TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
CENTRAL CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

A
FINAL YEAR PROJECT REPORT
ON
WIRELESS CONTROL MOBILE INVERTED PENDULUM

BY
Ashish Devkota (Exam Roll No. : 69456)
Bishes Adhikari (Exam Roll No. : 69464)
Buddha Raj Shrestha (Exam Roll No. : 69465)
Dipendra P. Rauniyar (Exam Roll No. : 69468)

December, 2014
TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
CENTRAL CAMPUS

WIRELESS CONTROL MOBILE INVERTED PENDULUM

By:
Ashish Devkota (Exam Roll No. : 69456)
Bishes Adhikari (Exam Roll No. : 69464)
Buddha Raj Shrestha (Exam Roll No. : 69465)
Dipendra P. Rauniyar (Exam Roll No. : 69468)

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND
COMPUTER ENGINEERING IN PARTIAL FULLFILLMENT OF THE
REQUIREMENT FOR THE BACHELOR’S DEGREE IN ELECTRONICS &
COMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
LALITPUR, NEPAL

December, 2014
LETTER OF APPROVAL

The undersigned hereby certify that they have read, and recommended to the Institute of Engineering for acceptance, a project report entitled "Wireless Control Mobile Inverted Pendulum" submitted by Ashish Devkota, Bishes Adhikari, Dipendra P. Rauniyar and Buddha Raj Shrestha in partial fulfilment of the requirements for the Bachelor’s degree in Electronics & Communication Engineering.

Supervisor
Mr. Jitendra Kumar Manandhar
Assistant Campus Chief
Central Campus

Internal Examiner
Prof. Dr. Shashidhar Ram Joshi
Department of Electronics and Computer Engineering,
Institute of Engineering, Central Campus,
Tribhuvan University, Nepal

External Examiner
Mr. Om Bikram Thapa
CTO
Vianet Communication

Dr. Nanda Bikram Adhikari
Deputy Head
Department of Electronics and Computer Engineering,
Institute of Engineering, Central Campus,
Tribhuvan University, Nepal

Dr. Dibakar Raj Pant
Head
Department of Electronics and Computer Engineering,
Institute of Engineering, Central Campus,
Tribhuvan University, Nepal

DATE OF APPROVAL:
COPYRIGHT

The author has agreed that the Library, Department of Electronics and Computer Engineering, Central Campus, Institute of Engineering may make this report freely available for inspection. Moreover, the author has agreed that permission for extensive copying of this project report for scholarly purpose may be granted by the supervisors who supervised the project work recorded herein or, in their absence, by the Head of the Department wherein the project report was done. It is understood that the recognition will be given to the author of this report and to the Department of Electronics and Computer Engineering, Central Campus, Institute of Engineering in any use of the material of this project report. Copying or publication or the other use of this report for financial gain without approval of to the Department of Electronics and Computer Engineering, Central Campus, Institute of Engineering and author’s written permission is prohibited.

Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

__________________________________________
Head
Department of Electronics and Computer Engineering
Central Campus, Institute of Engineering
Lalitpur, Kathmandu
Nepal
ACKNOWLEDGEMENT

It is a pleasure to thank all the people who have supported us throughout the project. Firstly, we thank the Department of Electronics and Computer Engineering, Central Campus, IOE for providing us with an opportunity to be involved in something that helps us not only in increasing our knowledge, but also in being involved in a team project. We owe our deepest gratitude to Mr. Jitendra Kumar Manandhar, our project supervisor for all his interest and suggestions for the project.

Thanks are due to our senior Rakesh Shrestha for his continuous support and information on the project. We are deeply indebted to our junior Anil Kamat for his help in the mechanical construction of the project. We should also like to thank Mr. Ramesh Chaudhary, Instructor of the Robotics Club, Central Campus for his guidance and all the other members of the Robotics Club, Central Campus for their help.

We are grateful to Mr. Dinesh Baniya Kshetri for his interest in our project. His suggestions have been much appreciated.
ABSTRACT

An inverted pendulum is a pendulum upside down. An example of pendulum is the pendulum of a clock and of the inverted pendulum is balancing a stick on a finger. The inverted pendulum is one of the most popular problems in control theory. The reason for its fame is that it is a readymade problem and thus a very good teaching topic. The second reason is its application. Its use in balancing humanoid robots, fixing the vertical orientation of the rockets before take-off among many others are the reasons it is important.

To model an inverted pendulum, we have two wheels balancing the base that they carry. The two wheels have the same axis of rotation. These wheels are connected to motors which drive the wheels and the motors are in turn controlled by the main control system; the main component of the system. The control system moves/ drives the motor in the direction necessary to balance the system. Sensors are added to know whether the system is balanced or not. If it isn’t, what is the direction of its tilt is known. However, the output from the sensors is not free from noise. Such noise are filtered out and then input to the controller.

Another component of the project is the wireless data communication. This is used to collect the information about the orientation of the system. This information is plotted in real time. Further, a Graphical User Interface is present in which the model mimics the real system and its orientation.

Thus, ‘Wireless Control Mobile Inverted Pendulum’ attempts to construct a self-balancing two-wheeled inverted pendulum and its real time data plotting along with its simulation in a computer GUI application by implementing wireless communication.
TABLE OF CONTENTS

LETTER OF APPROVAL ........................................................................................................... ii
COPYRIGHT ............................................................................................................................... iii
ACKNOWLEDGEMENT ............................................................................................................... iv
ABSTRACT ................................................................................................................................. v
TABLE OF CONTENTS .............................................................................................................. vi
LIST OF FIGURES .................................................................................................................... ix
LIST OF TABLES ....................................................................................................................... xi
LIST OF ABBREVIATIONS ....................................................................................................... xii
LIST OF SYMBOLS .................................................................................................................. xiv

1. INTRODUCTION .................................................................................................................. 1
   1.1. Background ....................................................................................................................... 1
   1.2. Problem Statement ............................................................................................................ 2
   1.3. Objectives ........................................................................................................................ 4
   1.4. Scope Of Work ................................................................................................................... 5

2. CHAPTER ORGANIZATION AND BACKGROUND THEORY ................................................. 6
   2.1. Understanding Inverted Pendulum .................................................................................. 6
       2.1.1. Problem Statement ...................................................................................................... 7
       2.1.2. PID Control ................................................................................................................ 13
   2.2. Accelerometer and Gyroscope ........................................................................................ 16
   2.3. Complementary Filter ...................................................................................................... 16
   2.4. The Kalman Filter ........................................................................................................... 18
       2.4.1. Modelling a system ..................................................................................................... 20
       2.4.2. Algorithm .................................................................................................................... 23
   2.5. Voltage Level Shifter Circuit .......................................................................................... 26
2.6. Control System and PID Controller ................................................................. 27
2.7. Motor Driver and H-bridge ........................................................................... 29
2.9. Wireless Data Transfer .................................................................................. 33

3. LITERATURE REVIEW .................................................................................. 34
3.1. Complementary Filter ................................................................................... 34
3.2. Kalman Filter ................................................................................................. 35
3.3. PID Controller ............................................................................................... 36
3.4. XBee ............................................................................................................... 37

4. SOFTWARES AND TOOLS USED ................................................................. 39
4.1. For Simulation ............................................................................................... 39
   4.1.1. MATLAB® ............................................................................................ 39
   4.1.2. Python IDLE and Extensions: Pyserial, Pygame, matplotlib, Numpy .... 39
   4.1.3. Code::Blocks ....................................................................................... 39
4.2. Microcontroller Tools ................................................................................... 40
   4.2.1. Arduino Mega 2560 ............................................................................. 40
   4.2.2. XBEE2 ................................................................................................. 41
   4.2.3. GY-81 IMU ......................................................................................... 43
4.3. Design Tools ................................................................................................. 47
   4.3.1. PCB Wizard ......................................................................................... 47
   4.3.2. KiCad ................................................................................................... 47
   4.3.2. SolidWorks 2013 ................................................................................ 47

5. METHODOLOGY, SYSTEM DESIGN AND IMPLEMENTATION .................. 48
5.1. Mechanical Construction .............................................................................. 48
   5.1.1. Custom Configurable and Manufacture Parts ....................................... 49
   5.1.2. Components and Materials .................................................................. 49
5.2. Sensors .......................................................................................................... 50
   5.2.1. Accelerometer ...................................................................................... 50
5.2.2. Gyroscope .................................................................................................................. 51
5.2.3. Combining the accelerometer and gyroscope data: Filtering .......................... 51
5.3. Simulink model of the Inverted Pendulum ................................................................. 54
5.4. Processor ....................................................................................................................... 56
5.5. Power Supply ................................................................................................................. 56
5.6. Circuit Design and Construction .................................................................................. 57
  5.6.1. Voltage Shifter Circuit ............................................................................................. 57
  5.6.2. Motor Control and H-bridge Circuit ...................................................................... 58
5.7. Permanent magnet DC motor ....................................................................................... 60
  5.7.1. Dead-band ................................................................................................................ 60
5.8. Wireless Data Transfer using XBee ............................................................................... 61
5.9. Graphical User Interface and Live Graph Plotting ......................................................... 62
5.10. System Block Diagram ................................................................................................. 66
6. RESULT .......................................................................................................................... 67
  6.1. Mechanical construction: ............................................................................................ 67
  6.2. Sensor Output ............................................................................................................... 68
  6.3. Simulink Output .......................................................................................................... 71
  6.4 Deadband and RPM .................................................................................................... 71
  6.5. GUI Interface and Live Graph Plotting ...................................................................... 73
7. COST ESTIMATION ......................................................................................................... 74
8. CONCLUSION AND RECOMMENDATION .................................................................... 75
BIBLIOGRAPHY ..................................................................................................................... 76
APPENDIX A. CIRCUITS ........................................................................................................ 80
INDEX ..................................................................................................................................... 82
LIST OF FIGURES

Figure 1.1 Forces acting on an inverted pendulum .......................................................... 1
Figure 2.1 Free body diagram of an inverted pendulum .................................................. 7
Figure 2.2 Object with center of mass at P rotating about O in the plane xy ................... 8
Figure 2.3 Open Loop Impulse Responses ..................................................................... 12
Figure 2.4 Open Loop Responses for the Step Input ...................................................... 13
Figure 2.5 Simple Controller Model .............................................................................. 14
Figure 2.6 Controller in Feedback .................................................................................. 14
Figure 2.7 PID Response for unit kp, ki, kd .................................................................. 15
Figure 2.8 PID response after changing to kp=100 ...................................................... 15
Figure 2.9 PID response for kp=100, ki=1, kd=20 ...................................................... 16
Figure 2.10 A simple RC lowpass filter ......................................................................... 17
Figure 2.11 Red plot shows the modeled system for MIP and black plot shows the measurement value plotted against time (MATLAB) ............................................... 25
Figure 2.12 Green Plot shows the Kalman estimation of the above plotted against time ... 26
Figure 2.13 Voltage Level Shifter .................................................................................. 27
Figure 2.14 Error Signal, e(t) vs time, t ........................................................................ 28
Figure 2.15 Control Signal, m(t) vs time, t for a proportional controller ...................... 28
Figure 2.16 Control Signal, m(t) vs time, t for an integral controller ............................ 28
Figure 2.17 Control Signal, m(t) vs time, t for a derivative controller ......................... 29
Figure 2.18 Control Signal, m(t) vs time, t for a PID controller ................................... 29
Figure 2.19 Operation of an H-bridge ......................................................................... 30
Figure 2.20 Condition for forward spinning and reverse spinning of the motor .......... 31
Figure 2.21 Short through condition ............................................................................. 31
Figure 2.22 PWM showing duty cycle .......................................................................... 32
Figure 4.1 Arduino Mega 2560 .................................................................................... 40
Figure 4.2 XBee transceiver ......................................................................................... 42
Figure 4.3 Top and bottom view of GY-81 IMU ............................................................. 43
Figure 4.4 BMA180 Accelerometer Pin Description ...................................................... 44
Figure 4.5 ITG-3205 Gyroscope Pin Description .......................................................... 45
Figure 5.1 Cross-section and two dimensional Wireframe View of the design ............. 48
LIST OF TABLES

Table 2.1 Operation of H-bridge.................................................................31
Table 4.1 Arduino Mega 2560 specifications..............................................41
Table 4.2 BMA180 Accelerometer Pin Description......................................44
Table 4.3 ITG-3205 Gyroscope Pin Description........................................46
Table 5.1 Major components in our H-bridge ............................................59
Table 7.1 Cost Estimation...........................................................................74
### LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>DOF</td>
<td>Degree of Freedom</td>
</tr>
<tr>
<td>2.</td>
<td>AC</td>
<td>Air Conditioner</td>
</tr>
<tr>
<td>3.</td>
<td>PID</td>
<td>Proportional Integral Derivative</td>
</tr>
<tr>
<td>4.</td>
<td>IMU</td>
<td>Inertial Measurement Unit</td>
</tr>
<tr>
<td>5.</td>
<td>CG</td>
<td>Centre of Gravity</td>
</tr>
<tr>
<td>6.</td>
<td>VTOL</td>
<td>Vertical Take Off Landing</td>
</tr>
<tr>
<td>7.</td>
<td>MOSFET</td>
<td>Metal Oxide Semiconductor Field Effect Transistor</td>
</tr>
<tr>
<td>8.</td>
<td>I/O</td>
<td>Input/output</td>
</tr>
<tr>
<td>9.</td>
<td>AC</td>
<td>Alternating Current</td>
</tr>
<tr>
<td>10.</td>
<td>EM</td>
<td>Electromagnetic</td>
</tr>
<tr>
<td>11.</td>
<td>kHz</td>
<td>Kilohertz</td>
</tr>
<tr>
<td>12.</td>
<td>MHz</td>
<td>Megahertz</td>
</tr>
<tr>
<td>13.</td>
<td>GHz</td>
<td>Gigahertz</td>
</tr>
<tr>
<td>14.</td>
<td>UART</td>
<td>Universal Asynchronous Receiver Transmitter</td>
</tr>
<tr>
<td>15.</td>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>16.</td>
<td>I²C</td>
<td>Inter-Integrated Circuit</td>
</tr>
<tr>
<td>17.</td>
<td>RFM</td>
<td>Radio Frequency Module</td>
</tr>
<tr>
<td>18.</td>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>19.</td>
<td>IDLE</td>
<td>Integrated DeveLopment Environment</td>
</tr>
<tr>
<td>20.</td>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>21.</td>
<td>ICSP</td>
<td>In-circuit Serial Programming</td>
</tr>
<tr>
<td>22.</td>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>23.</td>
<td>MIP</td>
<td>Mobile Inverted Pendulum</td>
</tr>
<tr>
<td>24.</td>
<td>PAN</td>
<td>Personal Area Network</td>
</tr>
<tr>
<td>25.</td>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>26.</td>
<td>EDA</td>
<td>Electronic Design Automation</td>
</tr>
<tr>
<td>27.</td>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>28.</td>
<td>GNU</td>
<td>GNU’s Not Unix</td>
</tr>
<tr>
<td>S. No.</td>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>29.</td>
<td>GPL</td>
<td>General Public License</td>
</tr>
<tr>
<td>30.</td>
<td>LiPo</td>
<td>Lithium Polymer</td>
</tr>
<tr>
<td>31.</td>
<td>SDA</td>
<td>Serial Data Line</td>
</tr>
<tr>
<td>32.</td>
<td>SCL</td>
<td>Serial Clock Line</td>
</tr>
<tr>
<td>33.</td>
<td>VDD</td>
<td>Voltage Drain to Drain</td>
</tr>
<tr>
<td>34.</td>
<td>IGBT</td>
<td>Insulated Gate Bipolar Transistor</td>
</tr>
<tr>
<td>35.</td>
<td>ID</td>
<td>Identifier</td>
</tr>
<tr>
<td>36.</td>
<td>ADC</td>
<td>Analog to Digital Converter</td>
</tr>
<tr>
<td>37.</td>
<td>RPM</td>
<td>Round Per Minute</td>
</tr>
<tr>
<td>38.</td>
<td>RGB</td>
<td>Red-Green-Blue</td>
</tr>
<tr>
<td>39.</td>
<td>COM-PORT</td>
<td>Communication port</td>
</tr>
</tbody>
</table>
# LIST OF SYMBOLS

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Symbols</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$\theta_p$</td>
<td>Tilt angle</td>
</tr>
<tr>
<td>2.</td>
<td>$\Delta$</td>
<td>Angle in the XZ plane measured from x-axis</td>
</tr>
<tr>
<td>3.</td>
<td>xRM</td>
<td>Position vector in the direction of positive x-axis</td>
</tr>
<tr>
<td>4.</td>
<td>$F$</td>
<td>Force applied to the cart</td>
</tr>
<tr>
<td>5.</td>
<td>$N$</td>
<td>Horizontal component of the reaction force at the hinge or pivot of the pendulum</td>
</tr>
<tr>
<td>6.</td>
<td>$P$</td>
<td>Normal component of the reaction force at the hinge or pivot of the pendulum</td>
</tr>
<tr>
<td>7.</td>
<td>$M$</td>
<td>Mass of the cart</td>
</tr>
<tr>
<td>8.</td>
<td>$\dot{\theta}$</td>
<td>First order derivative of $\theta$</td>
</tr>
<tr>
<td>9.</td>
<td>$\ddot{\theta}$</td>
<td>Second order derivative of $\theta$</td>
</tr>
<tr>
<td>10.</td>
<td>$\Omega$</td>
<td>Angular velocity</td>
</tr>
<tr>
<td>11.</td>
<td>$K$</td>
<td>Unit vector along the axis of rotation</td>
</tr>
<tr>
<td>12.</td>
<td>R</td>
<td>Position vector</td>
</tr>
<tr>
<td>13.</td>
<td>A</td>
<td>Acceleration</td>
</tr>
<tr>
<td>14.</td>
<td>$A$</td>
<td>Angular acceleration</td>
</tr>
<tr>
<td>15.</td>
<td>$a_t$</td>
<td>Tangential component of acceleration</td>
</tr>
<tr>
<td>16.</td>
<td>$a_n$</td>
<td>Normal component of acceleration</td>
</tr>
<tr>
<td>17.</td>
<td>$M$</td>
<td>Mass of the pendulum</td>
</tr>
<tr>
<td>18.</td>
<td>$L$</td>
<td>Distance of the center of mass of the pendulum from the pivot</td>
</tr>
<tr>
<td>19.</td>
<td>$B$</td>
<td>Co-efficient of friction</td>
</tr>
<tr>
<td>20.</td>
<td>$G$</td>
<td>Acceleration due to gravity</td>
</tr>
<tr>
<td>21.</td>
<td>$\phi$</td>
<td>Deviation of the pendulum from vertical, same as $\theta_p$</td>
</tr>
<tr>
<td>22.</td>
<td>$U$</td>
<td>Control force applied to the cart, same as $F$</td>
</tr>
<tr>
<td>23.</td>
<td>Alpha</td>
<td>Tuning parameter for the gain of Complementary filter</td>
</tr>
<tr>
<td>24.</td>
<td>gyroAngle</td>
<td>Angular rate obtained from gyroscope</td>
</tr>
<tr>
<td>25.</td>
<td>accAngle</td>
<td>Angle obtained from the accelerometer after calculation</td>
</tr>
<tr>
<td>26.</td>
<td>Angle</td>
<td>Output obtained from the Complementary filter</td>
</tr>
<tr>
<td>S. No.</td>
<td>Symbols</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>27.</td>
<td>$dt$</td>
<td>Sample time interval</td>
</tr>
<tr>
<td>28.</td>
<td>$V_{in}(t)$</td>
<td>Input voltage as a continuous function of time</td>
</tr>
<tr>
<td>29.</td>
<td>$V_{out}(t)$</td>
<td>Output voltage as a continuous function of time</td>
</tr>
<tr>
<td>30.</td>
<td>$R$</td>
<td>Resistance, Variance of measurement noise (Kalman Filter)</td>
</tr>
<tr>
<td>31.</td>
<td>$i(t)$</td>
<td>Input current as a continuous function of time</td>
</tr>
<tr>
<td>32.</td>
<td>$Q_C(t)$</td>
<td>Charge stored in the capacitor</td>
</tr>
<tr>
<td>33.</td>
<td>$C$</td>
<td>Capacitance</td>
</tr>
<tr>
<td>34.</td>
<td>$P(A</td>
<td>B)$</td>
</tr>
<tr>
<td>35.</td>
<td>$P(A)$</td>
<td>Probability of event A</td>
</tr>
<tr>
<td>36.</td>
<td>$F/F_t$</td>
<td>A state transition matrix applied to the (known) previous state $X_{t-1}$</td>
</tr>
<tr>
<td>37.</td>
<td>$X/X_t$</td>
<td>A state vector/ State vector at time $t$</td>
</tr>
<tr>
<td>38.</td>
<td>$B/B_t$</td>
<td>A control matrix $B_t$ which is applied to a control vector</td>
</tr>
<tr>
<td>39.</td>
<td>$U/U_t$</td>
<td>Control vector/ At time $t$</td>
</tr>
<tr>
<td>40.</td>
<td>$W/W_t$</td>
<td>Process noise vector on state prediction/ At time $t$</td>
</tr>
<tr>
<td>41.</td>
<td>$Y/Y_t$</td>
<td>An observation vector/ At time $t$</td>
</tr>
<tr>
<td>42.</td>
<td>$H/H_t$</td>
<td>An observation vector/ At time $t$</td>
</tr>
<tr>
<td>43.</td>
<td>$V_t$</td>
<td>Observation or measurement noise/ At time $t$</td>
</tr>
<tr>
<td>44.</td>
<td>$Q$</td>
<td>Variance of the process noise</td>
</tr>
<tr>
<td>45.</td>
<td>$K_t$</td>
<td>Optimal Kalman Gain or just Kalman Gain/ At time $t$</td>
</tr>
<tr>
<td>46.</td>
<td>$K_p$</td>
<td>Constant for proportional control</td>
</tr>
<tr>
<td>47.</td>
<td>$K_i$</td>
<td>Constant for integral control</td>
</tr>
<tr>
<td>48.</td>
<td>$K_d$</td>
<td>Constant for derivative control</td>
</tr>
<tr>
<td>49.</td>
<td>$e(t)$</td>
<td>Error signal</td>
</tr>
<tr>
<td>50.</td>
<td>$T(on)$</td>
<td>On-time</td>
</tr>
<tr>
<td>51.</td>
<td>$T(off)$</td>
<td>Off-time</td>
</tr>
<tr>
<td>52.</td>
<td>$Dbf$</td>
<td>Forward dead-band</td>
</tr>
<tr>
<td>53.</td>
<td>$Dbr$</td>
<td>Reverse dead-band</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

1.1. Background

In simple words, an inverted pendulum can be explained with the analogy of balancing a long stick on a finger, something that all of us have tried in childhood. [1]

Much work has been done on the inverted pendulum for the last 50 years.

Above figure shows a mobile inverted pendulum, JOE [2], with 3 degrees of freedom - motion shown by $\theta_p$, $\delta$ and xRM. It is able to rotate around the z-axis (pitch), a movement described by the angle $\theta_p$ and the corresponding angular velocity. The linear movement of the chassis is described by the position xRM and the speed. Additionally, the vehicle can rotate around its vertical axis (yaw) with the associated angle $\delta$ and angular velocity. These six state variables fully describe the dynamics of the 3 DOF system. We have to balance...
the freedom along $\theta$ so that the vertical orientation of the two wheeler is maintained by moving it along xRM.

The six variables stated above are enough to describe the equations defining the dynamic model of the system. These equations may be derived from the free body diagram of the model of two wheels separately and of chassis separately, thus giving rise to three equations. These equations can be combined into one in a state space model thus giving rise to a dynamic equation [2]. Though when we derive this dynamic equation, we will not separate the wheels and the chassis. The derivation of the system equation is shown in section ‘2.1. Understanding Inverted Pendulum’.

The next step is finding appropriate sensors to integrate to measure the system’s state. This step is called state measurement. Then, developing a control algorithm or a control system along with the orientation projection will complete the system for the project.

Of course, the tests will be done and results will be simulated before the hardware development is done when all of the processes are complete to develop the system.

1.2. Problem Statement

The word ‘control’ in itself is a broad term and projects related to control theory are always useful. In today’s world where everything is moving towards automation (take the example of Google driverless car, temperature maintained room), the first step that comes into the implementation of anything automatic is the control of its constituent variables.

There are mainly two variables we need to consider when we are talking about a control system. They are ‘Set-point’ and ‘Feedback’ of the system. Take a simple example of a temperature controlled room with an Air Conditioner (AC). Its exhaust fan should maintain the temperature of the room (normally $25^0$C). $25^0$C is thus the ‘set-point’ and the temperature sensor’s output will be our ‘feedback’. Now, using the feedback from the sensor, we can increase or decrease the exhaust fan’s speed according to the surrounding temperature and maintain a near-about $25^0$C. This is surely an example of a fairly simple automatic control system. As we go higher in search of making fully automatic systems,
the problems related to control become tougher and hence require complex algorithms to solve.

Let us look at another example. Suppose, we want to control the speed of a motor, taking feedback from optical encoders. We cannot solve the problem as easily as we did in our previous example. In such systems, the problem of oscillation might occur in the speed of the motor as the Feedback speed might sometimes be higher or sometimes lower than the Set-point (imagine a Set-point of 50mph and the system is continuously oscillating from 48mph to 52 mph i.e. the system is unable to track the Set-point properly), which is not tolerable as we desire to control the motor speed instantaneously.

Focusing on the inverted pendulum, it is a popular problem in control theory. As we explained the concept of set-point and feedback in our previous examples, the set-point will be the angle (90° with respect to ground) and the feedback is from the sensors that give the orientation of the pendulum. Comparing with the balancing stick problem, it is quite difficult to balance a stick vertically with a single finger, where our brain takes the feedback from our eyes. Trying to implement this concept onto a system is relatively a difficult task when compared to our previous examples.

There is no exact mathematical relation that can balance the inverted pendulum (even though there might be some approximations on what the overall system equation might be). There only exists a general idea of balancing the system i.e. giving reaction against the direction where the system is tilted towards, same as in the case of balancing stick.

Since there are very few practically implementable control algorithms, there arises a huge problem in order to make an inverted pendulum. The most common of the control algorithms is the PID controller which might not provide a satisfactory result under every circumstance. The controlling requires a lot of time to manually tune its parameters; the proportional, the integral and the differential term and is the biggest problem.

Another issue is filtering the output from sensors. An IMU consisting of an accelerometer and a gyroscope among others has been used. The accelerometers data is very noisy in presence of vibrations due to the motors. Hence using the data of only the accelerometer
could cause abnormal behaviour to the system as the feedback taken by the system fluctuates randomly and with large amplitude. The problem with using only the gyroscope is the dreaded gyro drift. Even if the drift has a small value, it adds up over time. Also another problem with using only gyroscope is the lack of initial tilt as it measures the angular velocity and not the angle itself. Hence, filtering must be done to get the exact tilt of the system. Some commonly implemented filters like Complementary filter, Kalman filter, etc. are implemented. Tuning the constants involved in filters in-order to reduce the error is the main problem with sensors.

Another important part is the mechanical construction of the system. The mechanical construction should be precise. The CG (Center of Gravity) of the system needs to be in the exact center of the base of the system. If it is not so, then it will be very difficult to control the system as the weight will not be properly balanced on either side and the system will tend to incline in one direction. Also, the coupling of the plates from the body of the system to the motors needs to exact. Lack of such application will cause vibrations in the system amounting to noisy sensor output from the IMU.

1.3. Objectives

The objectives of this project can be summarised as follows:

1. Working model: The robot should be able to balance itself on two wheels without any external attachments i.e. sustain a vertical orientation.
2. Well designed accurate sensors (noise elimination): The data from various sensors need to be filtered out because of either the sensors are affected by noisy forces (e.g. movement contributing to force components that are used to calculate the tilt angle) and the errors in the sensors itself (e.g. gyro bias)
3. Wireless data communication: The data (e.g. tilt angle) can be transferred to the receiver which can be used to plot the configuration of the system. Basically, these data are used to define the system state.
1.4. Scope Of Work

The inverted pendulum serves as the fundamental benchmark in control theory.

“Since the 1950s, the inverted pendulum benchmark, especially the cart version, was used for teaching linear feedback control theory to stabilize open-loop unstable systems. The first solution to this problem was described by Roberge in 1960 and then by Schaefer, and Cannon in 1966. This benchmark was considered in many references as a typical root-locus analysis example. Subsequently, it has been also used in many books to solve the linear optimal control problem and the complex nonlinear control problem for unstable systems.” [3]

The inverted pendulum is applied in the area of precision control and robotics, such as [1]

- High precision robotic arms
- Launching of a rocket
- Control of a Vertical Take-Off and Land (VTOL) aircraft.

Due to the advent of humanoid robots, where this concept is applied for its balancing, recently a lot of work is being done to try and find new control methods. [1]

In 2001 the company Segway introduced an interesting construction of a transportation vehicle which was based on the physical dynamics from the inverted pendulum. On the vehicle the cart is replaced by two wheels. Between the wheels a platform is placed with room for one passenger. The Segway is able to balance the passenger in upright position and at the same time drive forward or backward.
2. CHAPTER ORGANIZATION AND BACKGROUND THEORY

This section describes the theory behind the inverted pendulum and the theory involving various components of the project. These components are the sensors, the control system, the circuits that make up the actual system and their driving followed by wireless data communication.

We’ve mentioned (in section ‘1.2. Problem Statement’) that the output of the sensors need to be filtered out. So, the theory behind the filtering methods used for the purpose will be stated. Under the control system, PID controller will be explained. The theory behind motor driver circuits and voltage shifter circuit that are used to bridge the working voltage gap between the processor circuit and the motor driver circuit will also be covered.

It is important to keep in mind that this section only gives the background theory necessary to understand the stated topics. How these components are implemented in the project will be explained under the chapter ‘METHODOLOGY, SYSTEM DESIGN AND IMPLEMENTATION’.

2.1. Understanding Inverted Pendulum

The inverted pendulum can be modelled as a pendulum mounted on a motorized cart as shown below. The inverted pendulum system is a very popular example in the control system world mainly due to the fact that it is unstable without control. This means that the pendulum will simply fall over if the cart isn’t moved to balance it. Additionally, the dynamics of the system are nonlinear. The control system applies the force to balance the cart that the pendulum is attached to. This section is based on [4]. We are going to state all of it here again to explain the initial equations with the use of vectors and state the parameters even properly so that the readers can understand the topic clearly in this all important section.
2.1.1. Problem Statement

Consider the following two-dimensional structure. The control input is the force $F$ that moves the cart horizontally. This force for a motorized cart consists of applying the required voltage for the motor to drive the wheels. The outputs based on which this control force is applied are the angular position of the pendulum $\theta$ and the horizontal position of the cart $x$.

![Free body diagram of an inverted pendulum](image)

Figure 2.1 Free body diagram of an inverted pendulum [4]

Figure 2.1 not only paints the picture of our description of the inverted pendulum but also shows the forces involved in the inverted pendulum.

The following are the names of the parameters and forces, shown or not shown in the above diagram, playing a significant role as to why the inverted pendulum behaves as it does.

- $F =$ Control force applied to the cart
- $M =$ mass of the cart which needs moving with an acceleration
- $m =$ mass of the pendulum which moves with an acceleration, is pulled down due to gravity and rotates about the hinge that is located at the point of contact of the cart and pendulum. The rotation is not shown in the figure.
- $b =$ coefficient of friction opposing the control force
\( R \) = Force of unknown direction acting on the frictionless pin or hinge that is the pendulum. This force is directed at a certain angle from the vertical. \( R \) is thus resolved to the forces \( P \) and \( N \). Remember that the force is acting on the pendulum but as per Newton’s Third Law of motion, the cart experiences the forces of equal magnitude in the opposite direction which is shown clearly in the free-body diagram.

\( \chi \) = direction of the movement of the cart in order to balance the pendulum.

2.1.1.1. **System Equations.** Summing the forces in the free-body diagram of the cart in the horizontal direction,

\[
F - N - b\dot{x} = M\ddot{x}
\]  

(2.1)

Consider a mass \( P \) rotating about \( O \). \( P \) is rotating counter clockwise as we see. Let the bold letters denote vectors and the rest denote scalars. It is important to know that the angular velocity \( \omega \) and angular acceleration \( \alpha \) is directed outward at point \( O \) which we label as \( k \).

Our aim is to calculate the acceleration \( a \) experienced by \( P \) based on [5] and then use it on the free body diagram of the cart.

![Figure 2.2 Object with center of mass at P rotating about O in the plane xy](image)

The position vector of \( P \),

\[
\mathbf{OP} = \mathbf{r}
\]

(2.2)
The magnitude of $\mathbf{OP}$ is its length $r$. The velocity is given by,

$$v = \frac{dr}{dt} \quad (2.3)$$

The magnitude of velocity $v$ is,

$$v = \frac{ds}{dt} = r\dot{\theta} \quad (2.4)$$

Where, $s = r\times\theta = r\theta$, is the arc traced by $P$.

The direction of $v$ is tangent to the point of arrival since $P$ is travelling in a circle. The proper definition vectorially would be, if

$$\omega = \omega k = \dot{\theta}k$$

$$v = \omega \times r \quad (2.5)$$

(2.6) explains the tangential direction of $v$. The acceleration $a$ is given by, as per the rules of differentiation of vectors,

$$a = \frac{dv}{dt} = \frac{d(\omega \times r)}{dt} = \frac{d\omega}{dt} \times r + \omega \times \frac{dr}{dt}$$

$$= \alpha \times r + \omega \times (\omega \times r) \quad (2.7)$$

The angular acceleration is given by,

$$\alpha = \alpha k = \ddot{\theta}k \quad (2.8)$$

Thus, from (2.7) and (2.8),

$$a = \alpha k \times r - \omega^2 r \quad (2.9)$$
From the above equation, the magnitude of the tangential component and the normal component is given in order by,

\[
a_t = r\alpha \\
a_n = r\omega^2
\]  

(2.10)  

(2.11)

The tangential acts normal to the pendulum and the normal component acts along the pendulum and \( r \) is replaced by \( l \). Thus, the horizontal equation for the pendulum

\[
N = m\ddot{x} + ml\dot{\theta}\cos\theta - mlt^2\sin\theta
\]  

(2.12)

The physical meaning of this equation is rotation is the effect of the reaction (labelled as \( R \) before) to the hinge that is pendulum. Combining (2.1) and (2.12), we get

\[
(M + m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - mlt^2\sin\theta = F
\]  

(2.13)

This is our first equation of motion. For the second equation we will not equate the vertical forces. We will instead sum the forces perpendicular to the pendulum. Solving the system along this axis, we get

\[
N\cos\theta + P\sin\theta - mgsin\theta = ml\ddot{\theta} + mx\cos\theta
\]  

(2.14)

To get rid of the \( P \) and \( N \) terms in the equation above, sum the moments about the centroid of the pendulum to get the following equation

\[
-Nl\cos\theta - Pl\sin\theta = I\ddot{\theta}
\]  

(2.15)

Combining (2.14) and (2.15),

\[
(I + ml^2)\ddot{\theta} + mglsin\theta = -ml\ddot{x}\cos\theta
\]  

(2.16)

Let us change (2.13) and (2.16) into linear form by removing the trigonometric functions if possible. This is necessary because the control technique that we will be using applies only
to linear systems. Let $\phi$ represent the deviation of the pendulum from vertical. For two reasons that are one, for the sake of linearization and the second we plan on not letting the pendulum deviate more than 20 degrees i.e. around 0.05 radians, $\phi$ is small. Thus

\[
\begin{align*}
\cos \theta &= \cos(\pi + \phi) \text{ nearly equals } 1 \\
\sin \theta &= \sin(\pi + \phi) \text{ nearly equals } -\phi \\
\dot{\theta}^2 &= \dot{\phi}^2 \text{ nearly equals } 0
\end{align*}
\]

After substitution, we arrive at two linear equations of motion, where $u$ is substituted for $F$.

\[
\begin{align*}
(I + ml^2)\ddot{\phi} - mgl\phi &= ml\dot{x} \\
(M + m)\ddot{x} + b\dot{x} - ml\ddot{\phi} &= u
\end{align*}
\]

(2.17) \hspace{1cm} (2.18)

2.1.1.2. Transfer function. For the transfer function from the two latest equations, one needs to take the Laplace transform of the system equations. For the derivative terms, assume zero initial conditions. The resulting equations include the terms $\Phi(s)$, $X(s)$, $U(s)$. The transfer function is the relationship between a single input and single output at a time. The first transfer function will have the output $\Phi(s)$ and the input $U(s)$ through the elimination of $X(s)$, which is quite simple. The transfer function is then

\[
P_{\text{pend}}(s) = \frac{\Phi(s)}{U(s)} = \frac{mlq}{s^3 + \frac{b[I + ml^2]}{q}s^2 - (M + m)gq - \frac{bmgq}{q}}
\]

(2.19)

Where,

\[
q = [(M + m)(I + ml^2) - (ml)^2]
\]

Similarly,
The analysis was carried out by observing various responses, printing pole zeros in MATLAB. The values were given for the transfer functions’ unvalued elements or parameters. The response of the open loop transfer function for Impulse input and Step input follows.

\[
P_{\text{cart}}(s) = \frac{X(s)}{U(s)} = \frac{(I + ml^2)s^2 - gml}{s^4 + b(I + ml^2)q s^3 - (M + m)mgls^2 - \frac{bmgl}{q}s}
\]

2.1.1.3. Analysis. The analysis was carried out by observing various responses, printing pole zeros in MATLAB. The values were given for the transfer functions’ unvalued elements or parameters. The response of the open loop transfer function for Impulse input and Step input follows.

![Open Loop Impulse Response](image)

**Figure 2.3 Open Loop Impulse Responses**
The figure clearly shows that without any control action, the system is unstable. The inclination tends to increase with time i.e. it falls. The cart’s position shown by x means it moves to the right. As per definition, an unstable system produces unbounded output for bounded input. The pole zero plot for the transfer function also shows that the pole has positive real part or it lies in the right side of the complex s-plane.

It helps to know that, Multiple Input Multiple Output (MIMO) systems have same poles (but different zeros) unless there are pole-zero cancellations.

2.1.2. PID Control

Under this sub-heading, we will just see the effect of PID controllers on the system that has just been explained. The theory about PID controller and the meaning of kp, ki and kd will be explained later.

We will attempt to maintain the pendulum vertically upward when the cart is subjected to a 1-Nsec impulse. We also define our design criteria: i) Settling time of less than 5 seconds ii) Pendulum should not move more than 0.05 radians away from the vertical.
Rearranging Figure 2.5 to give it a more common look with the controller as a feedback component, we get Figure 2.6.

The resulting transfer function from the diagram is followed by the response of the system for various values of kp, ki and kd.

\[
T(s) = \frac{\Phi(s)}{F(s)} = \frac{P_{\text{pend}}(s)}{1 + C(s)P_{\text{pend}}(s)} \tag{2.21}
\]
Figure 2.7 PID Response for unit kp, ki, kd

Figure 2.8 PID response after changing to kp=100
2.2. Accelerometer and Gyroscope

All the necessary explanation and derivations required for the explanation of accelerometer and gyroscope and their combination is given in [6]. We will state their general definitions here. Accelerometer is a device which measures the acceleration of an object. A three degree or a triaxial or a triple axis accelerometer measures the acceleration experienced by an object along three coordinate axes giving three values. A gyroscope is a device which measures the rate of change of inclination. Our application of the components of the IMU also is based on the concept given in [6].

2.3. Complementary Filter

The complementary filter is one of the ways of combining the sensor output obtained from more than one sensor. We use two sensors for tilt calculation accelerometer and gyroscope. Complementary filter is not computationally complex and gives a satisfactory result most of the time. We need both accelerometer and gyroscope for the following reasons. On the short term, we use the data from the gyroscope, because it is very precise and not
susceptible to external forces. On the long term, we use the data from the accelerometer, as it does not drift. In its most simple form, the filter looks as follows: [7]

\[
\text{Angle} = \alpha \times (\text{Angle} + \text{gyroAngle} \times dt) + (1 - \alpha) \times \text{accAngle}
\]

Alpha is the tuning parameter for the gain, Angle is the measured angle, gyroAngle is the angular velocity, \( dt \) is the difference in time between iterations and accAngle is the angle measured by the accelerometer.

This equation is obtained from a simple passive low pass filter shown in Figure 2. The derivation is given below: [8]

\[ R \]
\[ C \]
\[ \text{V}_\text{in} \]
\[ \text{V}_\text{out} \]

Figure 2.10 A simple RC lowpass filter [8]

\[
\text{Vin}(t) - \text{Vout}(t) = Ri(t) \quad (2.23)
\]

\[
Qc(t) = C \times \text{Vout}(t) \quad (2.24)
\]

\[
i(t) = \frac{d(Qc)}{dt} \quad (2.25)
\]

From (2.23), (2.24) and (2.25),

\[
\text{Vin}(t) - \text{Vout}(t) = R \times C \times \frac{d \text{Vout}(t)}{dt} \quad (2.26)
\]

Convert this into discrete form as we will be taking measurements at a fixed interval of time,
\[ X_i - Y_i = RC(Y_i - Y_{i-1}) \]  
(2.27)

Rearranging terms gives the recurrence relation,

\[
\begin{align*}
Y_i &= X_i \left( \frac{dt}{RC + dt} \right) + Y_{i-1} \left( \frac{RC}{RC + dt} \right) \quad (2.28) \\
Y_i &= X_i \times \text{alpha} + Y_{i-1} \times (1 - \text{alpha}) \quad (2.29)
\end{align*}
\]

2.4. The Kalman Filter

The Kalman filter is another filter that can be applied for sensor fusion. The Kalman filter is very powerful in several aspects; it supports estimation of past, present and even future states, and it can do so (predict) even when the precise nature of the modelled system is unknown. [9]

Why the word filter? Finding out the best estimate amounts to filtering out the noise. Kalman filter doesn’t only clean up the measurements but also projects these measurements onto the state estimate. [10]

After this introduction, let us move to the steps involved in using the Kalman filter.

This filter is based on predictions and estimates, it uses probabilistic theories, more particularly Bayes’ Theory which involves conditional probability.

What is conditional probability? The following information on conditional probability is based on [11]. Conditional probability takes into account information about the occurrence of one event to find the probability of another event. So the main goal of the conditional probability is to determine the unknown probability, on the basis of the information supplied by the past records or experiments. This concept can be extended to revise probabilities based on new information. The procedure of revising these probabilities is called **Bayes’ Theorem**. It is nothing more than modified form of conditional probability under statistical dependence. Mathematically, Bayes’ Theorem can be defined as
\[ P(H|D) = \frac{P(H) \times P(D|H)}{P(D)} \]  

Where,  
- \( D \) = Data  
- \( H \) = Hypothesis  
- \( P(H|D) \) = Probability of hypothesis given the data  
- \( P(D|H) \) = Probability of data given the hypothesis

Rewriting this in the iterative frame, where \( t \) is the iteration number.

\[ P(H_t|D_t) = \frac{P(H_{t-1}) \times P(D_t|H_{t-1})}{P(D_t)} \]  

So, we are updating the Hypothesis at point \( t-1 \) by using the data at point \( t \). For the next step,

\[ P(H) = P(H|D) \text{ or } P(H_t) = P(H_{t-1}|D_t) \]

Then we increase each \( t \) by 1. The constraint to this theorem is that the state remains constant. The data is constantly received and the hypothesis is continuously estimated but the state isn’t changing. We are estimating for the same state.

The Kalman filter adds to this by modelling the equation for the state using a control variable. For example: Suppose we want to know the distance or position of a moving object that has a GPS attached to it. This GPS is responsible for data, \( D_t \). The position of the object is initially given by estimation \( H_{t-1}, \) afterwards by \( H_t \). Now to change the position of the object a control variable \( A_t \) or control equation using such a variable is used. If \( A_t \) wasn’t used, the object doesn’t move and \( H_t \) and \( H_{t-1} \) should be close as it is for the same position. So the state is changed using this control variable \( A_t \). [12,13,14]

Now let us move to the equation or the process section of the Kalman filter.
2.4.1. Modelling a system

The Kalman filter uses linear system of equations [15],

State prediction

\[ X_t = F_t \times X_{t-1} + B_t \times U_t + W_t \]  

Sensor Prediction

\[ Y_t = H_t \times X_t + V_t \]  

Where,

- \( U/U_t \) = control variable
- \( F/F_t \) = State transition matrix
- \( X/X_t \) = State vector which includes the variable whose value needs to be filtered out
- \( B/B_t \) = Control vector
- \( W/W_t \) = Process noise vector
- \( V/V_t \) = Measurement noise vector

\( W_t \) and \( V_t \) are assumed to have Gaussian probability distribution with zero mean and variances as shown below. [9]

\[ P(W) \sim N(0, Q) \]
\[ P(V) \sim N(0, R) \]

An example of such a modelling is now presented. The method of modelling, that is specifying X, U, F, B, H can be done as per convenience. Specifying these matrices means assigning the elements as variables. Since, we have two sensors i.e. an accelerometer and a gyroscope, the gyroscope can be used to model the process (state prediction) and accelerometer to model the measurement (sensor prediction). If the process has a known equation, the equation itself can model the process. After that a single sensor or multiple
sensors can model the measurement. Thus the method of modelling depends on the one using the Kalman filter. We can use the former method in the following manner:

\[ X = \begin{bmatrix} \text{angle} \\ \text{gyro drift} \end{bmatrix} \quad (2.35) \]

\[ U = \text{angular velocity} \quad (2.36) \]

\[ F = \begin{bmatrix} 1 & -dt \\ 0 & 0 \end{bmatrix} \quad (2.37) \]

\[ B = \begin{bmatrix} dt \\ 0 \end{bmatrix} \quad (2.38) \]

\[ H = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (2.39) \]

Here,

angle = angular tilt

angular velocity = rate of change of angular tilt.

dt = time interval or time between two measured values. This is also the time interval between two predicted values because for each measured value (sensor prediction) there is a value predicted by the equation used to describe the process (state prediction).

As we’ve mentioned, the process and the measurement noise are assumed to have, in probabilistic terms, Gaussian or Normal distribution with zero mean and variances \( Q \) and \( R \) respectively. This is hardly the case in reality. But, let us choose Gaussian noises anyway for MATLAB.

\[ W = \text{ProcessNoiseMag} \times \begin{bmatrix} dt \times \text{randn} \\ 0 \end{bmatrix} \quad (2.40) \]

Here, \( \text{randn} \), is a pseudorandom number drawn from the standard normal distribution. The value for the term \( \text{randn} \) is different each time. If the value of \( \text{ProcessNoiseMag} \) is high, then the process noise is high. This noise is the noise in the gyro, as it transfers to the state which is angle it must be multiplied with \( dt \) to convert the noise in rate to angle. \( \text{ProcessNoiseMag} \) is the scaling term.

Similarly,
\[ V = \text{MeasurementNoiseMag} \times \text{randn} \]  \hspace{1cm} (2.41)

*MeasurementNoiseMag* is the scaling term for measurement noise. Now the covariance or variance (For two different random variables it is covariance, but in this case we are just finding the variance meaning for one random variable i.e. \( W \) for process noise and \( V \) for measurement noise.) for the process and measurement noise is given below.

\[
Q = E[WW^T] = \text{ProcessNoiseMag}^2 \times \begin{bmatrix} dt \\ 0 \end{bmatrix} \times \begin{bmatrix} dt \\ 0 \end{bmatrix} \hspace{1cm} (2.42)
\]

\[
R = EVV^T = \text{MeasurementNoiseMag}^2 \hspace{1cm} (2.43)
\]

Thus, now we can write the following equations to model the system and follow the

\[
X = F \times X + B \times u + W \hspace{1cm} (2.44)
\]

\[
Y = C \times X + V \hspace{1cm} (2.45)
\]

This is the modelling that we’ve used for the project as well. Firstly we tested this modelling in MATLAB and then applied it to the main project. For MATLAB, the sensor outputs were not used. The Kalman algorithm was tested by producing the sensor outputs in the program itself. Instead of obtaining the angular velocity from the gyroscope, we used

\[ U = \text{randn()} \] \hspace{1cm} (2.46)

Initially *angle* is assumed to be 0. Also, we assume that there is no *gyro drift*. For the measured or observed value, instead of using the accelerometer output, we just used (2.45). How this works is (2.44) gives you the actual state or value of \( X \). (2.45) then adds the already defined measurement error or noise (\( V \)) to give the observed value.

(Note: When applying the Kalman algorithm to the real system, \( U \) is obtained from gyroscope instead of (2.46). Initially, *angle* is obtained from the accelerometer and *gyro drift* is taken as zero. But *gyro drift* is updated constantly as will be seen in section ‘Combining the accelerometer and gyroscope data: Filtering’ under chapter ‘5.'
METHODOLOGY, SYSTEM DESIGN AND IMPLEMENTATION’. Plus in each iteration angle from the accelerometer is used. Also, there are three variances; two for state prediction owing to angle (measured by gyroscope) and gyro drift and one for sensor prediction owing to angle (measured by accelerometer). These variances are either statistically calculated or constant values are assumed.)

2.4.2. Algorithm

Now after the system has been modelled as above, we move to the algorithm. Following are the equations for the Kalman Filter, each equations are followed by their variance counterparts. [15]

Time Update/ Prediction phase

\[
X_{t|t-1} = F_t \times X_{t-1|t-1} + B_t \times U_t
\]

\[
P_{t|t-1} = F_t \times P_{t-1|t-1} \times F_t' + Q_t
\]

Measurement Update/ Update Phase

\[
Z_t = Y_t - H_t \times X_{t|t-1}
\]

\[
S_t = H_t \times P_{t|t-1} \times H_t' + R_t
\]

Here, \(Y_t\) is the observed or measured value. The term \(Z_t\) pits the predicted value (gyroscope output) against the measured value (accelerometer value).

Innovation Step or the Correction phase

\[
X_{t|t} = X_{t|t-1} + K_t \times Z_t
\]
As \( Z_t \) is higher, the term \( K_t Z_t \) corrects the prediction \( X_{t/t-1} \) by a higher value. But before we do this we must calculate \( K_t \). Following is the formula.

Optimal Kalman Gain

\[
K_t = P_{t|t-1} \times H_t' \times (H_t \times P_{t|t-1} \times H_t' + R_t)^{-1}
\]  

(2.52)

Thus higher the variance in measurement, \( R_t \), the value of \( K_t \) decreases. This means we trust the measurement less or we value the term \( K_t Z_t \) less. The optimal Kalman gain gives how important the correction term is, lesser the Kalman Gain, the correction term becomes less informative and less important. [13]

\[
P_{t|t} = (I - K_t \times H_t) \times P_{t|t-1}
\]

(2.53)

To understand the change in variance, consider \( K_t=0 \). This means \( R_t \) is incredibly high. The measurement cannot be trusted due to a high variance in measurement noise, thus providing no information. But if \( K_t \) has a higher value, then the term \((1-K_t H_t)\) becomes smaller and the variance \( P_{t/t} \) becomes smaller or in a more sensible term, it becomes tighter, thus giving a better estimate of the state. [13]

The updated state \( X_{t/t} \) and the updated variance \( P_{t/t} \) can now be used to calculate the values of the next loop, \( X_{t/t+1} \) and \( P_{t/t+1} \).

The output of implementing the Kalman filtering algorithm for the modelling given in the previous section is shown below. \( \text{ProcessNoiseMag}=0.5 \), \( \text{MeasurementNoiseMag}=10 \) :
Figure 2.11 Red plot shows the modeled system for MIP and black plot shows the measurement value plotted against time (MATLAB)

The red plot is the actual process and its values are not exactly zero but close to it. The red plot is the process itself. But the zigzag curve is the measured value. The values have large random fluctuations because of the large value of *MeasurementNoiseMag*. Changing the values of variance by changing *ProcessNoiseMag* and *MeasurementNoiseMag*, an even wicked curve can be generated for measurement and process. Though the processes cannot have a high variance (as is the general nature), the measurement can; this is the reason for values 0.5 and 10 respectively. If the measurement noise variance is high the Kalman Filter cannot approximate this smooth, though it is not unsatisfactory by a long mile even under high noise variance.
Figure 2.12 Green Plot shows the Kalman estimation of the above plotted against time

2.5. Voltage Level Shifter Circuit

Present technology processes for integrated circuits with clearances of 0.5 μm and less limit the maximum supply voltage and consequently the logic levels for the digital I/O signals. To interface these lower voltage circuits with existing 5 V devices, a level shifter is needed. For bidirectional bus systems like the I2C-bus, such a level shifter must also be bidirectional, without the need of a direction control signal. The simplest way to solve this problem is by connecting a discrete MOS-FET to each bus line.

Devices operating at different voltage level could be connected to the same bus by using pull-up resistors to the supply voltage line. By using a bidirectional level shifter, it is possible to interconnect two sections of an I2C bus system, with each section having a different supply voltage and different logic levels. Such a configuration is shown in Figure 2.13. The left ‘low-voltage’ section has pull-up resistors and devices connected to a 3.3 V supply voltage and the right ‘high-voltage’ section has pull-up resistors and devices connected to a 5 V supply. The devices of each section have I/O with supply voltage related logic input levels and an open-drain output configuration. The level shifter for each bus line is identical and consists of one discrete N-channel enhancement MOS-FET, TR1 for the serial data line SDA and TR2 for the serial clock line SCL.
Figure 2.13 Voltage Level Shifter [16]

The gates (g) have to be connected to the lowest supply voltage VDD1, the sources (s) to the bus lines of the ‘lower-voltage’ section, and the drains (d) to the bus lines of the ‘higher-voltage’ section.

2.6. Control System and PID Controller

Control System is a system designed to control the output of a system. Control Systems are broadly classified into two types; they are open loop control system and closed loop control system. The comparison between open loop and closed loop [17] shows that closed loop systems are better. PID controllers fall under closed loop controllers. PID controllers are widely used. It is basically a method used in programming and if tuned properly, can be incredibly effective and accurate. PID stands for Proportional Integral Derivative, 3 separate parts joined together, though sometimes you don't need all three.

Let us now look at a brief theory on PID Controllers. This overview of the PID controllers is based on [18].

The combination of proportional, integral and derivative control action is called PID controller and the controller is called three action controller. Consider an error signal, e(t), as shown in figure below. e(t) is plotted against time, t.
Let the output of the controller be denoted by $m(t)$. The output of the controller at $t=0$, $m(0)$, is the no error signal. In a controller with proportional control action, there is a continuous linear relation between the output of the controller and actuating error signal. Mathematically,

$$m(t) = K_p \times e(t) \quad (2.54)$$

In a controller with integral control action, the output of the controller is changed at a rate which is proportional to the actuating error signal.

$$m(t) = K_i \int_0^t e(t) + m(0) \quad (2.55)$$
In a controller with derivative control action the output of the controller depends on the rate of change of actuating error signal.

\[ m(t) = K_d \frac{d}{dt} e(t) \]  

(2.56)

Figure 2.17 Control Signal, m(t) vs time, t for a derivative controller

Mathematically, a PID controller can be defined as

\[ m(t) = K_p e(t) + K_p \frac{1}{T_i} \int_0^t e(t)dt + K_p T_d \frac{de(t)}{dt} \]  

(2.57)

Figure 2.18 Control Signal, m(t) vs time, t for a PID controller

2.7. Motor Driver and H-bridge

A motor driver circuit is required to control the motor which in turn controls the direction of rotation of wheels. The wheels must move in accordance with the output of the controller. The controller is implemented in the processor. But the output of the processor i.e. the signal just does not have enough voltage or power to drive the motor properly. Also on the other hand, the processor should not be directly interfaced with the motors to protect it from back emf generated.
The following explanation and required figures have been taken from [19]. H-Bridge is a method used to control the activation and output rotational direction of a motor. It consists of four switches which are closed and open in alternate pair. MOSFETs are used as switches in this bridge.

The operation of H-bridge is explained with a few figures and a table which provides a summary.

![Figure 2.19 Operation of an H-bridge [19]](image)

The switching elements (Q1, Q2, Q3, Q4) are usually bi-polar or FET transistors, in some high-voltage applications IGBTs. The diodes (D1, D2, D3, D4) are called catch diodes and are usually of a Schottky type. These diodes protect the circuit from back emf of the motor. The top-end of the bridge is connected to a power supply and the bottom-end is grounded. All four switching elements can be turned on and off independently, though there are some obvious restrictions.

The basic operating mode of an H-bridge is simple: if Q1 and Q4 are turned on, the left lead of the motor will be connected to the power supply, while the right lead is connected to ground. Current starts flowing through the motor which energizes the motor in the forward direction and the motor shaft starts spinning.

If Q2 and Q3 are turned on, the reverse will happen, the motor gets energized in the reverse direction, and the shaft will start spinning backwards.
In a bridge if we close switch Q1 and Q2 or Q3 and Q4, then we have created low resistance path between power supply and ground, effectively short circuiting power supply. The braking of the motor is achieved. This condition is called ‘shoot-through’ and is an almost guaranteed way to quickly destroy bridge.

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Motor Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Forward</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Reverse</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Forced brake</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Forced brake</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Free running</td>
</tr>
</tbody>
</table>

Table 2.1 Operation of H-bridge
2.10. Pulse Width Modulation (PWM)

To obtain variable speed of motor Pulse Width Modulation is used. PWM uses a single bit output to generate an analog value proportional to the desired speed of the motor. For this switching frequency or baseline period is established. The frequency is selected according to response of the motor. The signal is switched on and off continuously at a fixed frequency. The ratio of on time to total time period is duty cycle. It value ranges from 0-100%. The speed of motor changes according to duty cycle.

\[
V(\text{avg}) = V(\text{operating}) \times \frac{T(\text{on})}{T(\text{on}) + T(\text{off})}
\]  

(2.58)

![Figure 2.22 PWM showing duty cycle](image)

There are two methods of driving dc motor using PWM signal. They are Sign Magnitude and Locked Anti-phase.

**Sign Magnitude**

It uses two signal to drive the motor. First one gives direction and second gives speed. Small value of duty cycle slow speed where as high value gives high speed.

**Locked Anti-phase**

It uses a single pin to give both direction and speed. When the PWM duty cycle is 0-50% the motor rotates in one direction and other direction can be achieved by increasing duty cycle above 50%.
2.9. Wireless Data Transfer

Radio frequency is a term that refers to alternating current (AC) having characteristics such that, if the current is input to an antenna, an electromagnetic (EM) field is generated suitable for wireless broadcasting and/or communications. These frequencies cover a significant portion of the electromagnetic radiation spectrum, extending from nine kilohertz (9 kHz), the lowest allocated wireless communications frequency (it’s within the range of human hearing), to thousands of gigahertz (GHz).

Using these principles, people have made different RF modules that work on several protocols like UART, SPI, I2C etc. so that the implementation of the wireless data transfer could be done easily, without having to take care of the antenna design and other hardware parts. We will only be required to program the module to send the data that needs to be transmitted via the above mentioned protocols and it will be transmitted. Some of the wireless modules available are XBee, RFM transceivers, UART based transmitter-emitter etc.
3. LITERATURE REVIEW

For the last fifty years, the inverted pendulum system has been the most popular benchmark, among others, for teaching and researches in control theory and robotics. [3] It also is a favourite experiment in control system labs. The highly unstable nature of the plant enables as impressive demonstration of the capabilities of feedback systems. The inverted pendulum is also considered a simplified representation of rockets flying into space. Due to the advent of humanoid robots, where this concept is applied for its balancing, recently a lot of work is being done to try and find new control methods.

The inverted pendulum has been implemented at various different levels or Degree of Freedom (DOF). The ones implementing 1st DOF are mounted on a fixed base either sliding along a straight direction (Linear Inverted Pendulum) or rotating around vertical axis (Rotary Inverted Pendulum). Other models with higher DOF (double or triple DOF) are mounted on a mobile platform or a Cart structure. As an example, “Segway”, a commercial version of two wheeled inverted pendulum used for personal transportation. Most researches have mainly focused on the balance, while few have tried the driving control and trajectory planning. Trajectory planning means the controller has an additional burden of desired movement along with maintaining balance.

Some of the famous methods of filtering that is used are “The Complementary Filter”, “The Kalman Filter”, “Mahony Filter” etc. and then, for the control loop, PID control system is implemented.

3.1. Complementary Filter

The complementary filter is a frequency domain filter. In its strictest sense, the definition of a complementary filter refers to the use of two or more transfer functions, which are mathematical complements of one another

The complementary filter is satisfactory filtering technique for sensors that are low-cost inertial measurement unit:
…question of using a nonlinear complementary filter for attitude estimation of fixed-wing unmanned aerial vehicle (UAV) given only measurements from a low-cost inertial measurement unit. A nonlinear complementary filter is proposed that combines accelerometer output for low frequency attitude estimation with integrated gyrometer output for high frequency estimation.” [20]

The advantages of complementary filter over classical filters is that it is less complex:

“The main advantages are the low complexity of implementation and the high quality of the results for the case of navigation in outdoor environments (uneven terrain). The results obtained through this system are compared positively with those obtained using more complex and time consuming classic techniques.” [21]

This filter is also widely used for the sensor fusion process for attitude adjustment and navigation system and is rather much easier to implement than the other filters, especially “The Kalman Filter” with a desirable noise reduction.

### 3.2. Kalman Filter

“In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem. Since that time, due in large part to advances in digital computing, the Kalman filter has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation.” [9]

“It was during a visit by Kalman to the NASA Ames Research Center that he saw the applicability of his ideas to the problem of trajectory estimation for the Apollo program, leading to its incorporation in the Apollo navigation computer.” [22]

Kalman filters have been vital in the implementation of guidance and navigation system plus the attitude control systems and they were initially used in these sectors. But these days, Kalman filter is most widely used to solve tracking and navigation problems.

Theoretically, amongst the entire linear filter design, Kalman filter has the least mean square error: “The Kalman filter is a minimum mean-square error estimator.” [22]
“Kalman filter addressed the problems of practical implementation of Weiner filter. Weiner filter requires the need to compute the filter’s impulse response, which imposes difficulty in computation. The Kalman filter, on the other hand, uses state space method which makes the implementation much simpler in discrete time domain” [23]

3.3. PID Controller

For the purpose of controlling the system PID controllers are widely used, which is basically a method used in programming and if tuned properly, can be incredibly effective and accurate. PID stands for Proportional Integral Derivative, 3 separate parts joined together, though sometimes you don’t need all three. It has been used since the 1890’s for automatic ship steering till now and are used widely in industry. One of the earliest examples of a PID-type controller was developed by Elmer Sperry in 1911, while the first published theoretical analysis of a PID controller was by Russian American engineer Nicolas Minorsky. Minorsky was designing automatic steering systems for the US Navy, and based his analysis on observations of a helmsman, observing that the helmsman controlled the ship not only based on the current error (distance/value remaining), but also on past error and current rate of change; this was then made mathematical by Minorsky. His goal was stability, not general control, which significantly simplified the problem. While proportional control provides stability against small disturbances, it was insufficient for dealing with a steady disturbance, notably a stiff gale (due to droop), which required adding the integral term. Finally, the derivative term was added to improve control. From that time to now, PID controlled systems are most commonly used system and in fact, 95% of all closed loop systems in industrial processes are run off PID control.

“PID controllers date to 1890s governor design. PID controllers were subsequently developed in automatic ship steering…In the early history of automatic process control the PID controller was implemented as a mechanical device.” “Electronic analog controllers can be made from a solid-stated or tube amplifier, a capacitor and a resistor. Electronic analog PID control loops were often found within more complex electronic systems, for example, the head positioning of a disk drive, the power conditioning of a power supply, or even the movement-detection circuit of a modern seismometer…Most modern PID
controllers in industry are implemented in programmable logic controllers (PLCs) or as a panel-mounted digital controller.” [24]

This shows that the earliest PID controllers were mechanical controllers. Most modern PID controllers are implemented in the software.

PID controllers have been around for more than a century, and a lot of study has been made on them:

“Extensions to PID-achievable performance assessment, trade-off between performance and robustness, and trade-off between deterministic and stochastic performance objectives are discussed. Future directions are pointed out for research and practice in regard to root-cause diagnosis, plant-wide performance assessment, multivariable assessment, adequacy assessment of existing control strategies performance assessment of model predictive control, and the use of intelligent field devices and artificial intelligence to form a systematic diagnostic methodology.” [25]

While PID controllers can be used to balance an inverted pendulum, multi-input-multi-output (MIMO) systems require more than one PID controller for balance:

“The swinging inverted pendulum robot was successfully balanced through the movement of the car along to and fro horizontal direction using a simple PD controller.” “However, this technique has some limitations from mechanical and electrical points of view such as vibrations, slip of wheels, insufficient current, and heating of transistor. Use of accurate and multiple sensors can increase the accuracy and robustness of the system” [26]

“Although PID controller is a good controller for controlling the single-input-single-output (SISO) systems, only one PID controller cannot be able to control both the cart position and the pendulum angle.” [27]

3.4. XBee

XBee is engineered to support the unique low-cost, low-power wireless sensor network. [28] The modules require minimal power and provide reliable delivery of data between remote devices within a few kilometres range. [29]
Also, it is easy to configure and simple to implement in software as it operates on UART protocol. Due to this, the use of XBee has increased to a great amount, mainly in electronics projects. To name a few, projects like Temperature monitoring system, Home automation system, Data logging etc. have used XBee to good effect.
4. SOFTWARES AND TOOLS USED

4.1. For Simulation

4.1.1. MATLAB®

MATLAB is a high level technical computing language and interactive environment for algorithm development, data visualization, data analysis and numeric computation. It can be used for a wide range of applications. The Simulink models were used for the purpose and were used to test the responses of the system and effects of control tuning. MATLAB provides us with a large number of building blocks which can be used to simulate and test various complex systems. Later, we also implied the use of MATLAB for plotting the real time graphs of accelerometer data, gyroscope data and their filtered data respectively.

4.1.2. Python IDLE and Extensions: Pyserial, Pygame, matplotlib, Numpy

The Python IDLE is used for creating a GUI for the simulation as well as plotting the real time graph of the IMU reading. The extension Pyserial is used to carry out the serial communication from the hardware to the computer. Pygame is used for creating various shapes for designing the object similar to our hardware. Matplotlib and Numpy extensions helped us to create the plot of the received angle in real time.

4.1.3. Code::Blocks

Code::Blocks is a full-featured IDE (Integrated Development Environment) aiming to make the individual developer (and the development team) work in a nice programming environment offering everything he/they would ever need from a program of that kind. Its pluggable architecture allows the developer to add any kind of functionality to the core program, through the use of plugins. Code::Blocks is a software which provides an IDE for high level programming languages such as C, C++. C was used to check if the Kalman filter for filtering the noisy measurements was working or not. C turns out to be an ideal language because of its capabilities of file handling among others.
4.2. Microcontroller Tools

4.2.1. Arduino Mega 2560

Since we required processing the data of the sensors and implement control algorithms, the processor we need processors that could handle such processing speed and give desired result were needed. Since for processing data from the IMU the sampling frequency needed to be around 100Hz to 250Hz [30] and to implementation of the control system required sampling frequency of about 50Hz to 60Hz, for which minimum sampling frequency is about 32Hz [31], using AVR microcontroller with CPU frequency of 20Mhz was suitable to implement so we used Arduino Mega 2560 microcontroller board based on ATmega 2560.

![Arduino Mega 2560](image)

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller. It can be simply connected to a computer with a USB cable or powered with an AC-to-DC adapter or battery to get started. The table of specifications follows.
<table>
<thead>
<tr>
<th>Microcontroller</th>
<th>ATmega2560</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Voltage</td>
<td>5 V</td>
</tr>
<tr>
<td>Input Voltage (recommended)</td>
<td>7-12 V</td>
</tr>
<tr>
<td>Input Voltage (limits)</td>
<td>6-20 V</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>54 (of which 15 provide PWM output)</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>16</td>
</tr>
<tr>
<td>DC Current per I/O Pin</td>
<td>40 mA</td>
</tr>
<tr>
<td>DC Current for 3.3V Pin</td>
<td>50 mA</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>256 KB of which 8 KB is used by bootloader</td>
</tr>
<tr>
<td>SRAM</td>
<td>8 KB</td>
</tr>
<tr>
<td>EEPROM</td>
<td>$ KB</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>16 MHz</td>
</tr>
</tbody>
</table>

Table 4.1 Arduino Mega 2560 specifications

The Arduino Board is used for carrying out the entire control operations. It reads the angle data obtainedwirelessly from the IMU sensor and filters it using complementary filtering method. The filtered data is then used to calculate the corresponding PWM required according to the PID control parameters. It then provides the PWM signals to the motor driver circuits. The PID parameters are tuned according to the responses of the system. Arduino-board uses a microcontroller chip which is loaded with program written in C-language through the Arduino-1.0.5-r2. It is also used to transmit the angle serially to the computer through the XBee module for plotting the real time graph and simulating the MIP.

4.2.2. XBEE2

XBee modules are widely used wireless transceivers for wireless communications. They can send and receive data over a serial port, thus they’re compatible with both computers and microcontrollers (like Arduino). And they’re highly configurable so that we can have meshed networks with many modules as well as just a pair swapping data. They have a
wide range of applications. X-CTU is used to configure XBees, test connections, and pass data between our computer and remote XBees as per our requirements. [33]

Most XBees operate on the 2.4GHz 802.15.4 band, and the channel further calibrates the operating frequency within that band.

XBEE2 is a Series 2 XBee. One of the differences between Series 1 and Series 2 is that the latter series runs ZigBee mesh firmware and not 802.15.4 firmware only. [34]

![Figure 4.2 XBee transceiver](image)

The XBee wireless transceivers are used to transmit and receive wireless data stream within Radio Frequency ranges from the MIP to the computer and in the reverse direction as per requirements. It can handle bidirectional transfers up to very long distances in different baud rates. The modules are first configured properly to assign the baud rates and directions of transfer. In our system, it sends the serial angle data from hardware end to the computer end for creating the simulation and live updating graph.
4.2.3. GY-81 IMU

The GY-81 IMU has BMA180 three-axis accelerometer, ITG-3205 triaxial gyroscope, HMC883L three-axis magnetometer and BMP085 pressure sensor. The IMU is used as a sensor to calculate the inclination of the system. Following is the description of GY-81.

1. 10DOF (Three-axis gyroscope + triaxial accelerometer + 3-axis magnetic field + pressure)
2. Immersion Gold PCB process
3. The use of chip: ITG3205 + BMA180 + HMC8833L + BMP085
4. Power supply: 3-5V
5. LLC (fully compatible with the system 3-5V)
6. Module Size: 25.8mm * 16.8mm mounting hole 3mm
7. Standard 2.5mm pin interface, convenient bread plate experiments connection

Figure 4.3 Top and bottom view of GY-81 IMU [37]
### 4.2.3.1. BMA180 Accelerometer

![BMA180 Accelerometer Diagram](image)

Figure 4.4 BMA180 Accelerometer Pin Description [38]

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DNC</td>
<td>Do not connect</td>
</tr>
<tr>
<td>2</td>
<td>VDD</td>
<td>Analog supply voltage</td>
</tr>
<tr>
<td>3</td>
<td>VSS</td>
<td>Ground</td>
</tr>
<tr>
<td>4</td>
<td>INT</td>
<td>Interrupt output</td>
</tr>
<tr>
<td>5</td>
<td>CS8</td>
<td>SPI chip select</td>
</tr>
<tr>
<td>6</td>
<td>DNC</td>
<td>Do not connect</td>
</tr>
<tr>
<td>7</td>
<td>SCK</td>
<td>Serial clock</td>
</tr>
<tr>
<td>8</td>
<td>SDO</td>
<td>Serial data output</td>
</tr>
<tr>
<td>9</td>
<td>SDI</td>
<td>Serial data in/ out</td>
</tr>
<tr>
<td>10</td>
<td>VDDIO</td>
<td>Digital interface power supply</td>
</tr>
<tr>
<td>11</td>
<td>DNC</td>
<td>Do not connect.</td>
</tr>
<tr>
<td>12</td>
<td>DNC</td>
<td>Do not connect.</td>
</tr>
</tbody>
</table>

Table 4.2 BMA180 Accelerometer Pin Description [38]
Features [38]

1. Wide variety of measurement ranges (±1g, 1.5g, 2g, 3g, 4g, 8g and 16g)
2. 14- or 12-bit ADC conversion
3. 2 selectable I2C addresses
4. Programmable integrated digital filters (no external components necessary)
   8 low-pass filters, 1 high-pass filter, 1 band-pass filter
5. Programmable interrupt features:
   Wake-up, Low-g detection, High-g detection, Tap sensing, Slope detection
6. 2 main standard modes: low-noise and low-power
7. Sleep mode
8. Wake-up mode
9. Self-test capability

4.2.3.2. ITG-3205 Gyroscope

![ITG-3205 Gyroscope Pin Description](image)

**QFN Package**
24-pin, 4mm x 4mm x 0.9mm

**Orientation of Axes of Sensitivity and Polarity of Rotation**

Figure 4.5 ITG-3205 Gyroscope Pin Description [39]
### Table 4.3 ITG-3205 Gyroscope Pin Description [39]

<table>
<thead>
<tr>
<th>Number</th>
<th>Pin</th>
<th>Pin description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CLKin</td>
<td>Optional external reference clock input. Connect to GND if unused.</td>
</tr>
<tr>
<td>8</td>
<td>VLOGIC</td>
<td>Digital IO supply voltage. VLOGIC must be less than or equal to VDD at all times.</td>
</tr>
<tr>
<td>9</td>
<td>AD0</td>
<td>I²C Slave Address LSB</td>
</tr>
<tr>
<td>10</td>
<td>REGOUT</td>
<td>Regulator filter capacitor connection</td>
</tr>
<tr>
<td>12</td>
<td>INT</td>
<td>Interrupt digital output (totem pole or open-drain)</td>
</tr>
<tr>
<td>13</td>
<td>VDD</td>
<td>Power supply voltage</td>
</tr>
<tr>
<td>18</td>
<td>GND</td>
<td>Power supply ground</td>
</tr>
<tr>
<td>11</td>
<td>RESV-G</td>
<td>Reserved-Connect to ground.</td>
</tr>
<tr>
<td>6, 7, 19, 21, 22</td>
<td>RESV</td>
<td>Reserved. Do not connect.</td>
</tr>
<tr>
<td>20</td>
<td>CPOUT</td>
<td>Charge pump capacitor connection</td>
</tr>
<tr>
<td>23</td>
<td>SCL</td>
<td>I²C serial clock</td>
</tr>
<tr>
<td>24</td>
<td>SDA</td>
<td>I²C serial data</td>
</tr>
<tr>
<td>2, 3, 4, 5, 1, 15, 16, 17</td>
<td>NC</td>
<td>Not internally connected. May be used for PCB trace routing.</td>
</tr>
</tbody>
</table>

#### Features [39]

1. Digital-output X-, Y- and Z-Axis angular rate sensors (gyros) on one integrated circuit
2. Digitally-programmable low-pass filter
3. Low 6.5mA operating current consumption for long battery life
4. Wide VDD supply voltage range of 2.1V to 3.6V
5. Standby current: 5uA
6. Digital-output temperature sensor
7. Fast mode I²C (00kHz) serial interface MHz to synchronize with system clock
8. Pins broken out to a bread board friendly 7-pin 0.1” pitch header
4.3. Design Tools

4.3.1. PCB Wizard

PCB Wizard is a powerful package for designing single-sided and double-sided printed circuit boards (PCBs). It provides a comprehensive range of tools covering all the traditional steps in PCB production, including schematic drawing, schematic capture, component placement, automatic routing, Bill of Materials reporting and file generation for manufacturing. In addition, PCB Wizard offers a lot of new features that do away with the steep learning curve normally associated with PCB packages. [40]

4.3.2. KiCad

KiCad is also a designing platform for PCBs. It’s a software suite for the creation of professional schematics and printed circuit boards up to 16 layers. KiCad runs on Windows, Linux and Apple OS X and is released under the open-source GNU GPL v2. The KiCad software tool comes with an additional component library package and a documentation package. Once we install all three packages, we are set with a full fledged software tool for the PCB design.

We have used PCB Wizard and Kicad to create the circuit layouts for the circuits required for the hardware.

We used these softwares to create circuits for voltage shifter circuit and two H-bridge circuits. First we designed the circuits optimally and effectively and then printed the layouts to create circuits. [41]

4.3.2. SolidWorks 2013

SolidWorks is a solid modelling CAD software that runs on Microsoft Windows. [42]
5. METHODOLOGY, SYSTEM DESIGN AND IMPLEMENTATION

Inverted pendulum systems are non-linear systems and require a precise design of the mechanical parts as well as for the controlling part. If there are flaws in the design or in the feedback control of the system, the system will be highly unstable. Following methods were adopted for the cause.

5.1. Mechanical Construction

For the design of the mechanical part, a two wheeled system, both wheels at an equal distance from the centre of mass, has been constructed. The material of the base is Aluminium. The bases are stacked as shown in Figure 5.1 to create layers for the placement of circuits and other system components. The structural design is a very critical topic for projects which are based on high precision control of its parameters and requires proper design and fabrication to achieve control accuracy. In the case of inverted pendulums, the centre of mass should be located at the midpoint [43]. Any error in its design or construction might lead to a highly unstable system for which, tuning the parameters of the controller might become a nightmare.

For the designing phase, SolidWorks 2013 was used to design a model for our inverted pendulum and after its completion, we started the construction part.

Figure 5.1 Cross-section and two dimensional Wireframe View of the design
Figure 5.2 Three Dimensional view of the design

Figure 5.1 and Figure 5.2 are taken from SolidWorks 2013 during the design phase. Also refer section ‘2.1. Understanding Inverted Pendulum’, the mechanical model is based on Figure 2.1. But the frictional force was not taken into account. The length $l$ is taken from the centre of the mass. So, the summary is that the design is focused on bringing the centre of mass to the midpoint.

5.1.1. Custom Configurable and Manufacture Parts

Aside from general machining and fabrication, most of the components within the physical system were designed and manufactured in the Robotics Club, Central Campus, which is the where most of the equipment and the machines for the manufacturing and the fabrication of the design were found. The various machines used are lathe, milling, shaper, surface grinder, drilling machine. To save time, some of the equipment were ordered from the nearby Surya Enterprise.

5.1.2. Components and Materials

The pendulum bar was specified as mild steel. Aluminium was selected as material for the base to reduce the additional weight in the lower portion of the pendulum. The particular version of Aluminium (having high modular section ratio) allows for a low-cost, yet strong option to sustain the forces exhibited within the system. The bar was also a thin walled
version which is again a lower cost due to the drawn manufacturing processes. Selecting the bar in Aluminium also allows for more adjustability with weight fluctuation. By initially starting with less overall weight in the bar, the user can add more weight to the bob or slide the bob weight up or down to adjust the moment of inertia. For the motor plate aluminium stripe was selected whose thickness was 6mm, 3mm, 6mm and 4mm nut bolts and screws were used for the fastening parts instead of welding which would make the pendulum bulky. Ply wood was used for the lower base where as the plastic plate was selected for the upper base which is used as hood. The photograph of the system along with the circuit and components is shown in the section ‘6. RESULT’.

5.2. Sensors

For the feedback of the orientation of the system, an Inertial Measurement Unit (IMU) consisting of an accelerometer, a gyroscope, a magnetometer and a pressure, temperature sensor, each with their individual data, has been used. The name of the IMU is GY-81 IMU. The problem arises due to the fact that for determining the orientation of the system, using the readings from only accelerometer or gyroscope causes unstable prediction of the orientation of the system. The reasons have already been stated in section ‘1.2. Problem Statement’ and section ‘2.3. Complementary Filter’. Because of a lack of credibility of accelerometer in vibrating environment (in our case, the vibrations are caused due to motors) and the drift of the gyroscope, result obtained from individual sensor data is not satisfactory [44]. As such they need to be integrated using commonly used filters (also known as sensor fusion). Complimentary Filter and Kalman Filter were tried for the purpose. Currently, complementary filter is serving the cause. This data is very important for the control loop as a form of feedback.

5.2.1. Accelerometer

The information about the usage of the accelerometer and conversion of the binary output provided by the accelerometer to the unit ‘g’ is very well explained in [6]. The methodology employed for this project is mainly based on the same concept. Figure 5.3 shows the basic data acquisition from the sensors and Figure 5.4 shows the filtering applied to the data from the sensors.
5.2.2. Gyroscope

The implementation adopted for gyroscope is also explained in [6].

5.2.3. Combining the accelerometer and gyroscope data: Filtering

5.2.3.1. The Complementary Filter The section ‘2.3. Complementary Filter’ explains the theory behind the complementary filter. For the implementation part of how we have combined the sensor outputs into one is based on [7]. The concept of complementary filter can be explained with the help of figures below.

![Figure 5.3 Output of accelerometer and gyroscope [7]](image1)

![Figure 5.4 Basic working principle of complementary filter [7]](image2)
A high pass filter (HPF) is an electronic filter that passes high frequency signals but attenuates (reduces the amplitude of) signals with frequencies lower than the cut-off. The actual amount of attenuation for each frequency varies from filter to filter. [45]

A low pass filter is a filter that passes low frequency signals and attenuates (reduces the amplitude of) signals with frequencies higher than the cut-off frequency. The actual amount of attenuation for each frequency varies depending on specific filter design. [8]

Now, using these concepts of the high pass and the low pass filter, we pass the signal from accelerometer into the low pass filter such that drastic change in the sensor reading (often due to noises) are removed and the signal from gyroscope is passed through high pass filter such that drift from the sensor reading (often due integration of noise and offset error, which are low frequency signal) are removed. Hence, tuning the appropriate value of alpha will give the proper estimation of orientation of the object with the removal of noises. This can be realized with the help of block diagram shown in Figure 5.4. This is the algorithm that has been used to implement the complementary filter. One of the algorithm that employs the concepts just discussed is given in [6].

5.2.3.1. The Kalman Filter Based on the overview of the Kalman Filter in section ‘2.4. The Kalman Filter’, we modelled our system for simulation in MATLAB and then application in the Arduino platform. In MATLAB, we checked how good the modelling would be for the Gaussian Noise and then applied the same model in Arduino. The following shows the mathematical calculations involved in implementing the algorithm in Arduino. These algorithms are based on [12], [13], [14], [46] and section 2.4.

The difference between the MATLAB simulation and the Arduino application is the fact that we generated the Gaussian noise, for which the Kalman Filter works the best, ourselves. In the real environment, the noise is certainly Gaussian cannot be said. It is now a good time to revise section 2.4.2.

\[
\begin{bmatrix}
\text{angle} \\
\text{gyrobias}
\end{bmatrix} =
\begin{bmatrix}
1 & -dt \\
0 & 1
\end{bmatrix}
\times
\begin{bmatrix}
\text{angle} \\
\text{gyrobias}
\end{bmatrix} +
\begin{bmatrix}
dt \\
0
\end{bmatrix}
\times
\text{gyrorate}
\] (5.1)
\[
\begin{bmatrix}
P_{00} & P_{01} \\
P_{10} & P_{11}
\end{bmatrix} =
\begin{bmatrix}
1 & -dt \\
0 & 1
\end{bmatrix}
\times
\begin{bmatrix}
P_{00} & P_{01} \\
P_{10} & P_{11}
\end{bmatrix}
\times
\begin{bmatrix}
1 & 0 \\
-dt & 1
\end{bmatrix}
\]
\[+
\begin{bmatrix}
dt^2 & 0 \\
0 & 0
\end{bmatrix}
\times ProcessNoiseMag^2
\]

Initially, angle is obtained from the accelerometer, gyrobias=0 and gyrorate is obtained from the gyroscope. The next step is to get the new accelerometer value newangle. Remember that the angle calculated from the accelerometer output is our observed value. Thus instead of devising the observation matrix \( H \), we simply put

\[
z = \begin{bmatrix} 1 & 0 \end{bmatrix} \times \begin{bmatrix} \text{newangle} \\ 0 \end{bmatrix}
\]

\[
y = z - \begin{bmatrix} 1 & 0 \end{bmatrix} \times \begin{bmatrix} \text{angle} \\ \text{gyrobias} \end{bmatrix}
\]

\[
\begin{bmatrix}
S_{00} & S_{01} \\
S_{10} & S_{11}
\end{bmatrix} =
\begin{bmatrix}
1 & 0 \\
0 & 0
\end{bmatrix}
\times
\begin{bmatrix}
P_{00} & P_{01} \\
P_{10} & P_{11}
\end{bmatrix}
\times
\begin{bmatrix}
1 \\
0
\end{bmatrix}
\]
\[+
ProcessNoiseMag^2
\]

Optimal Kalman Gain

\[
K = S^{-1} \times P \times H^T
\]

\[
K = \begin{bmatrix} P_{00} & P_{01} \\
P_{10} & P_{11}\end{bmatrix} \times \begin{bmatrix} 1 \\
0 \end{bmatrix} \times S^{-1}
\]

\[
K = \begin{bmatrix} K_0 \\
K_1 \end{bmatrix} = \frac{\begin{bmatrix} P_{00} \\
P_{10} \end{bmatrix}}{S}
\]

For \( X \),

\[
\begin{bmatrix}
\text{angle} \\
\text{gyrobias}
\end{bmatrix} =
\begin{bmatrix}
\text{angle} \\
\text{gyrobias}
\end{bmatrix}
+ K \times y
\]

\[
\begin{bmatrix}
\text{angle} \\
\text{gyrobias}
\end{bmatrix} =
\begin{bmatrix}
\text{angle} \\
\text{gyrobias}
\end{bmatrix}
+ \begin{bmatrix} K_0 \\
K_1 \end{bmatrix} \times y
\]

For \( P \),

\[
P = (I - K \times H) \times P
\]
\[
\begin{pmatrix}
P_{00} & P_{01} \\
P_{10} & P_{11}
\end{pmatrix}
= 
\left(\begin{pmatrix}1 & 0 \\0 & 1\end{pmatrix}
- 
\begin{pmatrix}K_0 \\
K_1\end{pmatrix}\times
\begin{pmatrix}1 & 0 \\
0 & 1\end{pmatrix}\right)
\times
\begin{pmatrix}
P_{00} & P_{01} \\
P_{10} & P_{11}\end{pmatrix}
\] (5.12)

Thus, (5.1) to (5.12) is our algorithm for the Kalman Filter applied for sensor fusion.

The tradeoff between the Kalman Filter and the Complementary Filter is the computational complexity and better estimation. The computational complexity contributes to delay. Thus if complementary filter’s estimation turns out to be satisfactory, it is better to use since it is less complex and doesn’t burden the processor.

**Complementary Filter has been used at present time** due to some difficulties that has been discussed in section ‘6. RESULT’.

### 5.3. Simulink model of the Inverted Pendulum

Before starting the implementation of our project, we wanted to study the theoretical details about the Inverted Pendulum. We found from [4], the system model for Inverted Pendulum, considering the direction of force acting on the system to balance itself was only considered in one direction. We implemented that model on to Simulink and found different responses to the varying parameters of the PID controller.

![Simulink model for inverted pendulum](image.png)

**Figure 5.5 Simulink model for inverted pendulum**
Then from the feedback angle ‘theta’ given in the figure above, we extracted the accelerometer and gyroscope data by simply writing a Interpreted MATLAB Function, in which the accelerometer data were extracted by computing the cosine and sine function and the gyroscope data was extracted by computing the difference in angle from current sample to previous sample, using sample time of 4ms both the readings. Using these data, we added noise from a random source block in MATLAB with mean zero value and added a high frequency noise to accelerometer data by using high pass filter and low frequency noise to gyroscope data using low pass filter. These data were used to test the filter design part, which showed satisfactory output. Then the filtered output was passed through the control system block which sent control signals to the Inverted Pendulum system.

Studying and implementing this model in MATLAB helped us gain the theoretical analysis part, as we knew the importance of each component placement and its effect on the system as a whole, and also provided the test platform for the filter design and the control system stage and to tune its parameters to make a perfectly tuned system, hence able to balance itself after certain interval of time.
5.4. Processor

For the overall processing of our project, we have used Arduino Development Board, which consists of ATmega2560 microcontroller. Initially, we tested the output from the IMU (i.e. of the accelerometer and gyroscope). As discussed in the previous sections, due to the problem in accelerometer and lag problem in gyroscope, we needed to combine the both sensor values (digital filtering) to obtain optimum result. This was implemented in Arduino and the results were tested using live plotting in Python. Then after filtering the sensor output, we implemented digital PID controller which took input from the filtered output of the sensor, having a Set-Point of 90 degrees, vertical from the ground-surface. Tuning of the PID controller parameters was done manually using XBee module, in which the Master sends the tuning parameters to the receiver via Python GUI, serially on to the XBee module. Then at the receiver side, Arduino receives the command from the XBee module serially and hence sets the tuning parameters and returns the angle values back to the Master. All the processing of the filtering and the PID controller was done with a sample time of 4ms.

5.5. Power Supply

We have used Lithium battery as our power supply, commonly known as LiPo (Lithium Polymer) battery, specifically of 3 cell. Due to their several advantages, these batteries are quite popular. They have a high charge-discharge efficiency (above 90 %). Unlike cylindrical and prismatic cells, LiPos come in a soft package or pouch, which makes them lighter but lack rigidity. They are very portable and have limited environment impact since Lithium oxides and salts can be recycled, and is very suitable for our project due to its energy efficiency, light weight and portability. But there are some risks involved while working with these batteries, since overcharging or discharging any of the cells could cause explosion. Li-ion batteries use organic solvents to suspend the lithium ions. In situations where the structure of the battery is compromised, the solvent can ignite and vent from the pressurized battery. The result is dangerous and toxic fireworks [47]. A proper charger should be use to charge or discharge the battery with the rating specified by the vendor and since LiPo battery are expensive, their charger is also expensive. Thus these batteries are expensive.
5.6. Circuit Design and Construction

As our system consists of two motors, it requires two motor driver circuits and as the IMU sensor works on 3.3V compared to 5V signal of the microcontroller, a voltage level shifter is required. These circuits are designed on PCB design software like KI-CAD and PCB Wizard, the specifications for which have been given in section ‘SOFTWARE AND TOOLS USED’, for the placement of circuit components of the overall system.

5.6.1. Voltage Shifter Circuit

This is a simple interface, which is used to interface Arduino with other peripherals or interface. The voltage level shifter has been used for conversion of 3.3V to 5V and vice-versa using the concept from [16].

The specifications, schematic and PCB design of the voltage shifter circuit and H-bridge are described in this section.

5.6.1.1. Schematic The following figure shows the KiCad schematic of Voltage Level Shifter used.

Figure 5.7 Test circuit of voltage level shifter
5.6.1.2 PCB Design, Fabrication and Etching The PCB layout of the hardware interface were designed and fabricated. Etching and Fabrication of our PCB were done after that in the Robotics Club, Central Campus, IOE. The PCB was then designed using ‘PCB Wizard’ software. Generally important design considerations were taken while designing the PCB. Strip thickness were considered according to the flow of current. The return were designed so that minimum losses occur. Heat sinks are provided for power MOSFET. To avoid dry soldering, pads were made large. Right angles were avoided to reduce reflectance in routing. After the completion of design it was printed on glossy paper [48].

The design was printed on glossy paper. The copper board was rubbed with a rough surface so that it was clean and free from anything but copper. Then, the copper was washed and dried. The design was placed on the copper and plate was heated with the help of iron. After ironing, it was sprayed with cold water and paper was removed. Then the PCB was placed in the etching solution, of Hydrochloric acid and Peroxide, and agitated for 20-30 minute until all copper was dissolved around the design. The board was left to dry after that and impurity was removed using acetone. At last, holes for component placement were made using a drill machine. And the PCB became ready for soldering the component.

5.6.2 Motor Control and H-bridge Circuit

Motor control is a fundamental part of the system because the system balances itself in wheels. The wheels are driven in turn by motors. So, the generation of proper motor control signals as an output of the control system is important. The motor control system involves design and implementation of H-bridge circuit driven by PWM signal.

Major Components in H-BRIDGE

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Component Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>IRF2110</td>
</tr>
<tr>
<td>2.</td>
<td>Opto-coupler -6N137</td>
</tr>
<tr>
<td>3.</td>
<td>Opto-coupler -PC123</td>
</tr>
<tr>
<td>S.No.</td>
<td>Component Name</td>
</tr>
<tr>
<td>------</td>
<td>------------------------</td>
</tr>
<tr>
<td>4.</td>
<td>IRF3205</td>
</tr>
<tr>
<td>5.</td>
<td>Diode</td>
</tr>
<tr>
<td>6.</td>
<td>Capacitor</td>
</tr>
<tr>
<td>7.</td>
<td>Digital gate 74LS04</td>
</tr>
</tbody>
</table>

Table 5.1 Major components in our H-bridge

5.6.2.1. Schematic The following figure shows the KiCad schematic of H-bridge used.

Figure 5.8 H-bridge schematic

5.6.2.2. PCB Design, Fabrication and Etching The same process as applied for the voltage level shifter circuit was applied for the design, fabrication and etching of the H-bridge circuit.
5.7. Permanent magnet DC motor

There is variation in RPM since the two motors we used were different. We used motors that were identical in dimension but not in speed. The reason for this is stiction. Also, no two motors are same even though they have the same manufacturer and identical model numbers. [49]

5.7.1. Dead-band

The region where the applied voltage has no effect is called dead-band. Trying to implement a control system without considering dead-band would lead to undesired response by the controller as it can be obviously seen that the output from the control system, in presence of small error would produce small output signal, which if lies in the region of dead-band would not give any response. This type of situation is not desired, especially when working on a control system project. A proper system for implementing control algorithms is the one, which would give immediate response to the control signal. Even though the problem of dead-band can be reduced alternatively by implementing new hardware designs, the simpler and cost efficient solution is to compensate the effect in software [50].

After implementing dead-band compensation in software, we plotted the response of the motor, giving ramp input signal to the motors, which showed proportional response to the input signal.

The algorithm implemented for dead band compensation in software is

```c
if( controlOutput < 0 )
    compensatedOutput = dbf + controlOutput
else
    compensatedOutput = dbr + controlOutput
```

where,
controlOutput is the output obtained from the control algorithm
the compensatedOutput is the PWM signal send to the motor drivers,
dbf = forward dead-band
dbr = reverse dead-band

The compensated input signal to remove stiction was implemented successfully and now can be used to implement the control system, where linear response is obtained to the given control signal.

5.8. Wireless Data Transfer using XBee

For the wireless data-transfer, an RF module has been used in both transmitter and receiver part using standard protocols. Our system transmits the orientation feedback to the Master Computer. The Master Computer then transmits the gain parameters for the control system, which saves a lot of time to tune the parameters of the control system. We also plotted the values of the orientation transmitted by the system in a GUI, which also helps in data analysis of the system as a whole. The system then receives the value, decodes the message and passes it to the control system section and thus changes the gain parameters.

We can use XBee to remotely control our robot, or arrange them all over our house to monitor temperatures or lighting conditions in every room. X-CTU is the software used to easily configure an XBee, test connections, and pass data between our computer and remote XBee. [33]

As we’ve mentioned, XBees are of great use because they’re highly and easily configurable. Most of the XBee configuration settings come down to controlling which other XBees it can talk to. Three of the most important XBee settings are: PAN ID, MY address, and destination address.

There are a few levels to XBee networks. First, there’s the channel. This controls the frequency band that your XBee communicates over. Most XBee’s operate on the 2.4GHz 802.15.4 band, and the channel further calibrates the operating frequency within that band. You can usually leave the channel setting alone, or at least make sure every XBee you want to have on the same network operates on the same channel.
The next level of an XBee network is the **personal area network ID (PAN ID)**. The network ID is some hexadecimal value between 0 and 0xFFFF. XBees can only communicate with each other if they have the same network ID. There being 65536 possible ID’s, there’s a very small chance that someone else will be operating on the same network.

Finally there are MY and destination addresses. Each XBee in a network should be assigned a 16-bit address (again between 0 and 0xFFFF), which is referred to as **MY address**, or the “source” address. Another setting, the **destination address**, determines which source address an XBee can send data to. For one XBee to be able to send data to another, it must have the same destination address as the other XBee’s source.

For example, if XBee 1 has a MY address of 0x1234, and XBee 2 has an equivalent destination address of 0x1234, then XBee 2 can send data to XBee 1. But if XBee 2 has a MY address of 0x5201, and XBee 1 has a destination address of 0x5200, then XBee 1 cannot send data to XBee 2. In this case, only one-way communication is enabled between the two XBee’s (only XBee 2 can send data to XBee 1).

In our system, a XBee transmitter sends the serial angle data from Arduino end to the XBee receiver at the computer end for creating the simulation and live updating graph.

### 5.9. Graphical User Interface and Live Graph Plotting

For the enhancement of the project, a GUI application has been created for finding out the orientation of the inverted pendulum at any time being read by the computer. The computer display has to show the rotations or inclinations of the system in real time so that we can take actions to maintain its equilibrium.

The implementation of the GUI application was performed using the Python 3.3.4 along with its binaries Pyserial, Pyside and Pygame. The application receives serial data stream from the hardware through the XBee wireless transceiver module and simulates the orientation of the Inertial Measurement Unit (IMU) sensor in the computer screen precisely so that we can get the exact position on the MIP from distance. We can also examine the
The basics of a python code are used and the required extensions are imported in the python code using appropriate syntax. The 3-Dimensional object created for the simulation is rotated about its z-coordinate with its x and y coordinates fixed in certain angle.

The 3-D object, which looks like the side view of our hardware, is first created using 32-vertices and the faces are drawn using those vertices. The object is produced in the 2-D screen using the formulas involved in perspective projection. And the rotation is done simply by using the method of rotating 3-D objects about various axes. The angles are the object variables which are the basis of object simulation. The inclination of the object to the x-axis is fixed to -15 degrees for the suitable initial orientation. And the inclination of the object to the z-axis is initialized to 0 and the orientation is updated according to the data obtained from the serial interface. Thus, the application can provide the real time simulation of the IMU sensor orientation.

The width of the simulation window can be set by parameters and the colors of the faces of the object can also be changed using the combination of RGB-colors.

Some important commands and functions involved for this purpose are listed as follows:

1. `def __init__(self, “initializations”):`
   This function initializes the object variables or parameters within the class.

2. `pygame.display.set_mode((win_width, win_height), 0, 32)`
   It initializes the dimensions of the pygame screen to be displayed.

3. `rotateX(self, anglex) and rotateZ(self, anglez):`
   These are the functions for rotating the points in the given angles about x and z-directions respectively. Positive values of the angles rotate them in the corresponding axes in counter clockwise direction. The basic algorithm for 3-D object rotation was implied to achieve this.

4. `project(self, win_width, win_height, fov, viewer_distance):`
   This function is used to perform the perspective projection since we have to animate the object with 3 coordinates in a 2-D screen. The shape of the perspective
object depends on the value of fov, viewer distance, width and height of the window.

5. `pygame.display.set_mode((win_width, win_height))`
   It sets the size of the window and displays the simulation on the screen according to the values of window’s width and height.

6. `pygame.draw.polygon(self.screen,self.colors[face_index],pointlist)`
   It creates a polygon in the given screen size, using assigned colors and the list of provided points.

7. `pygame.draw.circle(self.screen,self.circlecolor, (h, k), r, w)`
   It draws the circle with the provided color, centre (h,k), radius (r), width (w) and assigned colors.

8. `Point3D(x_coord, y_coord, z_coord)`
   It defines the coordinates of a 3-D point which construct the face of the object and faces combine to construct the total object. Colors can be filled with R,G and B parameters in order to distinct faces from each other.

In this way, the object is created with appropriate dimensions and it is rotated by initializing the angle with x-axis to -15 degrees while making the angle with z-axis change along with the displacement of the IMU. The simulation of the object is then displayed using the command form.show(). Now, finally the program is run by using command app.exec_().

The live data received from the serial port was used to plot the real time graph in Python IDLE using Matplotlib and Numpy extensions. For this, we referred [51] and [52]. The application receives the angle at which the IMU sensor is oriented, from the XBee wireless transceiver and plots the data accordingly in real time as the sensor is moved back and forth. Initially, it was also used for testing the effectiveness of the filtering method by viewing the plots from both filtered and unfiltered data. The filtered plot had smooth edges and more immune to noise.

Some of the commands and functions for this purpose are listed as follows:

- `ser = serial.Serial('COM5', 9600, timeout=0)`
It sets up the serial connection of the arduino board to the computer. COM5 is the sample port of the computer at which the arduino is connected. The serial data transfer is accomplished at the baud rate of 9600 KBd. We must make sure that the baud rates are synchronized in the sending and receiving end.

- plt.ion()
  It sets the plot in the animation mode.
- plt.ylim(x_val,y_val)
  It sets up the limits for y-coordinate.
- xline.set_xdata(val) and xline.set_ydata(val)
  These commands set xdata and ydata to the new list lengths according to their arguments.
- xline, = plt.plot(x)
  It initializes the plot and sets up the future lines to be modified.
- y = (ser.readline().decode('utf-8')[:-2])
  It gets the data from the serial port and saves it on the variable y.
- fig.canvas.flush_events()
  It updates the serial incoming data in the variable y.
- plt.show()

This command finally draws the plot according to the updated values of y.

The program receives angles at the baud rate of the processor. But, due to the limitations of python extensions, we cannot plot all those data in real time. Therefore, we must compensate by plotting after a number of data is received. The value of angle varies from 0 to 180 degrees being 90 degrees under perfectly stable condition. It helps us to view the system response and helps us to perform the control tuning using PID algorithm.
5.10. System Block Diagram

**BLOCK DIAGRAM AT THE PENDULUM END**

- IMU sensor
- RF transceiver
- Filter
- Tilt
- PID Controller
- PWM
- Right motor
- H-bridge
- Left motor

**BLOCK DIAGRAM AT THE COMPUTER END**

- Input devices
- Display for GUI and Real Time Plot
- Communication Protocol
- RF transceiver
- Processor
6. RESULT

Since testing or finding out how our final outcome will work like as well as avoiding probable errors when designing hardware, we had to perform simulations. We have made a Simulink model in MATLAB for the surface simulation of what the hardware would be like. It was done by using blocks in the Simulink library as the hardware components and programming those blocks as per our requirements. There are a large number of blocks in the library so that we can easily use them to simulate before creating the real hardware. We used, various basic blocks along with PID controller and filters. Besides, we performed the following tasks.

6.1. Mechanical construction:

![Bare mechanical model of the system](image)

Figure 6.1 Bare mechanical model of the system

The figure above shows the mechanical components of the system. This means that this is the original inverted pendulum that we need to control. The rest of the additions which include circuitry and base for their placement are the extras that are added due to the presence of the control system i.e. they are the baggage of the control system.

A photograph of the system along with the circuit and components is shown in Figure 6.2.
6.2. Sensor Output

The Complementary filter has been employed in the Arduino to check the pitch and roll of the IMU. We can see from the figure below, the result obtained by using complementary filter was satisfactory enough as spikes obtained from the data of accelerometer was reduced. Although the spikes are reduced, still the lag of the response is also a crucial factor for an unstable system like the Inverted Pendulum. So proper value of alpha has to be chosen such that noises are also removed and still there is no lag in response from the filtering output. If too much preference is given to the gyroscope data, there will be obvious lag in the response due to the gyro drift which is cumulative and if accelerometer is given more preference, then noise spikes will barely be removed. Hence, proper tuning has to be done in order to obtain optimum result for an unstable system like the Inverted Pendulum.

The results for the complementary filter thus satisfactory. The algorithm for the Kalman Filter was tested on both C and Arduino platform. The output from the accelerometer and gyroscope were transferred to a file. Then the filtering was applied to the contents of the same file using C programming. The output shown is the result of the same. However in the Arduino platform, the output got stuck on initial value for the same algorithm. The reason for this was not known.
Figure 6.3 Accelerometer angle vs. Complementary Filter output

Figure 6.4 Accelerometer angle vs. Low Pass Filter output
Figure 6.5 Kalman Filter Output for process noise variances $1 \times 10^{-9}$, $3 \times 10^{-6}$ and measurement noise variance $3 \times 10^{-7}$

Figure 6.6 Kalman Filter Output for process noise variances $1 \times 10^{-8}$, $3 \times 10^{-6}$ and measurement noise variance $3 \times 10^{-7}$
6.3. Simulink Output

Figure 6.7 Simulink output, the above figure is the filtered output of the noisy plot below using complementary filter.

With reference to the section ‘5.3. Simulink model of the Inverted Pendulum’, the Figure 6.7 shows the filtered output of the Inverted Pendulum System modelled in Simulink. We implemented complementary filter in the Interpreted MATLAB function along with an inbuilt PID block, which used the filtered output from the complementary filter, in Simulink to simulate the behaviour of the Inverted Pendulum.

6.4 Deadband and RPM

Figure 6.8 shows the deadband of the motors which is about 54% of the right motor and about 45% of the left motor. Both the motors had around-about same dead-band, which were compensated in software.

Figure 6.9 shows the motor response used in our system, which is about 150 RPM.

The RPM of the motors were calculated using a tachometer.
Figure 6.8 Deadband in motor

Figure 6.9 No load full speed response of motor
6.5. GUI Interface and Live Graph Plotting

We have already said that we created a GUI at the computer end to animate our system. This was used to determine the behaviour of the system. The GUI took data from the pendulum end wirelessly through XBee module. Although the response of the GUI had a few seconds lag, it gave us a good platform to study the behaviour of the system and tune the control parameters according to it. The following are the two sample frames during the simulation.

Figure 6.10 Two frames during the simulation by rotating the position of the IMU

Figure 6.11 Real time plot of IMU in angle versus time
### 7. COST ESTIMATION

<table>
<thead>
<tr>
<th>Particulars</th>
<th>Per Unit Cost (NRs)</th>
<th>Quantity</th>
<th>Cost (NRs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino Uno Development Board</td>
<td>2000</td>
<td>1</td>
<td>2000</td>
</tr>
<tr>
<td>IMU GY-81</td>
<td>4000</td>
<td>1</td>
<td>4000</td>
</tr>
<tr>
<td>H-bridge</td>
<td>2000</td>
<td>2</td>
<td>4000</td>
</tr>
<tr>
<td>Battery</td>
<td>5000</td>
<td>2</td>
<td>10000</td>
</tr>
<tr>
<td>DC Motor</td>
<td>1000</td>
<td>2</td>
<td>2000</td>
</tr>
<tr>
<td>Glass fiber</td>
<td>100</td>
<td>1</td>
<td>200</td>
</tr>
<tr>
<td>Tyre</td>
<td>1000</td>
<td>2</td>
<td>2000</td>
</tr>
<tr>
<td>XBee</td>
<td>2938</td>
<td>2</td>
<td>5876</td>
</tr>
<tr>
<td>Battery Indicator</td>
<td>600</td>
<td>1</td>
<td>600</td>
</tr>
<tr>
<td>Terminal Converters</td>
<td>-</td>
<td>-</td>
<td>200</td>
</tr>
<tr>
<td>Voltage Shifter</td>
<td>400</td>
<td>1</td>
<td>400</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>-</td>
<td>-</td>
<td>4724</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td><strong>36000</strong></td>
</tr>
</tbody>
</table>

Table 7.1 Cost Estimation
The proper mechanical design, satisfactory filtering of the sensor output followed by a strong control algorithm provides a satisfactory result. Among these one is not important than the other. Proper mechanical design involves a mechanical system with the centre of mass at the midpoint of the base. The connections such as connection between the shaft and motor should not have a backlash and all the other connections should be stable. This project, even though, has yielded a satisfactory result, it still could have been better. But a lack of a satisfactory mechanical system has compromised the time for which the pendulum balances itself. If a robust mechanical design is not done, the tuning becomes a nightmare. Even a little shift of a component that contributes to the mass of the pendulum calls for a change in tuning parameters because the system becomes unstable again.

Further, even better controllers such as the Fuzzy Logic Controllers can be used. There superiority to PID controller lies on the fact that Fuzzy Logic Controllers use a more human like approach. [1]

Finally, this project has been a singular experience because of the amount of knowledge that has been gained from the project. The doing of this project has involved a lot of hard work, especially while tuning the control parameters and filtering the sensors.
BIBLIOGRAPHY


[13] (http://www.youtube.com/watch?v=NT7nYv9RI2Y)


[37] (2014, October)

[38] (2014, March)


APPENDIX A. CIRCUITS

Figure A.1 H-bridge interface

Figure A.2 IMU sensor GY-81 interfaced with voltage shifter circuit
Figure A.3 High discharge Li-PO battery

Figure A.4 Overall System
INDEX

A
accelerometer ........................................................................................................ 18
Accelerometer angle vs Complementary Filter output ........................................ 84
Accelerometer angle vs Lowpass Filter output ....................................................... 84
Aluminium ............................................................................................................. 63
Arduino Mega 2560 ................................................................................................... 55
Arduino-1.0.5-r2 ...................................................................................................... 56
B
Basic working principle of complementary filter .................................................... 66
baud rates ................................................................................................................ 57
Bayes’ Theorem ....................................................................................................... 33
BMA180 .................................................................................................................... 57
BMA180 Accelerometer ............................................................................................ 58
C
Center of Gravity .................................................................................................... 19
channel ...................................................................................................................... 57
closed loop control system ...................................................................................... 41
Codeblocks ............................................................................................................. 54
Combining the accelerometer and gyroscope data: Filtering .................................. 66
Complementary Filter .............................................................................................. 31
complementary filtering ......................................................................................... 56
Components and Materials .................................................................................... 64
conditional probability .......................................................................................... 33
control algorithms .................................................................................................. 18
Control System ....................................................................................................... 41
Controller in Feedback ........................................................................................... 29
Controller Model ..................................................................................................... 28
Cross-section and two dimensional Wireframe View of the design ......................... 63
D
Dead-band........................................................................................................75
Degree of Freedom (DOF)..................................................................................49
derivative term .................................................................................................51
driving control.................................................................................................49
duty cycle........................................................................................................47

E
electromagnetic radiation spectrum ................................................................47

F
Feedback ..........................................................................................................17
Forces acting on the pendulum ..........................................................................16
forward spinning and reverse spinning of the motor ..........................................45
frames during the simulation .............................................................................88
Free body diagram ............................................................................................22

G
Graphical User Interface and Live Graph Plotting ...........................................77
GY-81 IMU ........................................................................................................57

H
H-Bridge ............................................................................................................44
highpass filter.....................................................................................................67
humanoid robots ..............................................................................................v

I
I2C-bus ..............................................................................................................40
integral term .....................................................................................................51
inverted pendulum .........................................................................................v, 16, 18, 20, 21, 22, 49, 63, 77, 82
is filtering the output from the sensors .............................................................18
ITG-3205 ...........................................................................................................57
ITG-3205 Gyroscope .........................................................................................60

K
KiCad .................................................................................................................62
**L**
- LiPo (Lithium Polymer) battery ................................................................. 71
- Live Graph Plotting .................................................................................. 77
- Locked Anti-phase ..................................................................................... 47
- lowpass filter ......................................................................................... 67

**M**
- Matplotlib ............................................................................................... 79
- Mechanical Construction ........................................................................... 63
- METHODOLOGY, SYSTEM DESIGN AND IMPLEMENTATION ................... 63
- MOSFETs ................................................................................................. 44
- Motor Control .......................................................................................... 73
- Motor Driver ............................................................................................ 44
- motorized cart ......................................................................................... 21

**N**
- Numpy ..................................................................................................... 79

**O**
- open loop control system ....................................................................... 41
- operating frequency ................................................................................ 57
- operation of H-bridge .............................................................................. 44

**P**
- PCB Design, Fabrication and Etching ....................................................... 73
- PCB Wizard ............................................................................................. 61
- Permanent magnet DC motor .................................................................. 75
- PID Controller ......................................................................................... 41
- processor ................................................................................................. 55
- Pulse Width Modulation .......................................................................... 46
- Pyserial, Pyside and Pygame ................................................................... 77
- Python 3.3.4 ........................................................................................... 77
- Python IDLE and Extensions .................................................................. 54

**R**
- Radio frequency ..................................................................................... 47
- real time graph ....................................................................................... 79
real time simulation of the IMU sensor orientation ....................................................... 78
response of the open loop transfer function ................................................................. 27

S
Segway .......................................................................................................................... 20
Set-point ......................................................................................................................... 17
Sign Magnitude ............................................................................................................. 47
simple passive low pass filter ...................................................................................... 31
SolidWorks 2013 ........................................................................................................... 62
System Block Diagram ................................................................................................. 81
System Equations ......................................................................................................... 23

T
The Kalman Filter ......................................................................................................... 32
Three Dimensional view of the design ......................................................................... 64
Trajectory planning ....................................................................................................... 49
Transfer function of pendulum ..................................................................................... 26

W
Wireframe View of the design ..................................................................................... 63
Wireless protocols ........................................................................................................ 48

X
XBee modules .............................................................................................................. 56
X-CTU .......................................................................................................................... 56