Table of Contents

1. Introduction

ARES: Autonomous Robotic Engagement Sentry is a robot of the land vehicle variety that is geared towards military applications. This technical report details the research, design, development and testing that went into creating ARES to perform that specific purpose.

I decided on this project because it relates to two of my greatest interests: artificial intelligence and military technology. An autonomous robot that can engage a target suits these two topics very well. Also, after some research, I realized that the project was actually feasible so I decided that this was the project. In addition, I wanted to do a project that would be substantial and would keep me busy and interested.

This project in and of itself is not important to the world at large. The concept of the project is though. Similar machines already perform similar functions to what ARES is designed to do: consider the Predator Unmanned Aerial Vehicle. It is important to me because it provides insight into exactly what goes on in robotics research and research in general. I do get asked a lot about “Terminator” and “The Matrix” though.

From the beginning, I wanted to achieve a working sentry and actually score a hit on a moving human controlled target that would be a very rough representation of what could be expected in a real-world live environment. I ultimately achieved this goal. The following sections will detail the research, design, construction, development, implementation and testing that went into the creation of ARES.

2. Overview of Hardware

ARES is a hybrid system consisting of and constructed with components from several companies including Vex Robotics, Pololu Robotics and Electronics, airsplat.com and others. ARES's target is a remote controlled vehicle manufactured by New Bright.

2.1. Vex System Components

The Vex system components consist of all parts purchased from Vex Robotics. These components are divided into five categories in relation to their functionality: control, motion, power, sensors and structure.

2.1.1. Control

These components play a part in controlling motion, interpreting sensor data or sending/receiving signals. Included in this group are the Cortex Microcontroller (CM), VEXnet Remote Controller(VEXRC), VEXnet USB wireless sticks, Male-Male USB cable and Serial-USB cable.

1. The Cortex Microcontroller (CM) is the "brain" of ARES. It features eight 3-wire motor ports, two 2-wire motor ports, one I2C "smart sensor" port, two UART serial ports, eight 12-bit analog inputs and twelve digital I/O ports. Of these, ARES uses all eight 3-wire motor ports, one 12-bit analog input and six digital inputs. It is powered by a 7.2V battery and can be connected to a computer via a male-male USB cable or through a VEXnet wireless connection.

2. The VEXnet Remote Controller (VEXRC) allows for user operation of ARES. It features two 4-button d-pads, two analog joystick and four shoulder buttons. It is powered by 6 AAA and also features competition, program, partner and USB ports for connection to a computer or another VEXRC.
3. The VEXnet USB sticks allow for a wireless connection between the CM and the VEXRC. Two are required for operation: one plugs into the VEXRC and the other plugs into the CM.
4. The Male-Male USB cable allows for a wired connection between the CM or the VEXRC and a computer. It is necessary for initial setup and can be used to download programs to the CM.
5. The Serial-USB cable allows for a wired connection between the VEXRC and a computer. This connection is necessary for downloading programs wirelessly via VEXnet.

2.1.2. Motion

This category includes two pieces of hardware, motors and servos, which give ARES the ability to move.

1. The motors allow for continuous motion to occur on some structural component and operate on a speed value. ARES uses five motors.
2. The servos allow for motion to a set position and operate on a position value. ARES uses two servos.

2.1.3. Power

These components provide power to Vex sensors, motors and servos through the CM. A 7.2V 2000 mAh provides this power.

2.1.4. Sensors

These components give ARES (or its operator) the ability to make decisions based on its environment. These include a sonar range finder, potentiometer, wireless camera and Pololu's Infrared Sensor (See Section 2.2.).

1. The sonar range finder emits an ultrasonic wave. This wave will either bounce off a surface and return to the sonar range finder, at which point the CM will be able to tell the distance to that surface, or it will not return.
2. The potentiometer calculates the position of a shaft over 360 degrees. In ARES, a potentiometer is used to measure the angular position of the turret to ensure that it does not over-rotate the reach of the wires on its structure.
3. The wireless camera gives ARES's operator an added ability to remotely control it. However, ARES has no interface to this camera; it only provides power to it.

2.1.5. Structure

This category consists of parts that give ARES its form. This list includes aluminum structure pieces, wheels, drive shafts and a multitude of connection pieces (screws, washers, etc.).

2.2. Pololu System Components

Pololu manufactures the Infrared (IR) sensors that ARES uses to detect its target. They come in pairs with each pair having the ability to detect the other. The set ships partly unassembled, consisting of two sensor boards, eight IR sensors, two 4-pronged connectors and two 3-pronged connectors and must be soldered together prior to use. They are each powered by a 9V battery.

2.3. Airsoft System Components

ARES features a modified UHC MP5 A5 mini electric Airsoft Gun. It has been modified to work in conjunction with the Vex system components.

2.4. Remote Controlled Car (Enemy Target)

This is a 1:16 scale blue Dodge Ram manufactured by New Bright and includes a forward/reverse and left/right turning remote controller. It has been modified by placing a cardboard tower on its bed on which the IR sensor sits and is protected.

2.5. Miscellaneous Hardware

This category includes hardware used in the construction of other system components and includes wire, tape, super glue, Velcro and tools such as a hacksaw, allen wrenches, screwdrivers, clamps and a soldering iron.

3. Overview of Software

The language used in programming the CM is RobotC, developed by the Carnegie Mellon Robotics Institute. It is a C-based language including features such as I/O, methods, structs and concurrently running tasks, among others. Specifically, its libraries include all of the code unique to RobotC: the code which controls the CM, sensors and all moving parts.

The development environment provides a simple and easy to use GUI. The usual facilities, such as file creation, exist along with other helpful features. Most importantly, the environment includes sample programs which provide new RobotC users with a means to jump right in and learn the language. Also, it includes a powerful debugger that not only detects problems in the code but allows for live feedback from the CM.

4. Hardware Design and Challenges

Constructing and combining the hardware was done in roughly four major sections. Initially, the base and turret were constructed. Next, the Airsoft gun was analyzed, modified and integrated with the turret and the IR sensors were constructed and tested against each other. Finally, the remote controlled target car was modified.

4.1. Structure Construction

Initial construction began on a prototype called "Tumbler", the guidelines for which were included with the Vex Robotics kit. This step served to understand the parts and get an initial idea of exactly what lay ahead. It was straightforward and simple, requiring only a couple hours to complete. However, this prototype was clearly too small to act as a base, so it was modified to increase its length. This modification provided more space for necessary parts, as well as helping to balance ARES since the Airsoft gun was longer than the prototype. Finally, the CM was placed slightly forward of the middle of the base and the battery was attached to the base in front of it with Velcro. Next, the turret was constructed.

Since the major component of the turret was the Airsoft gun, it was the primary consideration in deciding exactly where and how to build the turret. The initial design proved effective but incomplete. ARES was modified with a crossbar across that base positioned slightly more than a quarter of the length from the rear. A motor was mounted under this crossbar with the motor approximately 1-2 inches from the base. Finally, a shaft was connected to the motor and an aluminum structure piece approximately 3 inches by 7 inches

was attached to the shaft. Unfortunately, this design had two flaws. The turret tended to wobble a good amount during initial testing and, most importantly, it did not provide a position where the potentiometer could be mounted.

The second and final design proved complete. The motor was moved from under the high crossbar and placed under a lower crossbar positioned under the high crossbar. This new motor position decreased the wobble of the turret and provided a structure point on which to mount the potentiometer.

4.2. Airsoft Gun Analysis, Modification and Integration

The Airsoft gun is a simple device to use. Simply insert the batteries in the handle, fill the top slot with BBs, pull down on the safety and pull the trigger. Immediately, it was clear that the safety was unnecessary and would only add complexity to the design. The internal components of the gun are a slightly more complicated. Pushing down on the safety removes a lock that prevents the trigger from sliding along its path. Pulling the trigger connects a switch between the batteries and an electric motor. The electric motor turns a set of gears that compress a spring at the end of the barrel and a BB falls into the barrel. The spring is released and generates the force to propel the BB out of the barrel.

The initial modification involved cutting off excess plastic that served no purpose except to cause an obstruction. A test mounting on the turret proved that the gun was obscenely unbalanced due to the battery compartment in the handle, which was sawed off. Finally, the firing switch and trigger were removed from the casing, and the trigger was discarded since it is a piece

separate from the switch. Another test mounting was performed and this time the gun remained balanced. Unfortunately, the separated components of the gun had to be reconnected, which involved locating a suitable position on the base for the battery compartment and the switch.

The final step involved integrating the gun components with the already constructed base and turret. To begin, the approximate placements of the battery compartment and firing switch on the base were determined. The gun components were then reconnected by soldering wires of the appropriate length to the correct positions on each of the components. The mechanical parts (that fire the BB) are all placed on the turret. The battery compartment is placed on the base near the back right wheel. It faces backwards so the batteries can be easily removed or inserted. Finally, the switch was placed near the back left wheel. A servo mechanism is used to actually move the switch between the “on” and “off” positions.

One small problem that happened a few times in regard to the separated components of the gun is the newly soldered connections came loose. The problem was not difficult to solve: the loose pieces were just soldered back on. A connection to the electric motor did come loose twice. In this case, the gun had to be removed from the turret and taken apart.

4.3. IR Sensor Construction, Hardware Testing and Integration

The IR sensors were shipped partially unassembled, so it was necessary to solder the included pieces onto the circuit board. The soldering was not

technically complex. However, the parts being soldered were quite small so that presented a level of difficulty. Eventually, both sensors were constructed.

After construction was completed, both sensors were tested against each other. This step required connection of two 9V batteries, one to each of the sensors and confirmed the success of the previous step. One problem did come up at this point: both sensors seemed to have a more "powerful" north detector, a phenomenon which is explained by how close the four detectors on each sensor are placed together. Ultimately, this problem was fixed by limiting the north sensor's field of view (See User's Guide Appendix-B: Infrared Sensor View Limiter).

Finally, one sensor was placed on ARES at its highest point on the turret which provided an unobstructed field of detection and allowed for all modes to function properly. Physically connecting the IR sensor to the CM proved more difficult. Each of the four sensors connected to the board transmits either a "0" or "1" to the CM with a "0" being detected and "1" being not detected. As this is a digital reading, a digital input port must be used for each of the four sensors, but the CM features 3-prong connectors for these ports; the IR sensor only transmits on a 1-prong connector. Since the pin layout was not included in the documentation, research was conducted into exactly how to connect a 1-prong connector to a 3-prong input. A pin layout guide of the CM indicated that input connections must be on the inner most prong.

4.4. Remote Controlled Target Car Modification

This step involved placing the second IR sensor on the remote controlled target car in a manner that both protected the sensor and allowed for it to emit IR signals. This optimal position is higher than the Airsoft gun's aim but lower than ARES's IR sensor: too low and it wasn't protected; too high and it would be difficult for ARES to detect it. A cardboard tower was constructed in the bed of the truck that satisfied both of the requirements. The IR sensor's emitters slightly protruded over the edge of the top of the tower, and the 9V battery was placed inside the tower.

5. Software Design and Challenges

Initial research into RobotC and its functionalities began by looking at the sample programs included with the development environment. Many sample programs are included and help with understanding how the CM interacts with all types of Vex motors and sensors. Also, the development environment includes a feature that allows for easy setup of all motors, sensors and ports.

The feature most unique to RobotC (compared to C, C++, Java and other general purpose languages) is the task construct. First, main() (which is also a task) must be included or the compiler will throw an error. However, it must be compiled last; if it is not compiled first, any startTask calls it makes to fork any child tasks will throw an error. The difference between a task and a method (or function) deals with sequencing. Where a method occurs sequentially with reference to the rest of the program, a task occurs concurrently with other tasks based on priority. This is quite important for ARES which must make decisions based on concurrent readings from various sensors and the values of certain variables.

Since tasks are separate processes, a standard object-oriented approach severely limits the capabilities of RobotC; in fact, RobotC does not allow for pointers. Instead, global variables (primitives, structs and enumerated types) must be used since they are the only method of communicating among multiple tasks. However, the use of global variables does require careful and planned implementation so ripple effects are minimized.

5.1. Testing of All Vex Sensors and Motors

As ARES was constructed, all motors, servos and sensors were tested to make sure they worked and to get an understanding of exactly how they worked in conjunction with the code.

5.1.1. Four Wheel Drive Testing

The first motors tested were the four that composed the four-wheel drive aspect of ARES. This step involved understanding the range of speeds involved and how to get the four motors to work together. Each motor can turn either clockwise or counterclockwise, so for each motor it was necessary to discover which direction moved the wheel forward and which direction moved it backward. The motors were set up such that 3 of the 4 had to be "reversed" in the code so that positive values for speed moved ARES forward and negative values moved it backwards.

5.1.2. Turret Testing

This step involved testing the motor responsible for rotating the turret, the servo responsible for panning the gun and the servo responsible for firing the gun. It required three steps (See Section 5.1.4. Potentiometer Testing for step three).

First, both the motor and servos were tested to make sure they worked, to discover which values (positive or negative) in the code determined the directions of rotation, panning and firing and exactly how fast the turret would rotate. This step was performed before either the gun or potentiometer were integrated into the turret.

Second, the gun was properly integrated into ARES and all three motion components were tested again to obtain the limits imposed on the turret motor and panning servo by the firing mechanisms and to identify the firing servo's position necessary to move the firing switch from "off" to "on" and vice versa. More about the limitations will be discussed in step three.

5.1.3. Sonar Sensor Testing

Sonar sensor testing proved to be one of the most problematic steps of the entire project. At first, no problems were evident. Initial testing indicated the sensor worked. However, later testing revealed three important problems.

First, ARES would randomly back up (meaning the sonar sensor had detected an obstacle within a specified range) but clearly no obstacle was present. After further testing, it was realized that the cause of this erratic behavior could be attributed to the vibrations caused when the Airsoft gun fired. It seems these vibrations caused the erroneous readings by vibrating components within the sonar sensor or possibly even emitting the same frequency on which the sensor operated. Correcting the problem involved moving rewriting the code so that the sonar sensor and gun never operated at the same point. This solution worked soundly but did require a large amount of code modification.

Problem two is indirectly related to problem one. During the course of identifying and correcting problem one, the sonar sensor went unused but

remained in position on the front of ARES. Unfortunately, due to developer misjudgment, ARES made head on contact with walls on several occasions; the sensor did not survive these impacts. A new sensor was purchased.

Problem three also caused the same result as problem one: ARES would randomly back up. However, problem one had already been accounted for and corrected. This new problem was caused by the newly purchased sonar sensor: it seemed that internal components were aimed at a slightly downward angle, causing ARES to randomly detect ultrasound rebounding from the floor. This problem required two modifications. First, the sensor was placed higher on the front of ARES. Second, a new detection algorithm had to be developed. Instead of relying on one detection at some minimum range to avoid hitting an obstacle, the code was modified such that ARES is now required to make two sequential detections at some range a few inches greater than the minimum implied detection range.

5.1.4. Potentiometer Testing

The potentiometer did not require much testing once it was mounted, just making sure it worked and discovering how it worked. Testing focused on discovering roughly the “centered” position for the turret and determining the limits of both clockwise and counterclockwise rotation due to the constraints imposed by wiring around the turret.

5.2. Testing of IR sensor

The IR sensor is not a Vex component so there are no "built in" features of the language that allow for easy connection to the CM. However, the software does allow for digital inputs and outputs of any nature as long as they conform to the voltage setting (5V) on which the CM operates. The first task, after properly connecting the sensor to the CM, was to determine how to get useful information from it. This proved to be easier than expected. Since the development environment allows for real-time variable tracking, determining the correct reading was a matter of assigning a variable to each input of the four sensors. The sensor closest to its counterpart registered a "0" while the others maintained a "1".

Getting the sensor to perform its assigned task was the most difficult problem of the entire project. Adjusting ARES in real-time based on the current IR sensor reading proved terribly ineffective due to the sensor itself making approximately twenty readings per second. Occasionally, a misread would occur, causing ARES to temporarily head in the wrong direction. This occurred often enough that ARES seemed to act confused. To solve this problem and after testing various numbers of readings, ARES now takes ten readings and uses the mode of the readings to determine in which direction to head; anymore and ARES responds too slowly, any less and ARES displays the "confused" behavior.

One last problem remained: overcoming the "north" sensor's seemingly more "powerful" field of detection. As discussed in section 4.3., this was

accomplished by use of the infrared sensor view limiter. The inclusion of this piece of hardware drastically increased the pursuit accuracy of ARES and resulted in the greatest leap of progress over the entire project. Now, ARES only heads north if it is roughly (give or take a few degrees increasing with distance to the target) head on with its target. Essentially, this modification puts more emphasis on the "east" and "west" sensors, where previously the emphasis was placed on the "north" sensor.

5.3. Testing of VEXnet and Remote Control Functions

Remote Controlled Mode was the easiest of the three modes to implement but it did require a new understanding of how VEXnet and the remote controller worked. Learning the code for this part was very simple and involved reading input from the controller. The buttons issue a digital reading while the joysticks issue an analog reading, appropriate for variable speed adjustments. Also, by the time this mode was implemented, all sensors and motors had been tested thoroughly so the implementation was almost as simple as "copy and paste".

6. Conclusion and Final Thoughts

Final, live testing has been conducted for both the Mobile Mode and Remote Controlled Mode. As of writing this report, Stationary Mode is ready for a live test but due to injuries, sickness and other obligations I have been unable to perform such a test. By my presentation day, a final test will have been performed and a video demonstrating its capabilities will be available. The one problem I have run into during testing is determining exactly when a hit occurs. A hit issues a light plastic or cardboard thud sound but is difficult to hear over the sound of the gun firing. I did hear a hit for approximately 5-10% of all shots, but a hit is indistinguishable in the videos.

This project was a great learning experience and offered a unique approach to my Computer Science curriculum. I obtained firsthand experience on the limitations of current robotics through my work and research. I became familiar with some features of C which is important due to C's ubiquity. In addition, I performed some electrical and mechanical work, including soldering wires and the Airsoft gun modifications, outside the confines of a simple Vex Robotics kit.

I achieved a greater understanding of Robotics through this project and gained insight into the process of formal research. I have a newfound respect for those who work with robots in real-world applications because ARES is no more than a toy (though it can be dangerous) but still presented a high degree of difficulty.

For follow up work, I would consider increasing the autonomy of ARES. Ultimately, any robotic application is limited by both its computer, mechanical and sensing hardware. Understanding RobotC was easy; combining it with the hardware proved much more of a challenge. If I had unlimited resources, I would like to one day

see soldiers on the battlefield replaced by robotic surrogates, under remote control of those same soldiers.

If I could redo this project, I would make two changes. First, I wouldn't break my ankle and lose two weeks of work. On a more serious note, the only reasonable change I would make would be to connect the four directional sensors to extension wires, allowing for greater flexibility in their usage. Now, they are soldered directly to the board. Unreasonably, i.e. outside of my budget and resources, I would purchase hardware that would allow for true real-world testing, similar to robots used in the show "Battle Bots". In the future, this may be a possibility but I will have to wait and see.

7. Sources and Support

http://www.vexrobotics.com/ and its forums and documentation

http://www.pololu.com/ and its forums and documentation

http://www.airsplat.com/

http://www.robotc.net/ and its forums and documentation