Progress Report for Search and Rescue Quadcopter

Project Team: Lekabari Nghana
Hesam Fatahi
Avi Gill
Gurjeet Matharu
Mehrdad Ahmari

Contact Person: Gurjeet Matharu
gsm9@sfu.ca

Submitted To: Dr. Andrew Rawicz – ENSC 440W
Dr. Steve Whitmore – ENSC 305W
School of Engineering Science
Simon Fraser University

Issue Date: November 17th, 2014

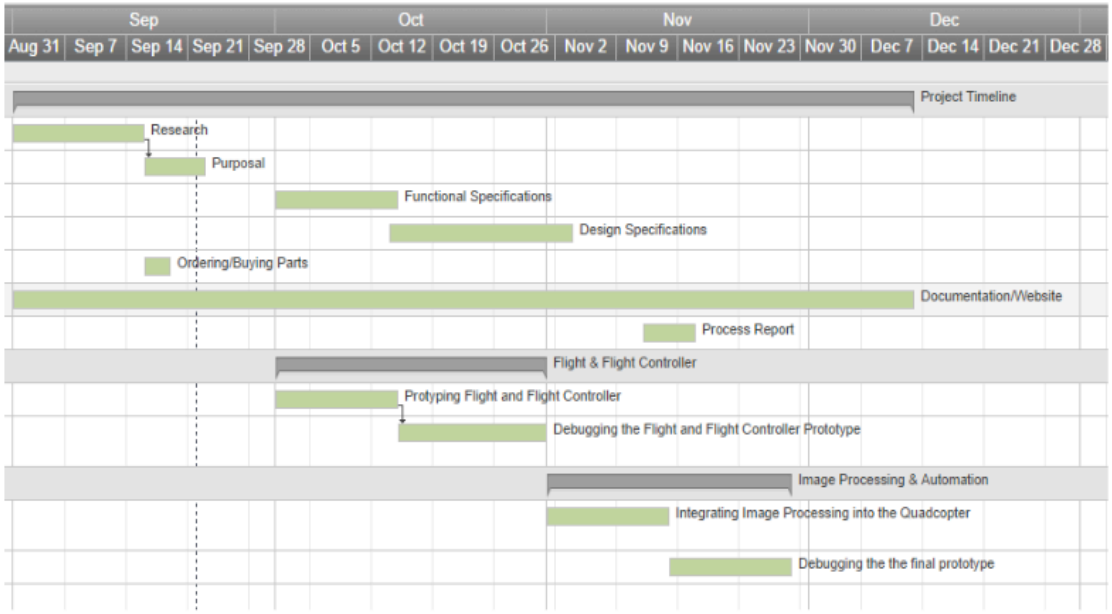
SEARCUE

Introduction

The main purpose of the Searcue team is to improve the success rate of search and rescue operations. There are areas that pose danger to humans and may lead to ineffective searches. In order to reduce human limitations, we are implementing a UAV system with a camera mounted onto it. This system will use digital image processing to identify humans or objects of interest saving search and rescue teams time. The UAV would be capable of operating in manual and autonomous flight modes allowing for versatility. The operator can specify a particular area to search in autonomous flight mode and can also switch to manual mode allowing for a more precise control. The GPS and compass on board allows the UAV to send the operator co-ordinates making the time needed for search and rescue teams to respond greatly reduced. By splitting up work and delegating responsibilities, we are confident that our team would be able to complete the proof-of-concept model by December 16th, 2014.

Schedule

Figure 1 below shows the original timeline for the project. As shown in the figure below the image processing should be integrated into the quadcopter, and final prototype testing should have started. However, the image processing is in its final debugging stages and will be ready to integrate into the quadcopter by the end of the month. This essentially makes the image-processing component of the quadcopter slightly off schedule by approximately one week.
Additionally the quadcopter itself has had numerous issues, mostly pertaining to the power supply and motor calibrations. This has set back the timeline for the quadcopter and its flight considerably. This quadcopter is now almost two weeks behind.



Gantt Table – Showcasing the Project Timeline

Figure 1: Original project schedule

Finances

Financially we are currently under our budget, since we predicted we would spend approximately $700. We applied for funding through ESSEF and received $500. This amount is not enough to cover our total project expense; however it is relatively close. Our expense breakdown is shown in Table 1.

Equipment	Cost ($)
Turnigy 9ch Transmitter and 8ch Receiver	**75**
GPS Shield Kit	**75**
Raspberry Pi and Pi Camera	**65**
4x DJI 920KV Brushless Motors	**50**
4x Electronic Speed Controllers (ESCs)	**50**
9DOF Sensor Stick	**50**
Cable Connectors, spacers and wires	**50**
Zippy 4000mAh Battery	**30**
Carbon Fiber Frame	**30**
AeroQuad Shield	**30**
Power Supply/Distribution Board	**30**
Carbon Fiber Propellers	**20**
BMP180 Sensor	**15**
Subtotal	**570**
Shipping and Duty	**60**
Total	**630**

Table 1: List of parts and costs

Progress

Hardware

The original schedule for the hardware part of our project was to get the UAV to fly by the beginning of November. The parts were ordered on September 15th 2014, but due to some shipment and compatibility issues we had to wait until the second week of October to receive all the required parts. We calibrated the 4-in-1 Electronic Speed Controller (ESC) using the throttle hub. At first, we tried testing the motors using an oscilloscope, but the current and voltage wasn't enough to run the motors. This made us realize we had to use the ESC in conjunction with the Zippy battery to run the motors. The frame was built and all the connections from the ESC to the 4 motors were made. We bound the receiver to transmitter in order to control the UAV for manual flight. The binding process didn't take too long, but we had to do some research to figure out the proper connections. After research we determined the correct channel pins on the receiver to control the yaw,

pitch, roll and throttle. At the time of our oral progress report, we had tested the motors and made sure that they worked properly. We also tested the motors with the propellers and there was a small lift. Our team also calibrated the motors to make sure we were able to control the speed of rotation of the motors using the sticks of the transmitter.

Furthermore, we went on to simulate the flight using the AeroQuad configurator software. This enabled us to adjust settings for desired flight characteristics. We downloaded the software successfully, but faced connection problems with the Aeroquad software libraries. Time was spent fixing the problem, and was achieved with help using the AeroQuad forums. After the library problem was resolved, we went through the steps of loading the software into the configurator and changing settings to suit our system. We initialized the electrically erasable programmable read only memory (EEPROM), performed transmitter, magnetometer, accelerometer and gyroscope calibration. During transmitter calibration, we noticed that the software did not detect the movement of the transmitter's sticks.

Currently, we are working on controlling the motors and receiver by connecting to the pins directly on the Aeroquad shield. We are encountering serial port problems and time-out errors while using the AeroQuad configurator software. For the rest of the semester we plan to focus more on debugging the autonomous flight, as well as modify the AeroQuad flight controller software.

Image Processing

For the image processing portion of the Searcue project we decided to use a Raspberry Pi connected to the internet, to send the video and compute the image processing. We initialized the Raspberry Pi with the Raspbian operating system, which is based off of Linux. Then using the secure shell (SSH) protocol and through the local area network (LAN) we were able to connect to the Raspberry Pi and download the necessary files such as drivers, application programming interfaces (APIs) and other updates. Once the Raspberry Pi was up to date we connected a webcam via Universal Serial Bus (USB) to capture JPEG images, and video. However, we noticed that the compatibility of the USB webcam (Microsoft LifeCam) with the Raspberry Pi did not work well as it did not support video. To fix this issue we went ahead and bought the Raspberry Pi camera module that is developed for the Pi. The benefit of this is that it connects directly to the Raspberry Pi board and comes with drivers and inbuilt support for high quality video. The video capturing issue had been solved and using the MJPEG protocol we were able to stream live video over LAN (using WIFI). The next step would be to send the video over the internet to our server, where the video can then be processed.

We considered two options for the actual image processing, Matlab and C#; however we decided to pursue C# due to its large developer support and robustness. Furthermore, there are APIs and libraries available for C# that work with our application really well. We are also able to run C# right onto the Raspberry Pi with a Linux based compiler called Mono, allowing for more options. This allows us to process the images on the

Raspberry Pi itself or on an external server. Our current progress allows us to identify objects in single frame JPEG; we now need to identify objects from MJPEG.

Remediation

To ensure that our project meets our deadline, we have prioritized the core of our project, being the image processing and alert system. The quadcopter itself requires a few extra components for stable flight. Nevertheless, the team has broken up into two groups, where the first group is responsible for the autonomous flight of the quadcopter, and the second group is responsible for the software of image processing and alert system. The group has taken up an effort to meet during weekends and utilize developer tools such as Github to allow for better collaboration on software.
The way the Searcue system has developed has allowed for the software image processing and quadcopter flight to be addressed in parallel. The image processing was initially going to be sent through our main flight controller, the Arduino mega and peripherals, however, we chose to go with a separate processing unit. This was due to the fact that this will alleviate some of the computing load on the Arduino, resulting in improved flight performance. The chosen computing unit was a Linux-based system, the Raspberry Pi. This was chosen for it easy compatibility and relatively low cost. By splitting the computing load to two microcontrollers, the Raspberry Pi and the Arduino Mega, we are able to utilize their respective developer support and aim to optimize each component.

Summary

Although we have faced some issues and delays, which caused us to be slightly behind schedule, our project is almost in the final stage of development and all current issues with implementation are being actively resolved. We have been able to avoid any extra financial costs and we are hoping to have our proof-of-concept model, which is a search and rescue system, completed before December 16th.