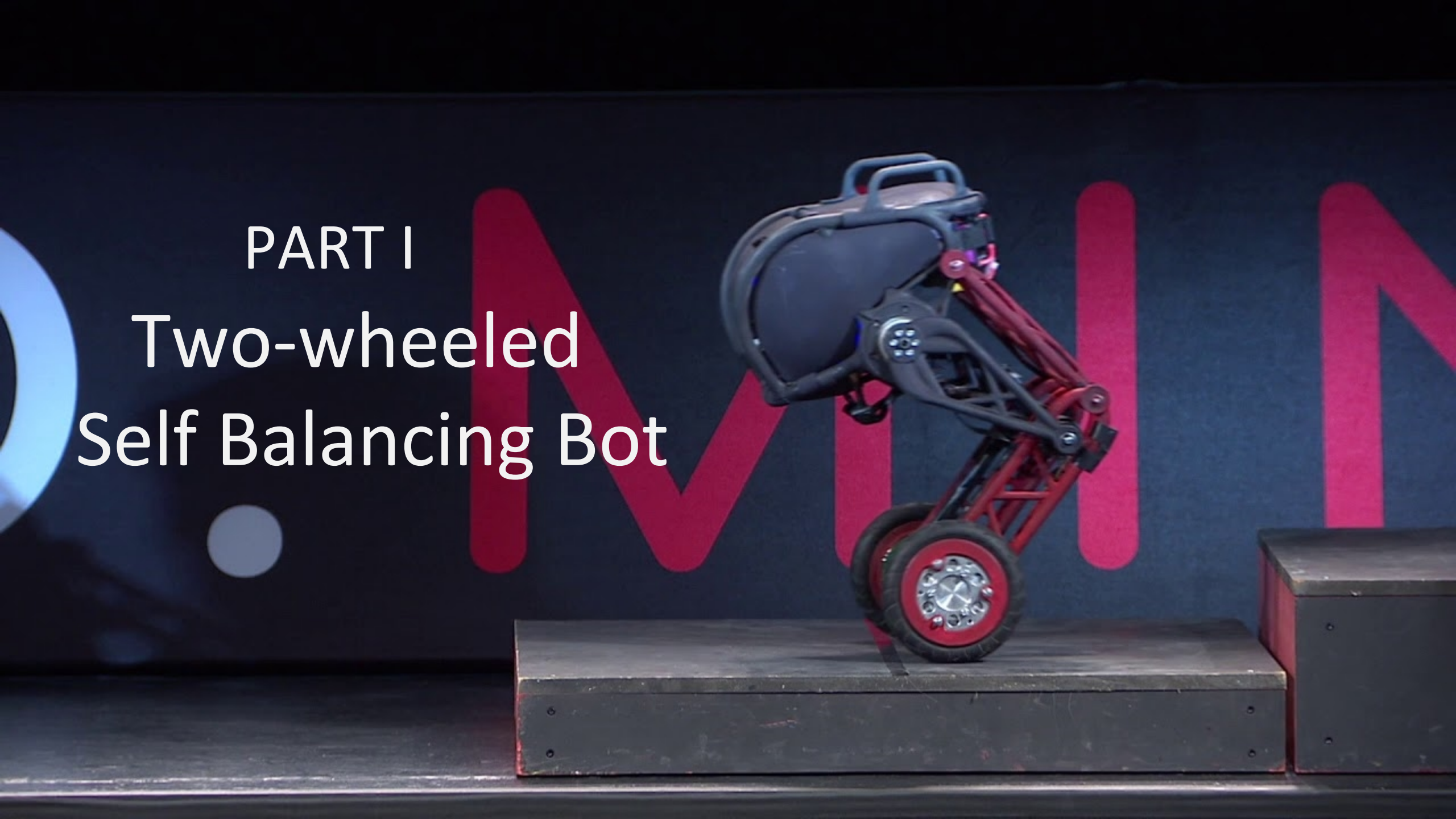
PART I

Two-wheeled Self Balancing Bot



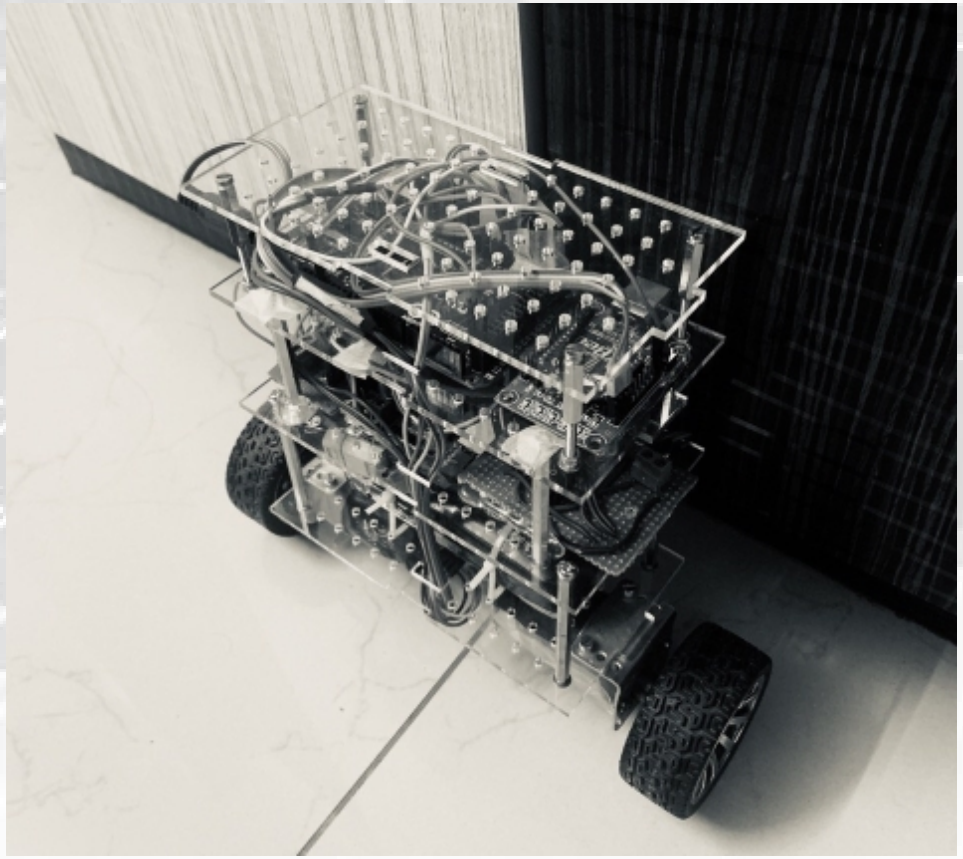
INDEX

1. Introduction
2. Theory
3. Hardware Design
4. Chassis Design
5. Signal Processing
6. Controller Design
    - 6.1 LQR Controller
    - 6.2 Cascaded P-D Controller
7. Software Design
8. Conclusion
9. Demo

INTRODUCTION

Balance bot is a two wheeled robot which can balance itself without any extra support. The balance bot can be considered as an advancement of the classic **inverted pendulum** problem.

In order to balance the bot, the magnitude and direction in which the bot is tilting is measured using accelerometer and gyroscope. The signals obtained are then used to move the wheels in the direction in which the bot is falling.
Complementary filter is used to obtain better results from the accelerometer and gyroscope.
Cascaded Proportional Derivative (PD) is used to control the motor speed.



Theory

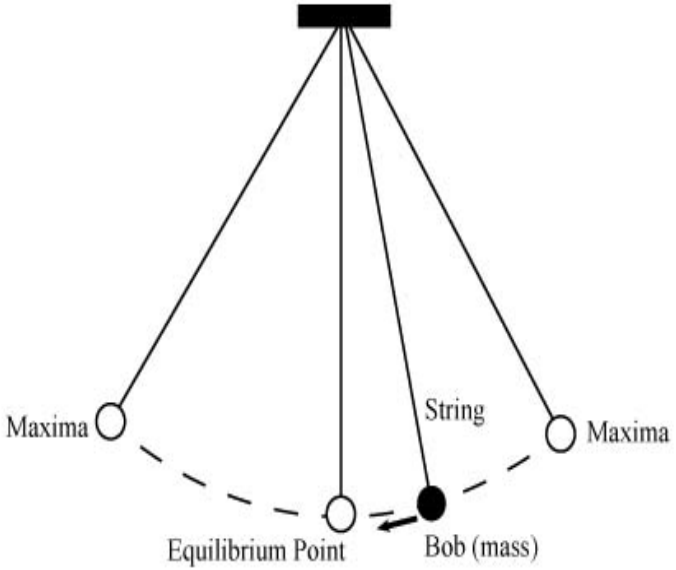
A **Simple pendulum** is one which can be considered to be a point mass suspended from a string or rod of negligible mass. When the Bob in the pendulum is left from the maxima point, the Bob oscillates to and forth and eventually comes to rest in the **Equilibrium position**.

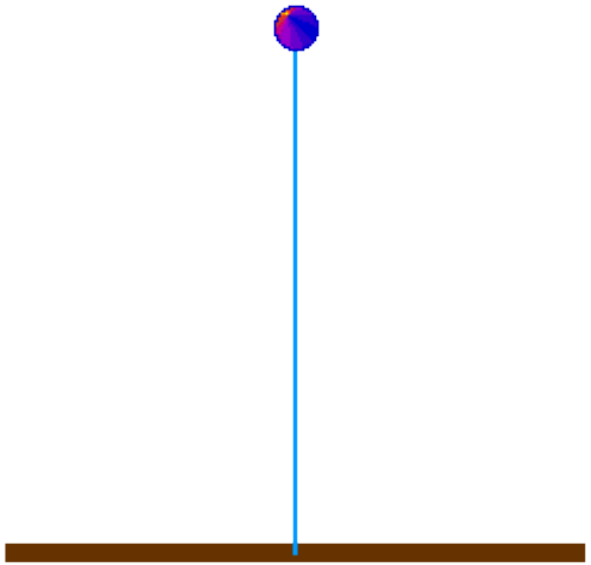
An **Inverted pendulum** is a pendulum that has its center of mass above its pivot point. It is **unstable** and without additional help will fall over. It can be suspended stably in this inverted position by using a **control system**.

Equilibrium Point of a system is the point at which the state of the system doesn't change.

Stable Equilibrium - If a system always returns to the equilibrium point after small perturbations.

Unstable Equilibrium - If a system moves away from equilibrium point after small perturbations



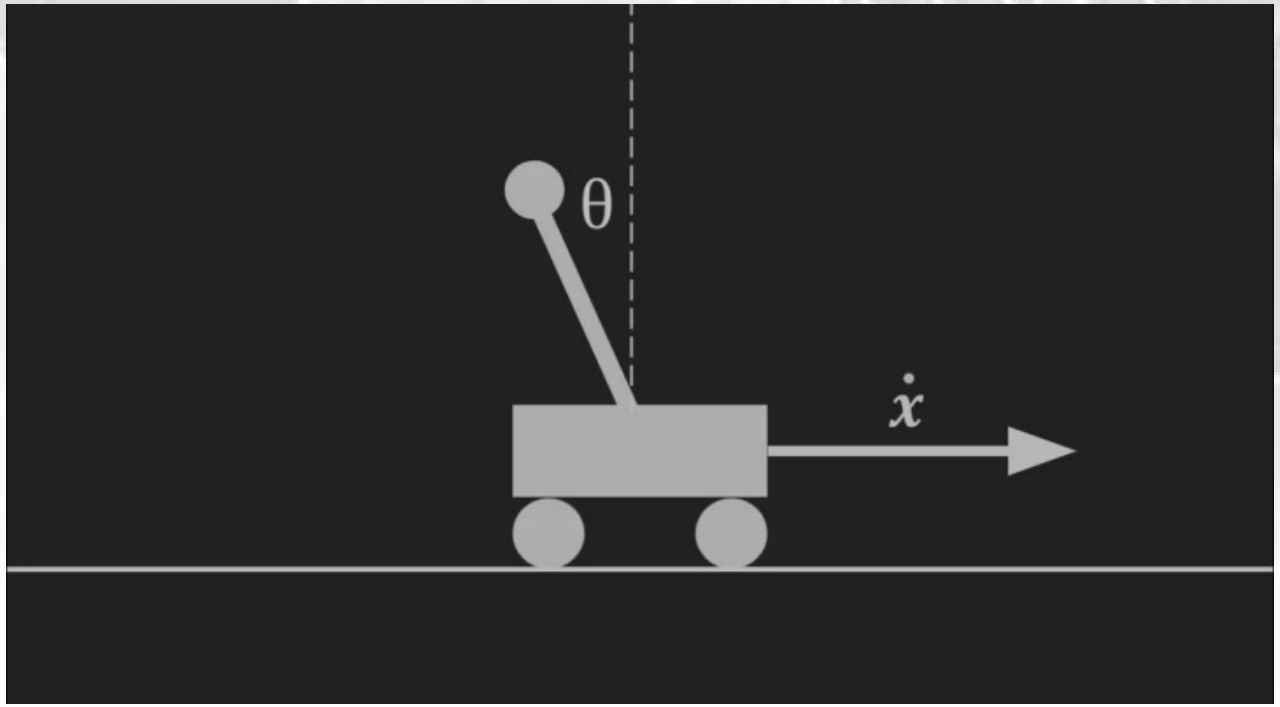


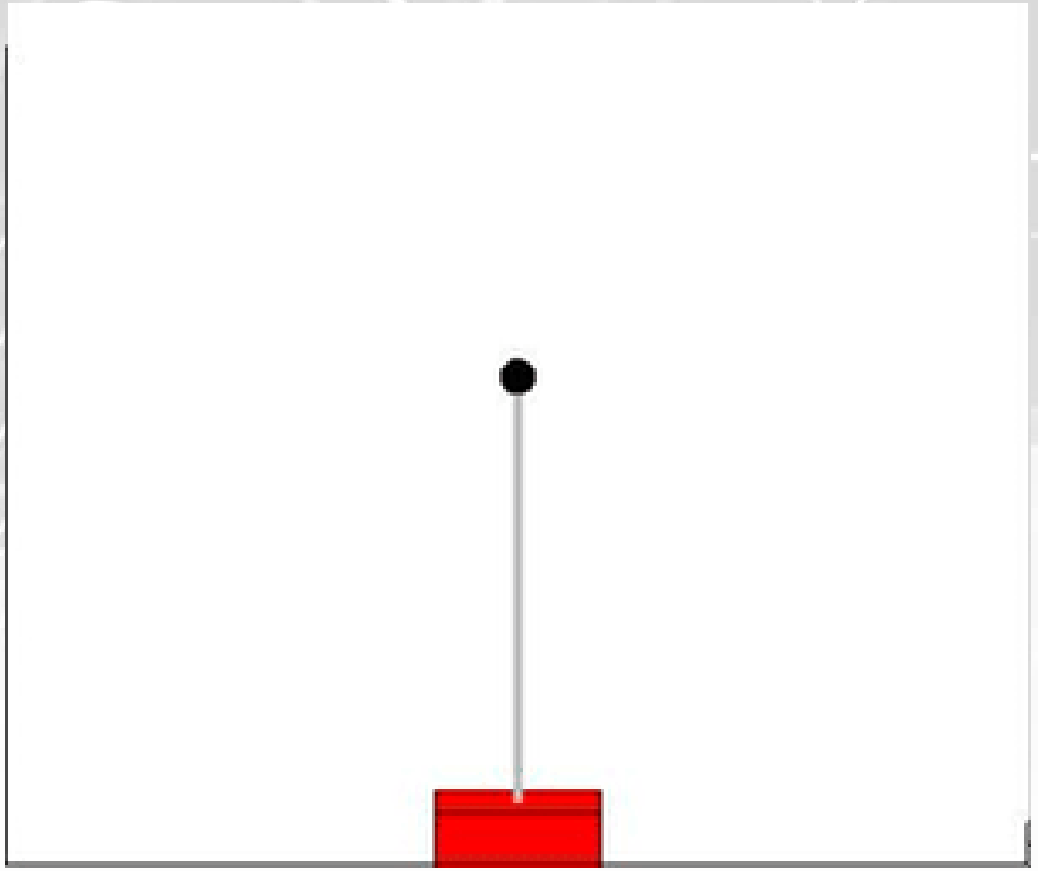
Theory

Cart Pendulum Model

The aim is to balance a pendulum upside down on a cart that is allowed to move about. A sensor is used to detect the position of the pendulum which sends the information over to a computer which performs certain computations and instructs actuators to move the cart in a way to make the pendulum vertical again.

The Goal of Cart Pendulum model is to keep $\vartheta = 0$ by moving the cart in the direction of the falling mass





Theory

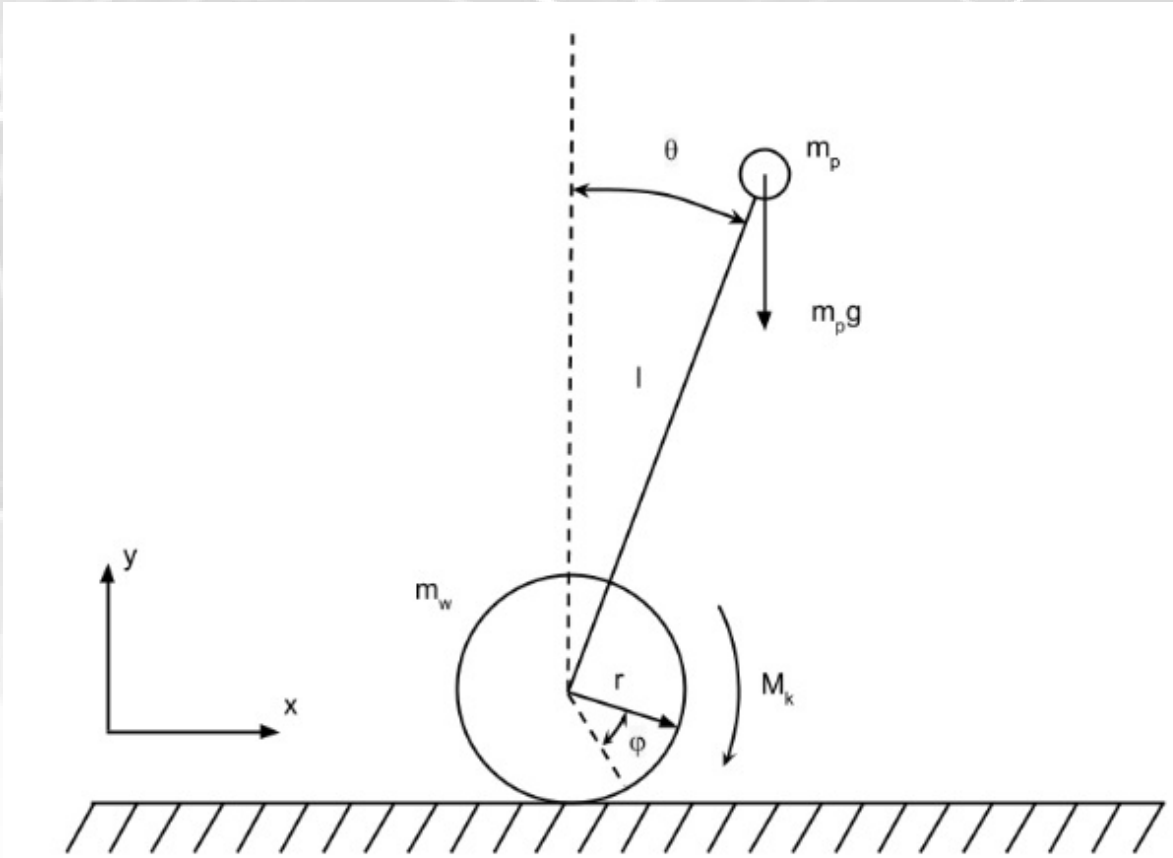
Mathematical Model

Consider the inverted pendulum system on the wheel, pictured below. Assume that the system moves without friction. The pendulum has a mass m_p, attached to a weightless rod of length l to the wheel. The wheel is considered as a ring of radius r and mass m_w. The engine torque M_k acts on the wheel

m_p : pendulum mass
m_w : wheel mass
l: pendulum length
r: wheel radius
θ: angle between the pendulum and the vertical line
ϕ: angle of rotation of the wheel relative to its initial position
M_k : motor torque

To study the system, we use the Euler-Lagrange equations. (1.1)

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}(q,\dot{q})\right) - \frac{\partial L}{\partial q}(q,\dot{q}) = \tau$$



Theory

Mathematical Model

Express the position of the center of the wheel through the angle of rotation:

$$x = r\phi \quad (1.2)$$

Note that the coordinates of the center of mass of the pendulum are found by the following relationships:

$$x_g = x + lsin(\theta) \quad (1.3)$$
$$y_g = lcos(\theta) \quad (1.4)$$

First, we calculate the representation for the kinetic energy of the system. Kinetic pendulum energy is equal to

$$T_p = \frac{1}{2}[m_p\dot{y}_g^2(t) + m_p\dot{x}_g^2(t)] \quad (1.5)$$

The kinetic energy of the wheel is

$$T_w = \frac{1}{2}[m_w\dot{x}^2(t) + m_w r^2\dot{\phi}^2(t)]$$

Substituting the above equations,
Total Kinetic Energy is

$$T = \frac{1}{2}m_p l^2\dot{\theta}sin(\theta)^2 + \frac{1}{2}m_p r^2\dot{\phi}^2 + \frac{1}{2}m_p l^2\dot{\theta}^2 cos(\theta)^2 + m_w r^2\dot{\phi}^2 + \frac{1}{2}m_p l^2\dot{\theta}^2$$

Potential Energy is

$$\Pi = m_p glcos(\theta)$$

Theory

Mathematical Model

The Lagrange function is given by the formula $L = T - \Pi$
As a result,

$$L = \frac{1}{2}m_p l^2 \dot{\theta} sin(\theta)^2 + \frac{1}{2}m_p r^2 \dot{\phi}^2 + \frac{1}{2}m_p l^2 \dot{\theta}^2 cos(\theta)^2 + m_w r^2 \dot{\phi}^2 + \frac{1}{2}m_p l^2 \dot{\theta}^2 - m_p g l cos(\theta)$$

We derive the equations of motion using the Lagrange equations of the second kind (1.1).
As generalized coordinates, we take the angles of rotation of the wheel and the pendulum. Then the vector of generalized coordinates is represented as vector of generalized forces is as follows:

$$q = \begin{pmatrix} \phi \\ \theta \end{pmatrix}$$

$$\tau = \begin{pmatrix} M_k \\ 0 \end{pmatrix}$$

Substituting the derivatives into the Lagrange equations (1.1), we derive the **equations of movement** :

$$r cos(\theta) l m_p \ddot{\theta} + r^2 (m_p + 2m_w) \ddot{\phi} - r sin(\theta) \theta^2 l m_p = M_k$$
$$\ddot{\phi} cos(\theta) l m_p r - m_p g l sin(\theta) + 2 m_p l^2 \ddot{\theta} = 0$$

Theory

Mathematical Model

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau$$

$$q = \begin{pmatrix} \phi \\ \theta \end{pmatrix}$$

$$M(q) = \begin{bmatrix} r^2(m_p + 2m_w) & rcos(\theta)lm_p \\ rcos(\theta)lm_p & 2m_p l^2 \end{bmatrix}$$

$$C(q,\dot{q}) = \begin{bmatrix} 0 & -r\dot{\theta}sin(\theta)lm_p \\ 0 & 0 \end{bmatrix}$$

$$G(q) = \begin{pmatrix} 0 \\ -m_p g l sin(\theta) \end{pmatrix}$$

$$\tau = \begin{pmatrix} M_k \\ 0 \end{pmatrix}$$

To control the resulting system, we construct a linear-quadratic controller. To do this, we linearize the obtained equations of motion in neighborhood of the zero position of the pendulum using Jacobians:

$$\frac{d}{dt}\begin{bmatrix} \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \end{bmatrix} = AX + BM_k = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-m_p g}{r(m_p+4m_w)} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{g(m_p+2m_w)}{l(m_p+4m_w)} & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2}{r^2(m_p+4m_w)} \\ 0 \\ \frac{1}{lr(m_p+4m_w)} \end{bmatrix} M_k$$

A – state matrix
X – state vector
B – input matrix
Mk – input vector

Hardware Design

Hardware used:

1. Arduino Mega 2560
2. MPU6050 Sensor
3. 300 RPM DC Gear Motors with Encoders
4. L298N Dual H-Bridge Motor Controller
5. XBee Module
6. 3 cell Li-Po battery

1. Arduino Mega 2560

The **Arduino Mega 2560** is a microcontroller board based on the **Atmega 2560**.

Features:

- 54 digital input/output pins (of which 15 can be used as PWM outputs)
- 16 Analog input pins
- 4 UARTs (hardware serial ports)
- 16 MHz crystal oscillator
- 256 KB of Flash Memory
- 6 external interrupts
- 5 V Operating Voltage
- 37 g in weight

2. MPU6050 Sensor

MPU-6050 is an 8 pin 6 axis gyro and accelerometer in a single chip. This module works on **I2C** serial communication. For I2C this has **SDA** and **SCL** lines.

3-axis Gyroscope

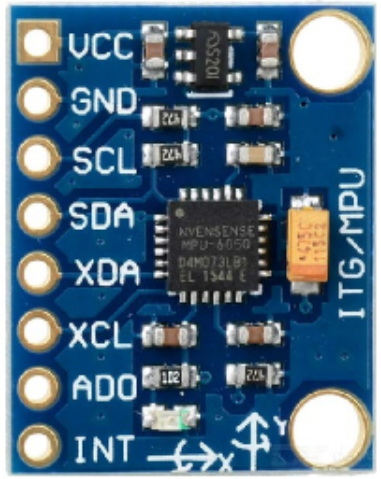
The MPU-6050 consist of a 3 axis gyroscope which can detect rotational velocity along the x,y,z axis with micro electro mechanical system technology (MEMS).

- When the sensor is rotated along any axis a vibration is produced due to **Coriolis effect** which is detected by the MEMS.
- 16-bit ADC is used to digitize voltage to sample each axis (32767 to - 32768).
- **+/- 250**, +/- 500, +/- 1000, +/- 2000 are the full scale range of output.
- Angular velocity is measured along each axis in **degree per second** unit.

3-axis Accelerometer

The MPU-6050 consist of a 3 axis accelerometer which can detect angle of tilt or inclination along the x,y,z axis with micro electro mechanical system technology (MEMS).

- Acceleration along the axes deflects the moving mass which in turn unbalances the differential capacitor which is detected with electro mechanical system technology (MEMS). The Output amplitude is proportional to acceleration.
- 16-bit ADC is used to digitize the values (32767 to - 32768).
- Acceleration is measured along each axis in terms of **g (9.8 m/s^2)**
- **+/- 2g**, +/- 4g, +/- 8g, +/- 16g are the full scale range of output.
- In normal position that is when the device is placed on a flat surface the values are 0 g on x axis, 0 g on y axis and +1 on z axis.



3. 300 RPM DC Gear Motors with Encoders

A Gear motor is a specific type of electrical motor that is designed to **produce high torque** while maintaining a low horsepower, or low speed, motor output. Gear motors are primarily used to reduce speed in a series of gears, which in turn creates more torque. This is accomplished by an integrated series of gears or a gearbox being attached to the main motor rotor and shaft via a second reduction shaft. The second shaft is then connected to the series of gears or gearbox to create what is known as a series of reduction gears. Generally speaking, the longer the train of reduction gears, the lower the output of the end, or final, gear will be.

Motor Specifications:

Stall Current = 2.3-2.5 Amps
Stall Torque = 8.5 Kg-cm
Base RPM = 9000 rpm
Shaft RPM after Gear Box = 300 rpm
Gear Box Ratio = 1:30

DC Gear Motor with Quadrature Encoder

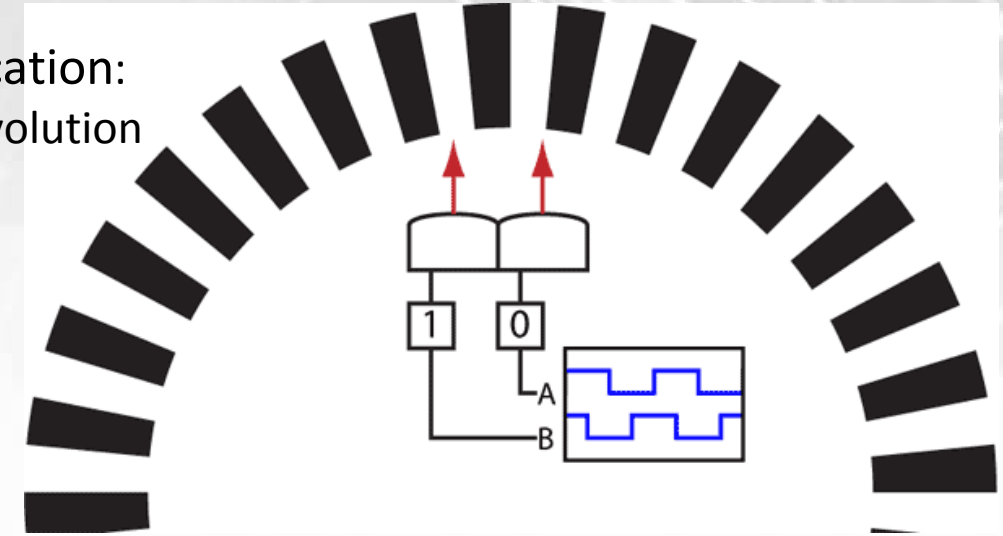
Quadrature Encoders

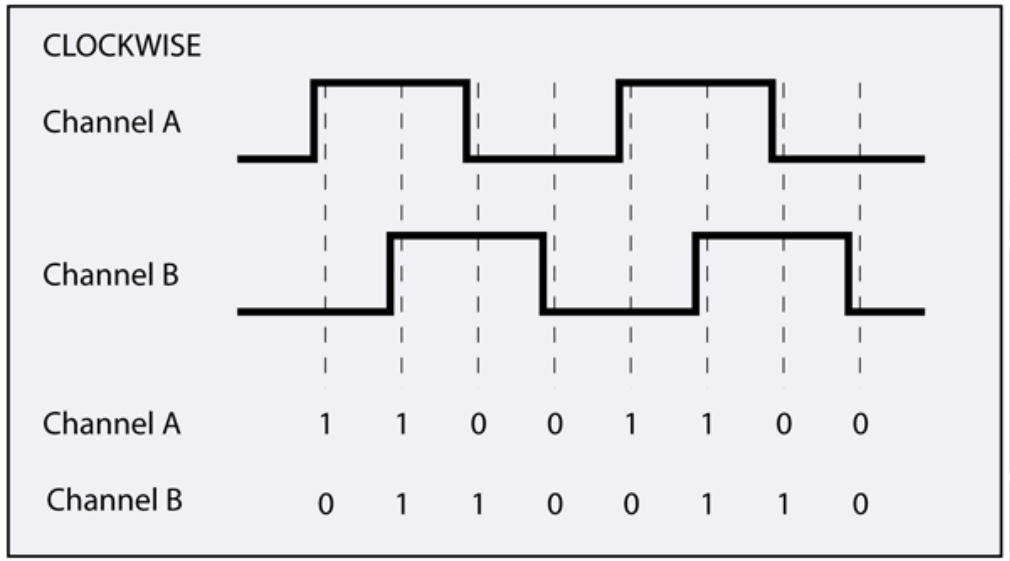
Quadrature Encoders are handy sensors that lets us measure the **speed** and **direction** of a rotating shaft and keep track of how far we have moved.

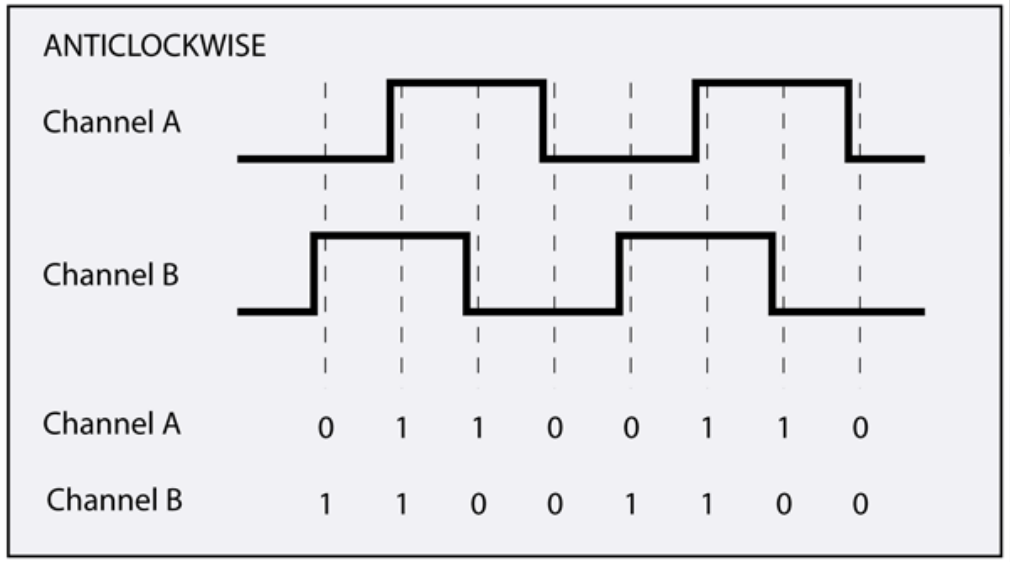
A quadrature encoder has two outputs - Channel A and B - each of which will produce digital pulses when the thing they are measuring is in motion.

We connect channel A of both the Motors as Interrupts in Arduino. We trigger the interrupts in **Rising Edge**. We then check the value of channel B at that instant. If channel B is 0, then the motor is moving in clockwise direction else if channel B is 1, then the motor is moving in anti-clockwise direction.

Encoder Specification:
270 Pulses per revolution
per channel.



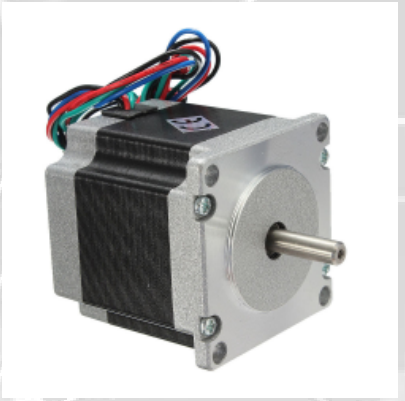




Why DC Motors with Encoders?

1. **DC Motors** are continuous rotation analog motors, which require only power signals to be driven. They have very high RPM and hence, they require gear reduction mechanism to provide sufficient torque, which is required for better stability for our Balance Bot. The gear mechanism introduces backlash error, which affects the motor response. DC motors are open loop actuators and do not provide any feedback. However, their operation is comparatively smooth, quiet and vibration-free.

2. **Stepper Motors** on the other hand require digital signals and they rotate in increments with the help of electromagnets. They provide higher accuracy because of this positional control no extra feedback programming is required. However, this working principle of rotational increments also reduces their reaction time and they will not be capable of balancing the robot quickly. Comparatively, they provide more torque at lower speeds. However, they have lower power efficiency. At higher speeds, the torque drops rapidly. They are bulky and are generally more expensive. Programming the rotation increments is slightly more complex compared to DC Motors.



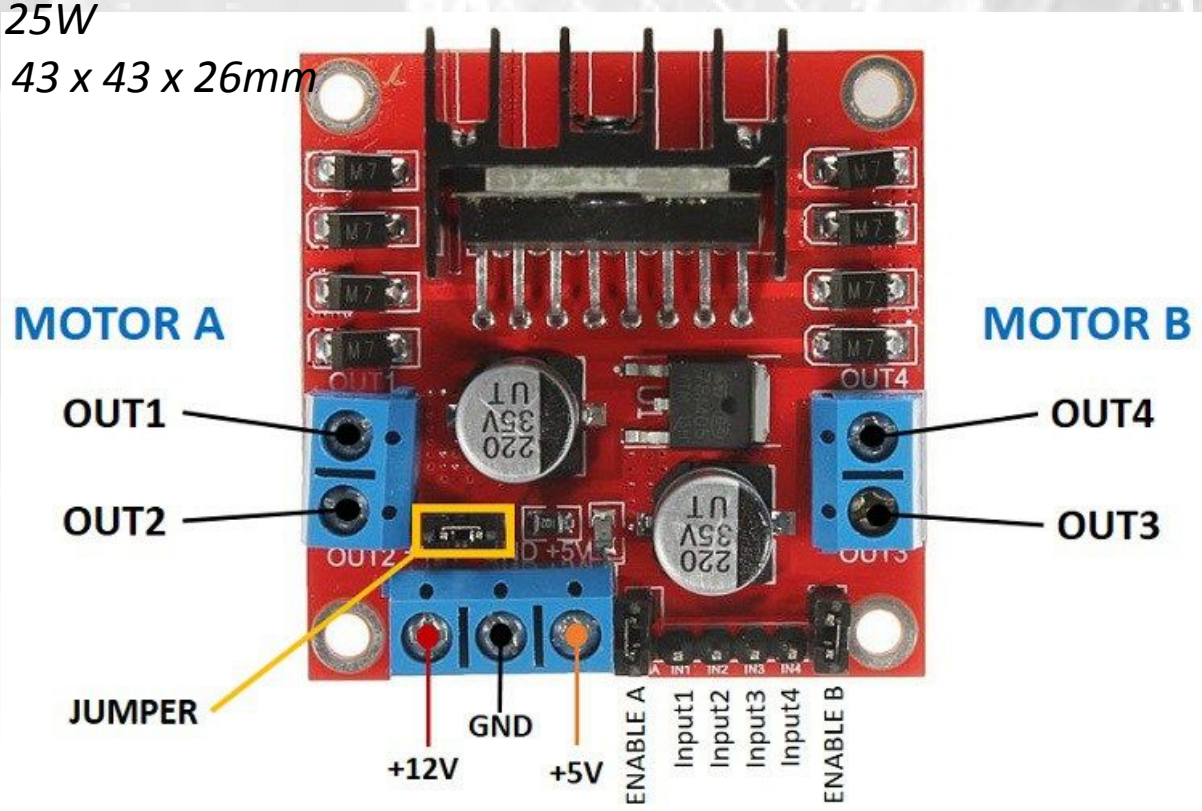
3. **DC Motors with Encoders** are the same as described previously but these form a closed loop system providing positional feedback with the help of incremental shaft encoders. Hence, we can measure accurate linear and rotational position of the robot, achieved through simple interrupt programming of encoder pulses. This is helpful in computing the robot's velocity for better motor control and hence overall **stability** of the robot. This is the best trade-off in terms of cost and performance.

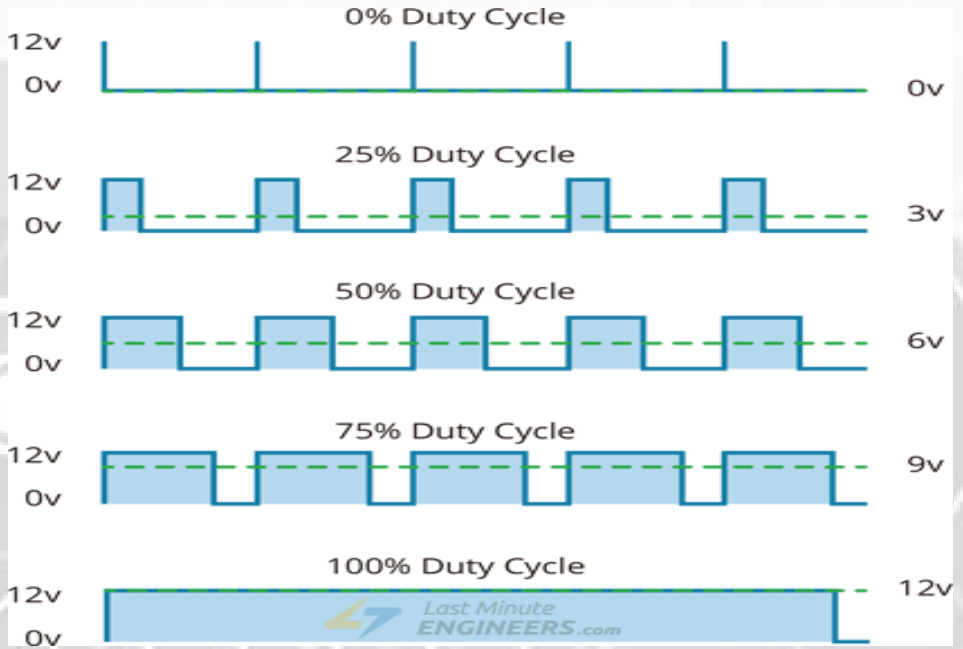
4. L298N Dual H-Bridge Motor Controller

H-Bridge is used in controlling motors **speed** and **direction.**
The speed is controlled by **Pulse Width Modulation (PWM)**.

Specifications:

- Double H bridge Drive Chip: *L298N*
- Logical voltage: *5V Drive voltage: 5V-35V*
- Logical current: *0-36mA Drive current: 2A (MAX single bridge)*
- Max power: *25W*
- Dimensions: *43 x 43 x 26mm*
- Weight: *26g*





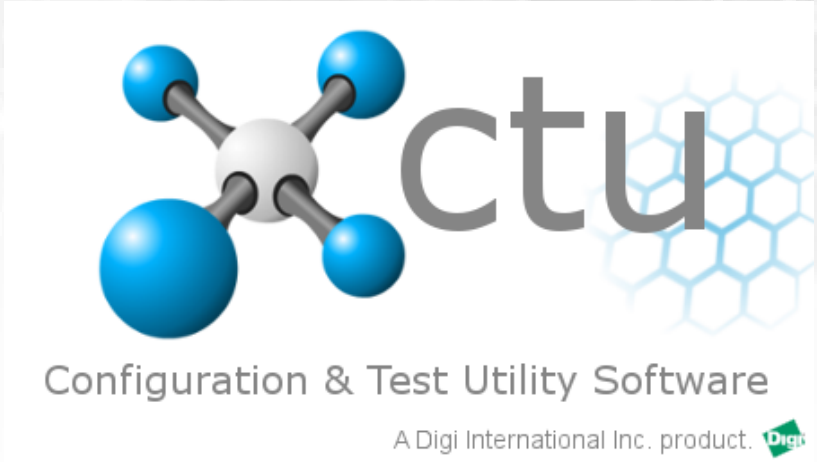
ENA	IN1	IN2	Description
0	N/A	N/A	Motor A is off
1	0	0	Motor A is stopped (brakes)
1	0	1	Motor A is on and turning backwards
1	1	0	Motor A is on and turning forwards
1	1	1	Motor A is stopped (brakes)

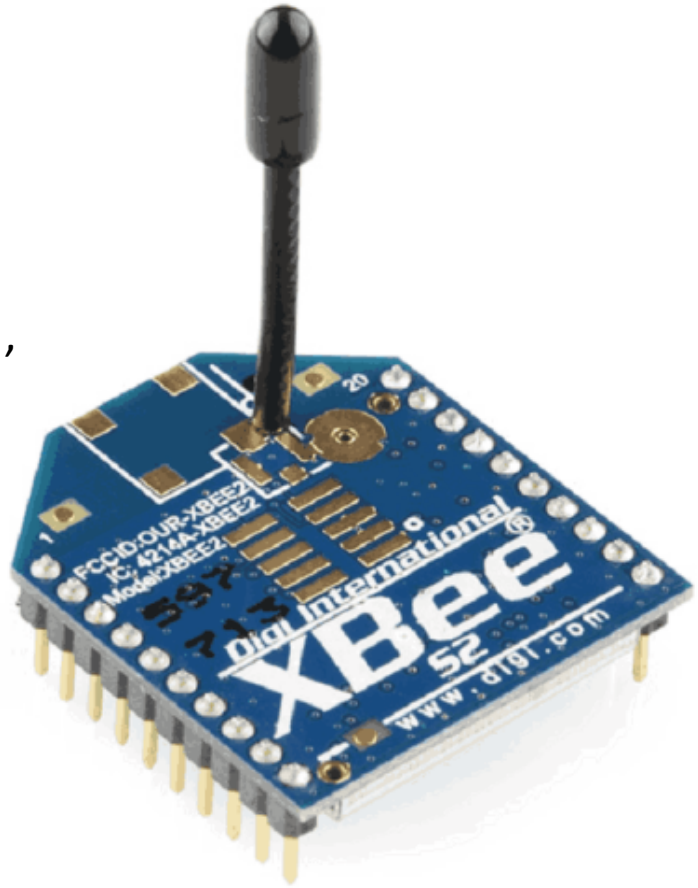
5. XBee Module

XBee S2C is a **RF module** designed for wireless communication or data exchange and it works on **ZigBee** mesh communication protocols that sit on top of IEEE 802.15.4 PHY. The module provides wireless connectivity to end-point devices in any ZigBee mesh networks.

Features and Electrical Characteristics

- Transmission Frequency: 2.4GHz to 2.5GHz
- Indoor/Urban Range: 200ft
- Outdoor RF line-of-sight Range: up to 4000ft
- RF Data Rate: 250,000 bps
- Supply Voltage Range: +2.1V to +3.6V
- Operating Current: 33mA (at3.3V, for Normal mode) , 45mA (at 3.3V,for Boost mode)
- Maximum output current on all pins together: 40mA





XBee Module

XBee Pin Configuration

6. 3 cell Li-Po Battery (Lithium-Polymer)

Specifications:

Voltage / Cell Count – 11.1 V / 3S

A LiPo cell has a nominal voltage of 3.7V. For the 11.1V battery, that means that there are three cells in series (which means the voltage gets added together). A "3S" battery pack means that there are 3 cells in **S**eries. So a three-cell (3S) pack is 11.1V, and so on.

Capacity – 2200 mAh

The capacity of a battery is basically a measure of how much power the battery can hold. The unit of measure here is milliamp hours (mAh). This is saying how much drain can be put on the battery to discharge it in one hour.

Discharge Rating ("C" Rating) – 35C

The C Rating is simply a measure of how fast the battery can be discharged safely and without harming the battery.
35C = 35 x Capacity (in Amps)
Calculating the C-Rating of our example battery: 35 x 2.2 = 77A
The resulting number is the maximum sustained load you can safely put on the battery. Going higher than that will result in, at best, the degradation of the battery at a faster than normal pace. At worst, it could burst into flames. So our battery can handle a maximum continuous load of **77A.**

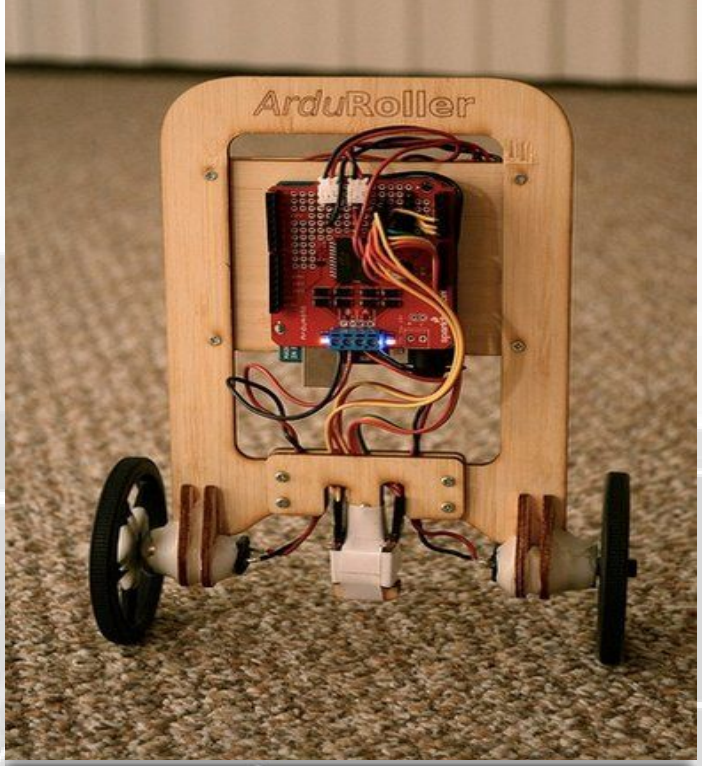
Chassis Design

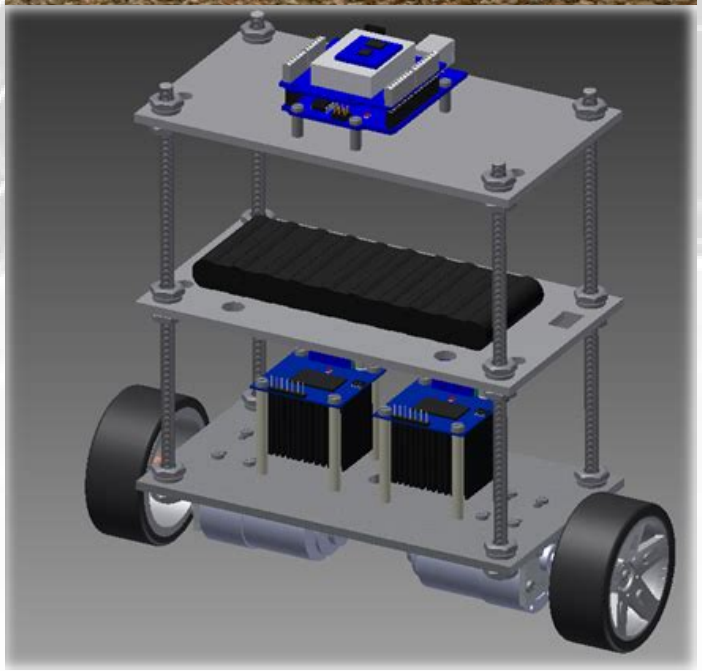
Two types of Chassis design:
1. Vertical design
2. Stackable design

We selected **Stackable** design for our Bot

Stability and Response – The moment of inertia of the robot must be high so that it tilts more slowly when external torque is applied. The mass in the vertical design is distributed evenly, whereas it is distributed as different tiers in the stacked design. Hence, the different tiers can be moved up and down in order to shift the center of gravity as required. The taller the robot, more the stable it is.

Construction – The stacked design is more feasible to construct compared to the vertical design. Various components can be easily mounted and removed, without having to worry much about the CG shift as the tiers can easily be shifted later as required. Another major advantage is that, the stacked design is scalable. For example, if we want the robot to carry some weight, this can be easily done in the stacked design by adding a new tier. The design can be constantly updated and new features can be incorporated with ease. On the other hand, creating vertical stacks (along the depth) is not very feasible.



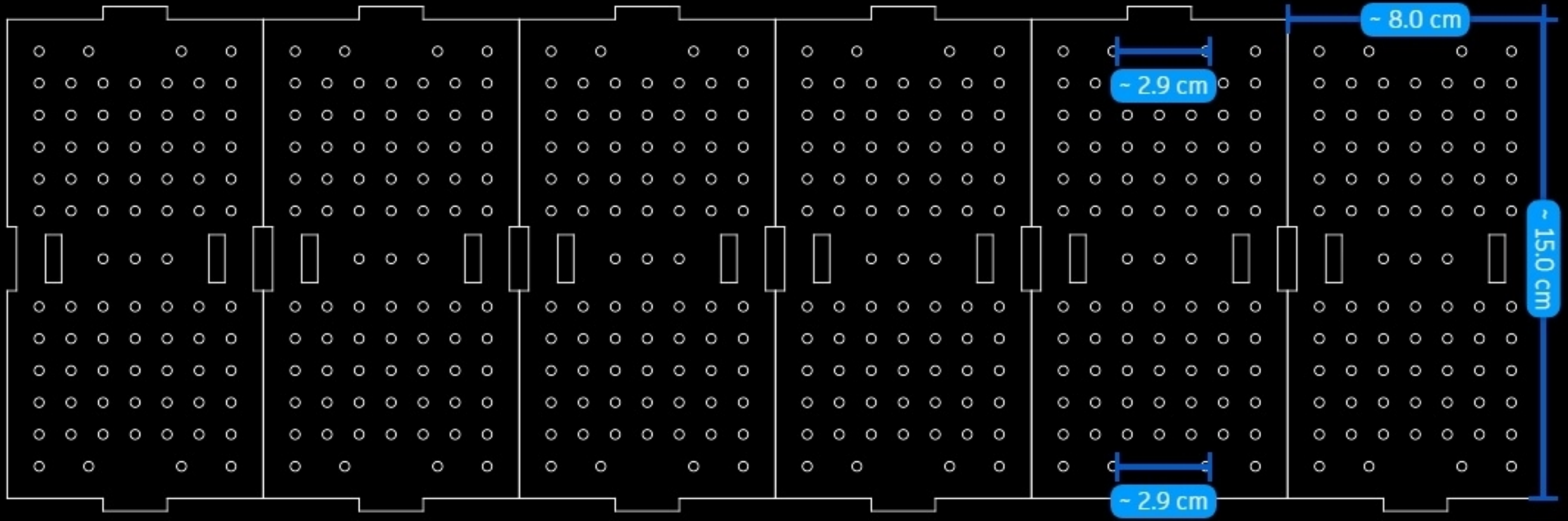


Chassis Design

Tier design

- Material selected: Acrylic sheet
- Thickness: 3mm

- Designed using AutoCAD software
- Fabricated with Laser cutter



Chassis Design

DC Motor Clamps

Specifications:

- Material: Stainless Steel
- Thickness: 2mm

Wheels

Specifications:

- Diameter: 6.4 cm
- Thickness: 2.6 cm

Brass Studs

1. M4x20 x 4
2. M4x48 x 4
3. M4x60 x 4

Chassis Design

Placement of Components

A total of **4** tiers are used, starting from Tier 0 (lower most tier) to Tier 3 (upper most tier)

Tier 0:
Lower side – Motors and MPU6050 Sensor
Upper side – Li-Po Battery

Tier 1:
Upper side – Power distribution circuit and Motor driver

Tier 2:
Upper side – Arduino Mega 2560 and XBee module

Tier 3:
No components are placed on this Tier. This tier is used for support and to increase the height of the Bot

Signal Processing

Signal processing is a field of engineering that focuses on analysing, modifying and synthesizing signals such as sound, images, biological measurements etc.

In signal processing, a "**Filter**" is a device or process that removes some unwanted components or features from a signal. Filtering is a class of signal processing, the defining feature of filters being the complete or partial suppression of some aspect of the signal. Most often, this means removing some frequencies or frequency bands

Frequency filter circuits (such as low-pass, high-pass, band-pass, and band-reject) shape the frequency content of signals by allowing only certain frequencies to pass through.

We have used 3 types of filters

1. Low pass filter (Used to filter Accelerometer data)
2. High pass filter (Used to filter Gyroscope data)
3. Complimentary filter (Used to combine Accelerometer and Gyroscope data)

Signal Processing

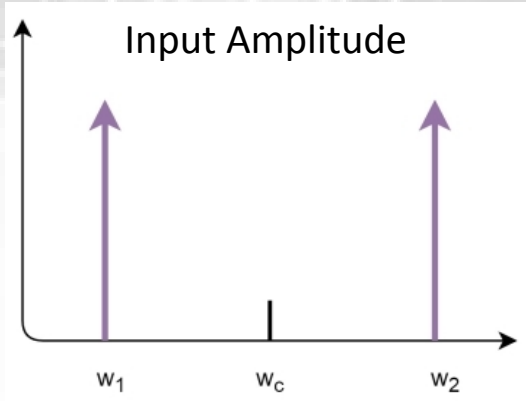
Low Pass Filter (Used to filter Accelerometer data)

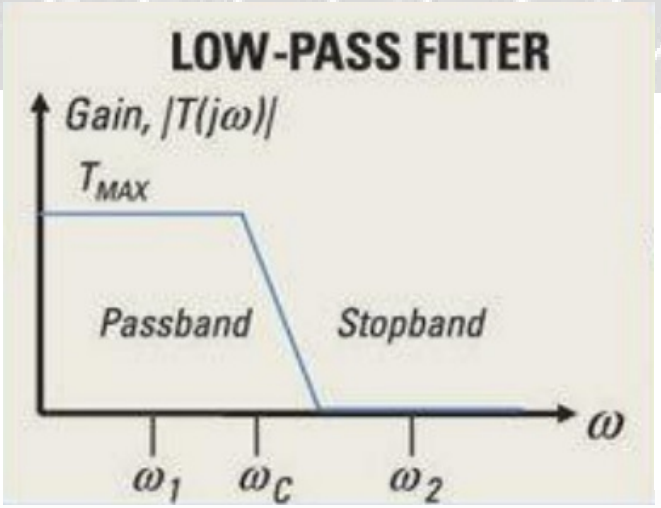
The low-pass filter has a gain response with a frequency range from zero frequency (DC) to cutoff frequency (ωc), which means that any input that has a frequency below the ωc gets a pass, and anything above it gets attenuated or rejected. The gain approaches zero as frequency increases to infinity

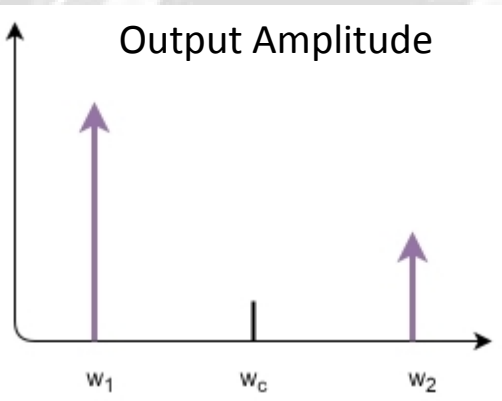
The input signal of the filter shown here has equal amplitudes at frequencies ω1 and ω2. After passing through the low-pass filter, the output amplitude at ω1 is unaffected because it's below the cutoff frequency ωc. However, at ω2, the signal amplitude is significantly decreased because it's above ωc.

Equation for low-pass filter:
y[n] = (1-alpha).x[n] + alpha.y[n-1]
x[n] is the unfiltered input signal
y[n-1] is the previous filtered output signal
y[n] is the filtered final output signal

alpha = (tau)/(tau + dt) where tau is the desired time constant (how fast you want the readings to respond) and dt = 1/fs where fs is your sampling frequency.





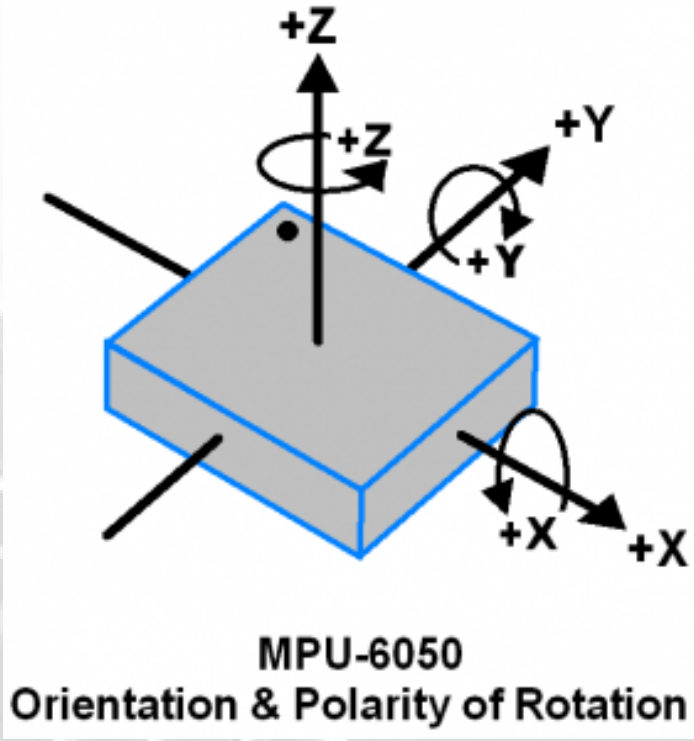


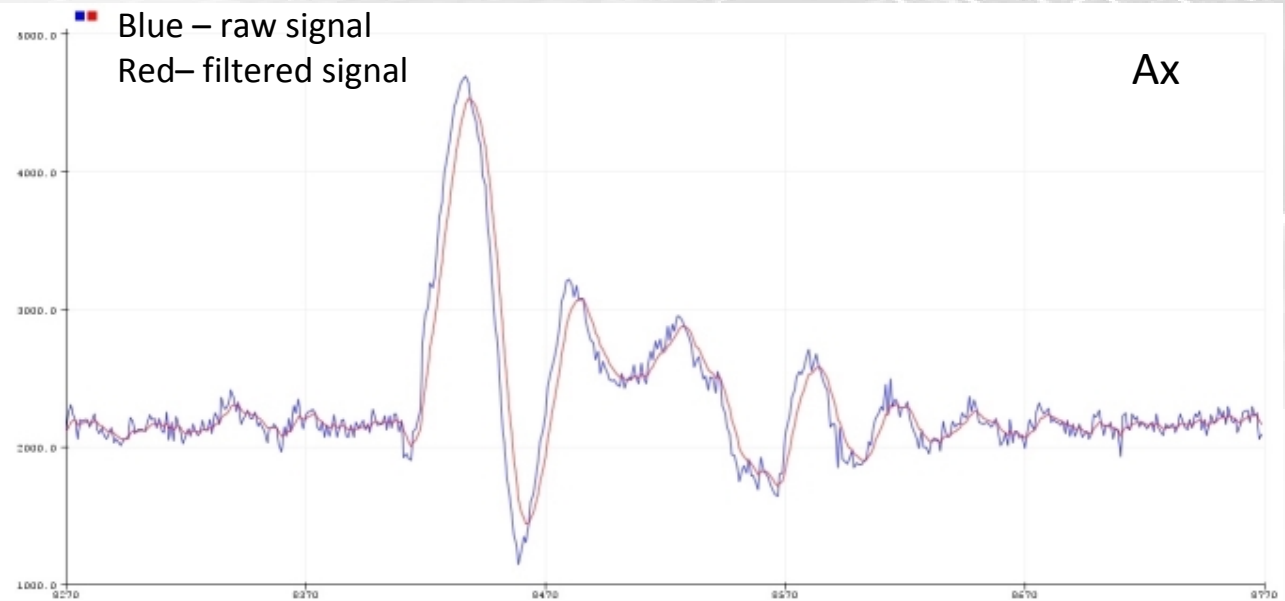
Signal Processing

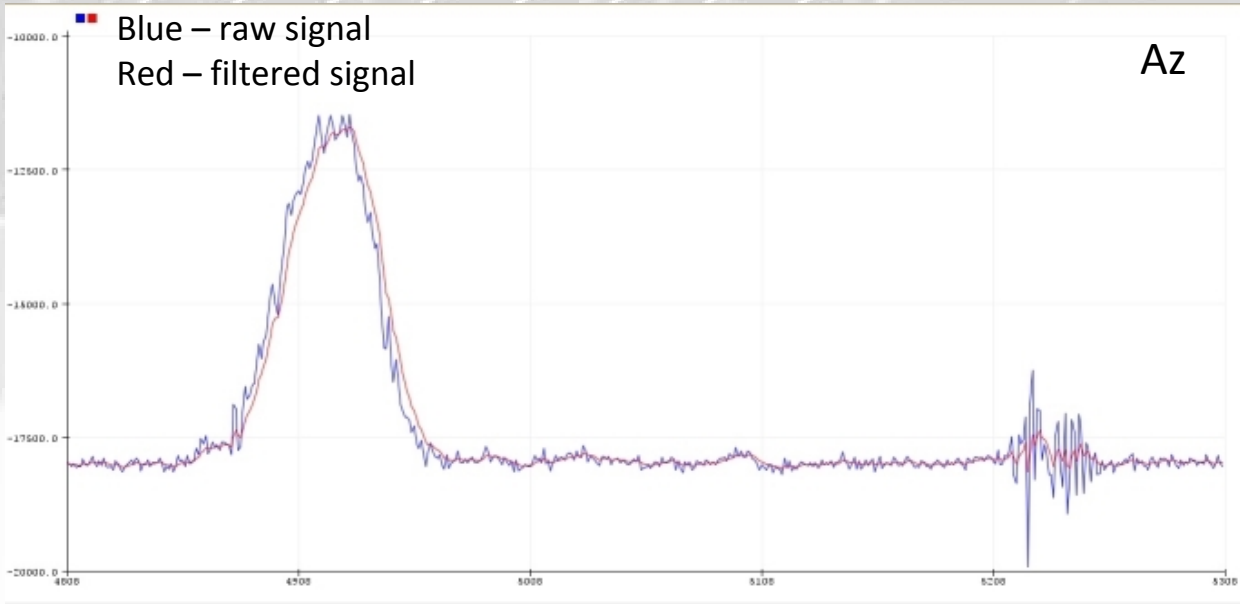
Low Pass Filter (Used to filter Accelerometer data)

Calculation of the Pitch angle using Accelerometer is done using the formula
acc = arcTan(Ax / Az)

First we pass the signals Ax and Az through Low pass filters, then we find the angle acc
f_cut= 5, dt = 0.01, alpha = 0.76







Signal Processing

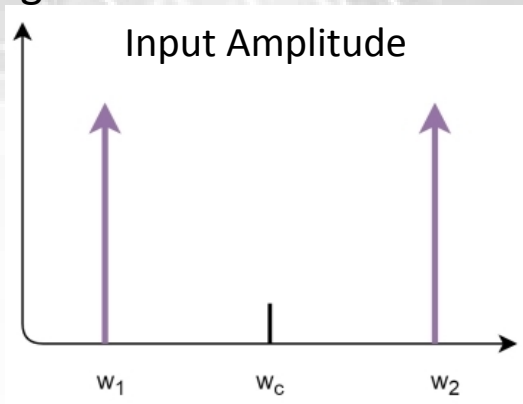
High Pass Filter (Used to filter Gyroscope data)

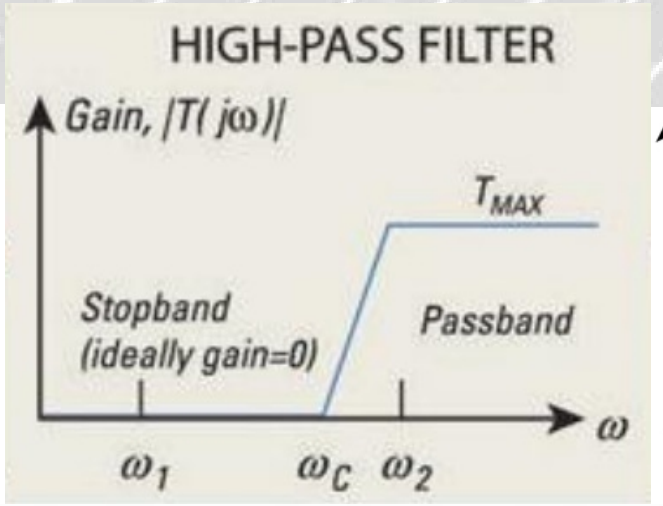
Opposite to the Low-pass filter, the high-pass filter has a gain response with a frequency range from the cutoff frequency (ωc) to infinity. Any input having a frequency below ωc gets attenuated or rejected. Anything above ωc passes through unaffected.

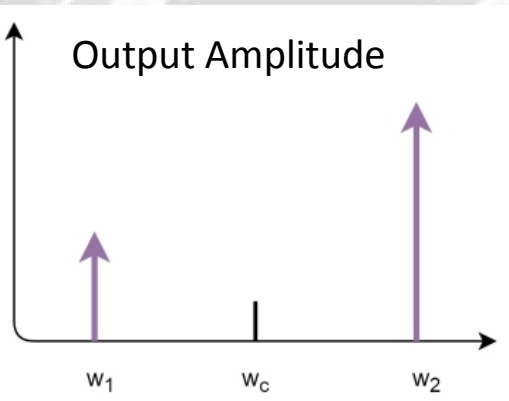
The input signal of the filter shown here has equal amplitude at frequencies ω1 and ω2. After passing through the high-pass filter, the output amplitude at ω1 is significantly decreased because it's below ωc, and at ω2, the signal amplitude passes through unaffected because it's above ωc

Equation for high-pass filter:
y[n] = (1-alpha).y[n-1] + (1-alpha)(x[n]-x[n-1])
x[n] is the unfiltered input signal
x[n-1] is the previous input signal
y[n-1] is the previous filtered output signal
y[n] is the filtered final output signal

alpha = (tau)/(tau + dt) where tau is the desired time constant (how fast you want the readings to respond) and dt = 1/fs where fs is your sampling frequency.



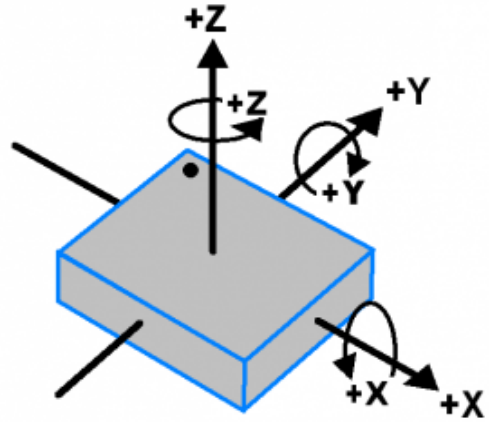




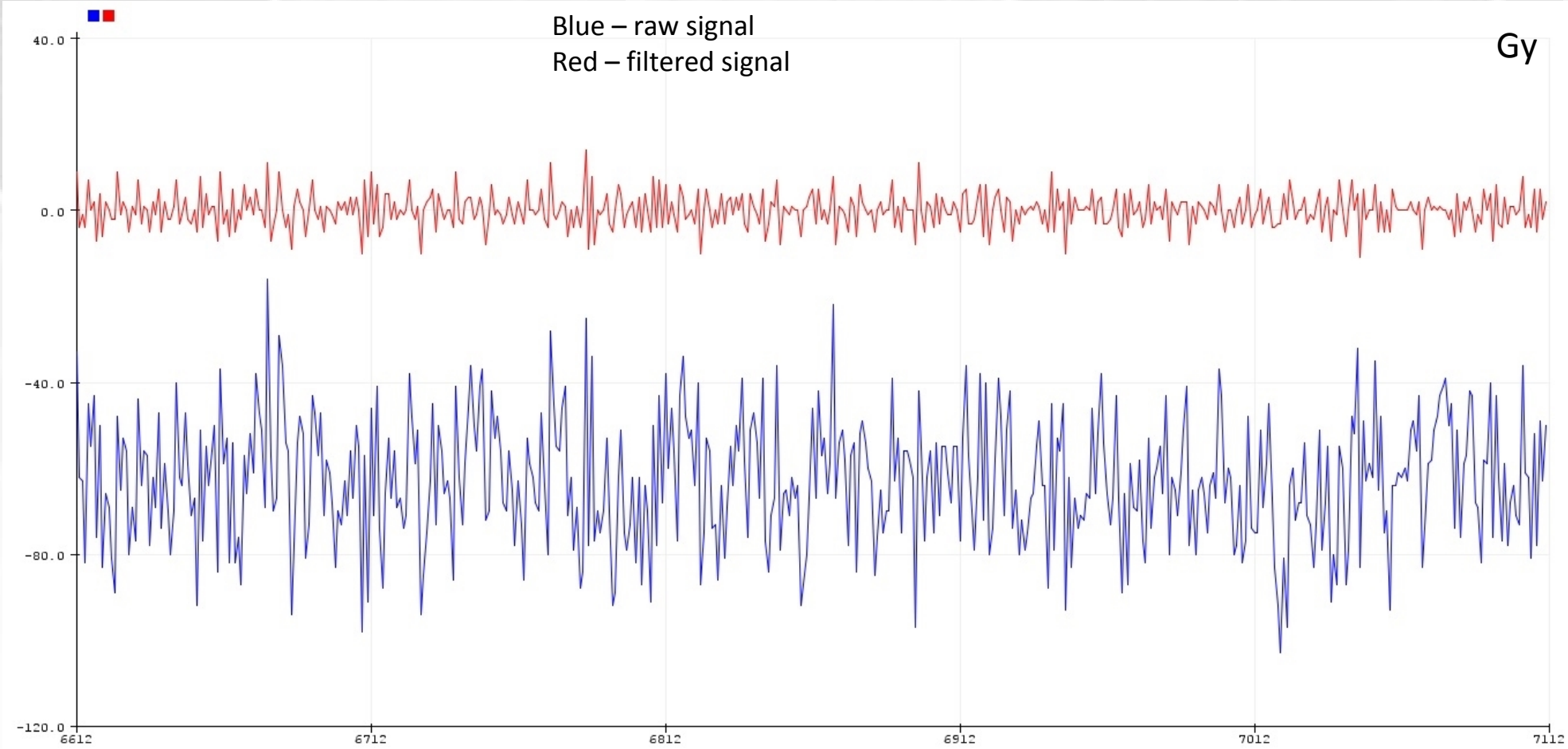
Signal Processing

High Pass Filter (Used to filter Gyroscope data)

The gyroscope data **Gy** is taken
f_cut= 5, dt = 0.01, alpha = 0.76



MPU-6050
Orientation & Polarity of Rotation



Signal Processing

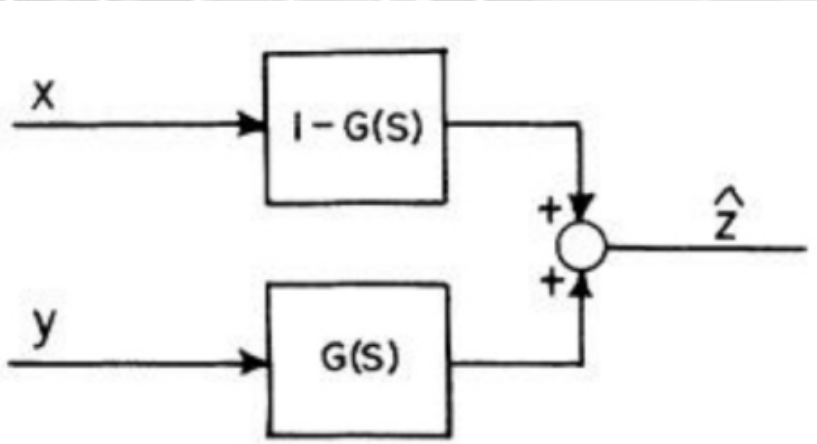
Complimentary Filter

Complementary filter is a simple estimation technique to combine measurements. The basic complementary filter is shown in the figure below where x and y are noisy measurements of signal z; $\hat{Z}$ is the estimate of z produced by the filter. Assume that the noise in y is mostly high frequency, and the noise in x is mostly low frequency. Then G(s) can be made a low pass filter to filter out the high-frequency noise in y. If G(s) is low-pass, [1-G(s)] is the compliment, i.e., a high pass filter which filters out the low-frequency noise in x.

The calculated tilt angle from the accelerometer data has slow response time, while the integrated tilt angle from the gyro data is subjected to drift over a period of time. In other words, we can say that the accelerometer data is useful for long term while the gyro data is useful for short term. Idea behind complementary filter is to take slow moving signals from accelerometer and fast moving signals from a gyroscope and combine them. Complementary filter is designed in such a way that the strength of one sensor will be used to overcome the weakness of the other sensor which is complementary to each other.

angle = (1- alpha)(angle + gyro * dt) + (alpha)(acc)
First reading is the angle as obtained from gyroscope integration. Second reading is the one from accelerometer

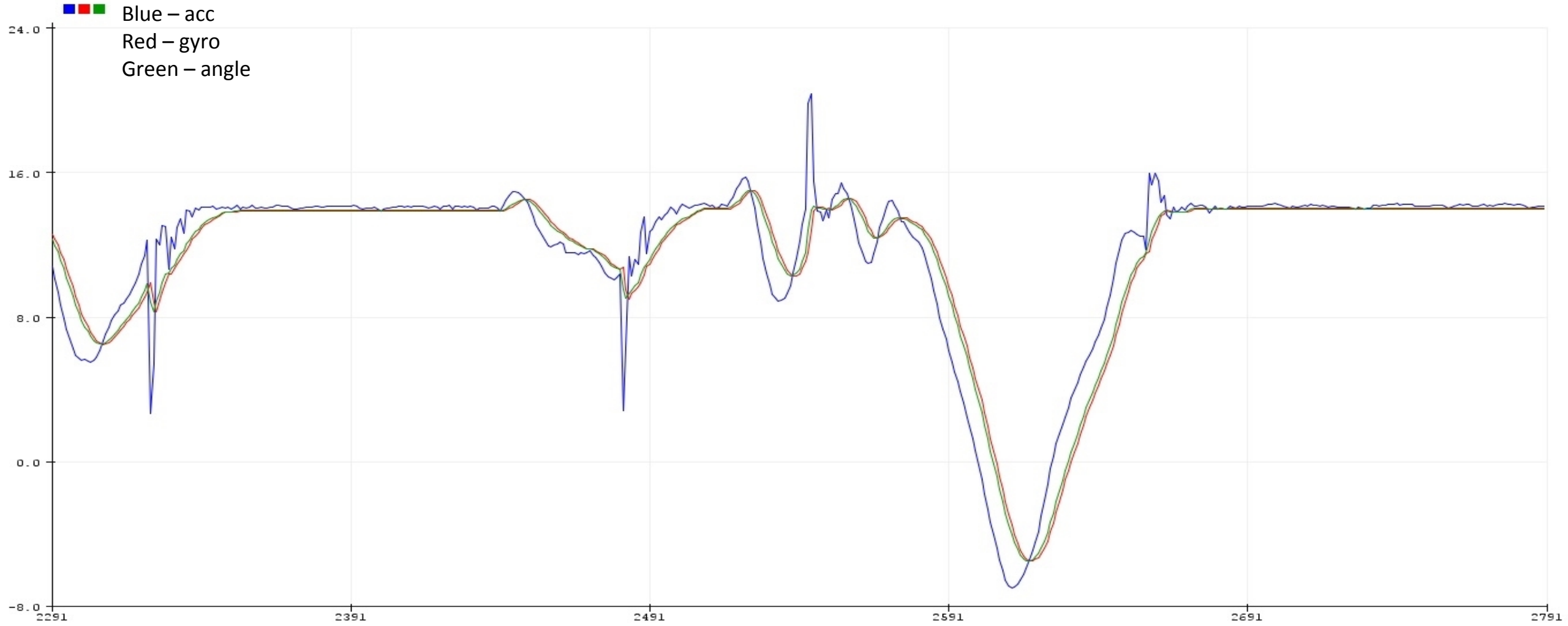
alpha = 0.15



Signal Processing

Complimentary Filter

alpha = 0.15



Controller Design

Linear Quadratic Regulator (LQR)

Linear Quadratic Regulator is a powerful tool which helps us choose the K matrix according to our desired response.
Here we use a cost function

$$J = \int_0^\infty (x^T Q x + u^T R u)\, dt$$

where Q and R are positive semi-definite diagonal matrices (positive semi-definite matrices are those matrices whose all the eigenvalues are greater than or equal to zero). Also to remind you that for a diagonal matrix, the diagonal entries are its eigenvalues. x and u are the state vector and input vector respectively.
Let us say that you have your system with four states and one input.

Let us say that you have your system with four states and one input. Then $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$ and input u

$$Q = \begin{bmatrix} Q_1 & 0 & 0 & 0 \\ 0 & Q_2 & 0 & 0 \\ 0 & 0 & Q_3 & 0 \\ 0 & 0 & 0 & Q_4 \end{bmatrix}$$

Controller Design

Linear Quadratic Regulator (LQR)

Then $x^T Q x + u^T R u = Q_1 x_1^2 + Q_2 x_2^2 + Q_3 x_3^2 + Q_4 x_4^2 + R u^2$

The controller is of form **u = −Kx** which is a Linear controller and the underlying cost function is Quadratic in nature and hence the name Linear Quadratic Regulator. With a careful look at the integrand of the cost function J, we may observe that each Qi are the weights for the respective states xi . So, the trick is to choose weights Qi for each state xi so that the desired performance criteria is achieved. Greater the state objective is, greater will be the value of Q corresponding to the said state variable. We can choose **R = 1** for single input system. In case of inverted pendulums, we know that angle θ with the vertical and the angular velocity $\dot{\theta}$ is very important and hence the weights corresponding to them should be more as compared to linear position x and linear velocity $\dot{x}$. LQR minimizes this cost function J based on the chosen matrices Q and R. Its a bit complicated to find out matrix K which minimizes this cost function. What is required to be done is to choose the Q and R matrix appropriately to get the desired performance. Mathematically its done with Algrebriac Ricatti Equation (ARE) but here we are using Octave's inbuilt lqr function.

Controller Design

Linear Quadratic Regulator (LQR) Simulation

mp = 0.985 r = 0.032
mw = 0.04 g = 9.804
l = 0.067

A=[0 1 0 0
0 0 (-1*mp*g)/(r*(mp+4*mw)) 0
0 0 0 1
0 0 (g*(mp+2*mw))/(l*(mp+4*mw)) 0]

B=[0
2/(r^2*(mp+4*mw))
0
-1/(l*r*(mp+4*mw))]

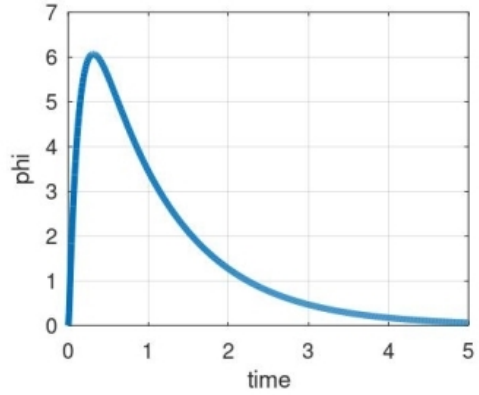
Q=[2 0 0 0 R=1
0 1 0 0
0 0 10 0
0 0 0 5]

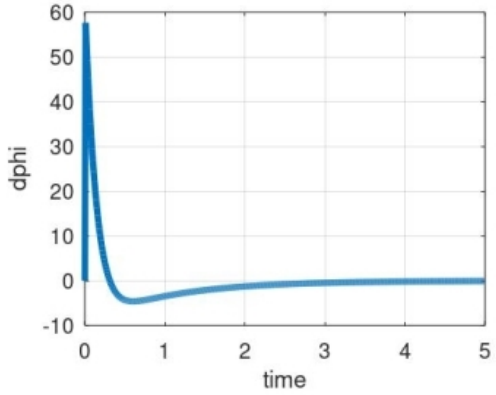
State vector = [phi phi_dot theta theta_dot]

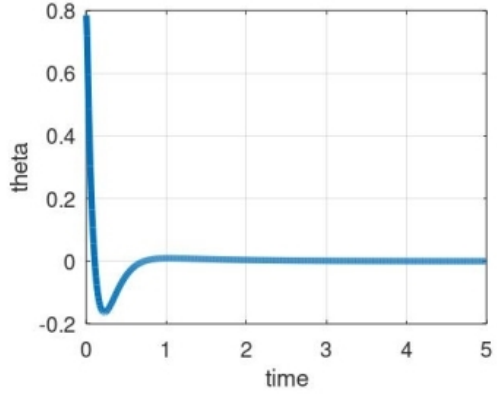
Initial condition= [0 0 3.14/4 0] final condition =[0 0 0 0]

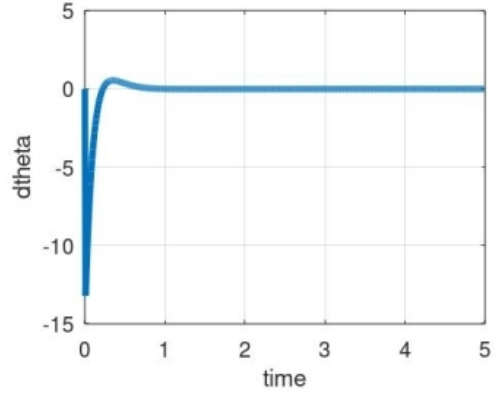
Controller Design

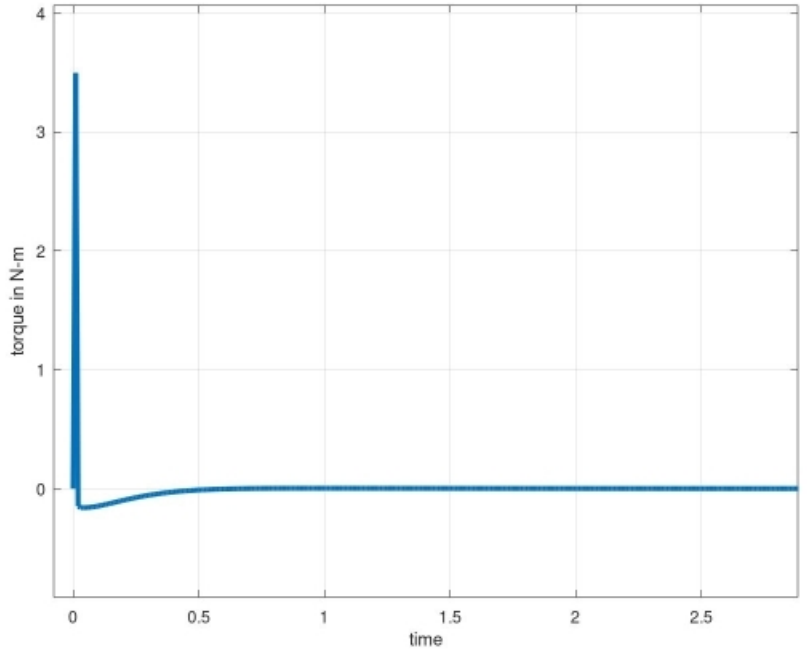
Linear Quadratic Regulator (LQR) Simulation Results











K = [-0.047409 -0.059344 -4.450723 -0.513819]

Simulation Peak Torque = **3.494** Nm

Practical Peak Torque = 0.833*2 = **1.66** Nm

Controller Design

Cascaded P-D Controller

e(t) = Error is the difference between the desired position (Set Point) and the current position (Tilt Angle).

Proportional Term

The response of the proportional term is simply a constant (Kp) times the current error. A very l**ow** gain (Kp) will make the robot very **unresponsive** to disturbances. Whereas, a **high** gain will make the robot **over responsive**. The controller will not only correct the present error but will also introduce its own error, and in turn tries the correct this self-induced error again. Hence, it induces oscillations about the desired position. As oscillations become larger and larger, the robot will became unstable and finally fall down.

Response of the proportional term is given as: **proportional_term = kp * error**

Derivative Term

The derivative term is proportional to rate of change of error. It is not dependent on the current magnitude of error and is incapable of minimizing the error on its own. The purpose of the derivative term is to **anticipate the future** behavior of the error. Thus, it helps in achieving the desired position much faster. However, a high magnitude of derivative gain (Kd) can make the robot very **jittery.** Derivate response is given by:

derivative_term = kd * (current_error – previuos_error)/sampleTime

Finally, the desired output response can be achieved by tuning these PD gains properly to control their individual contributions in the output. The output of the PD control algorithm is given as: **PD_output = proportional_term + derivative_term**

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

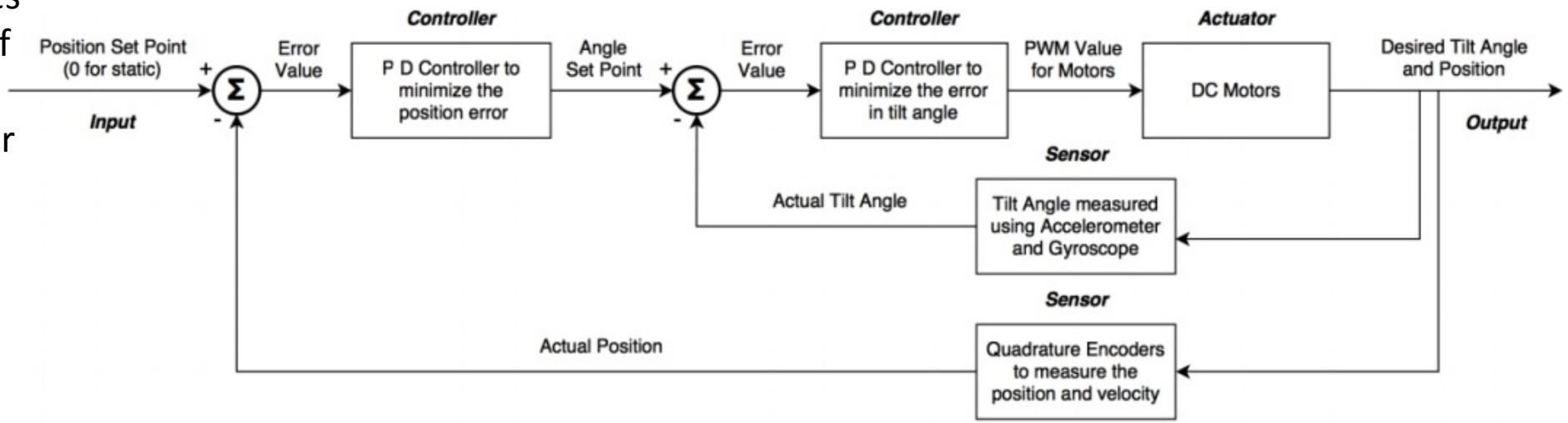
Controller Design

Cascaded P-D Controller

The **inner PD** loop controls the **tilt angle** and the **outer PD** loop controls the **position** of the robot.

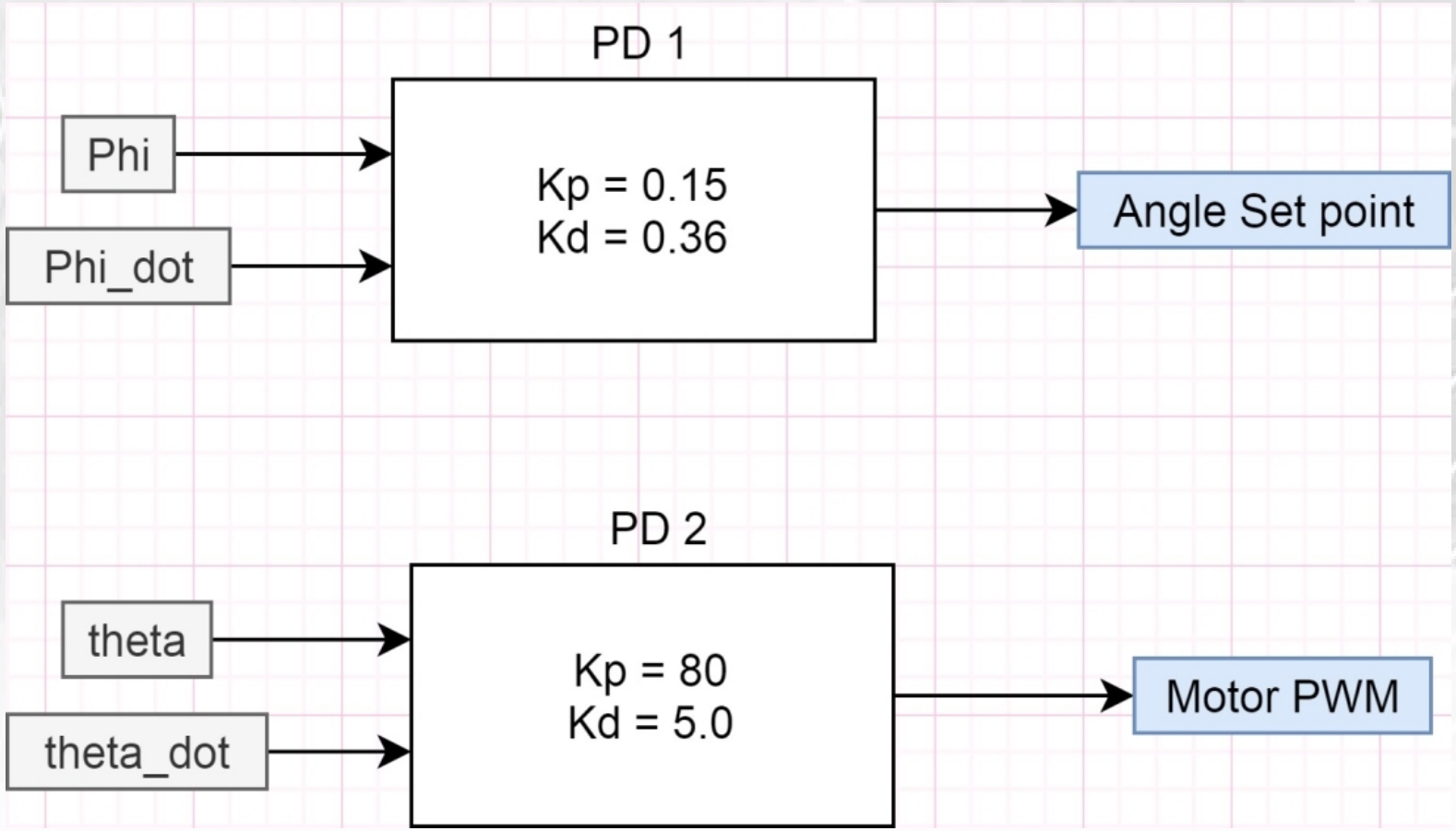
The tilt angle of the balance bot is measured using the accelerometer and the gyroscope fused using a complimentary filter. A PD controller is designed to minimize the error between current tilt angle and the desired set point. The **output** of the PD controller is directly used to set the **PWM** of the motors to control its speed and the direction (forward or backward) based on the direction of tilt.

Another PD controller is designed which minimizes the error between the current position and the desired position (zero for static). The output of this PD controller changes the Angle Set Point. Initially consider the angle set point to be zero. If the robot moves linearly for a given distance, the position PD controller will change the angle set point in the opposite direction (say +2 deg). This in turn makes the robot now to balance itself at +2 deg instead of 0 degrees (upright). As the position error approaches to zero, the angle set point also approaches to zero and thus the motor will automatically slow down and position error is corrected.

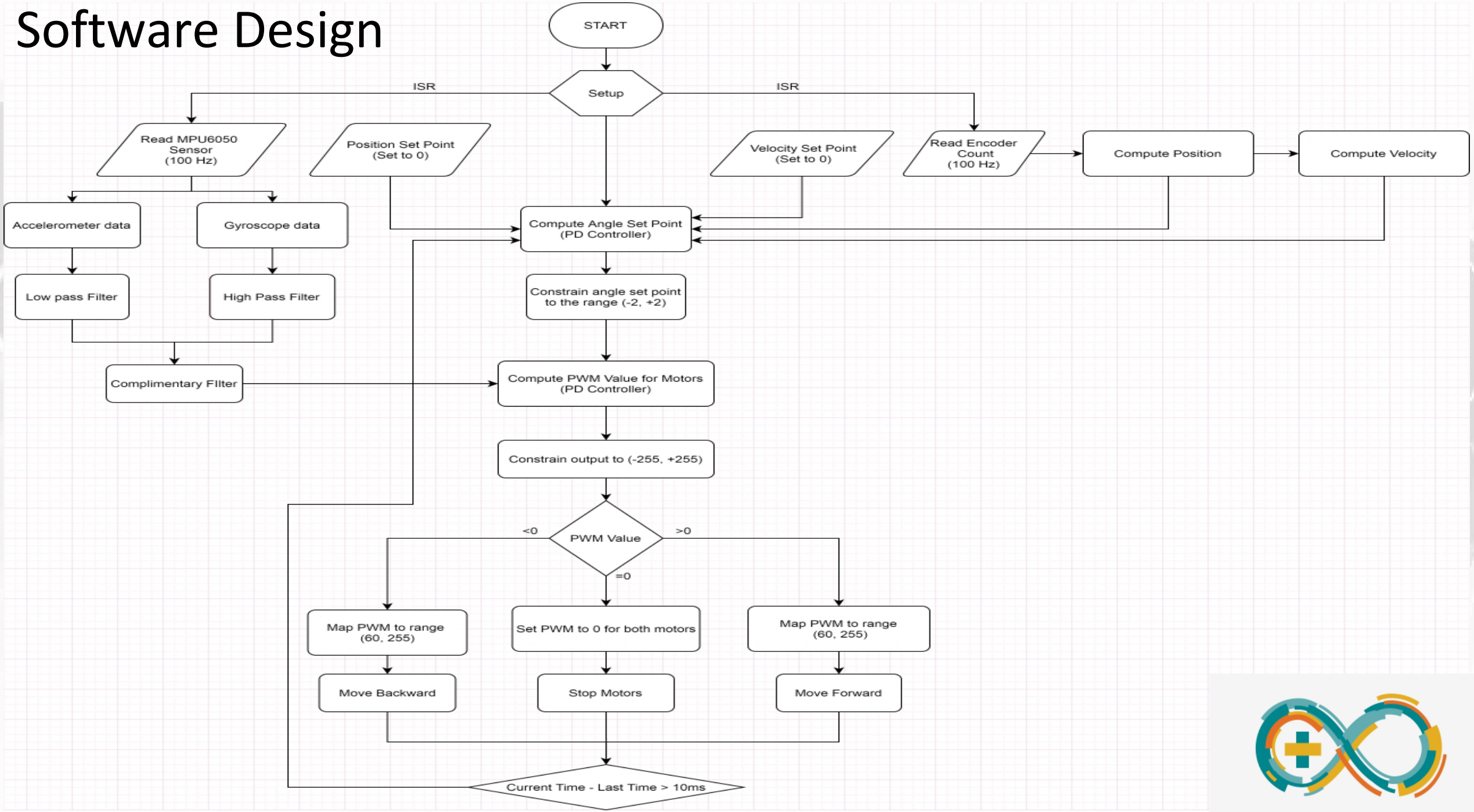


Controller Design

Cascaded P-D Controller Tuning Results



Software Design



Conclusion

The stacked design proved to be efficient in terms of stability because of the increased y axis depth.

Placing the battery at the lowest tier improved the response time of the Bot and increased the maximum recoverable tilt angle.

However while implementing the LQR controller, we noticed that the maximum torque produced by our Bot's motors did not meet the required criteria in the simulation.

We switched to cascaded PD design and noticed that the Bot is able to achieve **Tilt Angle Control** but not Position control. Hence the Bot is able to Balance itself on two wheels using the cascaded PD controller.

To improve the Bot further,

1. We can reduce the overall weight of the Bot by 3D printing few components such as studs
2. We can use higher torque motor

Demo