Proceedings Of The National Conference
On Undergraduate Research (NCUR) 2019
Kennesaw State University
Kennesaw, Georgia
April 11-13, 2019

Exploring Cooperative Robots Using Uninformed Search Strategies

Jackson M. Spivey
Computer Science
Randolph-Macon College
Ashland, Virginia 23005 USA

Faculty Advisor: Dr. John McManus

Abstract

This research project explores the use of a lowest cost first search and island hopping to solve a maze and a blackboard system to transfer the necessary information between agents to complete this task. Blackboard systems provide an effective yet simple method of transferring information between agents connected to the centralized server, or "blackboard". The lowest cost first search algorithm enables the robots to minimize the length of the path between the locations, known as the islands, of the partner robots.to the robots use the shared information to traverse the shortest path to the goal at the end of the maze. This research project evaluates the effectiveness of multiple robotic platforms being able to communicate with each other under the blackboard system and the effectiveness of the lowest cost first search algorithm to find the best solution between the islands in the maze. The combination of the lowest cost first search algorithm and island hopping is like adding waypoints on a trip from point A to point C. The best path is one where every waypoint between A and C is hit making it the shortest path with all goals in mind. This project demonstrates the usage and effectiveness of sensors to aid in the autonomous movements made by the robots.

Keywords: Robotics, Autonomous, Cooperating Intelligent Systems

1. Introduction

The topic of cooperative robotic problem solving was first explored in the late 20th century, with research continuing into the 21st century. The idea of utilizing multiple robots with sensors is a topic explored in different ways throughout the 21st century. Attempting to solve a maze with multiple robots as effectively and efficiently as possible is a focus of several projects. The types of technology required to navigate and avoid obstacles is already prevalent in devices such as rumbas and robotic pool cleaners that map an unknown area and avoid objects to complete their task. However, very little work has been done to address multiple robots working together to map an area as effectively as possible. For example, one cannot put multiple rumbas in a room and expect them to map the room, clean the room without overlapping, and clean as quickly as possible. A blackboard system, which is a system of knowledge sources sharing information to a centralized location, is an approach to sharing data and cooperating to solve a problem. The robots need to communicate their information quickly and effectively. A key issue presented with the maze solving problems is how to effectively utilize the blackboard system with any given number of robots utilizing problem-solving algorithms to solve a maze.

The objective of this project is to explore and identify an effective usage of the blackboard system and utilization of problem-solving algorithms to solve a maze. The research question is: can an uninformed search strategy, such as a lowest cost first search, be used by cooperating robots under a blackboard system to solve a maze together? Through the exploration of different methods of data collection and problem-solving techniques the problem can be solved as quickly and as efficiently as possible. With an effective implementation, one could utilize the approach for any issue that requires autonomous robotic navigation and path planning.

2. Background

2.1 Blackboard System

A blackboard system is a computational approach that allows multiple agents to efficiently share information with each other. The blackboard, likened to a physical blackboard, allows knowledge sources to write to a central server that processes the data and sends it back out to the knowledge sources that require it. In advanced systems this is important because it allows for an efficient transfer of data and frees up processing power for the knowledge sources that otherwise would be wasted trying to receive data from an N number of knowledge sources.[1] In this project the blackboard is necessary to distribute the problem solving task among the set of cooperating robots. This approach allows each robot to solve to solve their portion of the maze concurrently, reducing the total time spent computing a solution. The agents need all the resources available in order to drive through the maze and read from sensors properly.

2.2 Maze Solving Artificial Intelligence

The maze solving AI implemented a lowest cost first search algorithm and an island-hopping approach to compute the robot's path out of the maze. The lowest cost first search algorithm in this project uses distance as the cost and the islands are the starting points of the different robots cooperating to solve the maze. Cost can be variable from energy spent, to distance, to financial cost, etc., though distance is simply used as cost in this project. A search strategy such as the lowest cost first algorithm is important because the concept is utilized in the real world with things such as GPS where a lowest costing path, cost being time and distance, is found between points or subpoints to a goal. [2]

2.3 Raspberry Pi

The Raspberry Pi is a single board computer that is relatively inexpensive compared to its performance and functionality. The Pi features General Purpose Input and Output pins (GPIO) that allow for electrical signals and data to be sent between the computer and any sensors, electronics, and devices connected to the Raspberry Pi. This allows for versatility making it a perfect candidate for being the brain of a robot.

3. Methodology

3.1 Hardware

The hardware utilized in this project are the Adafruit BNO055 sensor, Raspberry Pi 3, Raspberry Pi 3 prototyping board, RPi PowerPack, Adafruit Motor Driver HAT, and Adafruit Robot Rover Chassis Kit. The robot kit was the basis for what was the final build of the robot. The robot's brain is a Raspberry Pi 3 found in Figure 1. Using a Raspberry Pi 3 allows for a relatively simple process of connecting and interacting with the motor controller and BNO055 sensor. The power source is an RPi PowerPack that allows for easier control of power and easier charging of batteries.

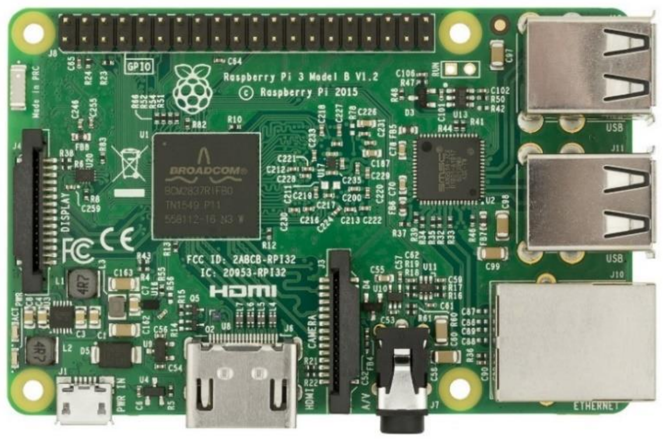
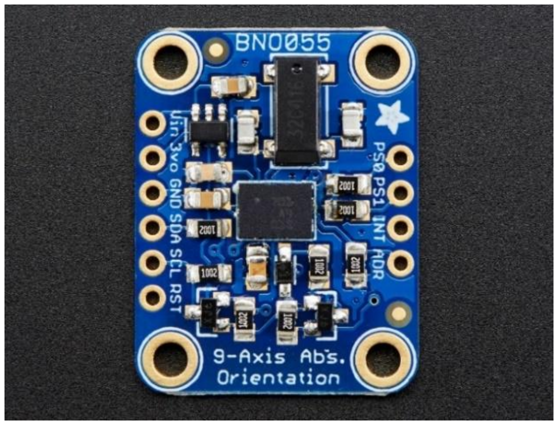

Figure 1. Raspberry Pi 3 Model B; Figure 2. BNO055 Sensor

Attached to the GPIO pins of the Raspberry Pi 3 is a prototyping board allowing for the BNO055 sensor to be mounted and connected in a stable and protected manner. The prototyping board allows for even more sensors and extra systems to be easily connected to the Raspberry Pi. Stacked on the prototyping board's GPIO pins is the motor drive HAT for the two DC micro servos that controls the wheels.

The sensor module used to aid in the navigation of the robot is the BNO055 sensor shown in Figure 2. This sensor features an accelerometer, magnetometer, and a gyroscope. The raw data is fed into a fusion algorithm that calculates where the sensor is in 3D space. The sensor can gather additional data. For this project it is set in a compass mode where the magnetic heading is utilized for turning and driving straight.[3] This sensor is necessary because sensor navigation is more precise and accurate than dead reckoning navigation. The difference is the sensor gathers data from the real world to apply to the driving whereas in dead reckoning the robot is simply told how long to drive and in what direction. The robot does not know where it is, just where it should be.

3.2 Software

The project utilized open source software and software developed by the Principal Investigator. The open source software consists of the Adafruit motor controlling software and robot drive software that abstracts the control of motors into simple functions. Adafruit's software and the BNO055 sensor software interface was utilized to read from the sensor.

Software developed for this project can be broken into multiple parts: the network code that handles the blackboard system; the algorithm that creates and solves mazes; and the main code that handles the control of the robots and their duties. The software to control the robots is designed to use information from Adafruit's BNO055 sensor in conjunction with the motor control software to turn the robots a certain amount of degrees and to keep the robots driving straight. It also is designed to interact with the server to exchange information with the blackboard system.

The maze software processes an input file that contains the description of the maze. This data is then translated into a representation that the robots use to solve parts of the maze and in the robot control software to exit the maze. The solution uses a least cost first search algorithm that takes in a maze and finds the least costing path from a start node to a goal node. Network software then takes this solution and sends it to the server where the server waits for all robots to send in their data. Once the data is received the server is designed to handle the processing of the data and send the results to each robot.

A simple socket server is used to accomplish the transfer of information. Threads are used to allow an N number of clients to connect and interact with the server. The server is told where each robots' start and end goals are, but not the path. This information is then sent to the connecting robots. The maze code is designed to allow for scaling with maze size and number of robots solving the maze.

3.3 Design of Robot

The original robot design was modified to fit the needs to the project. A swivel caster wheel was provided by the manufacturer but through testing it was found that the robot struggled to get the caster wheel to realign with the body of the robot when moving straight. This proved problematic as the robot could not go straight because of the caster alignment errors. The difference in the speeds of the micro servos also heightened the effects of this problem. The micro servos run at different speeds due to them being inexpensive and not all tuned the same. As a result, the swivel caster wheel was replaced with a ball caster that allowed for much easier turning and did not skew attempts at driving straight like the swivel caster did in conjunction with the servos running at slightly different speeds.

The stacking of the different components on the robots was redone to make space for the addition of the Raspberry Pi 3, breakout board, and RPi PowerPack. As displayed in Figure 3 the stacking is as follows from bottom to top, RPi PowerPack, Raspberry Pi 3, breakout board with the BNO055 sensor attached, and the motor HAT controller.

The BNO055 sensor requires 3.3V power connection to the I2C or Uart pins and a connection to any GPIO pin for a signal for resetting the sensor. Adafruit, the company that made the BNO055 used in this project, recommends using Uart over I2C due to a bug in the Raspberry Pi's. As a result, the decision was made to connect the BNO055 to the Uart pins of the Raspberry Pi as shown in Figure 4. The red wires are both 3.3V from the breakout board, the yellow and orange wires connect to the Uart pins on the breakout board, and the grey pin is connected to GPIO pin 18 to act as the reset for the BNO055.[3]

3.4 Experiment Procedures

A typical run-through of the maze was comprised of a Raspberry Pi based Python socket server acting as the blackboard and three robots acting as the agents of the blackboard. Before the robots connect to the server, they are given a text representation of the maze with no solution or goals. The robots are placed staggered throughout the maze. One typically as far as possible from the end, one a third through the maze, and the last roughly over halfway through the maze. Then, robots initialize communications with the server prior to beginning the physical maze run. After initial connection the server notifies the robots of their start and end nodes. The original intention was the maze would be visual based and the robots would have to map the maze themselves, but due to time constraints and budgetary concerns it was not possible to bring to fruition.

After communications are established, handshaking is done, and the server assigns start and end nodes for each robot the run begins. The robots run the Lowest Cost First Search algorithm for their part of the maze returning a path from their start node to goal node. Once this is done the paths are sent to the blackboard. Personalized paths for each robot based off what was sent from the other agents are packaged and sent to respective robots. For example, the first robot needs to have the paths of the other two robots placed further in the maze in order to have the full path out of the maze. Receival of the paths triggers the robots to drive the path out of the maze based off what is sent from the server.

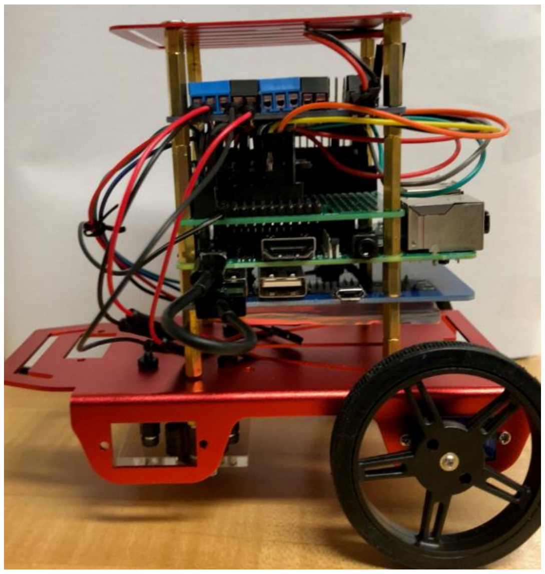
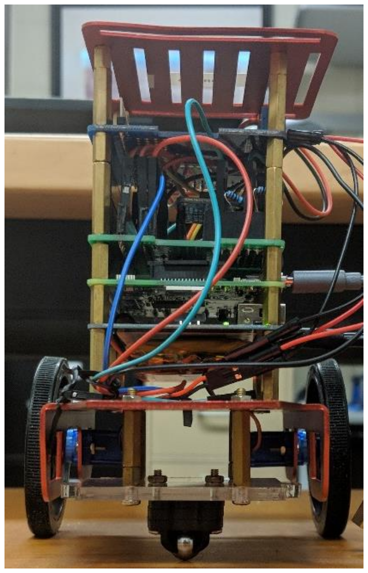
Figure 3. Robots utilized in the project. From bottom to top: RPi PowerPack, Raspberry Pi 3, GPIO breakout board with the BNO055 attached, motor HAT controller.; Figure 4. Display of fabricated ball caster attachment and the different wires attached to the BNO055.

4. Data

4.1 Maze Results

The maze software was tested multiple times with mazes that contained a least costing path to the end goal, multiple paths to the goal, and no path to the goal. The algorithm could identify the least costing path from the start node to goal node. It also was found that when given multiple paths it would find the least cost path. When given no path to the goal it quickly identifies that there is no path to the goal and proves that the least cost first algorithm designed for this project is properly implemented.

4.2 Robot-Server Interaction Results

The interactions between the robots and servers works without fail through testing. An input file was given to the server multiple times with varying start and end nodes. Robots were then connected to the server and waited for a response from the server with their individual start and end nodes. The robots then proceeded to solve for their paths to their goals and send those to the server. After intaking that information the server determined which robots requires what path or combination of paths based on the order in which the clients connected and sends that information. In effect, this is what the blackboard system looks to achieve. The facilitation of cooperation between the robots is successful and quick under this system and works consistently through testing.

4.3 Robot Drive Results

The drive of the robots received extensive testing. The tests produced varying results depending on where the robot was tested. In the lab the turn and straight functions, which are intended to make the robot turn a certain number of degrees and correct steering, worked consistently. The turn is sometimes ±1 degree off and the straight function sometimes gets caught in the middle of a course correction, so it is also on average ±1 degree off of its target. Consistently in testing in the lab on a near perfect flat surface with no magnetic distortion.

Once the robots were taken to the first location to drive, a classroom in the Copley Science Center, it was found that the robots would hit certain spots on the floor and drive in extremely inaccurate ways. The data the robot returned for the heading simply did not match what it should have been, and numbers were oscillating. Through further investigation and testing it was discovered that there was magnetic interference coming from under the floor that was powerful enough to skew the sensor's heading reading.

The tennis courts at Randolph-Macon were the only location available that had little to no magnetic distortion and a flat enough surface to attempt running tests with the robots to see how they drove. The heading readings were much more accurate than the readings on the floor in the lab. However, it was not possible for the robots to drive slow enough to move and not skew the compass readings due to the composition of the surface on the tennis courts. The robots reached the limit on their capabilities and are not able to drive in a semi-rough terrain that requires significant traction and a larger mass to navigate.

4.4 Drive Straight Results

The function designed to use the heading to correct the robot when not going straight was tested and the heading values were recorded throughout the duration of the test. Testing was also done in a program that replicates the effects of driving steering in a direction requiring the drive straight code to correct itself. This proved that the code was doing what it was intended to and set a baseline for what should be expected when testing driving. The code behaved as expected when tested with test cases. The heading deviated in the data from table 1, and the program corrects the speed of the motors to correct the heading.

Table 1. Test results of straight driving correction in a simulated environment.

Heading (Degrees)	Difference from Initial (Degrees)
263	0
262	-1
263	0
262	-1
263	0

The heading data extracted when doing the same process but with driving the robot on a table in the lab yielded similar results. As seen in table 2, the robot begins driving forward at a heading of 263.4375 degrees and starts to lean to the left. The fourth value displays the code reacting to this and correcting the heading to go closer to zero. Further inspection of the change in heading shows an oscillation between an overcorrection of ± 1 degree. This is expected and does not subtract from the driving. The numbers in Table 2 suggest that when driven on the flat table in the lab that it is perfectly capable of driving straight within ±1 degree of error.

Table 2. Test results of straight driving correction in the lab on a hardwood table.

Heading (Degrees)	Difference from Initial (Degrees)
261.75	-1.6875
261.375	-2.0625
262.75	-0.6875
264.375	0.9375
262.5	-0.9375
263.5	0.0625

When tested on the floor of the room with the maze the data suggest environmental distortion affecting the magnetic heading. As show in table 3 the data is oscillating. The heading began at 178 degrees and it immediately jumped to 195 degrees. This means that the robot drove way off course. However, observations suggested otherwise, and the robot appeared to be driving correctly at this point. The robot had only just started driving across the floor which led to an investigation on why the robot was returning that data during the test.

Table 3. Test results of straight driving correction on the floor of the lab where the maze was built.

Heading (Degrees)	Difference from initial (Degrees)
178	0
195.9375	-17.9375
198.625	-20.625
195.5625	-17.5625
190.9375	-12.9375
184.5	-6.5
181.1875	-3.1875
178.1875	-0.1875
177.3125	0.6875
178.1875	-0.1875

4.4.1 Testing the Environment

To figure out why the BNO055 sensor was getting the seemingly heavily skewed results from Table 3 a simple test was done to the environment. A standard compass was placed directly where the robot got numbers found in table 3 when driving forward. As show in Figure 5 the left compass reads about 230 degrees. In Figure 6 the right compass is stacked on books in the same exact place high enough to be mostly out of range of the magnetic disturbance and reads close to 5 degrees. The compass on the book was then compared to a compass on a smart phone and to itself in different parts of the room and through that it was discovered that something under the floor and close to the floor was causing a magnetic disturbance. The source of this magnetic disturbance is unclear, but there are spots like it all over the room that made the room not fit for further testing.

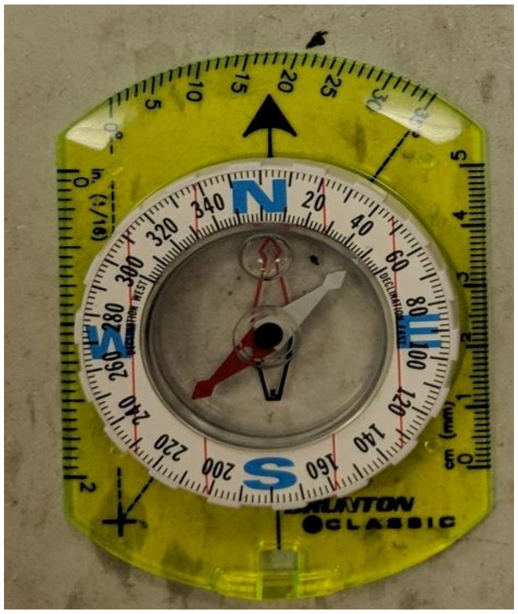
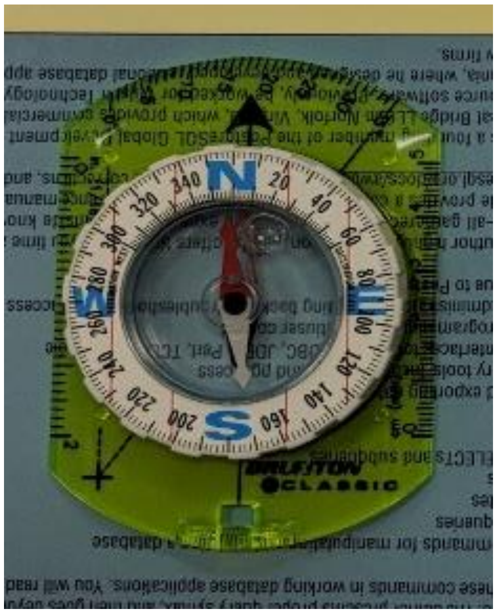

Figure 5. Compass placed on the same spot of the floor in the lab. Pictured right is the compass on a textbook. The left compass reads 230 degrees and the right compass reads 357 degrees.

4.5 Turning Results

The same experiments that were done with driving straight was done with turning the robot. First the code was tested with made up numbers same as before. The turn function was given a heading and a degree of turn. The heading simply increased until it reached the target heading determined by the degree of turn. The code work as intended and terminated once it reached its goal heading. Much of the data in Table 4 is omitted simply because it is redundant. The heading began at 178 degrees and finished at 268 degrees. This test program and data can be used to prove that the code works as intended and if the heading reads between ±1 from the start heading + the degree of change.

Table 4. Test results of robot turning in a simulated environment.

Heading (Degrees)	Change in Heading (Degrees)
262	2
264	2
266	2
268	2
Final degree of change	90

The turning function was tested on the same table in the lab used for the go straight testing. The heading was recorded and the end of each iteration of the loop that checks the heading and determines how close the robot is to its goal. The robot was told to turn 355 degrees and the results from the experiment shown in Table 5 suggests that the turn code is accurate at accomplishing this. Most of the table has been omitted for redundancy.

Table 5. Test Results Of Robot Turning On The Table In The Lab.

Heading (Degrees)	Change in Heading (Degrees)
153.875	1.375
156.375	2.5
161.8125	5.4375
164.3125	2.5
165.6875	1.375
169.8125	4.125
173.8125	4
Total Degrees Turned	355.8125

The same test was done on the floor where the maze was after it was established that the robot can turn within ±1 degree of accuracy on the lab table. The robot was purposely placed on a spot where magnetic distortion occurred to see how it would affect the heading readings. The most notable difference between the data in this experiment and the one in the lab is that the heading fluctuated up and down before increasing like normal. This was the only anomaly that occurred and is displayed in Table 6 with redundant results left out. The robot was meant to turn 355 degrees but ended up turning 10.375 degrees off at a total of 365.375 degrees. This further proves that the floor in the room with the maze has magnetic distortion.

Table 6. Test results of robot turning on the floor in the lab.

Heading (Degrees)	Change in Heading (Degrees)
111	12
110.0625	-0.9375
106.8125	-3.25
105.5625	-1.25
110.0625	4.5
113.6875	3.625
119.6875	6
115.5	-4.1875
119.125	3.625
Total Degrees Turned	365.375

5. Conclusion

Overall the project was a success proving that a least cost first search algorithm combined with island hopping and a blackboard system can allow for multiple robots to cooperate to solve and navigate their way out of a maze. Although, the robot drive functions could not work to physically display the ability of the robots to exit the maze, the interactions between server and client display that the robots have the correct path to the end returned to them. The blackboard system proved to be useful and extremely effective at allowing for the sharing of information to allow agents to cooperate. Without this system the robots would have to connect to each other in a robot by robot basis sapping much of the processing power into managing an N number of connections with other robots. That approach simply does not scale like the blackboard system can. The least cost first search algorithm and island hopping also displayed its ability to scale in conjunction with the blackboard system. Testing displayed that if there are enough robots connected and a large enough maze that the maze solving algorithm can find paths to sub goals and to the end goal of the maze.

The BNO055 sensor also increased the accuracy of the drive of the robots immensely. Before introducing the BNO055 sensor the turning had to be guess-work as there was no way other than time to ensure the robot would turn a certain amount. Utilizing the BNO055's compass heading enabled the turning to be without a doubt an accurate 90-degree turn. The sensor also aided immensely in driving straight because it allowed the robot to be able to tell when it is no longer going straight and to correct for it by increasing the speed on the side that needs it increased. The BNO055 proved its worthiness, though, other sensors could be used in places where the BNO055 struggled with magnetic distortion and therefore would be more reliable.

In the future, significant additional work can be done to realize the project's original goals. Most importantly the maze needs to be physical as opposed to abstract. Currently the maze is represented in a text file and can be laid out on the floor. Adding physical walls would allow the robot to navigate a maze without the text representation. The robots would require additional sensors in order to navigate a physical maze with walls. Furthermore, the robots would have no knowledge of the maze and would have to map their own maze and find the way out. If the maze has 40 nodes total and four robots are placed in, then the goal is to make four robots solve the maze more efficiently, in terms of time, than a singular robot solving the same maze. Multiple methods of searching and cooperation can then be tested and an entirely new design of the blackboard would have to be created. Additional processing is required to assemble the exit path, making the robots autonomous agents with no pre-shared model of the maze. The blackboard system then would be fully realized as the agents will do little to no processing of data leaving it for the blackboard to handle.

Due to time and hardware constraints more could not be done about the failure to drive well enough on surfaces in a less controlled environment than the lab. In the future hardware including, but not limited to, the weight of the robot and wheels could be changed out for something that favors imperfect environments more. Also, more powerful and accurate sensors could be used such as GPS. With GPS and a robot better suited for the outdoors further testing and stress can be put on the blackboard system and robots cooperating to use uniformed search strategies such as the lowest cost first search.

6. Acknowledgements

The author would like to express their appreciation to Dr. John W. McManus for the continued support throughout the duration of the project.

7. References

1. McManus, John, W. 1991. "Design and Analysis Tools for Concurrent Blackboard Systems." *10th Digital Avionics Systems Conference* 9.
2. Poole, David, and Alan Mackworth. 2017. *Artifical Intelligence 2E Foundations of Computational Agents.* Cambridge University Press.
3. Townsend, Kevin. 2015. *Overview.* Apr 22. Accessed June 2018. https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/overview.