Palestine Polytechnic University

College of Engineering

Graduation Project

Inverted Pendulum Robot

By:

Anan Barari Ahmad Zoghier

Supervisor:

Dr. Jasem Tamimi

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

Palestine Polytechnic University

December, 2018

Abstract

inverted pendulum robot is an applied research project for the design, analysis and construction of an electric robot with two parallel wheels. The robot works to maintain its balance. The robot accelerates forward and backward according to angle of inclination. The robot can rotate in a smooth manner. This project is a challenge in application and research. The robot maintains its balance, and this is contrary to the nature of the system where it is expected to fall. The challenge is to create a system and a control algorithm that works to maintain the balance of the robot using acceleration sensors and angle of inclination. This project targets the local, industrial and service sectors. Any company can use this robot and transfer equipment. It has the advantage of quick movement and the possibility of circumventing the narrow spaces. It consists of a mechanical structure driven by two electric motors and controlled by a robot-based control system. The sensor provides the control system with information about the condition of the robot. In this report we will be followed mechatronics method to build the robot to be a complete system of mechatronics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Inverted pendulum robot (IPR) is a self balance robot that is based on inverted pendulum theory. This robot is unstable system as well as has a nonlinear dynamic. In an inverted pendulum robot the inverted pendulum is coupled with two wheels. This pendulum always needs to be in upper vertical position as well as moves in a desired direction. To maintain it in a vertical position it must exceed an external force. In this project we will build this IPR to understand the nonlinear dynamic and behind it to solve the unstability problem. Then we use it to perform some tasks to drive the IPR in a desired direction. In addition, we will use one proportional integral derivative (PID) controller with optimal control and kalman filter to defuse an accelerometer and gyroscope signals to estimate the next position of robot. In the literature, many researchers have used and studied the IPR in their works, for example:

Shilpa et al [8] used a linear quadratic regulator (LQR) optimal control to incorporate all state variables to calculate the control signal value, and integral sliding mode control (ISMC) it used in a nonlinear system to achieve the set point control task.

Barjeev Tyadi et all [7] built an IPR by using LQR control from optimal control which take a states of the dynamic system and control input to make the optimal control dicision, and they used PID controller to stabilize the IPR in the upright position.

Imad Ali and Modasser Hossen [1] used PID controller with Kalman filter to defuse an accelerometer and gyroscope signals to optain on correct value.

From the above we conclude that there are many ways to build the controller for IPR, here, we build linear quadratic regulator to minimization of IPR (reduce the energy consumption of the system), and design PID controller to stabilize the robot. Additionally used several mechanical and electrical component to build IPR. Accelerometer and gyroscope have (MPU6050) is selected in this robot to detect tilt angle (angle between pendulum and Ground)then send the signal to microcontroller that contain input/output pins to receive and sent data. The microcontroller receive the signal from accelerometer and gyroscope sensor then send signal to motor driver witch use to control in speed and direction of motor and to provide the DC gear motor in voltage needed to run it. We used two DC gear motors that are connected with two wheels to move the IPR. Fig 1.1 show the structure of IPR.

.

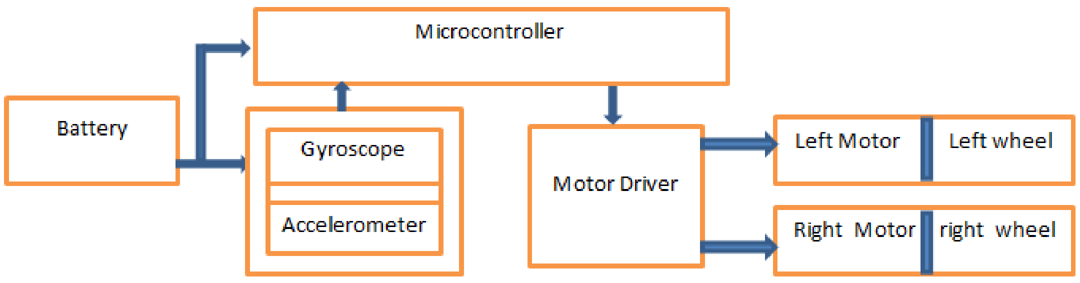


Figure 1.1 Structure of inverted pendulum robot

Chapter 2

Inverted Pendulum Robot Modeling

The system in this project consists of an inverted pendulum is mounted on two wheels. The inverted pendulum system is an example commonly found in control system literature. Its popularity derives in part from the fact that it is unstable without control, that is, the pendulum will simply fall over if the wheels is not moved to balance it. Additionally, the dynamics of the system are nonlinear. The objective of the control system is to balance the inverted pendulum by applying a force to the wheels that the pendulum is attached to. Fig 2.1 shows free body diagram of our inverted pendulum.

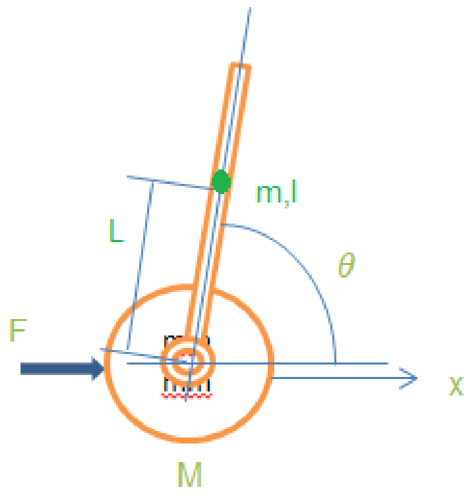


Figure 2.1 Free body diagram of inverted pendulum

2.1 Inverted Pendulum Dynamic

In this project we will consider a problem where the pendulum is constrained to move on the vertical plane as shown in the Fig 2.2. For this system, the control input is the force F that moves the wheels horizontally and the outputs are the angular position of the pendulum θ and the displacement in x-axis of the wheels x. Table 2.1

Table 2.1 Parameter and variable of inverted pendulum

M	mass of the wheels	0.07
m	mass of the pendulum	0.725
b	coefficient of friction for wheels	0.1
L	length to pendulum center of mass	0.15
I	mass moment of inertia of the pendulum	0.0318
F	force applied to the wheels	
x	wheels position coordinate	
θ	pendulum angle from vertical	

A free-body diagrams of the two elements of the inverted pendulum robot is shown in Fig **??**.

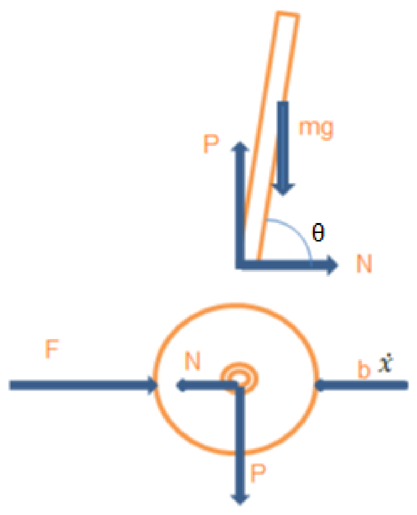


Figure 2.2 Free-body diagrams of IPR

Summing the forces that act on pendulum in wheels horizontal axis, we get the following equation of motion.

$$F = M\ddot{x} + b\dot{x} + N \tag{2.1}$$

The reaction force can be describe as:

$$N = mXg \tag{2.2}$$

The equation for Xg can be describe as

$$Xg = x + L\sin\theta \tag{2.3}$$

$$\dot{X}g = \dot{x} + L\cos\theta\dot{\theta} \tag{2.4}$$

$$\ddot{X}g = x + L\cos\theta\ddot{x} + L\sin\theta\dot{\theta}^2 \tag{2.5}$$

substitute Equation 2.5 in Equation 2.2, we get the following expression for the reaction force N.

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \tag{2.6}$$

If we substitute this Equation into the Equation (2.1), we get one of the two governing equations for this system.

$$F = (m + M)\ddot{x} + b\dot{x} - mL\sin\theta\dot{\theta}^2 + Lm\cos\theta\ddot{\theta} \tag{2.7}$$

To get the second equation of motion for this system, sum the forces vertical to the pendulum. We get the following equation.

$$P\sin\theta + N\cos\theta - mg\sin\theta = mL\ddot{\theta} + m\ddot{x}\cos\theta \tag{2.8}$$

To get rid of the P and N terms in the Equation 2.8 , sum the moments about the centroid of the pendulum to get the following equation.

$$PL\sin\theta - NL\cos\theta = I\ddot{\theta} \tag{2.9}$$

Combining Equation (2.8) and Equation (2.9), you get the second governing equation.

$$(I + mL^2)\ddot{\theta} + mLg\sin\theta = -m\dot{x}L\cos\theta \tag{2.10}$$

Since the analysis and control design techniques we will be employing in this system only to linear systems. This set of equations needs to be linearized. Specifically, we will linearize the equations about the vertically upward equillibrium position, $\theta = \pi$, and will assume that the system stays within a small neighborhood of this equillbrium. This assumption should be reasonably valid since under control. Let ϕ represents the deviation of the pendulum is position from equilibrium, that is, $\theta = \pi + \varphi$. Again presuming a small deviation (φ) from equilibrium, we can use the following small angle approximations of the nonlinear functions in our system equations:

$$\sin\theta = \sin(\pi + \varphi) = -\varphi \tag{2.11}$$

$$\cos\theta = \cos(\pi + \varphi) = -1 \tag{2.12}$$

$$\dot{\theta}^2 \approx \dot{\varphi}^2 \approx 0 \tag{2.13}$$

After substituting the assumptions (2.7)-(2.9) into our nonlinear governing equations, we yield two linearized equations of motion. Where u represent the input force F.

$$(I + mL^2)\ddot{\varphi} + mLg\varphi = -mL\ddot{x} \tag{2.14}$$

$$u = (m + M)\ddot{x} + b\dot{x} + mL\ddot{\varphi} \tag{2.15}$$

2.1.1 Transfer Function

To obtain the transfer function of the linearized system equations can be obtained by first taking the Laplace transform of the system equations with assuming zero initial conditions. The resulting Laplace transforms are then:

$$(I + ml^2)\Phi(s)s^2 - mgl\mathit{\Phi}(s) = mlX(s)s^2 \tag{2.16}$$

$$(M + m)X(s)s^2 + bX(s)s - ml\mathit{\Phi}(s)s^2 = U(s) \tag{2.17}$$

Recall that a transfer function represents the relationship between a single input and a single output at a time. To find our first transfer function for the output $\mathit{\Phi}(s)$ and an input of $U(s)$ we need to eliminate $X(s)$ from the Equation 2.16 and Equation 2.17 where $U(s)$, $\mathit{\Phi}(s)$ and $X(s)$ are system input,system output and system states respectivlly. Solve the first equation for $X(s)$.

$$X(s) = \left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\mathit{\Phi}(s) \tag{2.18}$$

Then substitute Equation (2.18) in Equation (2.17), we obtain on this equation:

$$(M+m)\left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\mathit{\Phi}(s)s^2 + b\left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\mathit{\Phi}(s)s - ml\mathit{\Phi}(s)s^2 = U(s) \tag{2.19}$$

Rearranging, the transfer function is then the following

$$\frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s^2}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \tag{2.20}$$

Where

$$q = [(M+m)(I+ml^2) - (ml)^2] \tag{2.21}$$

From the transfer function it can be seen that there is both a pole and a zero at the origin.Poles and Zeros of a transfer function are the frequencies for which the value of the denominator and numerator of transfer function becomes zero respectively. The values of the poles and the zeros of a system determine whether the system is stable. Her Pole and Zero can be canceled and the transfer function becomes the following.

$$P_{pend}(s) = \frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}} \qquad [\frac{rad}{N}] \tag{2.22}$$

Second, the transfer function with the wheels position $X(s)$ as the output can be derived in a similar manner to arrive at the following.

$$P_{wheels}(s) = \frac{X(s)}{U(s)} = \frac{\frac{(I+ml^2)s^2 - gml}{q}}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \qquad [\frac{m}{N}] \tag{2.23}$$

2.1.2 State-Space of IPR

state-space representation is a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations or difference equations. State variables are variables whose values evolve through time in a way that depends on the values they have at any given time and also depends on the externally imposed values of input variables. Output variablesâĂŹ values depend on the values of the state variables.

The linearized equations of motion can also be represented in state-space form if they are rearranged into a series of first order differential equations. Since the equations are linear, they can then be put into the standard matrix form shown in Equation (2.24).

$$\begin{pmatrix} \dot{x} \\ \ddot{x} \\ \dot{\varphi} \\ \ddot{\varphi} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+mL^2)b}{I(M+m)+MmL^2} & \frac{m^2gL^2}{I(M+m)+MmL^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{mIb}{I(M+m)+MmL^2} & \frac{mgL(M+m)}{I(M+m)+MmL^2} & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ \varphi \\ \dot{\varphi} \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{I+mL^2}{I(M+m)+MmL^2} \\ 0 \\ \frac{mL}{I(M+m)+MmL^2} \end{pmatrix} u \tag{2.24}$$

her we can define the output vector as $y = [x \;\; \varphi]^T$ then, the output equation is

$$y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ \varphi \\ \dot{\varphi} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} u \tag{2.25}$$

The C matrix has two rows because both the wheels position and the pendulum is position are part of the output. Specifically, the wheels position is the first element of the system output (y) and the pendulum is deviation from it is equilibrium position is the second element of (y).

Now submit the parameter of the system, we will obtain on Equation **??**

$$\begin{pmatrix} \dot{x} \\ \ddot{x} \\ \dot{\varphi} \\ \ddot{\varphi} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.569 & 34.6 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -2.7 & 210.7 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ \varphi \\ \dot{\varphi} \end{pmatrix} + \begin{pmatrix} 0 \\ 5.6980 \\ 27.05 \end{pmatrix} u \tag{2.26}$$

$$y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ \varphi \\ \dot{\varphi} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} u \tag{2.27}$$

Chapter 3

Mechanical Prototyping of the IPR

The inverted pendulum robot (IPR) consists of electrical and mechanical parts which handel this robot to move and balance itself. In this chapter we will toke about mechanical component of IPR and explain the function of each part. In general, the Main frame and the wheels are considered as primary parts in IPR.

3.1 Main frame

A main frame is one of primary parts of IPR, it consist from three shelves and eight rods. One of the main jobs of the main frame is to carry all of the electrical components of (IPR). The upper shelf carries a load, the middle shelf carries the microcontroller, battery, and sensors and the lower shelf carries a motor driver (L298N). The lower shelf is connected with the motor surface. The component is organized in this way in order to make the moving more ease. We test the robot by placing the battery in different height. If the battery is placed on the top then a high torque and high speed is needed to control back and forward leaning [6]. The torque is increasing with respect to battery goes to higher. On the other hand, if the battery is placed near to the DC gear motor, as a result robot gets very stable even if an external forced is applied.

3.1.1 Shelves

The main frame consists from three shelves. Each shelf is made from plastic with $(148mm \times 105mm \times 3mm)$. This dimension allows the robot to move in a small space. Lower and middle shelves are milled around midpoint and all shelves have four hole at edges as shows in Fig 3.1. The milled around midpoint is used to connect each electrical component together using smaller wire allowable.

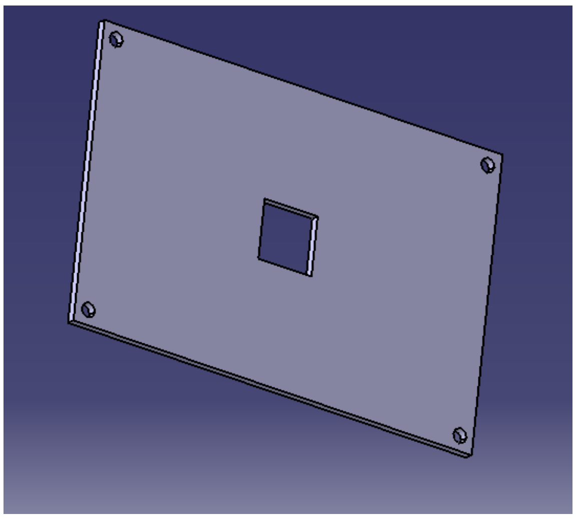


Figure 3.1 Middle and lower shelves in IPR

The milled around midpoint is used to connect each electrical component together by using smaller wire allowable.The upper shelf do not have milled around midpoint because it is carry the load and there are not exist any electrical parts in upper shelf to connect it by wired. Fig3.2 explain the upper shelf in IPR.

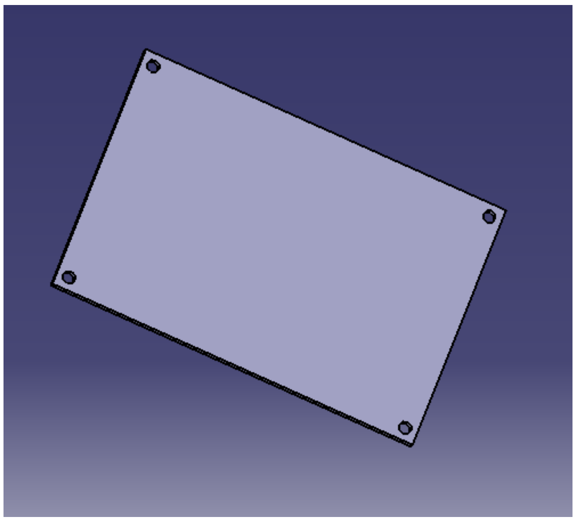


Figure 3.2 Upper shelf in IPR.

3.1.2 Rods

As we mentioned in shelves subsection, all shelves have four hole at edge, this holes used to connect each shelves together by rods. This rods have an one hundred mm length and ten mm diameter. Fig3.3 shows the rod in IPR.

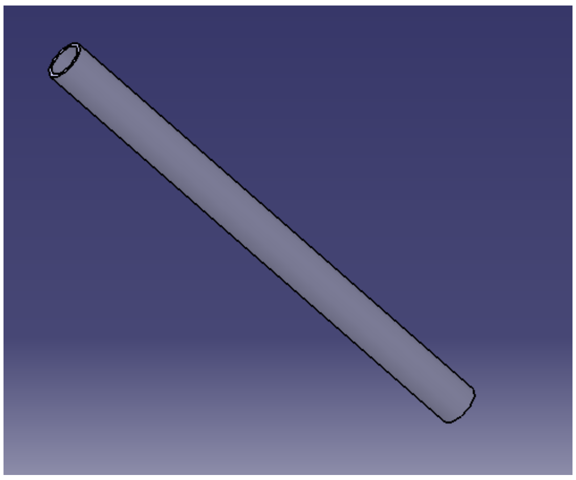


Figure 3.3 The rod in IPR by using catia program

3.2 Wheels

The main frame of IPR is based on two wheels. The wheels are placed parallel to each other. Each wheel has a radius of thirty-two cm and is connected with a dc gear motor. The wheels drive DC gear motor, then wheels move the robot back and forward balance it. Fig 3.4 shows the wheel used in build IPR.

Figure 3.4 The wheel used in build IPR

Chapter 4

Electrical system

In this chapter we present a complete content and interactions in the electrical system. First we present the electrical component of the robot which is mainly consist of a DC gear motor, Driver motor, sensors (MPU6050) and microcontroller.

4.1 The DC Gear Motor

The motor has the connection to the wheels and to the pendulum, In a robot the function of actuator is to induce motion or to stop it. In this system a DC gear motor will do all those activities. The DC gear motor will receive a voltage then induce torque that will move a wheel [2]. The DC gear motor advantages are lower cost, ease of control, deliver high starting torque and high efficiency. In the other hand, it need maintenance continuously and very expensive. In this project we use two DC gear motors connect with two wheels to move the robot back and forward.

4.2 The Driver

The motor driver is a device that acts as intermediary between the robot microcontroller, batteries and motors. The robot microcontroller has a low output voltage signal, and the actuators a high voltage signal to operate, so the driver used to amplify the signal, control the direction of actuators and it protect microcontroller from reverse current comes from motors. The driver should be able to provide the required power output as well as compatible to interface with the system controller. A L298N DC motor driver shown in Fig4.1 is used. The L298N is a dual H-Bridge motor driver which allows speed and direction control of two DC motors at the same time. The module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A.

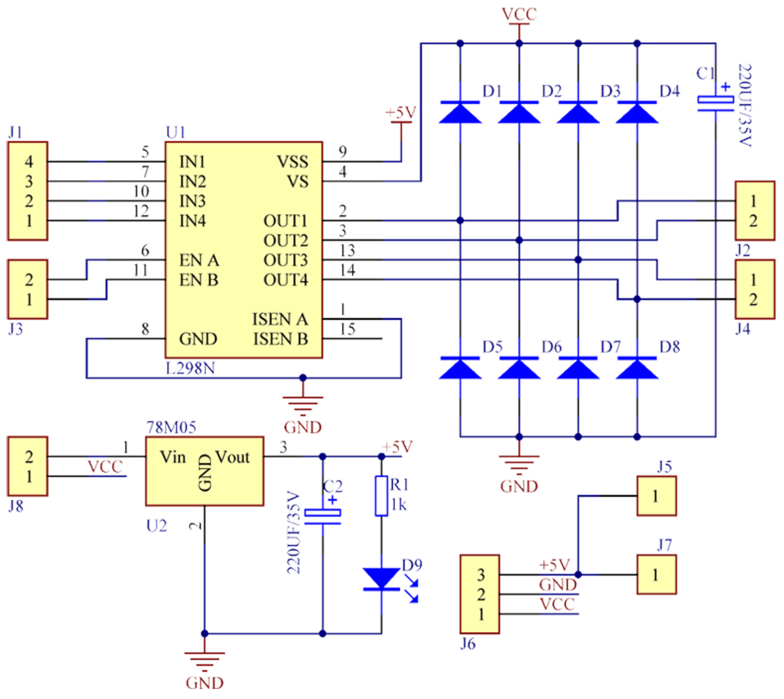


Figure 4.1 Driver Motor L298N

4.3 The sensors

When an inverted pendulum moves, it may tends in a specific angle it is called tilt angle. To measure the tilt angle we need a data from two sensors, accelerometer and gyroscope. Both have advantages and disadvantages [5]. The accelerometer measures the force with respect to gravity. Thus we calculate angle of the vehicle, the problem with this type of sensor is that it accounts for the rest of the forces the vehicle is experiencing it has a lot of error and noise. The gyroscope can measure the angular velocity, but the problem is measurement is that is not perfect and the integration has a deviation, such that in short run times the measure is valid, but for long time runs the value will start to deviate much form the real angle, bad repeatability. To avoid this situation we integrate these measurements and can accurately obtain the angle of the vehicle at its current position in any instantaneous time. To combine of both sensors, which is called sensor fusion, We use Kalman Filter in this application, as it is an algorithm very extended in the field of robotics, and offers very good results at very low computational cost. Also it is compatible with a lot of modern Kalman filter is used when we have uncertain information on a dynamic system and make a calculated estimate of the system is next position [9]. In Fig 4.3 show the gyroscope Where ws is the constant rate of spin of the wheel, in radians/second, wp is the constant rate of precession, in radians/second, L is the length of the rod, r is the radius of the wheel, Θ is the angle between the vertical and the rod and g is a gravity.

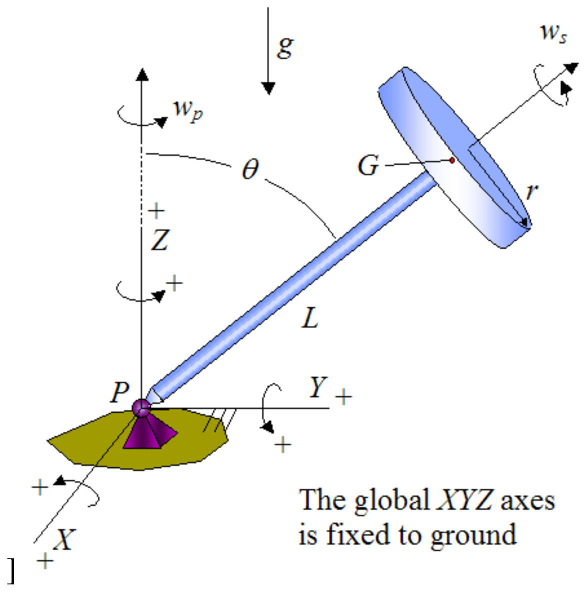


Figure 4.2 Gyroscope

4.4 The Microcontroller

The microprocessor normally is used to the process system with long words and big send of addresses[3]. However , in this project (inverted pendulum robot), one can use microcontrollers instead since they reliable and malfunctioning of PIC percentage is very less performance of the microcontrollers are very fast and It is easy to use. Simply stated, a microcontroller is a single integrated circuit that at least contains the necessary elements of a complete computer system: CPU, memory, a clock oscillator, and input and output peripherals[3]. Microcontrollers normally contains additional peripheral modules, such as serial and timer units. Famous types of the microcontroller are peripheral interface controller (PIC)and Advanced Virtual Risc (AVR)[3]. In this project we use Arduino microcontroller to achieve many control tasks . Arduino is a small development board created for experimenting and hobby purposes. It has twenty-eight I/O pins which includes, 6 analog inputs, It requires an operating voltage between 7 - 12V for optimal functioning and can be powered and be programmed through a USB-cable It also comes with a integrated development environment (IDE) This makes the board good for getting started quickly and be able to reprogram the board fast, however it comes with the cost of limiting programming possibilities The IDE uses both Arduinos adaptation of the processing language and/or C++ for writing code. Processing was a simplified version of C++ with that had some of the math functions and some classes included Some drawback with Arduinos IDE were that it lacks line-numbering, it did not give full error messages and it was not possible to get a fast overview of a functions syntax. These drawbacks were not known when considering to use Arduino. In this project, the arduino will receive the accelerometer and gyroscope signal then send the desired signal to the dc motor drive. Fig**??** show the Arduino uno schematic

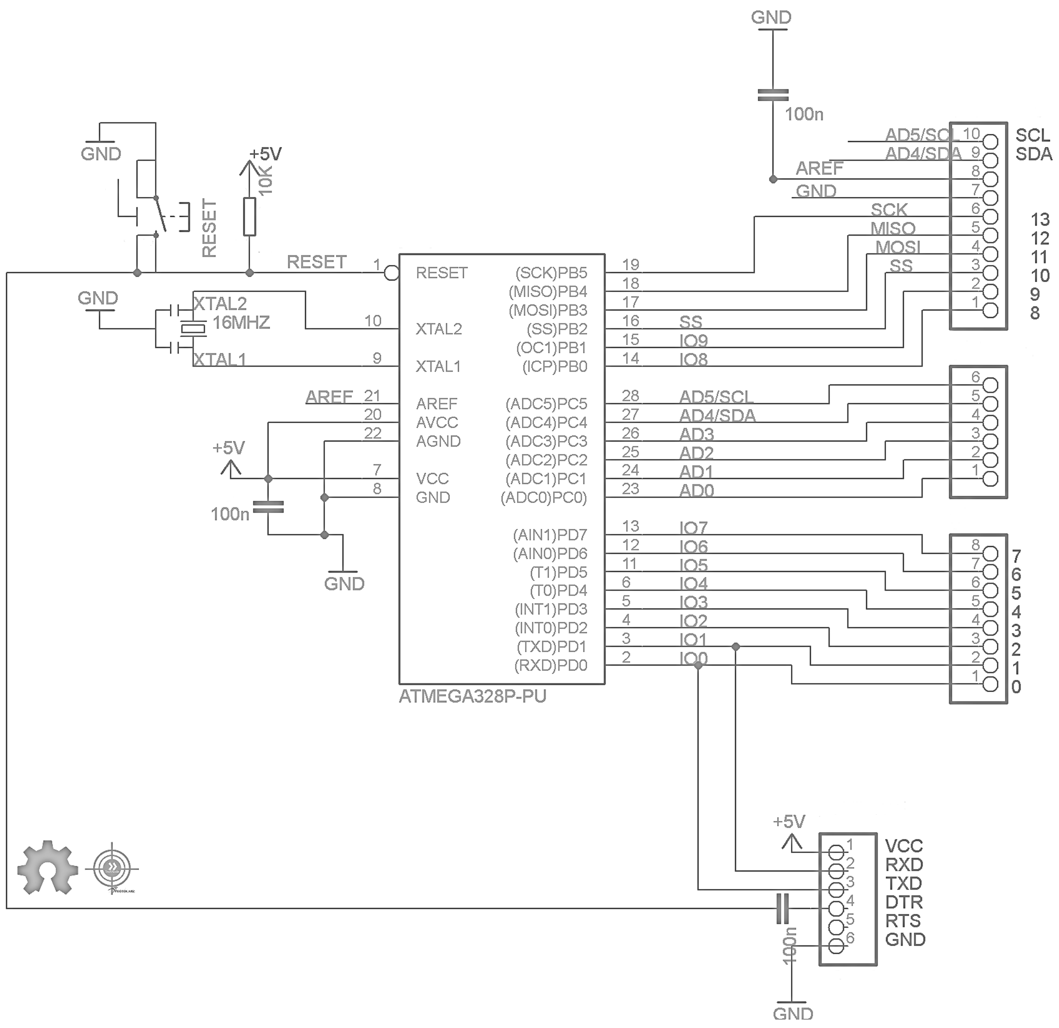


Figure 4.3 The Arduino uno schematic

Chapter 5

Control system

This chapter shows some control methods for an inverted pendulum robot such as proportional integral derivative (PID) controller and optimal control using linear quadratic regulator (LQR). The goals of using these controller is to stabilize the system and to make minimization (reduce the energy consume of the system).

5.1 Linear Optimal Control)

Optimal control refers to a class of methods that can be used to synthesize a control policy which results in the best possible behavior with respect to the prescribed criterion (i.e., control policy which leads to maximization of performance)[4]. The main objective of optimal control is to determine control signals that will cause a process to satisfy some physical constraints and at the same time extremize (maximize or minimize) a chosen performance criterion[4]. Linear quadratic regulator (LQR) is one of the optimal control methods, which takes into account the states of the linear dynamical system and control input to make the optimal control decisions. This is simple as well as robust. After linearization of nonlinear system equations about the upright equilibrium position having initial conditions as $x_0 = [0\ 0\ 0\ 0]^T$, the linear state-space equation is obtained as:

$$\dot{x} = Ax + Bu \tag{5.1}$$

Where x represents the states of the system, u the control input and A, B are two constant matrices which is obtained from the system dynamics. The state feedback control $u = -Kx$ leads to

$$\dot{x} = (A - BK)x \tag{5.2}$$

where K is derived from minimization of the cost function

$$J = \int_0^\infty (x^T Qx + u^T Ru)dt \tag{5.3}$$

where Q and R are positive semi-definite and positive definite symmetric constant matrices, respectively[2]. The LQR gain vector K is given by

$$K = R^{-1}B^T P \tag{5.4}$$

where P is a positive definite symmetric constant matrix obtained from the solution of matrix algebraic Riccati Equation (ARE)

$$AP + PA + Q - PBR^{-1}B^T P = 0 \tag{5.5}$$

In the optimal control of nonlinear inverted pendulum dynamical system using PID and LQR approaches, all the instantaneous states of the nonlinear system, pendulum angle θ, angular velocity θ, wheels position x, and wheels velocity $\dot{x}$ are considered available for measurement, which are directly fed to the controller. The LQR is designed using the linear state-space model of the system. The optimal control value of LQR is added negatively to the PID control value to have a resultant optimal control

5.2 PID Controller

Before explaining PID Controller, let is revise about Control System. There are two types of systems; open loop system and close loop system. An open loop system is also known as an uncontrolled system and close loop system is known as a controlled system. In open loop system, the output is not controlled because this system has no feedback and in a close loop system, the output is controlled with the help of controller and this system requires one or more feedback paths. An open loop system is very simple but not useful in industrial control applications because this system is uncontrolled. Close loop system is complex but most useful for industrial application, because in this system output can be stable at a desired value, PID is an example of Closed Loop System. Block diagram of this systems is as shown in Fig5.1

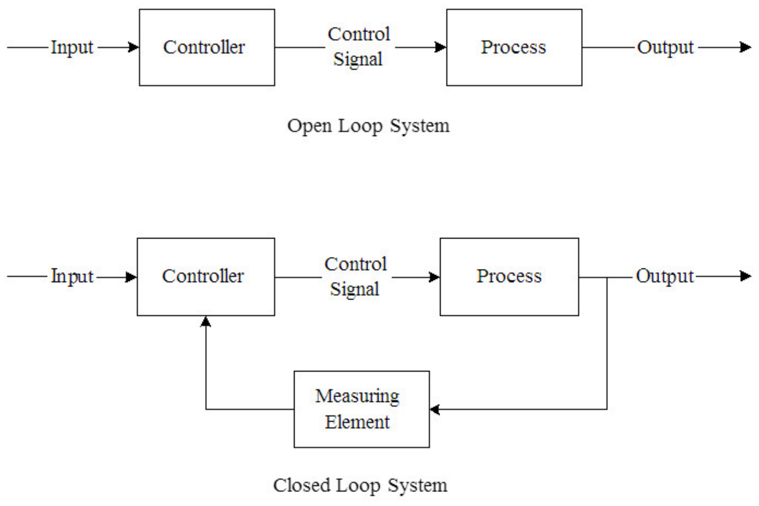


Figure 5.1 Open and Closed loop system for controllers

A close loop system is also known as feedback control system and this type of system is used to design automatically stable system at desired output or reference. For this reason, it generates an error signal. Error signal e(t) is a difference between the output y(t) and the reference signal u(t). When this error is zero that means desired output is achieved and in this condition output is same as a reference signal. PID controller is a Close loop system which has feedback control system and it compares the Process variable (feedback variable) with set Point and generates an error signal and according to that it adjusts the output of system. This process continues until this error gets to Zero or process variable value becomes equal to set point. Proportional, Integral and Derivative controller is a combination of P, I and D controller. Output of controller is summation of proportional, integral and derivative responses. Mathematical equation of PID controller is as shown below

$$u = K_p e(t) + K_i \int e(t)dt + K_d \frac{de}{dt} \tag{5.6}$$

5.2.1 Proportional (P)

Term P is proportional to the actual value of the error. If the error is large, control output is also large and if the error is small control output is also small.

Also taking in to account. Speed of response is also directly proportional to proportional gain factor (Kp). So, the speed of response is increased by increasing the value of Kp but if Kp is increased beyond normal range, process variable starts oscillating at high rate and make system unstable.

5.2.2 Integral (I)

Integral controller is generally used to decrease the steady state error. Term I is integrate (with respect to time) to the actual value of the error. Because of integration, very small value of error, results very high integral response. Integral controller action continues to change until error becomes zero. Integral gain is inversely proportional to the speed of response, increasing ki, decrease the speed of response. Proportional and Integral controllers are used combined (PI controller) for good speed of response and steady state response.

5.2.3 Derivative (D)

Derivative controller is used to with combination of PD or PID. It never used alone, because if error is constant (non-zero), output of the controller will be zero. In this situation, controller behave life zero error, but in actual there are some error (constant). Output of derivative controller is directly proportional to the rate of change of error with respect to time. Generally, Derivative controller is used when processor variables starts oscillating or changes at a very high rate of speed. D-controller is also used to anticipates the future behaviour of the error by error curve.

Stability in this project is to keep the inverted pendulum robot maintain in vertical position, to stabilize the inverted pendulum in the upright position and to control the wheels at the desired position(angle $\theta = 0$) we will use the PID control approach. PID used to compare between desired value (must be 0) and actual value (accelerometer and gyroscope signal) then correct the error (to be $\approx$ 0).

The expression of the PID control of IPR is given as:

$$u = K_p e_x(t) + K_i \int e_x(t)dt + K_d \frac{de_x}{dt} \tag{5.7}$$

5.3 Digital Controller Design

In this section, we will discuss converting continuous-time models into discrete-time models. Our first step in designing a digital controller is to convert the above continuous state-space equations to a discrete form. We will accomplish this employing the MATLAB function c2d. This function requires that we specify three arguments: a continuous system model, the sampling time (Ts in sec/sample), and the 'method'. In choosing a sample time, note that it is desired that the sampling frequency be fast compared to the dynamics of the system. One measure of a system's "speed" is its closed-loop bandwidth. A good rule of thumb is that the sampling frequency be at least 30 times larger than the closed-loop bandwidth frequency which can be determined from the closed-loop Bode plot. we used matlab program to convert continuous-time models into discrete-time models. discrete state-space model of the system is

$$\begin{bmatrix} x(k+1) \\ \dot{x}(k+1) \\ \phi(k+1) \\ \dot{\phi}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0.00997 & 0.00173 & 0 \\ 0 & 0.99843 & 0.03462 & 0.00173 \\ 0 & -0.0001352 & 1.011 & 0.01 \\ 0 & -0.02707 & 2.11 & 1.011 \end{bmatrix} \begin{bmatrix} x(k) \\ \dot{x}(k) \\ \phi(k) \\ \dot{\phi}(k) \end{bmatrix} + \begin{bmatrix} 0.0002848 \\ 0.0569 \\ 0.001352 \\ 0.2707 \end{bmatrix} u(k)$$

$$\mathbf{y}(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \dot{x}(k) \\ \phi(k) \\ \dot{\phi}(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u(k) \tag{5.8}$$

5.4 Simulation

The Simulink models for control of nonlinear inverted pendulum system begin by design LQR controller as show in Fig5.9. By using MATLAB we can find K matrix

$$K = \begin{bmatrix} -1 & -1.52 & 17.62 & 1.434 \end{bmatrix} \tag{5.9}$$

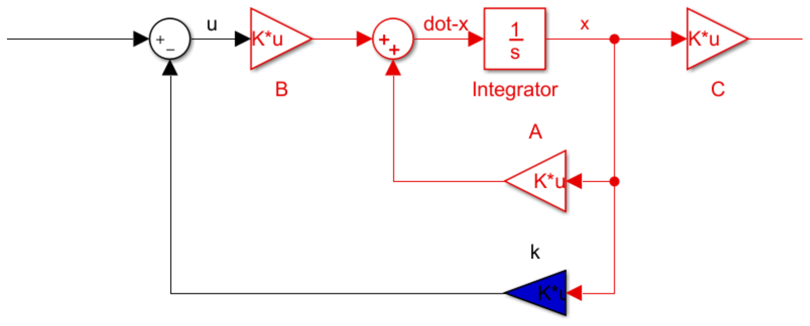


Figure 5.2 LQR control of nonlinear inverted pendulum system

The result of using LQR controller as show in Fig5.3.When input is one meter, angle goes to zero after six second, this result without using PID controller. The goal of using LQR is to achieve minimum energy with minimum cost.

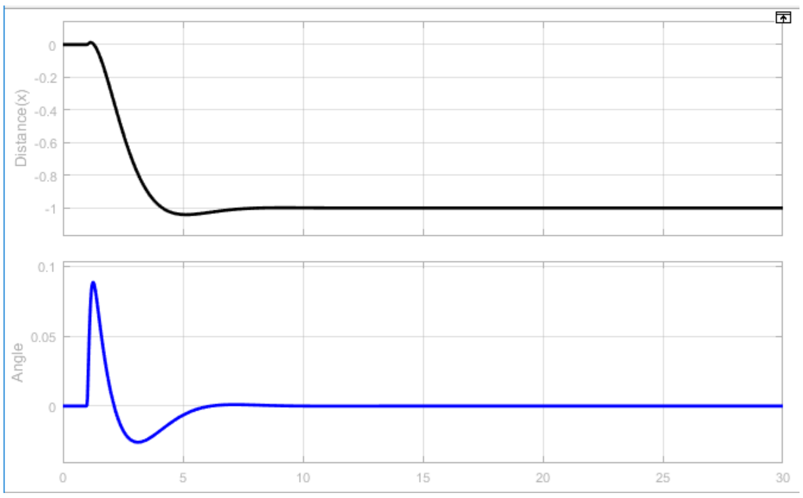
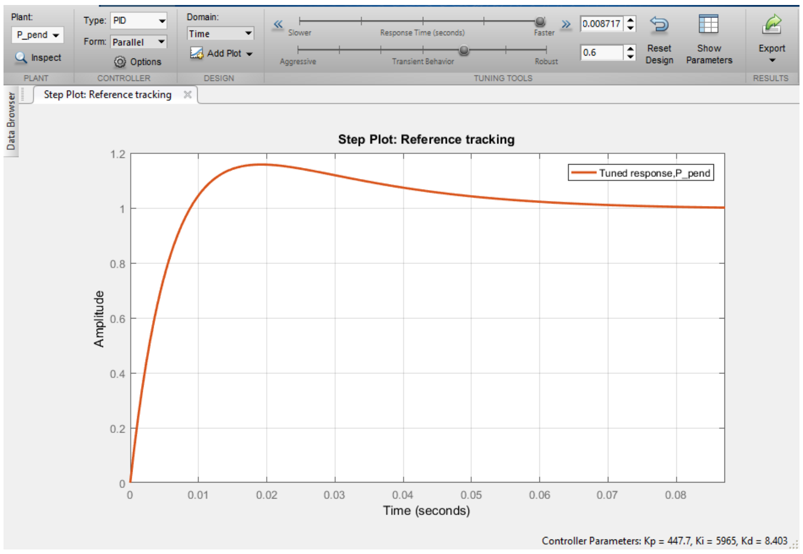


Figure 5.3 Response of IPR when using LQR controller

Now we will add PID controller to the system but we need to find the value of kp,ki and kd so we used PID tuner on MATLAB program as show in fig5.4


Plant: P_pend

Type: PID

Form: Parallel

Domain: Time

Response Time (seconds): 0.008717

Transient Behavior: 0.6

Controller Parameters: Kp = 447.7, Ki = 5965, Kd = 8.403

Figure 5.4 PID tuner on MATLAB program

We controlled in robust and speed to optimize system behavior then take the PID terms value. we used simulation to build PID controller as shown in Fig5.6

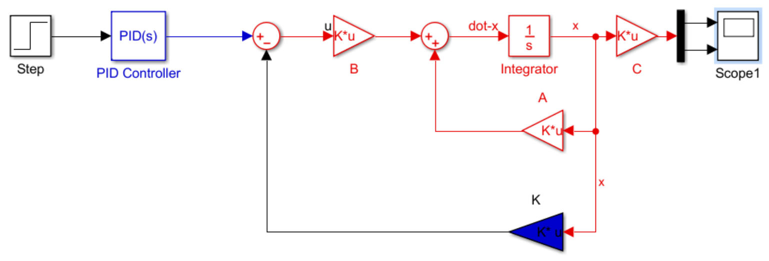


Figure 5.5 IPR simulation with LQR and PID controller

The result of IPR when we used PID controller and LQR given response more fast than the response when we used LQR controller alone. Fig5.6

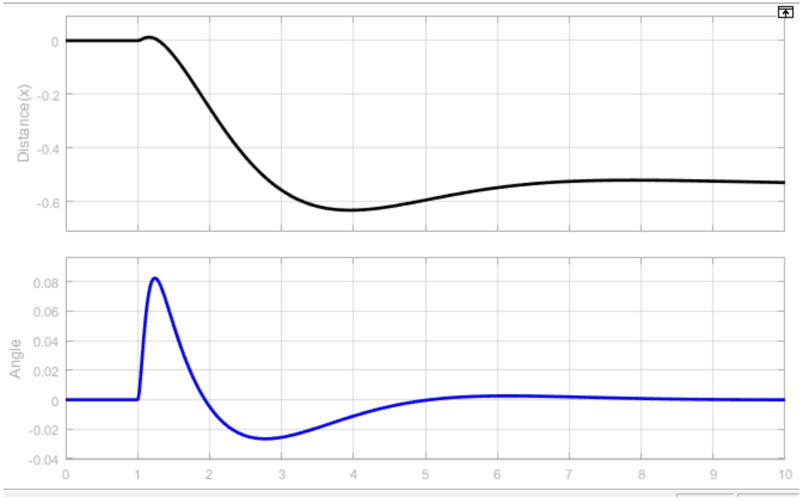


Figure 5.6 Result of IPR simulation when PID and LQR are used

Chapter 6

Conclusions

In this project, we use accelerometer and gyroscope sensors to measure a tilt angle for inverted pendulum robot, then use Kalmen filter to defuse the two sensors reading to obtain on accurate reading. The model of the inverted pendulum shows that the system is unstable, so we use linear optimal control to give a best possible behavior and to convert the system from instability to stability case. PID controller which remain on a vertical position for inverted pendulum robot and it is a challenging in this project. We used matlab to make simulation for this system, to get the expected motion profiels of inverted pendulum robot.

Chapter 7

Appendix

7.1 Code

$include < PID_v1.h >$

$include < LMotorController.h >$

$include"I2Cdev.h"$

$include"MPU6050_6Axis_MotionApps20.h"$

$include"Wire.h"$

$endif$

$defineMIN_ABS_SPEED30$

$MPU6050mpu;$

$//MPUcontrol/statusvars$

$booldmpReady = false; //settrueifDMPinitwassuccessful$

$uint8_tmpuIntStatus; //holdsactualinterruptstatusbytefromMPU$

$uint8_tdevStatus; //returnstatusaftereachdeviceoperation(0 = success, !0 = error)$

$uint16_tpacketSize; //expectedDMPpacketsize(defaultis42bytes)$

$uint16_tfifoCount; //countofallbytescurrentlyinFIFO$

$uint8_tfifoBuffer[64]; //FIFOstoragebuffer$

$//orientation/motionvars$

$Quaternionq; //[w, x, y, z]quaternioncontainer$

$VectorFloatgravity; //[x, y, z]gravityvector$

$floatypr[3]; //[yaw, pitch, roll]yaw/pitch/rollcontainerandgravityvector$

$//PID$

$doubleoriginalSetpoint = 172.50;$

$doublesetpoint = originalSetpoint;$

$doublemovingAngleOffset = 0.1;$

$doubleinput, output;$

$//adjustthesevaluestofityourowndesign$

$doubleKp = 60;$

$doubleKd = 2.2;$

$doubleKi = 270;$

PID pid(input, output, setpoint, Kp, Ki, Kd, DIRECT);

$doublemotorSpeedFactorLeft = 0.6;$

$doublemotorSpeedFactorRight = 0.5;$

$//MOTORCONTROLLER$

$intENA = 5;$

$intIN1 = 6;$

$intIN2 = 7;$

$intIN3 = 9;$

$intIN4 = 8;$

$intENB = 10;$

$LMotorControllermotorController(ENA, IN1, IN2, ENB, IN3, IN4, motorSpeedFactorLeft, m$

$volatileboolmpuInterrupt = false; //indicateswhetherMPUinterruptpinhasgonehigh$

$voiddmpDataReady()$

$mpuInterrupt = true;$

$voidsetup()$

$//joinI2Cbus(I2Cdevlibrarydoesn'tdothisautomatically)$

$ifI2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE$

$Wire.begin();$

$TWBR = 24; //400kHzI2Cclock(200kHzifCPUis8MHz)$

$elifI2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE$

$Fastwire :: setup(400, true);$

$endif$

$mpu.initialize();$

$devStatus = mpu.dmpInitialize();$

$//supplyyourowngyrooffsetshere, scaledforminsensitivity$

$mpu.setXGyroOffset(220);$

$mpu.setYGyroOffset(76);$

$mpu.setZGyroOffset(-85);$

$mpu.setZAccelOffset(1788); //1688factorydefaultformytestchip$

$//makesureitworked(returns0ifso)$

$if(devStatus == 0)$

$//turnontheDMP, nowthatit'sready$

$mpu.setDMPEnabled(true);$

$//enableArduinointerruptdetection$

$attachInterrupt(0, dmpDataReady, RISING);$

$mpuIntStatus = mpu.getIntStatus();$

$//setourDMPReadyflagsothemainloop()functionknowsit'sokaytouseit$

$dmpReady = true;$

$//getexpectedDMPpacketsizeforlatercomparison$

$packetSize = mpu.dmpGetFIFOPacketSize();$

$//setupPID$

$pid.SetMode(AUTOMATIC);$

$pid.SetSampleTime(10);$

$pid.SetOutputLimits(-255, 255);$

else

$//ERROR!$

$//1 = initialmemoryloadfailed$

$//2 = DMPconfigurationupdatesfailed$

$//(ifit'sgoingtobreak, usuallythecodewillbe1)$

$Serial.print(F("DMPInitializationfailed(code"));$

$Serial.print(devStatus);$

$Serial.println(F(")"));$

$voidloop()$

$//ifprogrammingfailed, don'ttrytodoanything$

$if(!dmpReady)return;$

$//waitforMPUinterruptorextrapacket(s)available$

$//nompudata - performingPIDcalculationsandoutputtomotors$

$pid.Compute();$

$motorController.move(output, MIN_ABS_SPEED);$

$//resetinterruptflagandgetINT_STATUSbyte$

$mpuInterrupt = false;$

$mpuIntStatus = mpu.getIntStatus();$

$//getcurrentFIFOcount$

$fifoCount = mpu.getFIFOCount();$

$//checkforoverflow(thisshouldneverhappenunlessourcodeistooinefficient)$

$//resetsowecancontinuecleanly$

$mpu.resetFIFO();$

$Serial.println(F("FIFOoverflow!"));$

$//otherwise, checkforDMPdatareadyinterrupt(thisshouldhappenfrequently)$ else
if (mpuIntStatus 0x02) $//waitforcorrectavailabledatalength, shouldbeaVERYshortwait$

$while(fifoCount < packetSize)fifoCount = mpu.getFIFOCount();$

$//readapacketfromFIFO$

mpu.getFIFOBytes(fifoBuffer, packetSize);

$//trackFIFOcounthereincasethereis > 1packetavailable$

$//(thisletsusimmediatelyreadmorewithoutwaitingforaninterrupt)$

$fifoCount- = packetSize$

$mpu.dmpGetQuaternion(q, fifoBuffer);$

$mpu.dmpGetGravity(gravity, q);$

$mpu.dmpGetYawPitchRoll(ypr, q, gravity)$

$input = ypr[1] * 180/M_PI + 180;$

Bibliography

[1] Md Iman Ali and Md Modasser Hossen. A two-wheeled self-balancing robot with dynamics model. In *Advances in Electrical Engineering (ICAEE), 2017 4th International Conference on*, pages 271–275. IEEE, 2017.

[2] NM Abdul Ghani, LK Haur, TP Yon, and F Naim. Dual mode navigation for two-wheeled robot. *World Academy of Science, Engineering and Technology*, 58:278–283, 2011.

[3] N Senthil Kumar, M Saravanan, and S Jeevananthan. *Microprocessors and Microcontrollers*. Oxford University Press, Inc., 2011.

[4] Damien Lamberton and Bernard Lapeyre. *Introduction to stochastic calculus applied to finance*. Chapman and Hall/CRC, 2011.

[5] L Niranjan, AR Suhas, and HS Chandrakumar. Design and implementation of self-balanced robot using proteus design tool and arduino-uno. *Indian J. Sci. Res*, 17(2):556–563, 2018.

[6] Lal Bahadur Prasad, Barjeev Tyagi, and Hari Om Gupta. Modelling and simulation for optimal control of nonlinear inverted pendulum dynamical system using pid controller and lqr. In *Modelling Symposium (Ams), 2012 Sixth Asia*, pages 138–143. IEEE, 2012.

[7] Lal Bahadur Prasad, Barjeev Tyagi, and Hari Om Gupta. Optimal control of nonlinear inverted pendulum system using pid controller and lqr: performance analysis without and with disturbance input. *International Journal of Automation and Computing*, 11(6):661–670, 2014.

[8] B Shilpa, V Indu, and SR Rajasree. Design of an underactuated self balancing robot using linear quadratic regulator and integral sliding mode controller. In *Circuit, Power*

and Computing Technologies (ICCPCT), 2017 International Conference on, pages 1–6. IEEE, 2017.

[9] AR Suhas and HL Viswanath. A comparative study between extended kalman filter and unscented kalman filter for traffic state estimation.