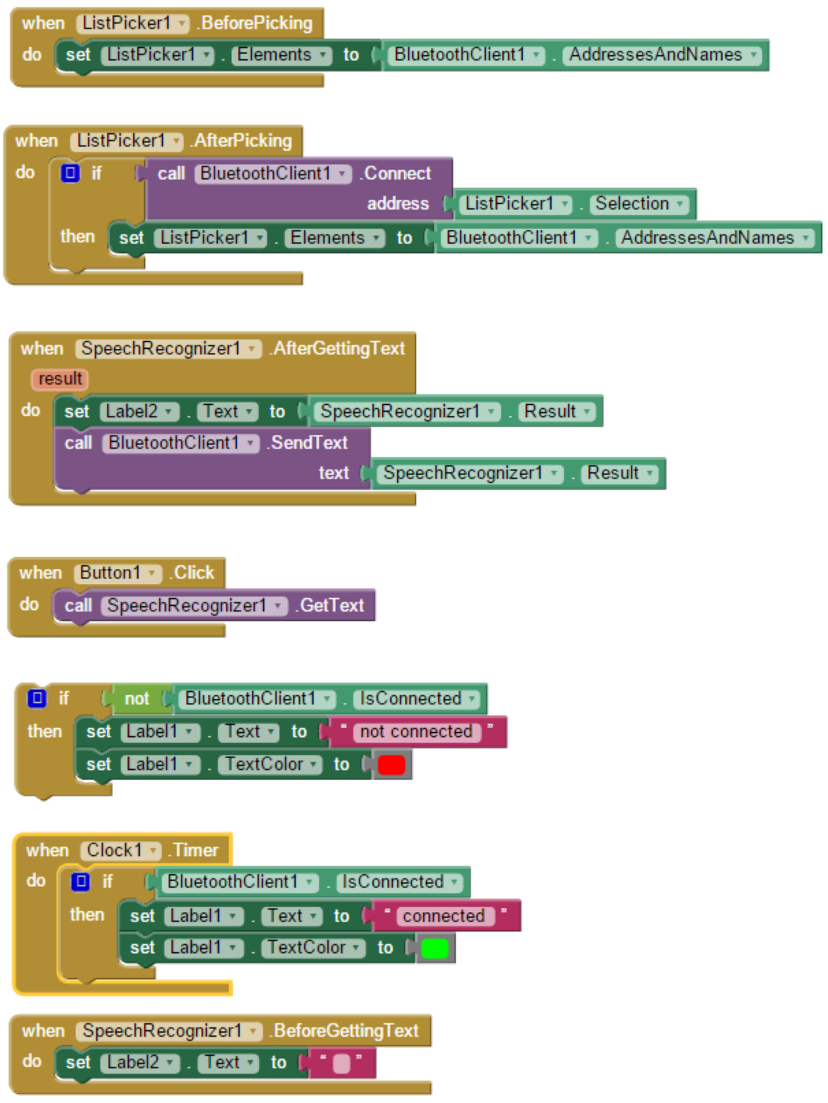
[13] Johann Borenstein, Liqiang Feng (19th.October.1996). “Measurement and Correction of Systematic Odometry Errors in Mobile R.obots” .: IEEE. pp1-2.

[14] Tomas.B.Co (12th.Feb.2004.). Ziegler-Nichols-Method:. Available:http://www.chem.mtu.edu/~tbco/cm416/zn.html. Last accessed 26th July 2015

[15] Smith M.J. (12th.Feb.2013). introduction to serial communication .Available: http://www.fiz-ix.com/2013/02/introduction-to-arduino-serial-communication/. Last accessed 28th Aug2015.

Appendix A

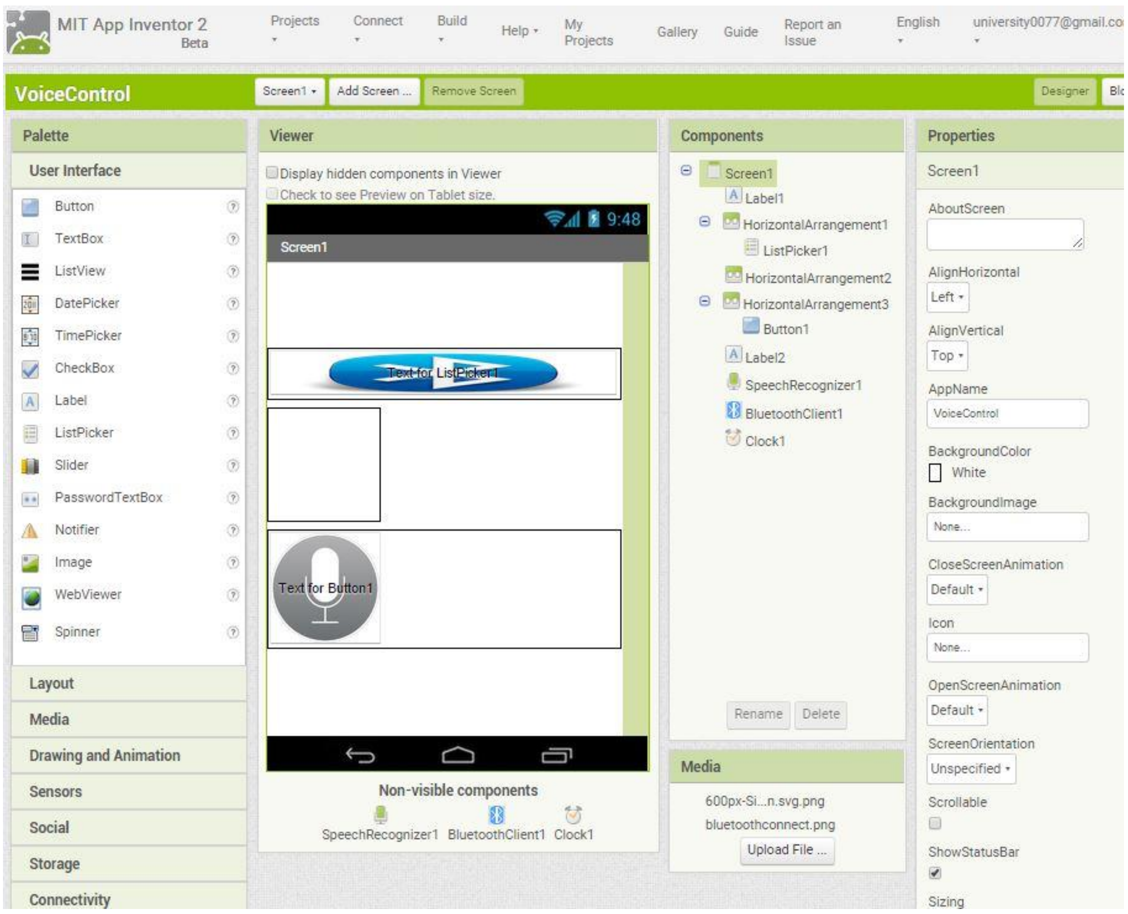
Backend implemntation for the Voice regoncion:



```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames

when ListPicker1 .AfterPicking
do  if    call BluetoothClient1 .Connect
                                address ListPicker1 . Selection
    then  set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames

when SpeechRecognizer1 .AfterGettingText
 result
do  set Label2 . Text to SpeechRecognizer1 . Result
    call BluetoothClient1 .SendText
                                text SpeechRecognizer1 . Result

when Button1 .Click
do  call SpeechRecognizer1 .GetText

if    not BluetoothClient1 . IsConnected
then  set Label1 . Text to " not connected "
      set Label1 . TextColor to 🟥

when Clock1 .Timer
do  if    BluetoothClient1 . IsConnected
    then  set Label1 . Text to " connected "
          set Label1 . TextColor to 🟩

when SpeechRecognizer1 .BeforeGettingText
do  set Label2 . Text to " "
```

APPENDIX B

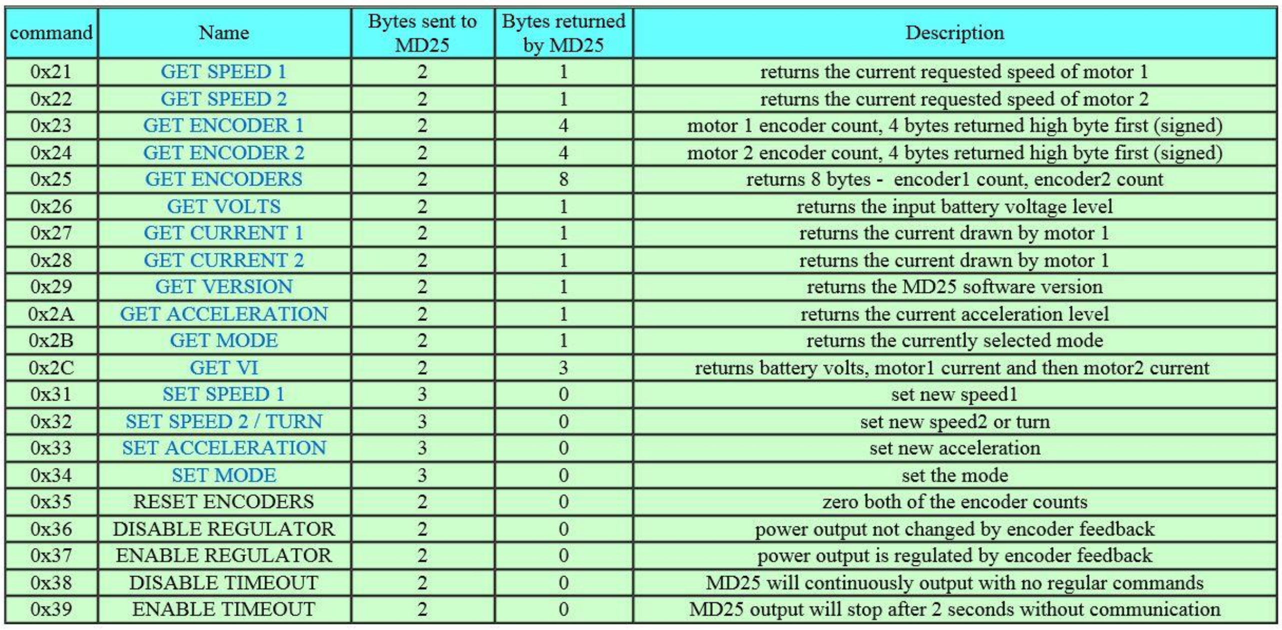
Front End MIT App Design:



Screen1
AboutScreen
AlignHorizontal: Left
AlignVertical: Top
AppName: VoiceControl
BackgroundColor: White
BackgroundImage: None...
CloseScreenAnimation: Default
Icon: None...
OpenScreenAnimation: Default
ScreenOrientation: Unspecified
Scrollable: ☐
ShowStatusBar: ✓
Sizing

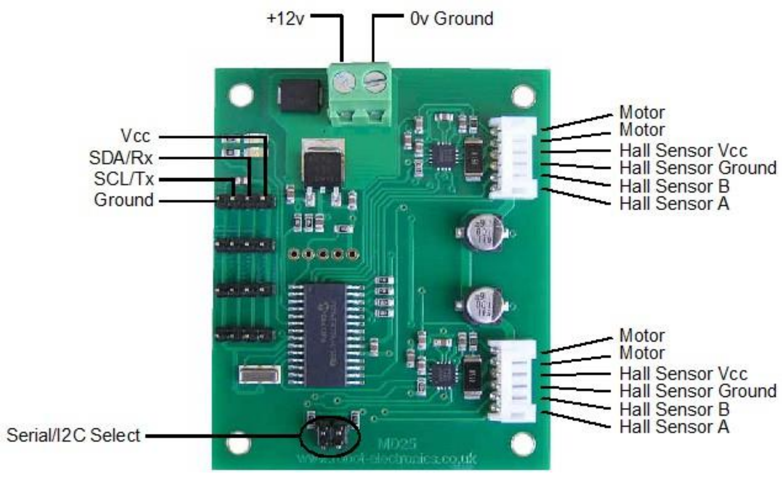
APPENDIX C

Table 6 serial communication address for MD25



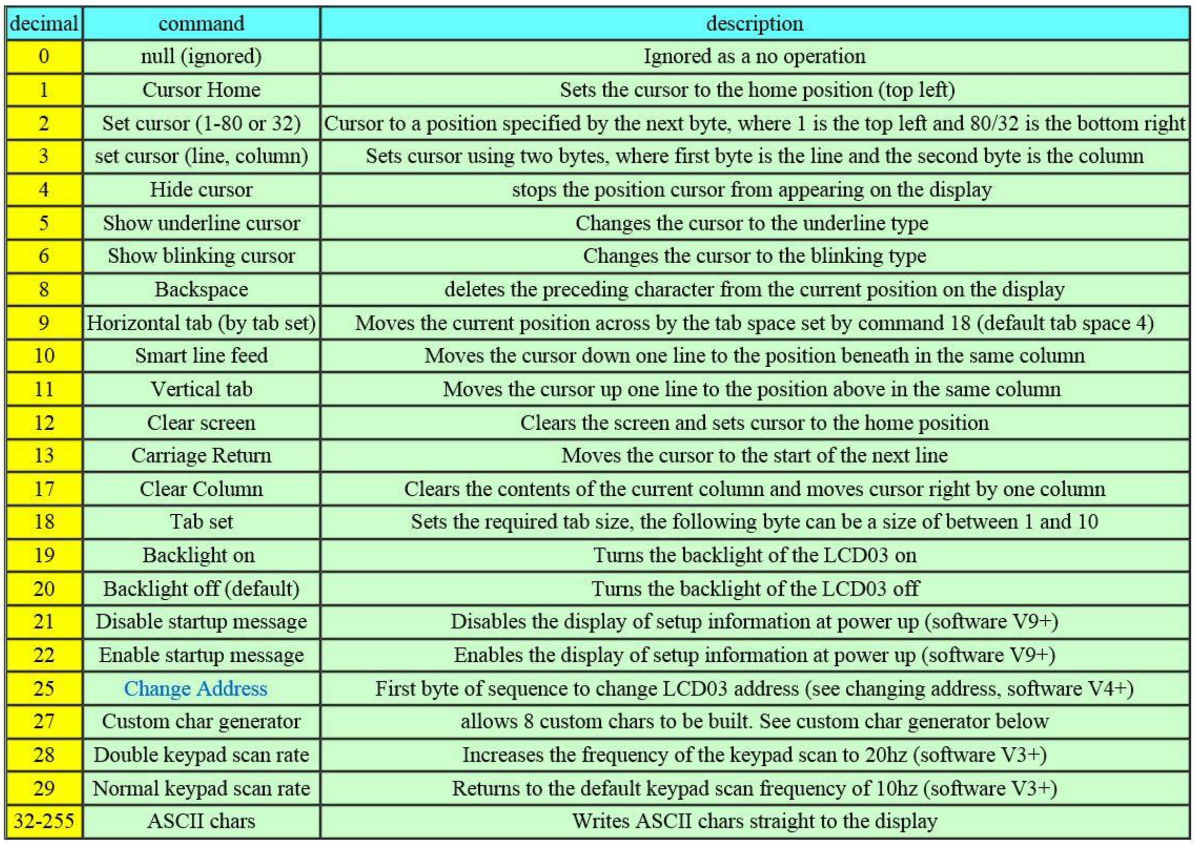
command	Name	Bytes sent to MD25	Bytes returned by MD25	Description
0x21	GET SPEED 1	2	1	returns the current requested speed of motor 1
0x22	GET SPEED 2	2	1	returns the current requested speed of motor 2
0x23	GET ENCODER 1	2	4	motor 1 encoder count, 4 bytes returned high byte first (signed)
0x24	GET ENCODER 2	2	4	motor 2 encoder count, 4 bytes returned high byte first (signed)
0x25	GET ENCODERS	2	8	returns 8 bytes - encoder1 count, encoder2 count
0x26	GET VOLTS	2	1	returns the input battery voltage level
0x27	GET CURRENT 1	2	1	returns the current drawn by motor 1
0x28	GET CURRENT 2	2	1	returns the current drawn by motor 1
0x29	GET VERSION	2	1	returns the MD25 software version
0x2A	GET ACCELERATION	2	1	returns the current acceleration level
0x2B	GET MODE	2	1	returns the currently selected mode
0x2C	GET VI	2	3	returns battery volts, motor1 current and then motor2 current
0x31	SET SPEED 1	3	0	set new speed1
0x32	SET SPEED 2 / TURN	3	0	set new speed2 or turn
0x33	SET ACCELERATION	3	0	set new acceleration
0x34	SET MODE	3	0	set the mode
0x35	RESET ENCODERS	2	0	zero both of the encoder counts
0x36	DISABLE REGULATOR	2	0	power output not changed by encoder feedback
0x37	ENABLE REGULATOR	2	0	power output is regulated by encoder feedback
0x38	DISABLE TIMEOUT	2	0	MD25 will continuously output with no regular commands
0x39	ENABLE TIMEOUT	2	0	MD25 output will stop after 2 seconds without communication

MD25 dual H-bridge board connection connections:



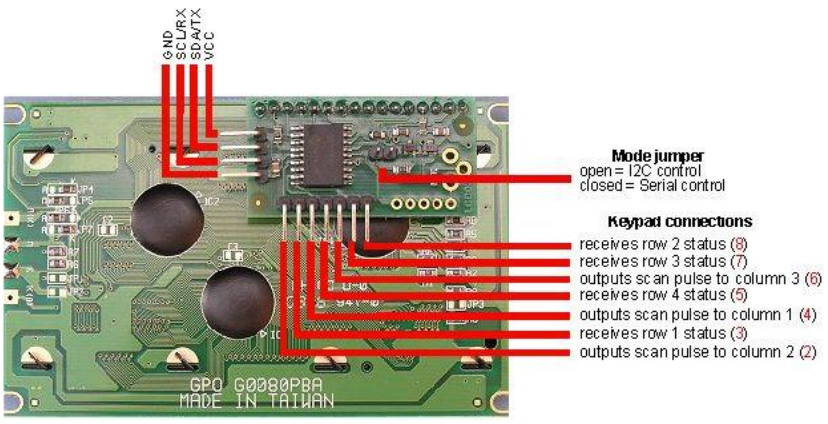
APPENDIX D

Table 7 Serial communication for LCD03



decimal	command	description
0	null (ignored)	Ignored as a no operation
1	Cursor Home	Sets the cursor to the home position (top left)
2	Set cursor (1-80 or 32)	Cursor to a position specified by the next byte, where 1 is the top left and 80/32 is the bottom right
3	set cursor (line, column)	Sets cursor using two bytes, where first byte is the line and the second byte is the column
4	Hide cursor	stops the position cursor from appearing on the display
5	Show underline cursor	Changes the cursor to the underline type
6	Show blinking cursor	Changes the cursor to the blinking type
8	Backspace	deletes the preceding character from the current position on the display
9	Horizontal tab (by tab set)	Moves the current position across by the tab space set by command 18 (default tab space 4)
10	Smart line feed	Moves the cursor down one line to the position beneath in the same column
11	Vertical tab	Moves the cursor up one line to the position above in the same column
12	Clear screen	Clears the screen and sets cursor to the home position
13	Carriage Return	Moves the cursor to the start of the next line
17	Clear Column	Clears the contents of the current column and moves cursor right by one column
18	Tab set	Sets the required tab size, the following byte can be a size of between 1 and 10
19	Backlight on	Turns the backlight of the LCD03 on
20	Backlight off (default)	Turns the backlight of the LCD03 off
21	Disable startup message	Disables the display of setup information at power up (software V9+)
22	Enable startup message	Enables the display of setup information at power up (software V9+)
25	Change Address	First byte of sequence to change LCD03 address (see changing address, software V4+)
27	Custom char generator	allows 8 custom chars to be built. See custom char generator below
28	Double keypad scan rate	Increases the frequency of the keypad scan to 20hz (software V3+)
29	Normal keypad scan rate	Returns to the default keypad scan frequency of 10hz (software V3+)
32-255	ASCII chars	Writes ASCII chars straight to the display

LCD03 daughter board connections:


GND
SCL/RX
SDA/TX
VCC
Mode jumper
open = I2C control
closed = Serial control
Keypad connections
receives row 2 status (8)
receives row 3 status (7)
outputs scan pulse to column 3 (6)
receives row 4 status (5)
outputs scan pulse to column 1 (4)
receives row 1 status (3)
outputs scan pulse to column 2 (2)
GPO G0080PBA
MADE IN TAIWAN

APPENDIX E

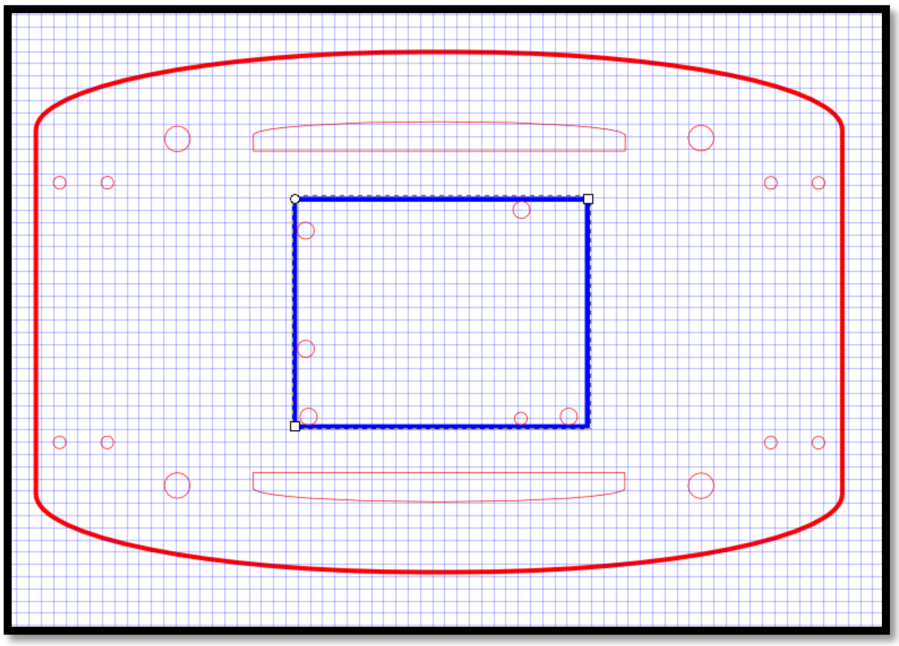


Figure 46 Layer two

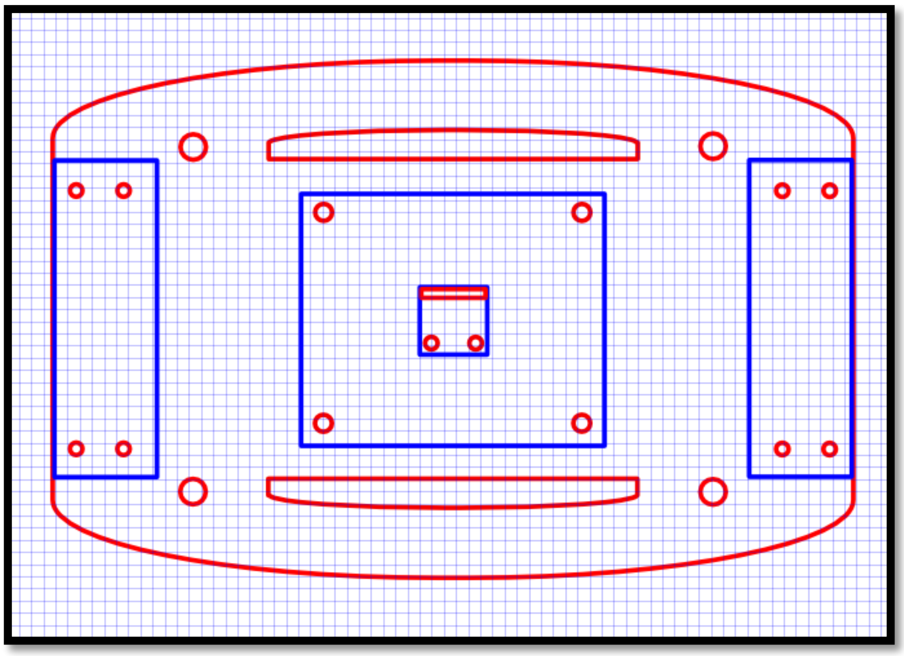


Figure 47 Layer one