Sheffield Hallam University

SHARPENS YOUR THINKING

SHEFFIELD HALLAM UNIVERSITY

ENGINEERING PROGRAM

A QUADCOPTER WITH AUTONOMOUS TAKE-OFF AND LANDING ON MOBILE ROBOT PLATFORM

FINAL INDIVIDUAL REPORT

Submitted by

Rathnayake R.M.K.M

EN13043360

M.Eng (hons.) in Electrical & Electronic Engineering

MODULE: 55-7618

December 2016

Supervised by

Dr. Migara Liyanage

Preface

This report describes Individual project work carried out within the engineering program at Sheffield Hallam University between March and November 2016.

The submission of the report is in accordance with the requirement of the Degree of M.Eng (hons.) in Electrical and Electronic under the auspices of the University.

Acknowledgement

It is a great pleasure for me to acknowledge the assistance and contributions of all the People who helped me to make my 4th year group project proposal a success. It would have been impossible to complete this project without the dedicated assistance given by those individuals.

I would like to take this opportunity to express my deep sense of gratitude and profound feeling of admiration to my project supervisor Dr. Migara Liyanage (senior lecturer-faculty of engineering, SLIIT) who was of great help during find a project idea and accept to supervise me through this project. His knowledge of control robotics and his assistance really helped me during the times when I was stuck and had nowhere to go for help. I would like to gratefully Acknowledge our lecture in-charge Dr. Minhua Ding and all the officials at faculty of engineering, Sri Lanka institute of information technology.

Finally, I would also like to thank my parents who have been very generous in their prayers and financially. As without their love and their financial support this project idea would not have been possible. And every individual (who I have not mentioned names above) who gave me even the slightest support (even by words) to make my project idea a success.

Thank you everyone...!

Rathnayake R.M.K.M.

Abstract

Automated Ground Vehicles (aka, AGVs) are devices that can move autonomously to accomplish their tasks. These AGV's could be used to search, destroy, find and relocate objects in many types of industrial manufacturing operations and unstructured environments. In this scenario, the target objects might reside with equal probability at any location in the unstructured or cluttered environment and, therefore, the robot must navigate and search the whole area autonomously and be equipped with specific sensors to detect objects. But with the limitations and the inefficiency of the sensors and the lack of improvements in vision-based object recognition, the area to be covered with minimum possibilities also become limited. so, this is where use of unmanned aerial vehicles (aka, UAV) comes in handy.

Nowadays, drone technology has become smaller and much affordable. Drones are originally developed for military purposes. In today's world, drones now have manifold civil applications such as photography, surveillance and delivery etc. since a quadcopter could fly around the free space, the area it can be covered is vast. Therefore, the researches of AGV equipped with UAVs become very popular. UAVs help AGV to achieve tasks that cannot be achieved alone by AGV itself. however, including constructing enhanced systems for navigation and vision-based object recognition has taken an approach to develop such systems and it requires perfect control algorithms. This report contains an individual work carried out underlying the idea of mobile robot with an automated quad copter which has ability to launch and landing on the same mobile platform while it's moving. In this proposed study the ground vehicle will act as the base station. The base station will perform all memory intensive copulations and communicate with the quad-copter. The quad copter equipped with a global positioning system (GPS), inertial measurement unit (IMU), altitude holding sensor and a FPV camera module for localization and navigation. The quadcopter will estimate its position data and send to the mobile station. Mobile robot (AGV) will carry out all computations to generate the trajectory required for the quad copter to land on a pre-defined marker using computer vision on the base station. The work of designing and implementation of the mobile station and the image processing algorithms for quadcopter are clearly described and explained in detail in upcoming chapters. All programming codes were developed using Python and C languages, and some libraries that included Open computer vision (OpenCV) were utilized for image processing. Each algorithm was tested separately. The experimental results demonstrate that the methodology use for this scenario has a considerable progress for further research studies.

Table of Contents

List of Figures

List of Tables

List of Equations

Abbreviations

AGV - Autonomous Ground Vehicle

UAV - Unmanned Aerial Vehicle

FPV - First Person View

GPS - Global Positioning System

IMU - Inertia Measurement Unit

OpenCV - Open Computer Vision

EOD - Explosive Ordnance Diffused

VToL -Vertical Takeoff Landing

PID - proportional Integral derivative controller

RPM – Revolutions Per Minute

UART - Universal Asynchronous Receiver Transmitter protocol

IBVS - image-based visual surveillance

1. Introduction

Robotics is said to be the next level of technology that comprise with the enhanced design, improved construction and operation of autonomous devices. While there is no generally accepted definition for the term “Robot”, it is often taken to mean a device that can move autonomously from place to place to achieve set of goals. Robotics have been used in variety of applications such as automated ground vehicles, unmanned aerial reconnaissance vehicles, manufacturing plants and health care (e.g. pharmaceutical delivery, surgical operations) etc. use of automated ground robots, and UAV’s in particular, are growing as the range of robot applications. Coordinating an unmanned aerial vehicle (UAV) to an automated ground vehicle (AGV) has become one of the major concentrations in robotics in late 20’s.

In this report, we propose an approach for automated and coordinated landing of an unmanned aerial vehicle on a mobile platform. In this work, mobile manipulator act as landing/ground station for the UAV involved in this scenario. The Aim of this project is to develop an algorithm which provide autonomously vertical take-off and landing (VToL) capabilities of UAV with obstacles avoidance on the mobile manipulator. the autonomous UAV that we already have been used consists with two controllers. Apart from the default KK2 controller, a custom made secondary controller has been added. This controller will have input such as analog proximity levels (Sonar), maintain speed up and down ramps (3-axis accelerometer) and the output thrust and throttle will be defined according to the parameter values of the PID controller which is simulated in the MATLAB Simulink environment. the output from the PID then encoded for the target micro controller, and then downloaded to the chip for execution. The automated ground robot also integrated with two controllers. A primary controller is used to perform all the memory intensive copulation for the quadcopter while secondary controller handle all the other automated operations of the mobile platform such as obstacles avoiding. both robots utilize real time control via RF link and Autonomous control by itself. As a solution for automating the VToL Operation of the quadcopter in addition to PID control algorithm, we have introduced a computer vision based algorithm. Since the autonomous landing of a UAV on moving platform by means of Computer Vision has been extensively investigated in this last decade, we also have implemented an algorithm which provide the ability of tracking, YAW orientation determining and vertical height estimation of the UAV. this project was undertaken because of each group member in the group has the prior knowledge, experience, and enthusiasm for building unmanned automated mobile robot with a UAV in it. From software simulation to real world implementation each group member has contributed their maximum effort for the completion of main objectives at the end.

2. Literature survey

The evolution of Vertical Takeoff and Landing (VToL) Unmanned Aerial Vehicle (UAV) in the field of service and rescue robotics has been increased recently. Because of the enhanced autonomy and unwavering quality of the modern UAVs, tasks like surveillance, bomb disposal and diffuse (EOD), mapping (SLAMM), and more, can now be executed with the aid of these new machines. [1] Those robots were majorly invented for military services. Recent researches however has managed to develop most of the issues that came out with previous versions so that they are faster, increased dexterity, equipped with tools making it easier for them to survive in military operations. Following sections will review the previous researches of mobile robots that has been invented until recently and their technology development. [2]

2.1 Previous work

The very first mobile robot for military services was invented in 1972. it's an EOD (Explosive ordnance robot) robot. It was built from the chassis of an electrically operated wheelbarrow and was designed to carry a hook beneath a car, for then it can be towed to a safe position without endangering a life of a bomb disposal technician. [3] These wheelbarrow robots evolved through decades to finally develop as modern machineries with pincers, disruptors and jammers having the ability to take out bigger and even expert made EODs. New more challenging scenarios are currently investigated in the research community. The AIRobots project has successfully launch inspection tasks with probing in real industrial scenarios. The Sherpa project targets the surveillance and rescuing in hostile environments, as such those in which alpine rescuers, forest guards operate and civil protection. We investigated two primary robot platforms including the Talon by Foster Miller, and the Remotec Andros F6A to understand the key features of existing robots. [4]

Remotec Andros

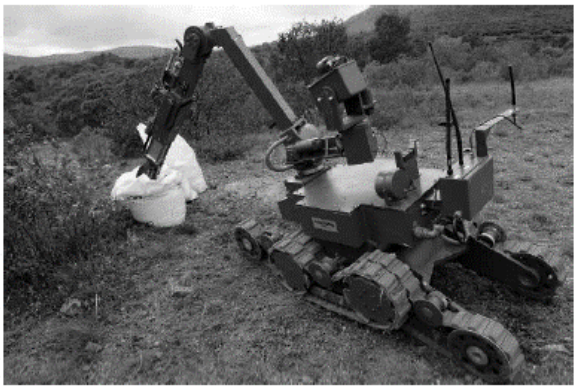


Figure 1 Remotec Andros

Figure 1 shows the Andros range of military robots is produced by Remotec, a subsidiary of Northrop Grumman. Over one thousand Andros robots are in operation across the world, to eliminate evolving threats such as improvised explosive devices (IEDs). Versions of Andros are available in F6, Titus and HD. The Andros family of robots can be utilized with a molded front-drive plate housing microphone, speaker, drive FPV wide angle camera, firing circuits and LED indicators. The manipulator arm with 360 degrees of rotatable gripper is mounted with a color sensor camera with low light switching, extra low light color camera and front/rear drive cameras. The robot is controlled using a hand portable Tactical Advanced Combat controller unit which presents vehicle movement, data from sensors, and errors and failures of revolute joints. The RF link communications ensure a maximum operating range of one kilo meter, while the 36V battery power source provides a maximum mission time of up to 6 hours. [4]

Talon

Figure 2 Talon Robot

Figure 2 shows the TALON robots from QinetiQ North America is generally sent over the world in Improvised explosive devices and Explosive ordnance diffuse. TALON is capable of observation, interchanges, CBRNE, security, overwhelming lift, safeguard and protect operations. These EOD robots have earned a notoriety for elite in military, law authorization and specialist on call missions. More than over thousand TALON robots have been conveyed to clients since its incitement in 2000. The abilities of TALON were effectively demonstrated in salvage missions at Ground Zero after the 2001 World Trade Center assault and amid EOD missions in Iraq. The standard TALON robot weighs up to 52kg in view of the mission profile. It is outfitted with an overwhelming lift arm with 360° rotatable gripper. The payload incorporates three infrareds lit up cameras, a shading zoom camera, discretionary cameras, and a Q-Tray. The two, 300W-hr, 36V batteries guarantee an ordinary run time of three hours. The TALON has a most extreme rate of 8.37km/h and its computerized radio correspondences permit operations up to a separation of 800. [4]

2.2 Technology Development

A perfect solution for the autonomous takeoff and landing of a UAV on a mobile platform is to use computer vision to acquire information in real-time video feed. Camera sensors are cheap, lightweight, low- power consumption, and passive systems, that can be easily integrated on both the UAV and the mobile platforms. [1]

A vision-based algorithm to control a VToL UAV while landing on a moving platform is proposed in recent studies Specifically, an image-based visual surveillance (IBVS) control scheme has been employed to track the pre-defined marker in two-dimensional image space such that it generates a velocity reference command to an adaptive sliding mode controller. [5] Contrasted with vision-based control algorithms, that recreate a 3D representation of the environment, IBVS is computationally less expensive and less sensitive to depth estimation.

Optical flow measurements have also been deeply used for the stabilization and landing of UAV. [6] By exploiting the normal optical stream, a nonlinear controller for hovering flight and landing of a UAV on a moving platform is proposed in late 20's. A similar approach, that makes use of a reconstructed stereo optical flow, has been employed in for the autonomous inspection of planar walls.

3. Statement of the problem

Recently, autonomous robots have been considered for military and space exploration applications. These mobile robots can include explosive ordnance robots (EOD), NASA space rovers, and delivery. However, for search services, robots must recognize specific people or objects, which they may then be required to approach, grasp and relocate. Novel challenges exist in developing a control system that helps a mobile robot to navigate through rough terrain and search its environment until it recognizes the target object. As the robot performs a visual search with limited specification, the choice of an exploration strategy and visual-based object recognition would be difficult.

The search strategy that directs the robot to move to a search site and to relocate as required, involves selecting the vision sensor's viewpoint. One aspect is the object detection process, which is challenging because the robot needs to navigate and place the object in the field of view. The robot also requires a vision system that utilize some image analysis techniques, which are sensitive to environmental conditions, such as lighting, texture and background color. In a classical path planning process, the robot is aware of the start and target locations. However, in a search robot application, the target position is unknown and therefore, the exploration path should cover the entire area and maximize the probability of detecting the target object. Sometimes, even though the object is recognized the directions of movement for the mobile robot are still limited. suppose such incident happened in space exploration rover, let's say for an example rover robot got stuck and obstacles are unavoidable. So, that would-be time consuming and waste of billions of dollars. Lastly, when using a remote-controlled robot with a joystick, its movement directions are limited. Also, when using remote controlled robots, there is a possibility of interference the signals due to weather conditions or signal jamming devices. So, to find a feasible and reliable solution we have proposed to implement autonomous mobile robot with an unmanned aerial vehicle to overcome the issues mentioned above.

4. Aims and objective

4.1.1 Aims

The aim of the project is to design and prototype of a cost-effective autonomous mobile robot with an aerial vehicle with improved functionality, dexterity and situational awareness. The objective of this work is not only a part related to the graduation research work but also a contribution to the country and to the world in terms of an autonomous system that will provide convince in exploration tasks in space.

4.1.2 Objectives

- Model the quadrotor system using SOLIDWORKS.
- Simulate the PID controller and analyze the mobile robot performance using MATLAB.
- Image processing based estimation of coordinates of the landing platform in real time.
- Coordinated control of a quad-copter based on the position of the mobile robot.
- Image processing based yaw control algorithm for quad copter
- Docking of a quad-copter on to a mobile robot platform
- Symbol following algorithm implementation for both quad and the trailer

5. Approach and Methodology

This project has two main parts. Those are quadcopter and the mobile robot. Table 5-1 shows each individual work carried out by each group member. As in the Table 5-1, quadcopter part and its simulation part was approached by other members of the group and highlighted areas indicate my contribution to the project. This chapter onwards I will discuss the procedures and steps that are taken in consideration to design and implementation of the mobile robot and image processing algorithm. Following chapters will describe the task specifications and the design process used to carry the design from the preliminary sketches through to a finished prototype. the mobile robot part is also divided into two main parts. Those are simulation (which is carried out by another member) and real-world implementation. In addition to that landing and direction following image processing based algorithms will be implemented and tested. As I researched further about the advances made to each mobile robot for years, it made me realized that there are pros and cons of each robots. The automation part seems a perfect solution to the problems I noticed in the most robots. Next, for this project to be carried out we needed an expert about the developing field to supervise the project. After finding a supervisor, work plan was drafted. Then I carried out a further literature review to know how my part can be developed according to the whole project work plan. then the expenses for the mobile robot part was estimated and cost estimation plan was written with minimized expenses for the developing process, which can lead to develop a low cost affordable and reliable robot. Next I planned on the designing process, the challenges I would face, steps or methods to overcome those challenges were taken into consideration like how to assemble the robot chassis, cameras, MCUs, electronic components and Heli-pad. the hardware components as well as the software components related to the process was decided and well planned. The designing process is not much complicated and the steps of designing were carefully and methodologically organized. Then I planned on implementing the process of tracking and following the objects via image processing with the help of a development board and open computer vision libraries which are for image processing applications. To follow the mobile robot, an FPV camera mounted under the quadcopter is used. The real-time video feed is then fed to a raspberry pi. An image processing algorithms running on the raspberry pi 3 will determine the position, orientation and the vertical height to the mobile robot relative to the camera and the data are then send to the quadcopter via a wireless RF link.

So, the entire methods/activities are to be done can be divided into four main phases. The expected tasks to be covered from each phases of the project followed very similar approach as in the work plan.

Phase 1:

Literature survey and mobile robot Design Specification:

It is a must to carry out a literature survey according to the problem definition define at the proposal. Then it's important to define the design specification that can be used to guide through the design. A detail list of specifications as shown below helps to achieve the key goals of the final product.

1. The robot size should not exceed its maximum size of 12' x 12.4' x 5.5'
2. The robot must weigh less than 7 lbs.
3. Robot must have onboard power which can operate up to several hours.
4. Robot must stream visual feedback to the user interface from onboard cameras.
5. Robot should have minimum ground speed of 5 mph.
6. Robot must communicate wirelessly with the user interface.
7. Production cost should not exceed $1000
8. Robot must have manual override function
9. Robot must contain modular component
10. 13. Vision systems must be in HD quality in order to identify objects
11. 14. Robot must provide real time position data via cameras.
12. Platform must have the ability of carrying the drone
13. Ground clearance of the landing platform
14. Quadcopter battery recharging capabilities

Phase 2:

Design and Development:

In this phase, all the design works will be carried out. Including gathering resources of components (e.g. sensors, motors, MCUS, cameras, PCBs etc.) to understand how features will work. Modeling physical system with the aid of CAD design, and perform a performance analysis on real world physical system to analyze the speed and the rotation drift in the motors using MATLAB Simulink PID tuner are some of major activities in this phase. All the assembly part and the programming part will be finished from here and make it up for the testing phase.

In addition to that image processing algorithms will be implemented. There are two main algorithms to be implemented. First one is a vision-based landing algorithm while determining vertical height. Second is yaw control algorithm. Even though those control algorithms written in separately however, at the end a combination of two algorithms will be act as one.

Phase 3:

Testing and validation:

In this stage, all the experimental testing's will be carried out. It is essential to confirm that the system will work and meet the design specifications. So, several experiments of mechanical structure and image processing coding will be made.

Phase 4:

Finalize the product:

The results from the design and development phase and the testing phase will be used to determine the changes needs to be done. Once these changes are made, the final product will be modeled and finalized.

Table 5-1 Individual work carried out by each group member

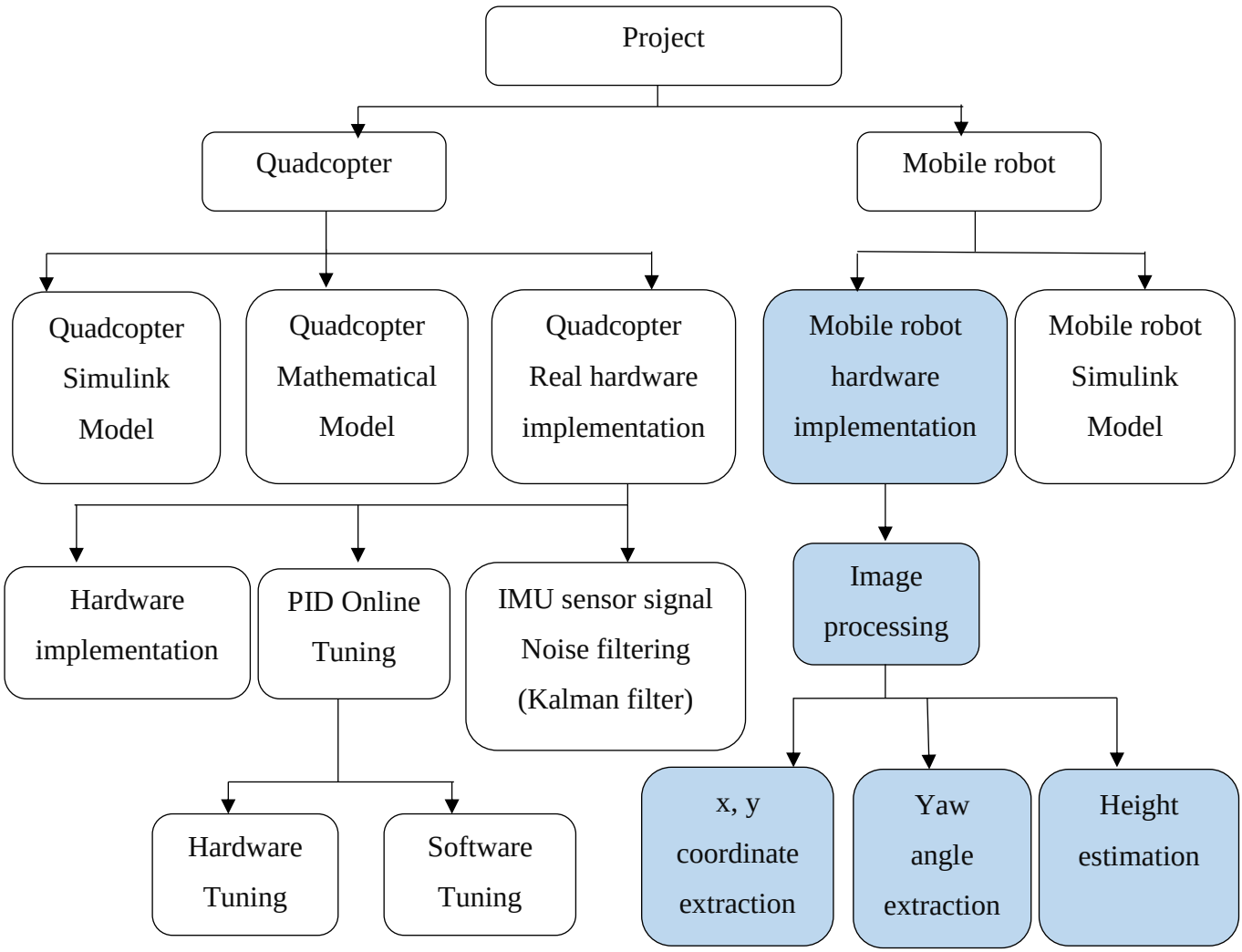


Table 5-1 depicts the individual tasks carried out by each member according to the work plan.

6. Relevant Theory and Analysis

6.1 PID controller

Figure 3 shows an implementation of **proportional–integral–derivative** controller (PID controller). it is a closed loop feedback controller which mostly used in today's industrial control systems. A PID controller repeatedly estimates an error value e(t) as the difference between a desired set-point and a measured process output and applies a correction based on proportional, integral, and derivative terms, (denoted as P, I, and D respectively) which give their name to the controller type. [7]

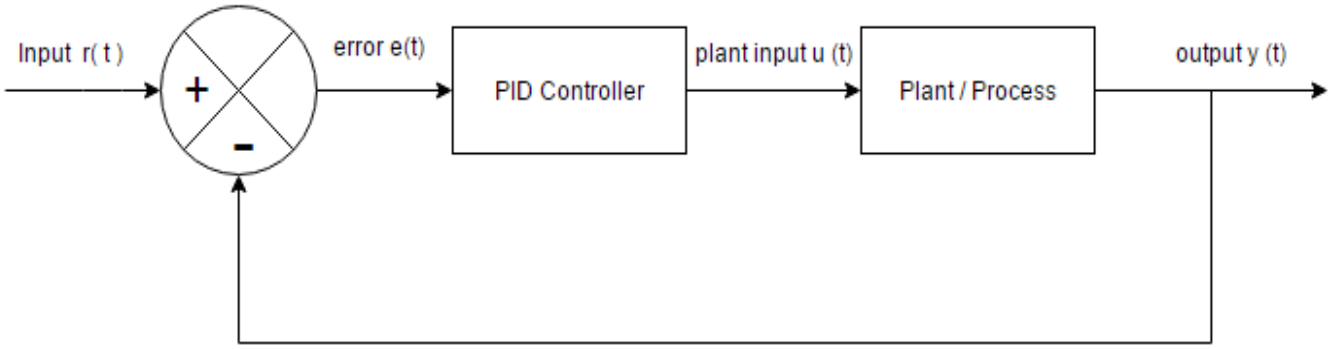


Figure 3 PID Controller

Equation (1) Represent below the mathematical formula of a continuous domain PID controller.

P = $K_p e(t)$, I = $K_i \int_0^t e(t)dt$, D = $K_d \frac{d(e(t))}{dt}$

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d(e(t))}{dt} \quad (1)$$

Equation 6:1 Mathematical model of the PID Controller

Table 6-1 Closed loop response of PID coefficients Kp, Ki & Kd [7]

CL Response	Rise Time	Overshoot	Settling Time	Steady-State Error
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Change	Decrease	Decrease	Small Change

Graph Figure 4 [7]shows the tuning result of an ideal PID controller which is described in the Table 6-1. [7]

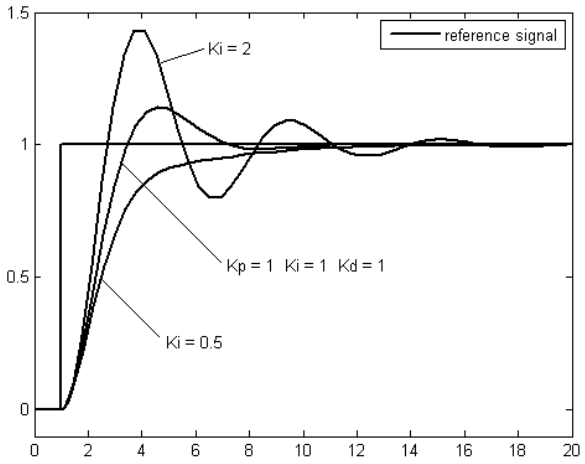


Figure 4 PID tuning

6.1.1 PID Tuning Methods

Ziegler–Nichols method

The Ziegler–Nichols tuning procedure is a heuristic methodology of tuning a PID controller. It was found and created by John G. Ziegler and Nathaniel B. Nichols. It is done by setting the Integral(I) and derivative (D)gains to zero. The "P" (corresponding) gain, is then extended (from zero) until it achieves a definitive gain, at which the yield of the control circle has steady and reliable motions.

Trial and Error method

The Trail and Error technique requires a closed loop system, it ventures through the system from proportional to integral to derivative. This strategy is a partition and conquer approach, first it puts the system into a rough solution from which little changes are performed to perfect the response. To begin, each element of the PID controller is first set to zero. The proportional segment is now considered by expanding its esteem until a steady oscillation is obtained. Scaling the existing proportional esteem downsized by a factor of two will give the resulting proportional gain. Applying this proportional value will expel the steady oscillations. Next the integral coefficient is increased until steady oscillations are again obtained. The present value of the integral coefficient is scaled up by a element of three and applied to the integral as the final value. This once again sets oscillations off, which brings up the derivative control, this value is expanded until for a final time, the oscillations are at a constant period and amplitude. The

coefficient of the derivative is then downsized by a factor of three and applied as the final value for the derivative control. The resulting yield may still have some noise associated with it, this must now be tuned by hand with small educated tweak of the different coefficients. I have used the Trial and Error PID tuning method for Tuning the of mobile robot.

6.2 Kalman filter

The Kalman filter (also called Kalman estimator) has the structure of a standard state observer, as illustrated in Figure 1. The difference between the measured system output and the estimated system output is scaled by the Kalman filter gain Kf and feedback to the observer. The Kalman filter gain Kf itself, is the solution of an optimization problem under the assumption that the process and measurement noise are uncorrelated white noise signals. I.e. the gain matrix, that minimizes the expectation of the estimation error, is chosen. As for the linear quadratic regulator problem, the optimal solution for the Kalman optimization problem is found by solving a discrete algebraic Riccati equation. [8]

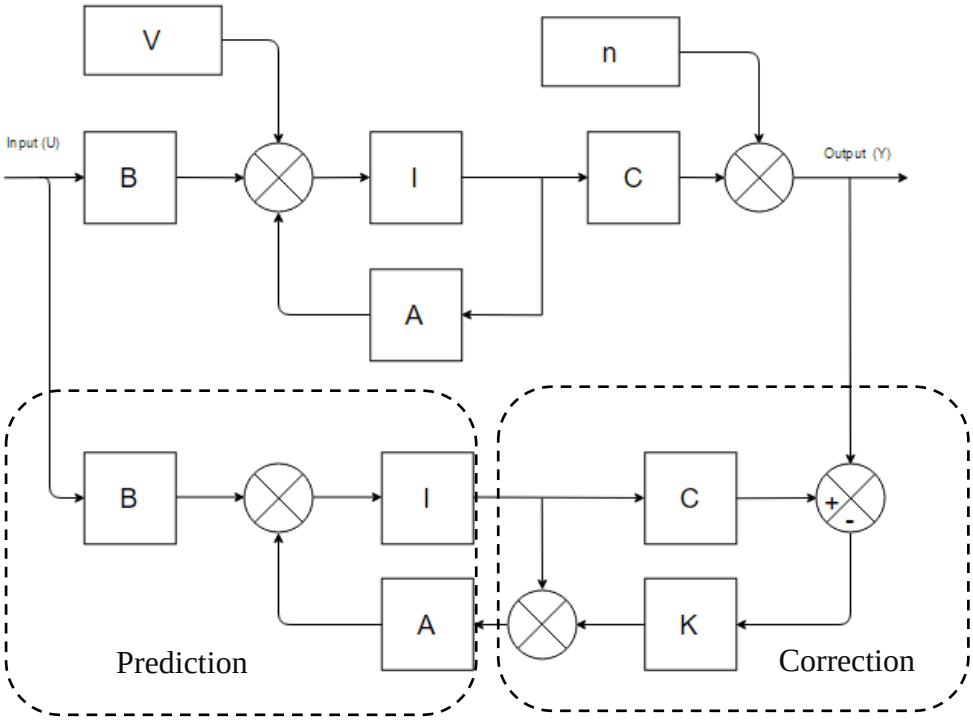


Figure 5 Kalman Filter

Consider the following system, where A, B, C, V are known State Space matrixes and n(k) and v(k) are zero mean stochastic noises at the input and at the output with covariance matrixes N and M.

Equation (2) expressed below represent the state space model of the system.

$$\mathbf{x}(k+1) = \mathbf{A}\,\mathbf{x}(k) + \mathbf{B}\,\mathbf{u}(k) + \mathbf{V}\,\mathbf{v}(k)$$
$$\mathbf{y}(k) = \mathbf{C}\,\mathbf{x}(k) + \mathbf{n}(k) \qquad (2)$$

Equation 6:2 State Space Representation of the System

Then Kalman Filter can be defined in the following form:

Where k is the present value and the k + 1 is predicted value.

Equation (3) expressed below the mathematical representation of the Kalman Filter.

$$\underset{\text{new estimate}}{\hat{\mathbf{x}}(k+1|k+1)} = \underset{\text{old estimate}}{\mathbf{A}\,\hat{\mathbf{x}}(k|k)} + \mathbf{B}\,\mathbf{u}(k) + \underset{\text{correction matrix}}{\mathbf{K}(k+1)}\,[\underset{\text{new measurement}}{\mathbf{y}(k+1)} - \underset{\text{predicted measurement based on old estimate}}{\mathbf{C}(\mathbf{A}\,\hat{\mathbf{x}}(k|k) + \mathbf{B}\,\mathbf{u}(k))}] \qquad (3)$$

Equation 6:3 Mathematical Model of the Kalman Filter

State estimation:

Equation (4) and (5) expressed below the prediction and correction values of the Filter.

$$\textit{prediction:} \quad \hat{\mathbf{x}}(k+1|k) = \mathbf{A}\,\hat{\mathbf{x}}(k|k) + \mathbf{B}\,\mathbf{u}(k) \qquad (4)$$

Equation 6:4 Mathematical Model for Prediction

$$\textit{correction:} \quad \hat{\mathbf{x}}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + \mathbf{K}[\mathbf{y}(k+1) - \mathbf{C}\,\hat{\mathbf{x}}(k+1|k)]$$

Equation 6:5 Mathematical Model for Correction (5)

Equation (6) Expressed below is the Correction matrix:

$$\mathbf{K} = \mathbf{P}\,\mathbf{C}^T[\mathbf{C}\,\mathbf{P}\,\mathbf{C}^T + \mathbf{N}]^{-1} \qquad (6)$$

Equation 6:6 Correction Matrix

Where P is covariance matrix of the estimation error and N is the covariance matrix of the output noise.

Equation (7) expressed below is the covariance matrix of the estimation error

$$\mathbf{P}(k+1) = E\left\{\tilde{\mathbf{x}}(k+1|k)\,\tilde{\mathbf{x}}^{T}(k+1|k)\right\} \quad (7)$$

Equation 6:7 Covariance Matrix of the estimation error

Equation (8) expressed below is the covariance matrix of the output noise.

$$\mathbf{N} = E\left\{\mathbf{n}(k)\,\mathbf{n}^{T}(k)\right\} \quad (8)$$

Equation 6:8 Covariance matrix of the output noise

6.3 Image Processing Theories

6.3.1 Morphological Transformation

While non-linear filters are often used to enhance grayscale and color images, they are also used extensively to process binary images. Such images often occur after a thresholding operation,

Equation (9) shows the thresholding operation.

$$\theta(f,t) = \begin{cases} 1 & \text{if } f \geq t, \\ 0 & \text{else,} \end{cases} \quad (9)$$

Equation 6:9 Thresholding

e.g., converting a scanned grayscale document into a binary image for further processing such as optical character recognition.

The most common binary image operations are called morphological operations, since they change the shape of the underlying binary objects. To perform such an operation, first convolve the binary image with a binary structuring element and then select a binary output value depending on the thresholded result of the convolution. The structuring element can be any shape, from a simple 3 X 3 box filter, to more complicated disc structures. It can even correspond to a particular shape that is being sought for in the image. [9] [10]

Figure 6 [9] shows the binary Image morphology of a letter. [9] [10]

Figure 6 Binary image morphology: (a) original image; (b) dilation; (c) erosion.

Equation (10) shows a close-up of the convolution of a binary image f with a 3 X 3 structuring element s and the resulting images for the operations described below. Let

$$c = f \otimes s \tag{10}$$

Equation 6:10 close up of a binary image

be the integer-valued count of the number of 1s inside each structuring element as it is scanned over the image and S be the size of the structuring element (number of pixels). The standard operations used in binary morphology include:

- dilation: dilate (f, s) = Θ (c; 1);
- erosion: erode (f, s) = Θ (c; S);
- majority: maj(f,s) = Θ (c, S/2);
- opening: open (f, s) = dilate (erode (f, s), s);

6.3.1.1 Erosion and Dilation

The most basic morphological operations are two: Erosion and Dilation. They have a vast range of uses, i.e.:

- eliminate noise
- Isolation of individual factors and joining disparate elements in an image.
- Finding of power knocks or openings in a picture

Dilation

These operations consist of convoluting an image A with some kernel (B), which can have any shape or size, most of the time it is a square or circle. The kernel B has a defined stay point, normally being the center of the kernel. As the kernel B is scanned over the image, we determine the maximal pixel esteem overlapped by B and replace the image pixel in the stay point position

with that maximal value. As you can find, this amplifying operation causes bright regions within an image to “develop” (therefore the name *dilation*). Take for instant the image below. By using dilation, we can get the background enlarge around the dark regions of an object.

Figure 7 [10]shows original image vs dilated image [9] [10]

Figure 7: (a) original image (b) dilated image

Erosion

This operation is the sister of dilation. What this does is to determine a least over the region of the kernel. As the kernel B is scanned over the image, we determine the least pixel value overlapped by B and replace the image pixel under the stay point with that minimal value. Similarly, for the instant for dilation, we can apply the erosion operator to the original image (shown below). You can see in the outcome below that the bright areas of the image background get thinner, through the dark zones get bigger. [10]

Figure 8 [10]shows the Original image vs Eroded Image.

Figure 8: (a) Original Image (b) Erode Image

6.3.2 Non-Linear Image Filtering

6.3.2.1 Gaussian blur

The so-called blur can be understood as taking a pixel as the average value of its surrounding pixels.

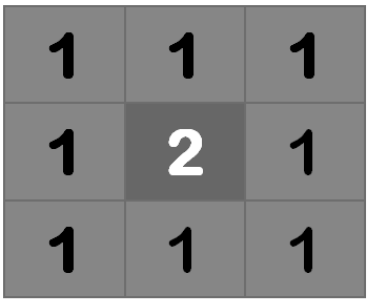


Figure 9 pixel as the average value of its surrounding pixels

On the above Figure 9 [11], 2 is the center point, the surrounding points are 1.

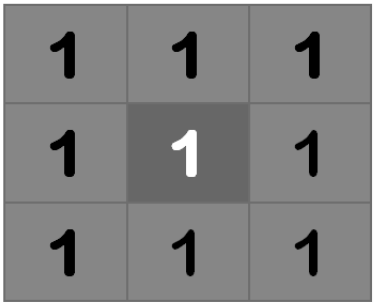


Figure 10 loosing center point

The center point will take the average value of its surrounding points, it will be 1. From value perspective, it's a smoothing. On graphic, it's a blur effect. As shown in the Figure 10 [11], the center point will lose its detail.

Weight of normal distribution:

Figure 11 [11] shows the Normal distribution is an acceptable weight distribution model.

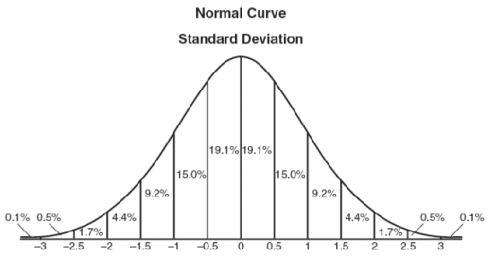


Figure 11 weight normal distribution

On graphic, normal distribution is a Bell-shaped curve, the closer to the center, the bigger the value.

Gaussian function:

The normal distribution above is one dimensional, the graph shown in Figure 12 [11] is two dimensional. We need two-dimensional normal distribution. [11]

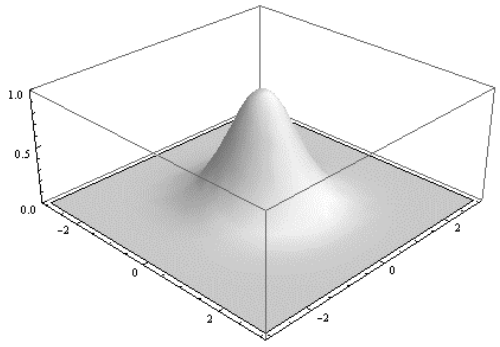


Figure 12 density function

The density function of normal distribution is called Gaussian function. The one dimension format is,

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \quad (11)$$

Equation 6:11 Gaussian Function

Here μ is the average of x, because center point is the origin point when calculating average value, so μ equals to 0.

Equation (12) is a derivation of Gaussian function when μ = 0.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2} \quad (12)$$

Equation 6:12 Derived from Equation (1)

Based on the one dimension function, we can derive the two-dimensional Gaussian function.

Equation (13) expressed below is the two-dimensional Gaussian function

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (13)$$

Equation 6:13 Two dimensional Gaussian Function

With this function, we can calculate the weight of each point.

Weight matrix:

Assume the coordinate of the center point is (0,0), then the coordinates of 8 points which are nearest to it are shown in Figure 13. [11]

(-1,1)	(0,1)	(1,1)
(-1,0)	(0,0)	(1,0)
(-1,-1)	(0,-1)	(1,-1)

Figure 13 the coordinates of 8 points

To calculate the weight matrix as shown in the Figure 14 [11], we need to set the value of σ, σ=1.5, then the weight matrix of blur radius 1 is, [11]

0.0453542	0.0566406	0.0453542
0.0566406	0.0707355	0.0566406
0.0453542	0.0566406	0.0453542

Figure 14 weight matrix calculation

Figure 15 [11] shows the sum of the weights of these 9 points is 0.4787147. If only calculate the Weighted average of these 9 points, then the sum should be 1, hence the above 9 values should divide 0.4787147.

0.0947416	0.118318	0.0947416
0.118318	0.147761	0.118318
0.0947416	0.118318	0.0947416

Figure 15 Sum of weights

Calculate Gaussian Blur:

With weight matrix shown in Figure 16 [11], we can calculate the value of Gaussian Blur. Assume we have 0 pixels now, the gray value (0-255):

14	15	16
24	25	26
34	35	36

Figure 16 weight matrix

Figure 17 [11] shows each point multiplies its weight value:

14x0.0947416	15x0.118318	16x0.0947416
24x0.118318	25x0.147761	26x0.118318
34x0.0947416	35x0.118318	36x0.0947416

Figure 17 multiplication with weight value

Now we have:

1.32638	1.77477	1.51587
2.83963	3.69403	3.07627
3.22121	4.14113	3.4107

Figure 18 resultant weight matrix

Add these 9 values up, then you will get the Gaussian Blur value of the center point. Repeat this process for all other points, then you will get graph after Gaussian blur. The example graphs shown in Figure 19 [12] show the original, 3 pixels' blur radius and 10 pixels' blur radius. The bigger the blur radius, the more blur the picture is. [11] [12]

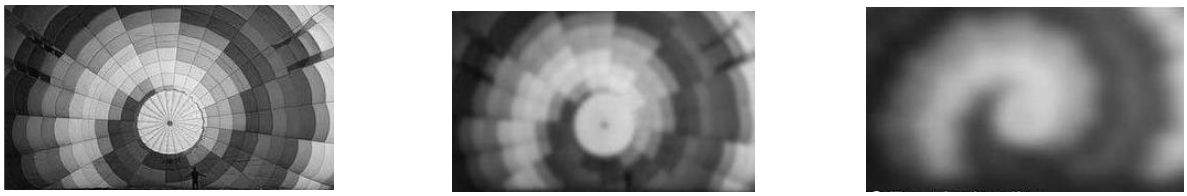


Figure 19 the original, 3 pixels' blur radius and 10 pixels' blur radius

6.3.3 Canny Edge Detector

The Canny Edge detector algorithm was originally found by John F. Canny in 1986. Also, referred to numerous as the optimal detector, Canny algorithm objective is to satisfy three main criteria: [13]

- Low error rate: Meaning a decent recognition of only existent edges.
- Good localization: e separation between edge pixels distinguished and genuine edge pixels must be minimized.
- Minimal response: Just a single identifier reaction for every edge.

Steps,

1. Filter out the unnecessary noise using a Gaussian or median filter.
   Equation (14) expressed below an example of a Gaussian kernel of size = 5.

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \quad (14)$$

Equation 6:14 Gaussian kernel

2. Find the intensity gradient of the image. For this, we follow a procedure analogous to Sobel:

a) Apply a pair of convolution masks (in x and y directions):

Equation (15) expressed below the mathematical model of convolution masks for x and y direction to find the intensity gradient of the image.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$
$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad (15)$$

Equation 6:15 Convolution masks

b) Find the gradient strength and direction with:

Equation (16) expressed below the Gradient and the direction

$$G = \sqrt{G_x^2 + G_y^2}$$
$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \quad (16)$$

Equation 6:16 Gradient and the direction

The direction is rounded to one of four possible angles (namely 0, 45, 90 or 135)

3. Non-maximum suppression is applied. evacuates pixels that are not thought to be a piece of an edge. Consequently, just thin lines (competitor edges) will remain.
4. Hysteresis: The final step. Canny does use two thresholds (upper and lower):

a) If a pixel slope is higher than the upper limit, the pixel is acknowledged as an edge
b) If a pixel slope esteem is underneath the lower edge, then it is rejected
c) If the pixel slope is between the two edges, then it will be acknowledged just on the off chance that it is associated with a pixel that is over the upper limit

Canny recommended an upper: lower ratio between 2:1 and 3:1.

For example, as shown in Figure 20 [13] and Figure 21 [13] using as an input the following image:

Figure 20 input image

we obtain the following result:

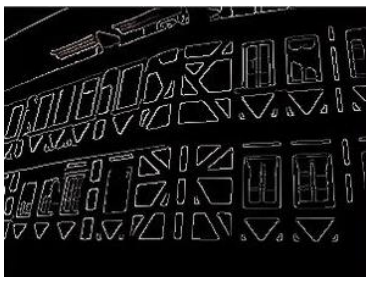


Figure 21 output after canny edge detection

7. Design and Implementation

7.1 Modeling DJI F330 using SOLIDWORKS 2015

Following Figure 22 depicts the workflow of designing the CAD model of the DJI F330 quadcopter.it is necessary to carry out a real-time simulation to avoid unnecessary damages and reduce the outrages of the cost. For this design, I have used SOLIDWORKS 2015 version. Each part was designed according to its original design specifications and real world materials are added in order to observe the mass properties of the CAD model. It is a must to match the mass properties with the actual parts of the quad copter such as inertia, weight etc. otherwise the simulation results could be varying from the expected outcome and the entire real time simulation would have failed.

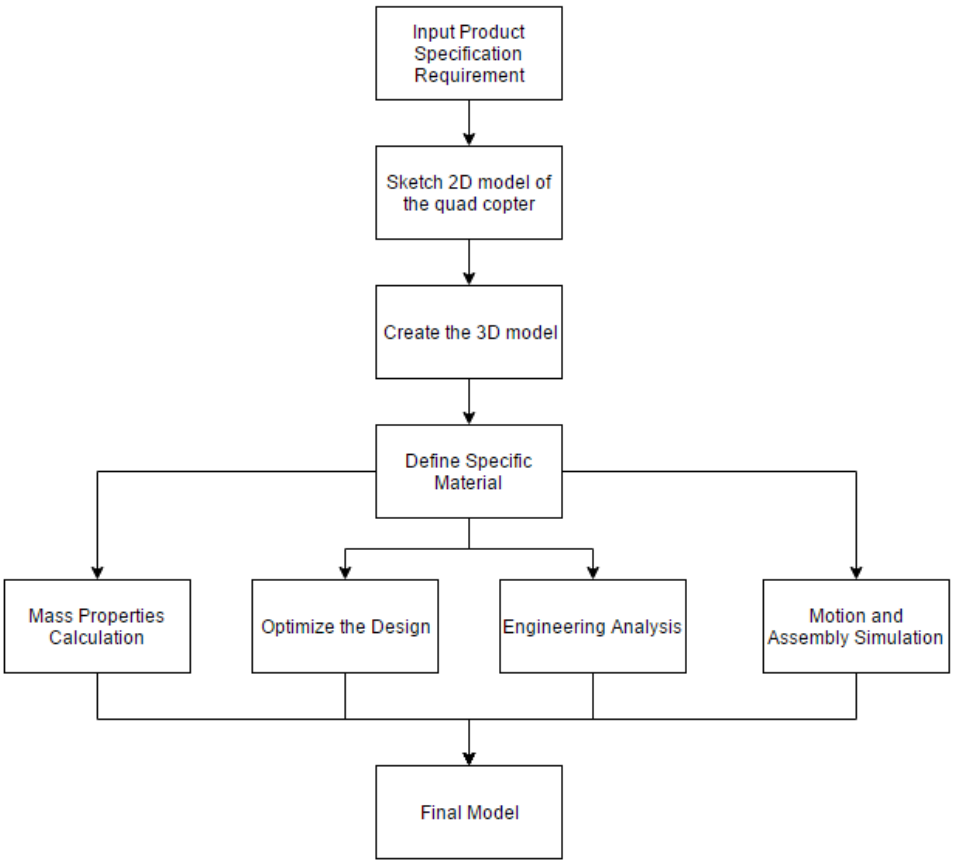


Figure 22 Modeling DJI F330 Quad Copter in SOLIDWORKS

7.1.1 Modeling the parts

Figure 23 shows the Clock wise and Counter Clock wise propellers

Figure 23 Clock Wise and Counter Clock Wise Propellers

Figure 24 shows Red and White Arms of the quad copter

Figure 24 RED and WHITE Arms

Figure 25 shows the Turnigy 1000KV rotor

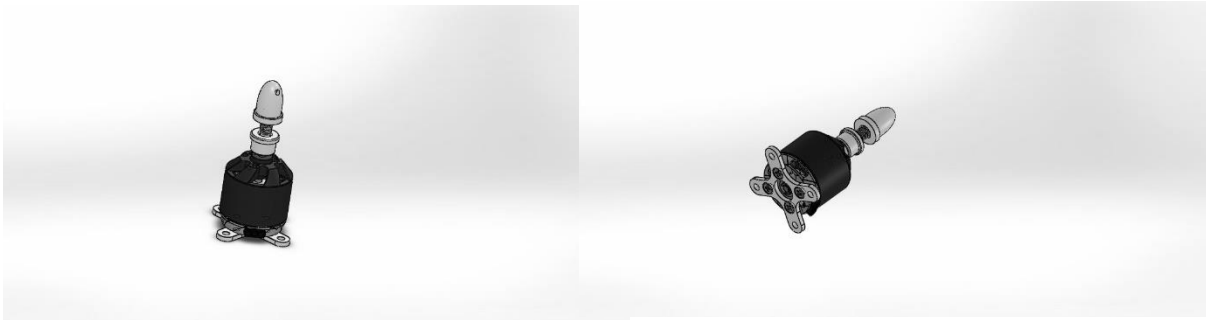


Figure 25 Turnigy 1000KV Rotor

Figure 26 shows the Upper and Bottom Frame

Figure 26 Upper and Bottom Frame

Figure 27 shows the Assembly of the Frame

Figure 27 Fully Assembled model of DJI f330

7.1.2 Mass Properties of the CAD model of DJI F330 Design

Table 7-1 Mass Properties of Solid parts

Part	Mass (g)	Length (mm)	Volume (cubic millimeters)	Surface area (square millimeters)	Center of mas (millimeters)		
					X	Y	Z
Propeller CW	6.11	76.2	5985.53	8577.78	0	1.29	0
Propeller CCW	6.11	6.2	5985.66	8577.86	0	1.29	0
ARM	14.23	330mm diagonal	5738.65	9135.40	0.02	78.87	0.15
Bottom Frame	104.68		42901.33	27311.80	0	2	0
Upper Frame	24.47		10027.77	11419.50	-0.01	0	1
Turnigy 1000KV rotor	81.33	69.04	12690.26	46867.11	0.01	16.5	0

Table 7-1 shows the mass properties given by SOLIDWORK 2015.

7.2 Mobile Platform Design

Following Figure 28 will explain the design methodology for the mobile robot. Work flow for this will be starting from three major parts. As in the chart shown below software, electronic and the mechanical are the three main branches. First the mechanical design step was taken into consideration. In this branch, the mechanical structure will be assembled and finalized. Then all the electronic components were placed whereas necessary. Finally, the programming part was implemented and downloaded to the MCU in the system which is to be executed.

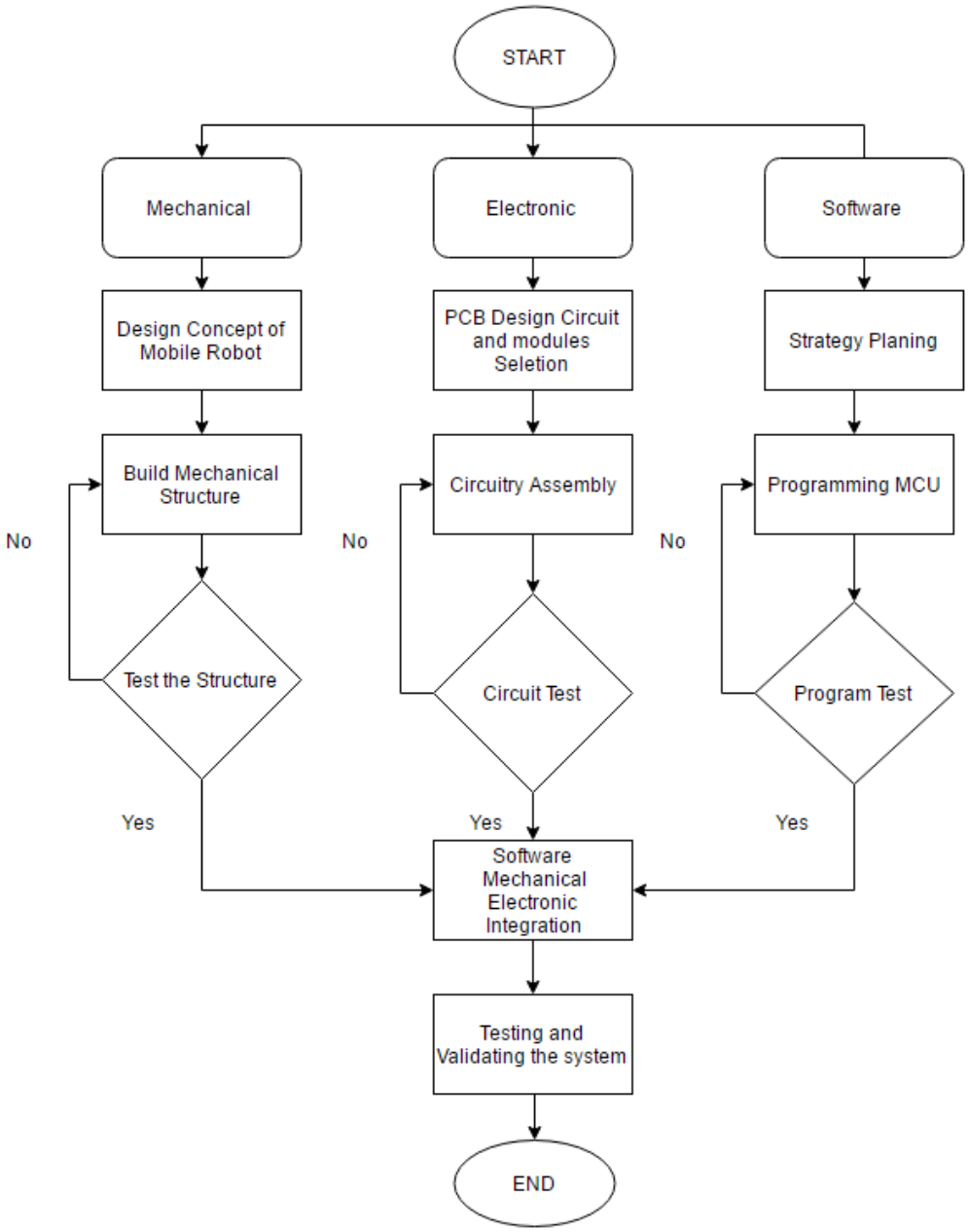


Figure 28 Mobile Platform Design Approach

7.2.1 Mechanical design

A durable mobile platform is very important part in robotics. Main objective of a base is to set a course the robot to the target. keeping in mind that the end goal to satisfy a task, the platform must be able to move the terrain required by the mission. The key components that permit a robot to navigate diverse landscapes include tracks, wheels, and flippers. Tracked bases have several advantages when navigating through unstable, rough terrains. As a result of multipurpose nature of versatile robots, tracks are frequently used to allow the robot to work in even the most extreme environmental conditions. Other mechanisms may also be included in the base to redistribute the weight allowing for heavy lifting. [14]

For the mobile platform, I chose a track driven system over wheeled system because wheeled system needs more than two gear motors. Also, wheeled system cannot stand on muddy surfaces as well as inclines. Sometimes it may lose its balance with its weight. One of the major advantages of a track driven system is that the large contact region with the surface being acrossed, effectively spreading the heaviness of the robot out over a substantial region so it does not sink into the surface. In fact, military robots put less force on the ground per square inch than a person's foot. As tracks give substantial advantages in soft, sloppy, sandy, rough, and loose terrain such as snow. [15]

They can also carry heavy loads which might sink on a wheeled vehicle. A famous example of this is the crawler-transporter used by NASA to carry space vehicles to the launch pad.in that case I thought a crawler robot is perfect choice for our project. Another advantage tank tread drives have is their continuous contact patch from front to back. This empowers them to traverse harsh territory without getting hung up or high-focused. Wheeled vehicles may wind up riding down into trenches or gaps that a tank tread drive will go directly over.

Wheeled vehicles have a little contact patch with the ground, though tank tread drive vehicles have a vast contact patch. Wheeled vehicles have awesome footing on hard surfaces like asphalt, yet on free surfaces the power of the wheel is connected to a little territory. In the event that the power is sufficiently awesome, it will bring about the ground to split away and the vehicle will lose footing. The tank tread drive applies the power over a bigger zone, and in that capacity the ground does not split away and the vehicle does not lose footing. For this reason, some vehicles rely on tank tread drives for higher traction. There are several designs of track driven systems like in the Figure 29 shown below. As per our requirement, we chose a triangle shaped track driven system. [16]

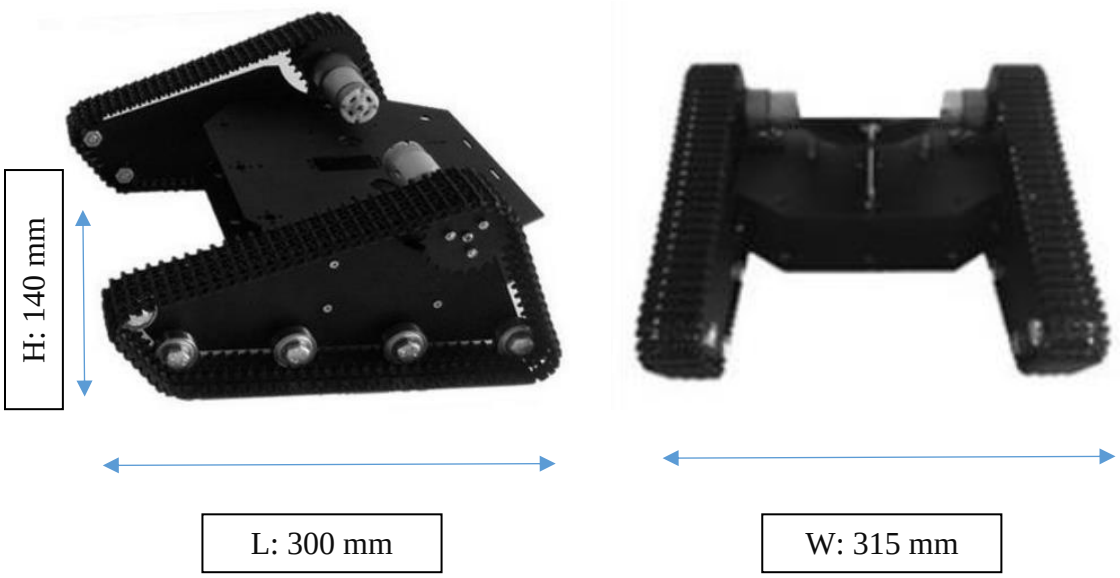


Figure 29 mechanical Construction

Components and the materials used are as follows:

1. M3 and M4 nuts and bolts
2. 2 mm Thick black aluminum board
3. Steel ball bearings (diameter: 2”)
4. M2 nuts and bolts
5. 4x saw tooth wheels
6. Reinforced nylon tracks
7. Steel spring washers
8. 6 1.5” hex nuts

Chassis description from the manufacturer [17]:

- Primary body is utilized as a part of all the hard 2mm thick aluminum composite material high hardness, quality regarding the steel material, and its light weight, the body to stay away from force utilization.
- Followed utilizing fortified nylon, can self-dismantling, simple to change the length of the track. Body appearance of the air is oppressive.
- Skeleton open, plentiful space, can self-controlled turntable base container, mechanical arms and other hardware.
- The body of the capstan motor imported metal gear motor, high speed.

- Ten steel wheel bearings, used to induce and support the track. Chassis body front left sensor mounting holes, can easily install sensors for secondary development.

7.2.2 Electronical design

- List of Modules and components used and their description as follows:

1. Arduino ATMEGA 2560

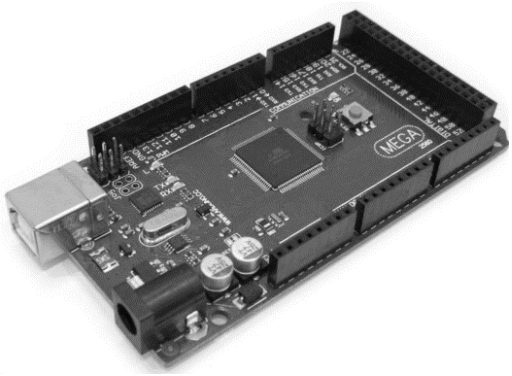


Figure 30 Arduino mega 2560

Figure 30 [18] shows an Arduino Mega 2560 development board. the Mega 2560 is a microcontroller board based on the ATmega2560. It consists with fifty-four digital input/output pins which fifteen of them can be utilized as PWM outputs. Six teen analog inputs (simple data sources), four UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB association, a power jack, an ICSP header, and a reset button. I used this MCU because of the vast range of availability and the support of resources. Since the quadcopter has the same MCU, communication between the two devices via long range RF made convenient for the entire operation.

Some of the technical specifications from the manufacturer as shown in the Table 7-2 below.

Table 7-2 Arduino Atmega 2560 specification

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm

2. Raspberry Pi 3 Model

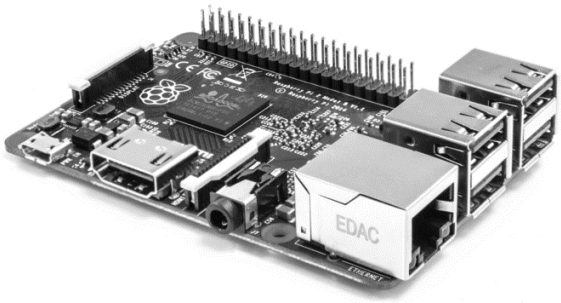


Figure 31 Raspberry Pi 3 Model B

The raspberry pi shown in the Figure 31 [19] is a progression of charge card measured single-board PCs. The first Raspberry Pi depends on the Broadcom BCM2835 framework on a chip (SOC), which incorporates an ARM1176JZF-S 700 MHz processor, Video Core IV GPU, and was initially sent with 256 megabytes of RAM, later overhauled (models B and B+) to 512 MB. The system comes with MicroSD sockets for boot media and persistent storage. The Foundation provides Raspbian ARM distributions for free download.

The top main reason was for using a raspberry pi in this project is the image processing part. As I mentioned above all the image processing part needs to be done by the mobile robot and send the positioning coordinates via RF link. The raspberry pi used here is the newest and the latest model, Raspberry pi 3 Model B. with compared to the previous versions of the Pi, this version comes with a 1.2 GHz 64-bit quadcore ARMv8 CPU. Like Pi 2 has, this model also has a 1 GB RAM,4 USB ports, 40 GPIO, Ethernet port, a HDMI port, CSI port, DSI port and VideoCore IV 3D Graphics core which is capable of faster performance in video processing applications. In our system, this mini PC known as to be the primary controller and Arduino mega is the secondary controller. Both MCUs communicate via UART link. All the image processing data will be sent to the Arduino and then transmit over RF link to the quad copter. [19]

3. LM298 motor driver

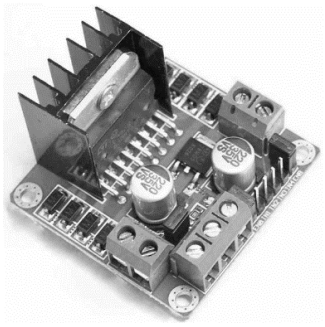


Figure 32 LM298 dual H bridged motor driver

Motor controllers are electronic components that specifically use for drive gear motors. Typically, they are connected between the 12v/5v power sources (battery) and the out device, controlled by a low power input PWM signal. Motor driver inputs low current input signal and provide high current power to the gear motors. This LM298 driver shown in Figure 32 [20] is capable of running continuous 2A through the motor with 12v / 5v. [20]

4. Ultra-sonic sensor HC-Sr04

Figure 33 HC-SR04 ultra sonic

These sonars are majorly used for obstacles avoiding purposes. The HC-SR04 ultrasonic sensor appeared in Figure 33 [21] utilize sonar to decide distance to an object like bats do. It gives incredible non-contact distance recognition with high precision and stable readings in an easy-to-use package. From 2 centimeters to 400 centimeters or 1 inch to 13 feet. It operation is not influenced by daylight or black material like Sharp rangefinders. It consists with ultrasonic transmitter and transceiver module. [21]

Features

- Power Supply: +5V DC
- Quiescent Current: <2mA
- Working Current: 15mA

- Effectual Angle: <15°
- Ranging Distance: 2cm – 400 cm/1″ – 13ft
- Resolution: 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mma

5. Ublox Neo-6m GPS module

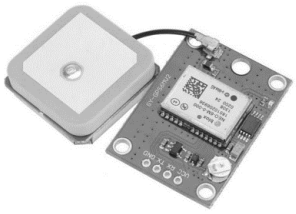


Figure 34 ublox GPS Module

Ublox Neo 6M (Ublox NEO6MV2) shown in Figure 34 [22] is a I2C compliant GPS module. This is mainly used for determine the current position of the mobile robot and to confirm the landing of the quadcopter. This is an Arduino compatible module with the accuracy of the 5m. some of the features for this module as follows: [22]

- UART, USB, DDC (I2C compliant) and SPI interfaces
- Available in Crystal and TCXO versions
- Onboard RTC crystal for faster warm and hot starts
- 1.8 V and 3.0 V variants

6. NRF24L01 RF module

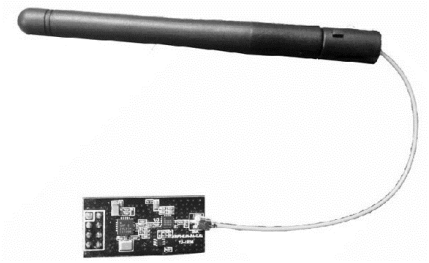


Figure 35 NRF24L01 RF Module

A quadcopter can fly up to several thousand feet from the ground level. In order to keep the communication between the two devices it is necessary to have long range and fast communication method added in the system. As per our requirement I chose this Arduino compatible Nordic NRF24L01 module. [23]

The Nordic nRF24L01+ shown in Figure 35 [23]. is a very coordinated, ultra-low power (ULP) 2Mbps RF handset IC for the 2.4GHz ISM (Industrial, Scientific and Medical) band. With pinnacle RX/TX streams lower than 14mA, a sub μA shut down mode, propelled control administration, and a 1.9 to 3.6V supply go, the nRF24L01+ gives a genuine ULP arrangement empowering months to years of battery life from coin cell or AA/AAA batteries. The Enhanced ShockBurst™ equipment convention quickening agent offloads time basic convention capacities from the application microcontroller empowering the usage of cutting edge and strong remote network with minimal effort outsider microcontrollers. The Nordic nRF24L01+ coordinates an entire 2.4GHz RF handset, RF synthesizer, and baseband rationale including the Enhanced ShockBurst™ equipment convention quickening agent supporting a rapid SPI interface for the application controller. No outside circle channel, resonators, or VCO varactor diodes are required, just an ease ±60ppm precious stone, coordinating hardware, and reception apparatus.

7. Accelerometer (ADXL335)

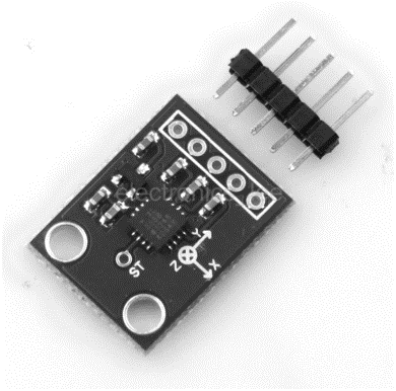


Figure 36 ADXL35 Analog IMU

The ADXL335 shown in Figure 36 [24] is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs. he item measures increasing speed with a base full-scale scope of ±3 g. It can gauge the static speeding up of gravity in tilt-detecting applications, and also dynamic increasing speed coming about because of movement, stun, or vibration [24]

8. 7.2v metal gear motors

Figure 37 7.2 V 291 rpm DC Gear Motor

The motors should be powerful enough to carry out over 2 Kg of weight. As shown in Figure 37 [25] high torque motors were selected because to extend the battery life and weight lifting operations.

Motor specs as follows.

- Weight = 4.22 oz. (119.63 g)
- Reduction = 30:1
- Stall Torque = 54.31 oz-in (3.91 kg-cm)
- Length (motor and gear) = 1.72" (4.37 cm)
- Length (shaft only) = 0.89" (2.26 cm)

- Diameter (motor and gear) = 1.45" (3.68 cm)
- Diameter (shaft) = 6mm
- Current (at 7.2v no load) = 200mA
- Current (at 7.2v locked shaft) = 3.80A

9. Pi camera module

Figure 38 FPV camera module

The Raspberry Pi FPV camera module as shown in Figure 38 [19] can be used to take high-definition video, as well as stills photographs. It's easy to use for beginners. There are lots of examples online of people using it for time-lapse, slow-motion and other video cleverness. one can also use the libraries we bundle with the camera to create effects. The module has a five-mega pixel fixed-focus camera that supports 1080p, 720p60 and VGA90 video modes, as well as stills capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi. The camera works with all models of Raspberry Pi 1, 2 and 3. It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Pi Camera Python library and OpenCV for image processing. This module is attach on the rear of the mobile robot to track down its trailer which is carrying the quadcopter using image processing. [26]

10. Optical encoders

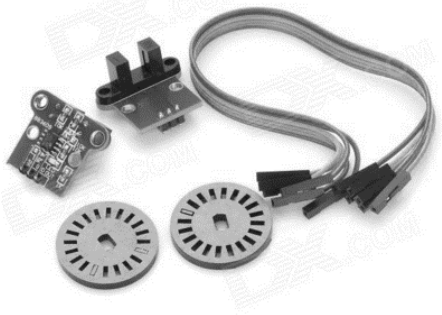
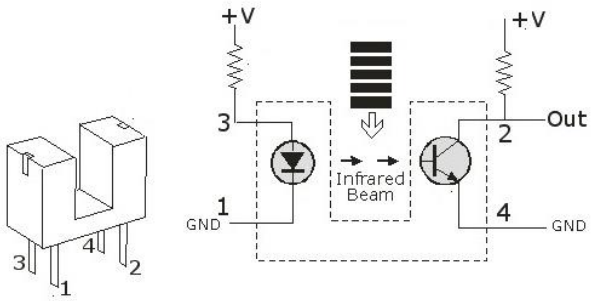


Figure 39 optical encoders

Nowadays, optical position encoders/rotary encoders as shown in Figure 39 [25] are widely used even in hobby robotics. One of common application is to control the DC motor position and the velocity. Our mobile robot equipped with two optical encoders to control the velocity.

Front end of a standard optical position encoder, utilized for these assignments, is an opened Photo Interrupter/Opto-Interrupter module with an IR LED and a Photo Transistor/Diode mounted confronting each other encased in plastic body. At the point when light discharged by the IR LED is blocked as a result of the rotating spaces of the encoder plate (otherwise called record circle), conduction level of the photograph transistor/diode changes. This change can be distinguished by a discrete equipment or by a microcontroller. To put it plainly, a photograph interrupter is made out of an infrared emitter on one side and an infrared indicator on the other by transmitting a light emission light from one side to the next, the photograph interrupter can distinguish when a protest goes between them, breaking the pillar [25]



As we need to create pulses, an encoder disc/index disc is very necessary here. The encoder disk that I have used contain 20 slots in it (saw tooth wheels). Two encoders were directly mounted to, two separate platforms below the both saw tooth wheels of the tracked system.

11. Hunger 3000 MAH Li-Po battery

Figure 40 14.8 V 4 cell 3000mah lipo battery

Power source should be powerful enough to handle the required maximum possible power from every electronic component. There are four common type of batteries used in mobile applications. The most common heavy duty battery that was research is Lead Acid battery. Nickel Cadmium batteries are a significant upgrade from lead acid batteries. When efficiency and light weight are of the utmost importance, metal nickel hydride batteries are the next step up. For the need of my required specifications I used the 4th type of the batteries as shown in Figure 40 [27]. That is Lithium polymer 14.8 V 3000mah 30C. Since its high capacity, they are used in many portable devices which has the ability of operating for hours.

12. PS2 wireless Controller

Figure 41 PS2 Dual Shock Controller

Every automated robot has a manual override function just in case of emergency operation purposes. In that case I have added a manual override controller for our mobile robot. A PS2 controller is shown in Figure 41 [14]. PS2 joystick controller is a Low cost, affordable and long range operations applicable controller with a frequency of 2.4 GHz. This controller can be directly plugged into the Arduino development boards. It contains 18 buttons which can be programmable up to 18 different functions.

7.2.2.1 Power distribution circuit design

After the completion of the mechanical design, next thing I focused was the power management system for the mobile robot. Since robot equipped with few sensors and two micro controllers it is really necessary to carry out a power management plan. This PCB contains two 10 A buck converter modules and two 5A buck converters. This regulator circuit converts the 14.8 V from the power source to several voltages. Main voltage divided into two at the beginning of the line and each line went through 10 A buck converter modules. The output from the converters 5V and 12V accordingly. Then again 5V line converted back to 3.3V line by the 5A buck converter. Finally, these 12V, 5V and 3.3V voltages then used to power up the motors, MCUs and low-powered sensors. In addition to that I have added two voltage indicator units to keep track of the battery voltage level and a cooling fan for cool down the system. The complete circuitry has shown in the Figure 42 below.

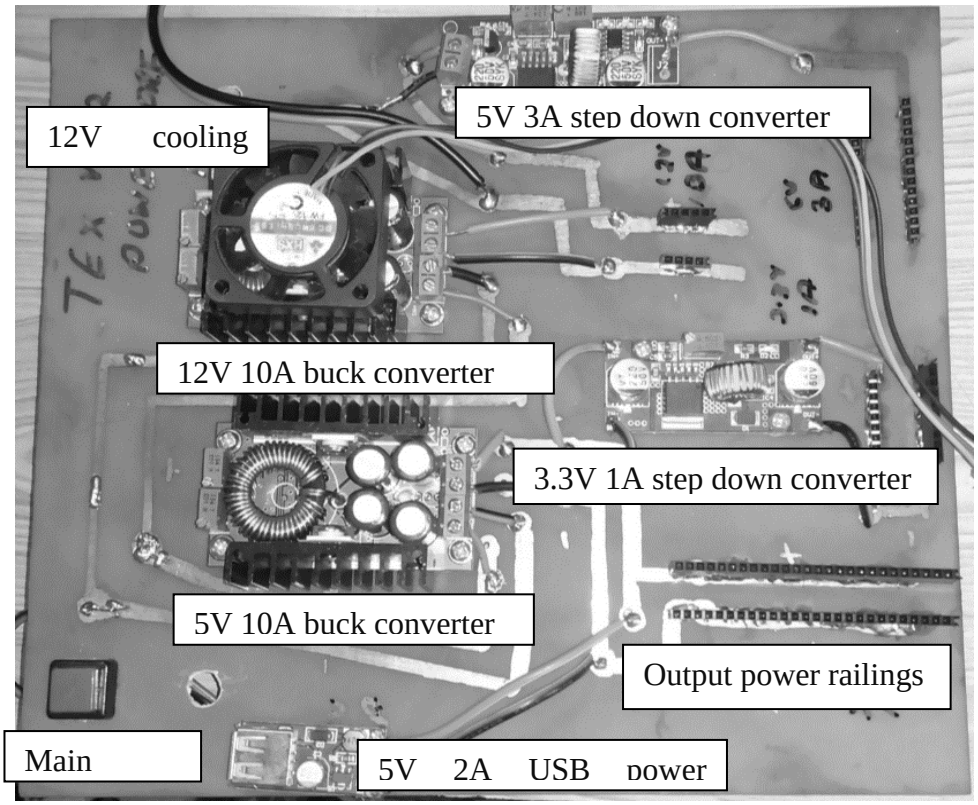


Figure 42 Main power supply circuit

7.2.2.2 Arduino Mega Pin configuration and wiring

Table 7-3 Pin Configuration with Arduino

Module	BUS type	Operating voltage (Vcc)	Pin Assignment
NRF24L01 RF module	SPI	3.3 V	MISO – 50 MOSI – 51 SCK – 52 SS – 53 CE – 8 CSN - 7
LM298 motor driver	Digital I/O PWM	12 V	Dir pin 1 – 9 Dir pin 2 – 10 Dir pin 3 – 11 Dir pin 4 – 12 ENBA – 5 ENBB – 6
MPU6050 IMU	I2C	3.3 V	SDA – 20 SCL – 21
Raspberry Pi 3	UART	5 V	TX0 – 1 RX0 – 2
Optical Encoders	Digital I/O (Interrupts)	5 V	Right – 20 Left – 21
Ublox NEO-6m GPS	UART (Software Serial)	3.3 V	TX – 18 RX – 19
PS2 Receiver	Digital I/O	5 V	PS2_DAT – 30 PS2_CMD – 31 PS2_SEL –11 PS2_CLK –12

7.2.2.3 **Finalized design of the Mobile Robot with its Landing Platform**

Figure 43 shows the finalized design of the ground station with its trailer (landing platform). See APPENDIX B for the component placement of the ground station and the design of the Landing platform step by step.

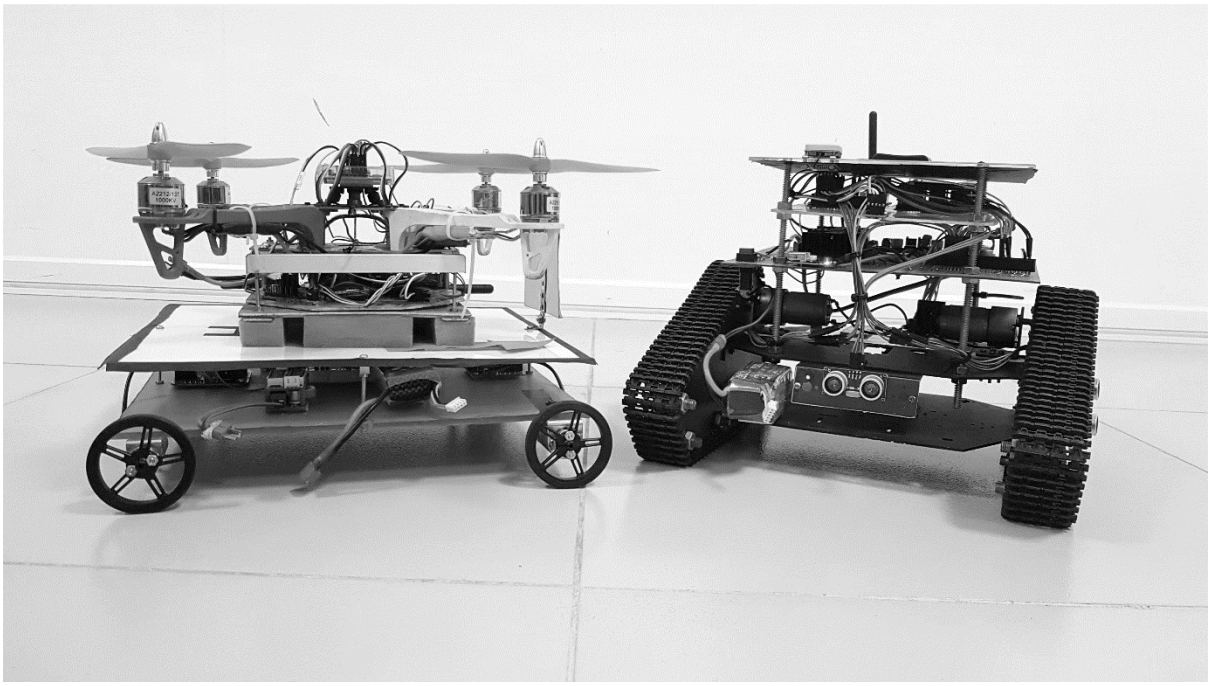


Figure 43 Finalized design of the mobile robot

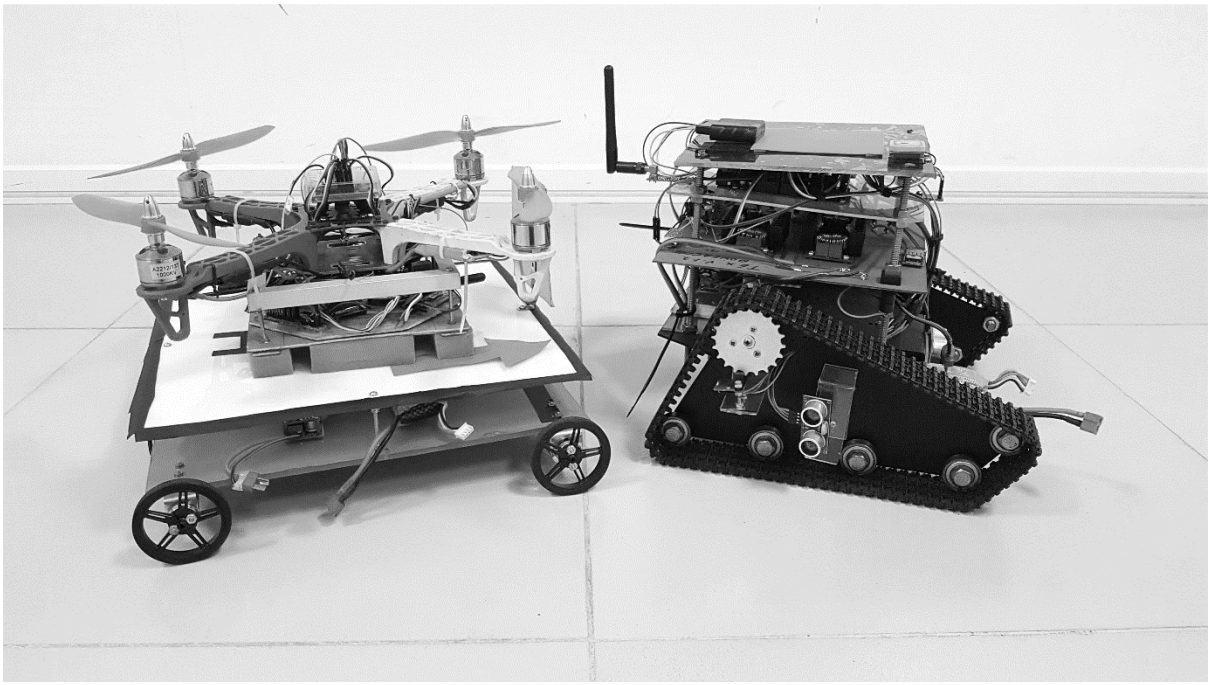


Figure 44 finished prototype

7.3 Software Simulations for Ground Station

7.3.1 A PID controller for straight course

In Robotics, especially in mobile robots using independent wheel control, applying the same power to each wheel generally does not result in the mobile platform moving straight. This is caused by mechanical and surface differences experienced by each of the wheels. In our case, straight movement of the robot is a must. Otherwise quadcopter will not able to land properly on to the landing platform. To reduce deviation in the robot trajectory, a better approach is to use a controller which can adjusts the power applied to two motors based on the difference in their rates of rotation. MATLAB & Simulink provide Arduino Mega 2560 compatible tool kit for test and correct the Rotation difference in wheels' Real time. With the aid of the tool kit following simulations were carried out. [27]

The actual system can be modeled as in the Figure 45 below.

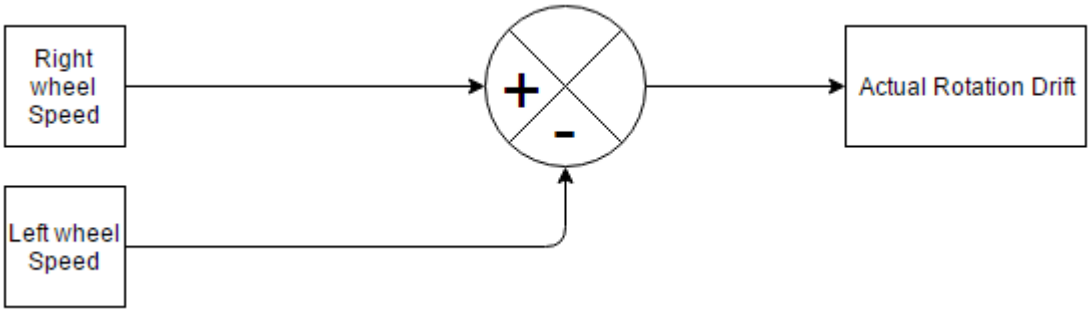


Figure 45 Actual System Response without PID

Speed (PWM) of the two motors are mapped into 0 – 100 scale. Normally the PWM value range is 0 -255. As for the experiment purpose, Initially I have applied same PWM value of 50 to the left and right wheels as shown in the figure. Then the real-time simulation run for 10 seconds. Actual rotation deviation observations are shown in the result section. The resultant Spectrum shows the left and right deviation of the mobile platform. The positive side shows the right wheel deviation and negative side (below zero) shows left wheel deviation. It can be seen that the left wheel has deviated than the left one. Which means that the platform is not moving straight but to the left direction. In order to fix this error I have added PID controller to the system.

PID controller MATLAB Simulink configuration as shown in the Figure 46 below:

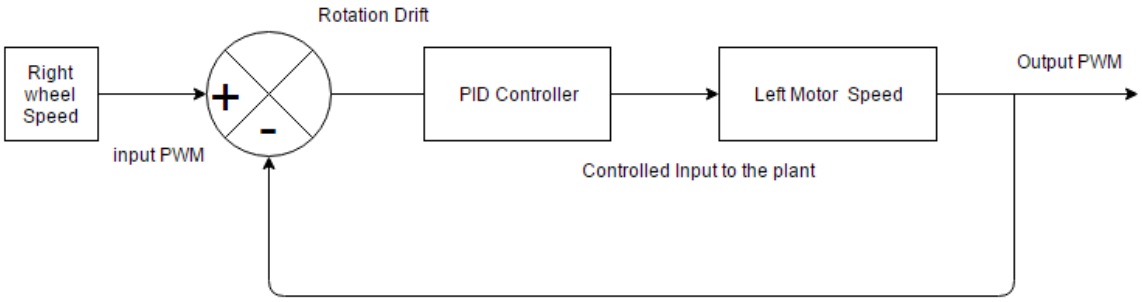


Figure 46 Actual System with PID Controller

What happen here is the optical encoders calculate the rotation difference between the right and left motors per revolution. Rotation difference (error) then fed back to the PID control block. PID block determines the maximum deviation of the right motor and then adjust the PWM applied to the left motor while keeping the right motor speed (PWM) constant. For different PID values the response of the spectrum become obvious. Simulation results can be seen in result section. Refer [APPENDIX A] for Implementation in MATLAB Simulink.

7.3.2 GPS (Global Positioning System)

Two GPS modules were used to track down the location of each devices. Before we integrate GPS to the mobile robot and to the quad copter we carried out a GPS test. For this simulation, we directly used the U-center IDE which come along with the Ublox NEO-6m series GPS modules. The IDE can simply monitor the longitude and the latitude of GPS equipped device. Most of the time accuracy for indoor is limited. normally NEO-6m series has 5m of accuracy even in indoors. So, we put our GPS module on to a test. When the system goes online the received data was plotted in several windows by the IDE shown in the Figure 47 [22]. Those windows can be seen in the result section.

Figure 47 U-Center IDE

7.3.3 Kalman Filter for Linear Velocity determination

It is really necessary to be equal the linear velocities of the two robots when landing. a simplified one dimensional Kalman filter implementation on Arduino along with ADXL335 analog sensor is used to obtain the Acceleration and the linear velocity of the mobile robot. The accelerometer puts out three-dimensional acceleration data (Zout, Yout and Xout). Since the sensor is placed to Y direction, in that case acceleration of Y was considered for determination of the linear velocity of the mobile robot. Resulting data graph of acerbation against velocity can be found in the result section.

Figure 48 shows the implementation of the Kalman filter on Arduino to obtain the linear velocity of the mobile robot.

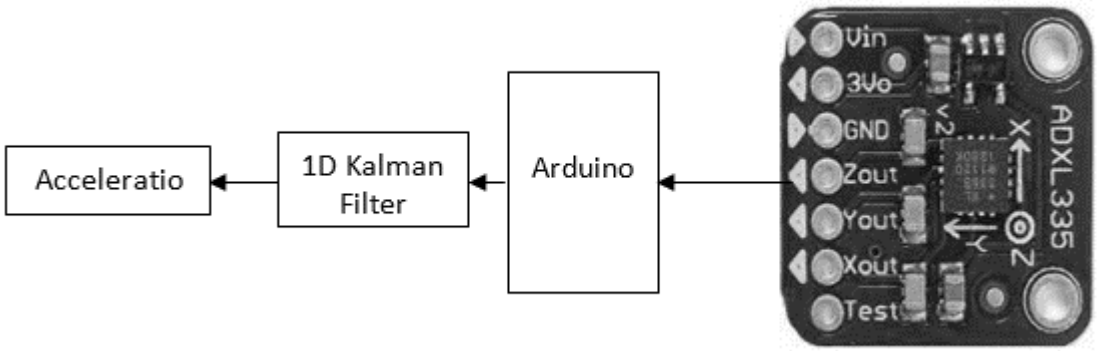


Figure 48 Kalman filter Implementation

7.4 Image processing Algorithm implementation

Our main objective is to takeoff off the quad copter from the mobile robot, follow the landing platform for few meters and landing onto the same mobile platform. After the few seconds of takeoff, quad copter needs to track down the pre-defined marker which is on the landing platform (trailer of the mobile robot) by itself autonomously. In order to achieve this goal I was assigned to implement an image processing algorithm.

Algorithm specifications as follows:

1. Track Specific Object / marker
2. Calculate its Angle
3. Determine the Tracked Object Orientation

The Figure 49 shown below depicts the approach and the Methodology of the processing procedure.

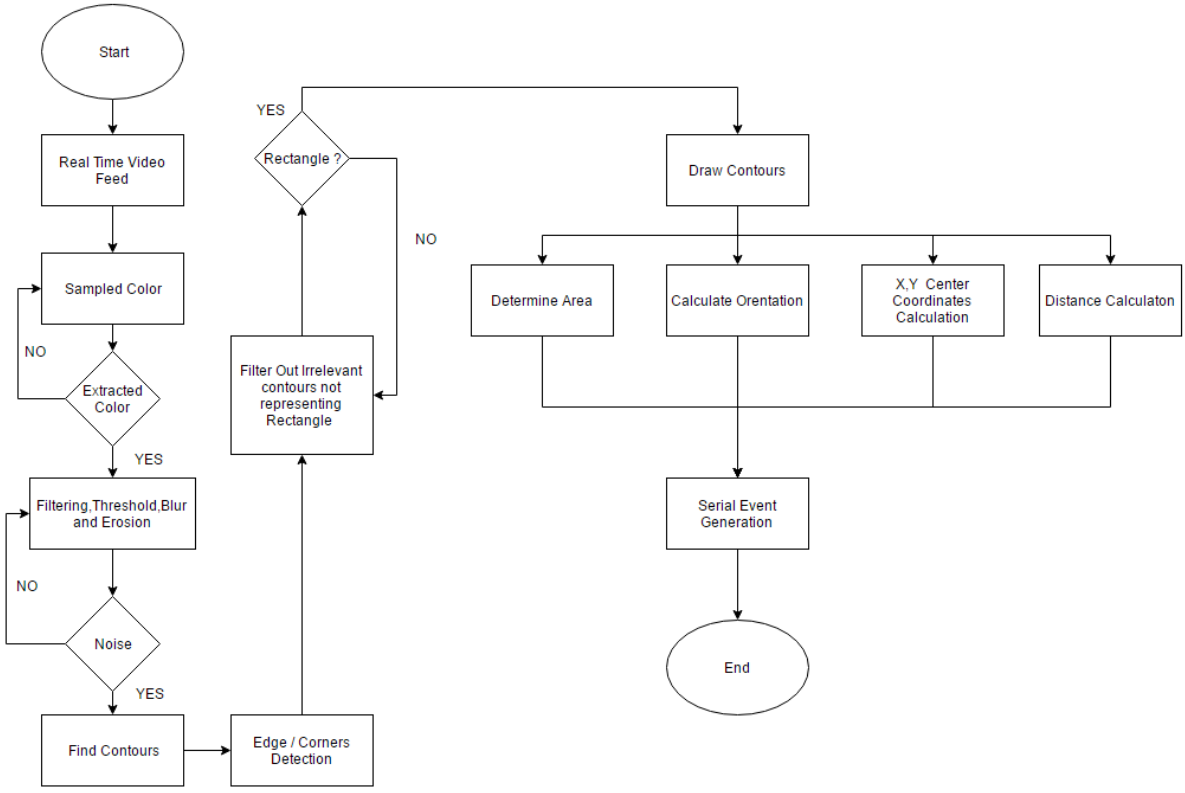


Figure 49 Methodology of Image Processing Algorithm

Programming language used:

- Python 2.7

  Python is an effective yet simple to-utilize programming dialect created by Guido van Rossum, initially discharged in 1991. With Python, you can rapidly compose a little venture. Be that as it may, Python likewise scales up pleasantly and can be utilized for mission-basic, business applications. The real objective of any programming dialect is to conquer any hindrance between the developer's cerebrum and the PC. The greater part of the famous dialects you've most likely known about, as Visual Basic, C#, and Java, are viewed as abnormal state dialects, which implies that they're nearer to human dialect than machine dialect. What's more, they are. Be that as it may, Python, with its unmistakable and straightforward tenets, is considerably nearer to English. Making Python projects is so clear. Python is sufficiently intense to draw in designers from around the globe and in addition organizations, for example, Google, IBM, Industrial Light + Magic, Microsoft, NASA, Red Hat, Verizon, Xerox, and Yahoo!. Python is likewise utilized as a device by expert diversion software engineers. Electronic Arts, 2K Games, and the Disney Interactive Media Group all distribute diversions that join Python. Python is protest situated programming dialect too. In Python, utilizing OOP strategies is discretionary. [28]

Software packages used:

- Eclipse IDE with Python 2.7 Interpretation

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages through the use of plug-ins including python. [29]

- OpenCV 2.4.11 Library

OpenCV (Open Source Computer Vision) is a library of programming capacities principally gone for continuous PC vision, created by Intel Russia investigate focus in Nizhny Novgorod, and now bolstered by Willow Garage and Itseez. It is free for use under the open-source BSD permit. The library is cross-platform. It concentrates for the most part on continuous image processing. On

the off chance that the library discovers Intel's Integrated Performance Primitives on the framework, it will utilize these exclusive upgraded schedules to quicken it. [30].

Hardware component used:

- Wireless FPV camera
- Wireless Receiver RC305
- EasyCap USB video converter

Other Software used:

- Dvdriver for convert FPV to USB
- VNC server for remote operation Raspberry Pi for remote access. 4

Figure 50 shows FPV camera placement of the Quad copter:

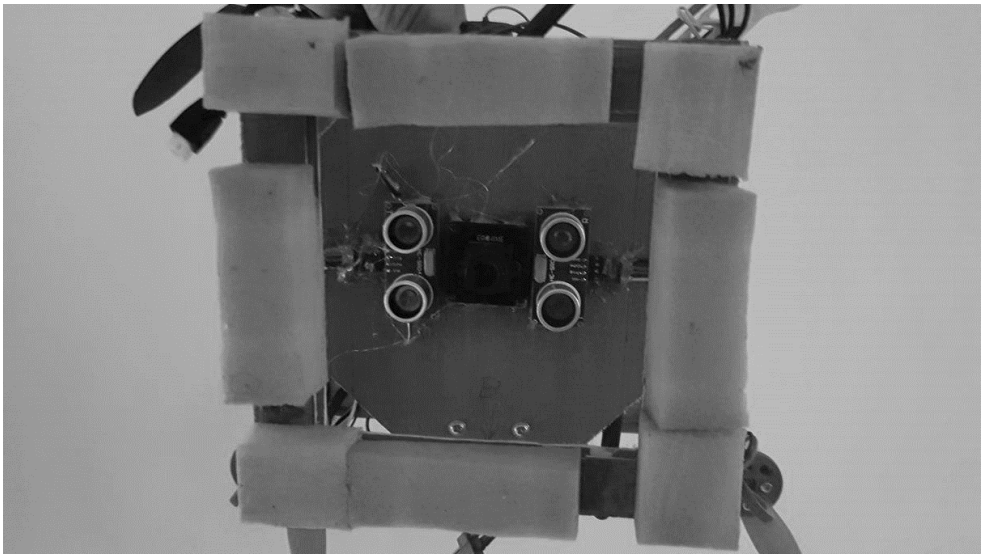


Figure 50 FPV Camera Placement

Note: *for complete algorithm please refer [APPENDIX C]. and for the usage of OpenCV 2.4.11 libraries and Python 2.7 syntaxes please refer Reference guides in reference section.* [31] [32] [30]

7.4.1 Pre-defined marker tracking and following algorithm

A Red color square shape sticker is glued on the landing platform of the mobile robot to identify as the pre-define marker. Then the FPV camera module mounted under the quad copter body as in the Figure 50. Since this is a wide-angle camera the distorted images can be expected. Therefore, first we calibrated the camera and calibrated results as follows.

fx = 511.16718125 Cx = 108.8047538]

fy = 536.68575864 Cy = 64.70238472]

; where fx, fy are camera focal lengths and Cx, Cy are optical centers.

distortion coefficients: (k1 k2 p1 p2 k3)

[-5.03769208e-01 1.03407209e+00 4.70793382e-04 -4.45855637e-03 7.63882593e-01]

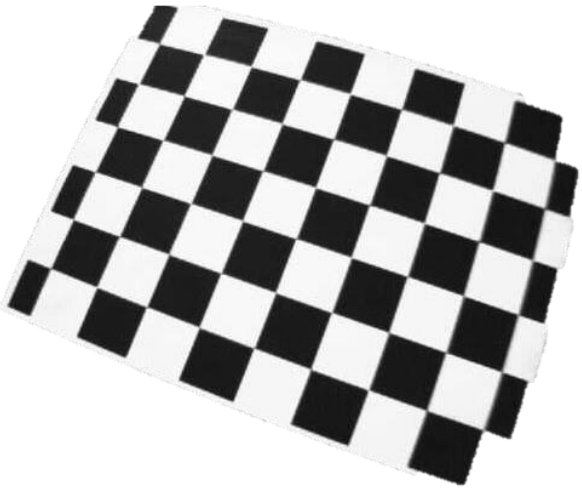


Figure 52 Before Calibration

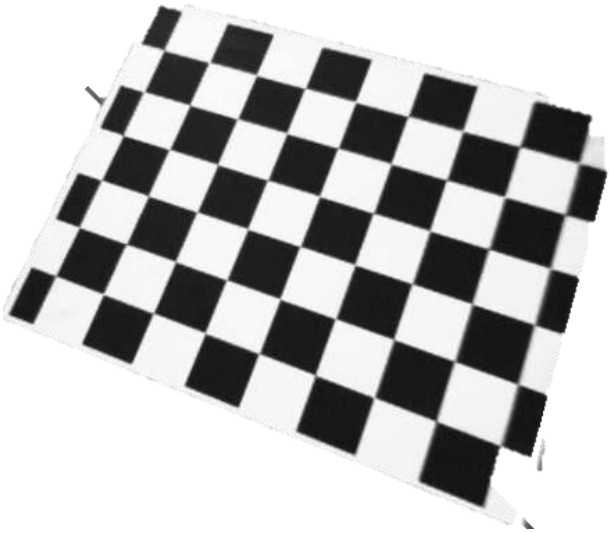


Figure 51 after calibration

As you can see in the Figure 52 a bevel shaped (fish-eye) distortion can be seen. This a common problem in wide angle cameras. Figure 51 shows the un-distorted image. A clear change can be seen after calibration. All Curved edges have been reconstructed in 2D plane.

Now we are all set to start the processing. As shown in the Figure 49 Real time video feed then fed to the camera. First it transforms the incoming image frames color scheme into HSV range. Using track bars as shown in the Figure 58, we fine-tuned the HSV range color we wanted. In this case the color was red. Then the extracted color filtered using a Gaussian filter to eliminate the noise. Then these frames are thresholded. Thresholded frame then put into morpho graphical transformation for eliminate further noise. Here incoming frames filter out 5 times to observe the exact color we wanted. After eliminating the noise, we found the contours of the extracted object. Then performed a canny edge detection operation to obtain the edges and corners of the object. Now we found the edges. As a geometrical property of square, square shape must have four edges with equal length in it. If the object has 4 edges and equal length of contours surrounded, then it confirmed that the square is detected. After confirming the detected object is a square then algorithm draw contours on real time feed as shown in Figure 57. Then the properties such as area, aspect ratio, solidity and perimeter are calculated. Center coordinates (X, Y) and area then transmit via Serial Link to the quad copter to keep its position steady. [See APPENCIX C Step 1- Step 12]

7.4.2 Determining YAW Angle

Suppose the mobile platform is travelling through a curved path. When the gradient of the path increases the maker of the object also start to relatively move in a curved path. In that case, since our quad copter not going to make any curved moves, the probability of missing the object becomes high. To overcome this issue, we attached an arrow shaped marker on the landing platform to determine the turning angle of the mobile robot. This will make easy to keep track of the mobile robot. In the same algorithm, while keeping track of the red colored square it will also track down the orientation of the landing platform autonomously by controlling its YAW according to the angle. Arrow shape definitely contains 7 edges in it. This property is used to find the contours of the Red colored arrow. The Triangle head of the array kept vertically to observe the reference point of the angle. Which means zero degree. Then by using fit ellipse () function in OpenCV the orientation was determined.

[See Appendix C Step 14 – Step 15]

7.4.3 Distance Determination using Triangle similarity

For quadcopter landing process, the area of the red color shaped square was taken into account. As a property of image processing science, when an object is closer to the capturing device the area of the object becomes larger than the actual one. therefore, triangle similarity was used to determine the distance between the quadcopter and the mobile platform. [33] In order to determine the distance from our camera to a known object or marker, we need to utilize triangle similarity. I place our landing platform in some distance *D* from our camera. then took a picture of our marker using our camera and then measured the apparent width in pixels *P*. This allows us to derive the perceived focal length *F* of our FPV camera:

Equation (17) expressed below represents the Perceived Focal length of the camera.

$$F = \frac{(P\ X\ D)}{W} \tag{17}$$

Equation 7:1 Perceived Focal length

When the known D = 30 cm, and the actual width of the marker W = 10 cm the perceived width of the paper was 225.166 pixels.

Then focal length,

F = (225.166px 30cm) / 10 cm = 675.498

After determine the Focal length of the FPV camera, I repeat the same procedure to calculate the distance to the object by changing the camera position for different heights. Then the distance was tabulated in a table. [See APPENDIX C Step 13]

Equation (18) expressed below shows the distance to the object,

$$D' = \frac{(W\ X\ F)}{P} \tag{18}$$

Equation 7:2 distance to the object

8. Results

8.1 PID tuning Results vs Actual Rotation Drift

Figure 53 Shows the Actual systems real time response from MATLAB Simulink.

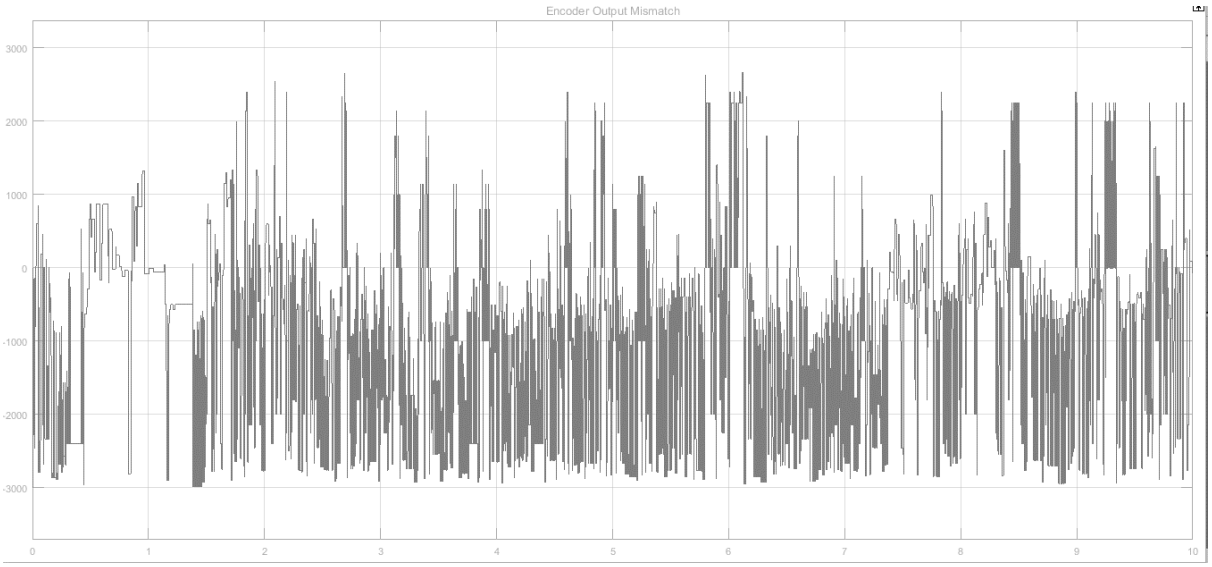


Figure 53 Actual System Response

Figure 54 shows the Actual system with PD controller real time response when

P = 0.89 D = 1.43

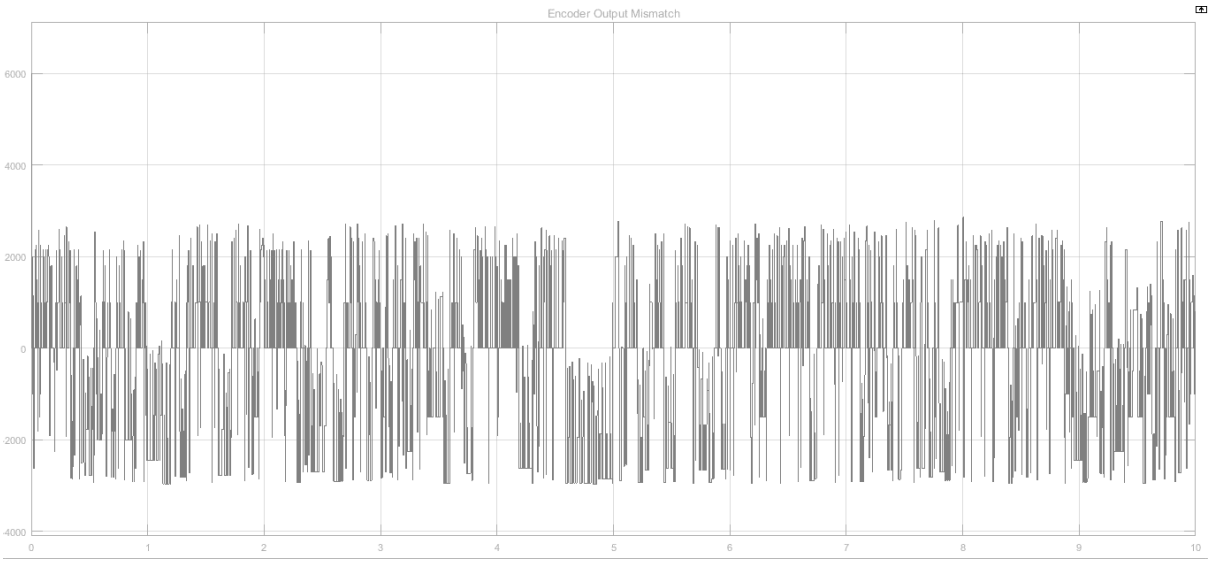


Figure 54 System Response with PID Controller

8.2 GPS test results

Figure 55 shows the GPS deviation map given by the U-center IDE.

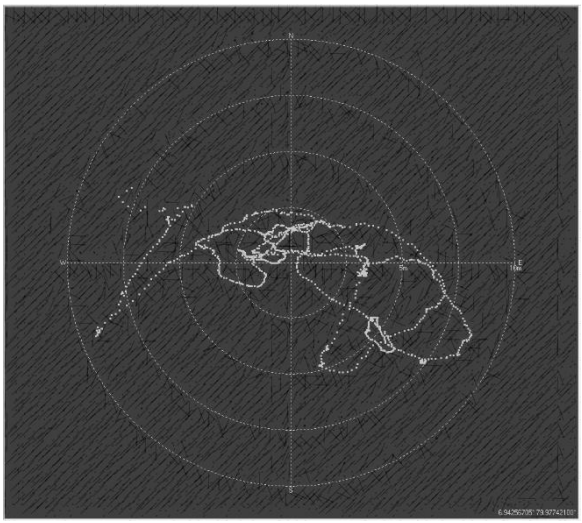


Figure 55 GPS deviation map

8.3 Kalman Filter Results

The Figure 56 depicts the acceleration vs velocity of the mobile robot. It's clear that the unnecessary noise has been suppressed by the Kalman Filter.

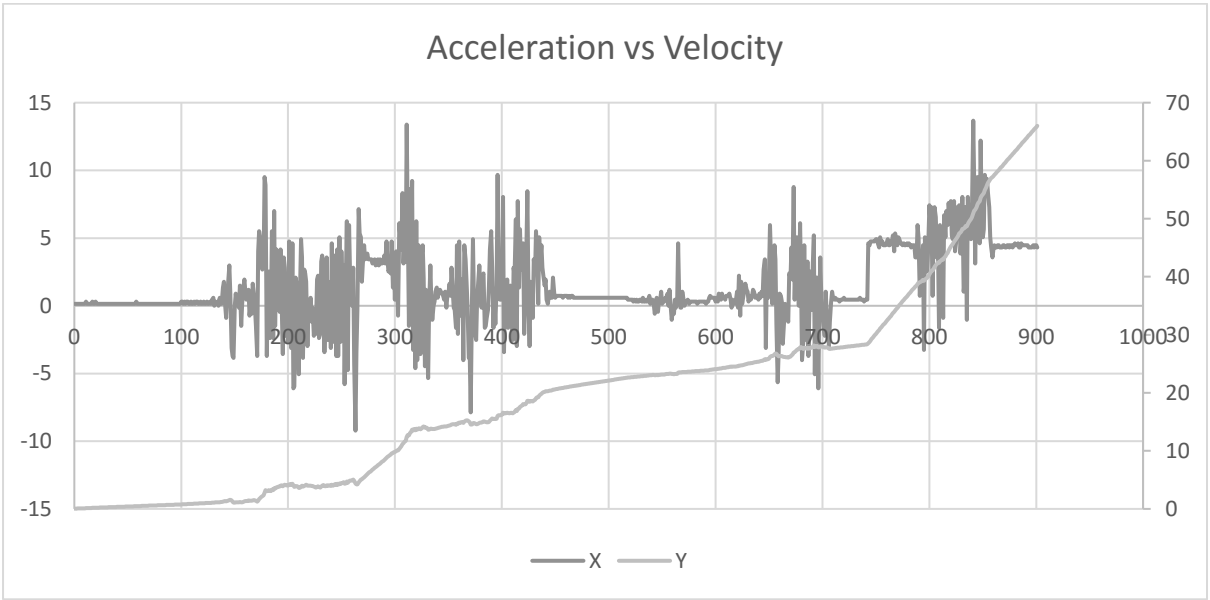


Figure 56 Kalman Filter Results

8.4 Pre-Defined Marker tacking results

Figure 57 shows the results of pre-defined marker tracking algorithm.

(a) Logitch c270 HD webcam (b) FPV camera

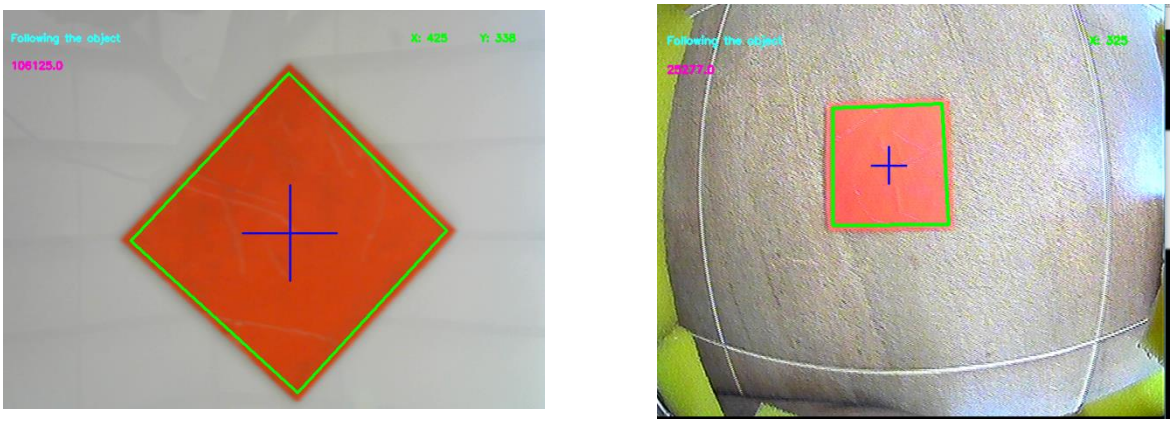


Figure 57 Image Processing results from normal camera and a wide-angle camera

Figure 58 shows the Real time HSV Tuning GUI for image processing Algorithm

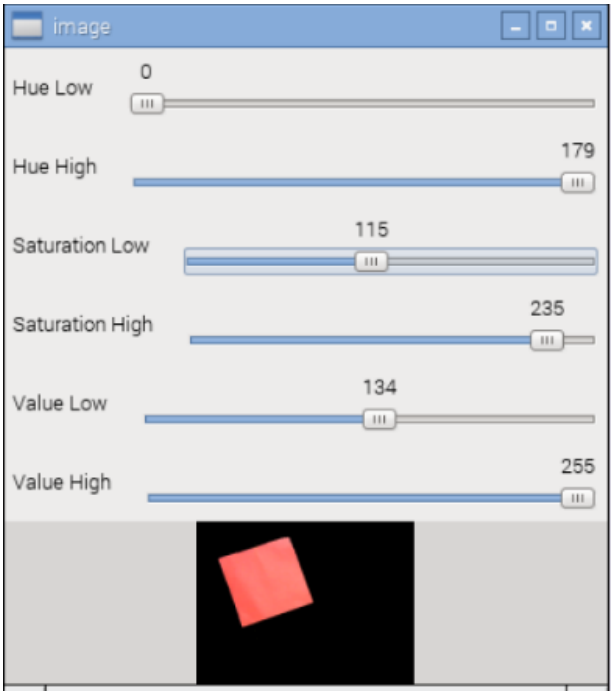


Figure 58 Real Time Color Tuning GUI

8.5 YAW Angle Determination Results

Figure 59 Shows the Angle Determining Results from the Algorithm.

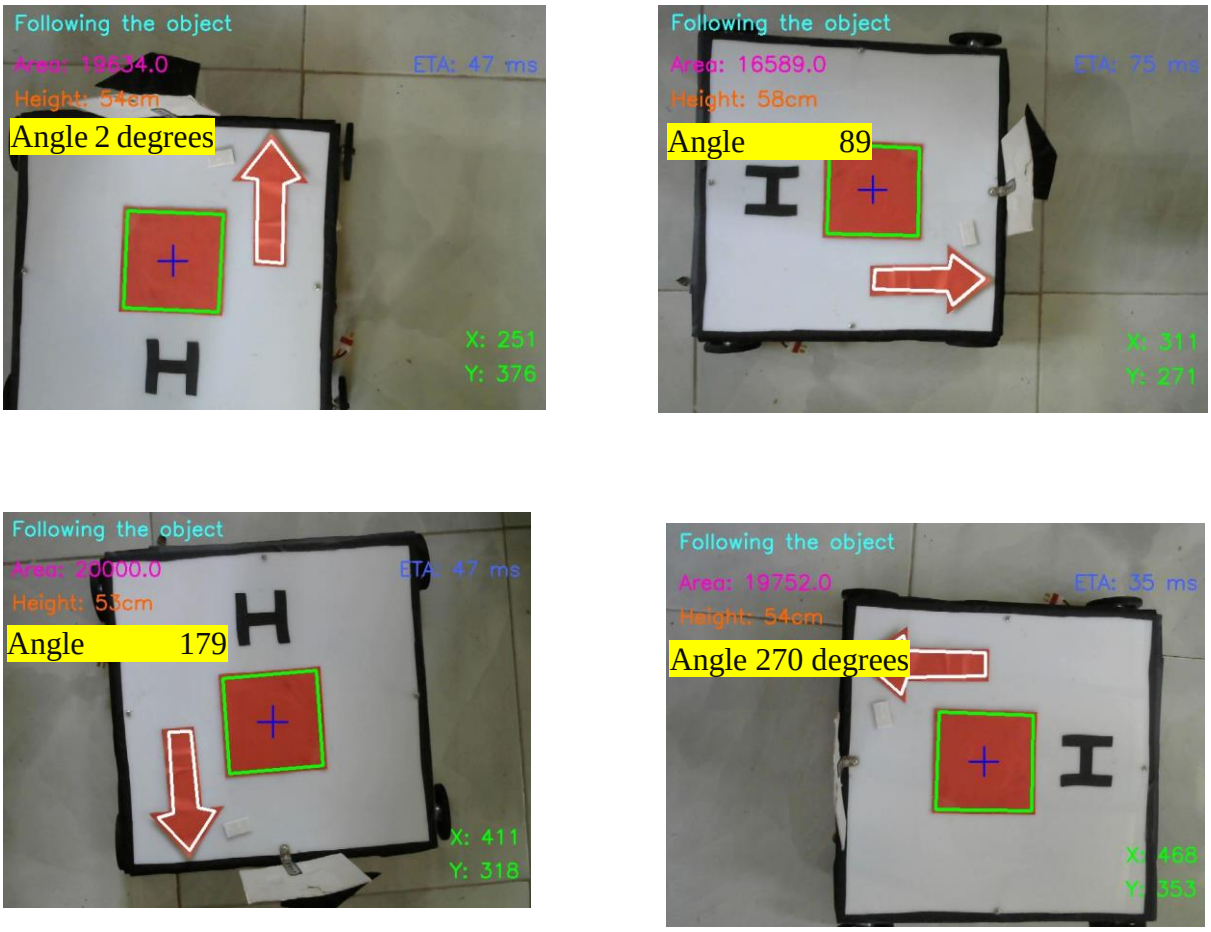


Figure 59 Angle Determination Algorithm Results

8.6 Distance determination Results Using Triangle Simillarity

Figure 60 Shows the Distance Calculation Results using Triangle Similarity.

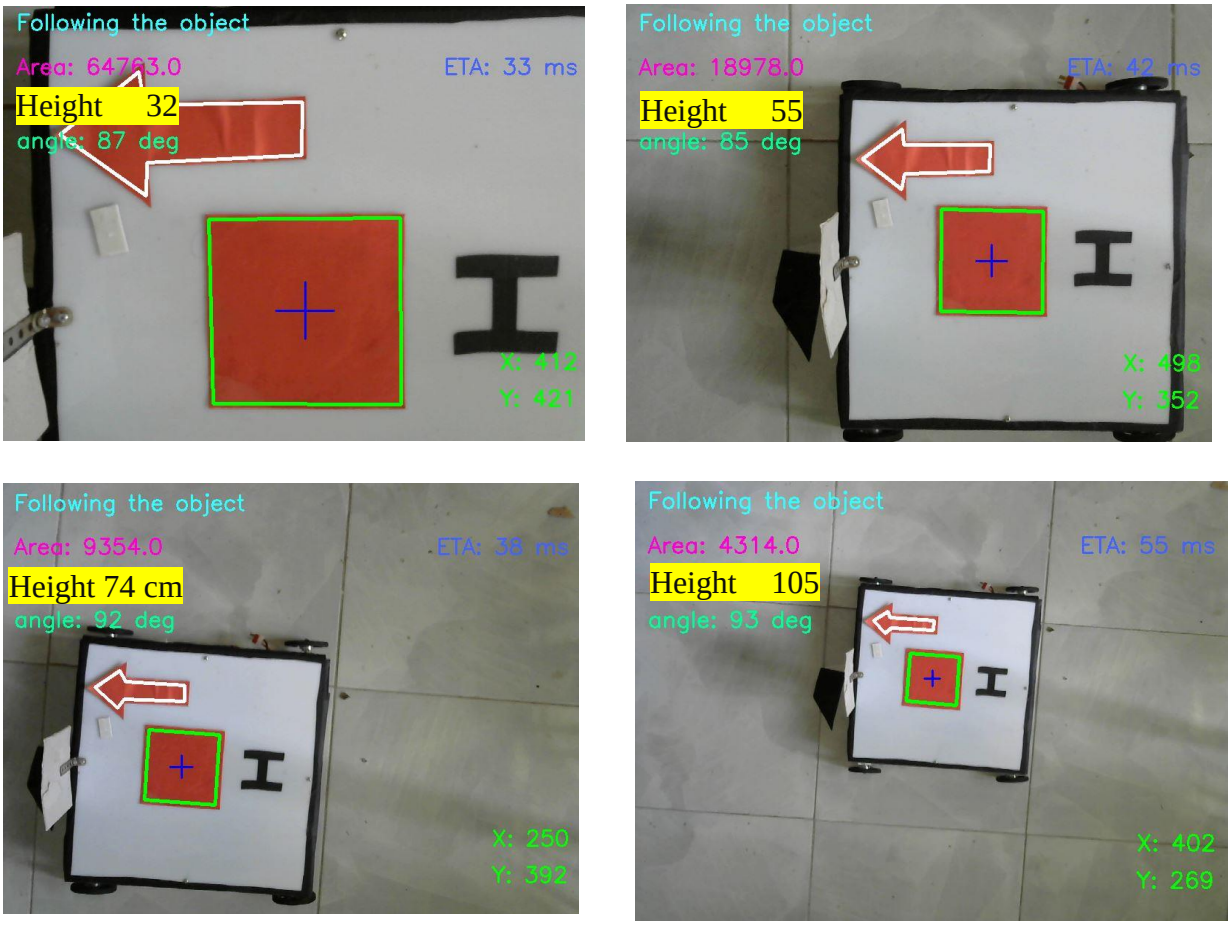


Figure 60 Distance Calculation Results

Table 8-1 Analytical Distance to the Object.

Table 8-1 Analytical distance

Perceived Focal Length	Perceived Width in pixels	Actual Width in centimeters	Area in Square pixels	Distance to the object in cm
675.498	225.1666	10	50700	30.00
675.498	134.1641	10	18000	50.34
675.498	88.31761	10	7800	76.49
675.498	64.03124	10	4100	105.495

Figure 61 shows the comparison between distance vs perceived width of the object.

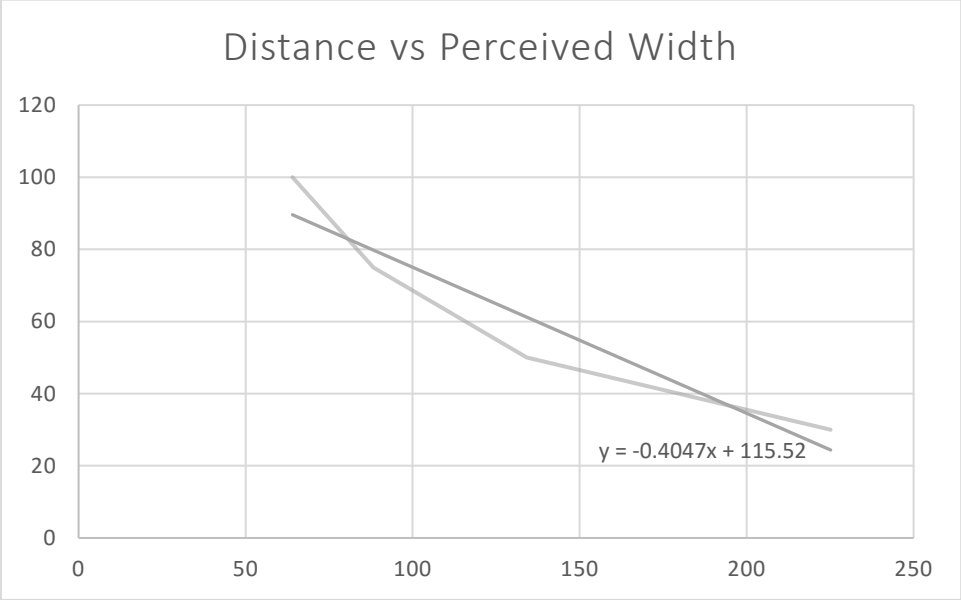


Figure 61 Distance vs Perceived width

Figure 62 shows the comparison between height vs area vs perceived width.

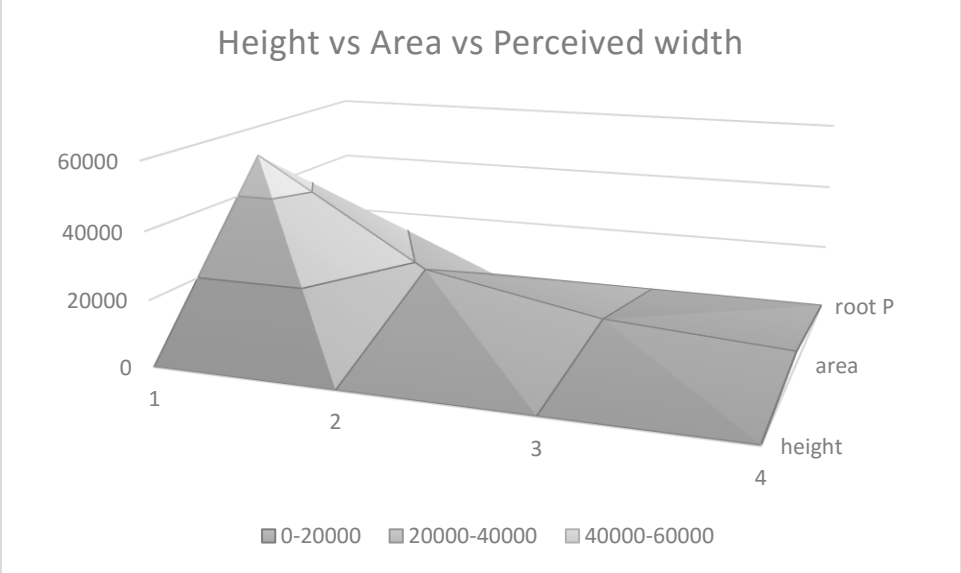


Figure 62 Height vs Area vs width

8.7 Discussion of results

8.7.1 PID Tuning for Straight course

The experiment is carried out for analyze the motor rotation difference of the mobile robot. The need of this experiment arose when we try to automate the ground station. our mission is to land a quad copter on an autonomous robot while moving on a straight line. however, the mobile robot should be able to keep its movement in that straight direction autonomously. Otherwise landing of the quad copter would be endangered. In that case, as shown in section 7.3.1 a closed loop controller (PID) was implemented. As in the section 8.1 with the PID controller for values of P = 0.89 and D 1.43, we were able to reduce the deviation between motors. Resulting graphs in section 8.1 shows that left motor has a higher rate of rotation drift and it caused to turn the mobile robot to a one side. With the PD controller, the distribution around the origin is symmetrical. Which concludes that the Rotation Difference Corrected by the PD controller. As a result, the actual ground robot manages to keep its movement nearly in a straight-line direction.

8.7.2 GPS

Throughout the initial design process, we have come up with several alternative solutions to various aspects of our project. One solution that we have considered is not to use GPS for the navigation of the mobile robot and the quadcopter. The reason for this is that GPS modules in the market are cannot be used for the distance range less than 5m. and the results shows that the accuracy for indoor usage is very low. Also, GPS not works for everywhere. Therefore, we cannot rely on controlling a quad copter with GPS. Once the GPS signal is lost, quad copter will be out of control to the ground station.

8.7.3 Image Processing Algorithm

A fully completed computer vision based algorithm was developed for autonomous quad copter which gives the abilities of following and landing on to a mobile platform autonomously. Algorithm was tested in both intel 2.9 GHz 64-bit core i5 CPU (PC) and a 1.2 GHz 64-bit quadcore ARMv8 CPU (Raspberry pi 3). In algorithm, both serial and the processing part took 40 milliseconds per frame on PC and 180 milliseconds on raspberry pi 3. With compare to PC, the raspberry pi 3 takes a lot of time for handling the processing part. From the testing results we observe that, for automation it is necessary to reduce time per execution cycle of the code. The minimum we could achieve was 40 milliseconds. That made a great change in handling quad copter autonomously.

We noticed that sometimes the crosshairs and bounding box regions of the detected target tend to “flicker”. This is because many computer vision and image processing functions are very sensitive to noise, especially noise introduced due to the motion and the vibration of the quad copter, a blurred square can easily start to look like an arbitrary polygon, and thus our target detection could be fail. Another thing is that the HSV value ranges are varying due to different weather conditions. For same color, we had to change HSV values frequently. Use of track bars made our work ease. Since we used a FPV wide angle camera, sometimes it fails to detect the square shape properties of the marker due to its wide-angle view. the marker looks like a rhombus or rectangle when quadcopter moveout from the target. Even though the color is visible to the computer vision, target acquiring might fails. To overcome this problem, we tried blob detection method. However, blob detection python script has been able to successfully detects the target in every possible term.

In the distance measuring part, when we captured the photos, our measuring tool had a bit of slack in it and thus the results are offset by roughly 1 mm to few centimeters. we also captured photos rashly and not 100 % on top of the marker which added another error to our calculations. This could cause varying results between actual and the measured heights. A comparison between Measured and the actual results are tabulated in Table 8-2.

Table 8-2 Comparison Actual vs Measured heights

Actual Measurement (cm)	Experimental Results (cm)
30	30.00
50	50.34
75	76.49
100	105.495

8.7.4 Design Constraints and Feasibility

Through the design process we were limited by several different types of constraints. The first of these constraints is budget. We did not have a fixed limit on how much we can spend for this project. However, we justified each purchase by making sure that it is the most cost effective part and that it is necessary to reach the final goal. I have been asked to keep the project to an approximate cost of five hundred dollars.

Another constraint of this project is Time. We only had 10 months to complete the final product. Because of that we had to be careful in how far we can achieve the overall project objectives. Therefore, we limited our project to take off and landing parts. Major effect of our time constraint is ordering parts online and purchasing high quality products. Since we had a fixed budget we ordered low cost affordable DIY parts. Those parts cannot be used for commercial range products. Also, Due to shipping delays, we had to be waited couple of times.

The technical scope and the lack of prior knowledge is yet another possible constraint. This is partially related to the time constraints as well. Since we are on a limited schedule, a project that is too difficult and too technically challenging would have been a problem in progress evaluation criteria. However, our project is challenging enough and we did a quite a lot of work throughout this project. Each member’s contribution is admirable.

9. Costing

Table 9-1 shows the Costing for mobile robot implementation.

Table 9-1 Costing

No.	Item	Qty.	Price(Rs.)
1.	Raspberry PI 3 model B	1	7850.00
2.	Arduino ATMEGA 2560	1	2000.00
3.	2A power supply	1	600.00
4.	Micro SD card 16GB	1	1065.00
5.	Logitech c270 HD webcam	1	2100.00
7.	Multifunctional servo bracket	1	275.00
17.	Robot chassis	1	18494.00
20.	LM298 Stepper motor driver	1	708.62
22.	5A power supply module	2	770.00
23.	12A power supply module	2	1584.32
24.	Cooling fan	1	136.00
25.	3.6” TFT touchscreen display for Raspberry pi 2	1	878.00
26.	USB power module	1	100.00
27.	Wi-fi adapter		590.64
28.	PS2 Controller joystick	1	1420.00
30.	TTL Logic converter circuit	1	150.00
31.	Digital DV 3.3-30V Red LED Indicator	2	300.00
32.	3000mah LIPO battery		3990.00
33.	Imax B6 charger		4000.00
34.	Wireless RF UART module	1	6000.00
35.	Analog accelerometer	1	180.00
36.	Others	N/A	1500.00
Total			54691.58

10. Conclusion and further work

10.1 Conclusion

This project is aimed towards design and develop of the Quadcopter with automated takeoff and landing on mobile robots Landing platform. Mobile robot and the Landing Platform designs were duly completed. The ground station is fully functioned to both autonomous and manual drive. Due to Time constraints, we decided not to use autonomous part for the quadcopter following part. As for the reason, we were unable to completely finish the axis movement controller for the quadcopter. Despite that we only did the take-off and landing parts. Implementation of the autonomous ground station and a Real-Time image processing (including: tracking, following, yaw controlling, distance measuring and landing) algorithm for a quad copter are the key results have been achieved so far.

Since the start of the implementation of the Image processing algorithm, we had to face a lot of problems. Some of them were solved. The raspberry pi 3 on the mobile robot did not give us the expected outcome of the image processing algorithm. We found out that it's processing power is not powerful enough to handle both serial event and the processing part at the same time. Therefore, we carried out testing on a PC. As we used a wireless FPV camera, the compatibility with the raspberry was raised another major problem. At the moment, we're trying to reduce the weight of the quad copter, so mounting a compatible wireless device would give additional weight which exceeds the weight lifting limit of the quad copter. The main challenge within the visual tracking process is that the environmental conditions because the system perform its tasks in dynamic environments. Changes in environmental conditions altered the HSV values significantly. However, with the Implementation of online HSV tuner the problem seems to be solved. When we were capturing the live video feed from the FPV camera, we identified that some frames were lost during hovering period. Since the FPV cameras are wide angle and wireless capturing devices we assumed that the reason for the loss might be signal drop or the resolution. but the actual issue was the fish eye effect of the camera module. Due to the fish eye effect the edges and the length of the square shape object got distorted. When the camera move to a corner of the defined resolution the probability of distortion seems to be very high. Therefore, camera calibration had to be done to eliminate and reconstruct the images that we want to have for the processing.

as for the conclusion, its clear robotic vision systems that are using image processing is a novel approach for replacing sensors that have been used in today robotics. One advantage is that the cost can be reduce by replacing sensors such as distance, thermal and color. But there more drawbacks exist because technology for the robot vision is not become common and yet it is in experimenting phase. In our research, we tried to achieve a goal by using image processing techniques to a system that consists with a mobile robot with a quadcopter. Dully completed algorithm is ready for the further work. This algorithm now can be used to automate our quadcopter. The experimental results demonstrate that the methodology use throughout this project nearly converged to achieved the main objectives of proposed project.

10.2 Further work

As For my further work, there are much work to be considered. There are number of possible Upgrades that can be made to improve the robot which makes the system more advance for do lot more operations. Because of the Time constraints and financial difficulties, I couldn't able to achieve some of objectives of this project which are optional. This implementation consists with separate controller for image processing and another one for hardware component control. Actually, this is not a professional implementation after all. Even though I used raspberry pi 3 for image processing still it has some lags. Therefore, I recommend a SOC FPGA development as for replacement of the two micro controllers. SOC FPGA are next level microprocessors which are thousands of times power full than a normal single chip processor. one could do image processing algorithms while having sophisticated; enhanced MCU design on the other side of same system. Another upgrade can be done to the system is a battery charging unit for the quad copter. Since quad copters are very high in power consuming it would be a better concept to implement such a unit on the mobile robot. For Ideal real time simulation of the quadcopter ArUCO algorithm can be use. Furthermore, an online automated camera calibrating algorithm would do a great job in different environmental conditions.

References

[1] R. M. a. F. R. Vincenzo Lippiello, "Visual Coordinated Landing of a UAV on a Mobile," PRISMA Lab – Department of Electrical Engineering and Information Technology, Naples, 2013.

[2] M. Shneier and R. Bostelman, "Literature Review of Mobile Robots for Manufacturing," Intelligent Systems Division Enginnering Laboratory, National Institute of Standards and Technology, unted states of America, 2015.

[3] M. 30091762, "The Very First Bomb Disposal Robot," Military.com Networ, 15 jan 2014. [Online]. Available: http://www.military.com/video/ammunition-and-explosives/explosive-ordnance-disposal/the-first-bomb-disposal-robot/3059244734001. [Accessed 14 9 2016].

[4] B. BAKER, "Analysis the features of bomb disposal robots evolution and-revolution," ARPA, 2014. [Online]. Available: http://www.army-technology.com/features/featurebomb-disposal-robots-evolution-and-revolution/. [Accessed 14 9 2016].

[5] L. S. D. A. E. Ser-Nam Lim, "A Scalable Image-Based Multi-Camera Visual Surveillance System," University of Maryland, College Park,Computer Vision Laboratory,UMIACS, Maryland.

[6] A. B. a. D. F. Jean-Christophe Zufferey, "Optic flow to control small UAVs," Laboratory of Intelligent Systems, Lausanne, Switzerland, 2011.

[7] N. Instruments, "PID Theory Explained," 29 mar 2011. [Online]. Available: http://www.ni.com/white-paper/3782/en/. [Accessed 15 09 2016].

[8] M. Anderson, "The Kalman Filter as State Observer," in *Optimal Filtering*, Dover publications, 2005, pp. 1-5.

[9] R. Szeliski, "Morphology," in *Computer Vision: Algorithms and Applications*, Russia, Springer, 2010, pp. 127-132.

[10] o. d. team, "Morphographical transformation," OpenCV, 23 Nov 2014. [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/opening_closing_hats/opening_closing_hats.html. [Accessed 15 Sep 2016].

[11] sonic0002, "Gaussian Blur Algorithm," pixelstech, 24 Nov 2012. [Online]. Available: http://www.pixelstech.net/article/1353768112-Gaussian-Blur-Algorithm. [Accessed 15 Sep 2016].

[12] o. d. team, "gausian median blur bilateral filter," OpenCV, 23 Nov 2014. [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html. [Accessed 15 Sep 2016].

[13] S. Price, "The Canny Edge Detector," ICBL, 4 Jul 1996. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MARBLE/low/edges/canny.htm. [Accessed 15 Sep 2016].

[14] M. &. Robotics, "Mobility Robotics," Jet Propulsion Laboratory, Califonia Institute of Technology, 23 Aug 2015. [Online]. Available: http://scienceandtechnology.jpl.nasa.gov/research/research-topics-list/spacecraft-technologies/mobility-robotics. [Accessed 17 Sep 2016].

[15] M. W. Carey, "Novel EOD Robot Design with a Dexterous Gripper and Intuitive Teleoperation," Worcester Polytechnic Institute, 2011.

[16] K. Kurc and D. Szybicki, "Kinematics of a Robot with Crawler Drive," Rzeszow University of Technology,Department of Applied Mechanics and Robotics, Lodz, 2012.

[17] ebay, "ebay," ebay.inc, [Online]. Available: http://www.ebay.com/itm/Robo-Soul-TK-100-Black-Crawler-Robot-Chassis-Triangle-Mobile-Platform-/332039107583?hash=item4d4f12b3ff:g:MDoAAOSwLnBXWlxX. [Accessed 18 04 2015].

[18] "ARDUINO & GENUINO PRODUCTS," [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardMega2560. [Accessed 14 Sep 2016].

[19] "Raspberry Pi 3 - Model B - ARMv8 with 1G RAM," [Online]. Available: https://www.adafruit.com/product/3055.

[20] STMicroelectronics, "LM298," STMicroelectronics, Italy, 2000.

[21] SparkFun, "Ultrasonic Sensor - HC-SR04," Sparfun.

[22] Ublox, "NEO-6 series,Versatile u-blox 6 GPS modules," [Online]. Available: https://www.u-blox.com/en/product/neo-6-series. [Accessed 14 Sep 2016].

[23] N. SEMICONDUCTORS, "nRF24L01 product specidications," NORDIC, 2016.

[24] "Analog Devices ADXL335".

[25] H. T.K., "Optical Position Encoder with Arduino," Electro schematics, 2008. [Online]. Available: http://www.electroschematics.com/10494/arduino-optical-position-rotary-encoder/. [Accessed 17 Sep 2016].

[26] R. Pi, "Raspberry Pi Learning Resource," [Online]. Available: https://www.raspberrypi.org/learning/getting-started-with-picamera/.

[27] "Drive with PID control," MAthWorks.Inc, 2016. [Online]. Available: https://in.mathworks.com/help/supportpkg/arduino/examples/drive-with-pid-control.html?prodcode=SL. [Accessed 17 sep 2016].

[28] P. S. Foundation, "Python 2.7," [Online]. Available: https://www.python.org/. [Accessed 30 05 2015].

[29] eclipse, "eclipse," [Online]. Available: http://www.eclipse.org/.

[30] O. d. team, "OpenCV 2.4.11," OpenCV.org, [Online]. Available: http://docs.opencv.org/2.4.11/.

[31] A.Rosebrock, "Finding targets in drone and quadcopter video streams using Python and OpenCV," 2015. [Online]. Available: http://www.pyimagesearch.com/2015target-acquired-finding-targets-in-drone-and-quadcopter-video-streams-using-python-and-opencv. [Accessed 3 july 2016].

[32] G. A. Stafford, "Object Tracking on the Raspberry Pi with C++, OpenCV, and cvBlob | Programmatic Ponderings," wordpress, 2013. [Online]. Available: https://programmaticponderings.wordpress.com 2013opencv-and-cvblob-with-raspberry-pi/. [Accessed 3 june 2016].

[33] A. Rosebrock, "Find distance from camera to object/marker using Python,Triangular Similarity and OpenCV," pyimagesearch, 19 Jan 2015. [Online]. Available: http://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/. [Accessed 30 05 2015].

[34] O. Liang, "quadcopter beginners guide learn to fly drones," Dronethusiast, 2016. [Online]. Available: http://www.dronethusiast.com/quadcopter-beginners-guide-learn-to-fly-drones/. [Accessed 14 09 2016].

Appendix A

1. Actual System Model Without PID Controller.

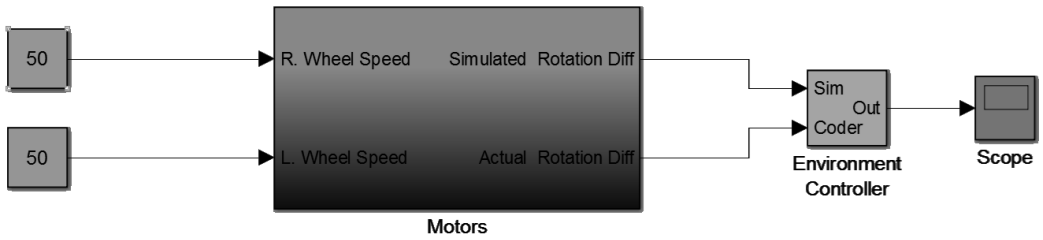


Figure 63 Simulink Model of Actual System

Motors Block Expanded -

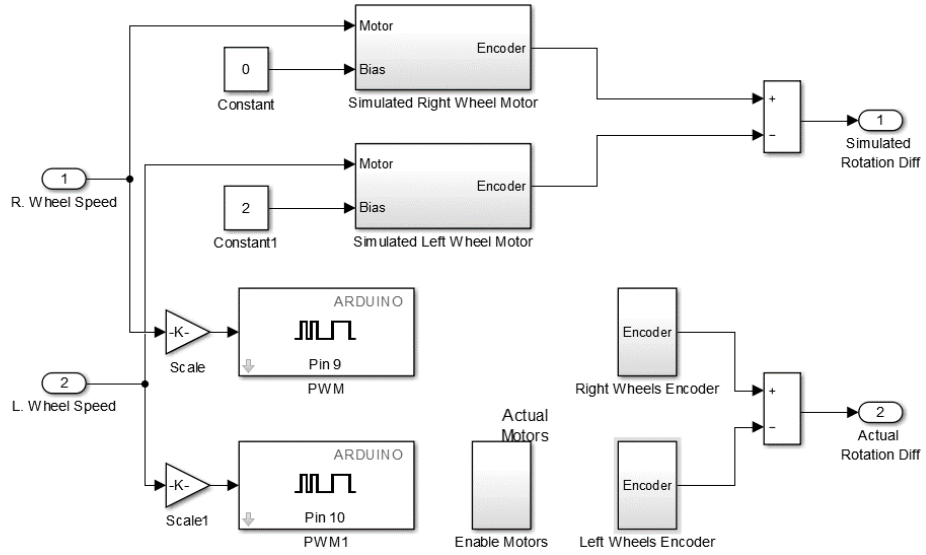


Figure 64 Expanded Simulink Model

2. Actual System Model with PID Controller.

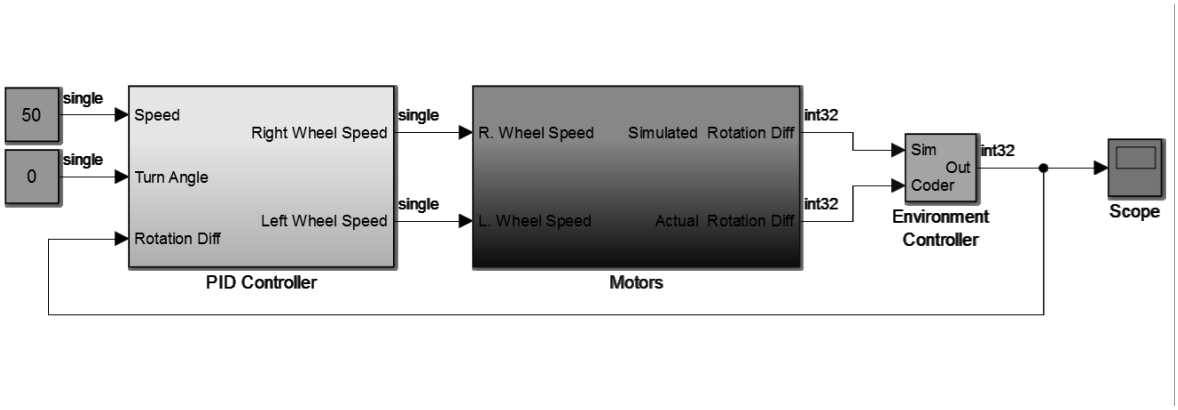


Figure 65 Simulink PID Model

PID Block Expanded -

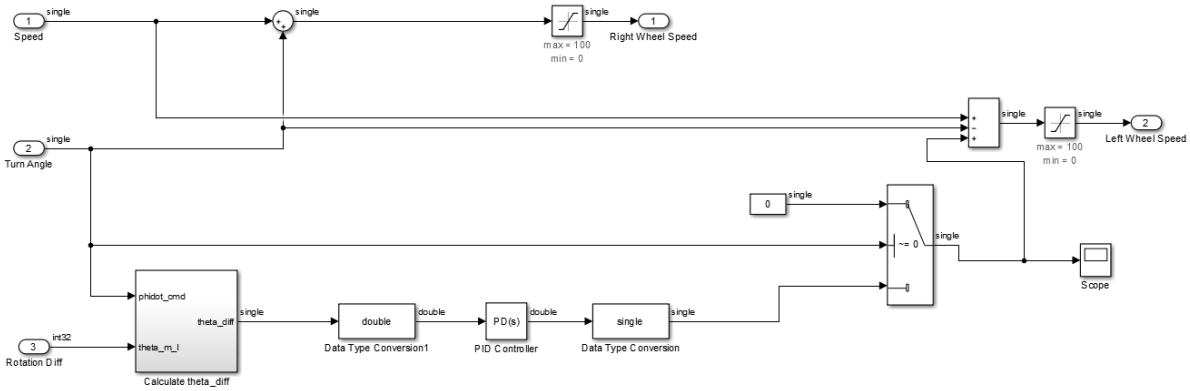
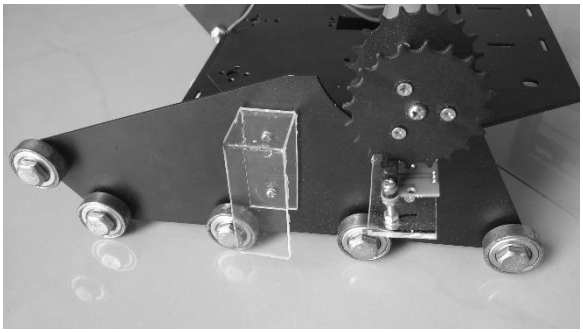


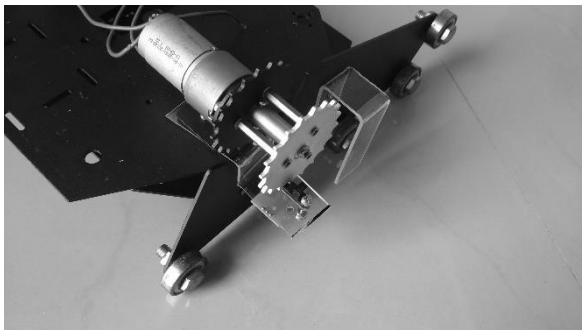
Figure 66 Expanded PID Model

Appendix B

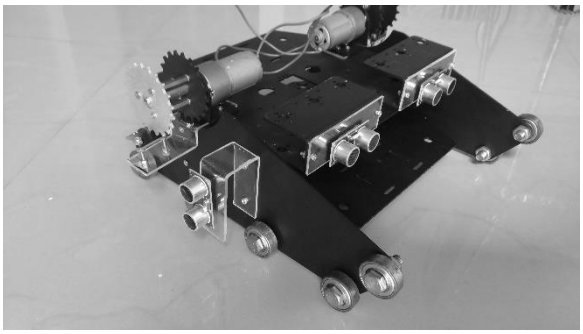
Mobile robot platform design

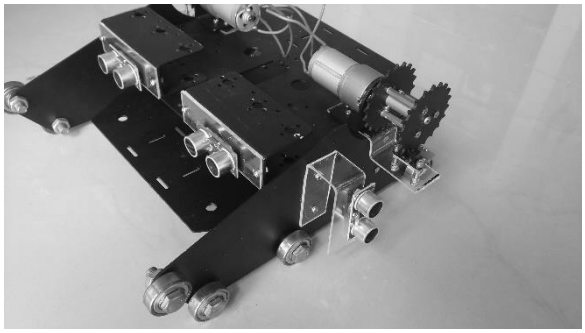
optical encoders





ultra-sonic sensors





power distribution circuit placement

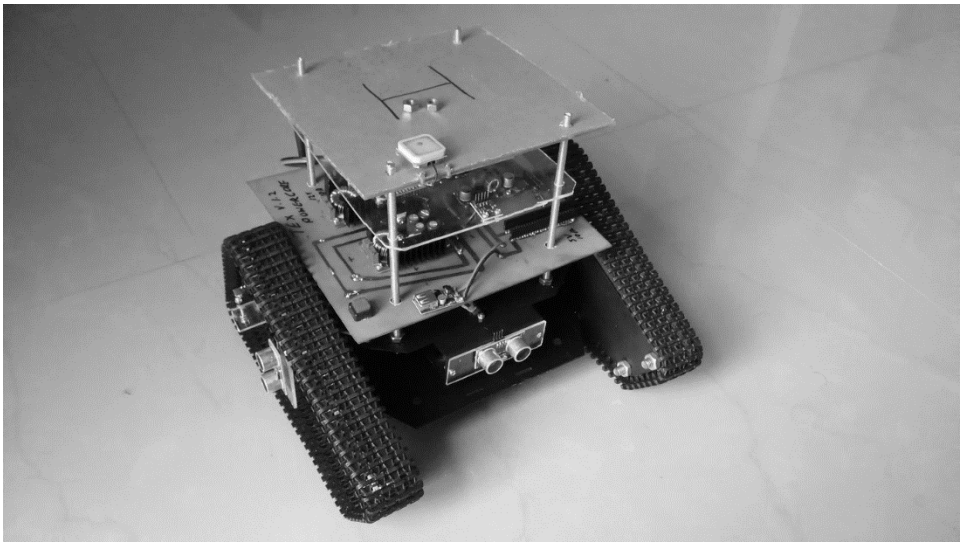
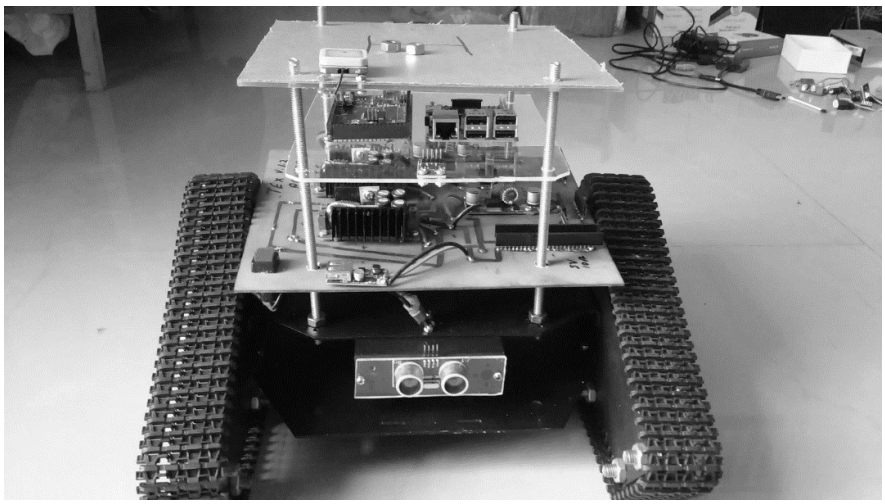
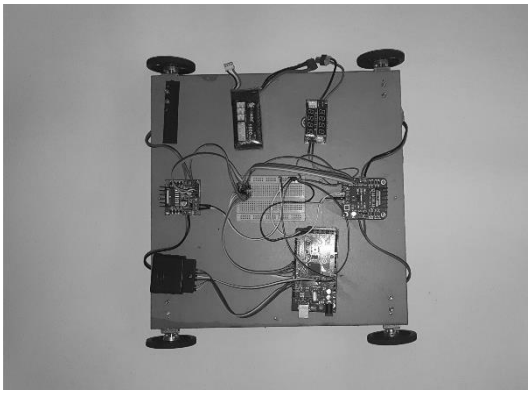


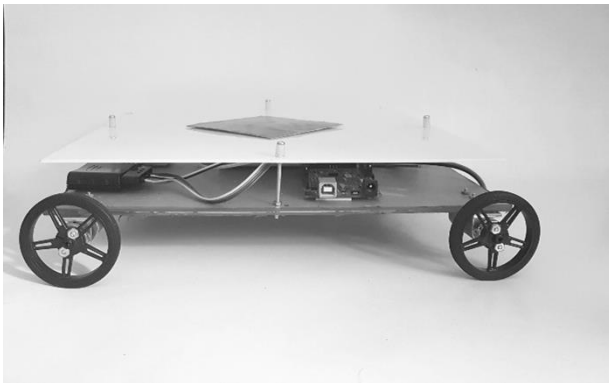
Figure 67 Mobile Robot Implementation

Raspberry pi 3, Arduino, MPU 6050, NRF24L01 and GPS module placement



Landing Platform design





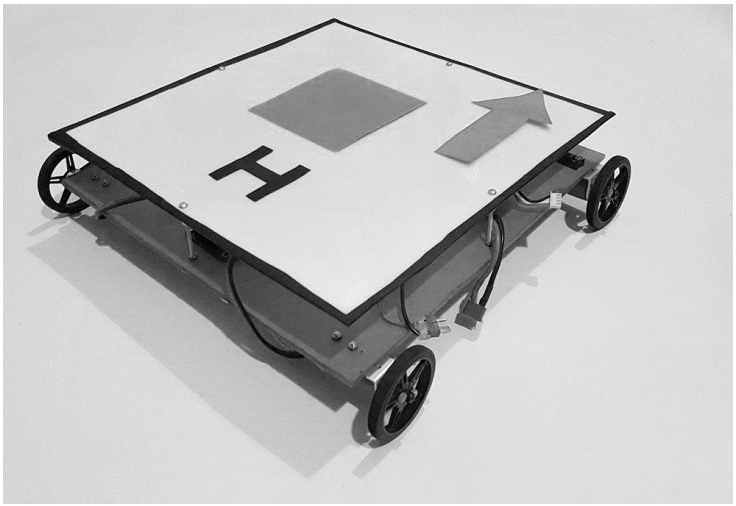


Figure 68 Experimental Landing Platform

Appendix C

Step 1: Define a resolution for the capture. The resolution was set to 704 x 576.

```
width = 704
height = 576

camera.set(cv2.cv.CV_CAP_PROP_FRAME_WIDTH, width)
camera.set(cv2.cv.CV_CAP_PROP_FRAME_HEIGHT, height)
```

Step 2: Create trackbars for online tuning.

function cv2.createTrackbar is used to create a slide bar feature for changing the HSV values.

```
hh='Hue High'
hl='Hue Low'
sh='Saturation High'
sl='Saturation Low'
vh='Value High'
vl='Value Low'

cv2.createTrackbar(hl, 'image',0,179,nothing)
cv2.createTrackbar(hh, 'image',0,179,nothing)
cv2.createTrackbar(sl, 'image',0,255,nothing)
cv2.createTrackbar(sh, 'image',0,255,nothing)
cv2.createTrackbar(vl, 'image',0,255,nothing)
cv2.createTrackbar(vh, 'image',0,255,nothing)
```

Step 3: define arrays for minimum and maximum range of HSV values.

```
rangeMin = np.array([hul, sal, val], np.uint8)
rangeMax = np.array([huh, sah, vah], np.uint8)
```

Step 4: Processing the image

image transformation to HSV -

```
imgHSV = cv2.cvtColor(frame, cv2.cv.CV_BGR2HSV)
```

Threshold the transformed image using minimum and maximum HSV values –

```
imgThresh = cv2.inRange(imgHSV, rangeMin, rangeMax)
```

Morphological Transformation: Erosion for better results.

```
imgErode = cv2.erode(imgThresh, None, iterations = 5)
```

Gaussian blur for eliminate the noise of the eroded image.

```
blurred = cv2.GaussianBlur(imgErode, (7, 7), 0)
```

Step 5: Canny Edge detection and Find contours of the blurred image.

Canny Edge detection algorithm is used to determine the number of corners in the image

```
edged = cv2.Canny(blurred, 50, 150)
```

finding contours of the detetcted edges.

```
(cnts,_)=cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
```

Step 6: check for contour length which is bound a square shape which has four edges in it.

```
if len(approx) >= 4 and len(approx) <= 6:
```

if condition is true, then determine the properties of the bounded object by contours. Following peace of code will determine the aspect ratio, area, hullArea, and solidity of the object.

```
 (x, y, w, h) = cv2.boundingRect(approx)
 aspectRatio = w / float(h)
 area = cv2.contourArea(c)
 hullArea = cv2.contourArea(cv2.convexHull(c))
 solidity = area / float(hullArea)
```

Step 7: Set the rules for detect only square shape term. Here square shape properties will be set.

```
 keepDims = w > 25 and h > 25
 keepSolidity = solidity > 0.9
 keepAspectRatio = aspectRatio >= 0.8 and aspectRatio <= 1.2
```

Step 8: Draw Contours

At this step the algorithm clarify the detected object is Red colored Square shape and graphically draw an outline of the window frame.

```
if keepDims and keepSolidity and keepAspectRatio and minArea > 50:


   cv2.drawContours(frame, [approx], -1, (0, 255, 0), 4)
```

Step 9: Find the Center of the Object

Method moments was used to determine the center of the tracked object. cX and cY gives the center coordinates of the tracked object.

```
M = cv2.moments(approx)

(cX, cY) = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
```

Step 10: Calculating the Area of the detected square shape.

```
area = round(M['m00'])
```

Step 11: Serial Event Generation

Calculated area and the center coordinates (cX, cY) of the detected square shape then sent as arguments to a function named quadMovement () through another function called Storage (). The serial event is generated inside the quadMovement () function. This function calls every 1s second in a separate Thread. Which is declared inside the Storage () function. For every second these data will be sent via RF link to quadcopter.

Define Serial COM port and the baud rate:

```
ser = serial.Serial('COM5', 115200)
```

Serial event exception handler:

```
try:
    ser.close()
    ser.open()

except Exception, e:

    print "error open serial port: " + str(e)
    exit()
```

Serial event generation function:

```
def quadMovement(directionX,directionY):

    value_1 = str(directionX)
    value_2 = str(directionY)

    ser.write("A")
    ser.write(value_1)
    ser.write(",")
    ser.write(value_2)
    ser.write(",")
    ser.write("1212")
    ser.write(",?#")
    print ser.readline()
```

Separate Thread Driven Function For fast event handling:

```
def Storage(quad_X,quad_Y):

    t1 = threading.Thread(target = quadMovement, args = (quad_X, quad_Y))
    t1.start()
    t1.join()
```

Step 12: Displaying the data on the screen

Draw a cross symbol on the center of the detected square shape using cv2.line function:

```
(cX, cY) = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
(startX, endX) = (int(cX - (w * 0.15)), int(cX + (w * 0.15)))
(startY, endY) = (int(cY - (h * 0.15)), int(cY + (h * 0.15)))

cv2.line(frame, (startX, cY), (endX, cY), (255, 0, 0), 2)
cv2.line(frame, (cX, startY), (cX, endY), (255, 0, 0), 2)

Displays a text when there is no target marker is detected:

status = "Searching the Object"
cv2.putText(frame,status, (20, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(0, 0, 255), 2)

Displays when tracking the object:

status_2 = "Following the object"
cv2.putText(frame, status_2, (20, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(255, 255,
51), 2)

Displays Center coordinates on screen:

cv2.putText(frame,"X: " + str(cX), (600, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(0,
255, 0), 2)
cv2.putText(frame,"Y: "+ str(cY), (700, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(0, 255,
0), 2)


Displays the area of the detected object:

cv2.putText(frame,str(area), (20, 100), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(210, 0,
255), 2)
```

Step 13: Distance Calculation

A separate method is calling when distance in needed. This will ultimately returns the Actual length to where it called.

```
def Distance(incom):

    F = 675.498
    W = 10
    A = int(incom)
    P = math.sqrt(A)

    D’ = int((W*F)/P)

    return D’
```

Step 14: check the bounded contours within the range of less than 8 and greater than 5.

```
if len(approx) >= 5 and len(approx) < 8:
```

Step 13: calculate properties of the arrow and draw contours.

```
(x, y, w, h) = cv2.boundingRect(approx)
```

```
aspectRatio = w / float(h)
area = cv2.contourArea(c)
hullArea = cv2.contourArea(cv2.convexHull(c))
solidity = area / float(hullArea)

cv2.drawContours(frame, [approx], -1, (0, 255, 0), 4)
```

Step 14: Determine angle

Following function calculates the ellipse that fits (in a least-squares sense) a set of 2D points best of all. It returns the rotated angle in which the ellipse is inscribed.

```
(p,q),(MA,ma),angle =  cv2.fitEllipse(c)
```