4.1.3 Programs

a) Python programs to control the quadcopter:

1. PID Controller
2. Kalman Filter
3. Quadcopter Control
4. Python Server

b) Python programs for Panorama Stitching:

1. Camera Calibration
2. Panorama Stitching

c) Python Programs for 3D Reconstruction:

1. Extract Metadata
2. Detect Features
3. Match Features
4. Create Tracks
5. Reconstruct
6. Mesh

d) HTML Code for Visualizing Sparse Reconstruction

e) Android App providing virtual joystick interface

4.2 TECHNOLOGY USED

4.2.1 Raspberry Pi 2 Microcomputer

The quadcopter is controlled using the Raspberry Pi 2. The Raspberry Pi (Figure 4.1) is a credit card sized microcomputer that runs on a Linux based operating system loaded onto its SD card .

It includes a 900MHz quad-core ARM Cortex-A7 CPU and 1GB RAM.

It can connect to a network by using an external user-supplied USB Ethernet or WiFi adapter. Furthermore, the RPi has four USB ports, a full HDMI port, 40 GPIO pins and an Ethernet port. The GPIO header depicting the pin configuration of the Raspberry Pi 2 has been shown in Figure 4.2. Figure 4.3 depicts the RPi's GPIO pins. Apart from this, the Raspberry Pi has a camera interface which allows a native Pi Camera to be configured and used. It also has a Micro SD Card slot. The advantage of using a RPi over traditional Arduino boards is that the Pi has much greater computational power and thus, more complicated and better stabilization algorithms can be run on it.

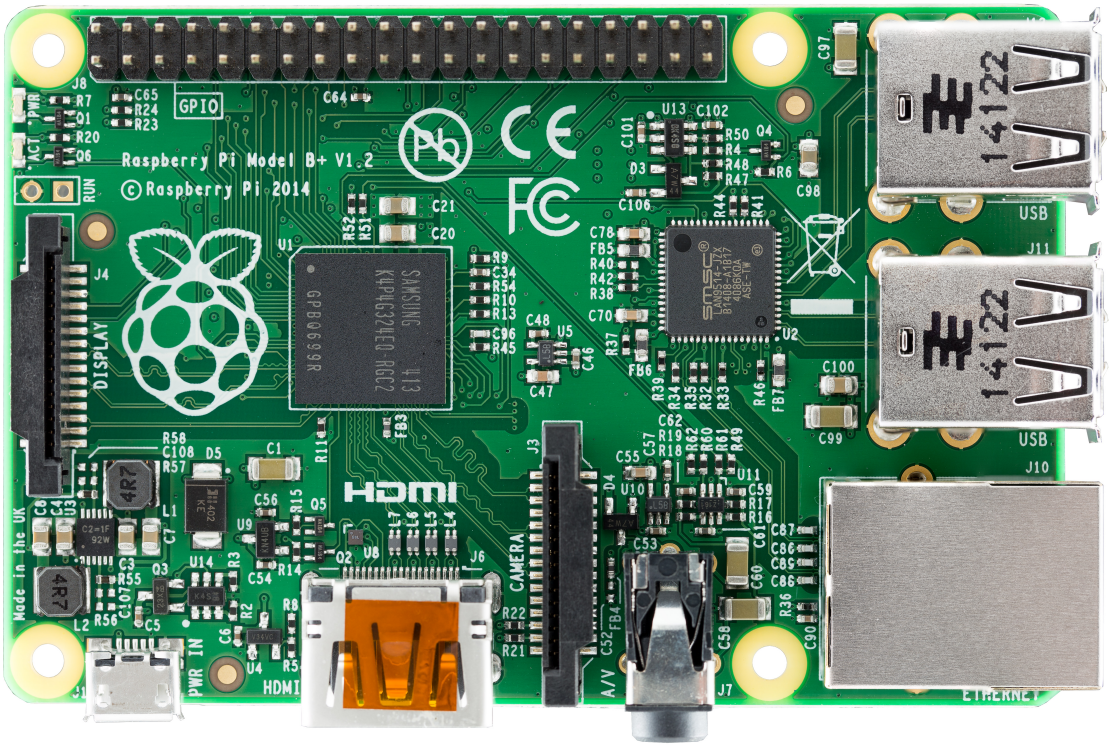


Figure 4.1: Raspberry Pi

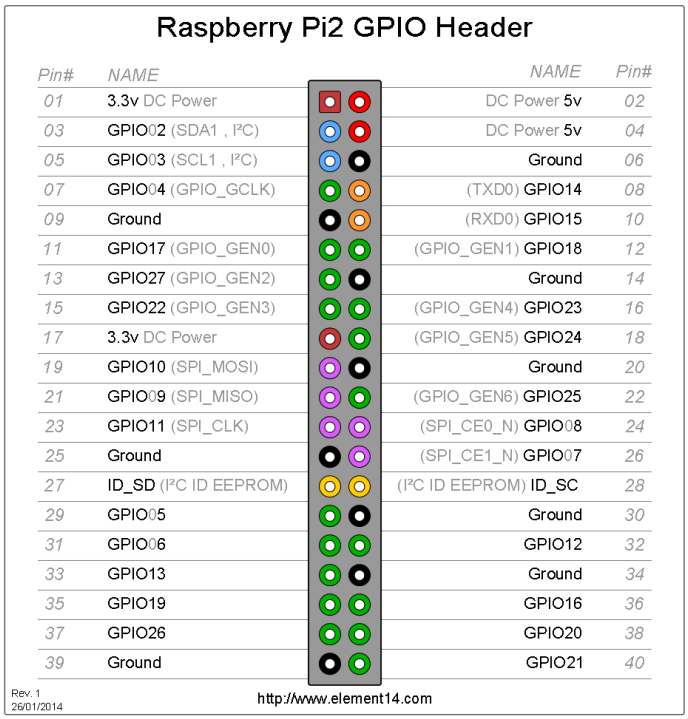


Figure 4.2: Raspberry Pi 2 GPIO Header

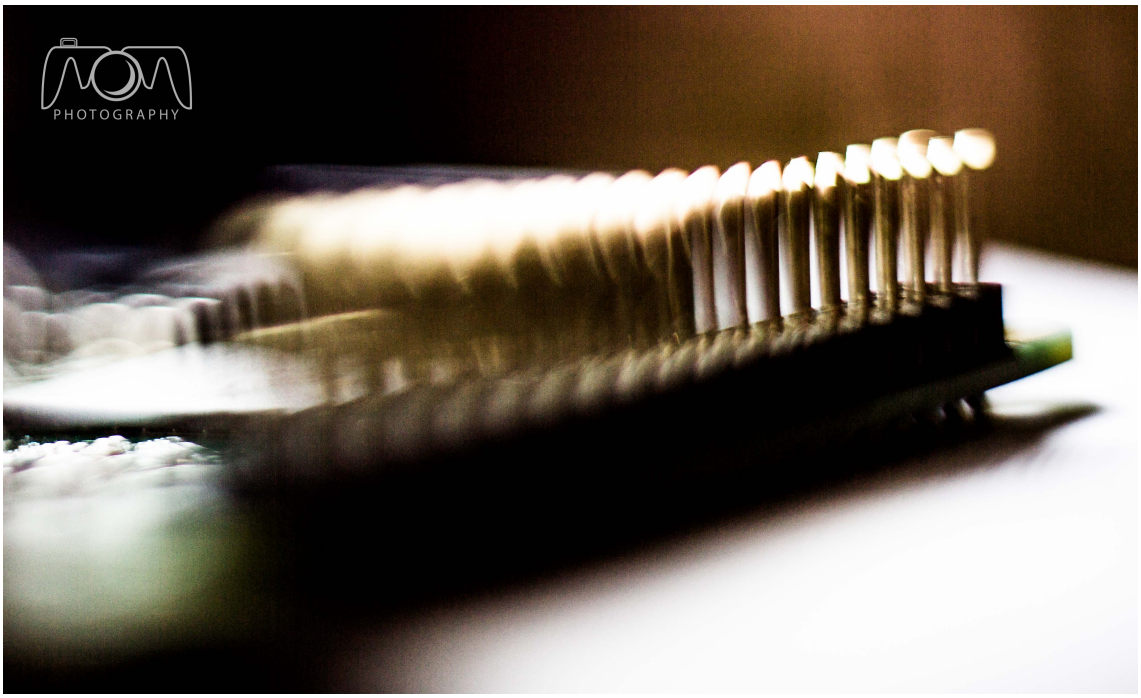


Figure 4.3: Raspberry Pi 2 GPIO Pins

4.2.2 Electronic Speed Controller (ESC)

The Electronic Speed Controller (Figure 4.4) is a circuit which varies the motor's speed, direction and acts like a dynamic brake. ESCs are most often used with BLDC motors necessary for translating digital signals to electric current. ESCs are an essential component of modern multirotor crafts that offer high power, high frequency, high resolution 3-phase AC power to the motors in an extremely compact miniature package. These crafts depend entirely on the variable speed of the motors driving the propellers.

The large variation and fine RPM control in motor/propeller speed gives all of the control necessary for a quadcopter (and all multirotors) to fly. On a quadcopter, each motor gets its own ESC, each of which connects to the controller. After computing the inputs(the amount of current and direction to be given to each motor), the controller directs each ESC to adjust its speed according to the PWM signal provided. This quadcopter uses a 40A, 3s ESC.

Figure 4.4: 40A 3s ESC

4.2.3 IMU Sensor

An inertial measurement unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the magnetic field surrounding the body, using a combination of accelerometers and gyroscopes and magnetometers. IMUs are typically used to maneuver aircraft, including UAVs. This project uses a 9DOF IMU sensor (Figure 4.5) with a 3-axis accelerometer, 3-axis gyroscope and 3-axis magnetometer. The accelerometer measures acceleration and has high accuracy. The gyroscope measures the angular velocity and is used to stabilize flight. It is less accurate than the accelerometer and is prone to drift errors.

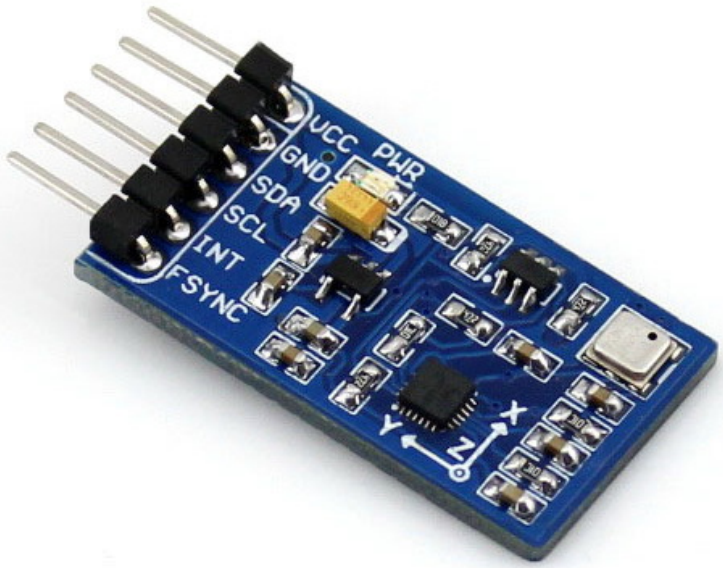


Figure 4.5: IMU Sensor

4.2.4 Raspberry Pi Camera

The Raspberry Pi camera module (Figure 4.6) can be used to take high-definition video, as well as photographs. The camera has a five megapixel fixed-focus camera that supports 1080p30, 720p60 and VGA90 video modes, as well as stills capture. It can be controlled programmatically. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.

It supports a frame rate upto 120fps. Picture formats supported by the RPi camera are JPEG, GIF, BMP, PNG, YUV240 and RGB8888. It also allows multiple filters and effects such as negative, solarise, posterize, whiteboard, blackboard, sketch, denoise, emboss, oilpaint, hatch, gpen, pastel, watercolour, film, blur and saturation for capturing images. It can be configured using the Python Picamera library. The project uses the Pi Camera to capture images for panorama stitching and 3D reconstruction.

Figure 4.6: Raspberry Pi camera

4.3 BLOCK DIAGRAM

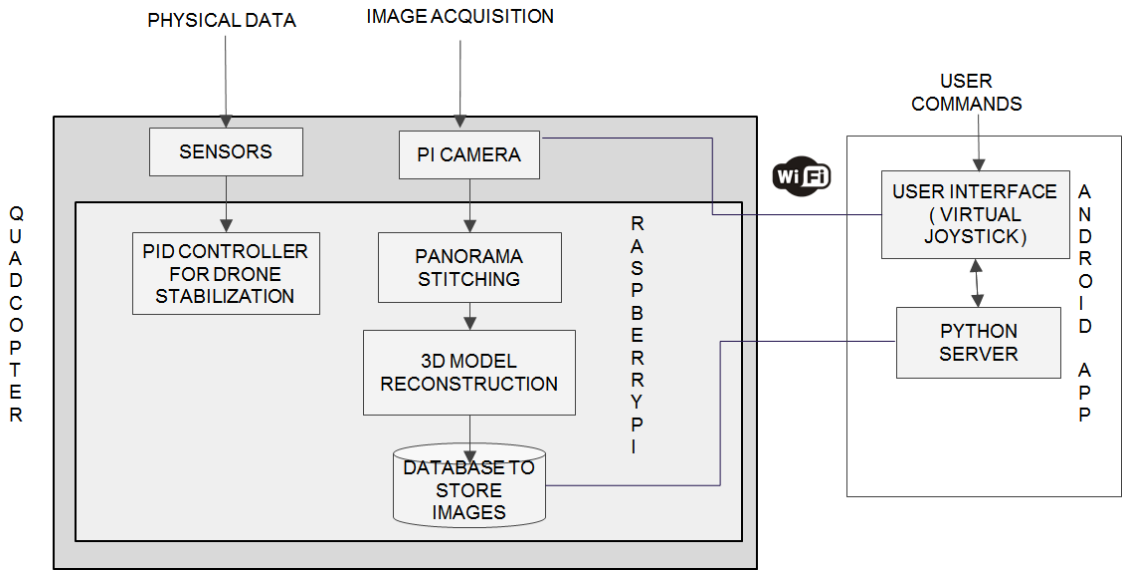


Figure 4.7: Block Diagram

4.4 CIRCUIT DIAGRAM

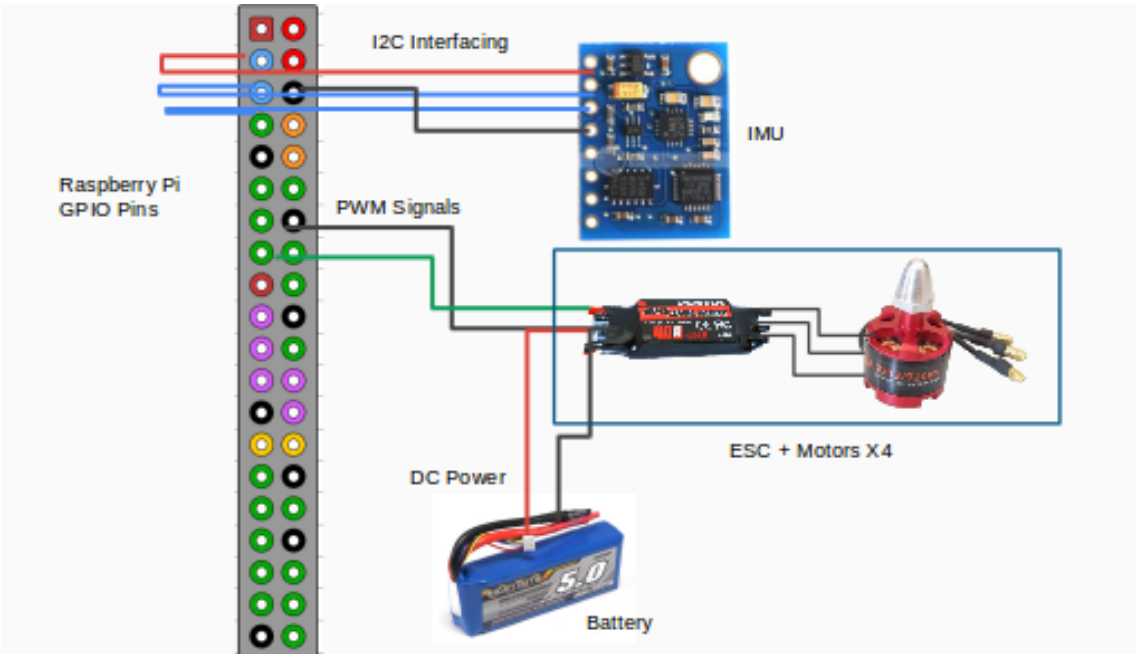


Figure 4.8: Circuit Diagram

4.5 FLOW DIAGRAM

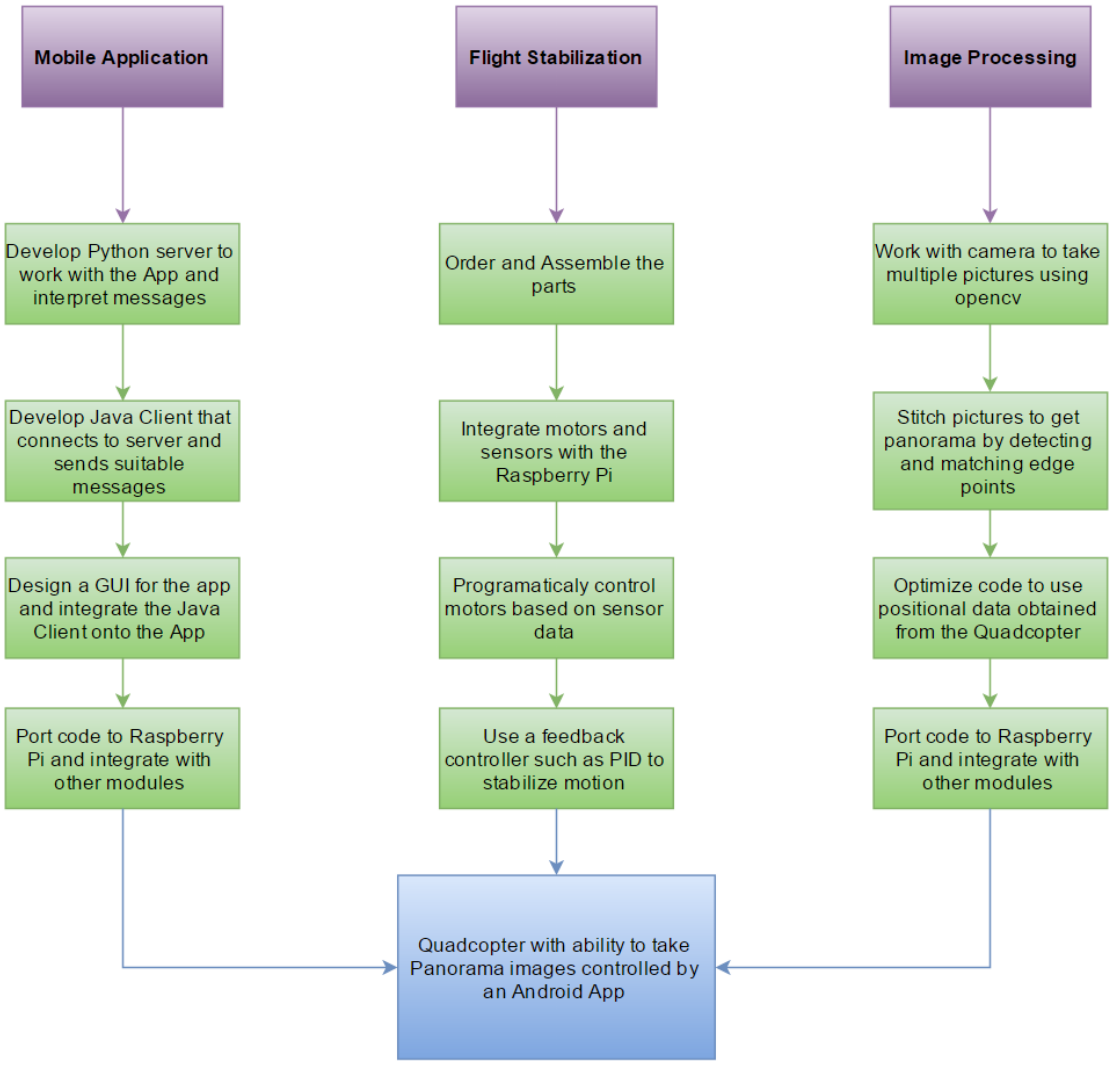


Figure 4.9: Flow Diagram from Input to Output

Description of Flow Diagram

The user begins the quadcopter program by clicking a button on the Android App which is sent as input to the Python server on the Raspberry Pi. These signals are sent over a WiFi network using an Android phone as a mobile hotspot. The user can control the altitude of the drone and the individual motors using the virtual Joystick on the app. Signals are sent as JSON-encoded data to the quadcopter. The orientation data i.e. the roll, pitch and yaw values are detected by the

on current rate of change.

The Proportional Gain Coefficient(P) is the most important value. This coefficient determines which value is more important - those measured by the gyroscopes or those defined by human control. The higher the coefficient, the more sensitive the quadcopter becomes to angular change. If 'P' is too high, the quadcopter will start to oscillate.

The Integral Gain Coefficient(I) is used to increase precision of angular position. This term is extremely useful for combating turbulence due to wind and motors. If this value becomes too high, the reaction speed of the drone will decrease.

The Derivative Gain Coefficient(D) is used to allow the quadcopter to reach the desired altitude more quickly as it amplifies the user input. It is dangerous to increase this value as the quadcopter may become highly susceptible to small changes.

The quadcopter flight and on-board operations are controlled via WiFi using an Android app. It receives signals encoded as JSON objects as steady stream from the App. The signal carries altitude, trims, motor speeds and camera information to the RPi. The signal is received by a Python server running on the RPi which is decoded and fed as input to the Quadcopter program.

The quadcopter also houses a native Pi camera module which has the ability to take high quality photos and videos. It interfaces with the Raspberry Pi using a Camera Serial Interface.

The function of the quadcopter is to take pictures and perform two major image-processing operations viz. Panorama Stitching and 3D-Model Reconstruction. The quadcopter itself is small enough to control its flight and large enough to hold all the on-board equipment.

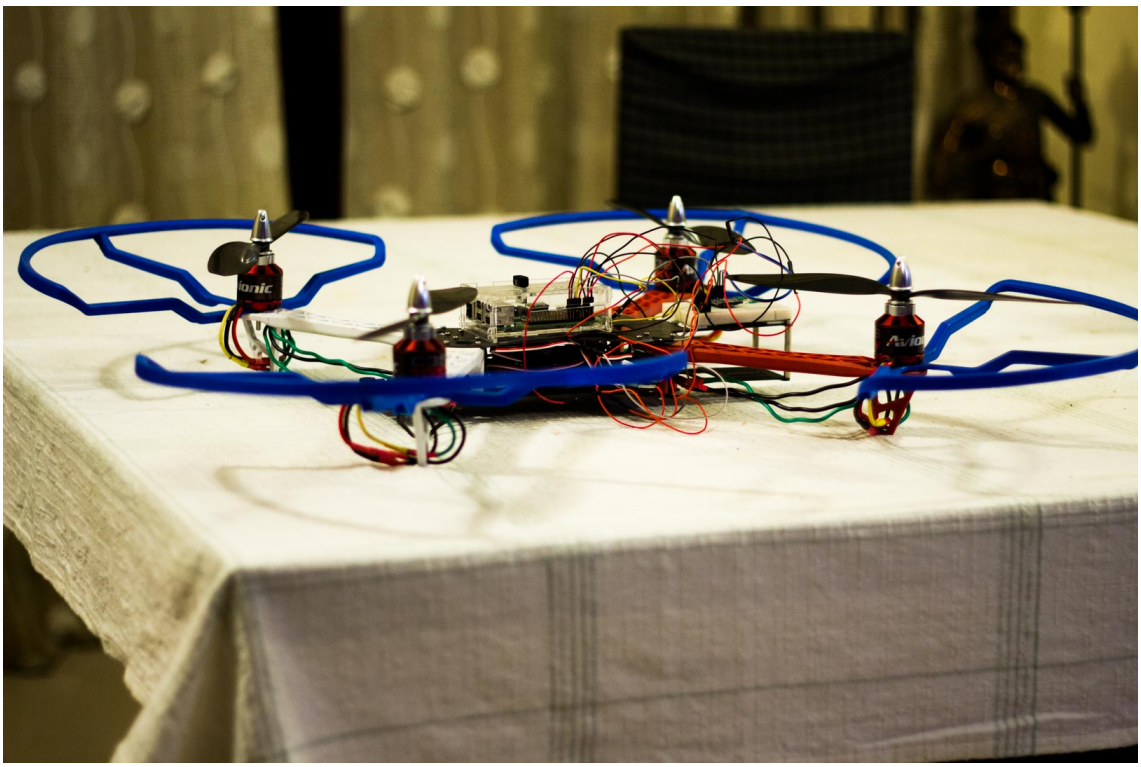


Figure 5.1: The Preliminary Quadcopter without the camera. It carries the ESC, IMU sensor and the RPi. It also has 4 propeller guards (blue) to protect the propellers from excessive damage.

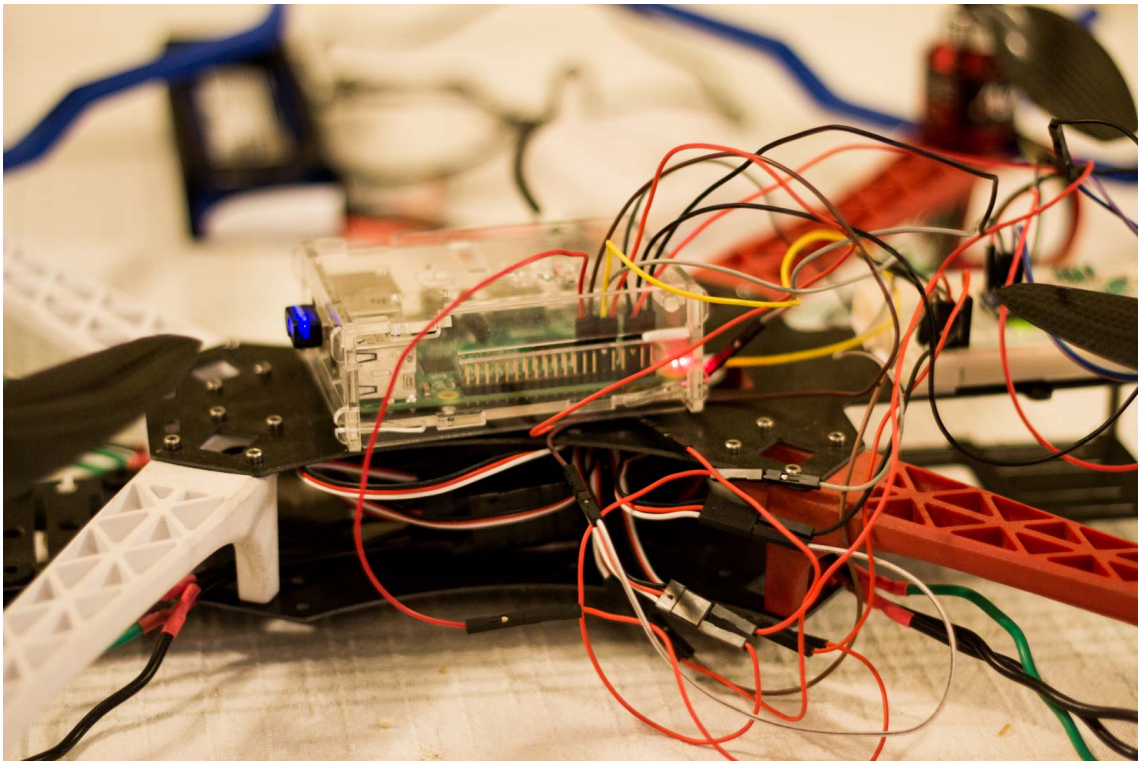


Figure 5.2: Closer look at the wiring of the quadcopter

5.3.2 Image-Processing Module

The objective of the project is to build a quadcopter that can perform Panorama Stitching and 3D Reconstruction of images taken during its flight. The reason we chose these functions is because the drone is mobile and can capture pictures of inaccessible locations like tall buildings and can perform mapping of wide-open spaces.
OpenCV is an open-source image processing library which has an enormous number of functionally-rich libraries and detailed online documentation. OpenCV Version 3.1.0 (the latest version) is used to provide image-processing support in this project.

The image-processing module is divided into two parts Stitching and Reconstruction. The initial stages are the same for both the operations i.e. Acquisition of images, Detecting Features and Matching

Features. Feature detection is accomplished using SURF (Speeded Up Robust Features algorithm and Feature Matching is done by building Kd-Trees and using (K-Nearest Neighbours) Flann Algorithm. Feature detection yields keypoints and descriptors used for matching images.

In panorama stitching, RANSAC (Random Sample Consensus) is used to detect outliers and construct a homography matrix. This is essential as it relates pixel coordinates in each pair of images. It is then used to warp the images and construct the panorama by aligning them. A multi-threaded stitching algorithm has been implemented that stitches images in parallel, reducing the time while maintaining the quality of the constructed image (Figure 5.3). The quality of the panorama is improved if images are taken with large overlapping features.

3D-Reconstruction is implemented by using the Structure from Motion(SfM) algorithm. The preliminary stage is to calibrate the RPi Camera and calculate intrinsic and extrinsic camera paramaters. Once camera calibration is complete, the images have to be preprocessed to collect metadata i.e. EXIF tags which hold important GPS, latitude, longitude and camera information.

The next stages are the feature detection and matching stages which have been described above. Matching features are then organized into tracks and a bipartite graph of these 'good tracks' is constructed. This is used to find common tracks and construct the 3D point cloud incrementally. Reconstruction begins with 'bootstrap reconstruction' using two images. The reconstruction consists of triangulating the 2D points to 3D points which is visualized as a point cloud. Consequently, the rest of the images' points are added incrementally to the point cloud.

After each addition, Bundle Adjustment is done to minimize the reprojection error. Once all the images have been added to the reconstruction, the final point cloud is generated and stored as a '.ply' file. The point cloud is visualized (Figure 5.4) in an interactive browser environment using a Javascript Library called, 'threeJS'.

Figure 5.3: Incremental panorama stitching of test images

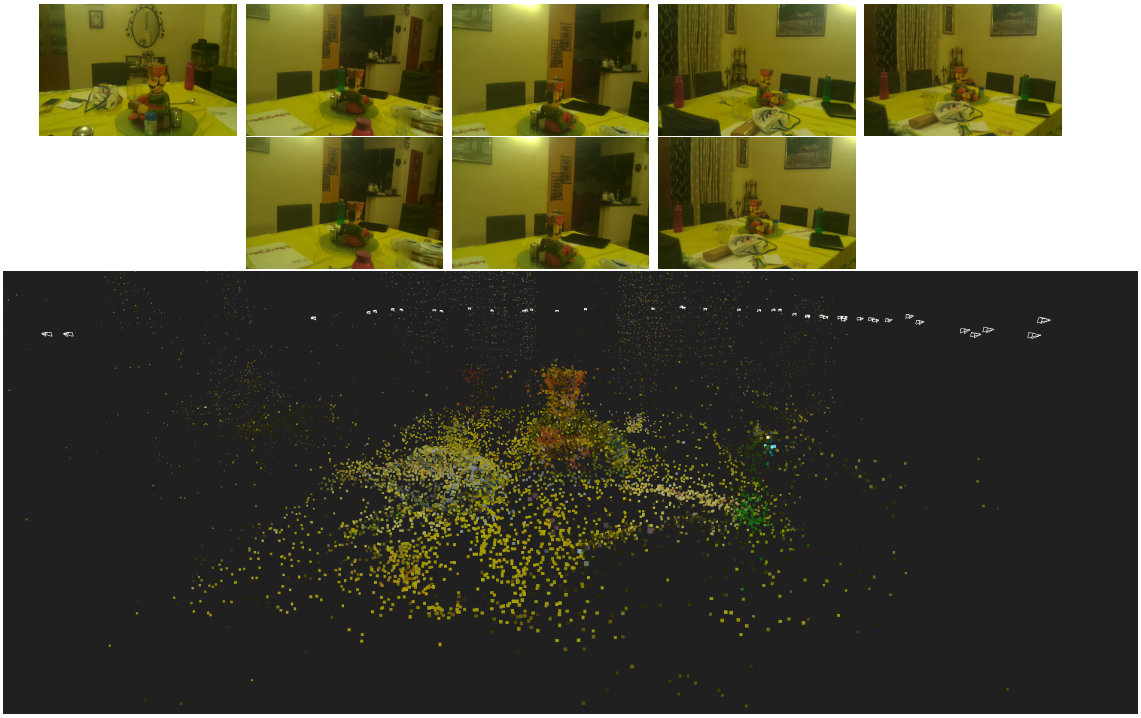


Figure 5.4: 3D Reconstruction of test images where colored point cloud is rendered on the browser which can be inspected closer.

5.3.3 Android App

The Android App is used to control all the operations on the quadcopter. It implements a virtual Joystick interface (Figure 5.5) for easy user-interaction. The interface consists of two joysticks, buttons to start the quadcopter program, enable camera mode, dynamically alter x and y trim values and a text box for feeding in the IP address of the RPi.

The two joysticks - left and right are for for altitude control and motor control respectively. The user can swipe upwards and downwards on the left joystick to increase and decrease the height of the quadcopter. Similarly, the user can swipe in all directions on the right joystick to control the speed of individual motors depending on the distance from the center of the joystick.

The x and y trim values can be dynamically changed using the app. These values are used to add bias to the quadcopter and control its stability. The camera button can enable or disable camera mode on the quadcopter. Enabling it allows the RPi camera to begin shooting pictures with an interval of 1 second while disabling it stops the camera program.

Using an Android app to remotely control UAVs is a level up from traditional RC controllers as it allows users greater flexibility. Users can now control many more aspects of the quadcopter apart from height and orientation such as camera control and bias using a simple and attractive user interface.

The next chapter deals with the Results and Evaluation of test cases.

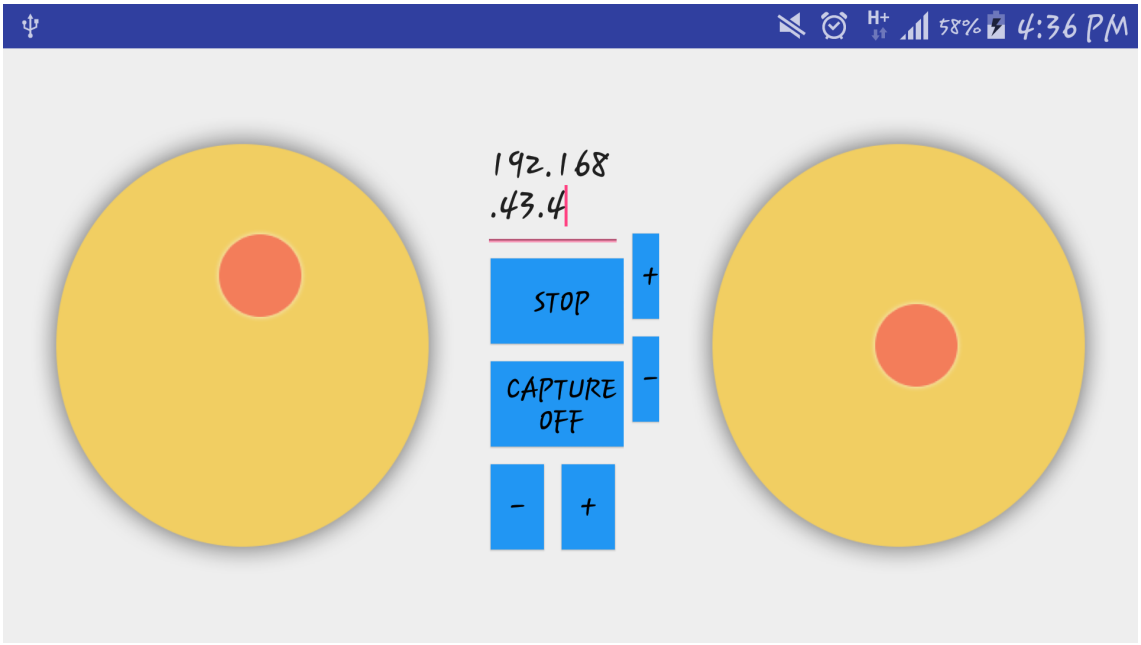


Figure 5.5: Android App Joystick Interface

5.4 OVERALL COMPONENT DIAGRAM

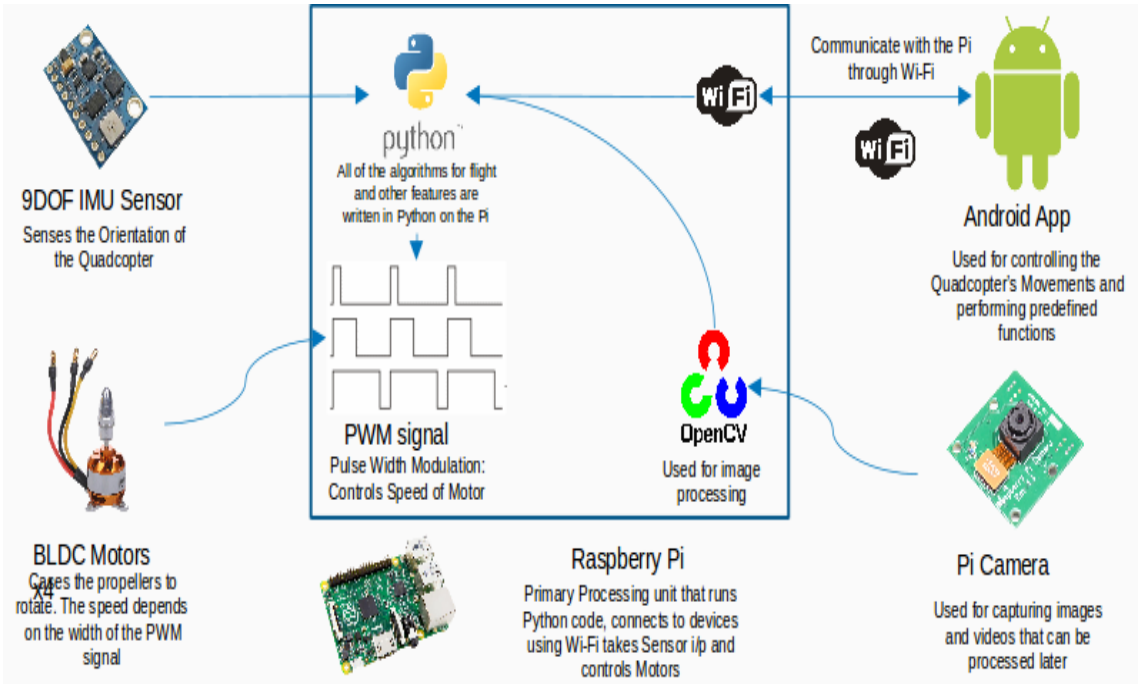


Figure 5.6: Overall Hardware and Software Component Diagram describing the flow of execution

CHAPTER 6

RESULTS AND DISCUSSION

6.1 RESULTS

6.1.1 Assessment

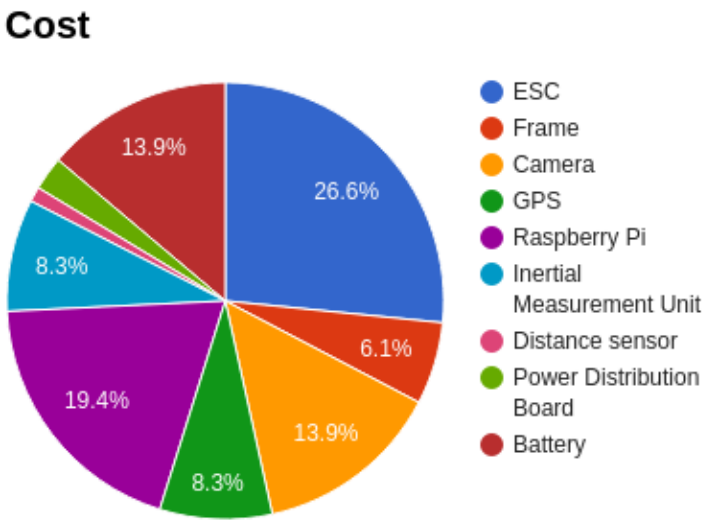


Figure 6.1: Project Cost for each Part

S.No	Signal	Result
1	Swipe left joystick up	Quadcopter flies upward
2	Swipe left joystick down	Quadcopter's height decreases
3	Swipe right joystick up	Quadcopter pitches forward
4	Swipe right joystick down	Quadcopter pitches backward
5	Swipe right joystick left	Quadcopter pitches left
6	Swipe right joystick right	Quadcopter pitches right
7	Click the 'start' camera button	Pi Camera starts shooting
8	Click the 'stop' camera button	Pi Camera stops shooting
9	Click the '+/-' button on the x-axis	Quadcopter's x-trim values change
10	Click the '+/-' button on the y-axis	Quadcopter's y-trim values change

Table 6.1: Test Cases and Results

6.1.2 Evaluation

There are several changes that have to be made when flying the quadcopter in varying environments. The PID values change according to the wind conditions and other disturbances. Although the SD Card is securely inserted into the RPi, due to a large amount of vibration caused in the course of the quadcopter's flight, it may loosen leading to connection loss between the RPi and the App.

The quadcopter is controlled by the App from a distance of about 10-15 metres since communication occurs via WiFi. Images are taken by a RPi native camera which is fixed securely on top. The quality of the images taken largely depend on the stability of the quadcopter which are subject to the vagaries of the environment. The pictures taken are stitched and/or reconstructed. The speed of the image-processing depends on the number of images taken and the quality of the images.

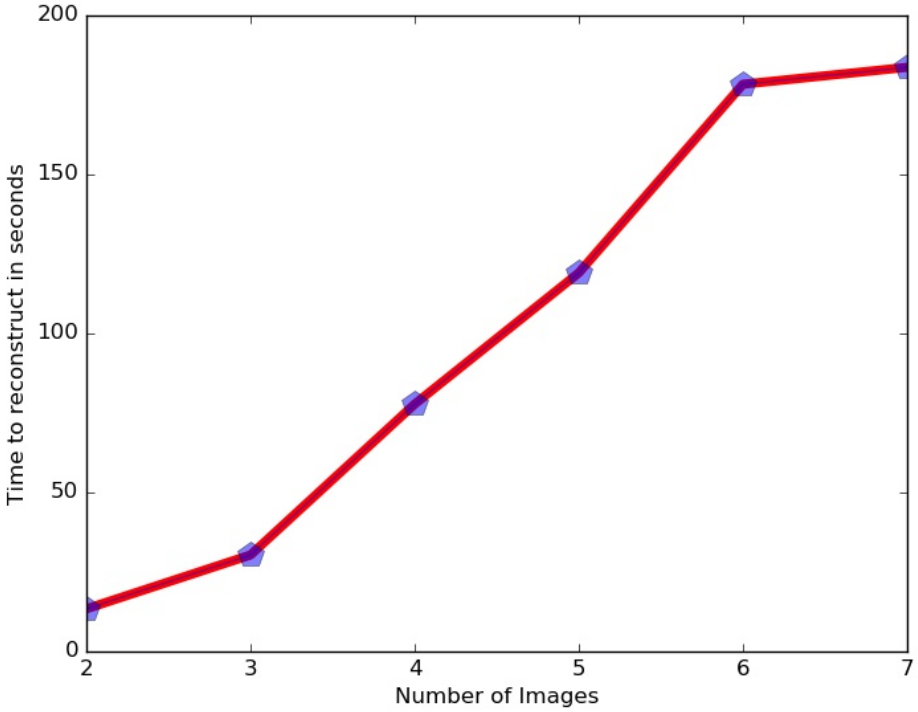


Figure 6.2: Time taken for computation of panorama stitching is directly proportional to the number of images. It stabilizes slightly when the image count is greater than seven.

Figures 6.3, 6.4, 6.5a and 6.5b depict the data obtained during a test run of the quadcopter, where the sensor/output(y-axis) data is plotted against time in seconds(x-axis). The first 10 seconds show the data where the motors run at minimal speed followed by intermittent high speeds to allow lift-off till the 20 second mark. At about 25 seconds the quadcopter had attained a height of roughly 1 feet from the ground and was hovering with mild oscillations. At about 45 seconds the quadcopter encountered a breeze causing it to temporarily tilt in the direction of the breeze and oscillate with increasing amplitude till 55 seconds, at which point it was powered down.

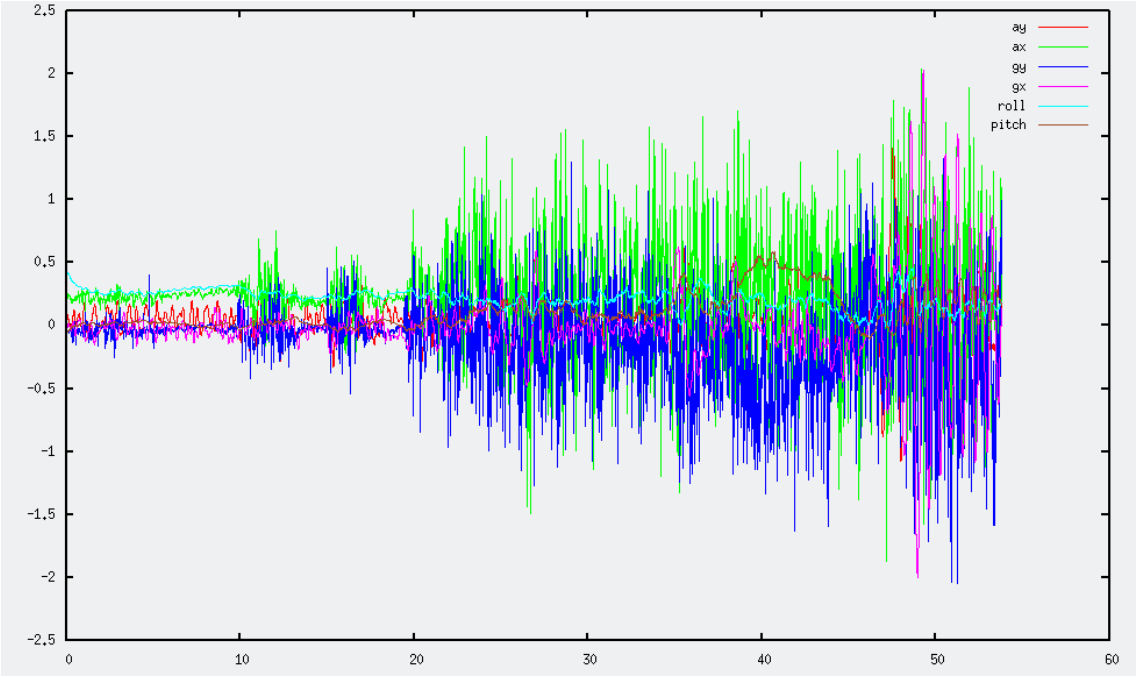


Figure 6.3: Measurement of pitch and roll using complementary filter

The orientation of the quadcopter is sensed using the accelerometer and the gyroscope present in the IMU sensor (Fig 6.3). An accelerometer measures all forces that are working on the object, it will also see a lot more than just the gravity vector. Every small force working on the object will disturb our measurement completely and result in noise[Fig 6.3 ax and ay]. On an actuated system (like the quadrocopter), the forces that drive the system are visible on the sensor as well. The accelerometer data is reliable only on the long term, so a ”low pass” filter has to be used.

A Gyroscope accurately measures the change in angle (Fig 6.3 gx and gy). Accurate measurement of the current angle can be obtained by integrating the change in angle over time to obtain the overall change in angle. However, the value has the tendency to drift, not returning to zero when the system goes back to its original position.

The complementary filter combines the two to get a reliable sense of orientation (Fig 6.3 pitch and roll). On the short term, data from the gyroscope is used and on the long term, data from the accelerometer is used. The filter looks as follows (Equation 6.1):

$$angle = 0.98 * (angle + gyrData * dt) + 0.02 * accData \tag{6.1}$$

The gyroscope data is integrated every timestep with the current angle value. After this it is combined with the low-pass data from the accelerometer (already processed with atan2). The constants (0.98 and 0.02) have to add up to 1 but can be changed to tune the filter properly.

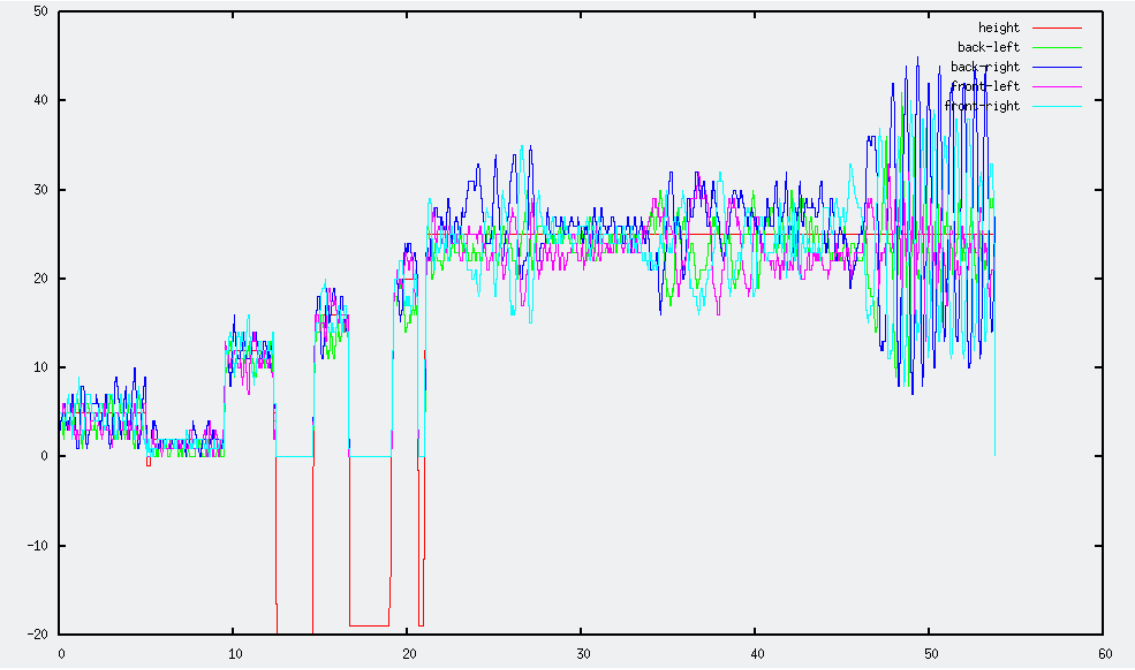
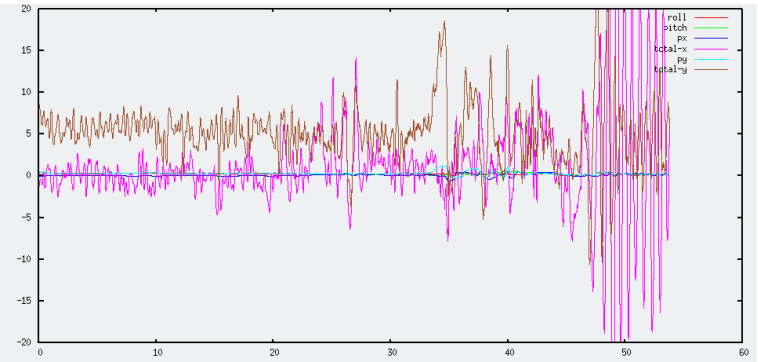
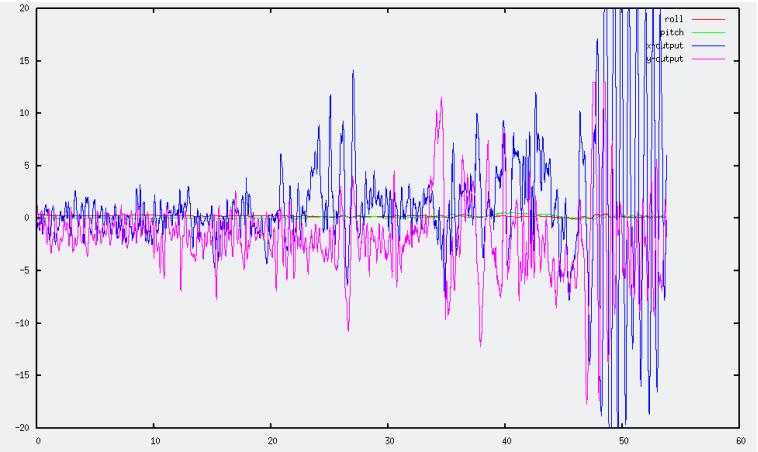


Figure 6.4: Output of motors with respect to height

The motor outputs require to be offset by the required height to allow the quadcopter to increase or decrease the overall height while balancing using the offset received from the PID controller[Fig 6.4].



(a) Derivation of output using PID controller



(b) Final output with respect to pitch and roll

Figure 6.5: Derivation and final output from sensed angles

Figure 6.5a shows how the final output and Figure 6.5b shows how sensed pitch and roll are converted to the final output by the PID controller to be used as offset to the motor speed for balancing the quadcopter. The total-x and total-y are computed using the following PID formula (Equation 6.2):

$$motor_{command} = K_p^r * ((K_p^a(\theta_{input} - \theta) + K_i^a \int (\theta_{input} - \theta)dt) - \frac{d\theta}{dt}) \tag{6.2}$$

The last chapter discusses the project's contributions and future work.

CHAPTER 7

CONCLUSION

7.1 CONTRIBUTIONS AND FUTURE WORK

This project establishes a method to integrate the latest advances in aerial robotics and image processing that can ultimately be used for a multitude of applications in domains including aerial photogrammetry, terrain exploration and mapping, archaeological surveys, movie-making and security. The quadcopter combines the two most-often used image-processing functionalities - Panorama and 3D Reconstruction. The extreme mobility of the drone and powerful computing power of the Raspberry Pi form an effective combination for this purpose.

The ability to take panaroma pictures allows the quadcopter to reach remote destinations inaccessible by humans and perform a complete panaroma stitching to enable survey of such locations and 3D object reconstruction allows recreation of navigable 3D scenes from locations where the pictures are taken.

The quadcopter can be made completely autonomous by adding a GPS sensor for navigation and distance sensors or multiple cameras implementing obstacle detection to avoid obstacles. This will enable the user to input the location to the quadcopter where it flies and takes pictures that it will use for 3D reconstruction and panorama stitching.