Arduino Maze-Traversing Robot
ICS4U Summative Project

Miasya Bulger & Charles Zhang
January 12th, 2017

Table of Contents

Initial Proposal

(as submitted on October 11th, 2016)

Our summative project will be to develop an Arduino robot capable of autonomously traversing and mapping a maze. Through this project, we will be applying the programming concepts learned in class while improving our project management and collaboration skills. An Arduino project specifically will also allow us to explore our interests related to different technology-based careers such as hardware and electrical engineering. Due to our limited exposure to robotics, we will also gain valuable problem-solving experience. Lastly, a huge advantage of doing an Arduino project is that we will never run out of possibilities! In the case that we have extra time, we can always add more functionality or improve the robot's design using 3D printing or laser cutting equipment.

Equipment needed:

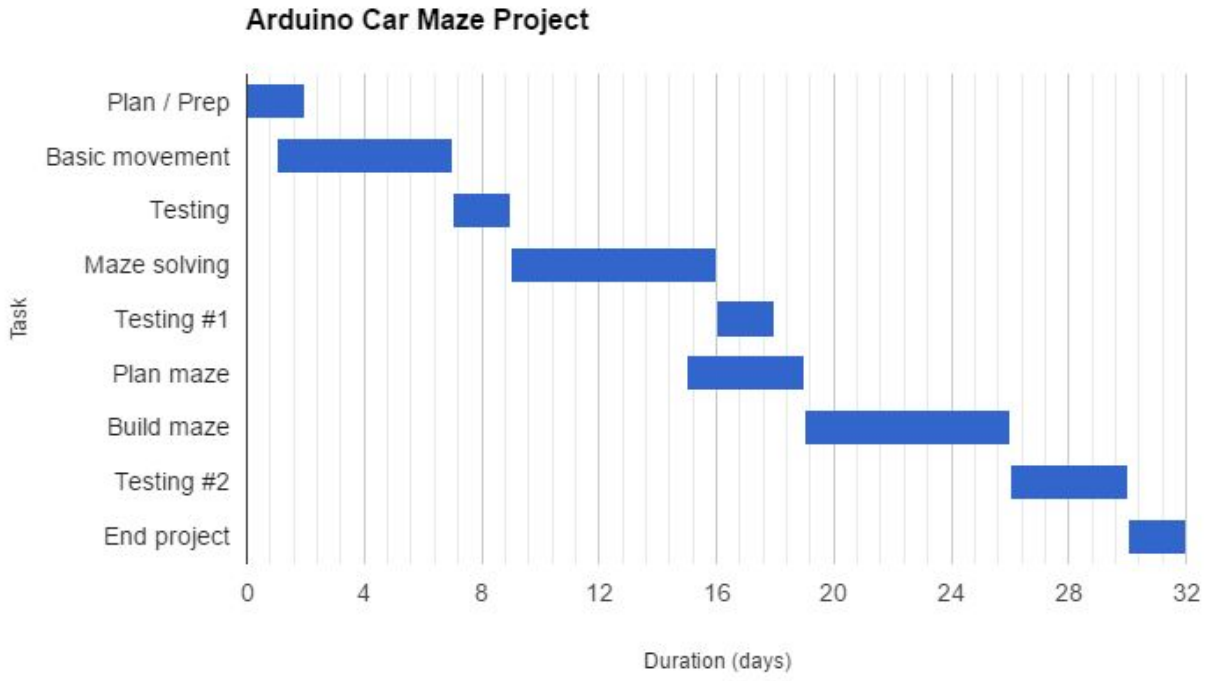
- Laptop with Arduino IDE
- Arduino or Pi
- Two Motors and Wheels
- Battery box
- 4 AA batteries
- USB power pack
- Breadbox
- Motor control
- Resistors
- Cables
- Chassis - 3D printed or laser cut

Project Management Tools
Gantt Chart

Why We Made a Gantt Chart

We created a Gantt chart to help us illustrate the timeline for our project, see the periods of overlap between tasks, and to let us evaluate the float of each task and be aware of how a delay would affect other tasks.

We also considered creating a PERT Chart (Planning, Evaluation, Review, Technique) to help us identify events, construct a visual aid, estimate time, then prioritize based off the chart. However, since the project was only a few weeks long, we found that we would would be wasting more time making the PERT chart than we would be saving from it. This is especially true considering that the key aspects, such as the sequence of tasks, the visual aspect, and the estimation of time, were already covered in our Gantt chart. The big advantage that a PERT chart offers is analysing resource management, but for our project, we did not have the need for it.



Project Management Tools
Software Scope Document

Why We Created a Scope Document

We wrote a scope document as a part of the project process. A scope document defines the project scope, which includes any expectations and objectives of the project, as well as specifying the deliverables, functionality, and project boundaries. The main reason we believed that a scope document would be important for us was because it would **help us prevent scope / feature creep**. Preventing feature creep was imperative to the success to our project as new, unplanned features would take time and resources that we had not factored in. Also, feature creep has a compounding effect and would prevent project closure. Having a clear, predefined scope that everyone agreed on would allow us to stick to our plan and deliver a quality project.

Project Outcome and Objective

Our objective is to create a robotic smart car using Arduino to traverse and map a modular maze using a depth first search primarily, and the “left hand rule” algorithm if time permits. The robot should include a LCD display to provide information to the user, as well as receive user input from buttons. The Arduino will be placed at the start location, then the end location will be indicated by a coloured card, detected by the infrared detectors on the front of the robot. The maze itself will be a 4x4 square maximum (40 walls). The walls are interlocking and modular so they can be manipulated into other configurations.

Project Deliverables

To the user, the Arduino will display debug information and status messages on a back mounted LCD display, as well as possible LED indicators for the buttons. When the robot has completed the maze, the robot will give a couple congratulatory message. After, the user can connect the robot to the computer to print the robot’s path, likely in ASCII.

Internally, the robot will have a matrix that contains a map of the maze and to hold the robot’s path and previous locations. The robot will also get data from the ultrasound to avoid crashing into walls.

Other tools and technology that we used to manage our project: Google Drive, Github, and Google Calendar.

Arduino Car Summative Project Progress Report

Week One | December 5th-9th, 2016

Progress Log - Week One

2016-12-05	General planning and documentation work. Shared all important documents with each other. Named our robot "Beep Boop." Determined how to transport all the parts of the Arduino robot safely to and from school.
2016-12-06	Installed Arduino IDE, explored capabilities of the Arduino car. Determined the wiring configuration for our car and wired the components appropriately.
2016-12-07	Worked on making the Arduino go straight and perform 90 degree turns. Booked a laser cutter and planned materials to purchase for maze construction. Made outline of maze specifications based on the car's size and components.
2016-12-08	Spent the ENTIRE period trying to figure out why the wheels did not spin only to find out that a wire had been unplugged. Tested alternative power sources but discovered that we can only use the provided batteries (which do not have much of a lifespan so we need to make sure they are properly charged in preparation for every work session)
2016-12-09	Configure and calibrated ultrasound and motors. (5.5s = left 180 degree turn. left ~ 3.25s, right ~ 3.2s) Added LCD display to complement the data taken from the sonar Worked on progress logging and documentation.

Status: ON SCHEDULE

So far, we have stuck to our original schedule. We made a lot of progress early on in the week, so when we hit some speedbumps on Thursday, we did not have trouble getting back on track.

We noticed that a lot of groups were held back by a lack of materials, so we are really thankful that we were able to successfully acquire most of the parts we need for our project before we started. All we need are a few wires and some MDF (hardboard) from Home Depot so we can laser cut components for our maze during the winter break.

Even though it has only been a week since we started the project, we have already learned a lot about project management and collaboration. The Gantt chart and scope document we made early in the week has really helped us stay on task during each work period by giving us a concrete idea of what we should be getting done. For example, we decided that it was imperative that we focus on coding and configuring basic robot movements, so each period, we got to work immediately because we knew what we needed to get done. Another thing we have learned is the importance of assigning roles to each person that you are working with. We found that it was inefficient for two people to work on very similar code at the same time, because a lot of the work ends up overlapping. Instead, we made sure every person had a specific role and responsibility. Like on Wednesday, Miasya brainstormed how to build the maze, while Charles coded turning functions for the Arduino car.

Moving forward, we will continue logging our progress as we start coding more advanced functions, testing the limits of our robot, and drafting more documentation.

Arduino Car Summative Project Progress Report

Week Two | December 12th- 16th, 2016

Progress Log - Week Two

2016-12-12	FIRST PROBLEM: Turning based of motor timings was too inaccurate and would cause the robot to crash into the walls of the maze. Our solution was to use the bottom-mounted IR sensors to follow a black line grid. We also decided to use the front mounted IR sensors to check for end position reached. Had to explore how to use the IR sensors and test turning at an intersection. After school, we purchased two large pieces of white cardboard to make our grid.
2016-12-13	SECOND PROBLEM: Line following was not working properly, and the robot would have delayed reactions as it tried to turn or correct itself. We also do not have enough ports for all sensors AND button input, so we may have to consider using the AX and RX ports. These ports are dedicated to serial communication with the computer, which may cause some problems. We worked on figuring out how to fix line following. We made the maze grid by taping black electrical tape onto the display boards we purchased yesterday.
2016-12-14	Figured out that the line detection was not working due to a delay used to turn the servo, which was causing lag. fixed this and worked on the turning at intersections algorithm. Explored a design idea for the maze walls and constructed a small prototype. Determined that we needed to cut teeth into the boards for them to interlock.
2016-12-15	Started designing maze algorithms. Began to experiment with Inkscape, an open source vector graphics editor, to familiarize ourselves with the software.
2016-12-16	Started basic design of scaled vector graphic for the maze walls using Inkscape, whilst considering the specifications of the Epilog 24 Helix laser cutter to ensure compatibility. Made more progress with the maze algorithm, specifically with designing internal storage for the maze and a basic algorithm

Status: One day behind schedule

We hit some speedbumps early on in the week so we were forced to adapt to the situation and modify the steps of our plan. However, we were able to readjust quickly and we are working on catching up.

Early on in the week, we realised it would be extremely challenging to make the robot turn exactly 90^0 consistently because the robot drives differently on different surfaces. We brainstormed ideas, and eventually determined that the best idea was to have the robot follow a grid using the infrared line detector sensors built into the robot. This would not only solve our problem, but also give us an opportunity to explore the uses and capabilities of infrared detection with Arduino. Using the line following method allows our project to be less “trial and error” and more algorithmic, modular, and error proof. With the old method of turning and driving based on timing, we would encounter problems if we tried to change the dimensions of the maze walls, if our wheels were slightly crooked, or if there ground surface were to change.

A major problem we encountered involved transporting the robot to and from class. Since we always have to bring the robot to different places, it tends to get bumped around, unplugging the wires. Whenever this happened, we would have to spend a long time trying to figure out why the display or servo was not working. If we can find a better way to secure the wires, it may increase our productivity.

To catch up with the original schedule, we had to keep our noses to the grindstone and problem solve. On some days, we had to redirect our efforts when the batteries died. However, overall we worked efficiently

and made good progress in the week. In the coming weeks, we will be finishing up our maze wall designs, completing our maze algorithm, and tackling issues as they come up.

Arduino Car Summative Project Progress Report

Week Three | December 19th- 23rd, 2016

Progress Log - Week Three

2016-12-19	Charles chilled in Mexico while Miasya worked hard. First, she began debugging the maze algorithm, fixed a lot of logic errors. Noticed that the current algorithm did not cover every possible scenario so she wrote pseudocode to develop her conception of the algorithm. Ensured no scenario had been overlooked.
2016-12-20	Miasya got an electric shock that really hurt because she touched an unplugged wire while the robot was on. Learned not to do that. She was pretty scared off from messing with the hardware more for the rest of the period, so she continued analysing the requirements for the maze algorithm. Considered stripping the code of different elements, like a struct and several functions, to save memory.
2016-12-21	While enjoying some great music, Miasya scrapped half the maze algorithm and wrote her pseudocode into actual code. Also added a lot of enums to make the code more readable. Improved modularization by isolating the findPath and move_forward functions. Gareth assisted Miasya by looking over her code and making sure everything flowed logically.
2016-12-22	Made the maze algorithm recursive and branching instead of linear. Tested the algorithm, but the folding creases in the board confused the robot's IR sensors, and the timing of the delays needed to be calibrated. Brought all materials home for the break.
2016-12-23	Not in class - Was setting up for winter spectacular!

Status: Slightly behind schedule (*__Note__ that we did <u>not</u> know that Charles was going to Mexico when we made our original plan)

Before Charles left for Mexico, we sat down after school together and discussed expectations and goals for the following week. We prioritized the maze algorithm over everything else. What good was a well built physical maze if our robot could not solve it?

The first alone period, I began at the very top of our code and followed the logic all the way through. Previously, I trusted Charles to code the variable and pin declarations, the setup procedures, and a few functions like turning. I was more involved in the logic behind all of the code and debugging sections. Now that I had to finish the code, I wanted to make sure that I understood everything and there were no errors before I began. Unfortunately, there were some minor logic errors that needed fixing, and the maze algorithm was a little less developed than we had thought, as I discovered some possible scenarios that we had overlooked. So, instead of beginning to code that period, I spent my time debugging our code then writing pseudocode for the incomplete sections of the maze algorithm.

The second period, I set off to test an isolated function that I had debugged, but a couple of wires had gotten unplugged during transport. I accidentally touched the metal bit of a wire while the power was on, which was a very silly mistake. It delivered a scary shock and I was pretty afraid to touch the robot again for a bit, so I went back to coding and left the testing for the next day. The shock gave me a pretty valuable lesson about hardware.

Knowing that we did not have enough ports for both the servo and the front IR sensors, I restructured the algorithm in a way where we could eliminate the need for a servo. Of course, the servo would make the solving more efficient, but we need front IR sensors to indicate the end of the maze, so the IR sensors

were much more important. Conceptually, the recursive function was hard to trace, so once I went through everything, I called over Gareth to double check everything the logic for me. While he was assisting me, he suggested we implement some enums to make our code more readable, and isolating a certain function for improved modularization. His ideas were great, and I made changes based on his input. Gareth's feedback helped me improve our program, which led me to realize how important it is to have a community of supportive peers to collaborate and brainstorm with.

Last working period alone! Back to the testing phase of the software development life cycle. Many functions worked successfully the first time, but some needed some fine tuning (timing delays and calibrating run speed), and some were incomplete. I flagged the appropriate areas to look over with with Charles over the break. I brought our materials home and scheduled a time to meet with Charles.

Overall, I spent a lot of time debugging and rebuilding before talking things over with Gareth and testing. My work cycle resembled closely the software development life cycle, not intentionally at first, but the process proved to be a helpful way to complete subtasks. I also learned a few things about hardware and the importance of having a supportive community around.

Right now, we are nearing the end of the project, and after a few more hours of work with Charles, we should have it ready for the showcase!

Complete User's Guide

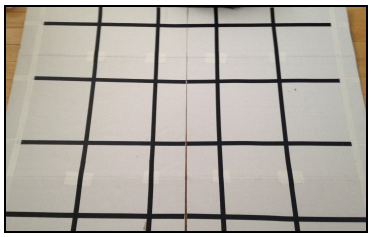
What is Beep Boop?

Beep boop is a fully autonomous maze solving robot, the newest in our line of self-driving Arduino robot cars. Beep Boop uses the latest advancements in sensor technology with two bottom mounted infrared sensors, two front mounted infrared sensors, and one front mounted ultrasound sensor. Your packages also comes with two cardboard SmartBoards™, forty interlocking laser cut SmartWalls™, twenty wooden SmartStands™, and one fully modular and customizable SmartCar, Beep Boop™.

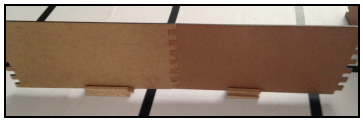
Beep Boop is based on the Arduino platform, and uses the Arduino Uno R3. It is able to communicate directly with your computer to display its path through the maze, and also comes equipped with a back mounted LCD display for an improved user experience

Setting up your maze

Place your boards -
First, place the two SmartBoards™ next to each other so that the black lines on the longer edge of one board line up with the lines on the longer edge of the second board.

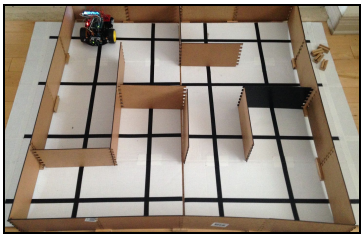


Set up your maze walls -
Next, create walls by connecting the SmartWalls™ as shown to the right. You can use the SmartStands™ for extra stability.



Watch our video! https://youtu.be/Wte0R6b4lrs

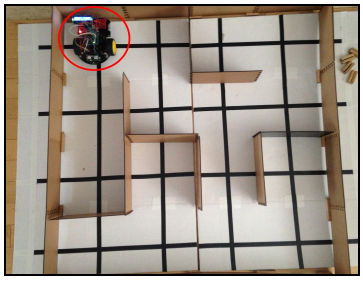
Build your maze -
Create by placing walls in between the black lines as shown.
Indicate the finish point using the black board.



Note: The maximum size of the maze is 4x4. If you create a 4x4 maze, you **DO NOT** need to put walls around the entire maze, but if you make a maze of any size smaller, you do.

Starting your Robot

Position your robot -
Place Beep Boop directly on top of the TOP LEFT intersection facing SOUTH.



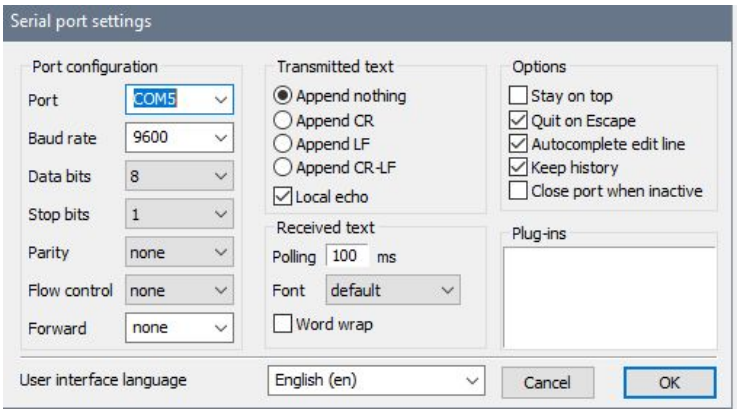
Start maze solving -
Turn him on by pressing the POWER BUTTON under the LCD.
Press the START BUTTON to begin solving the maze.

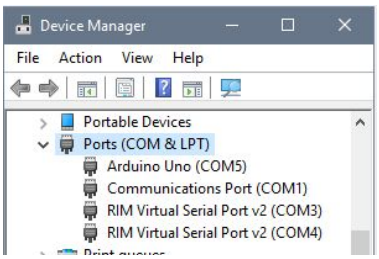
Printing your path to the computer

Once the robot has completed the pathfinding, plug a standard USB type B cable into the Arduino chip, and connect it to your computer. Then, open any Serial Monitor (the included serial monitor is called Termite) and press the start button on the back of the Arduino. You can then press the start button again to restart the program.

How to use Termite

When first opening termite, open the settings wizard and set the baud rate to 9600. Then, set Transmitted Text to “Append Nothing”, and set the COM port to the one the Arduino is connected to (left image)
To check the com port, open the windows Device Manager and go to Ports(COM & LPT) (right image).





Frequently Asked Questions (FAQ)

- The LCD is not displaying anything when I press the power button.
    - After you turn your robot on, wait two seconds for instructions to appear on the LCD display. If the LCD remains blank, or the robot does not turn on, check your wiring and charge the batteries before trying again.
- The robot says it has reached the end of the maze when it has not.
    - When the front infrared sensors detect that the board in front of it is dark, it displays the message “MAZE COMPLETE.” If you room is dimly lit, your robot’s infrared sensors will not be able to detect the difference between a regular board and the black goal board. Make sure the maze is well lit every time you run your program.
- The LCD says “PRINTING MAZE…” but nothing is happening.
    - After maze completion, you have to make sure you complete the following instructions **in order**: connect your robot to your computer, open your serial monitor (Termite), press the start button. If you press the button too early, or your serial monitor is not detecting the proper port, your maze will not print.
- My robot fell off the maze!
    - Make sure not to interfere with your robot while it is solving the maze. If you move the robot or move the maze, you have to restart the maze. In addition, make sure your maze is completely flat and that there is no gap between the two boards.
- Other problems:
    - Maybe your robot has developed a mind of its own, and it is going rogue, but probably not (or at least we hope so!). Try turning it off and on again. If the problem persists, call our service support number! Call us toll free at 1-800-MY-ROBOT (6976268) weekdays

from 9-5 EST or email the designers directly at mbulg1@ocdsb.ca or czhan10@ocdsb.ca.

Interpreting LCD Messages

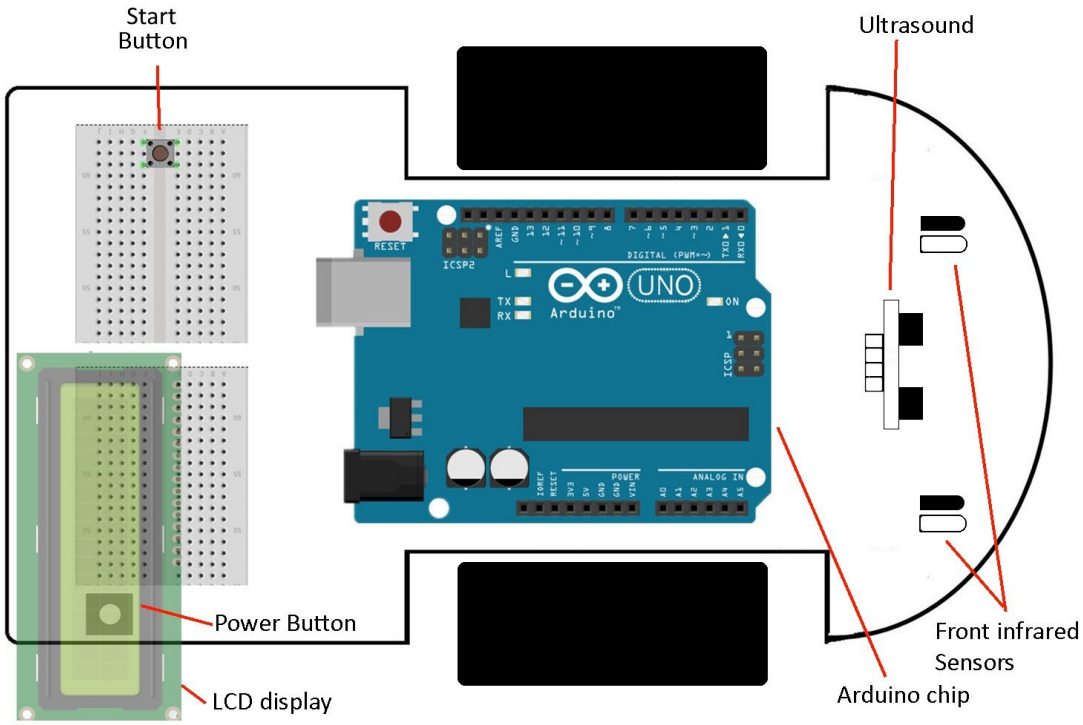
Message	Meaning
PRESS START TO BEGIN	Your robot has successfully powered up and it is ready to run. Press the start button once you have gotten your maze set up and you are ready to begin!
MOVING FORWARDS	Your robot has detected that it can move forwards unobstructed. It is moving forwards.
BACKTRACKING...	Your robot has reached a dead end and is now backtracking to the last vertex. It marks the dead end as CLOSED so it does not return there.
DISTANCE: __ CM	While maze solving, if there is a board directly in front of your robot within range (3 - 400cm), your robot will display the distance to that board if no other message is being displayed.
MAZE COMPLETE PLUG IN & PRESS	Your robot has detected that the board in front of it is black. You have finished the maze, and you can now connect your robot to your computer, open a serial monitor (Termite), and press the start button to display a map of your robot's path.
PRINTING MAZE...	Your robot is trying to print your maze to your serial monitor.
PRESS BUTTON TO RESTART	After printing the maze, you can place your robot on the top left corner intersection of a new maze, and press the start button to begin solving again.
ERROR: CAN'T BACKTRACK	If you receive this error, your robot has terminated the maze solving process. This error can appear for several different reasons. First, your maze is unsolvable. Second, your robot was interrupted by human interference. Third, your maze is set up improperly. Lastly, the hardware has malfunctioned and you must restart.

Major Hardware Components and Their Functions

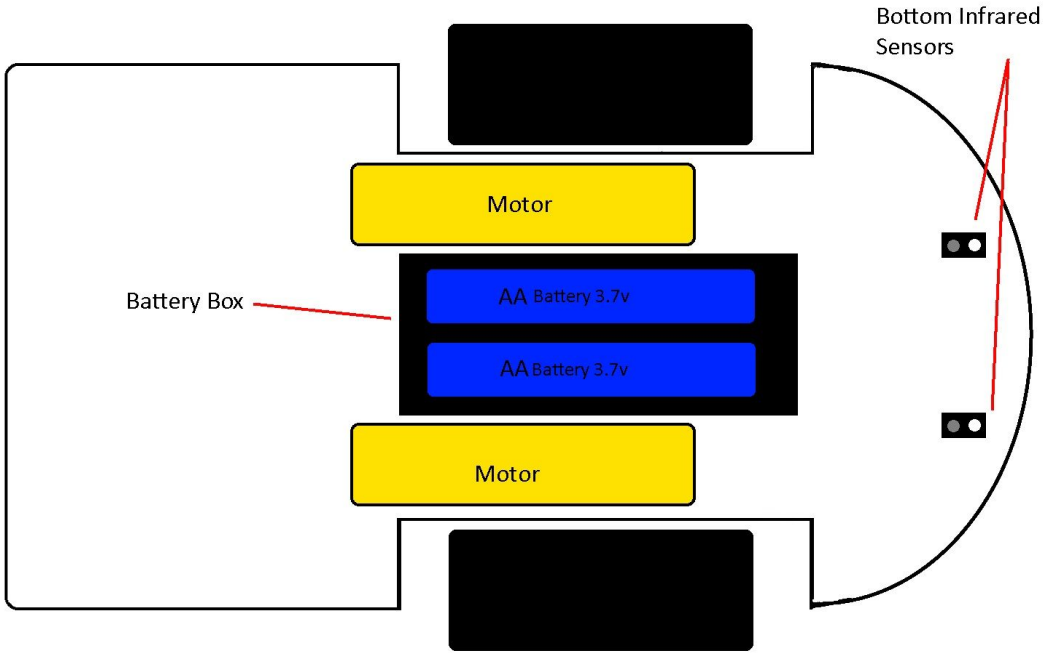
Name	Location	Function
Power Button	Top; Under LCD display	Turns the robot on and off
Start Button	Top; back right breadboard	When pressed, begins maze solving
Arduino Chip	Top; very centre	Sends and receives signals from different hardware components; follows coded instructions
LCD Display	Top; top left breadboard	Displays helpful messages for the user
Ultrasound Sensors	Top; front and centre	Calculates distance to objects in direct line of sight by sending out waves and timing how long it takes for the echo to return
Infrared sensors (top)	Top; front, one on each side	Detects if the board in front of it is dark or light (dark board indicates the goal)
Infrared sensors (bottom)	Bottom; front and centre	For line following, detects if board underneath it is dark or light
Motors	Bottom; middle on each side	Control the wheels, for movement

Battery Box	Bottom; very centre	Power source for the robot

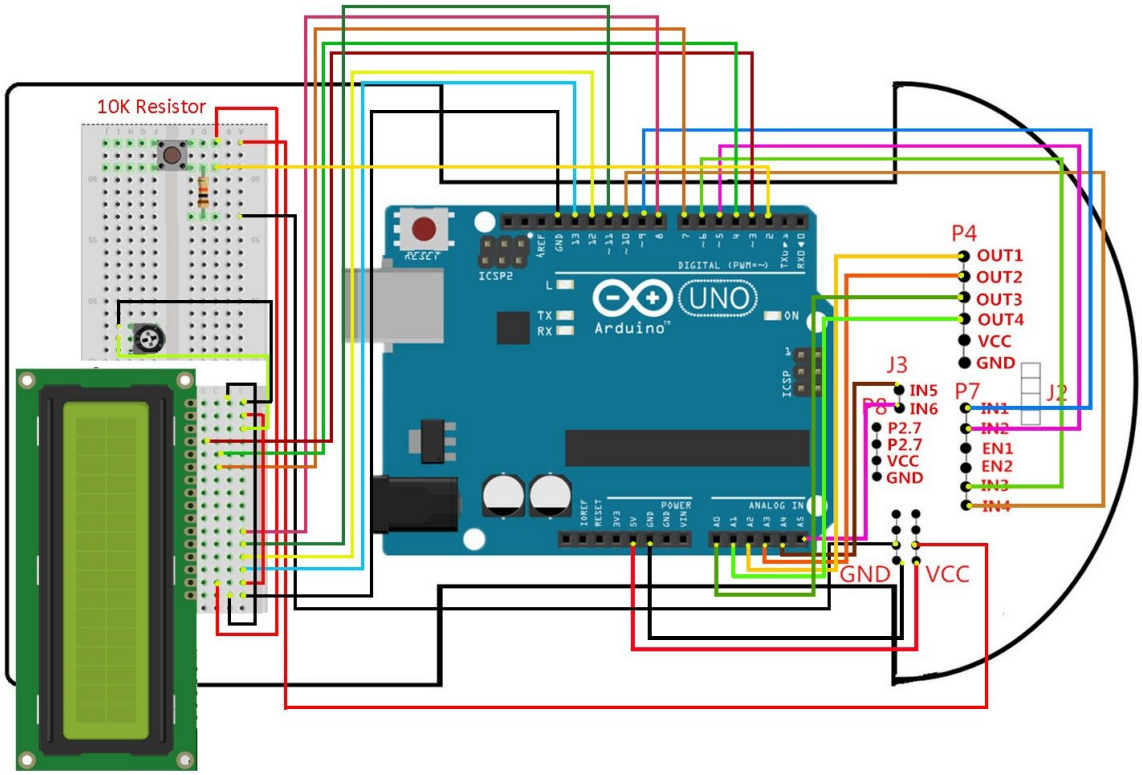
Robot Diagram - Top View



Robot Diagram - Bottom View



Wiring Diagram



Reflection

Overview

Our experience these past few weeks has been fantastic. This project has not only encouraged us to try new computer technologies, but it has supplied us with skills, experience, and confidence that will benefit us in future projects in school, at home, or at work.

To begin, we developed valuable technical skills by experimenting with real life technologies such as Arduino, Inkscape, Termite, and Epilog Helix laser cutting. In addition, we employed the programming concepts we learned in class to code our robot, as Arduino code is based on C++. We also had a chance to experience the project process and apply the software development life cycle to our very own project. Most importantly, our biggest struggles taught us that proper time management is paramount to a successful project.

Project Management and The Project Process

When planning our project, we kept in mind that we only had a few weeks to work on it, so we outlined the basic functionality that would be simple to code and build at first, but left space so we could add features if we had enough time. We also predicted that we would run into some problems and delays, so we broke each of our tasks into smaller chunks so that our plan was more agile and easier to manage. **As part of the project process, we wrote a brief development plan (project proposal) and scope document** to outline tasks, deliverables and our schedule and to help prevent excess scope / feature creep.

Each work period, we worked on a task depending on our stage of the software development life cycle. For almost every task and subtask, we would first analyse requirements as per the expectations of the project. Second, we would design the product (in isolation if possible). Third, we would implement the product if we had not already. Forth, we would test it under possible test cases. Finally, we would evaluate if the task needed more work or if it was complete (and finish any documentation as needed). We would repeat the steps outlined above if necessary.

While we were building our project, we frequently referenced the Gantt chart we created on early on in our project. Using the Gantt chart helped us identify core functionality and establish a timeline for the project. Not only that, but it also helped us absorb the information visually and easily spot periods of overlap and see how delays in one task would affect others. Another advantage of the Gantt chart was that it helped us focus on the most important aspects of our project so we would work to make sure our software and hardware was able to fulfill our needs before adding any unnecessary bonus features.

When we encountered small issues, we preserved the important aspects of our plan and tweaked our functions or design to adapt to the change. For example, we had to modify the robot movement from a timing based system to a line following based system due to hardware issues resulting in motor inaccuracy and timing inconsistency. We also had to adapt to unforeseen hardware limitations such as the unavailability of ports, which resulting in us having to remove the speaker from the robot. In terms of added features, we stayed modest and focused on improving the quality of our existing features instead of developing additional ones, with the exception of a visualization tool that we integrated to improve user experience. This additional feature allows the user to simply press a button to restart the program after the user has printed the robot's maze path to their computer screen.

In the end, we built an effective, quality robot with hardware and software that complemented each other. We were able to deliver the final project on time despite hitting the occasional roadblock during development and being one man down for a few periods.

To complete the project process, we first ensured completion by reviewing the requirements. Next, we self-evaluated our project by testing it and having a conclusion meeting. **Once the project was closed, we wrote a reflection in which we reviewed our accomplishments, mistakes, and the lessons learned from them.** Soon, we hope to celebrate the completion of the project as soon as the opportunity arises.

What We Did Well & Changes to Make in The Future

Throughout the project, we worked not only efficiently, but well together. We trusted each other to complete their part of the task and we helped out each other when we needed it. Our communication was great and we put time and effort into listening to each other's input when brainstorming and making changes. It was really great to have such a supportive partner through the project process because it minimized our stress levels and helped us tackle conflicts.

Another aspect of the project we did well was planning (for the most part). More specifically, we were very realistic with our expectations and we did a good job of assessing the length of time we would have to dedicate to each task. Up until Charles's absence in the last week, we were on track with our original plan, despite hitting some major roadblocks that required us to restructure some aspects of the initial design. If not for the absence, we might have completed our project before the break started, which would have given us time to develop more features or help our classmates more.

Looking back, our biggest issue involved time management. In our original plan, we were not aware that Charles's family was planning a trip to Mexico. So, when Charles was forced to go to Mexico for a week he was not able to make a significant contribution to the project, all he could do was help online. It was a little upsetting that our project ended up biting into our winter break, but now we have learned that sometimes things pop up unexpectedly. Miasya was able to adapt very well to the sudden loss of a team member, and worked efficiently with the time that she had. Upon Charles's return, he made up for his absence by making the robot turn based on line following, implementing backtracking, and adding computer display.

Now we truly realize that good project plans should have some flexibility where we have room to drop features if we run our of time, not only features that we can add if we have extra time. If we could go back

and make improvements to our plan, we would have made it more flexible so that we would have been more prepared when this problem came up.