NetGrok II

Project Report

XE401 Mini Design Project

Authors:
Tyler Reece
Daniel Young
Matthew Kim
Joshua Balba

Advisor:
Mr. Kyle King
Instructor:
LTC David Harvie

September 21, 2018

Contents

1 Problem Definition

1.1 Introduction

The use of autonomous ground vehicles (AGVs) in recent military operations is well documented, with use of such robotics in situations such as explosive ordnance disposal, unmanned vehicle convoys, and aerial and waterborne drones. This project investigates the viability, capabilities, and limitations of a prototype AGV using widely available, commercial hardware powered by the popular Arduino open source software. In particular, this project attempts to prototype a four wheeled AGV capable of moving through a course with obstacles and walls and navigating toward a light source in a darkened room. This project allowed the NetGrok II team to practice design concepts, learn to manage workflow and meet deadlines, as well as provide a challenging team building exercise.

1.2 Problem Statement

Design an AGV that is capable of navigating within a delimited area to a marked destination in less than 5 minutes. The AGV must be built from an MMP-5 Mobile Robot Platform using an Arduino microcontroller board with a Parallax BoE shield. The AGV can use a number of provided Parallax sensors (photoresistor light sensors, "whisker" touch sensors, IR detectors, and PING ultrasonic sensors, as well as others (pending instructor approval). The test area that the AGV must navigate is bounded by both real and virtual walls (indicated by black tape). The robot must navigate through the course and enter a darkened room, ultimately navigating towards a light in the room.

The AGV will be given two attempts to complete the course, and while it may not be reprogrammed between runs, the AGV may use information gathered in one run to improve the next run. The AGV design will be judged on its ability to navigate obstacles and move towards the light source within the allotted time.

1.3 Product Vision

NetGrok II's AGV will utilize the PING and IR sensors in order to autonomously navigate the test course and move towards the light source within the allotted 5 minutes. The AGV will utilize the "right hand rule," tracing the furthest right physical wall as a way to stay on course. The AGV will build off of open source Arduino software, including a custom-built library of movement and detection functions, in order to manipulate the robot and complete the course within the allotted time.

2 Requirements Analysis Phase

2.1 Specified Requirements

1. Design an AGV capable of navigating within a delimited area to a marked destination in less than 5 minutes
2. Build the AGV from an MMP-5 Mobile Robot Platform with an Arduino microcontroller board with a Parallax BoE Shield
3. Use only the provided Parallax standard sensors (photo-resistor light sensors, "whisker" touch sensors, IR detectors, and PING ultrasonic sensors
4. Ensure the AGV can navigate both real (physical) and virtual (black tape) walls
5. Ensure the AGV can find and enter a dark room and navigate toward a light source
6. Create functional block diagram to illustrate utilization of ports and sensors
7. Complete requirements for IPR on 06 SEP and present to instructor and evaluator
8. Complete requirements for Written Report and turn in along with Peer Evaluations on 21 SEP

2.2 Implied Requirements

1. Conduct rigorous testing to identify necessary sensors for AGV

2. Conduct calibration testing to ensure successful manipulation of robot
3. Create custom movement library for easier manipulation of robot
4. Request additional sensors and other hardware, if necessary
5. Create and test high level algorithm for autonomous navigation
6. Illustrate sensor test results in IPR and Written Report
7. Archive all code on Git repository located on GitHub (forked from EECS XE401)

3 System Design Phase

3.1 Design Methodology

The design methodology for this project stressed starting as small as possible. The design started from a very simple point and slowly improved. Functionality was incrementally added, and both unit and comprehensive tests were conducted after every addition to ensure that no functionality was broken between additions. Our approach heavily utilized the concepts of functional decomposition to create modular building blocks, adding complexity by combining only working, tested modules. Layers of abstraction were added through a custom built library, making the final, high-level loop much easier to work with. Throughout the project, tests were repeated continually, in the most realistic environment possible.

Our design methodology was helpful throughout the building phase of the project. When tests failed, we were able to quickly identify the exact function that caused the problem, and tweak only that function before rerunning tests to see the robot's improvement. We were able to continuously iterate and develop, adding only small amounts of code each time before testing. In this way, the link between our development and testing efforts was very strong, a practice that we hope to continue to utilize in future projects.

4 Detailed Design

NetGrok II's AGV primarily utilized a pair of PING sensors and a photoresistor to navigate through the course. While the original

concept design included IR transmitters and detectors to navigate virtual (black tape) walls, the team ultimately made the decision to remove IR equipment from their design due to an inability to gather reliable, consistent data that could inform the high level algorithm from the sensors. Instead, the AGV utilized a forward PING sensor, to avoid running into walls or obstacles in the front, as well as a right PING sensor, in order to allow the high level algorithm to utilize the "right hand rule" and trace the rightmost wall during navigation.

4.1 Functional Block Diagram

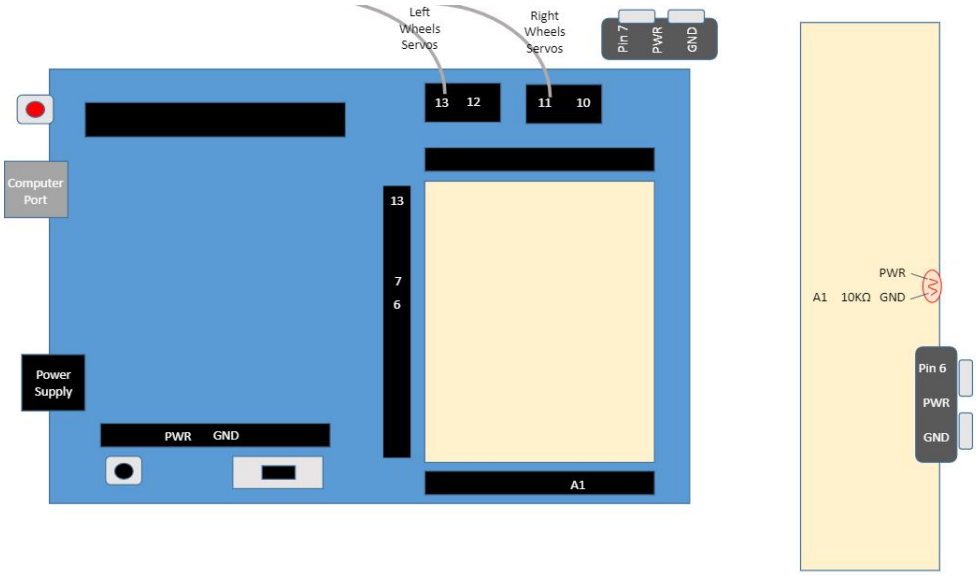

Figure 1: Functional Block Diagram

The functional block diagram was created as a way to visualize the hardware components and wiring setup of the AGV. This allows other groups to easily recreate our design for further testing and improvement. The functional block diagram can be seen in Figure 1.

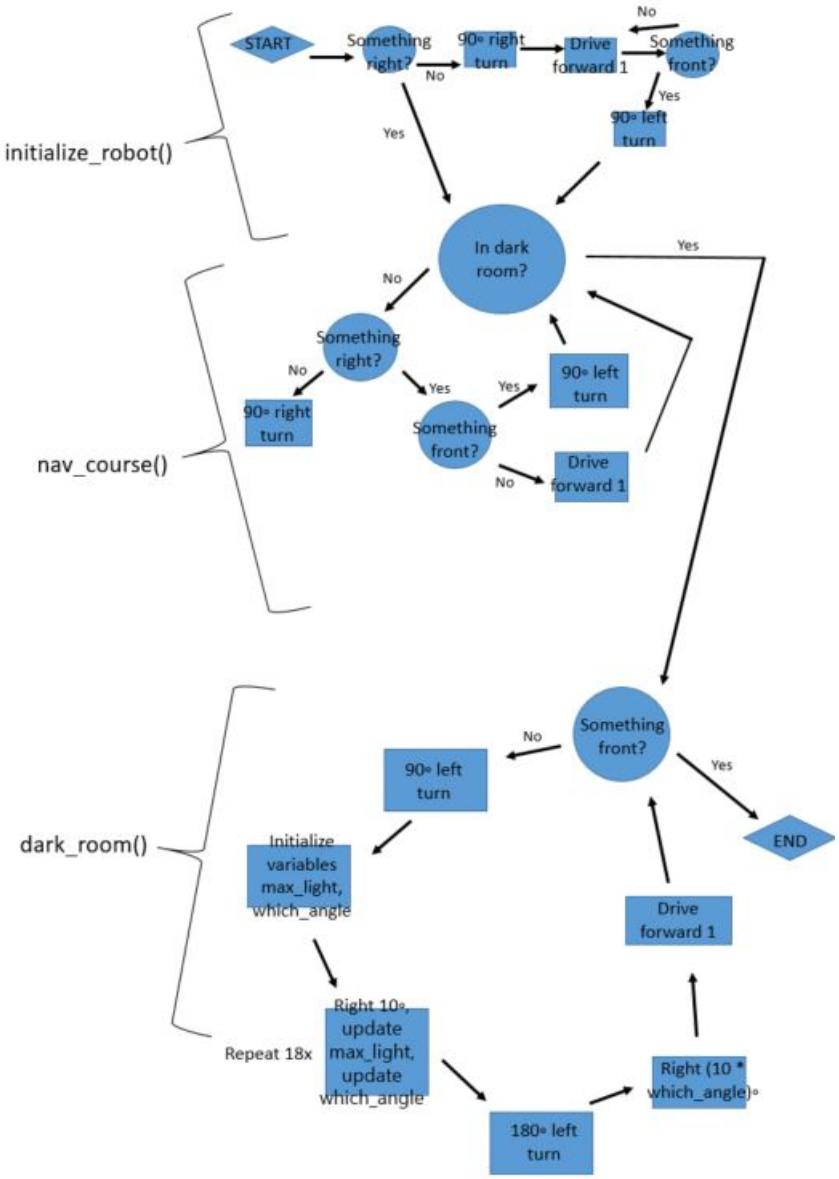

Figure 2: High Level Algorithm

4.2 Flowchart of High-Level Algorithm

The high level algorithm was instrumental in understanding the logic of the final algorithm and determining how to decompose complex interactions into small, manageable pieces that could be individually tested. The high level algorithm can be viewed in Figure 2.

5 System Integration and Test Phase

5.1 Tests

As explained the Design Methodology section, tests were decomposed into their smallest components, and repeatedly tested throughout the integration phase. Details describing the success and challenges of each test are described in detail below.

5.2 Unit Tests

These unit tests provided small, yet consistent building blocks that allowed more complex behavior to be successful. These unit tests were repeated after every major addition or change to the code, to ensure that no functionality was broken and that the robot would behave as expected.

5.2.1 Forward Movement

Forward movement was the first test that we attempted. First, we ensured that the code we wrote could power our external servo, before eventually migrating our code over to power the entire robot. The forward movement test took some calibration, as we found that one servo moved slightly faster than the other, which caused the robot to drift in one direction. However, after repeated tweaks, we eventually created behavior we were satisfied with.

5.2.2 Turning

After forward movement, turning was the next logical step. Originally, we had our robot make a wide turn, with only one servo running and the other remaining stationary. However, we then

realized that tighter turns, in which one set of wheels travels forward and one set travels in reverse, allowed the robot to pivot in a manner that would give it an advantage in a space-constrained course.

5.2.3 Backward Movement

Ensuring that the robot could move backward was fairly straightforward after completing the forward test.

5.2.4 Stopping (On Command)

The AGV was tested in its capacity to stop on command, after moving forward 4 feet. We were concerned with how long it would take to stop, and how this would effect our eventual high level algorithm. Through testing, we determined that the AGV was capable of stopping on extremely short notice, and did not skid or move past its mark when the servos were ordered to stop. This was extremely desirable behavior, and we were able to move forward confident that the AGV would not hit obstacles if properly programmed.

5.2.5 PING Sensor

The PING sensors were tested first in isolation, by wiring the sensor and printing distance results to the console as was done in the sensor lab. There were several times throughout the integration tests, when the AGV bumped into obstacles and dislodged the sensor, where we repeated these simple functionality tests. Later, we ensured the robot could effectively utilize information gleaned by the PING sensors by programming the AGV to stop a few inches before an obstacle. This functionality (described below) became instrumental in instantiating our high level algorithm.

5.2.6 IR Transmitter/Detector

The original plan was to utilize the IR transmitter/detector pair in a similar configuration as the PING sensors in order to detect and maneuver around virtual walls, which were indicated by black tape. However, after repeated tests, the sensor pair was not yielding reliable data that could be utilized by the AGV to maneuver.

We hypothesize that this was the case for two main reasons: first, the floor in the EECS area of Thayer is highly shined, which could cause IR interference and unreliable data; and second, because the black electrical tape we were using to test was fairly reflective. We believe that the combination of these two phenomena yielded unreliable data from the sensors, so we ultimately decided to improve our physical wall maneuverability, and disregard virtual walls. While we understand this would not be an acceptable solution in a production environment, for the sake of the Project Demo we believed that physical wall navigation would suffice.

5.3 Photoresistor

The next sensor to test was the photoresistor, which measured the level of light in a room. This sensor was instrumental for the second task in the course, navigating toward a source of light in a dark room. The photoresistor was tested in a similar manner to the PING sensor, by first conducting a simple test that printed the light level to the console, and then later ensuring the AGV could interpret the information and start and stop based on the light level.

5.4 Whisker Touch Sensor

While we were given the whisker touch sensors as a possibility for this project, our group decided early on that we were not going to utilize them, in favor of spending more time calibrating the other types of sensors. Our logic was that the whisker sensors could not provide any additional functionality that multiple PING sensors could not provide, and that we would rather avoid actually touching or running into walls or obstacles if at all possible. Thus, we did not conduct testing on the whisker touch sensors. If more time was provided, we likely would have conducted testing to see if they could provide any additional functionality, but because of the time constraints, we ultimately moved on to add additional complexity to our algorithm.

5.5 Integration Tests

Once the unit tests were complete, and we were satisfied with the basic building block functions of the AGV, we moved forward with

more complex tasks in preparation for the Project Demo.

5.5.1 Stopping (Physical Wall, 90 Degrees)

More advanced tests were completed for the PING sensor, ensuring that they could receive information from their environment and interpret it correctly to produce desired behavior. We first tested the front PING sensor to ensure it could cause the robot to stop in sufficient distance to avoid running into obstacles and walls. First, we tested to ensure the robot could stop when moving directly perpendicular to a wall. This worked in a fairly straightforward manner, although we realized that the AGV would need to be stopped at least 12 inches prior to the wall, to allow sufficient space to turn and not hit the wall.

5.5.2 Stopping (Physical Wall, 45 Degrees)

We then moved on to see how the AGV would react to walls that were not directly perpendicular, but rather at a 45 degree angle. From our experience in the Sensor Lab, we had prior knowledge that the PING sensors were occasionally unreliable when utilized at extreme angles. Through testing, we discovered that we could rely on the data from our PING sensors consistently at angles up to 35 degrees, but any sharper angles and the AGV only stopped occasionally. This was valuable information, as we knew how we would have to react in the Project Demo to physically pick up the AGV before it would hit obstacles.

5.5.3 Stopping (Physical Wall, Obtuse Angle)

Similar to the 45 degree angle, we wanted to test how the AGV would react to extreme angles. We discovered that physical walls at angles larger than 45 degrees were almost never detected by the PING sensors. While we considered angling our PING sensors in order to better detect such walls, we eventually decided on the simpler, but perhaps more risky strategy of relying on a straight forward PING sensor.

5.5.4 Navigating Test Course

We tested the algorithm for navigating the test course and tracing using the "right hand rule" in several different ways. First, we tested the AGV's ability to simply trace a straight wall, staying relatively close to, but never touching the wall. At first, the code we wrote only implemented a single value for the trace distance. We quickly realized that this was insufficient, and that two values were necessary to keep the AGV in a "lane" a few inches away from the wall. This ensured that the AGV was able to make small corrections to the left when it got too close to the wall, and to the right when it moved too far away from the wall. By making this lane fairly narrow, we were able to get repeatable results in moving the AGV down a straight wall, even for distances up to 50 feet.

After we were comfortable with the simple wall tracing behavior, we began testing how the AGV reacted to obstacles and walls in front. We discovered that obstacles in front were tricky, and that successful navigation relied on precise turning angles, which brought us back to further calibrating the turn tests. Similarly, we noticed the need for code that initialized the AGV, and allowed it to find a wall on its right hand side to trace, in case it was deployed in the middle of a course with no wall to its right. These repeated tests, while time consuming, yielded valuable knowledge that ultimately led to success in the Project Demo.

5.5.5 Navigating Towards Light Source

We tested the algorithm for navigating toward a light source in a darkened room in isolation, and then in combination with the algorithm for navigating the outside test course. Utilizing the elevator room on the first floor of Thayer, we had one group member with an iPhone flashlight and one group member following the AGV to ensure it did not crash into any walls. The AGV scanned from left to right to find the maximum light reading, before moving towards that direction. We noticed that the robot had trouble finding the light if it was located at an extreme angle relative to the door (i.e. if it was at a 90 degree angle), and that the AGV was prone to traveling back out the door into the lighted area out-

side. However, during the Project Demo, the door was covered so as to block light, which mitigated this problem.

6 Final Results

6.1 Project Demo Results

During the Project Demo, NetGrok II received the Orange Course, which was favorable to its design of the "right hand rule," as the course consisted of a 90 degree right turn with three obstacles, and another 90 degree right turn into the darkened room. In the first run, the AGV recorded a time of 1:36, which included a 30 second time penalty for an adjustment so that the robot would not run into an obstacle, which would have resulted in a point deduction. However, on the second attempt, the AGV did much better. While some of the run must be attributed to luck and obstacle placement, no adjustments were needed and the AGV navigated successfully to the room in 1:06. This was the fastest recorded time of any AGV in the course.

6.2 Completed Design Photo

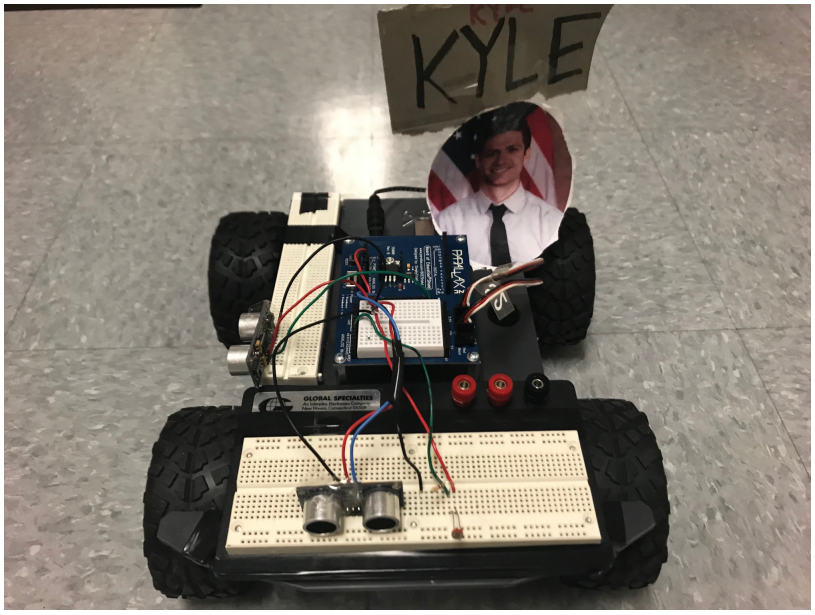

Figure 3: The completed design, bearing the name and dashing image of EECS favorite Visiting Professor

7 Conclusion

7.1 Directions for Future Work

While the Mini Project only allowed for a few weeks of work, there are many interesting future directions for similar AGVs and additions that could improve the design. If given more time, additional sensors could have been incorporated, such as the whisker touch sensor, which would have made obstacle detection and avoidance more robust, while providing a backup to the PING sensors in the case of hardware failure. Similarly, the NetGrok II team would have liked to utilize a basic learning algorithm in the design of the robot, so that the robot could store memory and learn about the course as it navigates through, making repeated forays into dangerous areas faster and less error prone. The hardware of the AGV could be improved in various ways before final deployment and use in combat, to include shielding to prevent wires becoming undone, as well as more robust tires that allow the AGV to better navigate harsh terrain.

The NetGrok II team sees the primary use case for such an autonomous robot in scouting missions, where, combined with video recording or live streaming devices, the robot would allow ground troops better knowledge of the terrain and enemy composition in harsh or restricted environments where soldiers are unable to go. Similarly, NetGrok II's AGV design could be used for search and rescue missions, autonomously navigating restricted cave areas or collapsed buildings to locate trapped individuals and offer assistance.

7.2 Final Thoughts

This project was incredibly useful in a variety of ways. First, the NetGrok II group got valuable experience in robotics on both the hardware and software side. However, even more valuable was the practice in design methodology and the lessons learned working together as a team. Moving forward, the NetGrok II team has a much greater appreciation for the power of functional decomposition, which made development much simpler, and also for repeated unit testing, which allowed the team to quickly iden-

tify and fix problems before technical debt was accumulated. The NetGrok II team met to discuss lessons learned, and is excited to work on even more challenging projects in the weeks to come.

The NetGrok II team would like to sincerely thank their instructor, LTC Harvie, for his advice and assistance throughout the project, as well as their advisor, Mr. King, for his support and the inspiration he provided in naming the robot.

8 References

The source code utilized in this paper can be found at: https://github.com/youngdaniel345/AY19_NetGrokII/tree/AGV.

The video for our IPR can be viewed at: https://www.youtube.com/watch?v=Rx7hgi24FS8.

This paper can be viewed in electronic form on Overleaf at: https://www.overleaf.com/read/mkgfgtnvrpqy.