

Automated indoor Surveillance Quadcopter with image  
recognition using OpenCV via Canny edge detection  
for avoiding stationary obstacles

by

**Luigi M. Moran**  
**Gary Albert B. Borja**  
**Yancy Howell A. Casupanan**

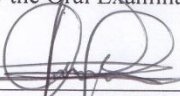
A Thesis Report Submitted to the School of EE-ECE-COE  
in Partial Fulfilment of the Requirements for the Degree

Bachelor of Science in Computer Engineering

Mapua Institute of Technology  
2014

## APPROVAL SHEET

This is to certify that I have supervised the preparation of and read the thesis report prepared by **Gary Albert B. Borja, Luigi M. Moran, and Yancy Howell A. Casupanan** entitled **“Automated indoor Surveillance Quadcopter with image recognition using OpenCV via Canny edge detection for avoiding stationary obstacles”** and that the said report has been submitted for final examination by the Oral Examination Committee.

  
\_\_\_\_\_  
**CARLOS HORTINELA IV**  
Thesis Adviser

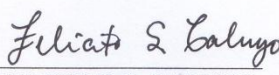
As members of the Oral Examination Committee, we certify that we have examined this thesis report and hereby recommended that it be accepted in fulfillment of the thesis requirements for the degree in **Bachelor of Science in Computer Engineering**.

  
\_\_\_\_\_  
**FEBUS REIDJ CRUZ**  
Panel Member

  
\_\_\_\_\_  
**JANETTE FAUSTO**  
Panel Member

  
\_\_\_\_\_  
**ISAGANI VILLAMOR**  
Panel Member

This thesis paper is hereby approved and accepted by the School of Electrical, Electronics, and Computer Engineering in partial fulfillment of the requirements for the degree in **Bachelor of Science in Computer Engineering**.

  
\_\_\_\_\_  
**FELICITO S. CALUYO**  
Dean, School of EECE

## ACKNOWLEDGEMENT

Firstly, we would like to express our deepest gratitude to our Adviser, Engr. Carlos Hortinela IV for giving us vital information and guidance in making our thesis paper possible.

Secondly, to Engr. Dionis Padilla for teaching us how to use the OpenCV in visual studio. In giving us possibilities to achieve our objectives.

We would also like to thank our Subject Chair, Engr. Noel Linsangan, facilitators, Engr. Jumelyn Torres and Engr. Voltaire De Leon, together with our panelists for the tutelage, suggestions, and recommendation to our thesis paper. Our Power Lab assistant Roger for providing material for our prototype.

Lastly, we would like to thank our family, friends, and the Lord Almighty for their support financially and emotionally, understanding and love in making our thesis paper possible.

## TABLE OF CONTENTS

TITLE PAGE	i
APPROVAL SHEET	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS	iv
LIST OF TABLES	vi
LIST OF FIGURES	vii
ABSTRACT	viii
Chapter 1: INTRODUCTION	1
Chapter 2: REVIEW OF RELATED LITERATURE	4
Unmanned Aerial Vehicle	4
Microcontroller	4
Flight Controller	6
Aerodynamics of a Quadcopter	6
Image processing	9
Canny Edge Detection	10
Hough Line Transform	11
Digital and Wireless Camera	12
Ultrasonic Sensor	12
Chapter 3: METHODOLOGY	14
Abstract	14
Introduction	14

Methodology	18
Data Flow Diagram	18
Quadcopter	20
Canny Edge Detection	21
Collision Detection Circuit	22
Image Detection Process	23
Result and Discussion	24
Chapter 4: CONCLUSION	29
Chapter 5: RECOMMENDATION	30
REFERENCES	31
APPENDICES	33

## LIST OF TABLES

Table 3.1: Quadcopter Movement	21
Table 3.2 Proximity sensor testing	23
Table 3.3 Measurement of altitude hold of quadcopter	24
Table 3.4 Quadcopter Stationary Obstacle Avoidance Result (One Edge detected)	25
Table 3.5 Quadcopter Stationary Obstacle Avoidance Result (One Edge detected)	26
Table 3.6 T-Test	27

## LIST OF FIGURES

Figure 2.1: Arduino usage	6
Figure 2.2: Quadcopter Structure	7
Figure 2.3: PID controller general equation	8
Figure 2.4: PID controller tuning process	9
Figure 2.5: Canny edge detector solution	10
Figure 3.1: Data Flow Diagram Part 1	18
Figure 3.2: Data Flow Diagram Part 2	19
Figure 3.3: Flow diagram of Canny edge detection	22
Figure 3.4: Camera to pc to microcontroller communication	23
Figure 3.5: %error formula	24

## **ABSTRACT**

The thesis involves the design of an automated indoor surveillance Quadcopter with image recognition using OpenCV via canny edge detection to avoid stationary obstacles an unmanned aerial vehicle surveillance drone which will be used for monitoring of a specified indoor area surveillance. The quadcopter will be designed using the DJI F450 airframe flamewheel frame mounted with an IP camera. Securing the area will be done through the IP camera (surveillance) which can also be a sensor to avoid stationary obstacles. The image recognition system will be used and applied as an algorithm for detecting the edges of the indoor area called canny edge detection. The quadcopter will be automated due to the microcontroller mini Gizduino w/ Atmega 328 which utilizes Arduino Integrated Development Environment and C/C++ language that will be used in programming the microcontroller for collision detection, avoidance, and automated navigation. Hobby King/Holybro MultiWii 328P Flight Controller is used for the flight stability.

**Keywords:** Drone, IP camera, Unmanned Aerial Vehicle, OpenCV, Canny edge detection



## **Chapter 1**

### **INTRODUCTION**

Indoor surveillance nowadays is becoming a need among establishments that stores valuable items either inside a vault or safe depending on the size of the item. Fixed surveillance systems or cameras were introduced to prevent heinous crimes such as theft with murder, robbery with maltreatment or theft only in happening. But, there are still problems about having an indoor surveillance system since it is fixed; there are blind spots that the surveillance camera may miss having a limited vision in the area and buying more indoor surveillance cameras and monitoring both can be a tedious work, thus a mobile surveillance device is recommended as an alternative source in fixed indoor surveillance systems.

Drones are widely used for search, surveillance, security and scout purposes. They are controlled either remotely or automated depending on the controlling medium the designer wants. In other countries such as USA that uses unmanned aerial vehicle. Hobbyists also use drones mostly unmanned aerial vehicle for their leisure time such as aerial photography or record videos in a bird's eye view with an improvised equipment such as frame, power supply, Arduino microcontroller/module, and digital camera or wireless spy cameras.

Small-scale wireless IP camera accepts orders send from a stationary central station then demand to capture single or many frames of digital image and sends the discrete image to the receiving central station for display. According to Agan, M.J., Olson, B.H., Pasqualino, and C.R., Stevens, G.L., "cameras today involves using CoMplementary Oxide Semiconductor or CMOS and Active Pixel Sensor or APS that have unwanted signal

discharge compared to charge coupled device or CCD with improved strength usage and act.” However, none of this study involves simple wireless or IP camera as a sensor that will trigger the Quadcopter device. The study will be using an IP camera and wireless router as a base station instead of a digital camera or wireless spy cameras as an alternative source mounted on an unmanned aerial vehicle for transferring the images in a personal computer, and that will send signal indicating the vehicle to the movement for quadcopter.

The general objectives are to create autonomous control to an unmanned aerial vehicle, to integrate an IP camera to an unmanned aerial vehicle, and to provide image recognition system using edge detection algorithm for image processing. The specific objectives are to create an unmanned aerial quadcopter that is controlled automatically via mini Gizduino w/ Atmega 328, to use OpenCV in digital signal processing for a simple video camera capture, and lastly, to use canny edge detection to avoid stationary obstacles and monitor the area.

Even for security and safety purposes such as monitoring hard to reach places that can be potential danger zone or maintaining a work area or any kind of area of an establishment into its desired order. This study will be helpful to the security on specific areas comparison to an ordinary security camera that provides flexible vision towards the area being monitored for military applications such as reconnaissance, strategic and tactical scouting and espionage. It provides early scouting in a post-disaster area in a search and rescue operations.

The study will use a simple IP camera that can transfer images of object obstruction for detection and its surroundings during a surveillance procedure. The micro controller mini Gizduino w/ Atmega 328 will be used to control the quadcopter automatically The Flight

Controller Hobby King/Holybro MultiWii 328P is used for the flight stability of the Quadcopter. This will be done in one room inside a building. Canny edge detection algorithm is used for the stationary obstacle avoidance and as an image recognition system. The communication for the flight controller used was a zigbee module and Wi-Fi IEEE 802.11b/g for the IP camera. The study is limited only to the power source for a Quadcopter battery 1800MAH~3600mAh 3S LiPo. The device should be limited to a building interior that does not have much interfering signals. The Quadcopter must be operated with having the lighting system of the area turned on to see the images properly. The Quadcopter will be sent inside a building in the assigned room by the user. The Quadcopter cannot see traps that are visually hidden which have the intention of destroying it. The range of the flight controller's communication and IP camera are different due to the zigbee module which has 200 meters of range used for the microcontroller and Wi-Fi IEEE 802.11b/g which has an indoor range of 35 and 38 meters used for the IP camera.

## **Chapter 2**

### **REVIEW OF RELATED LITERATURE**

#### **Unmanned Aerial Vehicle**

According to Mongkhun Qetkeaw A/L Vechian, this type of vehicle is well-known today, since many purposes of UAV can be used in diversity area such as save civilians, scouting, photography, farming and etc. In U.S Shoreline Sentry, unmanned aerial vehicles are placed with infrared equipment aid its mission to explore the mark. Researchers today tackle quad rotor aircraft compared to land mobile robots which is only limited to the ground while the quadcopter requires aerodynamics which is operated by thrust being produced by four motor placed far from each other like a pillar.

According to Mongkhun Qetkeaw A/L Vechian, “it has four input states and six output states ( $x, y, z, \theta, \psi, \omega$ ) and it is a complex system, since this enables the Quadcopter to lift more load.” Quadcopter has more benefits aside from the usual one main rotor copter, such as its physical design which is basic and has equally four propellers that provide surveillance capabilities. Quadcopter shifts direction by governing each propellers speed (Mongkhun Qetkeaw A/L Vechian, July 2012).

#### **Microcontroller**

A microcontroller is defined as a size of a microchip computer. Micro from the word itself is extremely small which is E-6 and a controller is a device used to direct or regulate instructions. Mainly, the microcontroller is capable of performing and saving a program. The

microcontroller consists of the same parts of a microprocessor (Alexandros Skafidas, December 2002).

According to Alexandros Skafidas (December, 2002) “programming a microcontroller normally involves the following steps:”

- Code and debug at a C++ type programming language. This is a tedious step.
- Compile the code into machine language in which a hex file is produced and read, while in this procedure, the compiler will be cautioned of errors.
- Burn the hex file into the microcontroller’s disk storage. Upload with a dedicated board which is attached to a PC’s port such as a COM or a parallel port.
- Check if the microcontroller tasks and behavioural patterns are the desired objectives. If not, the whole procedure will have to be repeated again. In distinction to a desktop, the code embedded onto the extremely small computer contains less resource as possible.

A study by Peter Jamieson showed, that the Arduino is widely preferred due to being an open source microcontroller platform based on its availability and easy to use hardware and software (Kishan Raj KC, October 2012).

Table 1  
THE LIST OF ACTIVITIES FOR THE STUDENTS

Activity type	Activity Goal	Group Size	Weeks	Arduino
Midterm	Interface with another chip or device	1 to 2	4	Yes
Final	Build a simple embedded system	1 to 2	4	Yes
Project	Build an embedded system	2 to 4	8-12	Yes
Class Activity 1	Design an alarm clock	1 to 2	1	Possibly
Class Activity 2	Design a remote control	1 to 2	1	Possibly
Class Activity 3	Design a led display	1 to 2	1	Possibly
Presentation 1	Present a peripheral	1 to 2	2	Yes
Presentation 2	Present an embedded system	1-2	2	No
Presentation 3	Present your final	1	1	Yes

**Figure 2.1: Arduino usage**

(Peter Jamieson, July 2011). (Kishan Raj KC, October 2012)

### **Flight Controller**

A flight controller contains an all in one microcontroller, gyroscope, and accelerometer which is a requirement in the flying of a multicopter from tricopter, quadcopter, and hexacopter. These eliminate the time consuming and resource intensive old-fashioned approach used to manage aircraft and large UAV system creating, implementing and authenticating the flight control system to achieve preferred objectives (Yew Chai Paw, December 2009).

### **Aerodynamics of a Quadcopter**

In a study done by Hongning Hou, Jian Zhuang, Hu Xia, Guanwei Wang, and Dehong Yu, Quadrotor aircraft is a strict plus-like structure consisting of four individual

motor used props, see Figure 2.2. Even if there are four drivers, it is unstable because of six coordinated outputs. It is imperative to reach all the objectives of controlling its motion coupling within the torque's rotation and movements, linking between turning twisting force and controls structure.

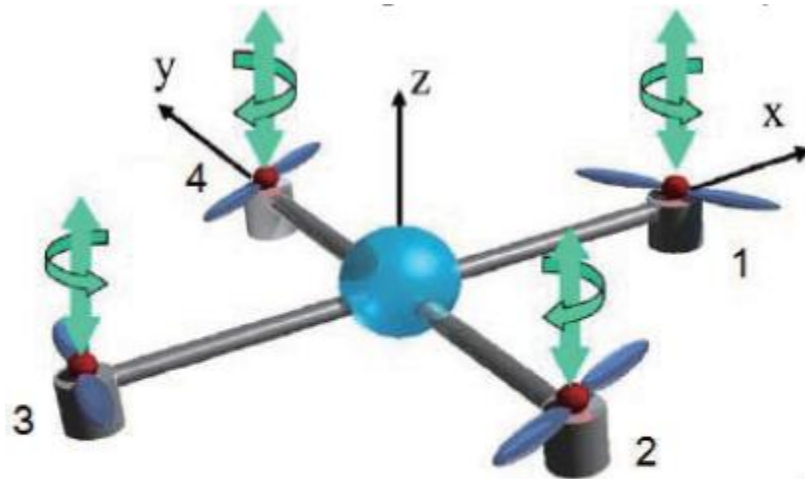


Figure 1 the structure of the quad-rotor

### Figure 2.2: Quadcopter Structure

As seen in Figure 2.2, two groups of props, the odd and even numbered props turns in the different way. By varying the prop's speed the quadcopter has varied lifts, several movements can be produced. Rising or lowering the speed of the rotation of each quadcopter simultaneously, vertical movements can be performed. Unlike when the speed of 2 and 4 changed side speed and rolled speed was performed. On the other hand, 1 and 3 performed pitch movement and other side speed. Yaw speed or movement is composite opposite force lever arm of the two groups of propellers up and down was performed by raising or lowering

the propellers simultaneously (Hongning Hou, Jian Zhuang, Hu Xia, Guanwei Wang, Dehong Yu, August 2010).

According to Barrett, Steven F., proportional, integral, derivative control or feedback system used in the industry today involves comparing the desired value to a preferred input of the user to where the device would maintain the value specified by the user. Given three variables: proportional, integral, derivative values P, I, and D for short applies to position control of DC brushless motor, toilet's flush system, control the position of a crane in an arcade place that is used to pick prizes or stuff toys. The general equation of the PID feedback/control system is as follows:

$$\text{Drive} = k_P * \text{Error} + k_I * \sum \text{Error} + k_D * dP/dT$$

where Error is the difference between the current value of the process variable (temperature, speed, position) and the desired set point, usually written as  $\text{Error} = (\text{Value} - \text{SetPoint})$ ;  $\sum \text{Error}$  is the summation of previous Error values; and  $dP/dT$  is the time rate of change of the process variable being controlled, or of the error itself. The proportional coefficient  $k_P$ , the integral coefficient  $k_I$ , and the derivative coefficient  $k_D$  are *gain* coefficients which *tune* the PID equation to the particular process being controlled. Drive is the total control effort (often a voltage or current) applied to actuators (heater, motor, valve) to achieve and hold the set point.

### **Figure 2.3: PID controller general equation**

Tuning a PID system, the tedious part of the feedback system in which not all three are used all the time and depend on the gain being generated by the device being used, even though there are logical processes, guidelines, and programs or softwares for choosing the variables gain, setting the preferred coefficient manually is still required.



**Coding a PID control algorithm:** Code for a PID system can be rather simple. The following is an example of some *pseudocode* to do PID:

```
PID:
  Error = Setpoint - Actual
  Integral = Integral + (Error*dt)
  Derivative = (Error - Previous_error)/dt
  Drive = (Error*kP) + (Integral*kI) + (Derivative*kD)
  Previous_error = Error
  wait(dt)
GOTO PID
```

**Tuning:** As mentioned earlier, the process of determining appropriate values for the gain coefficients  $k_P$ ,  $k_I$ , and  $k_D$  refers to “tuning” the system. A simple empirical approach is to start by zeroing the integral and derivative gains, and just use the proportional term. Setting the proportional gain increasingly higher will finally cause the system to oscillate. This is bad behavior; don’t allow it to continue. Reduce the proportional gain until you are just below the point of incipient oscillation. You can then try bringing up the derivative gain, which should act to forestall the start of oscillatory behavior. And finally adding a small amount of integral gain may help bring the system to the final set point.

The best coefficients for a given control system depend on the goals of the system. Bringing a subway train smoothly up to speed with no oscillations or overshoot is one goal; rapidly achieving a set point where some overshoot and oscillations are an acceptable tradeoff for fast response is a different goal. Different control goals require different tunings.

#### **Figure 2.4: PID controller tuning process**

(Barrett, Steven F., 2010).

### **Image processing**

According to U. Niethammer, S. Rothmund, U. Schwaderer, and J. Zeman, M. Joswig study, “in contemporary years, the use of UAV becomes accessible and the availability of small-scale digital equipments enabled UAVs to show reasonable and practical distant sensor-like board, giving adaptable and high resolution distant sensing investigations. UAVs distant sensing skirmishes significant numbers of air photography which consists of using image processing concepts, such as having a terrain model of digital which allows

surveillance of landslides easily and detailed (e.g. when using manual controlled UAVs). (September, 2011).

### **Canny Edge Detection**

In the article “An Improved Canny Algorithm for Edge Detection,” edge detection is an important part of the digital image processing. The edges are a group of pixel, whose surrounding area is quickly varying. The insides of the edge-dividing area are the same, while different areas have distinct classes. Edges are fundamental characteristics of an image (Ping Zhou, Wenjun Ye, Yaojie Xia, Qi Wang, May, 2011). While in the article “Enhanced Canny Edge Detection using Curvature Consistency,” canny edge detector is arguably the *de facto* standard for edge detectors. Its underlying process can be viewed as finding the zero crossings of the second derivative calculated in the gradient direction. A further process, hysteresis thresholding is sometimes used to dismiss weak edges whilst maintaining edge continuity. Before non-maximal suppression and hysteresis thresholding, the basic canny edge detector can be defined as the solution of:

$$\frac{\partial^2 I}{\partial g^2} = \mathbf{g} \left\{ \begin{array}{cc} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{array} \right\} \mathbf{g}^T = 0 \quad (1)$$

**Figure 2.5: Canny edge detector solution**

where  $g$  gradient direction and  $g$  also a scalar distance in that direction. Typically, there is a need to smooth the image prior to edge detection. This inevitably leads to a trade-off between

detecting weak and noisy edges, and oversmoothing genuine edges. In response to this, several researchers have presented adaptive smoothing techniques. The method presented in this paper uses local curvature consistency to adaptively control the smoothing of gradient estimates. The underlying idea is to use local variation in curvature labels to control the degree of smoothing applied to the gradient field. The researchers empirically demonstrate that this accomplishes the dual tasks of improving the curvature labeling of the image, and producing modified gradient estimates that can be used with the canny edge detector to find an improved set of edge contours (Philip L. Worthington, 2012).

### **Hough Line Transform**

Hough line transform is used to find the specific location of lines in an image or video which uses canny edge detection as basis for displaying the edges found within the vision of a camera.

According to Rabindra Kumar Murmur and Meena Jhaniya, “Hough transform, the points are related by finding the recline on the curve of stated figure. Opposite to the native study, where identified  $n$  dots of a picture is considered. To find the subgroup dots inside the straight line, a likely answer is search the total lines define by every couple of dots and then search the total subgroups of points that are near to the specified long straight marks. The difficulty for this process is that it involves finding  $n(n-1)/2 \sim n^2$  lines and then performing  $(n)(n(n-1)) \sim n^3$  comparisons of every dots to all lines. It is prohibited. Hough transform is a method that, can be used to find features of any figure in an image. It is only applied when searching for straight long marks or circles.” (Rabindra Kumar Murmur, Meena Jhaniya, 2009).

## **Digital and Wireless Camera**

This technology is a modern technology that uses digital signals to convert it to pixel to make an image of what is in the surroundings. The article “SMILE for the (wireless) camera” stated that, “wireless cameras have been in the analog age, but costly and limited usage is only available” (Davies, R., Oct.-Nov. 2004). There were also inventions such as low-power wireless video systems which means that it could be implemented for electronic communications. An increasing number of computer systems including multimedia capabilities for displaying and manipulating video (Meng, T.H., Jun 1998). According to Agan, M.J., Olson, B.H., Pasqualino, and C.R., Stevens, “small-scale wireless IP camera accepts orders sent from a stationary central station then demands to capture single or many frames of digital image and sends the discrete image to the receiving central station for display. According to Agan, M.J., Olson, B.H., Pasqualino, and C.R., Stevens, G.L., “cameras today involves using CoMplementary Oxide Semiconductor or CMOS and Active Pixel Sensor or APS that have unwanted signal discharge compared to charge coupled device or CCD with improved strength usage and act.” (Agan, M.J., Olson, B.H., Pasqualino, C.R., Stevens, G.L., Oct 1998).

## **Ultrasonic Sensor**

According to David Malgoza, Engers F Davance Mercedes, Stephen Smith, and Joshua West, “ultrasonic sensors work by broadcasting and waiting for the echo produced by the sensor itself. Aside from this is the preferred distance sensor its disadvantages are noise

due to multiple noise produced, and different areas (surface) it emits while receiving the data  
(David Malgoza, Engers F Davance Mercedes, Stephen Smith, Joshua West, March 2010).

## Chapter 3

### **Automated indoor Surveillance Quadcopter with image recognition using OpenCV via Canny edge detection for avoiding stationary obstacles**

#### **ABSTRACT**

The thesis involves the design of an automated indoor surveillance Quadcopter with image recognition using OpenCV via canny edge detection to avoid stationary obstacles an unmanned aerial vehicle surveillance drone which will be used for monitoring of a specified indoor area surveillance. The quadcopter will be designed using the DJI F450 airframe flamewheel frame mounted with an IP camera. Securing the area will be done through the IP camera (surveillance) which can also be a sensor to avoid stationary obstacles. The image recognition system will be used and applied as an algorithm for detecting the edges of the indoor area called canny edge detection. The quadcopter will be automated due to the microcontroller mini Gizduino w/ Atmega 328 which utilizes Arduino Integrated Development Environment and C/C++ language that will be used in programming the microcontroller for collision detection, avoidance, and automated navigation. Hobby King/Holybro MultiWii 328P Flight Controller is used for the flight stability.

**Keywords:** Drone, IP camera, Unmanned Aerial Vehicle, OpenCV , Canny edge detection

#### **Introduction**

Indoor surveillance nowadays is becoming a need among establishments that stores valuable items either inside a vault or safe depending on the size of the item. Fixed surveillance systems or cameras were introduced to prevent heinous crimes such as theft with murder, robbery with maltreatment or theft only in happening. But, there are still problems about having an indoor surveillance system since it is fixed; there are blind spots that the surveillance camera may miss having a limited vision in the area and buying more indoor

surveillance cameras and monitoring both can be a tedious work, thus a mobile surveillance device is recommended as an alternative source in fixed indoor surveillance systems.

Drones are widely used for search, surveillance, security and scout purposes. They are controlled either remotely or automated depending on the controlling medium the designer wants. In other countries such as USA that uses unmanned aerial vehicle. Hobbyists also use drones mostly unmanned aerial vehicle for their leisure time such as aerial photography or record videos in a bird's eye view with an improvised equipment such as frame, power supply, Arduino microcontroller/module, and digital camera or wireless spy cameras.

Small-scale wireless IP camera accepts orders send from a stationary central station then demand to capture single or many frames of digital image and sends the discrete image to the receiving central station for display. According to Agan, M.J., Olson, B.H., Pasqualino, and C.R., Stevens, G.L., "cameras today involves using CoMplementary Oxide Semiconductor or CMOS and Active Pixel Sensor or APS that have unwanted signal discharge compared to charge coupled device or CCD with improved strength usage and act." However, none of this study involves simple wireless or IP camera as a sensor that will trigger the Quadcopter device. The study will be using an IP camera and wireless router as a base station instead of a digital camera or wireless spy cameras as an alternative source mounted on an unmanned aerial vehicle for transferring the images in a personal computer, and that will send signal indicating the vehicle to the movement for quadcopter.

The general objectives are to create autonomous control to an unmanned aerial vehicle, to integrate an IP camera to an unmanned aerial vehicle, and to provide image recognition system using edge detection algorithm for image processing. The specific

objectives are to create an unmanned aerial quadcopter that is controlled automatically via mini Gizduino w/ Atmega 328, to use OpenCV in digital signal processing for a simple video camera capture, and lastly, to use canny edge detection to avoid stationary obstacles and monitor the area.

Even for security and safety purposes such as monitoring hard to reach places that can be potential danger zone or maintaining a work area or any kind of area of an establishment into its desired order. This study will be helpful to the security on specific areas comparison to an ordinary security camera that provides flexible vision towards the area being monitored for military applications such as reconnaissance, strategic and tactical scouting and espionage. It provides early scouting in a post-disaster area in a search and rescue operations.

The study will use a simple IP camera that can transfer images of object obstruction for detection and its surroundings during a surveillance procedure. The micro controller mini Gizduino w/ Atmega 328 will be used to control the quadcopter automatically The Flight Controller Hobby King/Holybro MultiWii 328P is used for the flight stability of the Quadcopter. This will be done in one room inside a building. Canny edge detection algorithm is used for the stationary obstacle avoidance and as an image recognition system. The communication for the flight controller used was a zigbee module and Wi-Fi IEEE 802.11b/g for the IP camera. The study is limited only to the power source for a Quadcopter battery 1800MAH~3600mAh 3S LiPo. The device should be limited to a building interior that does not have much interfering signals. The Quadcopter must be operated with having the lighting system of the area turned on to see the images properly. The Quadcopter will be sent inside a building in the assigned room by the user. The Quadcopter cannot see traps that are visually



hidden which have the intention of destroying it The range of the flight controller's communication and IP camera are different due to the zigbee module which has 200 meters of range used for the microcontroller and Wi-Fi IEEE 802.11b/g which has an indoor range of 35 and 38 meters used for the IP camera.

## Methodology

### Data Flow Diagram

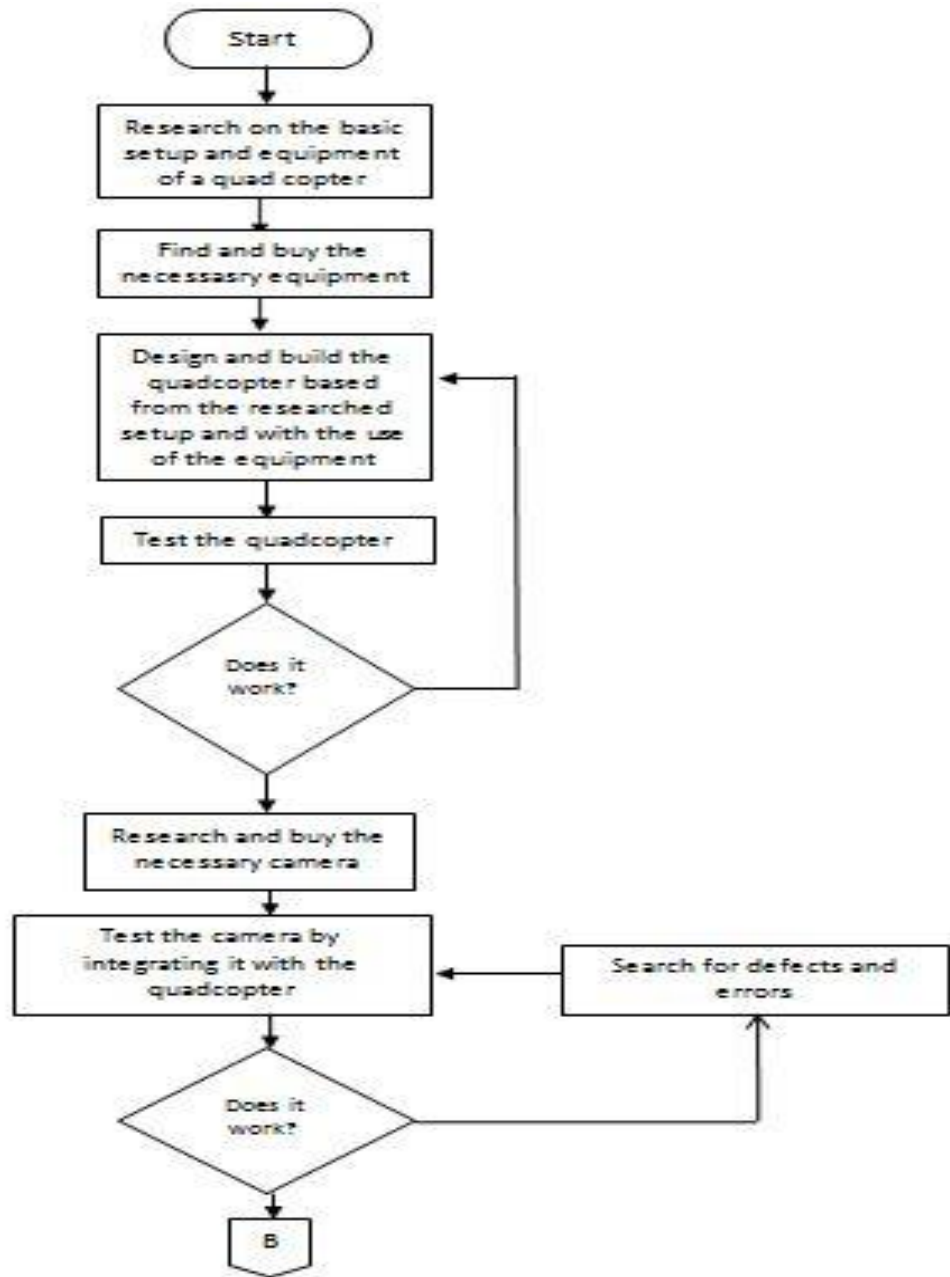
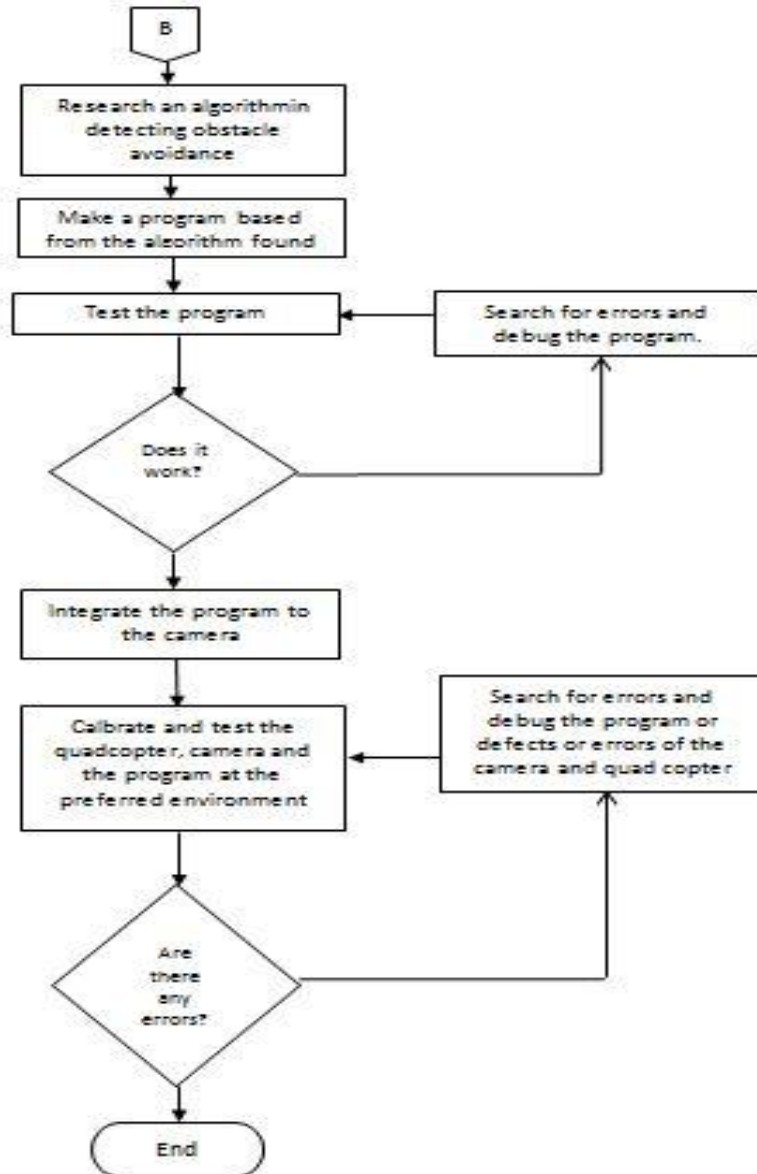


Figure 3.1: Data Flow Diagram Part 1



**Figure 3.2: Data Flow Diagram Part 2**

The proposed thesis will develop an unmanned aerial vehicle surveillance drone which will be used for monitoring an indoor building area to provide vision dynamically rather than in a fixed position which has limited vision towards an area and vulnerable to counter measurements, such as putting cover on the camera lens blocking the vision. With the provided instruction, putting the instruction through the

microcontroller based on the designers need, the Quadcopter will roam then hovers to monitor any suspicious activity or crimes occurring within the area or any individual roaming within the area who is either a threat or passing by only within the area provided with the user's discretion to declare a decision if spotted. It will be automated with the use of a microcontroller mini Gizduino w/ Atmega 328 as the controlling medium of the device. Hobby King/Holybro MultiWii 328P Flight Controller handles the stability of the quadcopter's flight. With the use of an IP camera for surveillance and as a sensor in avoiding stationary obstacles using image recognition program the canny edge detection to avoid stationary obstacles that could hinder the operation of the unmanned aerial vehicle.

## **Quadcopter**

The most common and basic setup of a Quadcopter are quadcopter frame, battery, brushless motors, propellers, ESC – electronic speed controller and the controller electronics (for controlling the ESC, receiving signal commands from the computer, etc.). Controller electronics, microcontroller (mini Gizduino w/ Atmega 328) will operate the ESC of the quadcopter since ESC controls the brushless motor movement of the quadcopter. Hobby King/Holybro MultiWii 328P Flight Controller controls the accelerometer and gyroscope for the stability of the quadcopter's flight. Since the equipment are available in the Philippines for the basic quadcopter, sensors and Zigbee will be added to the device while the controller that will be used is a mini Gizduino w/ Atmega 328 as the objective is to make the device work by itself alone. For the Quadcopter's frame the DJI F450 airframe flamewheel frame is use, the micro

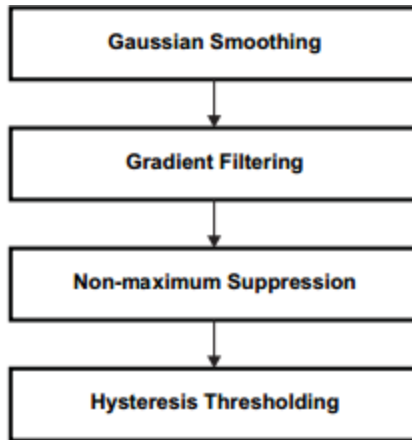
controller is mini Gizduino w/ Atmega 328 when the Quadcopter is in automatic mode.

Movements	Movement performed? Yes/No
Roll	Yes
Pitch	Yes
Yaw	Yes

**Table 3.1: Quadcopter Movements**

### **Canny Edge Detection**

Canny edge detection algorithm was the effective image sensor that would be applicable to the study. Every capture of the camera detection of the obstacles will be received by the laptop computer device. The computer device that has OpenCV will process the image of the camera that was captured by the IP camera which has an algorithm of Canny Edge Detection, which is the image detection and the program will output a signal to the com ports connected to the Zigbee for microcontroller interfacing.



**Figure 3.3: Flow diagram of Canny edge detection**

### **Collision Detection Circuit**

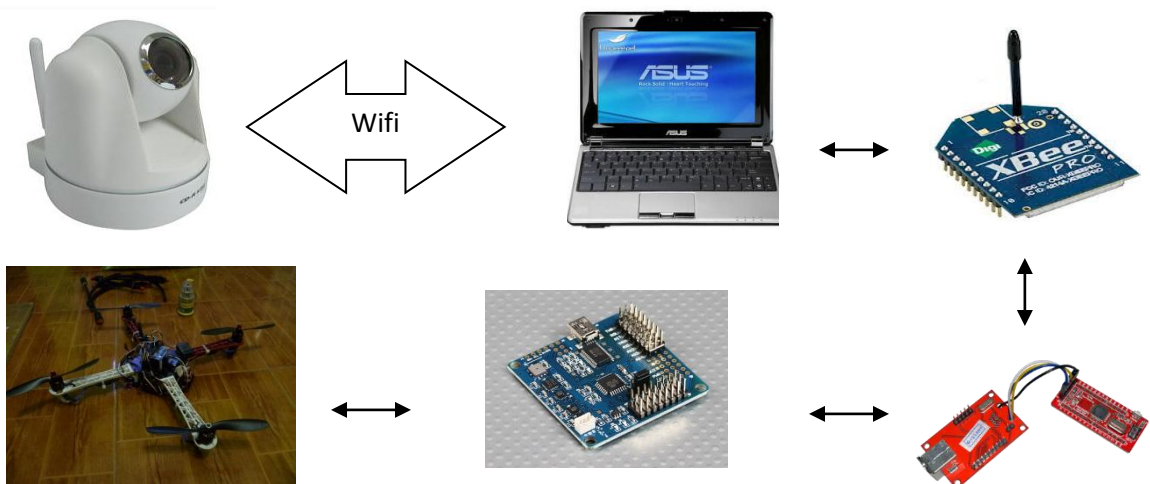
Sensors connected to a microcontroller which are placed in the areas not covered on the camera's quadcopter will be added to the quadcopter device for safety flight of the device. The microcontroller will be programmed in such a way the flight control of the Quadcopter will be safe. In this study, a camera that has a function like a CCTV will be attached to the Quadcopter and it will also act as a sensor that will make evasion to the obstacles that the device will encounter which the camera angle cannot capture. One of the sensors of the camera will be programmed in the computer that will send signals to the microcontroller. And the algorithm that will be used is the canny edge detection. Sensors will on placed to the left side most, right side most, and back side most.

Sensors	Sensor working? Yes/No
Left	Yes
Right	Yes
Back	Yes

**Table 3.2 Ultrasonic sonar testing**

### Image Detection Process

The process of how edges will be recognized through OpenCV image processing is by setting another image processing concept which is the Hough line transform which detects the specific edge of an object within collision range; the OpenCV will send data that signals a microcontroller to move the quadcopter to avoid the object obstacle. It will continuously send signal as long as there is no detected vertical edge within the collision range. For recognizing the vertical edge, the Quadcopter will go from either left or right until the vertical edge will be gone.



**Figure 3.4: Camera to pc to microcontroller communication**

## Results and Discussion

The quadcopter undergoes testing and calibration procedures. Whether the objectives are met which consists of two testing procedures, its altitude hold and obstacle avoided. The first testing procedure checks if the quadcopter is capable of surveillance through hovering within a specific altitude using an ultrasonic sensor to check the height in real-time. To have a desired vision of the surroundings, it monitors the use of an %error formula to determine how accurate the data is.

$$\% \text{ error} = \frac{\text{actual value} - \text{measured value}}{\text{actual value}} \times 100\%$$

**Figure 3.5: %error formula**

Time	Quadcopter Altitude using Ultrasonic sensor					
	0.3 meter		0.5 meters		0.7 meters	
	Actual	Experimental	Actual	Experimental	Actual	Experimental
1	0.3	0.35	0.5	0.52	0.7	0.68
2	0.3	0.34	0.5	0.5	0.7	0.7
3	0.3	0.32	0.5	0.48	0.7	0.7
4	0.3	0.3	0.5	0.49	0.7	0.71
5	0.3	0.28	0.5	0.5	0.7	0.73
6	0.3	0.29	0.5	0.5	0.7	0.73
7	0.3	0.31	0.5	0.51	0.7	0.71
8	0.3	0.33	0.5	0.53	0.7	0.71
9	0.3	0.34	0.5	0.52	0.7	0.7
10	0.3	0.32	0.5	0.52	0.7	0.7
Average	0.3	<b>0.318</b>	0.5	<b>0.507</b>	0.7	<b>0.707</b>
% Error	<b>6</b>		<b>1.4</b>		<b>1</b>	

**Table 3.3 Measurement of altitude hold of quadcopter**



Lastly, stationary obstacle avoidance is checked after the quadcopter hovers and moves to a flight pattern given and places an object preferably having a vertical edge near the collision range of the quadcopter's camera, then the quadcopter will evade either from left and right depending on the location of the vertical edge detected by the quadcopter's camera.

<b>Trial</b>	<b>Quadcopter Stationary Obstacle Avoided? Yes/No</b>
1	Yes
2	Yes
3	Yes
4	Yes
5	Yes
6	Yes
7	Yes
8	No
9	No
10	Yes

**Table 3.4 Quadcopter Stationary Obstacle Avoidance Result (One Edge detected)**

Failed results trial eight and nine were due testing to a dim lighting system and the IP camera sending the data to the laptop used is sluggish that resulted in accident crash landings.

<b>Trial</b>	<b>Quadcopter Stationary Obstacle Avoided? Yes/No</b>
1	Yes
2	Yes
3	Yes
4	Yes
5	Yes
6	Yes
7	Yes
8	Yes
9	Yes
10	Yes

**Table 3.5 Quadcopter Obstacle Avoidance Result (Two Edge detected)**

Two stationary obstacles were placed in a small obstacle course like pattern placing the first right side then left side far from each other within the IP camera vision to check the effectiveness of the obstacle avoidance. Tested in a properly lit area the quadcopter evades the obstacle the first edge of a stationary obstacle detected then executing evasive measurement which will evade left or right.

One-Sample two tailed T-test was used to determine if there was a significant difference in the quadcopter's flight to the prototype.

The hypotheses to be tested are:

- Null hypothesis:  $H_0: \mu = \text{Actual altitude} = \text{Experimental altitude}$
- Alternative hypothesis:  $H_1: \text{Actual altitude} \neq \text{Experimental altitude}$

No. Of Samples	10	10	10
Mean	0.318	0.507	0.707
Standard Deviation	0.023	0.0157	0.0149
Degree of Freedom	9	9	9
T value	-2.4751	1.4126	1.4812
P value (two-tailed)	0.0353	0.1914	0.1727

**Table 3.6 T-Test**

The data shows that the actual altitude of the quadcopter is reached with the prototype quadcopter, using the hypotheses developed are therefore proven. Small T- value presents that similar group are tested. Standard deviation is almost zero or low value which means that the data used and the mean is not far from each other. P value showing less than five percent shows that the data is not random and it is a proof for the evidence of the null hypothesis to be moderate.

With quadcopter parts and kits easily accessible in the Philippines even with smaller frame or build, mobile flying surveillance will be an alternative instead of placing fixed surveillance cameras since the vision is not limited to a specific place and it can monitor wherever the user wants to with the capability to avoid stationary obstacles automatically to prevent accidents from happening, while the quadcopter monitors the area lessening the user's worries. Drones today are also becoming a trend to the military as surveillance for rescue and scouting missions to help them in covering hard-to-reach areas to search safely and effectively since the Philippines has many islands where travelling by air aside from sea is preferred. It is also powered electrically which is environment friendly and can be changed to solar power since the Philippines is a tropical country. When dry season occurs this can save energy and resources.

The quadcopter is limited to one room inside a building. The battery can last from three to six minutes. Zigbee is used for communicating medium of the microcontroller while the camera uses Wifi where both can have different ranges; Zigbee's range is longer than Wifi. The system does not notify or alerts the intruder as it is within the user's discretion to conclude if there is a threat or intruder present within the room. Preferred altitudes 0.3, 0.5, and 0.7 meters based from a stand are used for safety measures. Lighting of the room is needed for the image recognition system to identify the edges. The user's judgment is needed to declare a foreign object within the surrounding which is a threat or not. Invisible traps cannot be seen unless there is irregularity between the surroundings which make the edges not fit to the area being monitored.

## **Chapter 4**

### **CONCLUSION**

Fixed indoor surveillance in establishments are not enough to eliminate threats of being trespass or ransack. One surveillance camera has only a fixed point of view in which blind spots is present, meaning the solution will be to buy more surveillance cameras to compensate for the lack of vision of the other camera but it will be both tedious to monitor and maintain. By providing mobile indoor surveillance introduces new alternative for indoor surveillance and opportunities to new technology. This thesis paper met the objectives of creating an autonomous control to an unmanned aerial vehicle due to the microcontroller used which is the mini Gizduino w/ Atmega 328 as its controlling medium; to provide an image recognition system by integrating an IP camera to the unmanned aerial vehicle with the use of an edge detection algorithm for the image processing which is canny edge detection using OpenCV, a library of programming function that aims real-time computer vision which includes image processing/recognition and provides a simple camera capture. Lastly, utilizing canny edge detection together with Hough Line Transform provides the solution of monitoring the area by detecting the specific edge in avoiding stationary obstacles.

## **Chapter 5**

### **RECOMMENDATION**

Further, development and improvement of the device will refine this thesis paper. It aims mobile surveillance which is recommended either indoor or outdoor with the growing technology of robotics that will help in scouting, conducting surveillance, and detecting intruders/trespassers which will be an asset even to the military. Longer lifespan of battery is needed for the quadcopter to offer further surveillance within the specified area, and should be light to prevent additional payload to the quadcopter to improve its mobility. The researchers recommend the frame to be used should be smaller than the DJI 450 framewheel or improvise to what desire since the components are small. Further research to canny edge detection is recommended due to its opportunity in detecting a camouflaged intruder/trespasser since edges are one of the key factors to discover if there are inconsistencies within the surroundings of the area being observed, especially camouflaged colors which are one of the factors in producing optical illusion to the human eye or a fixed surveillance camera. IP camera or camera with high frame rate and/or faster communication medium with the microcontroller are also recommended.

## References

- [1] Mongkhun Qetkeaw A/L Vechian (2012). WIRELESS CONTROL QUADCOPTER WITH STEREO CAMERA AND SELF-BALANCING SYSTEM (PDF). 1-2
- [2] Alexandros Skafidas (2002). Microcontroller Systems for a UAV (PDF). 4-6
- [3] Peter Jamieson (2011) Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat? (PDF). 3-4
- [4] Kishan Raj KC (2012) CONTROLLING A ROBOT USING ANDROID INTERFACE AND VOICE (PDF). 14
- [5] Yew Chai Paw (2009) Synthesis and Validation of Flight Control for UAV (PDF). Iv
- [6] Hongning Hou, Jian Zhuang, Hu Xia, Guanwei Wang, Dehong Yu (2010) A Simple Controller of Minisize Quad-Rotor Vehicle (PDF). 1-3
- [7] Barrett, Steven F. (2010) Arduino Microcontroller:Processing for Everyone. Morgan & Claypool, pp.325
- [8] Wiesel (1985) Digital image processing for orthophoto generation. Photogrammetria (PDF) 69-76

- [9] U. Niethammer, S. Rothmund, U. Schwaderer, J. Zeman, M. Joswig (2011) OPEN SOURCE IMAGE-PROCESSING TOOLS FOR LOW-COST UAV-BASED LANDSLIDE INVESTIGATIONS (PDF) 2-4
- [10] Ping Zhou, Wenjun Ye, Yaojie Xia, Qi Wang (2011) An Improved Canny Algorithm for Edge Detection (PDF) 1-5
- [11] Philip L. Worthington (2012) Enhanced Canny Edge Detection using Curvature Consistency (PDF) 1-4
- [12] Rabindra Kumar Murmur, Meena Jhaniya (2009) IMAGE SEGMENTATION USING HOUGH TRASFORM (PDF) 17-20
- [13] (Davies, R (2004) Smile for the (wireless) camera [Digital broadcasting] (PDF) 1-4
- [14] Agan, M.J., Olson, B.H., Pasqualino, C.R., Stevens, G.L. (1998) A highly miniaturized, battery operated, commandable, digital wireless camera (PDF) 1-5
- [15] David Malgoza, Engers F Davance Mercedes, Stephen Smith, Joshua West (2010) Quad-Copter Autonomous Surveillance Robot (PDF) 56

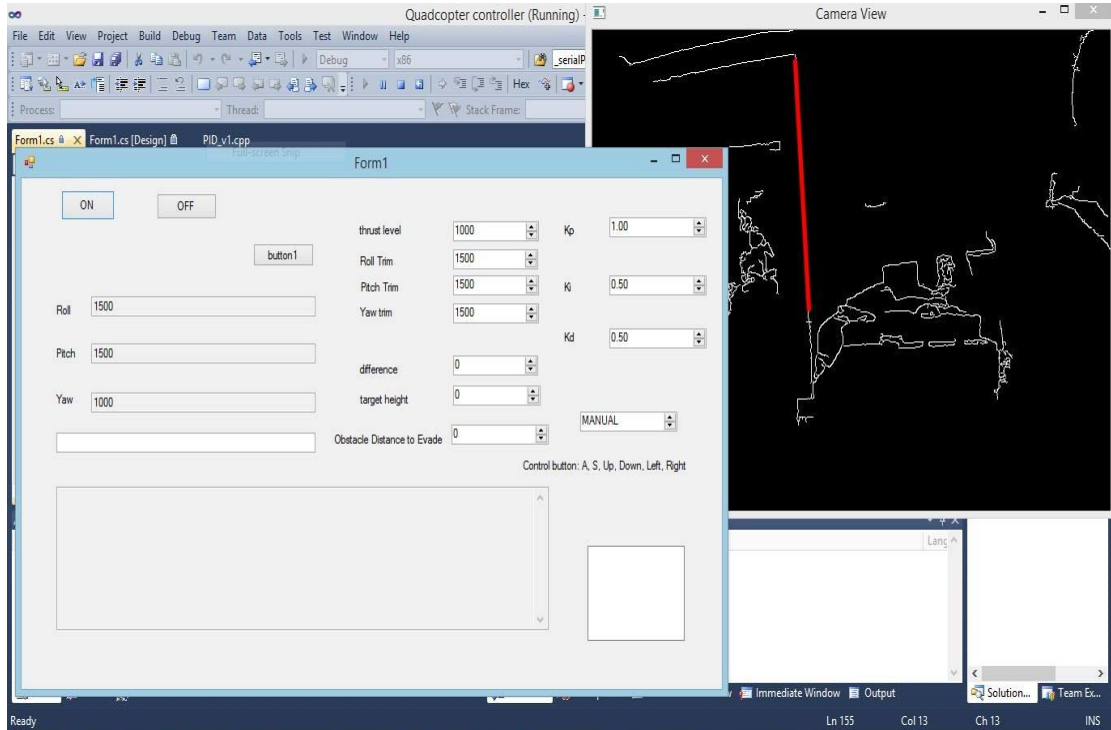


## APPENDIX

### A. User Manual

1. Check first if the battery of the quadcopter has enough charge for flight by plugging the battery to the battery charge counter provided within the quadcopter. If no charge then recharge to full cell also, do not overcharge it will damage the battery rendering it useless.
2. Plug all the necessary connections needed before running the GUI such as the connector between the microcontroller and zigbee (Receiver part) for the quadcopter and the transmitter part to the desktop or laptop, it is indicated in the program where COM port to plug since the orientation readings of a desktop system unit and laptop ports varies.
3. Connect to the IP address of the IP camera turning it on first to prevent errors while the image processing program runs.
4. Run the executable programs ImageProcessing.sln and Quadcopter controller.sln whoever runs first is fine.

## 5. A GUI will appear together with the IP camera image processing vision



6. To start the flight click the ON button which the quadcopter will spin first at a thrust level indicated which is 1000 you can change it as desired.
7. Adjust the target height or altitude for what height to maintain in centimeters
8. Note: Roll is the left and right movement of the quadcopter, pitch is the backward movement then Yaw is the spin or torque movement of the quadcopter.
9.  $K_p$ ,  $K_i$ ,  $K_d$  represents the PID feedback system of the quadcopter use with the automatic feature of the quadcopter to maintain stabilization altitude.
10. Trim and difference is use to contradict the opposing force that results in instability of the quadcopter use for tuning purposes.
11. Use manual if not using the feedback system.



12. Pictures of the prototype are indicated to provide visual aid on what should the appearance be.



## B. Quadcopter GUI

### Quad GUI (Program.cs)

```
using System;  
  
using System.Collections.Generic;  
  
using System.Linq;  
  
using System.Windows.Forms;  
  
namespace Quadcopter_controller  
{
```

```

static class Program
{
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1());
    }
}

```

### **Quad GUI (Form 1.Designer.cs)**

```

namespace Quadcopter_controller
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

```

```

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)
{
    if (disposing && (components != null))
    {
        components.Dispose();
    }
    base.Dispose(disposing);
}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()
{

```

```

this.components = new System.ComponentModel.Container();

this.OnButton = new System.Windows.Forms.Button();

this.serialPort1 = new System.IO.Ports.SerialPort(this.components);

this.OffButton = new System.Windows.Forms.Button();

this.RollTB = new System.Windows.Forms.TextBox();

this.label6 = new System.Windows.Forms.Label();

this.label7 = new System.Windows.Forms.Label();

this.PitchTB = new System.Windows.Forms.TextBox();

this.ThrottleValue = new System.Windows.Forms.NumericUpDown();

this.label8 = new System.Windows.Forms.Label();

this.label1 = new System.Windows.Forms.Label();

this.YawTB = new System.Windows.Forms.TextBox();

this.textBox1 = new System.Windows.Forms.TextBox();

this.textBox2 = new System.Windows.Forms.TextBox();

this.RollTrim = new System.Windows.Forms.NumericUpDown();

this.PitchTrim = new System.Windows.Forms.NumericUpDown();

this.YawTrim = new System.Windows.Forms.NumericUpDown();

this.ValueChanger = new
System.Windows.Forms.NumericUpDown();

this.label2 = new System.Windows.Forms.Label();

this.label3 = new System.Windows.Forms.Label();

this.label4 = new System.Windows.Forms.Label();

this.label5 = new System.Windows.Forms.Label();

```

```

this.label9 = new System.Windows.Forms.Label();
this.label10 = new System.Windows.Forms.Label();
this.height = new System.Windows.Forms.NumericUpDown();
this.label11 = new System.Windows.Forms.Label();
this.Kp = new System.Windows.Forms.NumericUpDown();
this.label12 = new System.Windows.Forms.Label();
this.Ki = new System.Windows.Forms.NumericUpDown();
this.Kd = new System.Windows.Forms.NumericUpDown();
this.label13 = new System.Windows.Forms.Label();
this.ModeUpDown = new
System.Windows.Forms.DomainUpDown();
this.label14 = new System.Windows.Forms.Label();
this.ObsUpDown = new System.Windows.Forms.NumericUpDown();
this.listBox1 = new System.Windows.Forms.ListBox();
this.button1 = new System.Windows.Forms.Button();

((System.ComponentModel.ISupportInitialize)(this.ThrottleValue)).BeginInit(
);

((System.ComponentModel.ISupportInitialize)(this.RollTrim)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.PitchTrim)).BeginInit();

```



```

((System.ComponentModel.ISupportInitialize)(this.YawTrim)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.ValueChanger)).BeginInit
();

((System.ComponentModel.ISupportInitialize)(this.height)).BeginInit();

        ((System.ComponentModel.ISupportInitialize)(this.Kp)).BeginInit();
        ((System.ComponentModel.ISupportInitialize)(this.Ki)).BeginInit();
        ((System.ComponentModel.ISupportInitialize)(this.Kd)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.ObsUpDown)).BeginInit()
;

        this.SuspendLayout();

        //

        // OnButton

        //

        this.OnButton.Location = new System.Drawing.Point(49, 12);

        this.OnButton.Name = "OnButton";

        this.OnButton.Size = new System.Drawing.Size(66, 30);

        this.OnButton.TabIndex = 0;

        this.OnButton.Text = "ON";

        this.OnButton.UseVisualStyleBackColor = true;

```

```

        this.OnButton.Click += new
System.EventHandler(this.OnButton_Click);

//

// serialPort1

//

this.serialPort1.PortName = "COM3";

this.serialPort1.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(this.serialPort1_DataRecei
ved_1);

//

// OffButton

//

this.OffButton.Location = new System.Drawing.Point(167, 15);

this.OffButton.Name = "OffButton";

this.OffButton.Size = new System.Drawing.Size(74, 26);

this.OffButton.TabIndex = 9;

this.OffButton.Text = "OFF";

this.OffButton.UseVisualStyleBackColor = true;

this.OffButton.Click += new
System.EventHandler(this.OffButton_Click);

//

// RollTB

//

```

```

this.RollTB.Location = new System.Drawing.Point(86, 118);

this.RollTB.Name = "RollTB";

this.RollTB.ReadOnly = true;

this.RollTB.Size = new System.Drawing.Size(277, 20);

this.RollTB.TabIndex = 15;

this.RollTB.Text = "1500";

this.RollTB.TextChanged += new
System.EventHandler(this.RollTB_TextChanged_1);

//

// label6

//

this.label6.AutoSize = true;

this.label6.Location = new System.Drawing.Point(40, 125);

this.label6.Name = "label6";

this.label6.Size = new System.Drawing.Size(25, 13);

this.label6.TabIndex = 17;

this.label6.Text = "Roll";

//

// label7

//

this.label7.AutoSize = true;

this.label7.Location = new System.Drawing.Point(40, 168);

this.label7.Name = "label7";

```

```

this.label7.Size = new System.Drawing.Size(31, 13);

this.label7.TabIndex = 18;

this.label7.Text = "Pitch";

//

// PitchTB

//

this.PitchTB.Location = new System.Drawing.Point(86, 165);

this.PitchTB.Name = "PitchTB";

this.PitchTB.ReadOnly = true;

this.PitchTB.Size = new System.Drawing.Size(277, 20);

this.PitchTB.TabIndex = 19;

this.PitchTB.Text = "1500";

this.PitchTB.TextChanged += new
System.EventHandler(this.PitchTB_TextChanged_1);

//

// ThrottleValue

//

this.ThrottleValue.Increment = new decimal(new int[] {
10,
0,
0,
0});

this.ThrottleValue.Location = new System.Drawing.Point(533, 43);

```

```

this.ThrottleValue.Maximum = new decimal(new int[] {
    1900,
    0,
    0,
    0});

this.ThrottleValue.Minimum = new decimal(new int[] {
    1000,
    0,
    0,
    0});

this.ThrottleValue.Name = "ThrottleValue";

this.ThrottleValue.Size = new System.Drawing.Size(106, 20);

this.ThrottleValue.TabIndex = 22;

this.ThrottleValue.Value = new decimal(new int[] {
    1000,
    0,
    0,
    0});

this.ThrottleValue.ValueChanged += new
System.EventHandler(this.ThrottleValue_ValueChanged);

//

// label8

//

```

```
this.label8.AutoSize = true;

this.label8.Location = new System.Drawing.Point(415, 45);

this.label8.Name = "label8";

this.label8.Size = new System.Drawing.Size(58, 13);

this.label8.TabIndex = 23;

this.label8.Text = "thrust level";

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(40, 214);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(28, 13);

this.label1.TabIndex = 25;

this.label1.Text = "Yaw";

//

// YawTB

//

this.YawTB.Location = new System.Drawing.Point(86, 214);

this.YawTB.Name = "YawTB";

this.YawTB.ReadOnly = true;

this.YawTB.Size = new System.Drawing.Size(277, 20);

this.YawTB.TabIndex = 26;
```

```

        this.YawTB.Text = "1000";

        this.YawTB.TextChanged += new
System.EventHandler(this.YawTB_TextChanged);

        //

        // textBox1

        //

        this.textBox1.Location = new System.Drawing.Point(43, 308);

        this.textBox1.Multiline = true;

        this.textBox1.Name = "textBox1";

        this.textBox1.ReadOnly = true;

        this.textBox1.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;

        this.textBox1.Size = new System.Drawing.Size(608, 143);

        this.textBox1.TabIndex = 29;

        //

        // textBox2

        //

        this.textBox2.Location = new System.Drawing.Point(43, 254);

        this.textBox2.Name = "textBox2";

        this.textBox2.Size = new System.Drawing.Size(320, 20);

        this.textBox2.TabIndex = 30;

        //

        // RollTrim

```

```

//
this.RollTrim.Location = new System.Drawing.Point(533, 71);
this.RollTrim.Maximum = new decimal(new int[] {
1800,
0,
0,
0});
this.RollTrim.Minimum = new decimal(new int[] {
1000,
0,
0,
0});
this.RollTrim.Name = "RollTrim";
this.RollTrim.Size = new System.Drawing.Size(105, 20);
this.RollTrim.TabIndex = 31;
this.RollTrim.Value = new decimal(new int[] {
1500,
0,
0,
0});
this.RollTrim.ValueChanged += new
System.EventHandler(this.RollTrim_ValueChanged);
//

```



```

// PitchTrim

//
this.PitchTrim.Location = new System.Drawing.Point(533, 97);
this.PitchTrim.Maximum = new decimal(new int[] {
1800,
0,
0,
0});
this.PitchTrim.Minimum = new decimal(new int[] {
1000,
0,
0,
0});
this.PitchTrim.Name = "PitchTrim";
this.PitchTrim.Size = new System.Drawing.Size(106, 20);
this.PitchTrim.TabIndex = 32;
this.PitchTrim.Value = new decimal(new int[] {
1500,
0,
0,
0});
this.PitchTrim.ValueChanged += new
System.EventHandler(this.PitchTrim_ValueChanged);

```

```
//  
  
// YawTrim  
  
//  
this.YawTrim.Location = new System.Drawing.Point(533, 125);  
this.YawTrim.Maximum = new decimal(new int[] {  
1800,  
0,  
0,  
0});  
this.YawTrim.Minimum = new decimal(new int[] {  
1000,  
0,  
0,  
0});  
this.YawTrim.Name = "YawTrim";  
this.YawTrim.Size = new System.Drawing.Size(106, 20);  
this.YawTrim.TabIndex = 34;  
this.YawTrim.Value = new decimal(new int[] {  
1500,  
0,  
0,  
0});
```

```

        this.YawTrim.ValueChanged += new
System.EventHandler(this.YawTrim_ValueChanged);

//

// ValueChanger

//

this.ValueChanger.Location = new System.Drawing.Point(533, 177);

this.ValueChanger.Name = "ValueChanger";

this.ValueChanger.Size = new System.Drawing.Size(105, 20);

this.ValueChanger.TabIndex = 35;

this.ValueChanger.ValueChanged += new
System.EventHandler(this.ValueChanger_ValueChanged);

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(415, 184);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(54, 13);

this.label2.TabIndex = 36;

this.label2.Text = "difference";

//

// label3

//

```

```
this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(415, 127);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(47, 13);

this.label3.TabIndex = 37;

this.label3.Text = "Yaw trim";

//

// label4

//

this.label4.AutoSize = true;

this.label4.Location = new System.Drawing.Point(415, 76);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(48, 13);

this.label4.TabIndex = 38;

this.label4.Text = "Roll Trim";

//

// label5

//

this.label5.AutoSize = true;

this.label5.Location = new System.Drawing.Point(416, 102);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(54, 13);

this.label5.TabIndex = 39;
```

```
this.label5.Text = "Pitch Trim";  
  
//  
  
// label9  
  
//  
  
this.label9.AutoSize = true;  
  
this.label9.Location = new System.Drawing.Point(617, 280);  
  
this.label9.Name = "label9";  
  
this.label9.Size = new System.Drawing.Size(208, 13);  
  
this.label9.TabIndex = 40;  
  
this.label9.Text = "Control button: A, S, Up, Down, Left, Right";  
  
//  
  
// label10  
  
//  
  
this.label10.AutoSize = true;  
  
this.label10.Location = new System.Drawing.Point(416, 214);  
  
this.label10.Name = "label10";  
  
this.label10.Size = new System.Drawing.Size(66, 13);  
  
this.label10.TabIndex = 41;  
  
this.label10.Text = "target height";  
  
//  
  
// height  
  
//  
  
this.height.Location = new System.Drawing.Point(533, 207);
```

```

this.height.Name = "height";

this.height.Size = new System.Drawing.Size(108, 20);

this.height.TabIndex = 42;

this.height.ValueChanged += new
System.EventHandler(this.height_ValueChanged);

//

// label11

//

this.label11.AutoSize = true;

this.label11.Location = new System.Drawing.Point(667, 45);

this.label11.Name = "label11";

this.label11.Size = new System.Drawing.Size(20, 13);

this.label11.TabIndex = 43;

this.label11.Text = "Kp";

//

// Kp

//

this.Kp.DecimalPlaces = 2;

this.Kp.Increment = new decimal(new int[] {

1,

0,

0,

131072});

```

```

this.Kp.Location = new System.Drawing.Point(726, 39);
this.Kp.Name = "Kp";
this.Kp.Size = new System.Drawing.Size(120, 20);
this.Kp.TabIndex = 44;
this.Kp.Value = new decimal(new int[] {
    1,
    0,
    0,
    0});
this.Kp.ValueChanged += new
System.EventHandler(this.Kp_ValueChanged);
//
// label12
//
this.label12.AutoSize = true;
this.label12.Location = new System.Drawing.Point(667, 102);
this.label12.Name = "label12";
this.label12.Size = new System.Drawing.Size(16, 13);
this.label12.TabIndex = 45;
this.label12.Text = "Ki";
//
// Ki
//

```

```

this.Ki.DecimalPlaces = 2;

this.Ki.Increment = new decimal(new int[] {
1,
0,
0,
131072});

this.Ki.Location = new System.Drawing.Point(726, 97);

this.Ki.Name = "Ki";

this.Ki.Size = new System.Drawing.Size(120, 20);

this.Ki.TabIndex = 46;

this.Ki.Value = new decimal(new int[] {
5,
0,
0,
65536});

this.Ki.ValueChanged += new
System.EventHandler(this.Ki_ValueChanged);

//
// Kd
//

this.Kd.DecimalPlaces = 2;

this.Kd.Increment = new decimal(new int[] {
1,

```



```

0,
0,
131072});

this.Kd.Location = new System.Drawing.Point(726, 150);
this.Kd.Name = "Kd";
this.Kd.Size = new System.Drawing.Size(120, 20);
this.Kd.TabIndex = 47;
this.Kd.Value = new decimal(new int[] {
5,
0,
0,
65536});

this.Kd.ValueChanged += new
System.EventHandler(this.Kd_ValueChanged_1);

//
// label13
//
this.label13.AutoSize = true;
this.label13.Location = new System.Drawing.Point(667, 152);
this.label13.Name = "label13";
this.label13.Size = new System.Drawing.Size(20, 13);
this.label13.TabIndex = 48;
this.label13.Text = "Kd";

```

```

//
// ModeUpDown
//
this.ModeUpDown.Items.Add("MANUAL");
this.ModeUpDown.Items.Add("AUTOMATIC");
this.ModeUpDown.Location = new System.Drawing.Point(690, 233);
this.ModeUpDown.Name = "ModeUpDown";
this.ModeUpDown.Size = new System.Drawing.Size(120, 20);
this.ModeUpDown.TabIndex = 49;
this.ModeUpDown.Text = "MANUAL";
this.ModeUpDown.SelectedItemChanged += new
System.EventHandler(this.ModeUpDown_SelectedItemChanged);
//
// label14
//
this.label14.AutoSize = true;
this.label14.Location = new System.Drawing.Point(384, 254);
this.label14.Name = "label14";
this.label14.Size = new System.Drawing.Size(140, 13);
this.label14.TabIndex = 50;
this.label14.Text = "Obstacle Distance to Evade";
//
// ObsUpDown

```

```

//
this.ObsUpDown.Location = new System.Drawing.Point(531, 246);
this.ObsUpDown.Maximum = new decimal(new int[] {
200,
0,
0,
0});
this.ObsUpDown.Name = "ObsUpDown";
this.ObsUpDown.Size = new System.Drawing.Size(120, 20);
this.ObsUpDown.TabIndex = 51;
this.ObsUpDown.ValueChanged += new
System.EventHandler(this.ObsUpDown_ValueChanged);
//
// listBox1
//
this.listBox1.FormattingEnabled = true;
this.listBox1.Location = new System.Drawing.Point(699, 367);
this.listBox1.Name = "listBox1";
this.listBox1.Size = new System.Drawing.Size(120, 95);
this.listBox1.TabIndex = 52;
//
// button1
//

```

```
this.button1.Location = new System.Drawing.Point(286, 65);  
  
this.button1.Name = "button1";  
  
this.button1.Size = new System.Drawing.Size(75, 23);  
  
this.button1.TabIndex = 53;  
  
this.button1.Text = "button1";  
  
this.button1.UseVisualStyleBackColor = true;  
  
this.button1.Click += new System.EventHandler(this.button1_Click);  
  
//  
  
// Form1  
  
//  
  
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);  
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;  
this.ClientSize = new System.Drawing.Size(929, 478);  
  
this.Controls.Add(this.button1);  
  
this.Controls.Add(this.listBox1);  
  
this.Controls.Add(this.ObsUpDown);  
  
this.Controls.Add(this.label14);  
  
this.Controls.Add(this.ModeUpDown);  
  
this.Controls.Add(this.label13);  
  
this.Controls.Add(this.Kd);  
  
this.Controls.Add(this.Ki);  
  
this.Controls.Add(this.label12);  
  
this.Controls.Add(this.Kp);
```

```
this.Controls.Add(this.label11);  
this.Controls.Add(this.height);  
this.Controls.Add(this.label10);  
this.Controls.Add(this.label9);  
this.Controls.Add(this.label5);  
this.Controls.Add(this.label4);  
this.Controls.Add(this.label3);  
this.Controls.Add(this.label2);  
this.Controls.Add(this.ValueChanger);  
this.Controls.Add(this.YawTrim);  
this.Controls.Add(this.PitchTrim);  
this.Controls.Add(this.RollTrim);  
this.Controls.Add(this.textBox2);  
this.Controls.Add(this.textBox1);  
this.Controls.Add(this.YawTB);  
this.Controls.Add(this.label1);  
this.Controls.Add(this.label8);  
this.Controls.Add(this.ThrottleValue);  
this.Controls.Add(this.PitchTB);  
this.Controls.Add(this.label7);  
this.Controls.Add(this.label6);  
this.Controls.Add(this.RollTB);  
this.Controls.Add(this.OffButton);
```

```
this.Controls.Add(this.OnButton);

this.KeyPreview = true;

this.Name = "Form1";

this.Text = "Form1";

this.Load += new System.EventHandler(this.Form1_Load);

((System.ComponentModel.ISupportInitialize)(this.ThrottleValue)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.RollTrim)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.PitchTrim)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.YawTrim)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.ValueChanger)).EndInit();

    ((System.ComponentModel.ISupportInitialize)(this.height)).EndInit();

    ((System.ComponentModel.ISupportInitialize)(this.Kp)).EndInit();

    ((System.ComponentModel.ISupportInitialize)(this.Ki)).EndInit();

    ((System.ComponentModel.ISupportInitialize)(this.Kd)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.ObsUpDown)).EndInit();

this.ResumeLayout(false);

this.PerformLayout();
```

```
}
```

```
#endregion
```

```
private System.Windows.Forms.Button OnButton;
```

```
private System.IO.Ports.SerialPort serialPort1;
```

```
private System.Windows.Forms.Button OffButton;
```

```
private System.Windows.Forms.TextBox RollTB;
```

```
private System.Windows.Forms.Label label6;
```

```
private System.Windows.Forms.Label label7;
```

```
private System.Windows.Forms.TextBox PitchTB;
```

```
private System.Windows.Forms.NumericUpDown ThrottleValue;
```

```
private System.Windows.Forms.Label label8;
```

```
private System.Windows.Forms.Label label11;
```

```
private System.Windows.Forms.TextBox YawTB;
```

```
private System.Windows.Forms.TextBox textBox1;
```

```
private System.Windows.Forms.TextBox textBox2;
```

```
private System.Windows.Forms.NumericUpDown RollTrim;
```

```
private System.Windows.Forms.NumericUpDown PitchTrim;
```

```
private System.Windows.Forms.NumericUpDown YawTrim;
```

```
private System.Windows.Forms.NumericUpDown ValueChanger;
```

```
private System.Windows.Forms.Label label2;
```

```

private System.Windows.Forms.Label label3;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.Label label9;
private System.Windows.Forms.Label label10;
private System.Windows.Forms.NumericUpDown height;
private System.Windows.Forms.Label label11;
private System.Windows.Forms.NumericUpDown Kp;
private System.Windows.Forms.Label label12;
private System.Windows.Forms.NumericUpDown Ki;
private System.Windows.Forms.NumericUpDown Kd;
private System.Windows.Forms.Label label13;
private System.Windows.Forms.DomainUpDown ModeUpDown;
private System.Windows.Forms.Label label14;
private System.Windows.Forms.NumericUpDown ObsUpDown;
private System.Windows.Forms.ListBox listBox1;
private System.Windows.Forms.Button button1;
}
}

//this.textBox1.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(CaptureKeyPress);

```



```

//this.textbox1.KeyDown           +=           new
System.Windows.Forms.KeyEventHandler(CaptureKeyDown);

//this.textbox1.KeyUp             +=           new
System.Windows.Forms.KeyEventHandler(CaptureKeyUp);

```

### Quad GUI (Form 1.cs)

```

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace Quadcopter_controller
{
    public partial class Form1 : Form
    {
        string X;

        string Y;

        string Pgain, Igain, Dgain, HeightString;

        string Throttle, Roll, Pitch, Yaw, Mode;
    }
}

```

```

string diff, obs;

public Form1()
{
    InitializeComponent();
    serialPort1.PortName = "COM3";
    serialPort1.BaudRate = 9600;
    serialPort1.Open();
}

protected override bool ProcessCmdKey(ref Message msg, Keys
keyData)
{
    if (keyData == (Keys.PageUp))
    {

        ThrottleValue.Value+=10;

        return true;
    }

    if (keyData == (Keys.PageDown))
    {

        if (ThrottleValue.Value < 1020)
        {

            ThrottleValue.Value = 1020;

```

```

    }
    else
    {

        ThrottleValue.Value-=10;
    }
    return true;
}
if (keyData == (Keys.End))
{
    ThrottleValue.Value = 1000;
    return true;
}
return base.ProcessCmdKey(ref msg, keyData);
}
private void Form1_Load(object sender, EventArgs e)
{
    this.KeyUp += new KeyEventHandler(Form1_KeyUp);
    this.KeyDown += new KeyEventHandler(Form1_KeyDown);

    Throttle = Convert.ToString(ThrottleValue.Value);
    Yaw = Convert.ToString(YawTrim.Value);
    Roll = Convert.ToString(RollTrim.Value);
}

```

```

Pitch = Convert.ToString(PitchTrim.Value);

Pgain = Convert.ToString(Kp.Value);

Igain = Convert.ToString(Ki.Value);

Dgain = Convert.ToString(Kd.Value);

HeightString = Convert.ToString(height.Value);

Mode = ModeUpDown.Text;

obs = Convert.ToString(ObsUpDown.Value);

diff = Convert.ToString(ValueChanger.Value);

serialPort1.WriteLine("1000 1500 1500 1000 " + Pgain + " " + Igain +
" " + Dgain + " " + HeightString + " " + diff + " " + obs + " "+ Mode);
}

private void DisplayText(object sender, EventArgs e)
{
    textBox1.AppendText(X + "\n");
}

private void Form1_FormClosing(object sender, FormClosingEventArgs
e)
{
    serialPort1.WriteLine("1000 1500 1500 1000 " + Pgain + " " + Igain +
" " + Dgain + " " + HeightString + " " + diff + " " + obs + " " + Mode);
    serialPort1.Close();
}

```

```
private void spinBtn_KeyDown(object sender, KeyEventArgs e)
{

}

private void Form1_KeyUp(object sender, KeyEventArgs e)
{
    //switch (e.KeyCode)
    //{
    //    case Keys.A:
    //        YawTB.Text = Convert.ToString(YawTrim.Value);
    //        break;
    //    case Keys.S:
    //        YawTB.Text = Convert.ToString(YawTrim.Value);
    //        break;
    //    case Keys.Left:
    //        RollTB.Text = Convert.ToString(RollTrim.Value);
    //        break;
    //    case Keys.Right:
    //        RollTB.Text = Convert.ToString(RollTrim.Value);
    //        break;
    //    case Keys.Up:
    //        PitchTB.Text = Convert.ToString(PitchTrim.Value);
    //        break;
```

```

// case Keys.Down:
//     PitchTB.Text = Convert.ToString(PitchTrim.Value);
//     break;
// //...
//}
}

private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    //switch (e.KeyCode)
    //{
    // case Keys.A:
    //     YawTB.Text = "1900";
    //     break;
    // case Keys.S:
    //     YawTB.Text = "1000";
    //     //ThrottleValue.Value = 1000;
    //     break;
    // case Keys.Left:
    //         RollTB.Text = Convert.ToString(RollTrim.Value -
ValueChanger.Value);
    //     break;
    // case Keys.Right:

```

```

        //          RollTB.Text = Convert.ToString(RollTrim.Value +
ValueChanger.Value);

        //    break;

        //  case Keys.Up:

        //          PitchTB.Text = Convert.ToString(PitchTrim.Value +
ValueChanger.Value);

        //    break;

        //  case Keys.Down:

        //          PitchTB.Text = Convert.ToString(PitchTrim.Value -
ValueChanger.Value);

        //    break;

        //  //...

        //}
    }

private void RollTB_TextChanged(object sender, EventArgs e)
{
    SendValues();
}

public void SendValues()
{
    System.Threading.Thread.Sleep(100);

    Throttle = Convert.ToString(ThrottleValue.Value);

    Yaw = Convert.ToString(YawTrim.Value);
}

```

```

Roll = Convert.ToString(RollTrim.Value);

Pitch = Convert.ToString(PitchTrim.Value);

serialPort1.WriteLine(Throttle + " " + Roll + " " + Pitch + " " + Yaw
    + " " + Pgain + " " + Igain + " " + Dgain + " " + HeightString + "
" + diff + " " + obs + " " + Mode);

    }

    private void serialPort1_DataReceived_1(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
    {
        X = serialPort1.ReadLine();

        this.Invoke(new EventHandler(DisplayText));
    }

    private void YawTB_TextChanged(object sender, EventArgs e)
    {
        SendValues();
    }

    private void PitchTB_TextChanged_1(object sender, EventArgs e)
    {
        SendValues();
    }

```



```
private void RollTB_TextChanged_1(object sender, EventArgs e)
{
    SendValues();
}
```

```
private void ThrottleValue_ValueChanged(object sender, EventArgs e)
{
    SendValues();
}
```

```
private void RollTrim_ValueChanged(object sender, EventArgs e)
{
    //RollTB.Text = Convert.ToString(RollTrim.Value);
    SendValues();
}
```

```
private void PitchTrim_ValueChanged(object sender, EventArgs e)
{
    //PitchTB.Text = Convert.ToString(PitchTrim.Value);
    SendValues();
}
```

```
private void ValueChanger_ValueChanged(object sender, EventArgs e)
```

```
{  
    diff = Convert.ToString(ValueChanger.Value);  
    SendValues();  
}  
  
private void Kp_ValueChanged(object sender, EventArgs e)  
{  
    Pgain = Convert.ToString(Kp.Value * 100);  
    SendValues();  
}  
  
private void Ki_ValueChanged(object sender, EventArgs e)  
{  
    Igain = Convert.ToString(Ki.Value * 100);  
    SendValues();  
}  
  
private void Kd_ValueChanged_1(object sender, EventArgs e)  
{  
    Dgain = Convert.ToString(Kd.Value * 100);  
    SendValues();  
}
```

```
private void height_ValueChanged(object sender, EventArgs e)
{
    HeightString = Convert.ToString(height.Value);
    SendValues();
}
```

```
private void YawTrim_ValueChanged(object sender, EventArgs e)
{
    SendValues();
}
```

```
private void OnButton_Click(object sender, EventArgs e)
{
    serialPort1.WriteLine("1000 1500 1500 1900 " + Pgain + " " + Igain +
" " + Dgain + " " + HeightString + " " + diff + " " + obs + " " + Mode);
}
```

```
private void OffButton_Click(object sender, EventArgs e)
{
    serialPort1.WriteLine("1000 1500 1500 1000 " + Pgain + " " + Igain +
" " + Dgain + " " + HeightString + " " + diff + " " + obs + " " + Mode);
}
```

```

private void ModeUpDown_SelectedItemChanged(object sender,
EventArgs e)
{
    Mode = ModeUpDown.Text;
    SendValues();
}

private void ObsUpDown_ValueChanged(object sender, EventArgs e)
{
    obs = Convert.ToString(ObsUpDown.Value);
    SendValues();
}

private void listBox1_SelectedIndexChanged(object sender, EventArgs
e)
{
}

private void button1_Click(object sender, EventArgs e)
{
    serialPort1.Close();
}

```

```
    }  
}
```

### **C. Image Processing GUI**

```
#include <opencv\cv.h>  
  
#include <opencv\highgui.h>  
  
#include <stdlib.h>  
  
  
using namespace std;  
using namespace cv;  
  
int main(){  
  
    ///Create matrix to store image  
  
    ///Mat image;  
  
    ///initialize capture  
  
  
    //VideoCapture cap("http://192.168.8.1/videostream.cgi/vid.mjpg");
```

```

//cap.open(0);

//if(!cap.isOpened()){

//    cout <<"cannot open camera"<<endl;

//}

///image = imread( imageName, 1 );

///cap>>image;

///create window to show image

///namedWindow("window", 1);

//while(true){

//    Mat cameraFrame;

//    Mat cannyImage;

//    //copy webcam stream to image

//

//    cap.read(cameraFrame);

//    //imshow("Source Input", cameraFrame);

//    cvtColor(cameraFrame, cannyImage, CV_BGR2GRAY);

//    Canny(cannyImage,cannyImage, 100bhhb, 200, 3);

//    imshow("test", cameraFrame);

//

//    //vector<Vec4i> lines;

//    //HoughLinesP(cannyImage, lines, 1, CV_PI/180, 50, 10, 10 );

//    //print image to screen

```

```

//    //imshow("window",image);

//    //delay 33ms
//    //waitKey((33));
//    if (waitKey(30) >= 0)
//        break;

//    }

Mat frame, frame_gray, edges, Dedges;

        //string throttle[100], roll[100], pitch[100], yaw;
        char roll[] = {0, 0, 0 , 0};

        VideoCapture
stream("http://192.168.8.1/videostream.cgi/vid.mjpg");

        if(!stream.isOpened())
        {
                cout<<"ERROR: cannot open camera";
//                return -1;
        }

        while(true)
        {

                //_serialPort->Open();

                //form->Show();

```

```

stream >> frame;

cvtColor(frame, frame_gray, CV_RGB2GRAY);

//src = imread ("D:\COLLEGE\thesis 2
(NEW)\windows application form\test\Debug", 1);

//cvtColor(src, src_gray, CV_RGB2GRAY);

Canny(frame_gray, edges, 15, 200, 3);

vector<Vec4i> hline;

HoughLinesP(edges, hline, 1, CV_PI/180, 100, 50, 10);

cvtColor(edges, Dedges, CV_GRAY2RGB);

//_serialPort->WriteLine("b");

cout<<"Straight"<<endl;

/*_serialPort->WriteLine(ThrottleTB->Text + " " +
RollTB->Text + " " + PitchTB->Text + " " + YawTB->Text);

_serialPort->Close();*/

for(size_t i = 0; i < hline.size(); i++)
{

    //_serialPort->Open();

```



```

        Vec4i D = hline[i];

        line(Dedges,          cvPoint(D[0],D[1]),
cvPoint(D[2],D[3]), Scalar(0,0,255), 3, CV_AA);

        //cout<<D[0]<<"    "<<D[1]<<"\t"<<D[2]<<"
"<<D[3]<<endl;

        if((D[0] < 300)&&(D[2] < 300))
        {
            cout<<"right"<<endl;
            //_serialPort->WriteLine(ThrottleTB-
>Text + " " + "1650" + " " + PitchTB->Text + " "+ YawTB->Text);
        }
        else
        {
            cout<<"left"<<endl;
            //_serialPort->WriteLine(ThrottleTB-
>Text + " " + "1450" + " " + PitchTB->Text + " "+ YawTB->Text);
        }
        //_serialPort->WriteLine("a");
        //
        form->
        //_serialPort->Close();
    }

```

```

//rectangle(Dedges, Point(150, 100), Point(500, 400),
Scalar(0,255,0), 0);

imshow("Camera View", Dedges);

if(waitKey(30) >= 0)
{
    //return 0;

    //imclose();

    break;
}

}

return 0 ;
}

```

#### **D. Arduino code**

```

/*
RC PulseIn Serial Read out
By: Nick Poole
SparkFun Electronics
Date: 5

```

License: CC-BY SA 3.0 - Creative commons share-alike 3.0

use this code however you'd like, just keep this license and attribute. Let me know if you make hugely, awesome, great changes.

```

*/

#include <Servo.h>

#include <NewPing.h>

#include <PID_v1.h>

#define SONAR_NUM 4 // Number of sensors.

#define MAX_DISTANCE 200 // Maximum distance (in cm) to ping.

#define PING_INTERVAL 29 // Milliseconds between sensor pings (29ms is about
the min to avoid cross-sensor echo).

unsigned long pingTimer[SONAR_NUM]; // Holds the times when the next ping
should happen for each sensor.

unsigned int cm[SONAR_NUM]; // Where the ping distances are stored.

uint8_t currentSensor = 0; // Keeps track of which sensor is active.

NewPing sonar[SONAR_NUM] = { // Sensor object array.
    NewPing(7, A3, MAX_DISTANCE), // Each sensor's trigger pin, echo pin, and max
distance to ping.
    NewPing(6, A1, MAX_DISTANCE),
    NewPing(5, A2, MAX_DISTANCE),
    NewPing(4, A4, MAX_DISTANCE),
    // NewPing(23, 24, MAX_DISTANCE),
    // NewPing(25, 26, MAX_DISTANCE),

```

```

// NewPing(27, 28, MAX_DISTANCE),
// NewPing(29, 30, MAX_DISTANCE),
// NewPing(31, 32, MAX_DISTANCE),
// NewPing(34, 33, MAX_DISTANCE),
// NewPing(35, 36, MAX_DISTANCE),
// NewPing(37, 38, MAX_DISTANCE),
// NewPing(39, 40, MAX_DISTANCE),
// NewPing(50, 51, MAX_DISTANCE),
// NewPing(52, 53, MAX_DISTANCE)
};

int PIDthrottleValue = 0, throttleValue = 1100, rollValue = 1500, pitchValue = 1500,
yawValue = 1500, currentRestoring = 0, a=0, flag = 0,
obstacleDistance = 100, pitch = 1500, roll = 1500, diff;

String  throttleString,  rollString,  pitchString,  yawString,  incomingString,
consKpString, consKiString,
consKdString, heightString, mode="MANUAL", modeString, diffString, obsString;

Servo Throttle;

Servo Roll;

Servo Pitch;

Servo Yaw;

//Define Variables we'll be connecting to

```

```

double Setpoint_alt, Input_alt, Output_alt, OutputLimit;

double Input_right, Input_left, Input_back;

//Define the aggressive and conservative Tuning Parameters

double consKp=0, consKi=0, consKd=0;

//Specify the links and initial tuning parameters

PID myPID_alt(&Input_alt, &Output_alt, &Setpoint_alt, consKp, consKi, consKd,
DIRECT);

long previousMillis = 0;    // will store last time LED was updated

// the follow variables is a long because the time, measured in milliseconds,
// will quickly become a bigger number than can be stored in an int.

long interval = 100;      // interval at which to blink (milliseconds)

void setup() {
  Serial.begin(9600);

  pingTimer[0] = millis() + 75;    // First ping starts at 75ms, gives time for the
  Arduino to chill before starting.

  for (uint8_t i = 1; i < SONAR_NUM; i++) // Set the starting time for each sensor.
    pingTimer[i] = pingTimer[i - 1] + PING_INTERVAL;

  Yaw.attach(8);

  Throttle.attach(9);

```

```

Pitch.attach(10);

Roll.attach(11);

//Serial.begin(9600); // Pour a bowl of Serial

//myPID_alt.SetMode(mode);

myPID_alt.SetOutputLimits(0, 100);

}

void loop() {

  unsigned long currentMillis = millis();

  sonarCheck();

  if(currentMillis - previousMillis > interval) {

    // save the last time you blinked the LED

    previousMillis = currentMillis;

    //

    //flag = throttleValue;

    if(mode=="AUTOMATIC"){

      //Input_alt = cm[3];

      //Setpoint_alt = ; //user input

      myPID_alt.SetMode(AUTOMATIC);

      double gap_alt = abs(Setpoint_alt-Input_alt); //distance away from setpoint

      myPID_alt.SetTunings(consKp, consKi, consKd);

      myPID_alt.Compute();

      PIDthrottleValue = throttleValue + Output_alt;

```

```

if(Input_right<obstacleDistance||Input_left<obstacleDistance||Input_back<obstacleDi
stance){
    if(Input_right<obstacleDistance && Input_left>obstacleDistance){
        if(Input_back<obstacleDistance) pitchValue = pitch + diff;
        else pitchValue = pitch;
        rollValue = roll - diff;
    }
    else if(Input_left<obstacleDistance && Input_right>obstacleDistance){
        if(Input_back<obstacleDistance) pitchValue = pitch + diff;
        else pitchValue = pitch;
        rollValue = roll + diff;
    }
    else if(Input_left<obstacleDistance && Input_right<obstacleDistance){
        if(Input_back<obstacleDistance) pitchValue = pitch + diff;
        else pitchValue = pitch - diff;
        rollValue = roll;
    }
    else if(Input_back<obstacleDistance){
        rollValue = roll;
        pitchValue = pitch + diff;
    }
}
}

```

```

else
if((Input_right>obstacleDistance)&&(Input_left>obstacleDistance)&&(Input_back>o
bstacleDistance)){
    pitchValue = pitch;
    rollValue = pitch;
}
Serial.print("Target height: ");
Serial.print(Setpoint_alt);
Serial.print(" time: ");
Serial.print(currentMillis);
Serial.print(" current height: ");
Serial.println(Input_alt);
// Serial.print(PIDthrottleValue);
// Serial.print(" ");
// Serial.print(throttleValue);
// Serial.print(" ");
// Serial.print(rollValue);
// Serial.print(" ");
// Serial.print(pitchValue);
// Serial.print(" ");
// Serial.print(yawValue);
// Serial.print(" ");
// Serial.print(" ");

```



```
// Serial.print(consKp);
// Serial.print(" ");
// Serial.print(consKi);
// Serial.print(" ");
// Serial.print(consKd);
// Serial.print(" ");
// Serial.print(Setpoint_alt);
// Serial.print(" ");
// Serial.print(diff);
// Serial.print(" ");
// Serial.print(obstacleDistance);
// Serial.print(" ");
// Serial.print(mode);
//   Serial.print(Input_left);
//   Serial.print(" ");
//   Serial.print(Input_right);
//   Serial.print(" ");
//   Serial.println(Input_back);

Throttle.writeMicroseconds(PIDthrottleValue);

Roll.writeMicroseconds(rollValue);

Pitch.writeMicroseconds(pitchValue);

Yaw.writeMicroseconds(yawValue);

}
```

```

else if(mode=="MANUAL"){

if(Input_right<obstacleDistance||Input_left<obstacleDistance||Input_back<obstacleDistance){

    if(Input_right<obstacleDistance && Input_left>obstacleDistance){

        if(Input_back<obstacleDistance) pitchValue = pitch + diff;

        else pitchValue = pitch;

        rollValue = roll - diff;

    }

    else if(Input_left<obstacleDistance && Input_right>obstacleDistance){

        if(Input_back<obstacleDistance) pitchValue = pitch + diff;

        else pitchValue = pitch;

        rollValue = roll + diff;

    }

    else if(Input_left<obstacleDistance && Input_right<obstacleDistance){

        if(Input_back<obstacleDistance) pitchValue = pitch + diff;

        else pitchValue = pitch - diff;

        rollValue = roll;

    }

    else if(Input_back<obstacleDistance){

        rollValue = roll;

        pitchValue = pitch + diff;

    }

}

```

```

    }
    else
if((Input_right>obstacleDistance)&&(Input_left>obstacleDistance)&&(Input_back>o
bstacleDistance)){
    pitchValue = pitch;
    rollValue = roll;
}
myPID_alt.SetMode(MANUAL);
Throttle.writeMicroseconds(throttleValue);
Roll.writeMicroseconds(rollValue);
Pitch.writeMicroseconds(pitchValue);
Yaw.writeMicroseconds(yawValue);
// Serial.print(PIDthrottleValue);
// Serial.print(" ");
// Serial.print(throttleValue);
// Serial.print(" ");
// Serial.print(rollValue);
// Serial.print(" ");
// Serial.print(pitchValue);
// Serial.print(" ");
// Serial.print(yawValue);
// Serial.print(" ");
// Serial.print(" ");

```

```
// Serial.print(consKp);
// Serial.print(" ");
// Serial.print(consKi);
// Serial.print(" ");
// Serial.print(consKd);
// Serial.print(" ");
// Serial.print(Setpoint_alt);
// Serial.print(" ");
// Serial.print(diff);
// Serial.print(" ");
// Serial.print(obstacleDistance);
// Serial.print(" ");
// Serial.print(mode);
//   Serial.print(Input_left);
//   Serial.print(" ");
//   Serial.print(Input_right);
//   Serial.print(" ");
//   Serial.println(Input_back);
Serial.print("Target height: ");
Serial.print(Setpoint_alt);
Serial.print(" time: ");
Serial.print(currentMillis);
Serial.print(" current height: " );
```

```

    Serial.println(Input_alt);

}

}

}

void echoCheck() { // If ping received, set the sensor distance to array.

    if (sonar[currentSensor].check_timer())

        cm[currentSensor] = sonar[currentSensor].ping_result / US_ROUNDTRIP_CM;

}

```

void oneSensorCycle() { // Sensor ping cycle complete, do something with the results.

// The following code would be replaced with your code that does something with the ping results.

```

// for (uint8_t i = 0; i < SONAR_NUM; i++) {

//   if(cm[i]<=0)

//   {

//     cm[i] = 200;

//   }

//   Serial.print(i);

//   Serial.print("=");

//   Serial.print(cm[i]);

//   Serial.print("cm ");

```

```

// //delay(10);

// }

// Serial.println();

Input_alt = cm[3];

Input_right = cm[2];

Input_left = cm[0];

Input_back = cm[1];

if(Input_right == 0)

    Input_right = 200;

if(Input_left == 0)

    Input_left = 200;

if(Input_back == 0)

    Input_back = 200;

}

void serialEvent(){

    if(Serial.available()){

        char ch = Serial.read();

        if(ch != '\n')

            incomingString += ch;

        else{

            //Serial.println(incomingString);

            currentRestoring = 1;

```

```
while(currentRestoring==1){
    if(incomingString[a]==' '){
        a++;
        currentRestoring=2;
    }
    else{
        throttleString+=incomingString[a];
        a++;
    }
}

while(currentRestoring==2){
    if(incomingString[a]==' '){
        a++;
        currentRestoring = 3;
    }
    else{
        rollString+=incomingString[a];
        a++;
    }
}

while(currentRestoring==3){
    if(incomingString[a]==' '){
        a++;
```

```

    currentRestoring = 4;
}
else{
    pitchString+=incomingString[a];
    a++;
}
}
while(currentRestoring==4){
    if(incomingString[a]==' '){
        a++;
        currentRestoring = 5;
    }
    else{
        yawString+=incomingString[a];
        a++;
    }
}
while(currentRestoring==5){
    if(incomingString[a]==' '){
        a++;
        currentRestoring = 6;
    }
    else{

```



```
    consKpString+=incomingString[a];
    a++;
}
}
while(currentRestoring==6){
    if(incomingString[a]==' '){
        a++;
        currentRestoring = 7;
    }
    else{
        consKiString+=incomingString[a];
        a++;
    }
}
while(currentRestoring==7){
    if(incomingString[a]==' '){
        a++;
        currentRestoring = 8;
    }
    else{
        consKdString+=incomingString[a];
        a++;
    }
}
```

```

}
while(currentRestoring==8){
    if(incomingString[a]==' '){
        a++;
        currentRestoring = 9;
    }
    else{
        heightString+=incomingString[a];
        a++;
    }
}
while(currentRestoring==9){
    if(incomingString[a]==' '){
        a++;
        currentRestoring = 10;
    }
    else{
        diffString+=incomingString[a];
        a++;
    }
}
while(currentRestoring==10){
    if(incomingString[a]==' '){

```

```
    a++;

    currentRestoring = 11;
}

else{

    obsString+=incomingString[a];

    a++;

}

}

while(currentRestoring==11){

    if(incomingString[a]==NULL){

        a++;

        currentRestoring = 0;

    }

    else{

        modeString+=incomingString[a];

        a++;

    }

}

throttleValue = throttleString.toInt();

rollValue = rollString.toInt();

pitchValue = pitchString.toInt();

yawValue = yawString.toInt();

pitch = pitchValue;
```

```
roll = rollValue;

consKp = consKpString.toInt();

consKp = consKp*0.01;

consKi = consKiString.toInt();

consKi = consKi*0.01;

consKd = consKdString.toInt();

consKd = consKd*0.01;

Setpoint_alt = heightString.toInt();

mode = modeString;

diff = diffString.toInt();

obstacleDistance = obsString.toInt();

throttleString = "";

rollString = "";

pitchString = "";

yawString = "";

incomingString = "";

consKpString="";

consKiString="";

consKdString="";

heightString="";

modeString="";

diffString="";

obsString="";
```

```

    a = 0;
  }
}
}

void sonarCheck(){

  for (uint8_t i = 0; i < SONAR_NUM; i++) { // Loop through all the sensors.

    if (millis() >= pingTimer[i]) {      // Is it this sensor's time to ping?

      pingTimer[i] += PING_INTERVAL * SONAR_NUM; // Set next time this
sensor will be pinged.

      if (i == 0 && currentSensor == SONAR_NUM - 1) oneSensorCycle(); // Sensor
ping cycle complete, do something with the results.

      sonar[currentSensor].timer_stop();      // Make sure previous timer is canceled
before starting a new ping (insurance).

      currentSensor = i;                      // Sensor being accessed.

      cm[currentSensor] = 0;                 // Make distance zero in case there's no ping
echo for this sensor.

      sonar[currentSensor].ping_timer(echoCheck); // Do the ping (processing
continues, interrupt will call echoCheck to look for echo).

    }
  }
}

//

//   Setpoint_alt = 20 + (flag-1560);

```

```

// Setpoint_right = 100;
// Setpoint_left = 100;
// Setpoint_back = 100;
// double gap_alt = abs(Setpoint_alt-Input_alt); //distance away from setpoint
// if(gap_alt<10){ //we're close to setpoint, use conservative tuning parameters
//   myPID_alt.SetTunings(consKp, consKi, consKd);
// }
// else{
//   //we're far from setpoint, use aggressive tuning parameters
//   myPID_alt.SetTunings(aggKp, aggKi, aggKd);
// }
// myPID_alt.Compute();
// //throttleValue = 1560+Output_alt;
// Serial.print(Input_left);
// Serial.print(" ");
// Serial.print(Input_right);
// Serial.print(" ");
// Serial.println(Input_back);

//Serial.println(throttleValue);
// Serial.println(Output_alt);
//Serial.println(gap_alt);
// Serial.print(" ");

```

```
// Serial.print(throttleValue);  
// Serial.print(" ");  
// Serial.print(rollValue);  
// Serial.print(" ");  
// Serial.print(pitchValue);  
// Serial.print(" ");  
// Serial.print(yawValue);  
// Serial.print(" ");  
// Serial.print(aggKp);  
// Serial.print(" ");  
// Serial.print(aggKi);  
// Serial.print(" ");  
// Serial.print(aggKd);  
// Serial.print(" ");  
// Serial.print(consKp);  
// Serial.print(" ");  
// Serial.print(consKi);  
// Serial.print(" ");  
// Serial.print(consKd);  
// Serial.print(" ");  
// Serial.println(Setpoint_alt);  
//throttleValue = flag;
```

