Search-and-Retrieval System (SARS)

Matthew Bahr, Brian Crabtree, Brendan Hall, Erick Makris

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — In recent years, drone technology has emerged at the forefront of scientific application and innovation. The purpose of SARS is to use automated vehicles to implement a search-and-retrieval system which shall have a number of real-world applications, ranging from military to commercial use. SARS consists of three major subsystems: 1) a quadcopter used for object location 2) a central hub in the form of a Linux-based application known as the Command Distribution Center 3) a land rover used for object retrieval. SARS makes use of a number of technologies including object detection by image processing, wireless communications, and automated vehicle control algorithms.**

***Index Terms* — Image processing, microcontrollers, object detection, pulse width modulation, satellite navigation systems, unmanned aerial vehicles, wireless communication.**

I. Introduction

System automation and artificial intelligence has recently taken a place at the forefront of scientific research and development. From self-driving cars to automated homes, the latest trend is to relegate as many menial tasks as possible to computers, allowing people to focus more on creative activities. A desire to learn more about AI and system automation served as the primary motivation behind SARS.

A multi-faceted aerial and ground object detection and retrieval system, SARS is composed of three major subsystems – a quadcopter, a Linux-based console application, and a land rover – which perform together to complete a successful search-and-retrieval mission. A mission shall be deemed a success if the system relocates a target object in the form of a tennis ball sitting on a sheet of white poster board from some starting position to the land rover's initial location in a completely autonomous fashion. Throughout the entire process, the Command Distribution Center (CDC), shall provide real-time diagnostics and a live video stream of the mission to the user via a laptop computer running the latest version of Ubuntu.

All three SARS subsystems have been designed to streamline the system function and ensure the highest rate of success. The CDC shall serve as the intermediary between the other two major subsystems, communicating over radio as well as Wi-Fi. The details of these subsystems shall be covered in the sections below.

II. System Components

SARS is best described through detailed explanations of its specific subsystems; thus, this conference paper shall be structured around these subsystems. Section II shall serve as a basic introduction to the quadcopter, land rover, and CDC subsystems before additional detail is provided in the subsequent sections.

A. SARS Copter

The SARS Copter was built from the DIYDrones Quad Kit sold by 3D Robotics. The individual hardware components of the quadcopter package include the quadcopter frame, the Pixhawk flight controller, the U-blox GPS module, four electronic speed controllers, the power distribution board, the RC telemetry radio, the PPM encoder, and the lithium polymer battery. The Pixhawk flight controller runs Arducopter, an open source multicopter UAV platform. Missions may be loaded onto the flight controller using the Mission Planner software application. A GoPro Hero3 White is mounted just beneath the SARS Copter's nose with its field-of-view directed towards the ground. Images taken with this camera are used to detect the target object on the ground. In the context of a mission, the SARS Copter travels to a series of waypoints making up a rough 7ft x 13ft grid to facilitate the search for the target object. At each of these waypoints an image is taken from the GoPro and saved by the CDC for processing.

B. CDC

The Command Distribution Center (CDC) is a Linux-based console application which oversees the function of all SARS subsystems and governs the success of all missions. As the SARS Copter travels between its mission waypoints, the CDC pulls images from the live video feed provided over Wi-Fi by the GoPro. Once the SARS Copter's mission has been completed and all the images have been captured, a blob detection algorithm runs on the CDC to check each image for the target object. If the object is detected, the GPS coordinates of its approximate location are relayed wirelessly to the microcontroller mounted on the SARS Rover. In addition to wireless

communications between the two unmanned vehicles and coordination of subsystem functions, the CDC acts as a user interface for the entire system. It provides a live video feed taken by the GoPro as well as diagnostics and mission event updates from the SARS Copter and Rover subsystems.

C. SARS Rover

The hardware components which make up the SARS Rover include the chassis, the anodized aluminum plate frame, six 6V motors and wheels, the Turnigy lithium polymer battery and power switch, the Sabertooth 2x25 Regenerative Motor Controller, the Texas Instruments Tiva C Series microcontroller, the Honeywell HMC5883L compass, six A3144 Hall Effect Sensors, the PING))) Ultrasonic Sensor, the Adafruit Ultimate GPS Breakout, , and two servo motors and a metallic robot arm for object retrieval. The SARS Rover function is not initiated until after the target object is located by the SARS Copter and CDC subsystems. It receives the approximate GPS coordinates of the target object and travels to those coordinates using its compass, GPS, and a control loop. Along the way, the rover uses the PING))) sensor to avoid obstacles. Once the SARS Rover has reached the received coordinates, it begins a systematic search of the area for the target object once again using the PING))) sensor. When the object has been found, the SARS Rover lifts it up with its robotic arm and travels back to its starting location.

III. SARS COPTER

A. Hardware

The Pixhawk flight controller houses the copter's internal measurement unit which contains a gyroscope, magnetometer, accelerometer, and barometer. The Pixhawk is loaded with Arducopter's firmware which provides stabilization to the copter and allows the quadcopter to be compatible with our testing software, Mission Planner, as well as the pyMAVLink protocol we use to communicate with the CDC. It is compatible with UART, SPI, and I2C communication protocols, the latter of which is used to communicate with the GPS, RC telemetry, and Turnigy receiver. The Pixhawk also uses a pulse-position modulation sum signal to regulate the ESC's, which was taken into consideration when purchasing the transmitter controller.

The peripheral components are the u-Blox GPS, 3DR telemetry radio, Turnigy receiver, and PPM sum encoder. The u-Blox GPS is a highly efficient external GPS module that is accurate up to 1 meter, an operating voltage of 3.3V, and a 5HZ update rate. The u-Blox has exceeded expectations and allows for waypoint navigation with pinpoint accuracy, often to within less than a foot. The telemetry radio allows for 2-way full duplex communication between the CDC and quadcopter using time-division multiplexing. Messages are sent over the 915 MHz bandwidth (the US standard), and the radios have built-in MAVLink protocol framing, as well as error correction for up to 25% if bit errors. Finally, the Turnigy receiver and PPM sum encoder are used to receive input from the Turnigy transmitter. The transmitter sends eight separate pule-width modulation signals across eight separate channels, but since the Pixhawk is not compatible with that protocol, the PPM sum encoder combines the PWM signals into one sum signal, which the Pixhawk then uses to adjust the orientation of the quadcopter and change flight modes.

The quadcopter is powered using a 14.8 V, 4000 mAh lithium polymer battery connected to a power distribution board. The PDB is wired to all of the ESCs and the Pixhawk, which allow it to efficiently power all of the separate subsystems on the quadcopter. Figure 1 below is a table summarizing the details of the PDB.

	3DR Power Module
Max. Input Voltage	18 V
Min. Input Voltage	4.5 V
Max Current	90 Amps
Weight	38 g
Flight Battery Cable	6" 14AWG red/black cable
ESC Cables	4 female Deans connectors 1 XT60 connector

Fig. 1. Table of the 3DR power module specifications.

The ESCs have a programmable throttle range which can be calibrated using the Mission Planner testing software, a maximum RPM of 70,000, and 20 A of continuous current. The quadcopter is able to sustain flight for between 20 and 30 minutes, which usually allows for five or six missions to be run before having to recharge the battery.

The GoPro camera is mounted in its hard case just beneath the nose of the SARS Copter using industrial strength fabric hook and loop fasteners. A layer of neoprene separates the camera case from the copter frame to dampen vibrations and help provide a stable image to be processed. For a picture of this mount setup, see Figure 2 below.

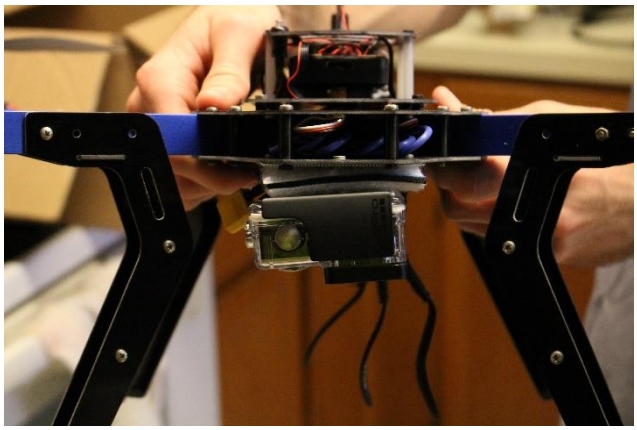

Fig. 2. Image of the homemade mount for the GoPro. A layer of neoprene separates the GoPro from the SARS Copter. Components are attached together using hook and loop fasteners.

B. Software

Communication with the quadcopter is done using a standard known as the MAVLink protocol, which stands for Micro Air Vehicle Link. The MAVLink protocol is a header-only message library that allows for the transfer of information based on a system ID (vehicle being communicated with), component ID (specific part of the vehicle from which information is being extracted such as RC servos, battery, etc.), and message ID and parameters (what piece of information is needed such as propeller speed, battery life, etc.). This communication protocol allows for easy transmission of commands and mission data that are used to control the copter while it is in the air.

More importantly, the MAVLink protocol is designed to allow for easy waypoint navigation, a feature critical to smooth autonomous flight. Before the mission is executed, a text file is prepared detailing multiple attributes of each waypoint the copter will be flying to. The parameters of each waypoint are: the sequence number, latitude and longitude, relative altitude to takeoff, waypoint radius (how close to the exact coordinate the copter has to be before it can determine it has successfully reached it), and a time (in seconds) to delay between reaching a waypoint and traveling to the next one. After the text file has been formatted, the mission can be loaded into the quadcopter and flown autonomously with ease.

IV. Command Distribution Center

A. Hardware

To enable the CDC to communicate with both the SARS Copter and SARS Rover, additional wireless communication hardware was utilized. The SARS wireless communication network consists of 3 dedicated connections, each utilizing its own wireless communication hardware interface. The three connections consist of a radio connection to the SARS Copter, a Wi-Fi connection to the SARS Rover, and another Wi-Fi connection to the GoPro camera. Figure 3 below diagrams the SARS wireless communication network.

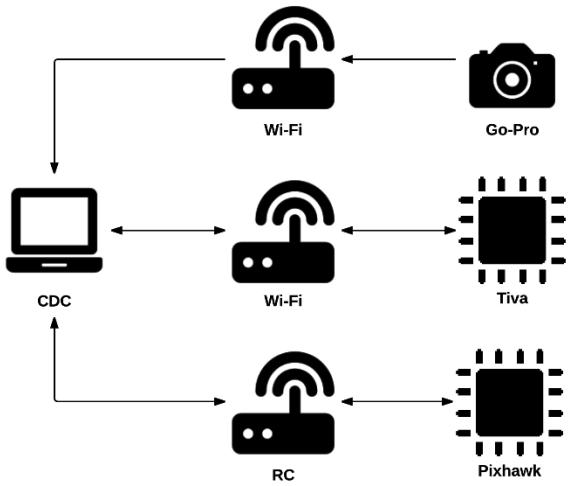

Fig. 3. The SARS wireless communication network consists of 3 separate wireless connections that all feed into the CDC.

A 915MHz telemetry radio provides wireless communication between the CDC and the SARS Copter. The telemetry radio is connected via a USB port, and it communicates at 57,600 baud. In conjuction with PyMavLink, this telemetry radio enables the CDC to issue commands and request parameters from the SARS Copter.

An 802.11n Wi-Fi connection to the SARS Rover is established using the internal Wi-Fi adapter of the laptop computer on which the CDC is running, and the CC3100 network processor connected to the SARS Rover. This connection utilizes the TCP protocol to send and receive messages between the CDC and the SARS Rover. This enables the CDC to receive status messages from the SARS Rover, and to transmit the start command which sends the GPS coordinate to the SARS Rover so that it can begin traveling to the target.

Another 802.11n Wi-Fi connection is established between the GoPro camera and the CDC. This connection utilizes an external USB Wi-Fi adapter and the internal Wi-Fi module of the GoPro to provide both live video streaming and image capturing. This secondary Wi-Fi adapter is an optional, but beneficial hardware component. The GoPro camera is only capable of ad-hoc Wi-Fi connections, which means that a centralized wireless

network infrastructure cannot be utilized, and both Wi-Fi connections must be handled separately. The internal Wi-Fi adapter could be used to alternate ad-hoc connections between the CC3100 and the GoPro, but this would constantly interrupt either the live video feed or the rover telemetry reporting. Utilizing the secondary Wi-Fi adapter provides a better user experience, thus the decision was made to include it.

B. Software

The Command Distribution Center (CDC) is a Linux-based application which coordinates the operations of both the SARS Copter and SARS Rover. As the SARS Copter traverses a series of mission waypoints, the CDC pulls still images from the GoPro's live video feed via Wi-Fi and saves them in an indexed array. Once the SARS Copter has completed its mission and all the images have been captured, a blob detection algorithm runs on the array to check each image for the target object. If the object is detected, the index of the image is cross referenced with a lookup table containing the GPS coordinate of each waypoint in the mission. This GPS coordinate is then transmitted wirelessly via Wi-Fi to the microcontroller mounted on the SARS Rover. The pseudo code in Figure 4 below illustrates the algorithm as it is implemented in the CDC.

```
launch quadcopter
while waypoint number less than max waypoint number
    travel to waypoint
    while distance to waypoint greater than 0
        wait
    wait 3 seconds
    capture GoPro image
land quadcopter
while current image number less than max image number
      run object detection on current image
if object found
    initiate rover
else
    mission failed
```

Fig. 4. Pseudo code for the mission SARS Copter mission algorithm.

In addition to handling both wireless communication and coordination of the two unmanned vehicles, the CDC acts as a user interface for the entire system. It is composed of 4 main interface components: the live video feed window, the telemetry window, the mission status window, and the command console. Figure 5 below provides a screen capture of the CDC.

The live video feed window utilizes the LibVLC media framework to connect to the GoPro's HTTP server and stream video utilizing the HLS (HTTP Live Streaming) protocol. Play and Stop buttons are included so that the video feed can be started or stopped at any time. This live video feed provides the user with a real time, first person view of the SARS Copter as it performs its mission.

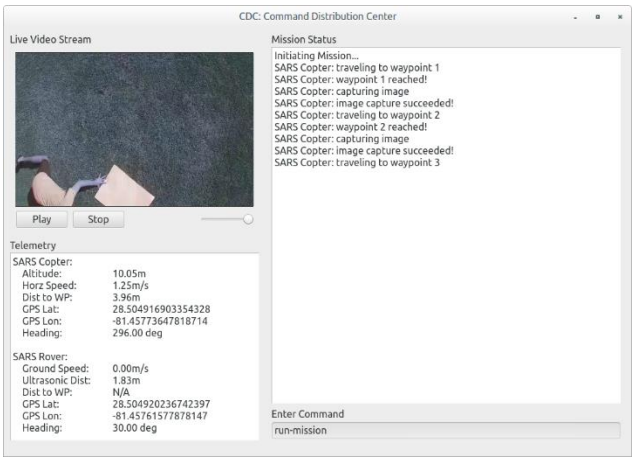

Fig. 5. Screenshot of the CDC application as it pulls data from the SARS Copter and displays the GoPro video feed.

The telemetry window utilizes the RC telemetry radio that communicates with the SARS Copter to retrieve telemetry such as altitude, horizontal speed, distance to next waypoint, GPS coordinates, and current heading. It also utilizes the Wi-Fi connection to the SARS Rover to retrieve telemetry such as ground speed, distance to target object, ultrasonic sensor distance data, GPS coordinates, and heading. This information is periodically requested from the vehicles and automatically displayed in the telemetry window so that the user can monitor the status of the vehicles as the mission progresses.

The mission status window reports the progress of a mission as the mission algorithm executes. It reports information such as the SARS Copter's current mission progress, the success or failure of image captures, the index of the image containing the detected object and the corresponding GPS coordinate, and the SARS Rover's progress as it travels to the target object. The mission status window gives the user an overview of a mission so they can determine if the mission can proceed or if it must be aborted at any time.

The command console is a simple one-line text box that allows the user to issue commands to the CDC. Commands for both initiating and aborting missions are recognized. Various debugging commands are also included so that specific aspects of the mission algorithm can be tested separately, such as capturing an image from the GoPro or running the object detection on captured images.

The CDC was developed with the Python scripting

language. The UI was designed using Qt, which is a cross-platform application framework for designing and programming UI's so that they have the native look and feel of the operating system on which they are currently running. Qt is designed for use with C++, but Python development was made possible through the use of the PyQt framework. PyQt takes a UI form designed with Qt and binds it to Python scripts instead of C++ code. This generated script serves as the backbone on which all other Python libraries, scripts, and functions required by the CDC have been built.

To live stream the GoPro video feed, the decision was made to integrate an existing media player rather attempt to design one from scratch for the sole purpose of integrated live video streaming. LibVLC was chosen as it is free, open-source, capable of HLS streaming, and has bindings for several languages already available, including C, C++, Java, and most importantly, Python. LibVLC is also the media framework on which the popular media player VLC is based – a media player renowned for its stability and wide range of protocol and format support.

The CDC pulls images from the GoPro using an open source Python library called GoProController. This library imports OpenCV to extract individual frames from the GoPro's video feed, which it accesses via the ad-hoc network created by the GoPro. These images are stored in an array before being processed. Once all waypoint images have been stored, the CDC runs a simple blob detection algorithm to identify the large white poster on which the tennis ball will be sitting. Essentially, this algorithm checks each image for 10 consecutive pixels whose RGB values average out above a certain value. This value may be adjusted based upon the intensity of the natural light at the time of the mission. For the pseudo code for this algorithm, see Figure 6 below.

```
for each pixel row
    for each pixel column
        if the average of the RBG values > 200
            increment white pixel count
            if white pixel count == 10
                Object Detected!
        else
            set white pixel count to zero
```

Fig. 6. Pseudo code for the image processing algorithm.

Communication with the SARS Copter is achieved using the PyMavLink library. PyMavLink is an open source, Python implementation of the MAVLink protocol. In conjunction with the telemetry radio, PyMavLink allows the CDC to wirelessly request almost every parameter stored on the Pixhawk of the SARS Copter. These parameters are then used to display important telemetry data in the telemetry window, and to assist the mission algorithm with determining when the SARS Copter has reached a waypoint.

V. SARS Rover

A. Hardware

The SARS rover is based around the Dagu Wild Thumper 6WD all-terrain chassis. This chassis is designed with an aluminum frame and includes a pre-designed suspension system built using tension cords to balance out each pair of motors and wheels. The included motors are mounted using a t-shaped housing and are geared with a 34:1 ratio, resulting in a 295RPM output shaft. Each motor operates on 6VDC and has a 5.5A stall current.

The rover is powered by a 5000mAh, 3S1P Lithium Ion Polymer battery. This battery provides 11.1V, as well as 20C constant discharge and 30C peak discharge. The power circuit also includes a single pole single throw (SPST) toggle switch. The switch supports up to 50A at 12VDC and up to 25A at 24VDC.

The first major electronic component is the motor controller. The controller selected is the Sabertooth 2x25 Regenerative Motor Controller. This board will serve as the power control for the motors as well as the rest of the rover system. The motors will be connected so that the three left side motors connect to one channel and the three right side motors connect to the other channel. Each channel is capable of supporting 25A continuously as well as 50A peak. The board operates on a nominal voltage of 6-30V, well within the range provided by our battery. This controller also has a built in battery eliminator circuit (BEC) capable of supplying 5V and 1A. The BEC is one of the most important features here because it will provide power to the rest of the electronic components included on the SARS rover. The Sabertooth will also be receiving commands from the main microcontroller over a UART connection. More details regarding this system will be included in the Rover Software section.

The next key component is the microcontroller that will serve as the main brain of the rover. The selected microcontroller is the Tiva C series by Texas Instruments. The selected model is the TM4C1294XL. This model implements an ARM Cortex M4 processor operating at 120MHz. It has 1024KB of flash memory and 256KB of RAM. It also has a multitude of connection technologies including 8 UART modules, 4 SSI/SPI modules, 10 I2C modules, and 140 GPIO pins. All of the external sensors and other modules will be connected to this board directly or via our printed circuit board.

Sensors are another key component of the rover system. The simplest sensor implemented is a set of 6 A3144 Hall

Effect sensors. These sensors operate communicate with GPIO and require a 10kΩ pull-up resistor. These sensors will be used to monitor wheel rotation to calibrate minimum power required to move on a given terrain as well as to monitor whether or not the wheels are rotating freely while the rover is not moving. The next sensor is the Parallax PING))) ultrasonic sensor. This sensor also operates on GPIO and can detect objects in front of it from 2cm to 3m. This will be used for obstacle detection and avoidance as well as target object detection.

Another important sensor in this system is the GPS module. We have elected to implement the Adafruit Ultimate GPS for this purpose. This module is capable of working on 66 channels and can update at up to 10Hz. In order to relay information, it sends NMEA style sentences over a UART connection. The internal antenna has -165dBm sensitivity and an external active antenna is added to provide an additional 28dB gain. Coupled with the GPS is a compass module so that the rover is directionally aware. The selected compass is the Honeywell HMC5883L. This module communicates its three axes of measurement via I2C.

The last electronic component of the rover is the CC3100 WiFi module. This module is a Texas Instruments product and is designed to interface with the Tiva C using SPI. More about this module is discussed in the CDC hardware section.

The final hardware component of the rover is the object retrieval arm. The object retrieval arm will mainly consist of a 20mm square aluminum beam. This beam will be cut into two major pieces and attached at a right angle. The beams will be slightly different in length, with the longer

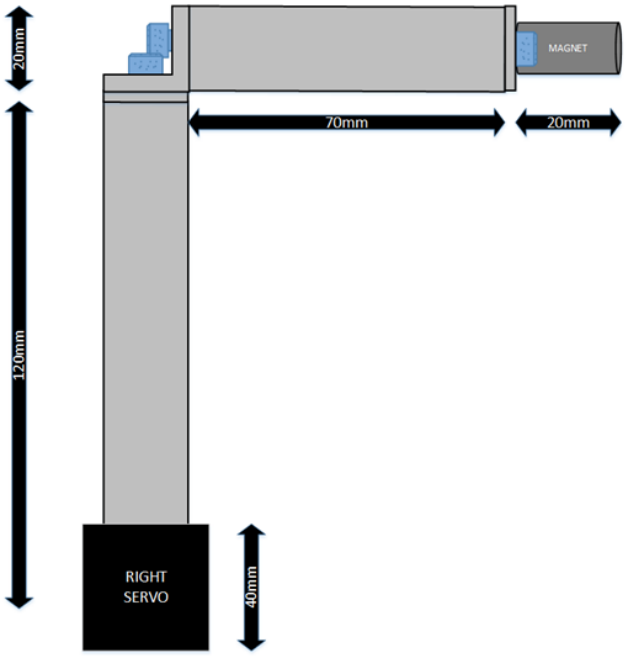

Fig. 7. Rover arm diagram.

piece attached to the control servos. Two servos will be used to ensure proper torque support. The general structure of this arm can be seen in Figure 7. Each servo is capable of handling 6.5Kg*cm. They are designed to operate at 5V and are capable of turning 60 degrees in .21 seconds. The servos also include an analog feedback line for maximum control. The target object will be picked up using a magnet attached to the end of the retrieval arm.

B. Software

The majority of the software for the rover is all custom written for this system. Coding is done using Energia, TI's version of the Arduino wiring language. Some external code is used for interfacing with various hardware components, but this will be referenced with regards to individual components of the software.

The first component of the software is the interface with the motor controller. The Sabertooth is connected to UART6 Tx on the Tiva C. This connection will be operated at a baud rate of 9600. This is decided by the configuration of the hardware switches on the Sabertooth. This connection will be used to transmit a series of byte-length commands to the Sabertooth, which will then execute those commands. The Sabertooth accepts a variety of byte patterns to designate the function to carry out. The selected set of commands operates using two major commands to the Sabertooth, one to control each channel. These commands are issued with a set of bit-level commands and the rest of the bits transmitted per byte of data define the direction and power to be transmitted to the motor. Each command gets 6 bits of resolution. The median value designates a stop value, while the max value indicates full forward and the minimum value indicates full reverse. The rover software is written to allow for easy command transmission using pre-calculated values. For example, for the main loop to tell send a movement command, it must simply call the transmit method indicating a direction (forward, reverse, left, or right) and a percentage (out of 100) indicating the desired power. This function is also augmented with an acceleration and deceleration function for smoother movement.

Software for the Hall Effect sensors is simple. It digitally reads the connected pin to produce a value. A low value indicates a magnetic field is detected and a high value indicates no magnetic field. Software for the PING))) sensor is also fairly simple. The control software pulls the output pin low to clear it and then writes it high to initiate sensing. Five milliseconds later, it pulls the line low again. The pin is then reconfigured as an input and read in as a pulse that defines the distance. Using a small

conversion, the distance is known in inches.

The next major component of the software is interfacing with the GPS module. This is also achieved using a 9600 baud UART connection, connected to UART2 of the Tiva C. Once a connection is established, the GPS module is configured to only send RMC sentences. RMC sentences are NMEA style sentences that contain the recommended minimum data from the GPS module. This data is transmitted to the Tiva C character by character in the form of a long string. Once a full string is received, which is denoted by a new line character, the string will be parsed to search for the appropriate data. The data parsing code is borrowed and slightly from the freely available Adafruit GPS code example. This parser will extract all of the relevant data and store it for later access.

In order to make any use of the GPS data, the compass must also be configured. The compass being used is in the form of a 3-axis magnetometer, so the values for each access must be retrieved and converted to a usable data heading. Communications with this module are initiated using I2C module 0 of the Tiva C. Code for pulling the data from this module is a combination of recommended commands from the device's data sheet as well as code from Adafruit for properly calculating the heading value. Data is transmitted to the Tiva C in two bytes per axis from the magnetometer and then using bit shifts, unit conversions, and geometry, it is converted to a heading angle between 0 and 360.

Now that all of the software for directly interfacing with the sensors and other components has been defined, the practical functions can be discussed. We will begin with some simple navigation commands that must be taken into account. First, a function must be written to find the heading of the destination location with respect to the current location. This can be done using some simple geometry. Once this heading has been determined, software can then be written to turn the rover so that it is oriented towards its target. This is done by calculating which direction is closer to the target heading (left or right), and then issuing a turn command to the motors until the target heading is reached, to within 5%. Due to the nature of the motors, ultra-precise movement is not possible, so approximate movement must be used. The angle between the current heading and the heading of the target will be consistently monitored to keep the rover on course by readjusting as necessary. Another important function is calculating the distance between the rover's current location and the target location. This cannot be done by simple geometry, but rather requires the Haversine formula for calculating the distance between two points on a sphere. This will be used to monitor progress to the destination.

The last main aspect of the software is the obstacle avoidance code. This will mainly rely on the PING))). The idea behind this sub-routine is to actively monitor the space in front of the rover. When an obstacle is detected within 6 feet in front of the rover, the rover will immediately slow down to a stop and then it will scan the area in front of it to decide whether it can escape the obstacle more easily by navigating left or right. It will then perform that navigation, and realign with the target heading to finish the mission.

Aside from the control code for the individual components, the other major portion of the code is the setup and control code. As depicted in Figure 8 below, Once the rover is powered on, the first step is to initialize the subsystems, including the all of the sensors as well as the motor controller. Once that has been completed, the rover will wait until a target location has been received via a start command. At this point, the main control loop can be initiated. This loop will run until the point when the mission is completed. The rover will first enter the inTransitToLocation state. While in this state it will be constantly monitoring the space in front of it for obstacles to avoid as well as comparing its current location and heading with the location and heading of the target. Once the target location is reached, the rover will enter target object retrieval mode. Upon completion of obstacle retrieval, the rover will return to its home location in a matter very similar to the way it moves to the target location. Finally, during each iteration of the main control loop, the rover's telemetry data will be transmitted back to the CDC, if it is requested.

```
initializeAllSubsystems

while start command not received:
    wait

while state is not missionCompleted:
    switch (state):
        case inTransitToLocation:
            if obstacleDetected:
                avoidObstacle
            checkLocation
            if location reached:
                state = targetLocationReached
            correctRoverHeading
        case targetLocationReached:
            retrieveTargetObject
            state = returnToStart
        case returnToStart:
            if obstacleDetected:
                avoidObstacle
            checkLocation
            if location reached:
                state = missionCompleted
            correctRoverHeading
    transmitTelemetry
```

Fig. 8. Psuedo code for the rover navigation program.

VI. Conclusion

SARS represents the forefront of drone technology and autonomous computer systems. Using two separate autonomous systems linked with communication software, SARS proves that automated drone technology will continue to innovate new applications, both military and commercial. Each of the SARS subsystems presented a significant design challenge to the engineers, and the project offered valuable integration experience. Over the past eight months, each member of the SARS Group has learned much and been allowed the opportunity to improve a number of engineering skills.

Acknowledgements

The SARS Group would like to thank everyone that has contributed in the process of developing this system. Primarily, thanks must be extended to our sponsors, Boeing and SoarTech. A special thanks is given to the employees of SoarTech for providing funding as well as guidance and input throughout the development process. Friends and peers of the SARS Group have also been invaluable throughout the process by providing insight and assistance into the progression of this project.

References

"Reference | Energia." Energia Reference. Texas Instruments, n.d. Web. 09 Apr. 2015. <http://energia.nu/reference/>.

"Adafruit Ultimate GPS." Overview. Adafruit, n.d. Web. 09 Apr. 2015. <https://learn.adafruit.com/adafruit-ultimate-gps>.

"3-Axis Digital Compass IC HMC5883L." (2010): n. pag. HMC5883L Datasheet. Honeywell. Web. <http://www51.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/HMC5883L_3-Axis_Digital_Compass_IC.pdf>.

"DIY Quad Kit - 3drobotics.com." 3droboticscom. 3D Robotics Inc., n.d. Web. 09 Apr. 2015. <http://3drobotics.com/diy-quad-kit/>.

"Tiva C Series TM4C1294 Connected LaunchPad Evaluation Kit (Rev. A)." Tiva C Series TM4C1294 Connected LaunchPad Evaluation Kit User's Guide (2014): n. pag. Texas Instruments. Web. <http://www.ti.com/lit/ug/spmu365a/spmu365a.pdf>.

"CC3100 Simplelink™ Wi-Fi® Network Processor, Internet-Of-Things Solution For MCU Applications." Texas Instruments, n.d. Web. 09 Apr. 2015. <http://www.ti.com/lit/ds/symlink/cc3100.pdf/>.

The Engineers

Matthew Bahr is a Computer Engineering student at the University of Central Florida. He graduates in Spring 2015. He is an officer in UCF's Cuong Nhu club and the front man of a local band. After his graduation, Matt will be traveling to Southeast Asia for an undetermined length of time.

Brian Crabtree is a Computer Engineering major at the University of Central Florida. He graduates in Spring 2015. He is a member of the Burnett Honors College and an officer for the Men's Ultimate Frisbee Club at UCF. After graduation, Brian will be going to work for Intel as an SOC Design Engineer.

Brendan Hall is a senior computer engineer at the University of Central Florida. He is a member of Sigma Nu Fraternity and has worked as a CWEP at Lockheed Martin for over two years. After graduation, he will be joining Citi's Technology Leadership Program in Jacksonville, FL.

Erick Makris is a senior computer engineering student at the University of Central Florida. He is currently working at FLIR Systems as a Software Engineering intern, and will be begin a full-time position as a Software Developer at FLIR upon graduation.