Autonomous Search and Rescue (SAR)

Matt Hansen
Kyle Hargreaves
Quin Thames
Miguel Vazquez

Table Of Contents:

1. Introduction

Search and rescue (SAR) is the act of searching for and provision of aid to people who are in distress or immediate danger. Depending on the threat level of the surrounding environment, current search and rescue operations can involve putting personnel in harms way, as well as diverting these same personnel from accomplishing other mission oriented tasks. Instead of saving lives, the search and rescue operation can lead to more casualties and deaths as seen in Figure 1. This scenario is an extremely real possibility in combat, natural disaster, or mass casualty environments. This project delves into the possibility of using autonomous platforms (quadcopter and ground rover) as a search and rescue team in order to reduce the risk of possible injury and allow effective use of all available resources. This report will cover the details of this project including the different subsystems of the project, results during testing, and how the project demonstration was conducted.

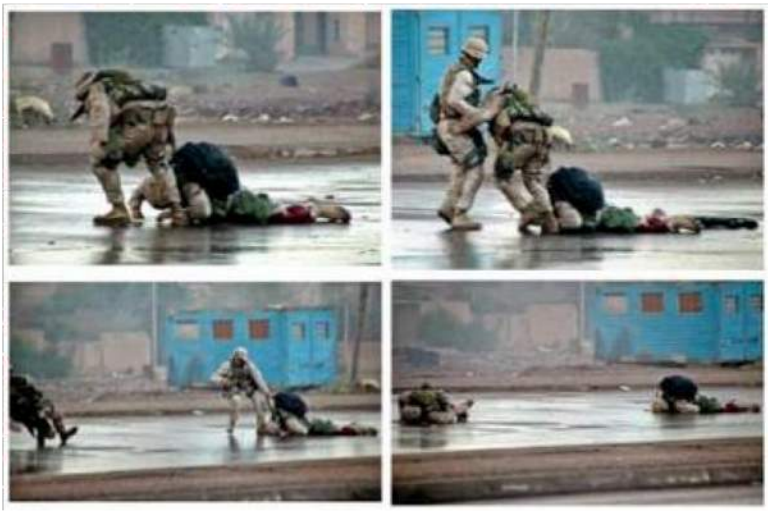


Figure 1: SAR Operation Causing Another Casualty

2. Objective and Purpose

The goal of this project is to design an autonomous search and rescue system that utilizes quadcopter and ground rover platforms. This system would likely be desired for use in military operations to avoid the need of putting additional lives at risk. Currently, the military has implemented remotely piloted drone rescue missions. However, if successful, this autonomous system would also eliminate the manpower required to control these drones. While the goal of this project is not to have a finished, field-ready SAR system, the project will serve as a proof of concept on a smaller scale that this kind of autonomous system is possible. For this project, the objective of the system is to be able to scan a field, find and recognize an object, and tag it with a ground rover. The system implements the following four subsystems in order to meet these objectives:

• GPS-guided quadcopter to conduct pre-programmed flight plan over desired search area
• Image processing unit to scan, interpret, and recognizes target object
• Wired/wireless serial communications between system platforms
• Rover to receive target object coordinates and “tag” the object

These four subsystems cover different areas of electrical engineering ranging from quadcopter/rover hardware to image processing, serial communication, and software design/modification.

The purpose of this project is to challenge the team members’ knowledge in multiple electrical engineering topics, to build a prototype to showcase our engineering knowledge, and to work as a team to prepare for real world projects.

3. System Description

The final design consists of a 3DR ArduCopter Quad C Frame with on-board GPS, a wireless transmitter, a receiver that is synced with the ground unit, and a computer which acts as a base station for the system. A Raspberry Pi with a camera module serves as an image processor and is integrated with the quadcopter’s flight controller (modified Arduino Mega). The ground unit is built from the Dagu rover 5 frame equipped with a Arduino Mega microcontroller, and wireless receiver that is synced with the base station computer. Figure 2 illustrates the level-1 system processes.

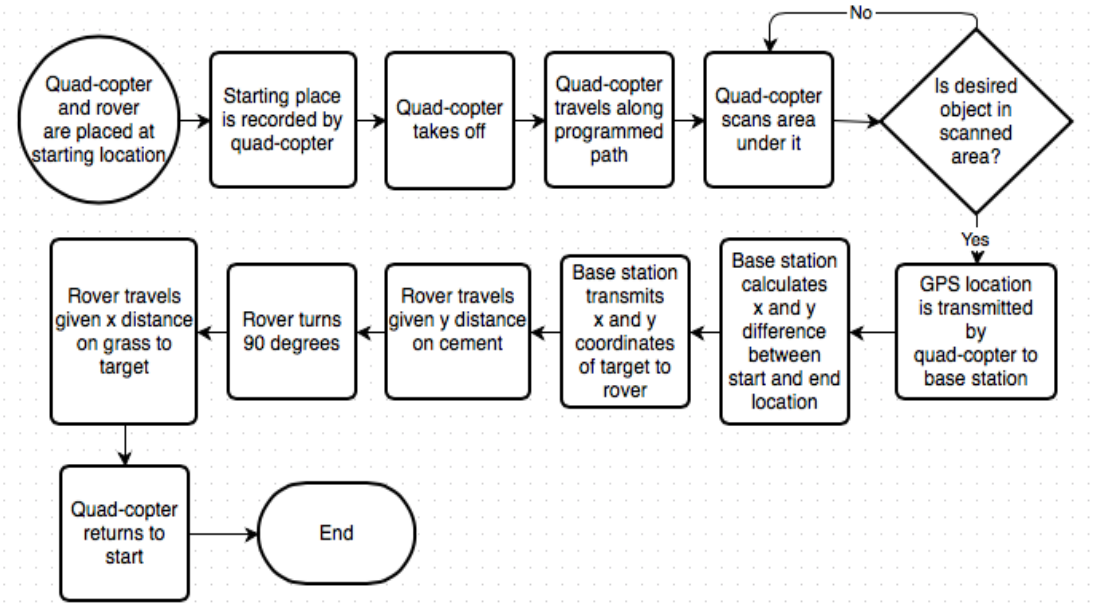


Figure 2: System Flow Chart

4. Subsystems

4.1. Autonomous Quadcopter - The quadcopter platform used was the 3D-Robotics ArduCopter C-Frame Kit with AdruPilot Mega (APM) 2.6 flight controller. The completed quadcopter, after wiring, assembly, and software initialization, is illustrated below in Figure 3. An Arduino-compatible GPS unit is used in conjunction with the on-board processor to create a predetermined flight plan. The quadcopter is uploaded with GPS waypoints to create a search pattern which the copter will fly autonomously. These waypoints are loaded before flight using the Mission Planner Software. A maximum error threshold for the GPS was determined (2.5m), which is taken into consideration when creating flight plans. The quadcopter has a flight time capability of roughly 15 minutes (dependent on flight speed and acceleration). This was calculated by testing different batteries (charge capacity is directly correlated to battery weight) and determining which weight to charge capacity ratio allowed for the longest flight time. After much testing the battery chosen for optimal flight time was the 4500mAh 3S 30C battery. This larger battery also has the added advantage of higher flight stability during high wind operations because the quadcopter is heavier and harder to be moved around by the wind.

Figure 3: 3D-Robotics ArduCopter C-Frame Kit

Once the Raspberry Pi with camera module, which is mounted on the underside of the quadcopter with the camera facing directly down, detects the target object (this process is described in the “Image Processing” section), the quadcopter needs to stop following the pre-programmed flight plan, loiter over the found object, send its current GPS location back to the base station, return to launch position and land. In order to accomplish this, the ArduCopter source code was modified. It was determined that once the object is detected, the Raspberry Pi sets one of its output pins to a logic high (5V). This output pin is connected to an analog input pin (A0) on the ArduPilot Mega 2.6 microprocessor. The modified code, which can be found in

Navigation.ino in the ArduCopter folder of the source code, continually reads the analog input pin to determine whether not it has received a logic high (5V) from the Raspberry Pi. If a high input is present, the code instructs the quadcopter to send its GPS location to the ground station using the telemetry system, loiter ten seconds, and return to its launch position (which is recorded and saved in memory during takeoff). A portion of the code modification is illustrated below in Figure 4.

```
int16_t found = 0; //variable is set to 1 once object is found
static void run_nav_updates(void)
{
    //checks to see if the object has been found and if the quadcopter has loitered for 10 seconds
    //once true the quadcopter will return to launch positon and land
    if(found == 1 && verify_loiter_time()){
      found = 0;
      set_mode(RTL);
    }
   //SONAR INTERRUPT -- pin A0
    int16_t temp_alt = sonar->read();
    if(temp_alt > 200) {
      found = 1;

      //MAVLink command with unique message ID used by outside program to know the next packet contains GPS location
      send_gps_raw(MAVLINK_COMM_0);

      //MAVLink command containing the the quadcopter's current latitude and longitude
      send_location(MAVLINK_COMM_0);

      //calls the function which will have quadcopter loiter for 10 seconds
      do_loiter_time();
    } else {

      // fetch position from inertial navigation
      calc_position();

      // calculate distance and bearing for reporting and autopilot decisions
      calc_distance_and_bearing();

      // run autopilot to make high level decisions about control modes
      run_autopilot();
    }


}
```

Figure 4: Modified Navigation Code

At the time of the project, the most stable ArduCopter version was 3.1.2. This version was chosen for modification even though it has since become outdated. The code is written in C++ with a suite of unique libraries. Because the code has extensive modified libraries, normal Arduino functions and variables such as “analogread( )” were not available and adding new libraries caused errors throughout the code. After studying the code and understanding its own functions and variables, it was determined that the analog pin A0 was reserved for the input of an optional sonar sensor and that sonar could be read by a function and the value (originally distance in cm) is returned in the variable temp_alt. The returned value when the Raspberry Pi output pin was low is ~33cm and the returned value when the output pin was set high was ~327 cm. Using this knowledge, the quadcopter would only break from its

pre-programmed flight plan to perform the previously described tasks if the value of "temp_alt" was greater than 200 cm. Additionally, the sonar check was placed before the "run_autopilot( )" function, which allows the quadcopter to stop following the preprogrammed flight plan. The "run_nav_updates( ) function is top-level code and is called at 100Hz. This ensures that the quadcopter checks the analog input before getting to lower level code and can immediately recognize that the target has been found.

Once the quadcopter is notified that the target has been found, two MAVLink commands are sent using the on board telemetry system (connects the quadcopter and base station wirelessly). MAVLink is a very lightweight, header-only message marshalling library for micro air vehicles (MAV). It can pack C-structs over serial channels with high efficiency and send these packets to the ground control station. Each MAVLink command has a unique message ID. Since the quadcopter is continually sending its coordinates back to the ground station by default, the send_gps_raw( ) command (message ID 24) is sent once the object is found prior to sending the send_location( ) command (message ID 33). This allowed us to search for the unique message ID of send_gps_raw( ), which will only be sent once, and look for the next send_location( ) message ID, which is sent continuously by the quadcopter for navigation purposes. Figure 5 illustrates the data being sent between the quadcopter and the ground station when the object is detected.

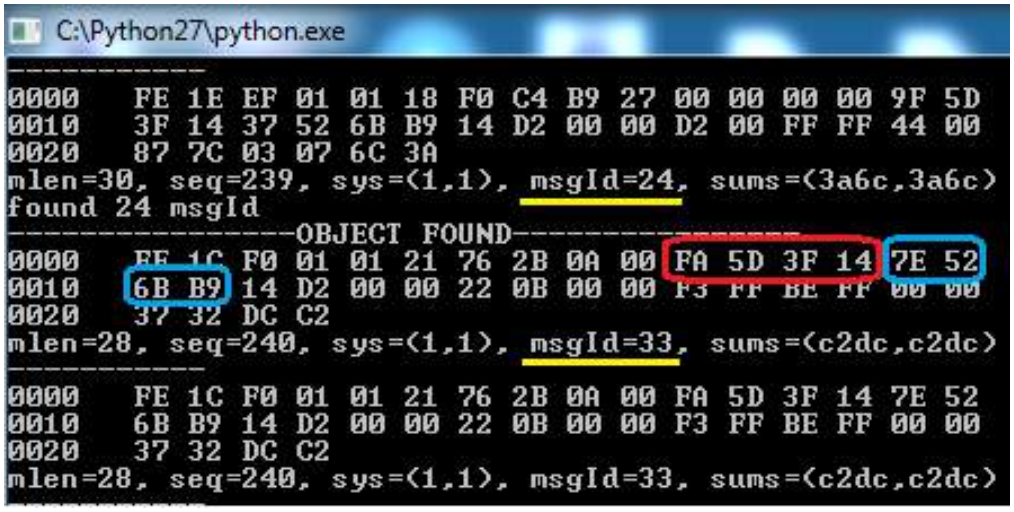


Figure 5: Example of telemetry data when object is found

A program which parses and displays the MAVLink data was modified to search for the messages with message ID 24, which is sent only when the target object is found. Once a message ID of 24 is detected, the next message with an ID of 33 is parsed in such a way that the bytes circled in and blue and red are saved. These bytes contain the latitude and longitude of the quadcopter at the time the object is found. They are stored in the message using the little Endian convention and are unpacked in such a way that reverses the order to big Endian format. The contents are then converted

from hex to decimal and the mantissae of the latitude and longitude coordinates are placed in a string. This string is saved in a text file that will be used by the rover.

Note: All source code mentioned in this section can be found at https://github.com/offDaCharts/arduCopterSketches

4.2. Image Processing - The image processing required a separate processor from the Arduino on board the quadcopter. A Raspberry Pi with a camera module (seen in Figure 6) was purchased. The image processing is required to find an orange star. Additional decoys of various shape and color will be placed in the search area. Therefore the processing algorithm will have to distinguish between shape and color. The image processing code looked at hue to distinguish between different colors and then used angles of points to differentiate between shapes. By having the image processing unit look at the angles of points, the system can determine a star shape even if the shape is partially blocked. The Raspberry Pi was mounted onto the bottom of the quadcopter. It was then linked to the microcontroller on the quadcopter so that when the target is located a high pulse is sent to the quadcopter to tell it to go into loiter mode. Once in the loiter mode the quadcopter can send its location to the base station. The following process describes how this is accomplished.

Figure 6: Raspberry Pi with Camera Module

The linux distribution Rasberrian was installed on the Raspberry Pi following this tutorial:
https://www.andrewmunsell.com/blog/getting-started-raspberry-pi-install-raspbian.
For development purposes, it is recommended to install ssh on the raspberry pi using the command:

```
sudo apt-get install ssh
```

Once ssh is installed, then the Raspberry Pi can be accessed from the shell on any computer on the network using the command:

```
ssh -X [IP ADDRESS OF PI] -l pi
```

Next, the OpenCV library that is used for the image processing was installed (https://github.com/andygrove/rasp_pi_camera_opencv). All of the code for the image processing can be found on github at

https://github.com/offDaCharts/autonomousSAR. The image processing program can be run directly from the command line in using the command:

```
sudo python scan.py
```

Or it can be configured to run the program automatically on startup by adding this line of code to the file /etc/rc.local:

```
(sleep 1;python /home/pi/autonomousSAR/scan.py)&
```

Note: Root permissions are needed to modify this file. The program outputs a high to pin number 7. The following figure shows how the wires from the quadcopter Arduino hookup to the Raspberry Pi.

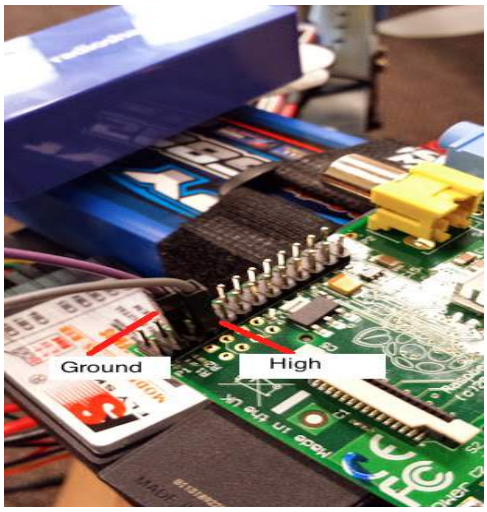


Figure 7: Raspberry Pi Pin Connections

The algorithm for the image processing works by first filtering the picture by color. This is done by taking a picture, and filtering each pixel to turn the image into a binary image where each image is black or white. Each pixel is filtered by comparing its hue, saturation, and value levels to a lower and upper threshold bound. The resulting image after this step is shown in the figure below.

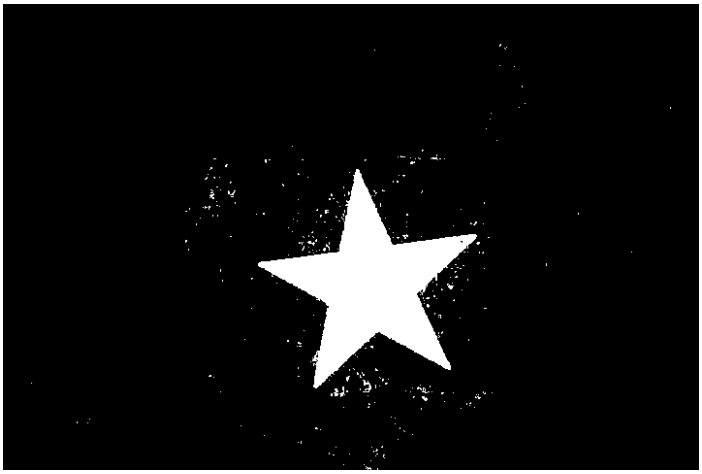


Figure 8: Filtered Binary Image

It should be noted that this filtering can be sensitive to lighting changes based on time of day or weather that can change the perceived color of the star causing the filter not to work. Once the image is filtered, then the lines of the star are detected using the Houghlines transformation (http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html). Once the lines are detected on the star, the image is as shown below. By the Houghlines transform, some of the lines can be detected more than once.

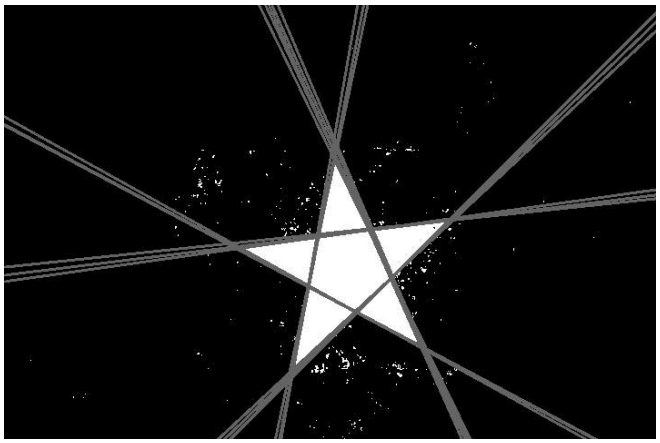


Figure 9: Filtered Image with Lines

Once the lines have been detected, then the angle between each pair of lines is calculated. If an angle falls within a threshold of the desired 36 degrees than a star point and therefore the star is detected and a high output is sent.

4.3. Wireless Communication - Once the target is located, its position, which is automatically sent back to the base station computer for navigation purposes, will be sent to the ground rover from the computer. Another reason why the the computer is needed in communications between the quad-copter and the rover is due to the fact that the telemetry systems used in the design have dipole antennas. The radiation pattern of dipole antennas, illustrated in Figure 10, is unable to transmit or receive data directly above or below the antenna. Since the antenna on the quadcopter is vertically fixed, it cannot transmit or receive from objects directly below it. This would have caused communication issues between the rover and quad-copter as the rover was approaching the target.

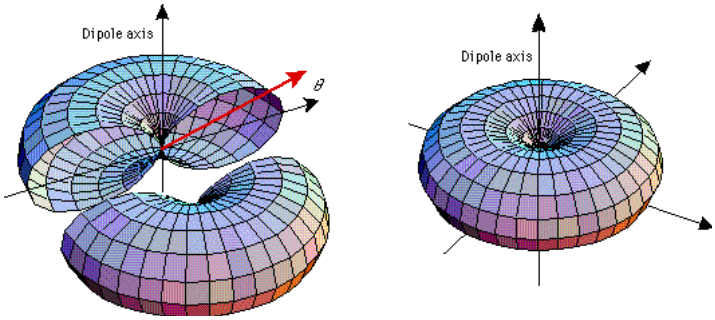
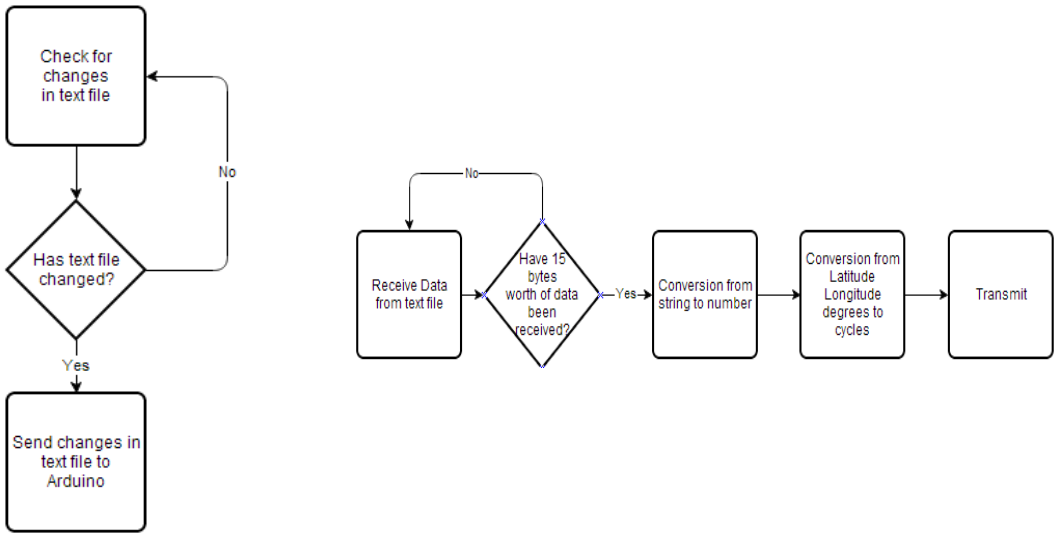


Figure 10: Radiation Pattern for Dipole Antenna

The system consists of the telemetry kit, an Arduino Micro and an Arduino Mega, as well as a pair of Parallax RF Transceivers. The quadcopter continuously sends GPS locations to the base station computer in order to track its location and the waypoints. Once the quadcopter finds the target it sends a command to the base station to save the next GPS location. The software for the quadcopter will then output the coordinates into a designated text file. Using the Processing IDE which will be running Java code, the text file will be scanned to see if any changes have been made to it. Should it sense that there has been a change it will write to the Arduino Micro. The Micro will be connected to the computer and continuously reading to see if anything is written to it. After it has read 15 bytes which is the size of the expected coordinate string it will stop reading and go through conversions. The string will be converted to numerical values and then turned into cycle values which is what the rover uses to move. When converting the string into numerical values it was calculated that for every 2×10^{-7} shift in degrees there was a one inch shift. This was tested by calculating the degree change along a latitude line in the Mission Planner software along the quad. The line was then measured using a measuring tape. The number of degrees was divided by the number of inches to find the number of degrees for a one inch shift. These cycle values will be placed into an array which is to be transmitted in a way that latitude cycles are sent first and then longitude. On the receiving end the Arduino Mega will be attempting to receive until it has a pair of values. After that it will go about the movement functions mentioned in the rover segment. A flowchart showing the way the transmission code worked can be seen in Figure 11a and Figure 11b.



(a) (b)

Figure 11: Code to read from file (a) and conversion and transmission (b)

4.4. Rover - The rover frame purchased and assembled is the Dagu 5 rover. This rover was chosen because there are four independent motors, wheel rotations can be counted, and it interfaces well with the Arduino Mega as seen in Figure 12.

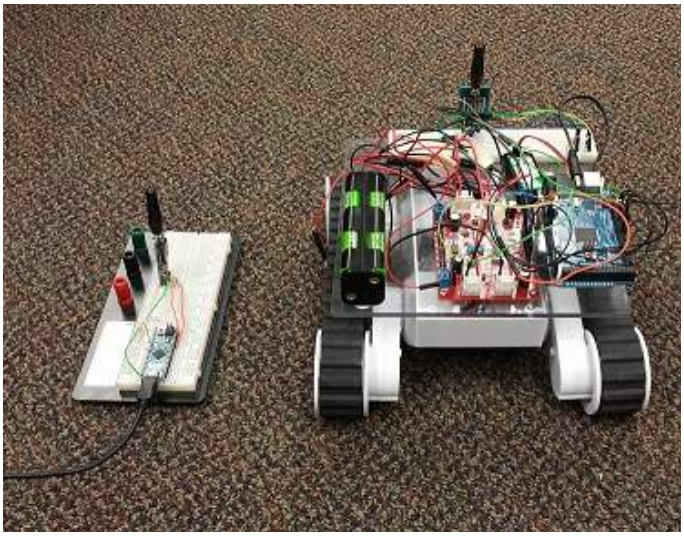


Figure 12: Rover Connected to Motor Controller and Arduino Uno

The rover is capable of moving forwards, backwards, and turning 90 degrees. This is accomplished by applying a voltage to the direction pins of the motor controller board. In order for the rover to move in the forward direction all the wheel direction pins are sent a logic 1. This make them all spin in the forward direction, pushing the rover

forward. Similarly, a logic 0 is sent to all of the direction pins in order to make the rover go in the opposite direction. Lastly, in order to make the rover turn the motors on one side of the rover are sent a logic 0 and the other side is sent a logic 1. This makes the two sides of the rover spin in opposite directions and rotates the rover. This can be seen in the code shown in Figure 13.

```
//Turn on all motors one direction to move straight
if ( i < cyclesLat )   {
  digitalWrite (DIRECTIONA, 1);
  digitalWrite (DIRECTIONB, 1);
  digitalWrite (DIRECTIONC, 1);
  digitalWrite (DIRECTIOND, 1);
  time_to_go = TIME_FORWARDS;
  i++;
}
//Turn on two motors to move forward, two to move backwards in order to turn
else if ( i == cyclesLat ) {
  digitalWrite (DIRECTIONA, 0);
  digitalWrite (DIRECTIONB, 1);
  digitalWrite (DIRECTIONC, 1);
  digitalWrite (DIRECTIOND, 0);
  latBoolean = false;
  time_to_go = TIME_TURN;
  i++;
```

Figure 13: Code Showing Forward and Turn Movements

During testing, it was observed that the rover had a tendency to go to the left on the cement when both sides motors were powered equally. In order to make the rover go straight on cement the left motors were powered higher than the right side. With this higher power the motors on the left turn at the same rate as the right and the rover moves in a relatively straight on the cement. This difference in power can be seen in the code in Figure 14.

```
analogWrite (MOTORA, 230);
analogWrite (MOTORB, 200);
```

Figure 14: Powering Left Motor Higher Than Left

The rover will receive a location of the target object from the quadcopter that is relayed through a base station using the Parallax RF Transceivers. The rover will receive X number of cycles needed to travel in the latitude and Y number of cycles needed to travel in the longitude to reach the location of the target. Each cycle makes the rover travel 56 inches. This was determined by sending the one cycle several times and measuring the distance it traveled. The rover will travel the Y number of cycles latitude on the cement, turn 90 degrees, and go the X number of cycles longitude on the grass. Finally, the rover will get to the general area of the target and “rescue” the object and stay at the target location. The block diagram of the system can be seen in Figure 15.

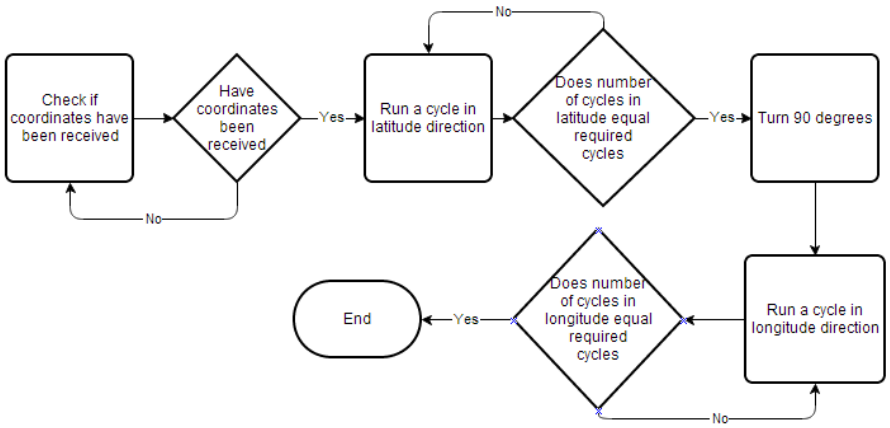


Figure 15: Block Diagram of Rover Function

5. Live Demonstration Procedure

The live demonstration for this project took place in Pereira Quad on May 6th, 2014 between 8am-12pm. For the live demonstration, three different shapes were placed in the quad area. The shapes consisted of a blue star, an International Orange triangle, and an International Orange star. The International Orange star was the target which the image processing unit was looking for. The quadcopter and rover started at one corner of the quad at a predetermined start location. The quadcopter, which had a flight pattern already loaded onto the microprocessor, started the flight pattern by flying up at a height of ten feet off the ground and start scanning the area using the image processing unit attached to the bottom of the quadcopter. Upon finding the International Orange star, the quadcopter went into loiter mode and sent its GPS coordinates to the base station. Once the coordinates had been sent the quadcopter loitered for 5 seconds and returned to base. The base station exported the coordinates into a text file so that the transceiver could import the coordinates and send them to the rover. The rover then moved in the Y direction the amount of latitude cycles sent on the cement, turned 90 degrees, and moved in the X direction on the grass. Once it had completed going to the area of the target the rover will stay at the target location. The demonstration can be seen in Figure 16.

Figure 16: Result from a demonstration

6. Setup Procedure

1. Calibrate the Quadcopter making sure the compass is correct along with switches and failsafes to have safety precautions like return to home or land.
2. Load the Quadcopter with the GPS waypoints using the Mission Planner Software.
3. Place the Quadcopter at the start position of the rover and record this GPS location.
4. Update the transmission code to include the new start location of the rover.
5. Upload the new transmission code, called “reading” onto the Arduino Micro and leave plugged into computer for power.
6. Upload Rover1 code onto Arduino Mega that is onboard the rover.
7. Open the programs
    a. Ensure the text file to be read is cleared
    b. Run Processing IDE code
    c. Run Python Parser
8. Place the Quadcopter in the starting location.
9. Plug in the Raspberry Pi to power source.
10. Place the rover in the starting location and plug in the power supplies (9V into Arduino Mega and 8AA Pack into the Motor Controller).
11. Place the shapes in the quad.
12. Start the motors on the Quadcopter.
13. Put the Quadcopter into autonomous mode and watch it move to the waypoints, find the star, and return home.

8. Project Improvements

Since this is the first prototype of the project there were areas in which improvements could be made in order to make the project better. One of these improvements would

be adding a control system to the rover. The rover did not have the ability to determine whether or not it was actually getting near the target as it was moving. It also could not determine whether or not it was moving in a straight line. A couple ideas were thought of in order to make the rover better. The first idea was attaching a GPS unit to the rover. The rover could then receive the GPS location and move toward the received location. If the rover was moving and not getting closer to the coordinate received, it would know that it is going the wrong way and not able to rescue the target. Another idea was attaching a compass to the rover and having it check the degree that it is facing every cycle. This would help the rover maintain a straight path without getting knocked off course. The third idea was to get the rover with a suspension system. This would allow the rover to traverse rough terrain without having every small bump or change in slope effect how much traction each side of the rover had. Lastly, a system in which the quadcopter could constantly update the rover and its location to the target would be the best. This system could tell whether or not the rover is headed in the right direction of the target and how far away from the target it is located. These improvements would help make the rover much better than its dead reckoning code.

Another improvement that could be done would be the usage of a better wireless transmission system. The project started up with an Arduino UNO using a wireless shield with an XBEE module. However connecting to the XBEE proved to be difficult when not communicating with another XBEE. However research did find that with another apm and proper microcontroller the rover could actually be connected to the Mission Planner software. That way the same software could potentially synchronize the copter as well as the rover with one another and lead to better results in terms of how close the rover was to the quadcopter location. This would still require a base station but would lower the amount of in between code and resources.

The image processing was susceptible to errors in different lighting and could not recognize the star when it was darker. A possible solution to this would be for the quadcopter to have a search light so the light that the star was in would be more consistent. Furthermore, the camera could not detect the star and if the quadcopter was moving too fast due to blurred images. This could be improved with a better quality camera that could take moving images more clearly.

9. Engineering Practices

According to the IEEE code of ethics in recognition of the importance of our technologies affecting the quality of life throughout the world, and in accepting a personal obligation to our profession, its members and the communities we serve, we

committed ourselves to the highest ethical and professional conduct. During the project we practiced safety for the users of the SAR system and those around us. The quadcopter was programmed to have safety measures including a return to home function and power down function if it was not behaving properly. We also would not fly the quadcopter in the quad with other students walking through the quad. When presenting the results of our search and rescue project we have been honest and realistic in stating claims or estimates based on the available data. Our project relied on using some open source codes for the image processing. This helped us improve the understanding of image processing technology as well as adapting it for appropriate application. As a group we feel that we have maintained and improved our technical competence and to undertake technological tasks for others only if qualified be training or experience, or after full disclosure of pertinent limitations. We have shown this by making slight changes to the project in order to get a working SAR system will the limitations of time. By presenting our work in periodic segments we have seeked, accepted, and offered honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others. Finally, we have assisted colleagues and co-workers in their professional development and have supported them in following the IEEE code of ethics by helping in overlapping areas of projects.

10. Summary

This project required a mixture of both hardware and software as well as a variety of different required technologies. The solution for the provided problem involved autonomous flight for a quadcopter, image recognition to find the target, wireless transmission between multiple systems, and a rover reaching the target by ground. This project shows that an autonomous search and rescue system is possible and can be implemented in a larger scale. The system can still be improved further by making the quadcopter hover directly over the International Orange star for more precise GPS location, having the image processing detect the rover and send the rover updates towards the star, and make the rover more accurate on different terrains. The system can also be improved by making the rover able to detect obstacles, so that the system can be implemented anywhere, along with having the rover locate the quadcopter while in loiter mode in order to be more accurate in finding the target. The system can also be altered for different situations such as in an avalanche. In this case the image processing could be switched for some type of thermal camera to look for buried skiers or snowboarders. Hopefully this project wins the Air Force research award and can be put into a larger scale operation for the military and can start saving lives in the next 5-10 years.

10. Appendix

10.1 ABET Outcomes

Below are a list of the ABET outcomes and examples of how they were demonstrated throughout this project:

A) An ability to apply knowledge of mathematics, science and engineering

The project required the calculation of a shift in degrees to inches in order to find how many cycles the rover needs to travel in the latitude and longitude of the quad. The project required designing a quadcopter and rover to use image processing and microcontrollers.

B) An ability to design and conduct experiments, as well as analyze and interpret data

One experiment that was run was to see if the image processing unit was detecting the target. By attaching a buzzer to the output pin of the image processing and putting the image processing unit over the target the buzzer would make a sound if the object was found or stay silent if it was not found. If the image was not found a still picture was taken and the image processing algorithm was run to see what was appearing in the image. If a star was not being seen the thresholds of the hue were changed until the star could be found. This was one of the many experiments designed and conducted in order to make a functioning search and rescue system.

C) An ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability

Throughout this project the goal was finding a way to create a system that could save lives. This system could be used by any group of people in order to help those in distress in times of war, natural disasters, or catastrophic events. The system designed does not help one group more than another.

D) An ability to function on multidisciplinary teams

This project required a lot of hardware and software. This meant having to work with electrical engineers and computer science majors to have a completed project. The electrical engineering majors in this project were put in charge of the hardware of this project. This was mainly building the quadcopter and rover to have basic functionality. Computer science was needed in order to get image processing working along with transmission of code, receiving code wirelessly, and rover movability.

E) An ability to identify, formulate, and solve engineering problems

Throughout the project there were times that the project faced problems. One of these problems was finding the GPS location of the quadcopter and getting it into a file for the transmitter to read. The Mission Planner software would not allow the GPS to be accessed while it was running. The solution was to take the Mission Planner code and have it send the coordinates to a text file when the interrupt was detected.

F) An understanding of professional and ethical responsibility

This project taught us to be professionally and ethically responsible by not falsifying our results. This project taught us to give updates on the project every few week in order to show the progress on the project and to manage our time wisely.

G) An ability to communicate effectively

Having the presentations to faculty, reports, and weekly meetings with Professor Lockenour has taught us the ability to effectively communicate our project. The feedback from the professors after the faculty review gave us areas we needed to work on which helped improve the final presentation.

H) The broad education necessary to understand the impact of engineering solutions in a global, economic, environmental, and societal context

The project allowed us to understand that this solution could help save hundreds to thousands of lives per year and to do it without putting more people in harms way. Our project also looks at ways to help people all over the globe and not just in war torn areas. The project could be expanded to help people in natural disasters or catastrophic events.

I) A recognition of the need for, and an ability to engage in life-long learning

One area in this project that showed us a need for and ability to engage in life-long learning was image processing. All four of the project workers had no previous experience with image processing and had to do research to find libraries about image processing. Another area that showed this need for continuous learning was the rover. When working with the rover many of us thought that it would be easy to get the rover to move in straight lines. As we worked more and more with the rovers it became apparent that this was a very common problem according to most forums. A couple of control systems were thought of in order to make the rover go straight, but because of time constraints this was not done.

J) A knowledge of contemporary issues

In light of years of military conflict in the Middle East, as well as recurrent large scale natural disaster including earthquakes, floods, and tornadoes, there is an increased need for a rescue method that eliminates the potential for more casualties or loss of

life during the actual rescue operation. This project aims to prove that this concept is is an extremely real possibility.

K) An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice

This project used modern tools such as GPS, RF transceivers, Arduino microprocessors as well as a Raspberry Pi. These required researching into their use as well as code that would work with them. Engineering practice includes research and constant innovation for solutions. This project required combining different tools to one system.

10.2. Project Costs

Aerial Unit:

3D-Robotics ArduCopter Quad C Frame Kit	$275.99
-- AMP Power Module w/ XT60 connectors	
-- APC Propeller set 10X47 SFP Style	
-- Motor 850Kv AC2830-358 with new prop adapter	
APM 2.6+ (with connection cables)	
$150.00	
3DR GPS uBlox LEA-6 w/compass	
$79.99	
3DR Radio Telemetry Kit	$99.00
Zippy Compact 5800mAh 3S 25C Lipo Pack	$45.66
FlySky FS-TH9X 9 Channel Remote Controller	
$119.99	

Image Processing Unit:

Raspberry Pi Model B board + 8GB SD Card	
$50.00	
Raspberry Pi Camera Module	$25.00
Portable Charging Station (micro USB)	$30.00
Raspberry Pi Heatsinks	$5.00

Ground Unit:

Rover 5 Robotic Platform	$59.95
Arduino Mega2560	$21.95
Motor Driver Board	$24.95

Wireless Transmission

Arduino Micro	$25.00

Parallax 433MHz RF Transceiver (x2)	$79.98
Total:	$1092.46

10.3. GitHub Repository Link

https://github.com/offDaCharts/arduCopterSketches
https://github.com/offDaCharts/autonomousSAR