Department of Electrical Engineering
The Cooper Union for the Advancement of Science and Art

Megamouse

An Autonomous Maze Solving Robot

Sharang Phadke Sameer Chauhan
phadke@cooper.edu chauha3@cooper.edu

This report documents the process of designing and constructing a robot that can autonomously solve a maze. As the robot explores the maze, it extracts information about the environment from sensors and uses the data to determine its orientation. Subsequently, it applies a maze solving algorithm to navigate to the center of the maze as quickly as possible. The robot described here was entered into the IEEE Region 1 Micromouse Competition in the Spring of 2014.

Advisor: Prof. T. Cumberbatch ECE 195A - Spring 2014

Contents

1 Introduction

Designing a robot that autonomously performs a task is an abstract engineering challenge that has numerous specific applications. Every such abstract problem involves implementing a system that makes intelligent decisions based on information gathered about the environment. Research in the larger field of Autonomous Mobile Robots (AMR) considers household, military, and commercial situations in which robots that perform such functions can assist humans.

Thus far, no single robot has been shown to be advanced or modular enough to perform arbitrary household tasks, nor can current systems completely learn complex new tasks. However, advances in the field of AMR have led to valuable solutions for specific tasks, such as Roomba®, the automated vacuum cleaner, Wheelbarrow, the famous bomb defusing robot, and of course Opportunity and Spirit, NASA's Mars exploration rovers.

This report describes the process by which a specific AMR problem - autonomously reaching the center of a maze - was solved. This problem has been solved many times. In fact, the IEEE hosts an annual Micromouse Competition in which competitor robots each attempt to reach the center of a 16 x 16 block maze composed of 18 cm x 18 cm squares. The specifications of the competition, detailed in Appendix 1, disqualify robots which jump over walls, damage parts of the maze, or separate into multiple parts. Competitors have a total of 10 minutes to access the maze, and the robot that completes the single fastest trip to the center is declared the winner. The Micromouse competition was made popular in the late 1970's, and remains popular in the U.K., U.S., Japan, Singapore, India and South Korea. National and international competitions bring together the top micromouse builders, including Kato-san, Ng Beng Kiat, and Nakashima-san. The current world record for the standard micromouse competition is 3.921 seconds, held by Ng Beng Kiat [1].

The challenge of this project was to physically implement a micromouse robot with little specific knowledge of design paradigms that have led other micromouse robots to success. Our approach was to decompose the problem into several high level sub-problems as can be seen in Figure 1, design a subsystem to solve each of these problems, and finally combine the subsystems into a single robot.

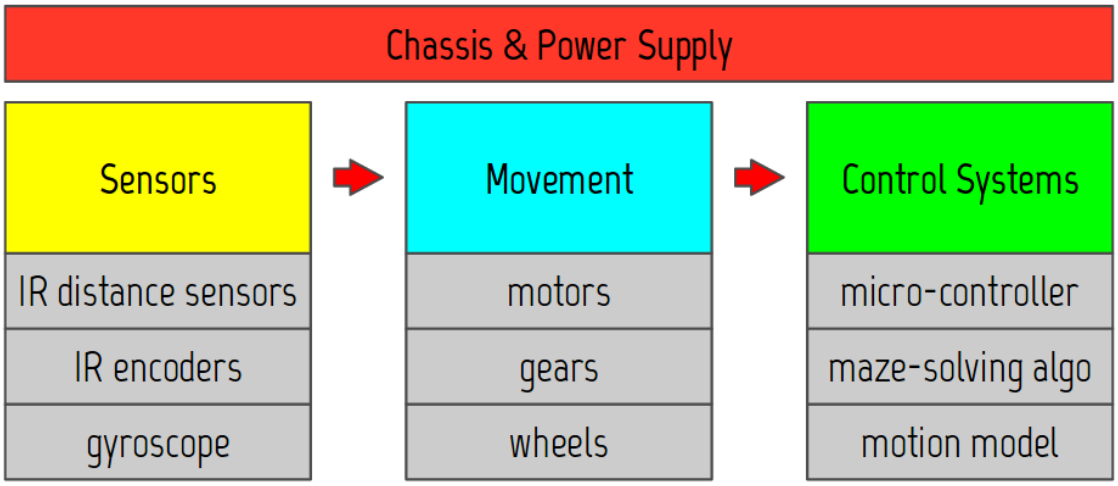


Figure 1: Design Considerations

The following subsystems were designed and implemented, adapting original designs as new challenges were encountered. A sensor system that allows the detection of surrounding walls precisely without compromising a high sampling rate was designed to provide important data about the environment. A movement system that balances speed and precision was designed to enable the robot to traverse the maze quickly. Most importantly, control systems and maze solving algorithms which use sensor data to navigate to the center of the maze as efficiently as possible were implemented.

2 Sensor System

The purpose of a sensor system of an AMR is to allow the robot to extract information about the environment and develop a representation of the robot's surroundings. For the micromouse challenge, this means sensors must detect walls surrounding the robot, as well as track the motion of the robot as it navigates throught the maze. Three important parameters were considered while designing the sensor systems: speed, coverage, and precision (Figure 2). Specifically, the system must have a high enough sampling rate to prevent the robot from making movement decisions based on stale sensor data. The number of sensors and the angles at which they are oriented must cover a wide enough area around the robot to detect walls in the periphery. The sensors must also be powerful enough to minimize the detector error rate (both false alarms and misses) by maximizing SNR. For example, a reasonable benchmark for precision would be re-sampling after the robot travels 5 mm. For a robot with a top speed of $0.5m/s$, this means sampling every 10 ms.

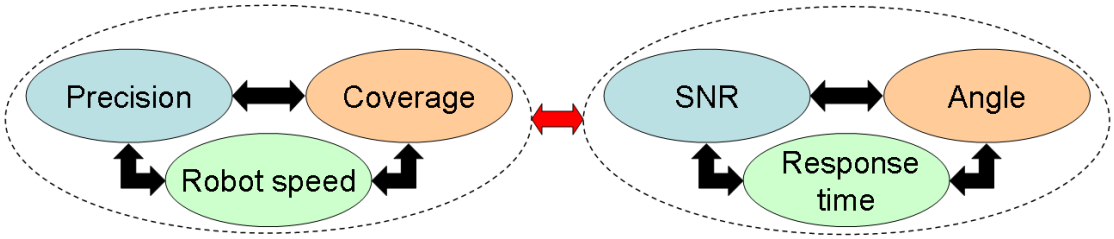


Figure 2: Sensor Trade-offs [2]

2.1 Distance Sensors

Three types of sensors are commonly used for AMR positioning systems: infra-red (IR) sensors, ultrasonic sensors, and simple CCD cameras. Ultrasonic sensors are typical in applications with unknown environments, because they are cheap, easy to implement, and are not subject to high noise conditions. However, they are not generally only precise to $2 - 5cm$. Vision systems, composed of simple cameras, are also not precise enough for this project (typically $5cm$), and require significant processing power to do effective pattern recognition [3] [4].

IR sensors, consisting of an IR LED and an IR sensitive photo-transistor or photo-diode, are the most common among top micromouse robots. The advantage of using IR sensors is that they are the most precise for their cost and power consumption. In addition, the fact that the walls of the maze are white while the floor is black adds to the effectiveness of sensing in the IR range, as IR light is readily absorbed by black paint and reflected by white paint.

The Sharp GP2Y0A21YK IR distance sensor, which is commonly used for amateur AMR applications, was chosen for this project. This sensor has an IR emitter and an adjacent strip of IR photo-resistors behind a lens. The lens directs IR light that is reflected back towards the sensor off objects onto a location on the strip of photo-resistors depending on the distance of the object. The resistance of the strip is altered by the location at which IR light lands on it, and the resultant voltage drop across the strip can be used to estimate the distance to the object. A diagram of this sensor can be seen in Figure 3.

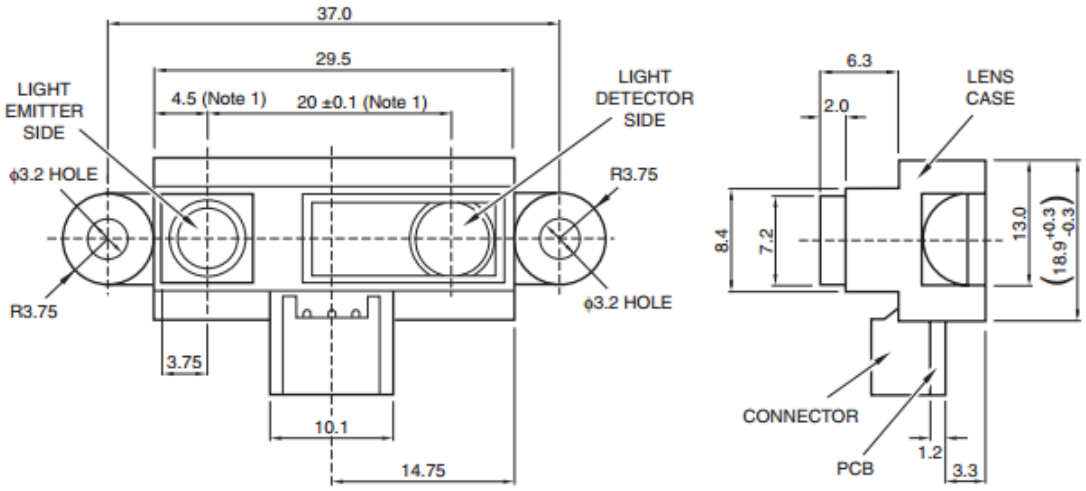


Figure 3: Sharp GP2Y0A21YK IR Sensor [5]

The technical specifications for the Sharp GP2Y0A21YK can be seen in the following table. Most notably, the response time of the sensors is low at 39 ms, but is not low enough to allow for a sampling rate in the 100 Hz or kHz range as desired. However, these sensors were still adequate for use in a micromouse, as they can be oversampled, and can still provide precise distance information that a good control system can use to navigate the maze.

Property	**Value**
Range	2 to 80 cm
Response Time	39 ms
Average Current Consumption	30 mA
Detection Angle	8°

Table 1: Sharp GP2Y0A21YK Technical Specifications

The best micromouse robots in the world use custom IR distance sensors consisting of higher power LED, bandpass filters, and a higher sampling rate to optimize the three design trade-offs [4]. In order to conserve power, the higher power LEDs are often pulsed when the sensors are activated. However, off-the-shelf Sharp sensors are the best way for an amateur team to begin designing a micromouse.

After selecting the Sharp GP2Y0A21YK, we began designing a chassis that placed three IR sensors in the orientation seen in Figure 4. This orientation would allow the robot to detect walls in the next square to the left, to the right, and directly in front.

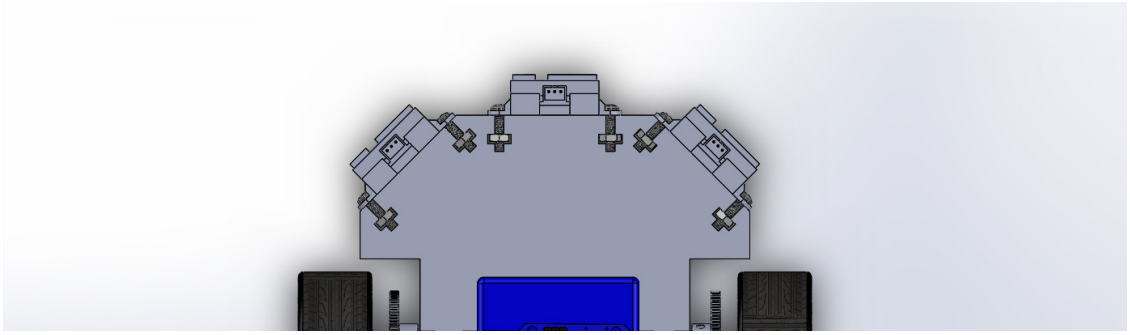


Figure 4: Initial Sensor Configuration

However, a test of the sensor's transfer characteristic indicated that this design would greatly increase the detection error rate. The test was carried out on an optical bench where an object was placed at specific distances from each IR sensor, and the output voltage was recorded. Figure 5 shows the observed distance vs. voltage curve for all three sensors, where each point is the average of 100 samples at a particular distance.

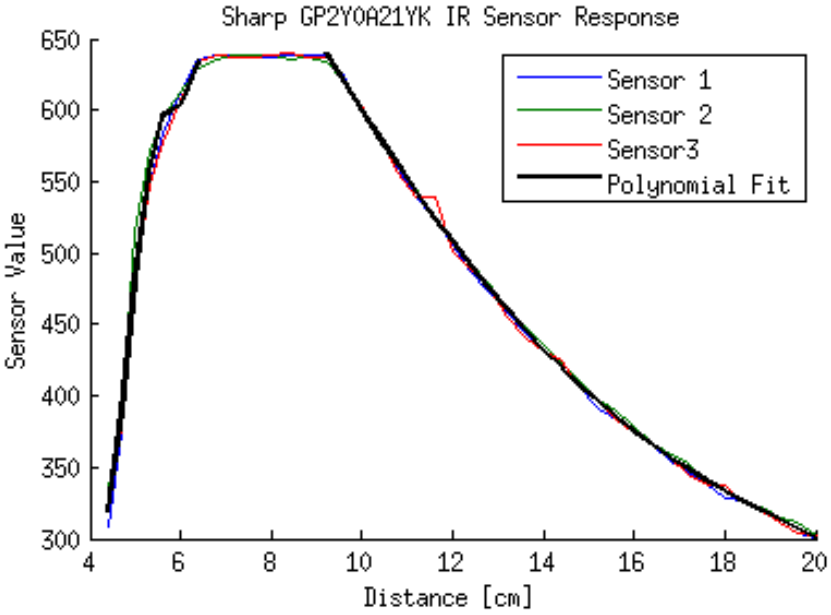


Figure 5: IR Sensor Response

Although the sensors were rated for distances greater than 10 cm, there was a clear need for detecting walls closer to the robot than than 10 cm. The plot above, however, shows that the response function that is not invertible. That is, given a voltage (especially in the range of 6 cm to 9 cm) there are multiple possible distance values that could have produced that voltage. As a result, the sensors were reconfigured to increase the distance from each sensor

to the wall it was designed to sense. The new, inverted orientation can be seen in Figure 6.

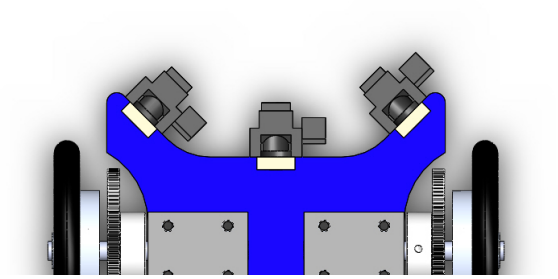


Figure 6: Final Sensor Configuration

2.2 Encoders and Localization

Essential to successfully solving an AMR problem like the micromouse challenge is the robot's ability to localize itself within the environment. Shaft encoders, which allow the robot to track the number of rotations of each motor, were used to solve this problem. Figure 7 shows the Polulu Optical Encoder Set, consisting of a notched wheel and a PCB that has a pair of IR sensors and a signal processing unit mounted on it.

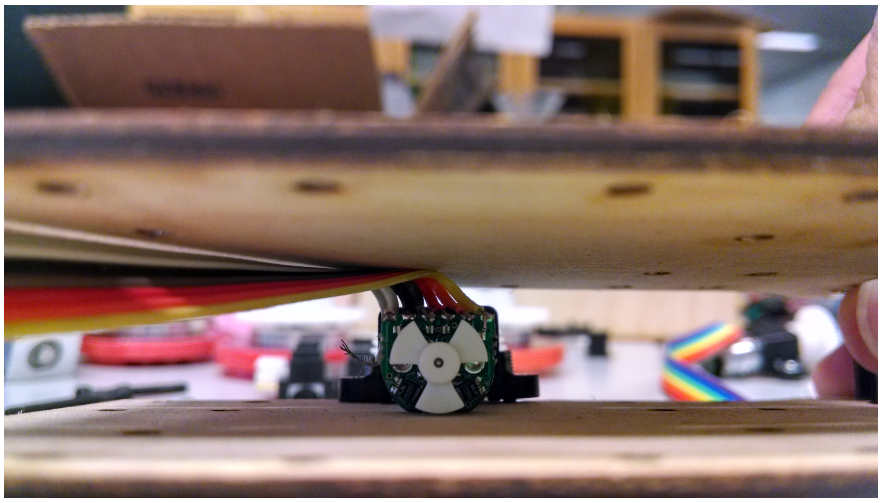
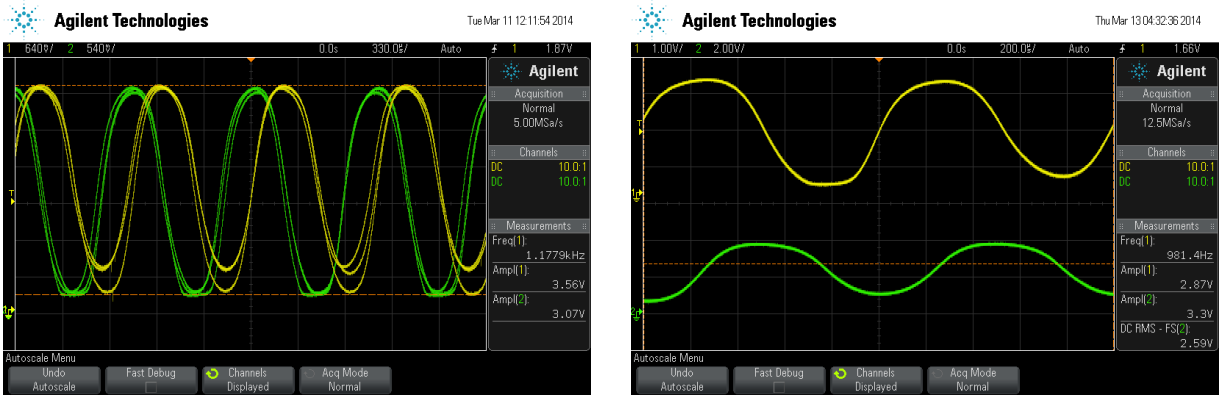


Figure 7: Polulu Optical Encoder Set

The PCB is mounted to the back of the motor itself, and the wheel is mounted onto the motor's rear shaft, which spins at thousands of revolutions per minute, depending on the power supplied to the motor. As the wheel spins, the two IR sensors mounted on the PCB produce signals that are out of phase with each other, as can be seen in Figure 8a.

In order to receive accurate encoder counts from these signals, the wheel had to be moved to just the right distance from the PCB, such that each of the signals clipped slightly at their maximum amplitude. This adjustment was made while viewing the waveform on an oscilloscope until the waveform resembled that of Figure 8b. After this adjustment, each encoder provided about 1600 counts per rotation, which is more than adequate precision for traversing the micromouse maze.



(a) Poorly Conditioned Encoder Output (b) Properly Conditioned Encoder Output

Figure 8: Encoder Outputs Before and After Conditioning

2.3 Sensor Calibration and Signal Post-processing

In order to ensure that the sensors chosen could represent the environment most accurately, we implemented calibration and post-processing schemes to increase the quality of sensor data and to reduce noise in the signal. The purpose of the calibration routine, which is called when the robot is first placed into the maze, is to remove any bias the lighting in a room, the quality of walls, or any other transient factor has on sensor data. We assumed our sensor values were Gaussian random variables, and used the calibration routine to estimate each sensor's distribution. This consisted of sampling the left and right sensors 50 times, computing the sample mean and variance, and normalizing each successive reading during the robot's traversal of the maze to these parameters. The parameter estimation and normalization process is described in the following equations.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1}$$

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{2}$$

$$x_{norm} = \frac{x_i - \mu}{\sigma} \tag{3}$$

In addition, the normalized sensor data was post-processed to ensure consistent measurements are obtained. This required filtering high frequency noise by using a low pass filter. The most simple and memory-efficient filter to implement was the Exponentially Weighted Moving Average (EWMA). The output of this filter, Y_t, is the weighted average of the current input, X_t and each of the previous inputs exponentially decreasing weight:

$$Y_t = \alpha\sum_{i=0}^{N}(1-\alpha)^i X_{t-i} \tag{4}$$

$$Y_t = \alpha X_t + (1-\alpha)Y_{t-1} \tag{5}$$

Equation 4 requires that all previous data points be stored in memory. However, the filter can be rewritten recursively, as in equation 5. This formulation requires only the previous result to be stored, and saves computational resources as well as memory. In both cases, the value of α can range from 0 to 1, and determines the size of the filter's time window. A value of 1 corresponds to a window of just one point (all previous inputs are ignored,) while a value of 0 weights all observed inputs equally. Thus, a lower value of alpha effectively increases the window of the filter, and increases the SNR of the output at the cost of decreasing the response time of the sensor system. An α value of 0.75 was experimentally determined to yield the best data from the sensors.

2.4 Detection Schemes

After achieving quality sensor readings, a scheme for detecting walls in the robot's surroundings was chosen. Several detection schemes are used in AMR applications, one of which is received signal strength intensity (RSSI). This scheme involves establishing a voltage thresh-

old for detection based on the distribution of sensor values when a wall is present and when a wall is not present. We found that a threshold of 1.2σ was an effective choice for minimizing the probability of detection error. Choosing a threshold that is dependant on the distribution estimated in the calibration phase allows the threshold to be automatically adjusted based on lighting conditions at the time of competition. This detection system was very effective for an amateur micromouse robot.

A more involved way of detecting walls is differential phase shift of arrival (DPSA), which can be seen in Figure 9. DPSA begins with a sinusoidal signal in the MHz range being modulated through the IR emitter. At the receiver, the signal is passed through a high pass filter to remove noise, passed through an I/Q demodulator, and finally differentiated to find the phase of the real and imaginary channels. The difference in phase, combined with the intensity at the receiver, can be used to detect walls [6].

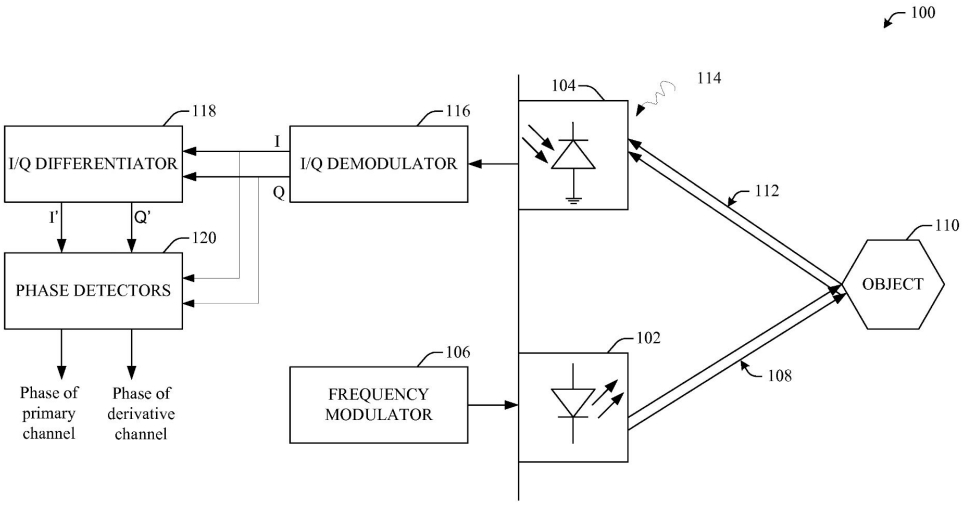


Figure 9: Differential Phase Shift of Arrival [6]

3 Movement System

An AMR movement system must allow the robot to navigate the environment precisely and quickly. Based on observing previous Micromouse competition submissions [7] [8], it is clear that a precise system for physically navigating the terrain can prevent the robot from getting stuck in the maze — one of the most common problems in this competition. Several design elements were considered in order to choose a movement system. These included the size and quantity of wheels; the overall structure of the locomotion system; the size, torque, and efficiency of motors; and the types of feedback mechanisms that guide motion.

3.1 Wheels

Many robots employ a wheeled locomotion system over one that employs legs, treads or other systems [9] because they are easy to implement, and they provide better acceleration and traction while being less prone to slippage.

Three main types of wheels are available: the standard wheel, the caster ball, which can roll in any direction, but cannot be powered, and the omni wheel, which can actually be powered and allow rotation or translation in nearly any direction. The choice of wheel type depends on the choice of locomotion structure. However, certain characteristics of wheels are important regardless of this choice. The drive wheels must be coated in a rubbery material that minimizes the risk of slippage. In addition, the drive wheels must have some degree of built-in suspension in order to maintain a straight course and accommodate any imperfections in the floor while navigating the maze. Finally, the diameter of the wheels is an important trade-off: larger wheels allow for greater torque, but sacrifice precision of motion, and may increase the risk of slippage [3].

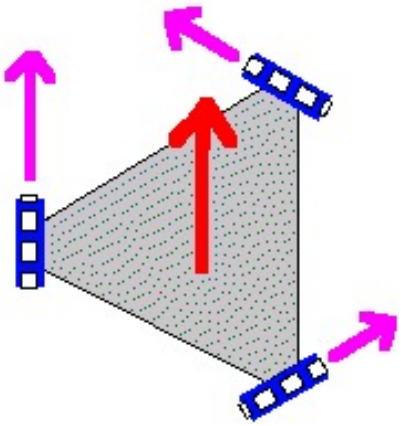
The Tamiya 70111 Sports Tire Set was chosen for the first iteration of the robot (Figure 10a). This included wheels with a 56 mm diameter and 25 mm width that have tacky rubber tires with treads. These tires were very unlikely to slip, and can be compressed approximately 5 mm by external forces, meaning that they were able to negotiate cracks, ledges, and other artifacts on the maze floor easily. However, after constructing the first iteration of the model, it was determined that the Tamiya wheels were too wide and restricted movement when turning. The robot was designed to turn in place, but with a diagonal length greater than the distance between walls, 18cm, it would get caught and was unable to move. In an effort to rectify this issue, wheels with a diameter of 56mm were laser cut from $\frac{1}{4}$ inch thick acrylic (Figure 10b). Since acrylic alone provides very little traction, a small groove was cut into the wheel and a rubber O-ring was fitted onto the wheel. The new wheels are thinner, and actually provide more traction than the Tamiya wheels.

(a) Tamiya 70111 Sports Tire (b) Custom Acrylic Wheel

Figure 10: Micromouse Wheels

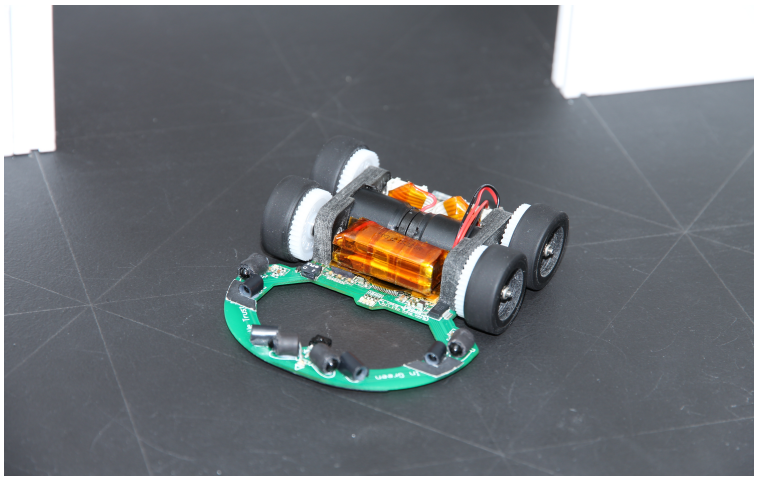
3.2 Locomotion System Structure

The structure of ground robot locomotion systems varies greatly, but can generally be divided into two groups: legged and wheeled locomotion. Given the size and time constraints of this challenge, wheeled locomotion was the clear choice for a fast, precise robot. However, several configurations of wheeled locomotion exist, each of which have distinct advantages. Three different configurations were candidates for this design: a three wheel design composed of omni wheels allowing synchronous drive, as in Figure 11a, a three wheel design composed of two standard wheels and a caster ball in a tricycle configuration, as in Figure 11b, and a traditional design composed of two rows of standard wheels, as in Figure 11c.



(a) Three Omni Wheel Configuration [10]

(b) Caster Ball Configuration [11]



(c) Standard Four Wheel Configuration [2]

Figure 11: Common AMR Locomotion Structures

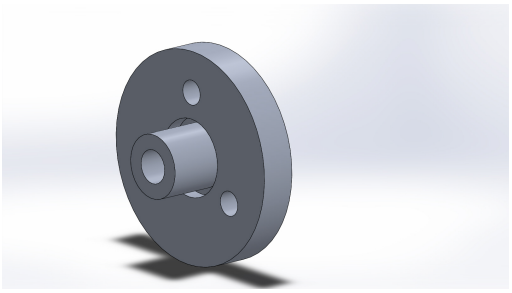
The advantage of the omni wheel configuration is that it allows holonomic motion (the ability to move freely with two degrees of freedom), which can prevent the robot from getting stuck in the maze. However, this configuration is also difficult to control, and requires additional sensors to keep track of the orientation of the robot.

The second configuration is the easiest to control because each of the two wheels are driven by separate motors, and the caster ball rotates freely, eliminating problems of synchronizing three or more wheels. The disadvantages of this structure are the increased risk of slippage and the inability to make quick, precise turns.

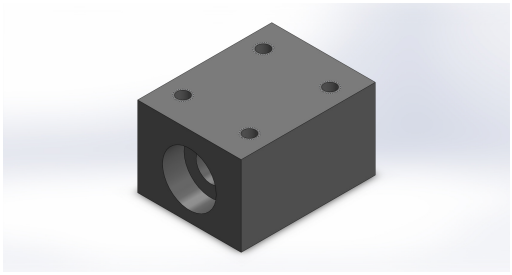
We chose to implement the standard four-wheel configuration, as this is the system of choice for the most competitive micromouse robots in the world. This system is difficult to configure, as each of the four wheels has to be aligned carefully, and must be precisely the same size. However, if implemented well, it allows for the fastest movement without sacrificing precision [3] [12].

In designing the four-wheel configuration, we chose to use two motors, one on each side of the the robot, as can be seen in Figure 12c. Each motor shaft had a 50-tooth gear attached to its shaft with a set screw, which was fit to two 100-toothed gears that were attached to keyed 5mm aluminum shafts with their own set screws. This 2:1 gear ratio was chosen to translate the speed of the motors to a robot movement speed of $0.9\frac{m}{s}$.

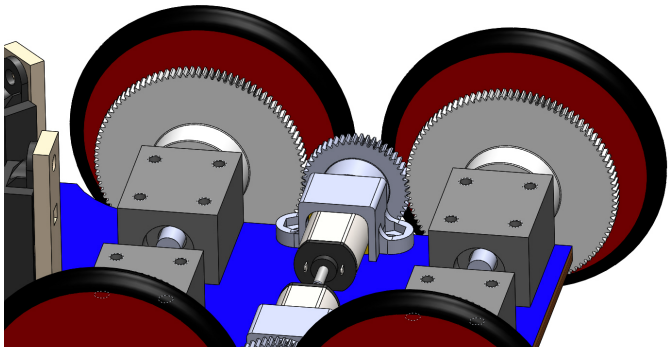
Aluminum hubs, as can be seen in Figure 12a were designed to attach the acrylic wheels to the aluminum shafts. The hubs are attached to the shafts with set screws. The shafts are press fit into ball bearings, which in turn are press fit into the aluminum pillow blocks that can be seen in Figure 12b. Each pillow block has two bearings to ensure that the shaft remains horizontal. The bearings we chose had a 5mm inner diameter, and consisted of two cylinders with tiny, free-moving metal balls in between. Bearings usually have two cylinders separated by tiny spheres which rotate in place. The bearings we chose were of lower quality, slightly less stable, and occasionally produced uneven amounts of friction between the wheel and shaft. However, they were cheap and were sufficiently friction-free for the robot we implemented. Once these components were chosen, the pillow blocks and bearings were aligned and was screwed onto the chassis.



(a) Wheel Hub



(b) Pillow Block



(c) Drivetrain Configuration

The process of designing the drivetrain described here was very involved because each of the components was designed from scratch. This provided a robust movement system, but was not an efficient use of our time for the micromouse competition. Based on this experience, we suggest that amateur micromouse competitors choose a two-wheeled design with DC or stepper motors, and implement it largely with off-the-shelf drivetrain components from robotics distributors.

3.3 Motors

Choosing a motor for our robot entailed a trade-off among drive power, power consumption, and weight. Drive power is an especially important for an AMR that must move fast. In this case, the micromouse was required to reach the destination square quickly in order to be competitive. However as drive power increases, power consumption and weight usually increase as well. Three types of motors were compared as candidates for our robot: stepper motors, servo motors, and DC motors.

Stepper motors are brush-less DC motors that divide a full rotation into a series of discrete "steps" that are controlled by the polarity of a number of electromagnets around the perimeter of the main rotor. The advantage of using a stepper motor is that stepper motors

are very precise, and eliminate the need for encoders or other feedback sensors to determine the distance traveled. However, these motors are significantly larger, heavier, and more power-hungry than servo or DC motors with equivalent torques, as can be seen in Table 2.

Servo motors are similar to stepper motors in that they have a simple built-in control system, however they use potentiometers and a series of gears rather than electromagnets. This allows servo motors to operate much faster than stepper motors, as can be seen below.

DC motors are used in the most competitive micromouse robots, because they provide by far the highest power-to-weight ratio. A major challenge associated with these motors, however, is that a control system must be implemented to track the actual power output by each motor. This control system usually consists of a pair of IR encoders, as well as an extra layer of computation on the micro-controller to account for encoder inputs.

Specification	**RB-Soy-20 Stepper Motor**	**Hitec HS-55 Servo Motor**	**Pololu DC Micro Gear-motor**
Torque [oz-in]	13	18	15
Weight [g]	140	8	10
Operating Voltage [V]	4.5	5	6
Free Run Current [mA]	670	150	100
Stall Current [mA]	670	150	1600
Free Run RPM	-	61	630

Table 2: Comparison of Common AMR Motors

Despite the challenges associated with DC motors, they were the only choice for a competitive micromouse robot because of the high power-to-weight ratio they provide. We chose to use Pololu DC Micro Gear-motors for our robot, because they are compact, yet provide a great deal of torque and drive power. An additional advantage of these motors is that we were able to use the IR encoder set that Pololu manufactures specifically for these motors.

4 Control System

The control system for an AMR must translate sensor readings and knowledge of the environment into decisions. For the micromouse competition, the control system must learn the layout of the maze by pushing sensor data through a decision algorithm and updating an internal representation of the maze. Then, it must identify the move that will move the

robot to the center of the maze most efficiently. Finally, the control system must initiate and control the movement process by applying drive to motors to move the robot while accounting for slippage and drift by integrating encoder readings. These needs were accommodated through a combination of a relatively inexpensive micro-controller and intelligent control algorithms.

4.1 Micro-controller

The micro-controller used to drive motors and process sensor data needed to be powerful enough to handle high data rates. Sensors on a micromouse are usually sampled at a rate between 0.2 kHz and 1 kHz, and the microprocessor must be capable of reading the sensor values (performing an A/D conversion), deciding where to move based on this information and on the location in the maze, and applying signals to each drive motor – all within a short time cycle in the range of 10's of milliseconds. In addition, solving the maze requires significant processing power and memory in order to represent the maze as a graph and quickly apply decision algorithms.

In order to accommodate these processing needs, a micro-controller such as the STMicroelectronics STM32F103RET6 is commonly used in Micromouse competitions. The specifications for this processor can be seen in Table 3: In addition to using (relatively) high powered processors, the most competitive Micromouse robots use custom PCBs to connect the processor to sensors, motors, and other system elements. We chose to start designing our robot with an Arduino Uno as the micro-controller, because it is very simple to interface with other components. The Arduino Uno has an ATmega328 processor, which we determined to be sufficiently powerful for our robot. The ATmega328 specifications can also be seen in the table below. The greatest advantage of using an Arduino micro-controller was the high quality of documentation that exists on the hardware as well as the numerous software libraries written for it. We were able to use the various on-line forums to quickly find solutions to issues we encountered. However, one disadvantage we encountered was the 2 KB of RAM. We found that this memory requirement was barely enough to store the various data structures we used to represent the environment and handle the computations needed for the algorithms we implemented.

Specification	STM32F103RET6	ATmega38
Data Bus	32 bit	16 bit
Clock Rate	45 - 72 MHz	16 MHz
A/D Converters	12 bit	10 bit
D/A Converters	12 bit	10 bit
RAM	64 KB	2 KB
Program Memory	512 KB	32 KB
Supply Voltage	5 V	5 V
Power Consumption	125 - 250 mW	100 - 150 mW
Analog Ports	16	6
Digital Ports	45	14

Table 3: Micro-controller Specifications [13], [14]

4.2 Alternate Architecture: Processing Sensor Data

An alternate architecture, involving two processors, could have been used to accommodate the data rates involved with sensor readings. In this scheme, a slave processor can be tasked with sampling sensors at around 1 kHz and reporting post-processed sensor readings to a master processor. For five sensors (three distance sensors and two encoders), this sampling rate translates to a data rate of 160 kb/s, assuming each reading is stored as a C `int`. Simple post-processing, such as a weighted average low pass filter with a 10 sample window, as explained in Section 2.3, can also be applied on the slave processor. The master processor could then read aggregated sensor readings at around 16 kb/s, a much lower data rate. This scheme would have allowed the master processor to allocate more computational resources to controlling locomotion and applying maze solving algorithms. However, a second processor would have increasee the power consumption of the control system, and taken up a significant amount of space on the chassis.

4.3 Kinematic Model

Many AMR systems implement a kinematic model to track the location of a robot relative to its environment. We explored one such model, as presented by Konolige in "Robot Motion: Probabilistic Model; Sampling and Gaussian Implementations; Markov Localization" [15]. This model consists of a set of kinematic equations that relate the signals applied to the driving motors to the motion of the robot. These equations account for the dimensions of

the robot, as well as the size of maze squares, and are abstracted away from the maze solving algorithm (i.e. the code for the maze solving algorithm can call functions like *turnLeft* and *turnRight*).

This kinematic model is an odometry-based model, meaning it uses the data from encoders to track the movement of each wheel. This data can then be used as input to the equations below to accurately determine the orientation of the robot as it navigates through the maze. The following equations describe the change in position of the robot between two samples of encoder readings in the global coordinate space. [15]:

$$\Delta s = \frac{\Delta r - \Delta l}{d} \tag{6}$$

$$\Delta \theta = \frac{\Delta r + \Delta l}{2} \tag{7}$$

$$x' = x + \Delta s \cos\theta \tag{8}$$

$$y' = y + \Delta s \sin\theta \tag{9}$$

$$\theta' = \theta + \Delta\theta \tag{10}$$

In this model, Δl is the translation of the robot, Δr is the counter-clockwise rotation of the right wheel relative to the left wheel, $\Delta\theta$ is the angle of this rotation, and d is the distance between the left and right wheels. A depiction of these parameters can be seen in Figure 13. If this model was implemented and tested, an additional parameter could have been added to increase its accuracy. The output of each encoder is inherently noisy, but this noise can be modeled as a random variable. In the most generic case, the noise that affects each encoder, k, can be modeled as Gaussian, and the parameters μ_k and σ_k can be estimated using maximum likelyhood (ML) estimation during testing. Then, a random sample from $\mathcal{N}(\mu_k, \sigma_k)$ can be added to each reading reading from the kth encoder. This model can allow the control system to train on a more accurate kinematic model, which could improve the overall decision making algorithm. In order to effectively solve the maze, a locomotion control system that is precise to $2-3mm$ will be necessary [3] [12], and each of these additions attempt to bring the precision closer to that value without sacrificing speed.

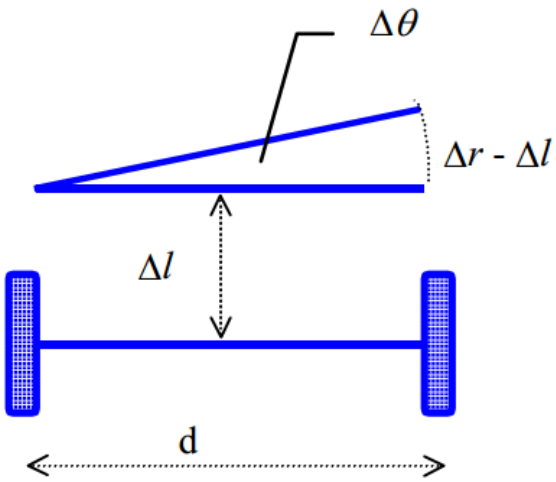


Figure 13: Kinematic Motion Model Parameters [15]

4.4 Motor Control Hardware

A standard mechanism for providing current drive to one or more motors in robotics is an H-bridge. An H-bridge is a switching circuit that allows for voltage to be applied across a load (in this case a motor) in either direction. For our drive train, the Texas Instruments L293D Quad Half-H Driver chip was chosen. The specifications of this chip-set can be seen in Table 4. The drive train configuration of the the micromouse, as outlined in section 8, was required a drive current of 100 mA for each of the two motors. The L293D could easily satisfy this requirement.

Specification	**Value**
Channel Output Current	600 mA
Peak Output Current	1200 mA
Supply Voltage Range	4.5 - 36 V

Table 4: L293D Quad Half-H Driver Specifications [16]

4.5 Motor Control Algorithms

As a first step, we decided to have our robot move one square at a time instead of continuously, as the top micromouse robots do. This allowed us to fine tune our movement and control systems. Though we never implemented continuous motion, we did refine two important algorithms that were used to drive motors in a controlled manner as the robot

navigated through the maze. First, a ramp algorithm was used to provide linear acceleration as the robot moved from square to square:

$$v_{new} = \frac{d_{square} - d_{enc}}{d_{square}} v \tag{11}$$

The simple ramp-down function returns the new motor power, v_{new}, as the product of the current power, v, and the ratio of the distance left, $d_{square} - d_{enc}$, to the size of a square, d_{square}.

In addition to slowing down upon entering a new cell, a control algorithm was used to keep the robot centered in the square it was in and prevent it from colliding with walls. The algorithm adjusted the speed of the motors based on the a sigmoid function:

$$v_{adj} = \frac{1}{1 + e^{-x}} - \frac{1}{2} \tag{12}$$

The inputs to Equation 12 were each of the normalized sensor values from the left and right sensors, and the output was the value by with the power to each each motor was adjusted. For example, an input value of 0 from the left sensor indcates that the right wall is exactly as far from the robot as it was when the sensors were calibrated. The output from the sigmoid function for this input is also 0, which correctly identifies that no adjustment is required. However, if the input value from the left sensor is greater than 0 (indicating that the robot is too close to the right wall), the output negative, and will apply negative feedback to the left motor. This will move the robot farther from the right wall, and closer to the center of the square. This process can be seen in Figure 14, which illustrates the possible inputs for a zero-mean, unit variance sensor input, and the corresponding outputs from the sigmoid function.

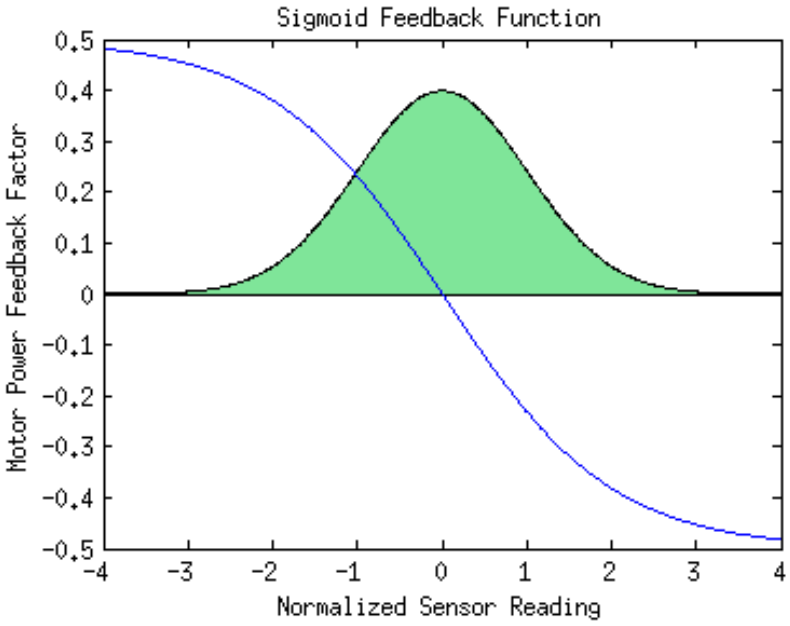


Figure 14: Sigmoid Feedback Function

5 Maze Solving Algorithms

The ability of the micromouse to find the shortest path to the center of the maze is a critical component of the competition. In fact, the problem of solving an arbitrary maze is the simplest component of the Micromouse, as this problem can be solved by several algorithms. The simplest solution to solving a maze is to follow the right or left wall of the maze, however Micromouse mazes are designed specifically so that this algorithm will not lead to the solution [17]. Two advanced algorithms are used commonly in Micromouse competitions - the Flood Fill and Shortest Path algorithms. These algorithms require additional memory and processing power, but these can easily be accommodated with modern micro-controllers.

5.1 Maze Representation

Before a maze solving algorithm could be implemented, the robot needed a way of representating the maze in memory. When the robot is first placed in the maze, the representation of the maze should contain only the boundary of the maze, and update wall locations inside the maze as it detects them. Initially, the maze was represented with a 16 by 16 array of C *int*s. Each position in the 2-dimensional array represented a cell in the maze, and the value at that position represented which positions in that cell actually had walls. Since there are

only 4 possible wall locations per cell, only 4 bits were needed. This meant that a large portion of the array of *int*s was unused. Therefore, the representation was modified to be an array of *unsignedchar*s where each cell only occupied 4 bits as opposed to 16 bits. Each *unsignedchar* held the wall information for two cells, giving a total array size of 128 Bytes, instead of the 1024 Bytes that an integer array would have occupied. This representation was implemented as a class member under the Maze namespace, and methods to get and set the wall information for a cell were written to make this choice of representation modular.

In retrospect, a better way of representing the maze would have been to use a graph where each node represented each cell and each edge represented a path to an adjacent cell. As the robot moved through the maze, edges would be removed if a wall was found between cells. This maze representation would have made it easier to implement more complex maze solving algorithms, but would have taken

5.2 Shortest Path Algorithm

The Shortest Path algorithm is a solution to the maze problem, and can be implemented with a breadth-first search using the A* algorithm, a common artificial intelligence algorithm. The basic idea of this method is to visit squares of the maze in increasing distance from the starting square until the goal square is reached. Squares can be tracked using a memory structure called a queue.

This algorithm is guaranteed to find the shortest solution to the goal state. Once this solution is found, the robot can return to the starting square, resetting its time, and make a second trip straight to the goal square. This technique of making repeated trips to the goal square is implemented by almost all competitive Micromouse robots, but requires that the robot be fast and have a low probability of getting stuck, because the robot must traverse many paths in the maze repeatedly [18]. The biggest challenge with this algorithm is the extra space needed to implement the algorithm, since it uses more complex data structures.

5.3 Flood-fill Algorithm

Another algorithm used to solve the mazes is The flood fill algorithm, which is actually an image processing method used to fill in connected, similarly colored areas. It can also be used to solve mazes by treating each square of the maze as a pixel. Under this assumption, four cases exist:

1. All four boundary squares are filled - the maze is solved
2. Three of the boundary squares are filled - only one path leads out of the current square
3. Two or less boundary squares are filled - choose one of the paths leading out of the square at random, and mark the current square (in memory) as a square that has been visited

Using these scenarios, a recursive function was developed to solve a maze by tracing out more and more of the maze until the solution (case 1) is reached. This algorithm is guaranteed to find a solution to the maze, but it may not be the optimal solution. This solution was simulated on a desktop to ensure proper function. When porting to the microcontroller, the recursive approach to solve the maze worked well for smaller mazes, but when tested on larger mazes the Arduino Uno would experience stack overflows, which corrupted other parts of memory not associated with the maze. So, rather than exploring each node (cell within the maze) with a separate function call, an iterative approach was implemented which stepped through nodes placed on a queue. As long as nodes popped from the queue had unexplored neighbors, the neighboring cells were placed on the queue. This improvement made the algorithm linear in space complexity and improved its speed.

6 Chassis

The chassis of an AMR must hold the robot's subsystems together rigidly. For the application of a maze solving robot, the chassis must also balance the weight of the power supply, motors, micro-controller, and sensors while preventing the robot from snagging on corners or toppling during turns. In addition, the chassis must account for potentially uneven terrain by having a slightly flexible body. Competitive robots use the same custom PCBs on which their sensors, micro-controller, and other components are mounted as the chassis of the robot, keeping the center of mass low, and maintaining a narrow profile to be able to take wider turns at high speeds. However, acrylic and ABS plastic are also common materials for Micromouse chassis, as they can be cut with common shop tools. The following table compares the mechanical properties of popular material choices.

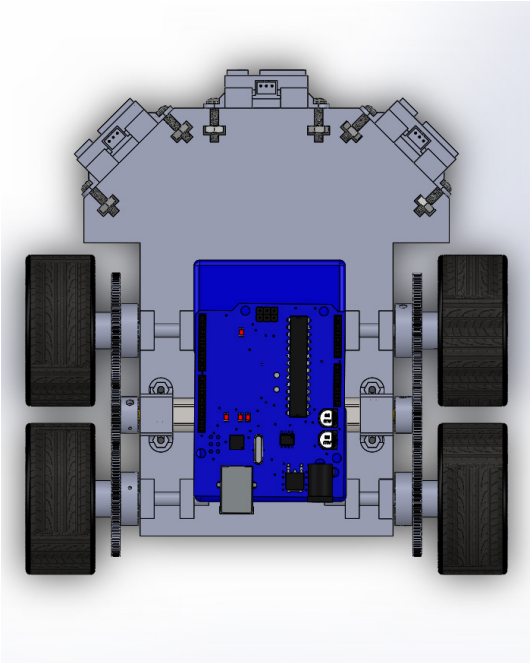
Material Property	**Acrylic**	**ABS plastic**	**PCB**
Tensile Strength (PSI)	10,000	4,100	45,000
Flexural Modulus (PSI)	480,000	304,000	65,000
Izod Impact ($ft \cdot lb/in$)	0.4	7.7	8
Density (g/cm^3)	1.17	1.06	1.85

Table 5: Comparison of Chassis Materials

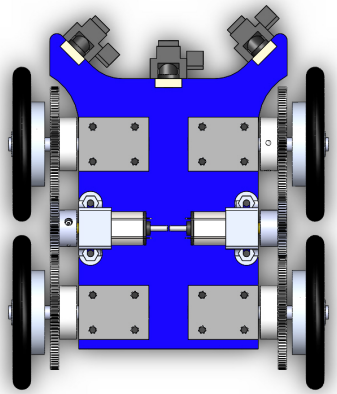
The comparison in Table 5 shows that acrylic has a higher tensile strength and flexural strength than ABS, but that it has a lower Izod Impact than ABS. This suggests that acrylic is a stronger material, but that it is brittle and can crack easily. As can be seen in Figure 15a, the chassis design requires numerous sharp cuts. Acrylic is likely to crack while fastening components to these areas of the chassis, while ABS plastic is a softer material, and can accommodate the pressure. Compared to acrylic, ABS plastic is a better choice for this design, as it provides the rigidity required for a stable chassis as well as the flexibility to handle potentially uneven terrain.

For the first prototype, a piece of ABS plastic 13cm x 15cm was laser cut to the design in Figure 15a. This design balanced the weight of the subsystems very well, and served as a good testbed for the locomotion system. However, the size of this chassis was far too large - the robot could not turn in place without hitting the walls of the maze. So, the chassis was redesigned to minimize the space the robot took up. In the process, several sub-systems needed to be modified. The Tamiya tires were replaced with thinner, custom-made wheels, the hubs that attached these wheels to the chassis were cut to be thinner, and the orientation of sensors was changed to suit a narrower chassis.

The final chassis design, wheels, and hubs, which can be see in in Figure 15b, occupy a total of 11cm x 13.3cm. In addition, two layers of the base chassis were used to provide additional structural integrity to the robot. The top layer was screwed on top of the base layer, with the motors and pillow blocks in between. The power source, electronic components, and micro-controller were placed on top of the upper layer. The layers of the chassis were prototyped from 1/4in wood, as the tools for cutting ABS in the precise shape needed were not available (ABS releases various carcinogens if cut with a laser).



(a) Initial Chassis Design



(b) Final Chassis Design

Figure 15: Chassis Design

Ideally, a custom PCB would have been printed to obtain similar specifications and eliminate stray wiring on the robot. Although common PCB materials have nearly double the density of ABS, printing a custom PCB eliminates the weight of a separate micro-controller board. However, this choice of chassis served as a strong first step that allowed the testing of other robot subsystems.

7 Power Source

Autonomous robots all depend on long-lasting batteries to effectively carry out their function. Three common options for rechargeable batteries were compared, as can be seen in Table 6:

Specification	**NiCd**	**NiMH**	**Li-Ion**
Relative Power/Weight Ratio	1	1.4	3
Memory Effect	Noticeable	Little Effect	No effect
Self-Discharge (charge/mo)	20%	30%	3%

Table 6: Battery Comparison

Based on these specifications, it is apparent that a lithium-ion battery was the best choice for the micromouse due to its greater Power/Weight Ratio and low self-discharge rate. An additional advantage is that Li-Ion batteries do not experience the memory effect. The memory effect is the process by which the capacity of a battery is reduced by repeatedly charging the battery before it has been completely discharged. Our robot was tested rigorously, and having a battery susceptible to the memory effect would have greatly hindered performance.

An important consideration as we chose a specific battery for our robot was battery capacity. The battery needed to power the Arduino Uno, two DC motors, three IR distance sensors, adn two encoders for at least 10 minutes without dropping significantly in voltage. Each of these sets of components can be powered with a 5 V source, and draws up to 50 mA, 2 $\times$ 100 mA, and 3 $\times$ 30 mA, and 2 $\times$ 24 mA respectively. This amounts to a total current of 384 mA as an upper bound. A pair of 3.7 V Lithium Polymer batteries, each with a capacity of 3000 mAh, were used for the final micromouse. These provided more than enough power for our robot, but were relatively heavy at 55 g each.

8 Results

After decomposing the problem of building an autonomous mobile robot to solve a maze, researching the best practices for each of the subsystems, and testing several prototypes, our final robot was a capable amateur maze solving robot, which can be seen in Figure 16.

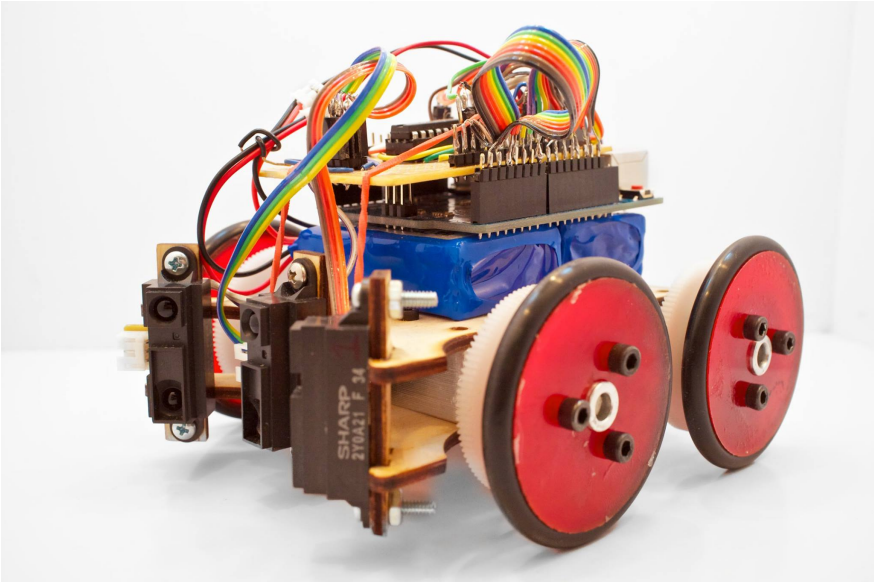


Figure 16: Final Robot

The Sharp IR sensors that we chose to use provide more than adequate precision and a high enough sampling rate (62 Hz) to quickly and precisely observe the environment. By configuring our sensors in an inverse orientation, we avoid issues that derived from the non-invertible response of the sensors. Furthermore, by normalizing and low pass filtering the distance sensor data, we removed significant sources of noise and provided clean signals for our control system to use. The IR encoders that we used to track the motion of the two motors provided very precise information about the rotations of each motor (about 1600 ticks per rotation).

Though designing our movement system took too much time out of the design process, our final structure is robust. The robot can move at up to 0.9 $\frac{m}{s}$ giving it the ability to accelerate and turn quickly. The chassis, as well as the choice of motors, gears, and wheels, provide a rigid body that holds the other subsystems in place. However, in retrospect, a two-wheeled design would have been easier to implement and would have allowed more effective turning.

The sigmoid control system that was implemented to keep the robot in the center of its path as it traversed the maze is effective, but additional control methods should have been explored. The maze solving algorithm used to reach the center of the maze was perfectly sufficient for our robot, but a graph-based algorithm may have scaled better if the robot was improved further.

Our robot had an unfortunate technical issue at the time of the competition, and did not complete the maze. Nonetheless, our work provides a guide for future amateur micromouse competitors.

Our code can be accessed at `https://github.com/SharangP/megamouse`, and a video of a test run of our robot can be found at `http://youtu.be/qaBy-hSmXGg`.

9 Conclusion

The IEEE Micromouse competition is an engineering challenge that requires the application of precise hardware design as well as efficient software algorithms. This project proposes a design for a robot that solves the competition maze as quickly as possible by integrating a durable and powerful drive train, efficient control systems, and precise sensor systems. The prototype detailed in this paper was constructed, tested rigorously, and improved over several iterations. This report serves as a guide for future micromouse competitors, outlining the advantages and disadvantages of various design decisions.

10 Acknowledgements

We would like to thank several individuals for their advice and support of this project. In particular, Toby Cumberbatch, the advisor for this project, helped maintain focus of the most crucial components of the project. In addition, Sinisa Janjusevic, Jeff Hackner, Jonothan Ostrander, Daniel Baamonde, and Vasily Kuksenkov provided invaluable advice on the design of our robot. Finally, we would like to thank our peers in the Cooper Union Class of 2014 for their support.

11 Bibliography

[1] Evan Ackerman, "Meet the New World's Fastest Micromouse Robot," Nov. 2011. [Online; accessed 12-22-2013].

[2] Luzhou Ye, "Green Ye's Personal Website." `http://www.greenye.net/` [Online; accessed 12-14-2013].

[3] Birmingham City University, "Micromouse Guide," 2013. [Online; accessed 12-24-2013].

[4] E.M. Gorostiza et al., "Infrared Sensor System for Mobile-Robot Positioning in Intelligent Spaces," *Open Access: Sensors*, 2011.

[5] *GP2Y0A21YK:Optoelectronic Device*, 2005.

[6] H. I. Intersil Americas Inc., Witter D.W., "Distance sensing by iq domain differentiation of time of flight (tof) measurements," Aug 2011.

[7] Jake Edgar, "Micromouse - 1st Place IEEE Region 1 Competition 2013." Youtube, Aug. 2013. `http://www.youtube.com/watch?v=hAHw0VYsi1c` [accessed 11-22-2013].

[8] Peter Boothe, "IEEE Region 1 MicroMouse competition 2012 - winning team." Youtube, Mar. 2012. `http://www.youtube.com/watch?v=DuG-WF9yT5c` [accessed 11-22-2013].

[9] Sven Bottcher, "Principles of Robot Locomotion." Seminar on Human Robot Interaction SS2006, Univ. of Dortmund, 2006A.

[10] "OMNI-WHEEL ROBOT - FUZZY." Society of Robots. `http://societyofrobots.com` [Online; accessed 12-10-2013].

[11] "DFRobot 2WD Mobile Platform for Arduino." Robotshop Distribution Inc. `http://robotshop.com` [Online; accessed 12-14-2013].

[12] J. Xiao, "Advanced Mobile Robotics: Locomotion/Motion Planning/Mapping ." Dept. of Electrical Engineering, CUNY City College. [Online; accessed 11-24-2013.

[13] "STMicroelectronics STM32F103RET6." Mouser Electronics Inc. `http://mouser.com` [Online; accessed 12-10-2013]".

[14] Arduino, "Specs Compare," 2013. [Online; accessed 12-10-2013].

[15] Kurt Konolige, "Robot Motion: Probabilistic Model; Sampling and Gaussian Implementations; Markov Localization," 2001. SRI International.

[16] "L293, L293D Quadruple Half-H Drivers." [Online; accessed 12-23-2013].

[17] IEEE Region 1 Student Conference, *Micromouse Competition Rules*, 2013.

[18] W. D. Pullen, "Maze classification," Apr. 2013. [Online; accessed 12-10-2013].

12 Appendix 1: Competition Specifications

The rules for the IEEE Micromouse Competition, as specified in [17], specify constraints that submissions must adhere to, as well as how the competition will run. Each submission must be a self-contained robot no larger than $25cm$ in either length or width. The standard maze is a 16 x 16 grid composed of $18cm$ x $18cm$ squares, with $5cm$ high walls each with a thickness of $1.2cm \pm 5\%$. These specifications are detailed in Table 7:

Specification	**Value**
Maximum Robot Width	$25cm$
Maximum Robot Length	$25cm$
Maze Wall Height	$5cm$
Maze Wall Thickness	$1.2cm \pm 5\%$
Maze Square Size	$18cm$ x $18cm$
Maze Grid Size	16 x 16 squares

Table 7: IEEE Micromouse Competition Specifications

Starting in one of the four corners, each robot attempts to solve the maze alone, with the goal of reaching the four center-most squares, which have no walls separating them and a $20cm$ high post at their center. As the robot traverses the maze, it cannot jump over or damage walls, nor can any of its parts detach from the main chassis.

Each team has a total of 10 minutes to access the maze, during which time it may attempt as many runs as possible. A successful run entails the robot starting in the designated corner of the maze and maneuvering to the destination (center) square. The run with the shortest time is the robot's official time, and the shortest overall time is awarded first place. If no mice reach the destination square, the robots are ranked by the number of unique cells that each of them passes through without being touched by a team member. If a team member does touch the robot (for example in the event of a collision), a 30 second penalty is applied to the total time remaining when the mouse is placed back into the maze.

The standard Micromouse competition strategy is to first explore the maze until the optimal path to the center is discovered. Then, using that information, several attempts can be made to reach the center of the maze as quickly as possible, increasing the robot speed (and the risk of error) each time.