# Designing, Building, and Testing an Autonomous Search and Rescue Robot — An Undergraduate Applied Research Experience

**Zachary Cody Hazelwood**

Cody Hazelwood is currently a software developer at the Alpha High Theft Solutions division of Checkpoint Systems. He received the B.S. degree in Professional Computer Science from Middle Tennessee State University in May 2013. He currently does freelance projects involving mobile software development, microcontroller applications, and electronics. He enjoys learning about and testing ways to improve people's lives with technology.

**Dr. Saleh M. Sbenaty, Middle Tennessee State University**

Saleh M. Sbenaty is currently a professor of engineering technology at Middle Tennessee State University. He received the B.S. degree in E.E. from Damascus University and the M.S. and Ph.D. degrees in E.E. from Tennessee Technological University. He is actively engaged in curriculum development for technological education. He has authored and co-authored several industry-based case studies. He is also conducting research in the area of mass spectrometry, power electronics, lasers, instrumentation, digital forensics, and microcontroller applications.

# Designing, Building, and Testing an Autonomous Search and Rescue Robot — An Undergraduate Applied Research Experience

## Preamble

Middle Tennessee State University Undergraduate Research Center, MTSU URC, was created in 2004 to promote research at the undergraduate level and to provide university support for undergraduate students and the faculty members who mentor them in scholarly and creative activities. This includes providing information and financial support through Undergraduate Research Experience and Creative Activity, URECA, grants. The URECA Committee evaluates proposals based on their merits. This committee is composed of accomplished and passionate faculty representatives from all five colleges at MTSU.

Universities usually do research as part of their missions (teaching, research, and service). As the institution with the largest undergraduate population in TN, MTSU is committed to being a leader in undergraduate education in the state. MTSU is known for student-centered learning and great classroom teaching. A natural extension of the classroom is the one-on-one interaction between a research student and his/her mentor that can shape a student's career.

### *URC Mission*

As part of the Office of Research, the URC mission is to be the central hub for communication about undergraduate research grant programs and other related opportunities on and off campus and to distribute university funds for undergraduate research and creative projects and travel to disseminate results.

### *URC Vision*

The URC is pursuing its vision to nurture a culture of research and creative activity through support for undergraduate students and their faculty mentors.

### *URC Values*

Implement the goals of the University's Academic Master Plan related to the URC mission with the following values:

- Excellence in research, scholarship, and creative projects.
- Opportunities for student-centered learning.
- Productive internal and external collaborations and partnerships.
- Success in academic and professional careers of our undergraduate students and their faculty mentors.

***Why Should MTSU Fund Undergraduate Research?***

Through URECA grants and other center activities, MTSU invests $120,000 in undergraduate research grants each year. Research involving undergraduates is a logical investment because it helps the university to:

- Maintain strong ties with alumni,
- generate a workforce of accomplished and sought-after graduates,
- build strong graduate programs,
- provide the extra challenge and preparation for high-end students going to top-notch graduate schools, and
- attracting the 'best and brightest' to our campus

We consider undergraduate research to be a signature program at MTSU

***Why Should an Undergraduate Student Do Research?***

Undergraduate students are encouraged to conduct research since this unique experience put them a cut above the rest when applying for jobs. This is true since undergraduate research helps students by:

- Integrating coursework through "hands-on" projects.
- Creating independence and autonomous researcher.
- Building Resume - writing a proposal, completing a research project, writing a final report, and orally present the results greatly enhance the student's experience.
- Preparing for graduate school, where a main goal is a research project.
- Developing "soft skills" important for entering into and succeeding in the job market.
- URECA grants allow students to build skills in their chosen fields without having to work an outside job.

The current paper describes the undergraduate research experience for an applied hands-on project and how this benefited the student.

## Introduction

Robotics is a relatively young field. One of the first demonstrations of Robotics as we know it occurred in 1898 when Nikola Tesla built a remote controlled boat and demonstrated it at Madison Square Garden. The *Unimate*, an industrial robotic arm, was first introduced at General Motors[1] in 1962. As robotics have improved, they have become very useful, especially in dangerous situations. They can enter into locations that humans could not without placing lives in danger. They can be strong enough to lift cars, or they can be accurate enough to perform delicate surgery. So why do we not use robotics more often? It is because robotics is an expensive field. For example, including development costs, the Northrop Grumman RQ-4 Global Hawk, currently used by the US Air Force and Navy, NASA, and the German Air Force,

costs $218 million dollars[2]. Despite these numbers, it is possible nowadays to develop a low-cost robot prototype that, with some minor physical improvements and up scaling of component quality, could be used and afforded by small-town law enforcement, rescue, and emergency management teams. The outcome of the project is important, because it demonstrates that the accessibility of robotics is increasing rapidly, and that it would be feasible for robots to be used, even on a budget.

Burning buildings, collapsed mines, and hostage situations all have one thing in common; they are dangerous for rescuers. Robots are, therefore, a preferred substitute! Robots can accomplish many tasks without requiring a human being to enter a dangerous location, thereby possibly saving lives. In the past, robots were not an option due to the high cost involved and the training required to operate them. However, with modern advancements in technology, robots are becoming increasingly used in situations and locations that were unthinkable in the past.

The objective of this paper is to develop an inexpensive, easy to operate, autonomous robot that is capable of navigating itself in dangerous situations. This would be a significant project, because it would demonstrate that using robots is not too far out of reach, even for local emergency crews and law enforcements. This project explores artificial intelligence as it relates to self-guided robots, microcontroller programming and code optimization, wireless video streaming, and remote control using a smartphone's or tablet accelerometer. Upon the project's completion, the plan is to develop a very simple to use robot capable of driving itself through a building while sending a video feed from a user-controllable camera back to the smartphone.

The strategy for implementing this project was to use and integrate the knowledge that the student had obtained from several courses such as Microprocessor Operation and Control, Intelligent Robot Systems, and Electronics along with open-source hardware and software libraries to develop a prototype robot. The project started by developing a very simple robot on which to build a software platform. The initial robot would have a remotely operative design and then move to the implementation of autonomy. One important issue to keep in mind when implementing autonomy was that there needed to be a proper balance between full autonomy and user control. In a search and rescue type environment, because of the unpredictability of an environment in a disaster situation, it is important that accurate manual control can be obtained if needed[3].

**Project Background**

Carnegie Mellon University is on the forefront of research for Search and Rescue robots. One of the most notable developments is the Snake Robot (also known as a hyper-redundant robot). The National Institute of Standards and Technology is also doing research on Search and Rescue robots. They have created a competition with three arenas of varying difficulties. Participants are required to navigate through the arenas to complete tasks without damaging the environment or the manikins. With careful data collection and observations of these competitions, they were able to explore the variety of robot implementations, the pros and cons of each implementation,

and human and robot interactions. Researchers are able to use the data collected to learn about and improve situational awareness, which is essential for robot autonomy.

According to the Springer Handbook of Robotics, there is a lot of work to be done in autonomous robotics for search and rescue. Due to the unpredictability of the environments encountered in disasters, better programming algorithms and better sensors need to be developed. Also, a proper balance needs to be struck between full autonomy and user control. Current search and rescue robots require hours of training for humans and animals.

This paper summarizes the development process of the robot. It covers the circuitry, hardware, and software components used (as well as some alternatives) and the reasoning for using those components. It also covers the results of this project, including reactions from the general populous upon presentation at the X X State University Scholar's Week poster presentations.

**Methodology**

1. Circuit Design

The circuit used for power distribution had to meet the following specifications: It should be powered by a single 7.2 volt 3300 mAh remote control car hobby battery; should provide two separately regulated power sources; providing up to one amp each; should be compact; and should be built on a custom printed circuit board. Based on these specifications, the 7805-voltage regulator was chosen.

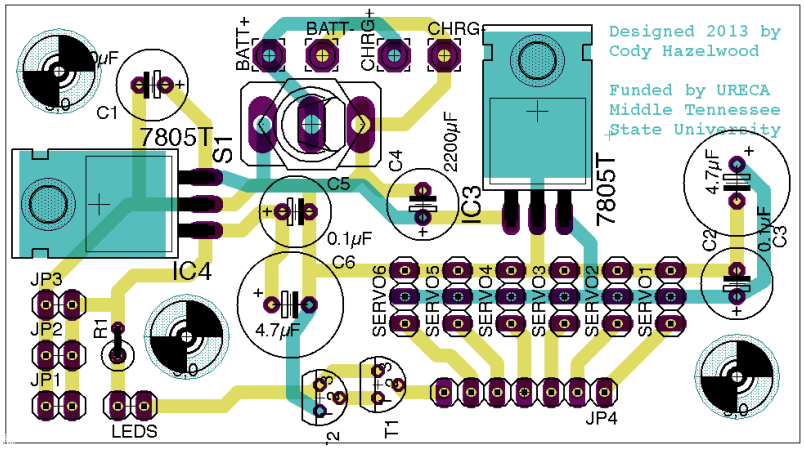



**Figure 1. Power Distribution Circuit Layout**

The 7805 is low-cost, and it is available locally at electronic and hobby stores. Along with a couple of capacitors, using two 7805s made it possible to design a single compact circuit board to power 5 servos, a microcomputer, a microcontroller, a camera, a light, and a handful of sensors. EAGLE CAD was used to design the schematic and PCB layout.

An assortment of hardware and software was needed for the robot. It needed a method of wireless communication, a device to host communication and delegate commands, a camera, sensors, motors, a body, wheels, and a device to control it. The robot was divided into several phases of functionality, and based on the requirements of those phases, appropriate hardware and software solutions were chosen.

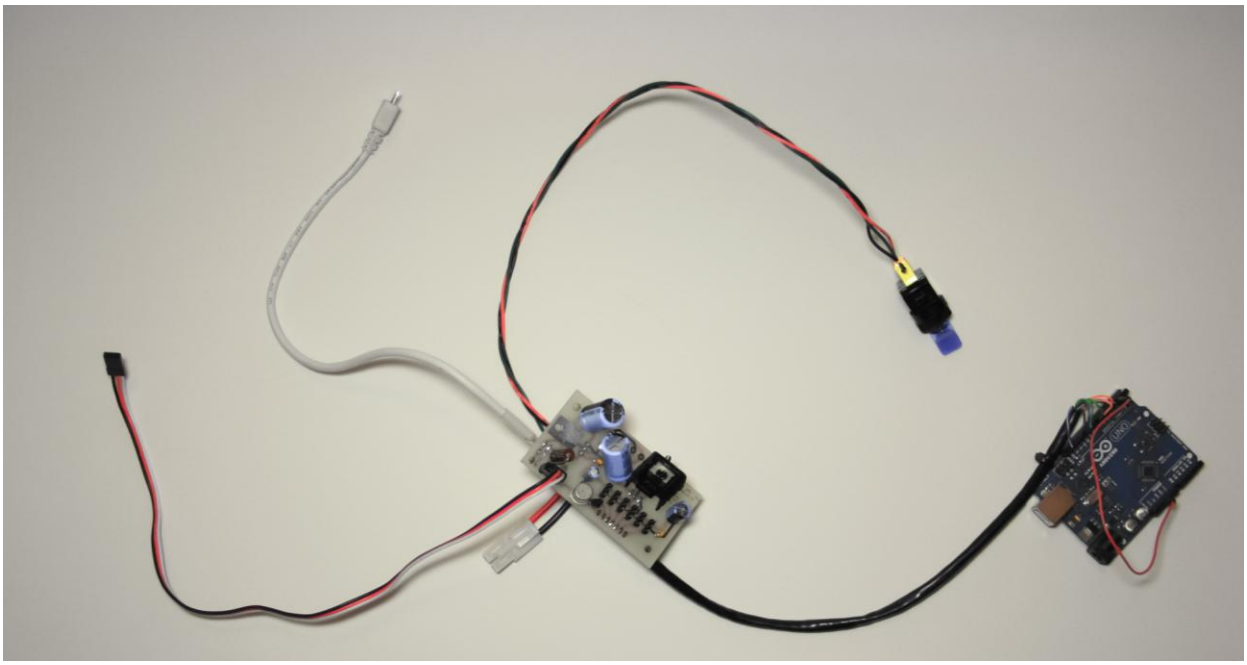


**Figure 2. Power Distribution and Microcontroller**

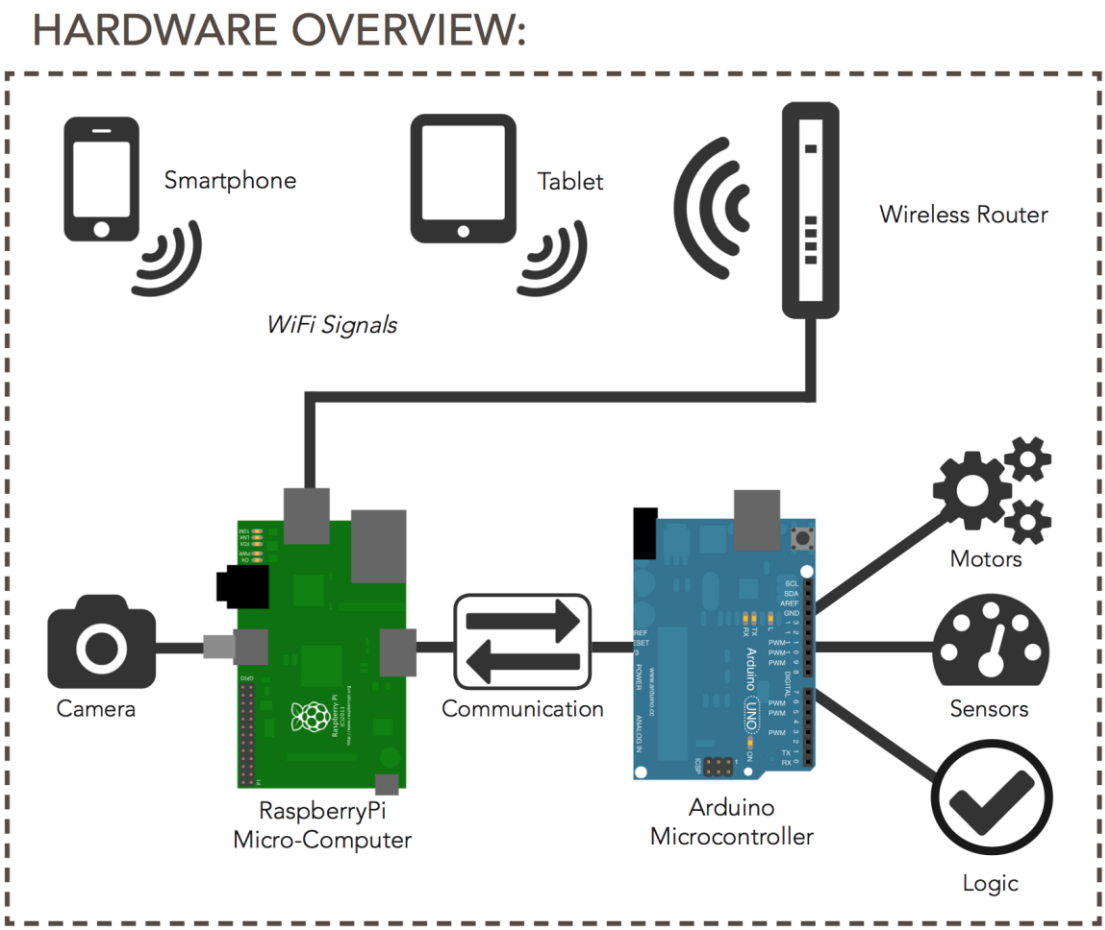




**Figure 4. TP-Link TL-WR703N**

**Figure 5. Logitech C310 – Image from Logitech.com**

## 2. Hardware Design

The first consideration for the hardware was the need for a high-quality video feed while keeping cost in mind. To tackle this issue, it was necessary to have a quality camera, high-speed data transfer, and a device to handle the video stream. For the camera, a standard Logitech C310 HD Webcam was chosen. The HD 310 is highly rated, very easy to obtain, cheap, and it provides acceptable video quality up to 1280 x 720 pixels with a JPEG-based video stream[4]. For wireless communication, the 802.11n wireless Ethernet protocol was chosen because of its wide support and large bandwidth. A TP-Link TL-WR703N was chosen to act as the central point of communication for the robot. The TL-WR703N is a miniature wireless router with a USB port, Ethernet port, and 802.11n Wi-Fi support. It is very "hackable" (easy to modify and install alternative firmware) and has large support in the hobby community. After flashing OpenWrt's alternative firmware[5] and installing MJPEG-Streamer[6], this wireless router is able to route network connections to the robot and handle all of the video streaming.

The next consideration for hardware was the need for a controller for motors and sensors. For this purpose, an Arduino Uno SMD microcontroller was chosen. The Arduino Uno is a very common open-source microcontroller platform based around an ATmega328 microcontroller. The ATmega328 provides 6 channels of PWM (pulse width modulation) output and 6 analog inputs along with several more digital input and outputs, which is plenty of I/O for a simplistic robot[6].

The next step in choosing hardware was to find a way to establish communication between the TP-Link wireless router and the Arduino Uno. There were a couple of options available for this with large differences in the pros and cons of each method. Adopting the appropriate method was essential, because this connection is the heart of all communication that occurs with the robot. The first option was to use an Arduino Ethernet shield. The Ethernet shield provides one

10/100 Mbit Ethernet port attached to a board that fits on top of the Arduino. This option would make networking very easy, but because of the speed of the Ethernet connection, it would use a lot of the Arduino's available processing power. The next option was to modify the TP-Link wireless router and use a connection to the built in serial circuit to directly route packetized serial messages to the Arduino's serial bus. This option was good, because it required no additional hardware. It would also have very low latency, because there is no additional device in the signal chain. However, it would not be expandable at all, and there is a risk of damage when modifying a multi-layer machine-soldered circuit board. The final option was to use a RaspberryPi to receive communications and pass them on to the Arduino. This is the option that was chosen, because it allows for expandability (such as adding computer vision for autonomy), and it allows more communications options than just packetized serial. The RaspberryPi Model-B provides a 700 MHz ARM11 CPU, a Broadcom VideoCore IV GPU with OpenGL support, and two USB 2.0 ports.[6]

**Figure 6. RaspberryPi Board**

Another important step in choosing hardware was to decide the body style of the robot and the method to be used for mobility. Since the goal of this project was to create a low-cost prototype, Polymethyl methacrylate (PMMA) (the proper name for what most people know as Plexiglas or Acrylite) was chosen. PMMA is easy to work with, and it provides plenty of strength for developing a simple robot. For movement, tank style treads were chosen. This will allow using only 2 motors for movement in any direction. Also, tank treads work well in unpredictable environments due to their large surface area and the teeth that normally characterize industrial tank tread design. In the case of this project, the treads were plastic. Standard Hitec HS-322HD hobby servos were used for tilting and panning of the camera. Two Hitec HS-425BB servos were chosen and modified to be continuously rotational for attaching to gears to drive the robot.

One additional step in choosing hardware was to determine the needed sensors for autonomy. Based on the original project specifications of a heat-seeking robot, a thermal sensor was necessary. Also, as is the case in any autonomous robot, sensors for obstacle avoidance were


**Figure 7. Parallax Ping))) Sensor**

**Figure 8. Sharp IR Sensor**

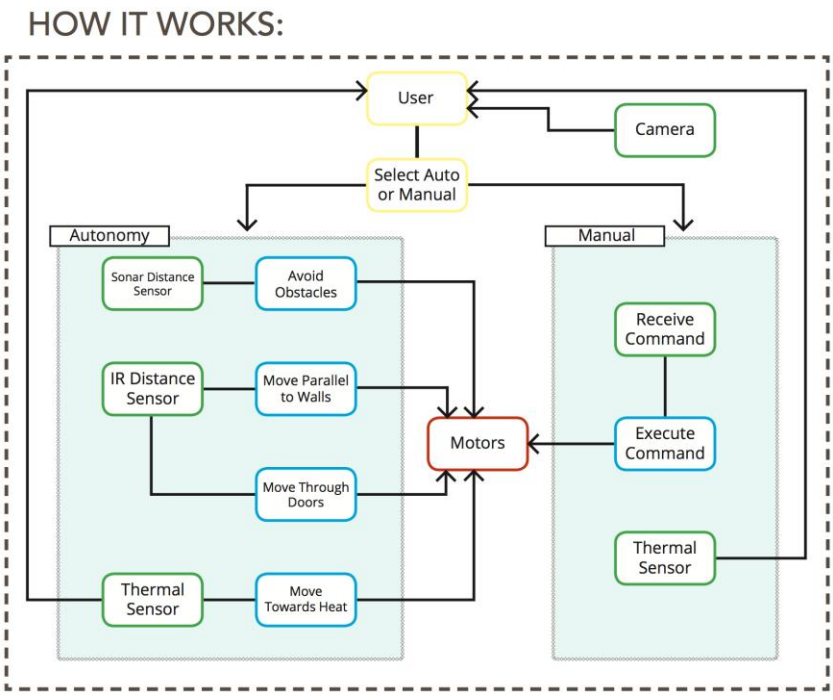
needed. For the temperature sensor, the Melexis MLX90614 Infrared Thermometer was chosen. This device is a medical grade temperature sensor that uses the 2-wire I2C protocol for communication. It can be programmed to have a temperature range of -70 ºC to 382.2 ºC[7].

For obstacle avoidance, multiple sensors were chosen. A Parallax Ping))) was chosen for the front of the robot. It mounts on a servo, so measurements can be taken in 180º. For the sides of the robot, Sharp GPD12 IR sensors were used. These can be used with minimal error as the robot is moving, so they work well for situational awareness such as centering the robot in hallways and doors.

The final step in choosing the hardware was to determine a method of remote control for the robot. Because of the desire to make control as user friendly as possible, an Android smartphone (Motorola Droid 3) and Android tablet (Asus Nexus 7) were chosen. By developing a remote control system on a mobile computing device, the learning curve for operation is decreased significantly as well as the cost of the project.

## 3. Software Design

The greatest challenge in designing a robotic system from the ground up is the software. For this project multiples layers of software were needed. It was important to keep the modules independent so that if one portion of the system changes, the entire software stack wouldn't have to be rewritten. The software consisted of modules for the microcontroller, the wireless router, the camera server, the communication and logic server, and for the Android device that would issue commands to the robot.




The first stage of software development consisted of developing a custom Android application to control the robot. This application needed to be easy to use, able to drive the robot using the accelerometer, able to issue commands to the robot, and able to view the camera feed. In addition, it helped to be able to observe the temperature readings obtained by the robot. In order to get some practice with Android

development, and to become familiar with the accelerometer API from the Android Software Development Kit, a simple test application was written to display accelerometer values to the screen. Once that was completed, the project was copied and modified to add some additional libraries. The Autobahn WebSocket library for Android was chosen for communication. It provided RFC 6455 WebSocket support, multi-threading, and an easy to use API.[8] Next, some code some code was added from a developer with a forum handle of 'padde' that provided a way to take an MJPEG stream and display it full-screen.[9] This code was heavily modified to remove features that were unnecessary, to clean up the coding style, and to increase the efficiency. It was also modified to allow a custom, resizable SurfaceView to be used, so that buttons to send commands to the robot could be displayed simultaneously while viewing the camera. Once these libraries were added, code was written to interpret accelerometer data and send it to the robot.

Once the Android application was complete, it was possible to work on the wireless router. The factory-installed software on the router was in Chinese, which made operation very difficult. It was also very limited in features. To overcome these limitations, OpenWRT was installed. OpenWRT is a custom Linux based firmware that allows heavy customization of the router's internal settings.[10] MJPEG Streamer, a software package that provides a simple web server and image streamer was also installed. This allowed the router to function as a video streaming server as well as a communications router.[11] The Linux install was customized to start streaming video as soon as it is powered on.

The next phase of software development involved the server (which runs on the RaspberryPi) and the microcontroller code (which runs on the Arduino). These pieces of software needed to be developed in tandem, because the logic for operating the robot was shared between the two components.

Before beginning development, a method of communication between the RaspberryPi and the Arduino needed to be chosen. The first method chosen was a library called Firmata. Firmata is an application that can be installed on the Arduino that allows nearly complete control of the board via serial. JohnnyFive, an asynchronous JavaScript library to interface with Firmata, was chosen for the server side. Tests worked extremely well for motor control and even turning an LED on and off fast enough to simulate PWM, however when adding in the time-sensitive Ping sensor, motors became jerky and the communications became unreliable.

The next method chosen was a system of short numeric codes that was created to allow very short serial messages to be sent. The server section of the code was based on the Node.JS platform. Node.JS was chosen, because it provides an asynchronous JavaScript based programming environment that is great for rapid development of scalable web applications.[12] By using web technologies, the application becomes extremely flexible, and very easy to develop cross-platform. Although this isn't technically a web application, it is important to have an

asynchronous application, because network connections and robot communications need to occur simultaneously. The Node.JS application used the Serialport-Node[13] library and the WebSocket-Node[14] library to receive JSON (JavaScript Object Notation) messages over a WebSocket network connection, parse and process the messages, and send them over serial to the Arduino. The Arduino software would receive messages, process these messages, and then send the appropriate messages to the motors. The Arduino was also responsible for sending temperature readings back to the client application. The Arduino stored states for motors and also handled obtained sensor readings when necessary.

| Code | Message | Code | Message |
|---|---|---|---|
| 1 | Stop All | 20 | Camera Left |
| 2 | LEDs On | 21 | Camera Right |
| 3 | LEDs Off | 22 | Camera Up |
| 4 | Autonomy On | 23 | Camera Down |
| 5 | Autonomy Off | 24 | Camera LR Stop |
| 10 | Drive Left | 25 | Camera UD Stop |
| 11 | Drive Right | 26 | Camera Stop |
| 12 | Drive Forward | 29 | Camera Dir |
| 13 | Drive Backward | | |
| 14 | Drive F Left | | |
| 15 | Drive F Right | | |
| 16 | Drive Stop | | |

**Table 1 – Numeric Message Codes**

For autonomous operation, the Arduino handled all logic. Communications were still maintained with the remote control client so the user can stop and start, reposition the camera, and take over manual control if needed. The autonomous system used the subsumption reactive robot architecture. The potential fields were used to process sensor information to avoid obstacles and move towards heat.

## 4. Project Assembly

Assembly was the last phase of the project. A frame for holding treads, a bottom panel, and a rear panel were designed using AutoCAD. The designs were uploaded to a laser cutter where the PMMA could be shaped to the design specifications.

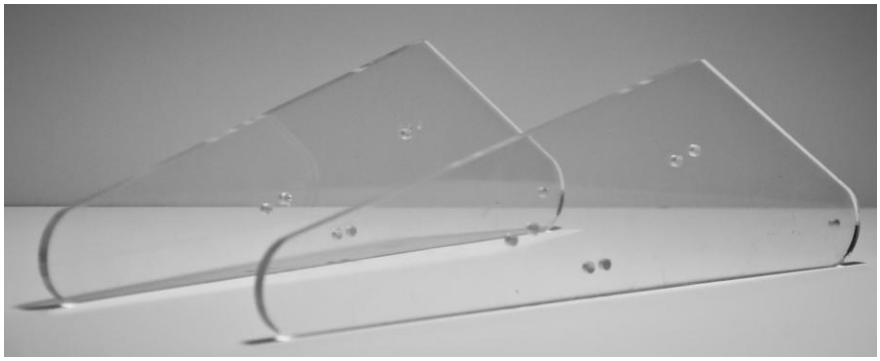


**Figure 8. Prototype Tread Frames**

Pieces of PVC pipe were used for mounting sensors that needed to be elevated as well as the camera and its pan and tilt servos. Once the pieces were cut, the frame was assembled using machine screws and a gel-based cyanoacrylate glue. Wires were laid out and cut to length. The power distribution board was populated and tested. Connections were soldered, and heat shrink was applied. Once everything was tested to be working, zip ties were used to keep the appearance of the wiring clean.

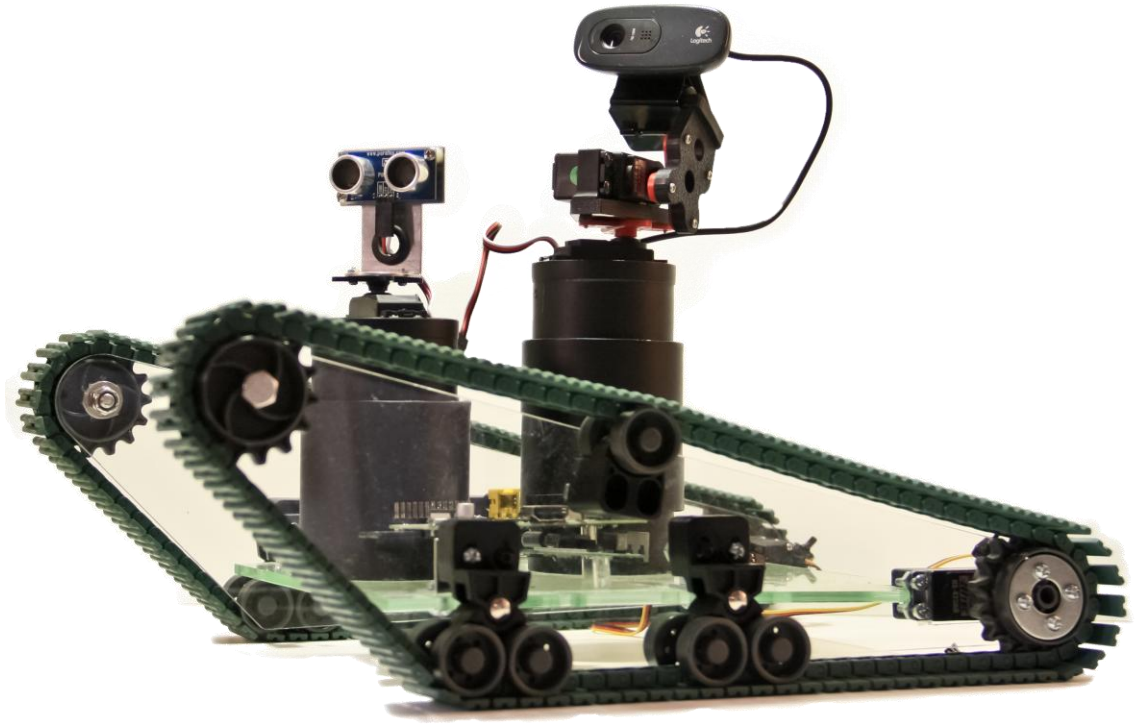


**Figure 9. Final Robot Assembly**

**Results**

Running at full power, the battery lasted about 45 minutes. Running without motors, the battery lasted about 3 hours. This demonstrates that the power distribution circuitry is fairly efficient and is good enough for usage in a real world application. The battery life could be improved by either using a larger battery or by using more efficient voltage regulators.

The hardware chosen fulfilled the project's specifications. The drive servos turned out to be slower than expected. One of them also had to be replaced during testing. The drive system could be greatly improved by using gears to drive the tank treads. Also it would be better if the servos were not supporting any weight from the robot. The wireless router was determined to be unnecessary, as a USB 802.11n Wi-Fi dongle could have been used to provide an Ad-Hoc network from the RaspberryPi and the MJPEG Streamer software is also compatible with the RaspberryPi. Since the router was already purchased, it was used anyway in the design. However, one advantage of using the router is that if something malfunctioned with the robot, video was still available. Also, keeping the video separate allows many people to view the video stream without compromising communication with the robot.

Software development accounted for a very large portion of the project's total time. A lot of research and trial and error went into the process of choosing various libraries and determining a

method of communication among all of the devices. In testing the software, the best-case time for the robot to respond to a message from the Android device was undetectable by the human eye. The worst-case time was still less than one second.

The first demonstrations of the robot's teleoperation occurred in small settings early in the project's development. Based on feedback from alpha-testers, accelerometer algorithms and camera control methods were modified until they were more comfortable to use. The result was very simple robot operation.

The robot was publicly demonstrated at the Middle Tennessee State University Scholar's Week Poster Presentations. The response from the public was highly positive. The MTSU UAV program as well as a forensic anthropologist that works for MTSU were interested in further discussions on ways the technology developed in this project could be used in their respective fields.

Further testing shows great success with the robot's teleoperation, however, there were many improvements that need to be made to the autonomy. Using computer vision to assist in tracking objects as well as implementing path planning algorithms would make the robot much more reliable. Also, modification of the heat sensor to allow it to focus better or adding a thermal imaging camera to be processed with computer vision would allow a much better tracking of heat.

**Conclusion**

This project was a very large undertaking for a first research project. Although the teleoperation portion of the project was very successful, the fully autonomous operation could not be completed in the URECA project timeline of 200 hours. Despite this shortcoming, the outcome of this project was still a success. This project proves that developing a robot for search and rescue that is within reach of local emergency crews is something that could occur in the very near future. With some minor hardware changes, an upgraded chassis, and additional software development time, a usable product is not far off from the prototype developed in this project.

**Appendix A: Project Timeline**

## References


1. Isom, James. "MegaGiant Robotics." *MegaGiant Robotics*. 2002. Web. 20 Apr. 2013.
2. Drew, Christopher. "Costly Drone Is Poised To Replace U-2 Spy Plane." *The New York Times*. The New York Times Media Group, 2 Aug. 2011. Web. 20 Apr. 2013.
3. Wolf, Alon, Howard H. Choset, Benjamin H. Brown, and Randall W. Casciola. "Design and Control of a Mobile Hyper-redundant Urban Search and Rescue Robot." *Advanced Robotics* 19.3 (2005): 221-48. Print.
4. Logitech. "Logitech HD Webcam C310." *Logitech*. Web. 20 Apr. 2013.
5. "Arduino Uno." *Arduino*. Arduino. Web. 21 Apr. 2013. <http://arduino.cc/en/Main/ArduinoBoardUno>.
6. "RPi Hardware." *ELinux.org RaspberryPi Wiki*. Web. 22 Apr. 2013. <http://elinux.org/RPi_Hardware>.
7. Melexis. "Melexis MLX90614 Datasheet." Microelectronic Integrated Systems, 30 Mar. 2009. Web. 22 Apr. 2013. <https://www.sparkfun.com/datasheets/Sensors/Temperature/SEN-09570-datasheet-3901090614M005.pdf>.
8. "Autobahn|Android." *Autobahn*. Tavendo. Web. 22 Apr. 2013. <http://autobahn.ws/android>.
9. Padde. "MJPEG on Android Anyone?" *Anddev.org*. Web. 22 Apr. 2013. <http://www.anddev.org/mjpeg_on_android_anyone-t1871.html>.
10. "TP-Link TL-WR703N." *OpenWrt*. Web. 22 Apr. 2013. <http://wiki.openwrt.org/ru/toh/tp-link/tl-wr703n>.
11. "Mjpeg Streamer." *Sourceforge*. 9 Apr. 2010. Web. 22 Apr. 2013. <http://sourceforge.net/apps/mediawiki/mjpg-streamer/index.php?title=Main_Page>.
12. "Node.js." *Node.js*. Joyent. Web. 21 Apr. 2013. <http://www.nodejs.org/>.
13. Voodootikigod. "Node-serialport." *GitHub Node-serialport*. 24 Aug. 2012. Web. 21 Apr. 2013. <https://github.com/voodootikigod/node-serialport>.
14. Worlize. "Websocket-Node." *GitHub*. Jan. 2013. Web. 21 Apr. 2013. <https://github.com/Worlize/WebSocket-Node>.