CONSTRUCTING A SELF-BALANCING ROBOT

AARON ONG*

CONTENTS

LIST OF FIGURES

* *Yale-NUS College*

1 INTRODUCTION

THE self-balancing robot is a classic case of the inverted pendulum control problem. To understand how it works, one must first be familiar with basic pendulum dynamics[1]

2 THEORIES

1. The Inverted Pendulum Control Problem
2. PID Control Theory
3. Complimentary Filter

2.1 Self-Balancing Robot

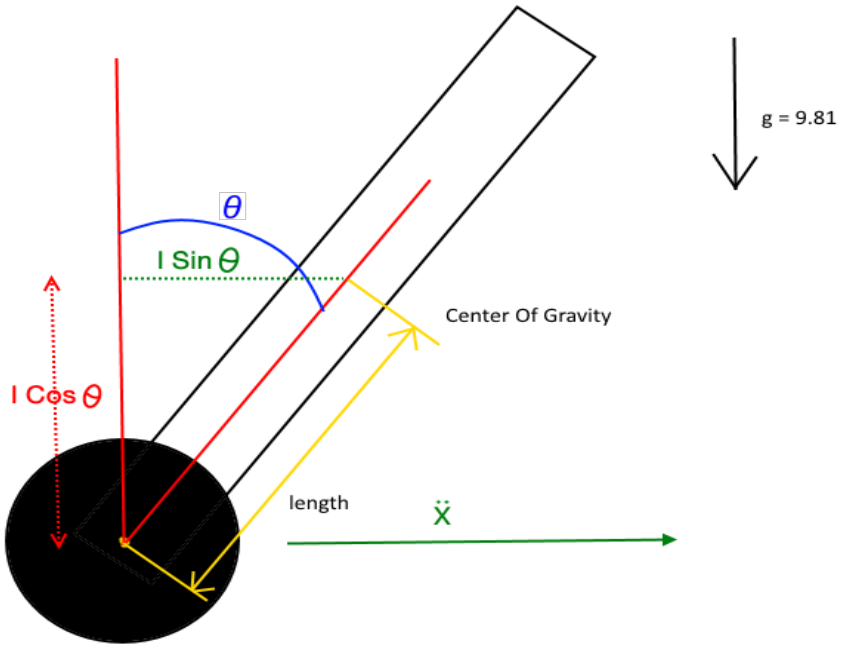


Figure 1: Free Body Diagram of a Self-Balancing Robot

The Self-Balancing Robot is an inverted pendulum mounted on top of two wheels. The pendulum is limited to one-degree of motion, a rotation around the wheel's axis (pitch). In this paper, we will define the angle θ as the tilt of the robot.(figure 1)

When the Self-Balancing Robot tilts towards the positive x direction, gravitational acceleration acts on the displaced center of gravity to generate a torque in the clockwise direction. To counteract this clockwise torque, the Self-Balancing Robot must induce a Torque in the opposite direction. This is done through accelerating in the positive x direction.

[1] *See Appendix 4.1 and 4.2*

Hence, unlike the inverted pendulum, the Self-Balancing Robot's tilt is controlled by two interacting forces:

1. Torque produced by gravitational acceleration
2. Torque produced by acceleration along the x-axis

The Self-Balancing Robot's tilt is defined by the two variable ODE:

$$\ddot{\theta} = \ddot{\theta}_g + \ddot{\theta}_{\ddot{x}} = \frac{g}{l}sin\theta - \frac{\ddot{x}}{l}cos\theta$$

Representing this equation in terms of θ, we arrive at the equation:

$$\ddot{x} = \frac{gsin\theta - l\ddot{\theta}}{cos\theta}$$

LIMITS OF AN ANALYTICAL SOLUTION While this equation gives us a analytical solution to the self-balancing problem, analytical solutions tend to work only for simple mathematical models[2]. The self-balancing problem, however, is a complex mathematical problem involving multiple complications such as air-resistance, friction, and wheel elasticity. Thus, it quickly becomes difficult to use any form of analytical solution to control the Self-Balancing Robot for two reasons:

1. The Arduino based robot does not have enough computation speed to analytically account for the various forces in the physical world and still maintain an adequate response time to keep the robot balanced.
2. The physical world involves stochasticity which analytical solutions cannot account for.

ALTERNATIVE SOLUTION Therefore, this project will opt to employ a control algorithm to solve the self-balancing problem instead. The next section will focus on explaining the PID Control Algorithm, what it does, and why it was chosen to control the robot's balance.

2.2 PID Control Theory

The PID Control algorithm is chosen in this project for two reasons:

1. It is able to control the balance of the Robot at a relatively low computational cost. This is especially essential for two main reasons:
   a) Self-balancing demands a fast response time
   b) The Arduino microcontroller is computationally slow
2. It is easily configurable to fit the build of the Robot.
   a) Allows for quick prototyping.
   b) Self-balancing requires delicate fine tuning.

The PID control algorithm is a form of feedback control processes[3]. The next section will explain the algorithm in detail.

[2] *Eric NeuMann, 2004-2010*
[3] *See Appendix 4.3*

THE PID CONTROL ALGORITHM: INTRODUCTION The PID Control Algorithm used in the engineering of the Self-Balancing Robot is a rather simple but effective control algorithm.

The acronym, PID, stands for Proportional, Integral, and Derivative. As the name suggests, the PID Control Algorithm determines the output by considering error in three forms:

1. Proportional: How large is the current error.
2. Integral: How large is the accumulated error.
3. Derivative: How fast is the error changing.

THE PID CONTROL ALGORITHM: THREE VARIABLE CONTROL According to Control Theory[4], an ideal control algorithm would have an instantaneous rise time and settling time, and have eliminated overshoot and steady-state error altogether. In the physical world, however, this ideal control algorithm is unlikely to exist. Trade-offs have to be made to achieve a not ideal but optimal control solution. The PID Control Algorithm, with its three measures of error, allows its users to make those trade-offs easily by granting different weight to each of its three constants. Before discussing further, lets define the PID Control Algorithm.

$$K_p = \text{Proportional Constant}$$

$$K_i = \text{Integral Constant}$$

$$K_d = \text{Derivative Constant}$$

$$e(t) = \text{Error at a given time}$$

$$\text{Output} = K_p e(t) + K_i \int_0^t e(t)dt + K_d e(t)\frac{d}{dt}$$

It may be observed that each of the three error variables affect the output separately, this allows for users of the PID Control Algorithm to alter the control system's dependency on each error variable without complicating its dependency on other error variables.

Table 1: Table of Control Constants and their effects (adapted from) http://www.chrismarion.net/index.php?option=com_content&view=article&id=122%3athe-segway-theory&catid=44%3arobotics

Control Variable	Rise Time	Overshoot	Settling Time	Stable-State Error
K_p	Decrease	Increase	Small Change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small Change	Decrease	Decrease	No Change

Table 1 summarises the three control constants and their effects on the control system. Now lets discuss each control constant in detail.

[4] *See Appendix 4.4*

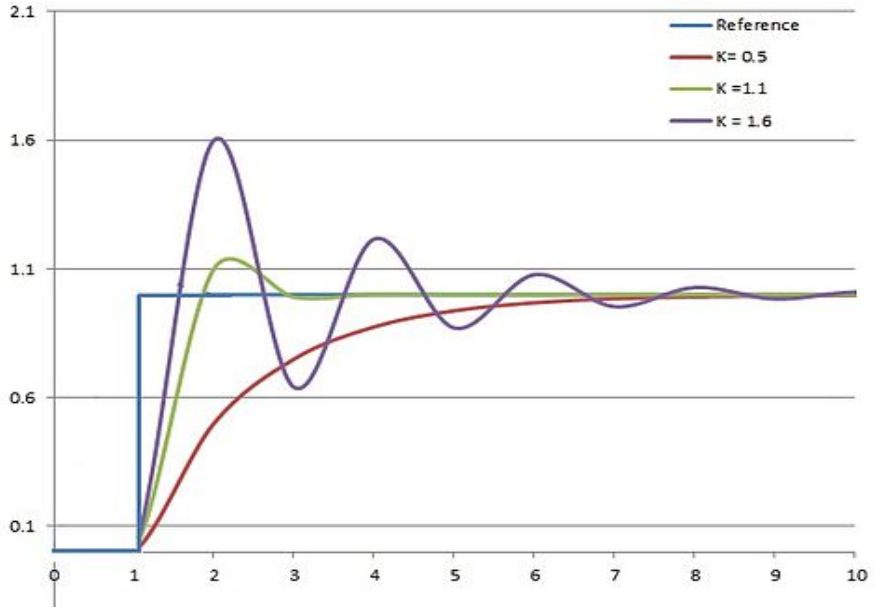


Figure 2: Effect of varying K_p value on output control
from https://en.wikipedia.org/wiki/PID_controller

THE PID CONTROL ALGOTITHM: PROPORTIONAL CONSTANT From figure 2, it may be observed that an increase in K_p value causes two immediately identifiable change. When K_p increases, rise time is drastically reduced and overshoot is drastically increased. No clear relationship may be observed between K_p and the other two variables. It seems that among the three graphs, the settling time is drastically lower when $K_p = 1.1$ suggesting that a critically selected K_p value is vital to the control system overall performance. The $K_p = 1.6$ graph is similar to that of a dampened oscillator graph. This behaviour is consistent with what we would expect from a control system that responds only proportionally to the error.

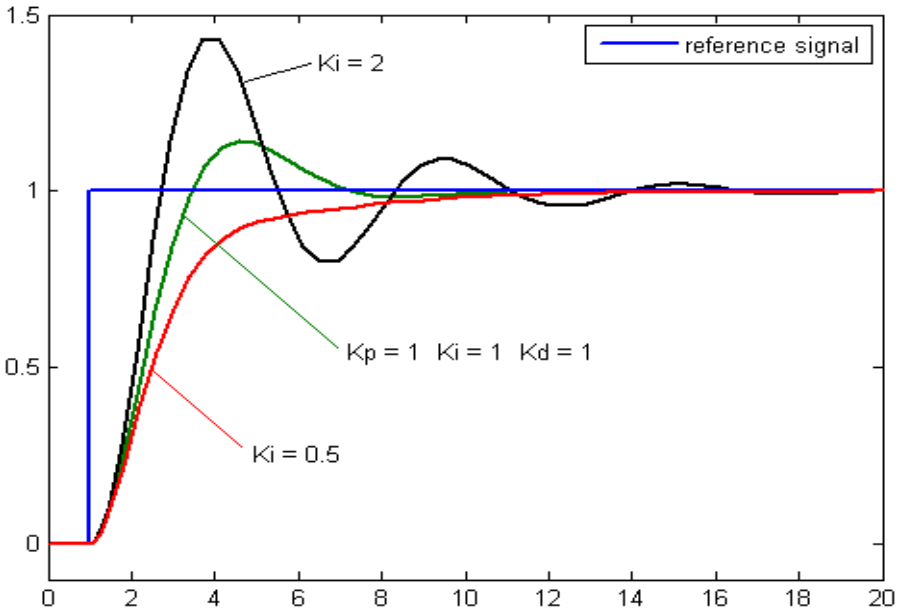


Figure 3: Effect of varying K_i value on output control
from https://en.wikipedia.org/wiki/PID_controller

THE PID CONTROL ALGOTITHM: INTEGRAL CONSTANT Similar to in changing the K_p value, changing the K_i value also seems to affect the rise time and the overshoot in the same way. A large drawback of increasing the K_i value is that it also drastically increases the settling time. However, the $K_i = 0.5$ graph shows that when the K_i value is selected carefully, it is able to significantly reduce the rise time, while keeping the overshoot in check. Also, the integral error of the system accumulates the stable-state error present in the system. The PID Control Algorithm then is able to use integral error to completely eliminate stable-state error.

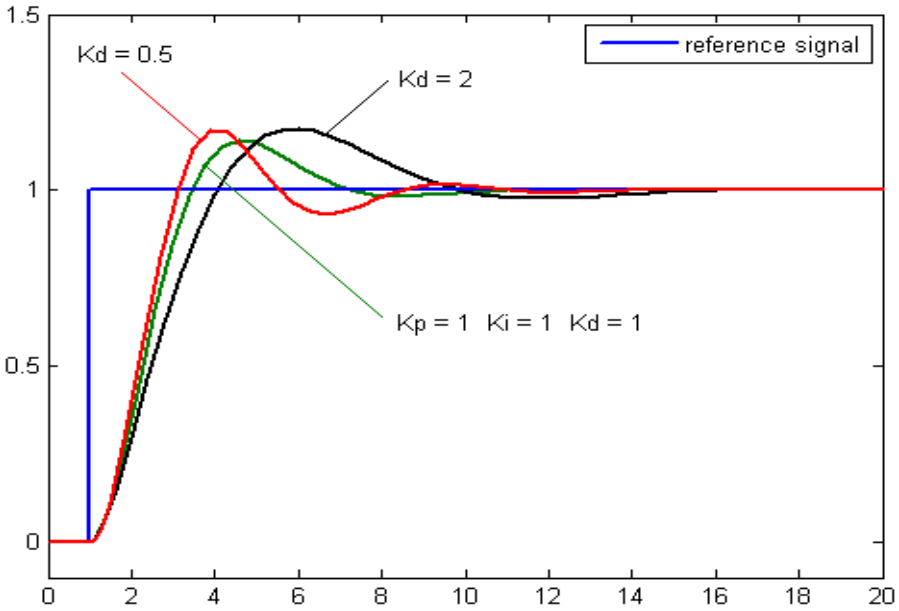


Figure 4: Effect of varying K_d value on output control
from https://en.wikipedia.org/wiki/PID_controller

THE PID CONTROL ALGOTITHM: DERIVATIVE CONSTANT The K_d value, on the other hand, predicts system behaviour and thus improves settling time and stability of the system[8]. Observing figure 10, there seems to be a negative correlation between the K_d value and the steepness of the graph. Though it is not apparent in this three examples, the gradient smoothening effect of the K_d term is likely to improve settling time and stability.

The efficiency of the PID Control Algorithm lies in its simplicity. Each of the three forms of error is simply multiplied by its respective constant and summed together to calculate the output. The computational efficiency of the PID Control Algorithm makes it a good fit for an Arduino based Self-Balancing Robot. Furthermore, it allows for very fine adjustments to be made to the control system, making it a suitable algorithm to solve the delicate Self-Balancing problem.

However, for all the the PID Control Algorithm is worth, it is ultimately reliant on an accurate and precise input. The next section will explore the complementary filter as a method of obtaining reliable angular readings.

[8] *Introduction: PID Controller Design. University of Michigan.*

2.3 The Complementary Filter

As explained in section 2.1, the Self-Balancing Robot's tilt is a measure of its rotation about the y-axis. Hence, for the robot to balance, its pitch must be well-regulated by the complementary filter.

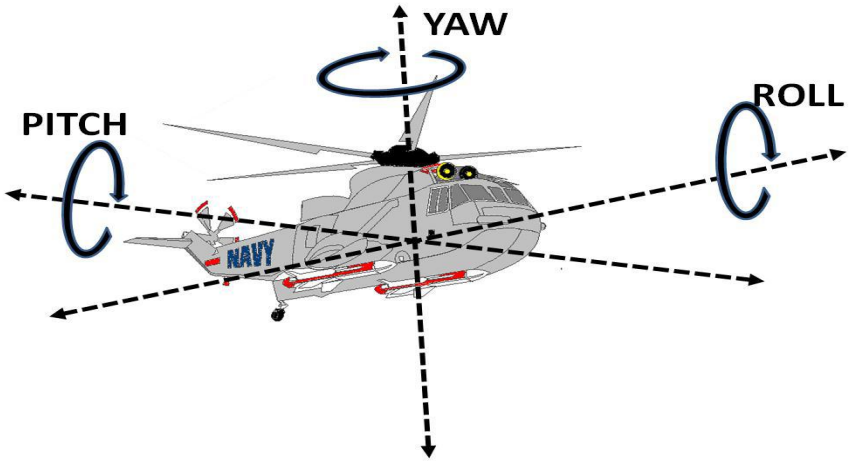


Figure 5: Identifying the Different Axes of rotation
from https://bayesianadventures.wordpress.com/2013/10/20/gyroscopes-accelerometers-and-the-complementary-filter/

ACCELEROMETER The Accelerometer is a device that measures acceleration. In particular, the 3-axis accelerometer used in the Self-Balancing Robot measures acceleration in all three axes. To understand how accelerometers are able to yield information regarding the pitch of the Self-Balancing Robot, it must first be noted that gravity itself is an accelerating force.

Especially in the context of the Self-Balancing Robot, where the acceleration generated by the movement of the robot is negligible when compared to gravitational acceleration, the accelerometer yields highly accurate information about the Robot's pitch. A major drawback of the Accelerometer, however, is that the output which it generates is extremely prone to noise. Thus, owing to its high accuracy and low precision, accelerometer measurements are accurate in the long term but inaccurate in the short term.

GYROSCOPE The Gyroscope, on the other hand, is a device that measures the rate of rotation around each of the Gyroscope's axes. Given a starting pitch, the gyroscope is able to deduce the Robot's current pitch using the following formula:

$$\mathrm{Pitch}_{new} = \mathrm{Pitch}_{old} + \mathrm{gyro}(\mathrm{dt})$$

As compared to the Accelerometer, the Gyroscope's readings are much less susceptible to noise and thus more accurate and precise in the short term. However, as it deduces the pitch of the Self-Balancing Robot through multiple rounds of summation, minute measurement errors build up in the long term. As such, the angular information deduced by the gyroscope is prone to drift in the long term.

THE COMPLEMENTARY FILTER The Complementary Filter capitalises on the Accelerometer's long term accuracy and the Gyroscope's short term accuracy to yield a reliable pitch measurement.

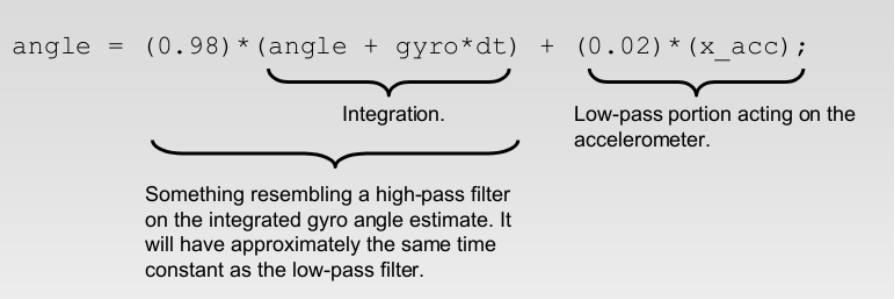


Figure 6: The Complementary Filter equation
from https://bayesianadventures.wordpress.com/2013/10/20/gyroscopes-accelerometers-and-the-complementary-filter/

For the complementary filter to derive an accurate calculation of the Self-Balancing Robot's pitch, it has to favour Gyroscope measurements in the short term and Accelerometer measurements in the long term.

As observed in Figure 6 above, the output angle calculated by the complimentary filter consists of two components: a high-pass portion and a low-pass portion.

The high-pass portion allows short-duration signals to pass through while filtering out signals that are steady over time. This is used to cancel out the drift of the gyroscope.

The low-pass filter, on the other hand,only lets long-term changes through, filtering out short-term fluctuations. it forces the changes to accumulate little by little:

$$\theta_{n+1} = [0.98 * \theta] + [0.02 * \theta_{accel}]$$

By the mechanism described above, the Self-Balancing Robot is able to retrieve accurate information concerning its orientation.

3 ENGINEERING

3.1 Designing the robot

Inverted pendulum dynamics[5] reveal that the pendulum's angular acceleration, $\ddot{\theta}$, is inversely related to its length. As such, the robot was designed to have a high centre of gravity, to minimize angular acceleration.

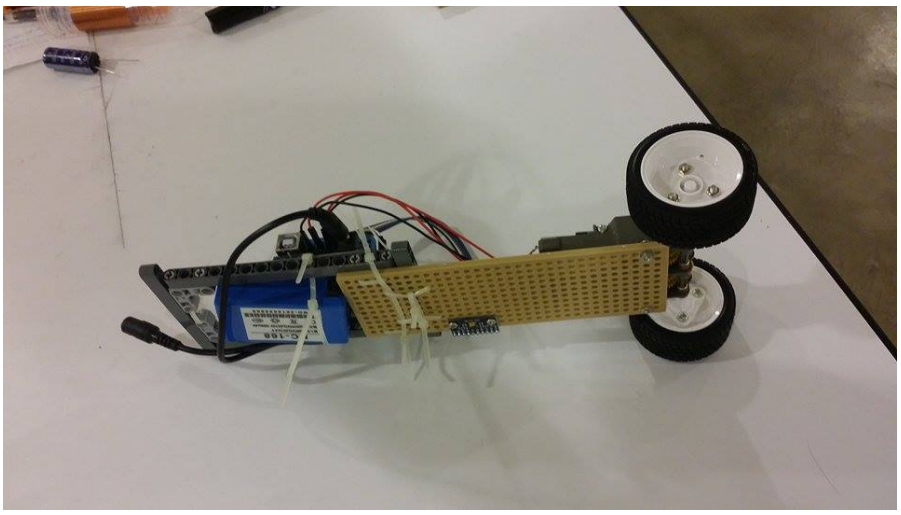


Figure 7: Side view of the Self Balancing Robot

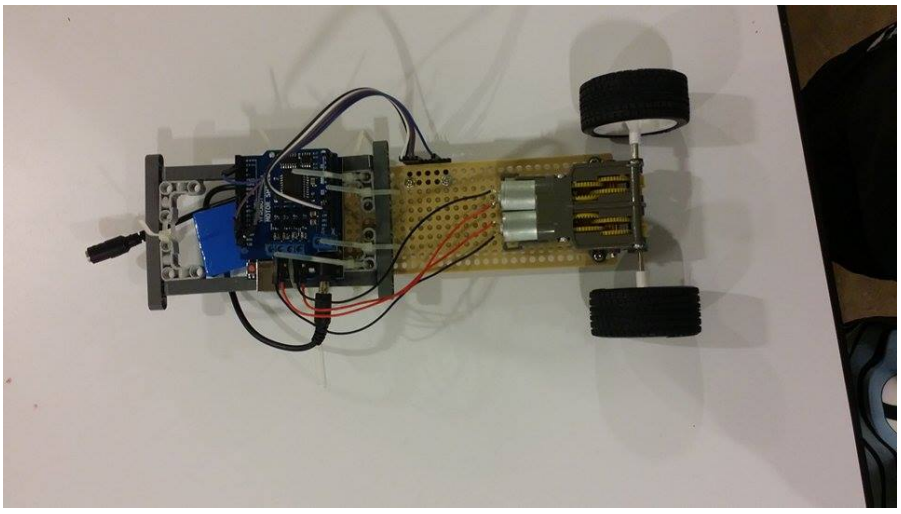


Figure 8: Front view of the Self Balancing Robot

[5] *See Appendix 4.2*

4 APPENDIX

4.1 The Simple Pendulum

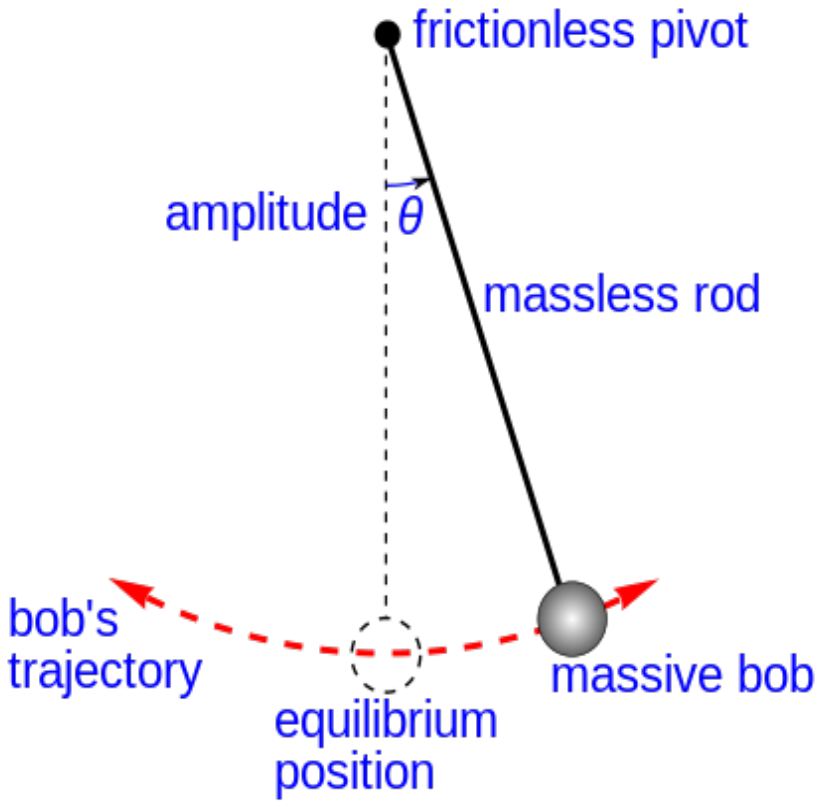


Figure 9: The Classic Pendulum

In a simple pendulum, gravity will always act as a restoring force to push the pendulum's centre of gravity to its equilibrium position. Any small perturbation experienced by the pendulum will always result in gravity accelerating the pendulum back into its original state. As such, the pendulum's equilibrium position is considered to be a stable fixed point. The pendulum's motion is represented by the 2nd order Ordinary Differential Equation (ODE):

$$\ddot{\theta} = -\frac{g}{l}\sin\theta$$

Thus, any angle (θ) occuring in the pendulum would result in an acceleration in the opposite direction, pushing the pendulum back into its equilibrium position.

4.2 The Inverted Pendulum

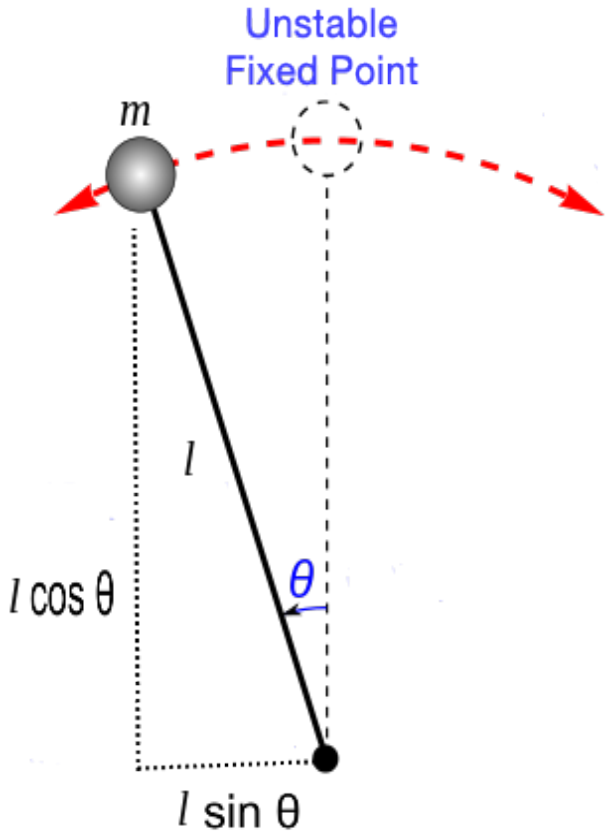


Figure 10: The Inverted Pendulum

In the inverted pendulum on the other hand, the pendulum's centre of mass is located above the axis of rotation. As such, while it too consists of a fixed point when $\theta = 0$, the fixed point is an unstable fixed point. Any small perturbation experienced by the inverted pendulum would result in gravity accelerating the pendulum away from its original fixed point. As a result, the inverted pendulum's original position is considered to be an unstable fixed point. The inverted pendulum's motion is represented by the 2nd order ODE:

$$\ddot{\theta} = \frac{g}{l}\sin\theta$$

Thus, any angle (θ) occuring in the pendulum would result in an acceleration that pushes the pendulum away from its original fixed point.

4.3 Control Process

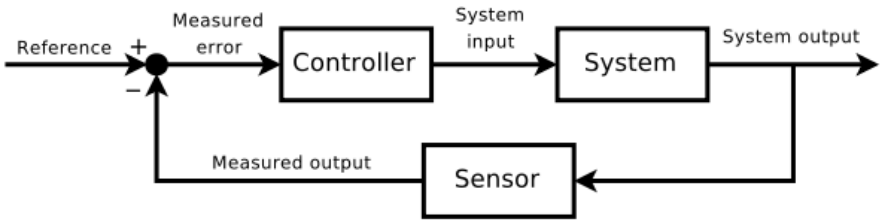


Figure 11: The Negative Feedback Loop

While feedback[2] systems are mostly deterministic, the physical world always involves a degree of stochasticity[3], making analytical models poorly suited to control feedback systems. Thus, control algorithms are often used over analytic solutions to regulate the output of feedback systems involving randomness.

A control process is basically a loop of four basic steps:

1. Setting a reference point
    a) This must be a measurable value that is/is associated with the desired outcome. For instance:
        i. The desired temperature of the room
        ii. The desired upright position of the balance robot.
2. Measuring the error
    a) The deviation from the desired temperature
    b) The deviation from the desired upright position
3. Compensating for the error
    a) If the temperature is too high, turn on the Air-Con
    b) If deviation is detected accelerate the robot
4. Measuring the current state
    a) Measuring the current temperature
    b) Measuring the current angle
5. Repeat from step 2, unless change in reference point required.

Thus the control process involves the consistent monitoring of the state of the system, comparing the actual state against the reference point, and responding to compensate for any deviations from the reference.

All control systems share the same process. However, they deviate in the way by which they compensate for the deviations. This is where the Control Algorithm fits in: it determines how the control system should respond to deviations from the reference point.

4.4 Control Theory

Control Theory is represented by four major principles.

[2] *when the outputs of the system are routed back into the system as inputs, forming a chain of cause and effect*
[3] *involving randomness which may be possible analyse but not precisely determined*

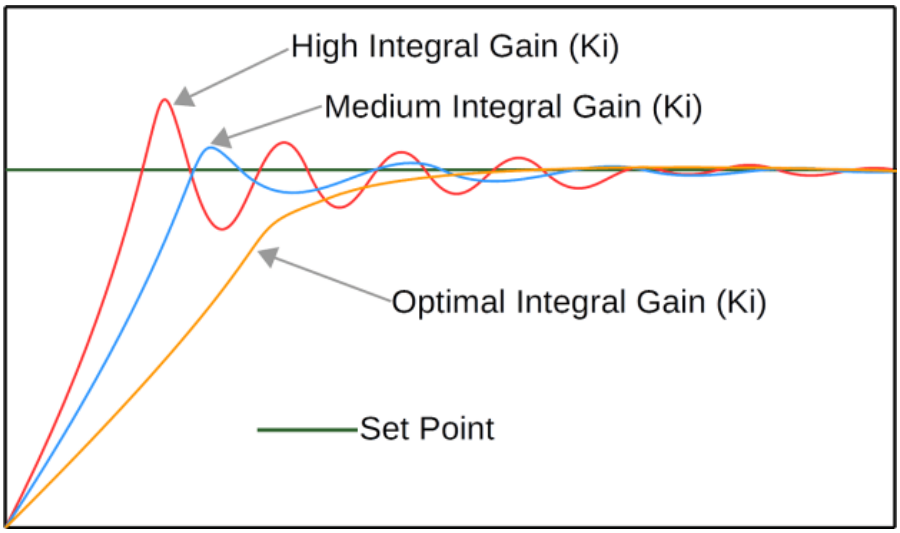


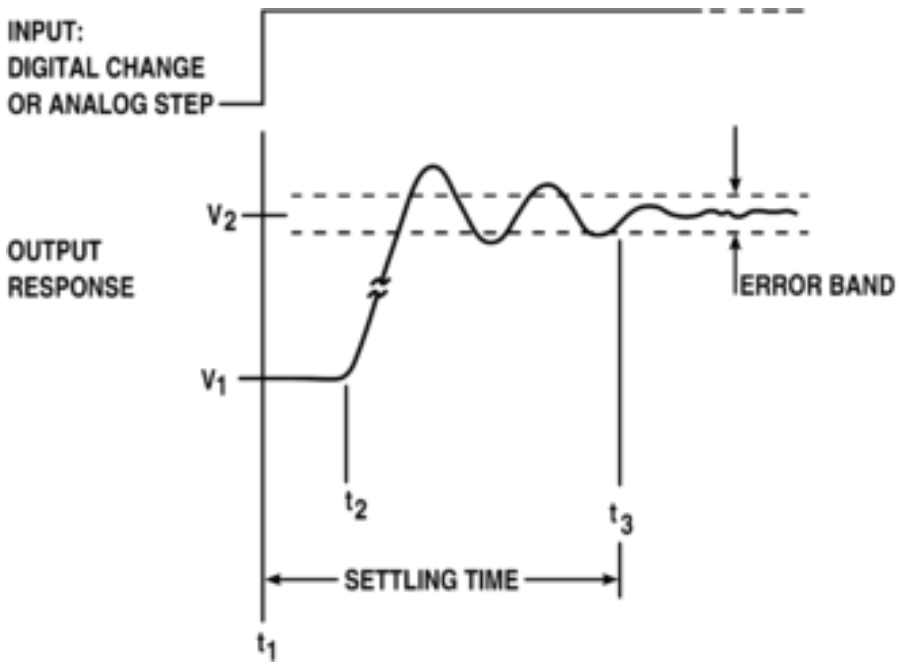
Figure 12: Illustrating Control Theory
from http://coder-tronics.com/pid-tutorial-c-code-example-pt1/

1. Low rise time
2. Low overshoot
3. Low settling time
4. Low steady-state error

The rise time is defined as the amount of time taken for the control system to rise from x% to y% of its final value. For critically damped systems, the 5% to 95% rise time is commonly used[4]. Referring to figure 5, the red graph has the lowest rise time. A low rise time is important because it determines how quickly the control system is able to counteract any error it measures. Specifically pertaining to the Self-Control Robot, the rise time is semantically equivalent to how quickly the Robot is able to eliminate a measured tilt.

The overshoot is defined as the extent by which the output exceeds its final, steady-state value[5]. Referring to figure 5, the red graph also happens to have the highest overshoot. This combination of a low rise time and high overshoot is characteristic of an underdamped system. A low overshoot is important because it prevents the control system from overcompensating for errors in measurement. In the specific context of the Self-Balancing Robot, a high overshoot will result in the robot having to oscillate a few times before settling into its relatively more stable upright position.

[4] *(Levine 2011, p. 9-3 (313)).*
[5] *Kuo, Benjamin C & Golnaraghi M F (2003).*


INPUT:
DIGITAL CHANGE
OR ANALOG STEP
OUTPUT
RESPONSE
V2
V1
ERROR BAND
t2
t3
SETTLING TIME
t1

The settling time is defined as the time required for the response curve to reach and stay within a range of certain percentage (usually 5% or 2%) of the final value[6]. This range is commonly referred to as the error band. Referring to figure 5, the yellow graph has the lowest settling time. Of the four control principles, a low settling time is perhaps the most important indicator of the control system's overall performance. It represents the control system's ability to stabilize at the reference point.For the Self-Balancing Robot, it affects the time taken for the Robot to achieve a relatively stable configuration at its upright position.

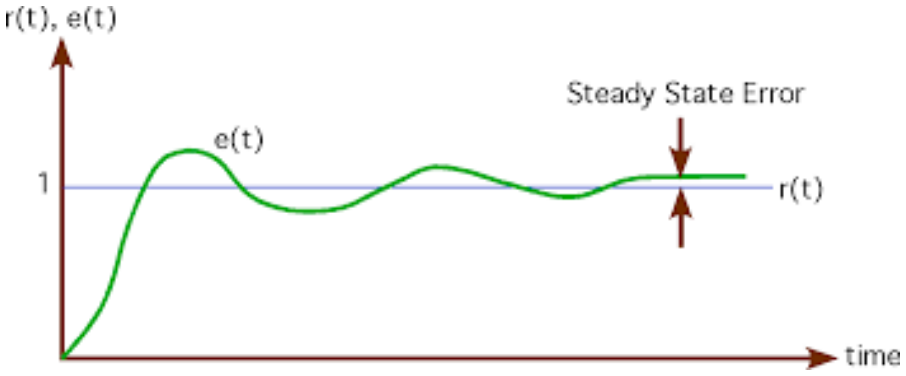


Figure 13: Graph of steady state error

The steady-state error is defined as the difference between the actual output of the system as opposed to its desired output as time approaches infinity[7]. This is represented by the green graph on figure 7 which never converges towards its desired output. The steady-state error represents how accurate the control system is in achieving its desired output. In the case of the Self-Balancing Robot, a steady-state error would cause the robot to favour tilting towards a single side. This would result an accumulating displacement away from the starting point as the robot attempts to keep its balance at a position where it is not stable.

While all four principles are important, they are unequally so. The type of control problem defines the order of priority. An efficient Air-Con for example, would prioritize a low steady-state error: the temperature in a room is not very volatile, thus a low steady-state error would be more important than a low overshoot. For breweries, however, overshoot holds a higher priority. Any overshoot in temperature might kill the yeast used in the fermentation process. For the Self-Balancing Robot, a low rise time is necessary to prevent the tilt from building. $\ddot{\theta} = \frac{g}{l} sin\theta$: As angular acceleration is proportional to tilt, the Robot becomes increasingly difficult to balance when the tilt gets too large. Hence, a low rise time is important.

[6] *Tay, Teng-Tiow; Iven Mareels; John B. Moore (1997). High performance control.*
[7] `http://ctms.engin.umich.edu/CTMS/index.php?aux=Extras_Ess`

5 BIBLIOGRAPHY

Control Tutorials for MATLAB and Simulink -. (n.d.). Retrieved December 3, 2015, from http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction§ion=ControlPID

Improving the Beginner's PID – Introduction. (n.d.). Retrieved December 3, 2015, from http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/

The Segway Theory. (2015, December 3). Retrieved December 3, 2015, from http://www.chrismarion.net/index.php

Gyroscopes, Accelerometers and the Complementary Filter. (2013, October 20). Retrieved December 3, 2015, from https://bayesianadventures.wordpress.com/2013/10/20/gyr accelerometers-and-the-complementary-filter/