Department of Electrical and Computer Engineering
Schulich School of Engineering
University of Calgary

Fall 2012

4th Year Project Report #1
Preliminary Design Document

Group # 30

Group Name:
Team Lunch

Autonomous UAV Terrain Avoidance System

Group Members:
Rami Abou Ghanem,
Adam Dickin,
Jennifer Patterson,
James Thorne

Sponsor' Name:
CDL Systems

November 9th, 2012

Table of Contents

Introduction

Over the past few years, there has been a huge increase in drones being used for military and police operations. Many of these drones do not have any system in place for collision detection and avoidance besides notifying the operator so that they can change course. This system works well when doing missions in a large open environment, but is insufficient for small, highly maneuverable drones in enclosed or cluttered spaces.

Operator reaction times are too slow to successfully avoid collision, especially when considering the latency introduced by a wireless control harness. Additionally, many of the commercial systems available do not focus on being affordable, but rather on including every feature possible. This results in a drone that is equipped for many tasks, but costs $50,000 or more. This pricing makes drone adoption difficult for casual users, and leaves little room for repair or replacement of drones involved in collisions.

There has been some improvement in this area over the past few years. In fact, some UAVs are now available to the general consumer for around $400. These drones also do not have any kind of collision avoidance, and limited control systems, and are therefore as easy to crash as their high-end counterparts. Some improvement must be made in this area for both the consumer market and the commercial market - ideally, both users would be able to use their drones safely and effectively knowing that systems are in place to avoid crashes.

Our goal as Team Lunch is to solve these problems, and to prove that every autonomous drone can be equipped with our solution. Our project will benefit our customer, CDL Systems Ltd., by demonstrating how such a system can be implemented cheaply with off-the-shelf technology available to the general consumer. Our project will also benefit the rest of the UAV market, because we will prove that such features are possible at a low cost. We hope that with time, they will become standard on every UAV. Such an adoption will greatly increase the safety of drones, not only by reducing the risk of a costly crash, but also by reducing the risk of injury to persons and damage to property.

We have decided to focus on taking a commercially available, off the shelf consumer UAV, and equipping it with an autonomous collision detection and avoidance system. This system will allow us to prove that automated accident avoidance is possible during operation of a drone, with minimal operator skill required. Our drone, once outfitted with these systems, will allow unskilled operators to purchase and fly the UAV in close-quarter environments with minimal risk, training, and possibility of damage to the UAV and surrounding property. Hopefully, the outcome of our project will convince drone manufacturers to adopt the use of a collision avoidance and detection system in production model aircraft. We also hope that with the increased availability and lower cost of these autonomous drones, many more organizations will decide to use drones in their everyday operations.

Background and Previous Works

Small UAVs are difficult to fly, easily damaged by collision, and expensive to repair. This presents a strong financial risk to potential buyers, and discourages widespread adoption. There are a few early adopters in the small UAV business, including the Merseyside police in the United Kingdom. They purchased a £13,000 drone, and crashed it during a routine training exercise in February 2010[1]. Initially the police had found several potential benefits of owning a surveillance UAV, but after some use and the resulting crash, they discovered there were also severe drawbacks.

There are several other police agencies that have started to see the benefits of having a small drone for investigations, including the RCMP and the Halton Regional Police [2,3]. These drones are ideally supposed to be easy to use, but are still prone to collision if an inexperienced operator makes a mistake. Safely controlling a small, highly maneuverable aircraft in a cluttered urban environment is difficult, and has slowed commercial adoption of such systems.

In previous years, several other teams have completed the 4th Year Project with an unmanned vehicle-related project. These projects have mainly focused on basic flight hardware - motor

controllers, battery-charging systems, and basic video feeds. Our project builds on this existing work by purchasing a fully functional vehicle outright, and then adding additional high-level intelligence and interfacing capabilities. We will not be considering the vehicle's basic flight hardware, or payload capabilities such as streaming video, as previous teams have. Our project is focused on adding collision avoidance, and improving the human-computer interaction between the vehicle and its operator.

Several models of quadcopters are commercially available, ranging in price from $400 to $100,000. Commonly utilized commercial aircraft include the Draganflyer and the Aeyron Scout, both of which are priced in the $20,000 - $50,000 range. The MiKroKopter, priced at approximately $2,000, is a 'do it yourself' aircraft that requires some assembly. For this project, we have selected the Parrot AR.Drone 2.0. It is inexpensive, readily available, and we believe it will provide a strong platform for our project. Although it has limited flight time compared to the Draganflyer and Aeyron Scout, its small size and low cost make it ideal for our project.

Product Requirements

The product that we have chosen to design is a lightweight autonomous quadcopter that will avoid collisions with its environment. In order to create such a product we will need a few pieces of hardware, which are easily obtained from local vendors. We will first need a pre built quadcopter which flies out of the box so that we do not need to design the hardware components of the system, aside from the sensor control board which will feed information to the copter's onboard computer. Secondly, we will need an embedded sensor control board along with 5 ultrasonic sensors to provide distance information so that we can determine how far we are from objects while flying. Third, we will need an Android tablet in order to run our control station application, which will send commands to the quadcopter to tell it where to fly. The sequence of events that we expect to occur is:

1. The user will turn on the quadcopter
2. The user will turn on the Android tablet application and connect it to the quadcopter

3. The user will then issue the start command to the quadcopter, at which point the quadcopter will start flying
4. The user will issue commands to the quadcopter to go up, down, left, right, forward, or backward
5. Then if the user issues a command that the collision avoidance software determines to be detrimental to the health of the quadcopter, it will ignore it or change it.
6. If the collision avoidance software at any time while flying determines that it is at risk of a collision, it will change the course of the quadcopter in order to avoid collision

In order to help explain how our system works collectively, we have created a diagram that shows the interaction between all of the components of our system.

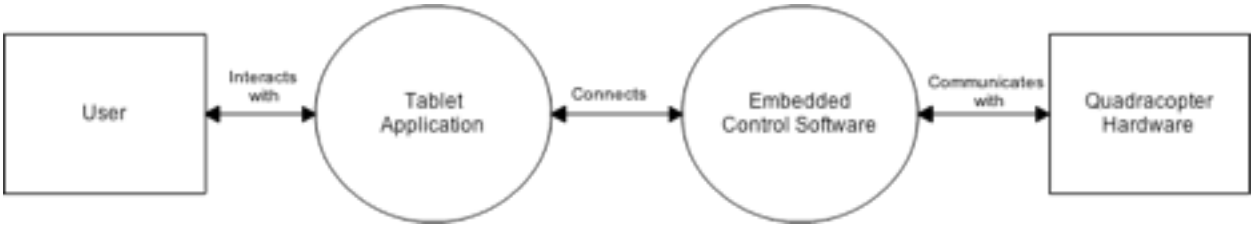

Figure 1: System Interaction Diagram

The system we have proposed is built off of a few existing systems in order for us to avoid the work that would be needed in order to have a flying quadcopter. We are building our collision avoidance software on top of a Parrot AR.Drone 2.0 quadcopter that already has its own control software, which controls all of the hardware on the drone except for the sensors that we are adding. The quadcopter has built-in Linux software, which runs the autopilot software that handles commands for the quadcopter to fly in different directions. The sensors that we have selected for our project are from a local vendor in Calgary called Phidgets. These sensors are ultrasonic sensors that provide us distance readings in between the range of 154 mm to 6.5 m [4]. Lastly, we will be using an Android tablet with an open source tablet application that we will modify to fit our needs. The tablet and application we will modify have yet to be determined since they are not needed for the first and primary phase of our project.

Functional Requirements

Our system has many requirements both functional and nonfunctional. All of the currently known functional requirements are listed below:

- The quadcopter should be able to avoid objects, while flying at full (20%) tilt, that are up to 2 meters away.
- Upon detecting an object, the quadcopter system will inform the operator of the potential collision with one (1) second, and autonomously reduce speed to prevent collision.
- The quadcopter will autonomously reject fly-forward, -reverse, -left, and -right commands that will result in collision, within one (1) second of detecting an obstacle.
- The quadcopter must detect objects that are transparent and reflective, such as glass walls or mirrors, at the full two-meter range.
- The quadcopter, upon losing connection to the control station, should safely land within ten (10) seconds, while not colliding with any obstacles.
- The quadcopter should have sufficient bandwidth in order to send a 720p video feed and sensor data back to the ground control station.
- The Android application must be able to send and receive all core flight commands identified in the Parrot AR.Drone API.

Nonfunctional Requirements

All of the known nonfunctional requirements are listed below:

- The system must be easy to use by an unskilled operator.
- The basic autonomous collision avoidance system must be compatible with any ground station that utilizes the Parrot AR.Drone API, including but not limited to the Android-based ground station we will develop.
- The reliability of the off-the-shelf quadcopter must not be reduced by the addition of our autonomous collision avoidance system.
- Our software must be built from self-documenting source code, with minimal extraneous code comments.
- All source code must follow the CDL Coding Standards, as remembered and defined by James Thorne and Adam Dickin, former CDL interns.

- The selected quadcopter, along with the sensor control harness, must not exceed our identified project budget of $5,000.
- Intellectual property rights must be assigned to CDL Systems Ltd., with no open sourcing of the code.
- Code must be stored on GitHub, a Git-based cloud-hosting provider.
- Automated acceptance tests must cover all core functionality, except for interface classes.
- Automated tests must run on a Linux-based Virtual Machine, while the compiled autopilot must run on the ARM-based quadcopter.
- The autonomous collision avoidance system must not be tightly coupled with the architecture of the Parrot AR.Drone - it must be possible to port it to other, similar vehicle platforms.
- Vehicle flight time with the collision avoidance system active must be at least five (5) minutes.

Potential Requirement Conflicts

- Vehicle flight time may be reduced due to weight of the onboard sensors, required to perform autonomous collision avoidance.
- Analysis of sensor data may induce latency in the operator commands, possibly conflicting with the low-latency requirement identified above.

Objective Analysis

To define our system, we started by collecting all the high level goals that we need to achieve in order to bring our project to completeness. We started by defining goals we would like to achieve and then broke them down one more level in order to describe each goal better. These goals were then given metrics to make them measurable. After, we prioritized and ranked each goal. We have also defined our constraints and any assumptions that were made.

Criteria of Success

The following list describes our high level goals for the overall project along with their respective metrics to measure them:

- Intuitive Interface
    - Clean (Number of controls for interface)
    - Usable (Learning time in hours)
- Avoids Collisions

    Moving Obstacles (Collisions per hour of flight time)

    Stationary Obstacles (Collisions per hour of flight time)
- Easy to Operate
    - Easy to activate (Learning time in hours)
    - Easy steering (Learning time in hours)
    - Easy landing (Learning time in hours)
    - Easy to repair (Repair time and cost in dollars)
- Durable
    - Durable against regular wear and tear (Flight time until failure)
    - Durable against crashes (Crashes until failure)
- Inexpensive
    - Based on a commercially-available Quadcopter (Cost in dollars)
    - Controller (Cost in dollars)
    - Maintenance and Replacement Parts (Cost in dollars)
- Reliable
    - Software “crashes” should not occur (Failures per hour)
    - Sensor data should be relayed reliably (Failures per hour)
- Transportable
    - Quadcopter (Size and weight)
    - Controller (Size and weight)
- Accurate
    - Relayed sensor data should be accurate
        - Video Data (Quality and frames per second of video)

        - Distance Data (Distance in meters)
- Modifiable
    - Modifiable sensors / add-ons (Time to modify in hours)
    - Modifiable functionality (Time to modify in hours)

Objective Priority and Ranking

We gave each objective a ranking based on a group vote. Giving a ranking to objectives will allow us to focus on more important objectives first and give us a sense of direction with regards to the overall project. Without a ranking, the project would be much more difficult to implement since the main focus of it will be lost to us.

Table 1: Priority Ranking

Priority/Rank	**Objective**
1	Avoids Collisions
2	Easy to Operate
3	Durable
4	Reliable
5	Accurate
6	Inexpensive
7	Intuitive Interface
8	Modifiable
9	Transportable

Constraints

Our project is constrained in many ways. The following list describes our constraints:

- We are constrained with time. This project must be completed by the end of the winter 2012 term. This means that we must have a complete schedule and follow it as closely as possible.
- We are constrained with budget. Our sponsor, CDL Systems, has provided us with a budget for our project. This limits how much we can spend on the various necessary components such as the quadcopters, sensors, and Android tablets.
- We are constrained with our knowledge on this project. We must familiarize ourselves with many topics including quadcopters, embedded development, sensor usage, android development and networking between devices.

Assumptions

We made several technical assumptions at the beginning of our project. The assumptions made were:

- We assume that obstacles will be visible to the sensors that we use.
- We assume that there will be no sensor interference.
- We assume that a Linux-based environment is available on the quadcopter.
- We assume that we have high-level access to the quadcopter's control APIs.
- We assume that a compatible ARM toolchain (i.e. compiler and linker) is available to us.
- We assume that during flight, the Linux environment on the quadcopter has enough processing power and RAM to run our onboard software.

Risk Assessment

Our project has some risks that could affect our ability to complete everything that we have planned for our project. We have broken down our risks into 3 categories: Technical, Organizational, and Management. For each risk in each category, we have determined their severity and likelihood.

Technical Risks

1. Our quadcopter will not be able to lift the added weight of the sensors and the embedded control board. This risk has a very high severity since it would stop our development in its tracks; however, the likelihood of this is low due to the mitigation strategy that has already been used. Before the purchase of the quadcopter, research was done online to determine the average weight that others had measured to ensure that the total weight of all our added sensors was low enough.
2. We will not be able to run custom avoidance code on the quadcopter's onboard computer. This risk comes with a severity of high since if we could not run our own code in addition to the autopilot we would not be able to use the selected hardware and would have to make a change. We did ensure before the start of the project that we could compile for the onboard computer as well as execute a simple hello world program so the likelihood of this is low.
3. Selected sensors will not be accurate or fast enough to allow for quick collision detection, or the sensors will interfere with each other enough that readings will be false. We can always change which sensors we are using so that the severity of this is medium, due to the fact that sensors cost extra money. The likelihood of this is low to medium but we have been analyzing two different sets of sensors and are in the process of choosing the best one. In the case that the sensors do interfere with each other, we can always chain the sensors together so that only one sensor is reading at a time.
4. Development skills of each team member may not be equivalent to every other member, so varying qualities of code may be produced. The severity of this is quite low and is likely to happen but the issue can be mitigated easily. Having every team member write

in the logbook what he or she is doing on each task they are assigned allows every other team member to check their work briefly to ensure that the task is being completed properly. This also allows less experienced team members to get feedback on how their work could be better and overall improves the skill level of the team.

Organizational

1. Regular meeting times are very hard to organize due to the fact that all of us are taking various courses and have different times available throughout the week. This does not pose much of an issue to our team and has already happened a few times, but since we have plenty of meetings scheduled throughout the week different members can touch base at different times and word gets around on what is going on with the project. We also touch base every Wednesday in lab as is compulsory, so this is a good time to ensure that we know what tasks have been done and what tasks still need to be finished or planned.

Management

1. Making a concrete development schedule to follow throughout the year is tedious and is likely not to get followed or have the deadlines met. The severity is low and is highly likely to occur due to the nature of different courses picking up their workload at different times. Having an overall general plan with just an end date can mitigate this. We will only plan tasks that we know can be done within 1-2 weeks when people have ample time to work on them.
2. Task estimations will not be accurate. This is a slight issue that is likely to happen since it is hard to exactly estimate how long a task will take in the end. We do not have any previous experience working on a project like this, so to mitigate the risk of task overrun we will ensure that everyone agrees on a time amount and then ensure that we adjust the task time limit once more information is discovered on how long something will take.
3. It is difficult to do significant amounts of planning when teammates are not available at every meeting. This is not a severe risk but is somewhat likely to happen. Since all teammates have everyone's email and phone number this will allow for more lines of communication. So, if a teammate is only required for a quick bit of information on a

task they can be called or if a response is not needed initially they can be emailed. Also many of us take our breaks in the 4th year zoo homeroom so teammates catch up with each other on a regular basis throughout the day even when there is no meeting set.

Scheduling

We developed our schedule by determining our high-level activities and estimating the resources needed for them. After doing this we used industry-standard software, called OnePMO, to generate our Gantt charts. This gave us more flexibility and made our schedule easier to determine.

High-level Activities

1. Create simple UI mockups to determine best layout for control station
2. Determine best sensor to use for object detection and accuracy of sensor
3. Get test bench model running to take readings from all sensors
4. Task planning sessions
5. Write report 1
6. Write report 2
7. Write report 3
8. Write report 4
9. Solder sensors to embedded control board and attach to copter
10. Write software for collision avoidance for the copter
11. Purchase AR Drone 2.0s and several sensors of various types
12. Test the collision avoidance software with test flights of copter
13. Select and purchase Android tablet for station
14. Select existing open source control station to modify
15. Design enhanced ground control station for Android tablet
16. Implement enhanced ground control station for Android tablet

Milestones

The following milestones will be met during the course of the two terms.

1. Initial project demo - Estimate date: November 25, 2012
2. Demo ground control test bench to customer - Estimated date: December 5, 2012
3. Demo collision avoidance with quadcopter - Estimated date: April 15, 2012
4. Demo enhanced ground station on tablet - Estimated date: April 15, 2012

Table 2: Resource Estimation and Dependencies

Activity #	Resource needed	Amount of resource	Predecessor
1	Time	1 day	14
2	Time	1 day	11
3	Time	2 days	2
4	Time	2 days	
5	Time	4 days	2, 3, 4
6	Time	4 days	5
7	Time	4 days	6
8	Time	4 days	7
9	Time	1 day	3
10	Time	10 days	9
11	Time	1 day	
12	Time	10 days	10
13	Time	1 day	
14	Time	1 day	
15	Time	10 days	13

16	Time	20 days	15

Gantt Chart

The following Gantt chart gives a visual representation of our high level activities. It allows us to determine when we should complete activities by and how much room there is for slack. The Gantt chart also displays when we should reach our various milestones for the project. This type of chart makes it much easier to follow a schedule for our project.

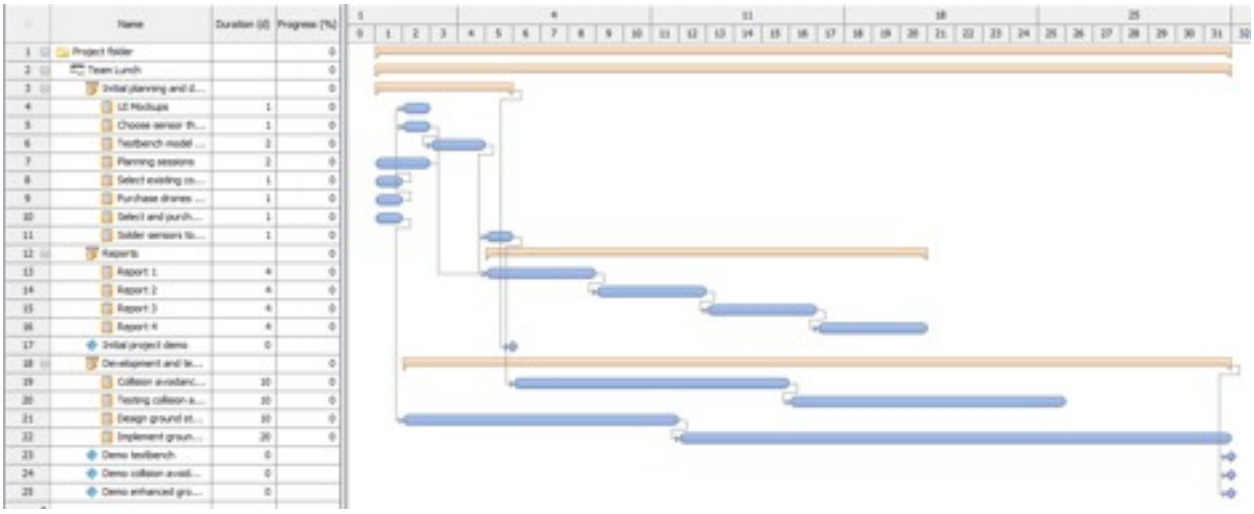

Figure 2: Gantt Chart of High Level Activities

PERT Chart

We organized our activities into a PERT chart as shown below. This allows us to have a more visual representation of our schedule, similar to the Gantt chart above. We can see which activities are dependent on each other and the duration of each one. This also allows us to determine the amount of slack time available for each activity based on our estimated project completion date.

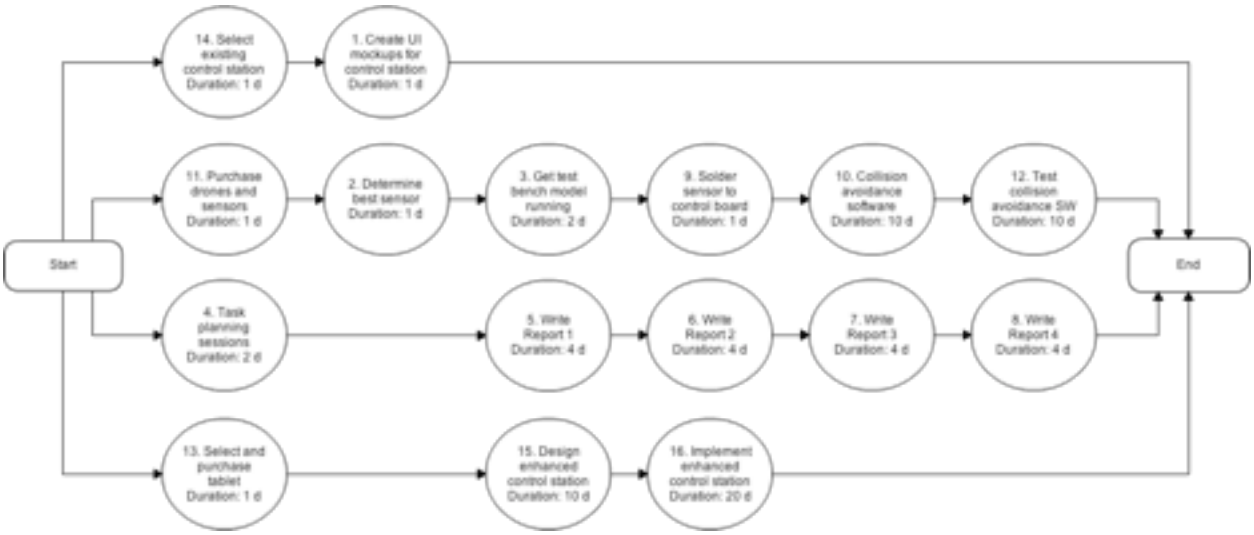

Figure 3: Pert Chart

Glossary

UAV(Unmanned Aerial Vehicle) - An aircraft without a human pilot on board. The flight of the drone is either controlled autonomously by computers onboard, or is under the remote control of an operator.

References

[1] BBC News. (2011, October). Police drone crashes into River Mersey [Online]. Available: http://www.bbc.co.uk/news/uk-england-merseyside-15520279

[2] Canadian RCMP Using Draganflyer RC Helicopter UAV to aid Traffic Investigations[Online] Available: http://www.draganfly.com/news/2012/07/17/canadian-rcmp-considering-draganflyer-rc-helicopter-uav-to-aid-traffic-invesitgations/

[3] Terry Wilson. (2012, April). Halton Regional Police Now Using Drone Aircrafts[Online]. Available: http://canadianawareness.org/2012/04/halton-regional-police-now-using-drone-aircrafts/

[4]Products for USB Sensing and Control[Online]. Available: http://www.phidgets.com/products.php?category=2&product_id=1128_0