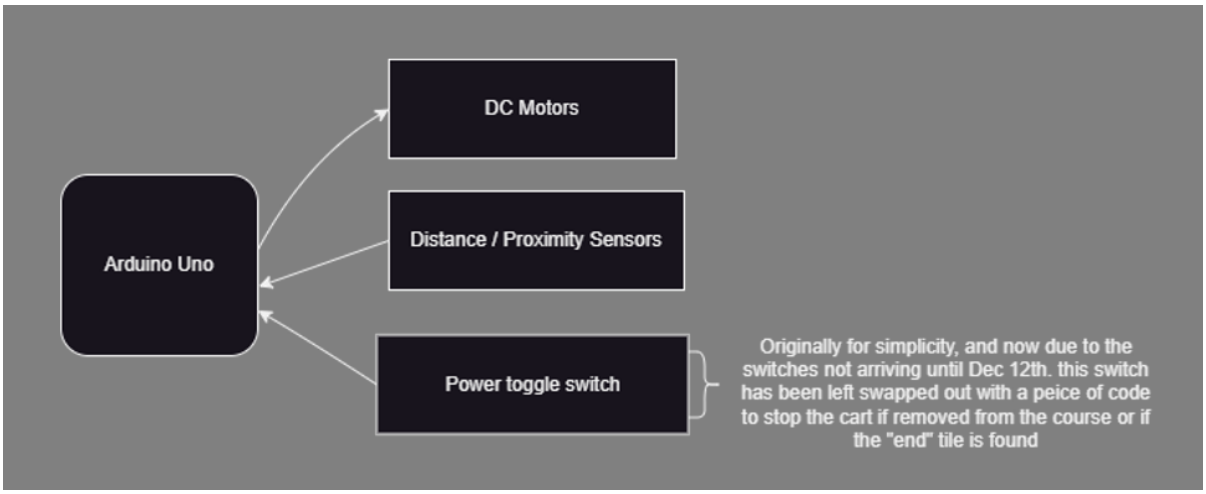
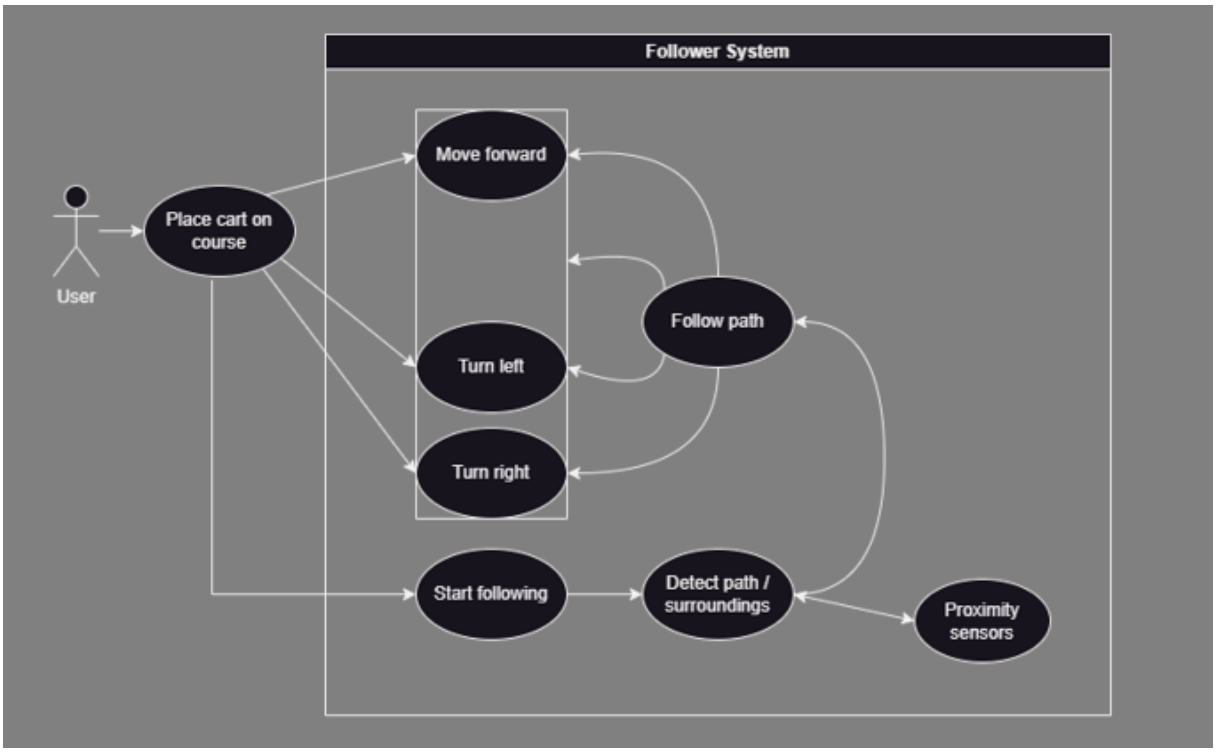
I. Prototype documentation and links

Architectural Block Diagram Final Version:



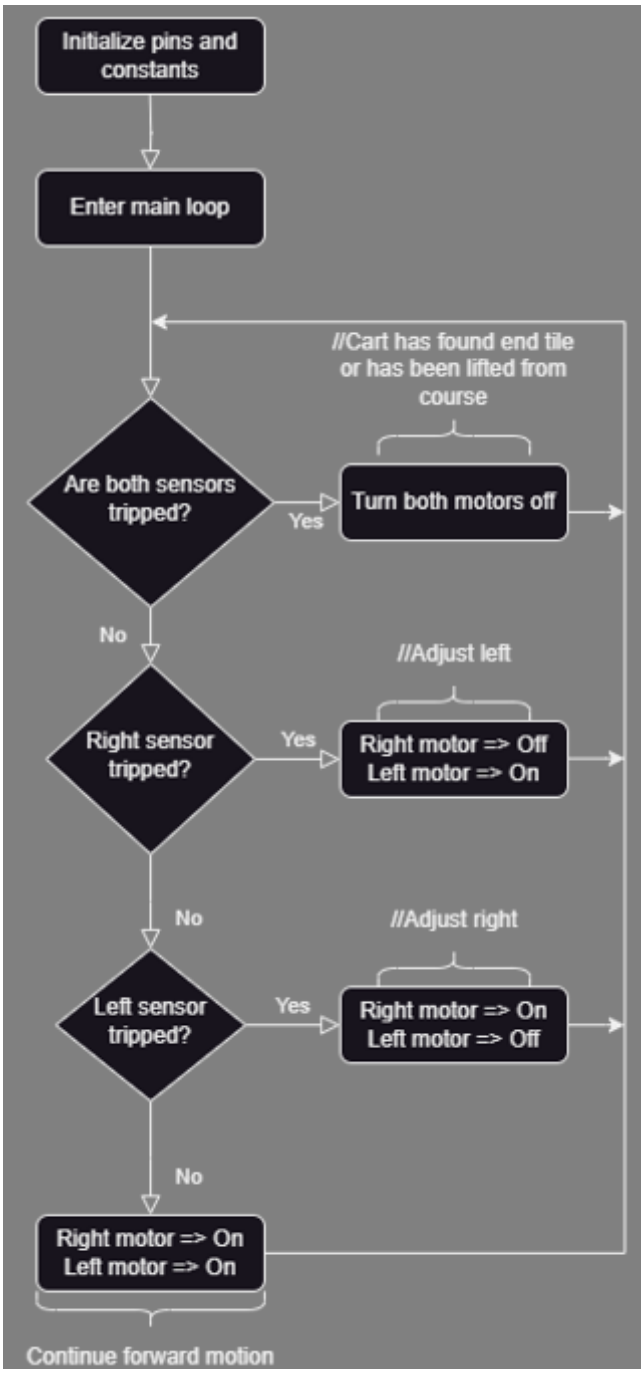
Future features that might be added: wireless control module (e.g., Bluetooth module), color sensor

UML Case Diagram Final Version:



In the future, we might change it to use a controller, such as a gamepad, to control the rover's movement, enabling it to perform actions like moving backward and pausing.

Control Flow Chart Final Version:



In the future, we might add branch logic to check Bluetooth inputs, such as verifying whether a "pause" or "move backward" command has been received. Additionally, branch logic could be included to determine whether the color sensor detects a colored line.

CRCards Final Version:

Class: Proximity Sensor

Responsiblilities:

- Calibrate sensors
- Continuously read proximity data from the environment

Collaborators:

- Main Controller
  - Sends Sensor data to main controller for processing

Class: Drive Controller

Responsibilities:

- Move / Control drive motors based on input signal from main controller
-

Collaborators:

- Main Controller
  - Receives movement input from main controller

Class: Arduino Uno / Main Controller

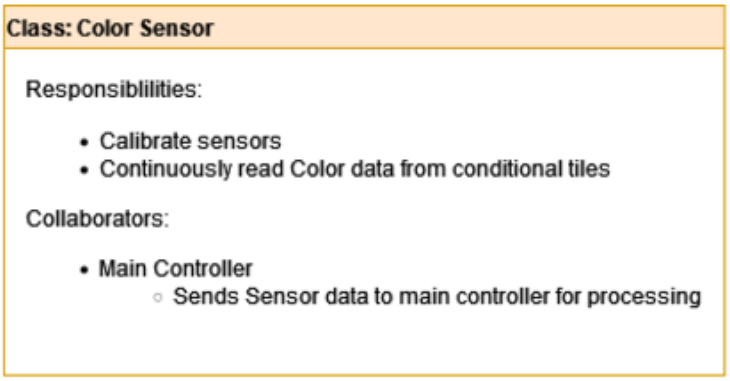
Responsiblilities:

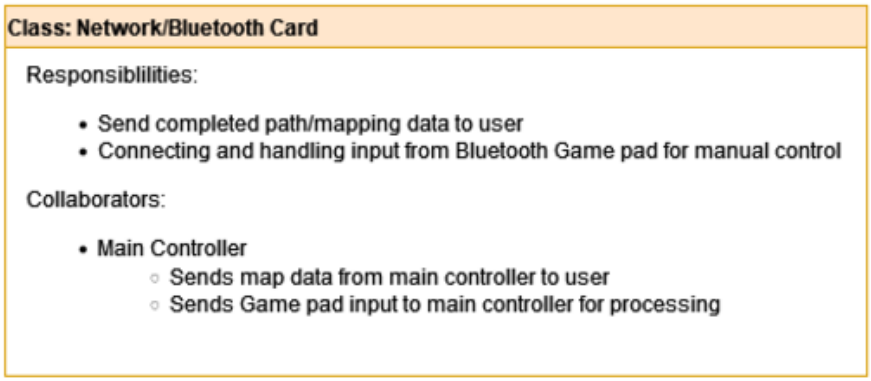
- Process Game pad input from Network Card
- Process data from Proximity Sensors
- Log path / environment data
- Make path-finding decisions based on sensor data and per-determined algorithm

Collaborators:

- Drive Controller
  - Sends movement input to Drive Controller
- Network / Bluetooth Card
  - Sends Logged environment data to network card for transmission to user
- Proximity Sensor
  - Receives sensor data from processing

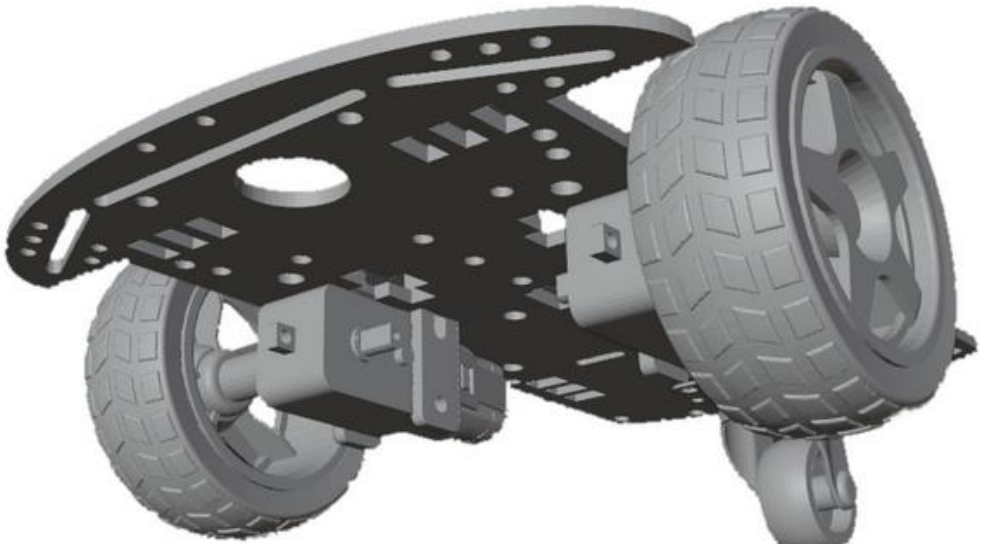
Classes that might be added in the future:

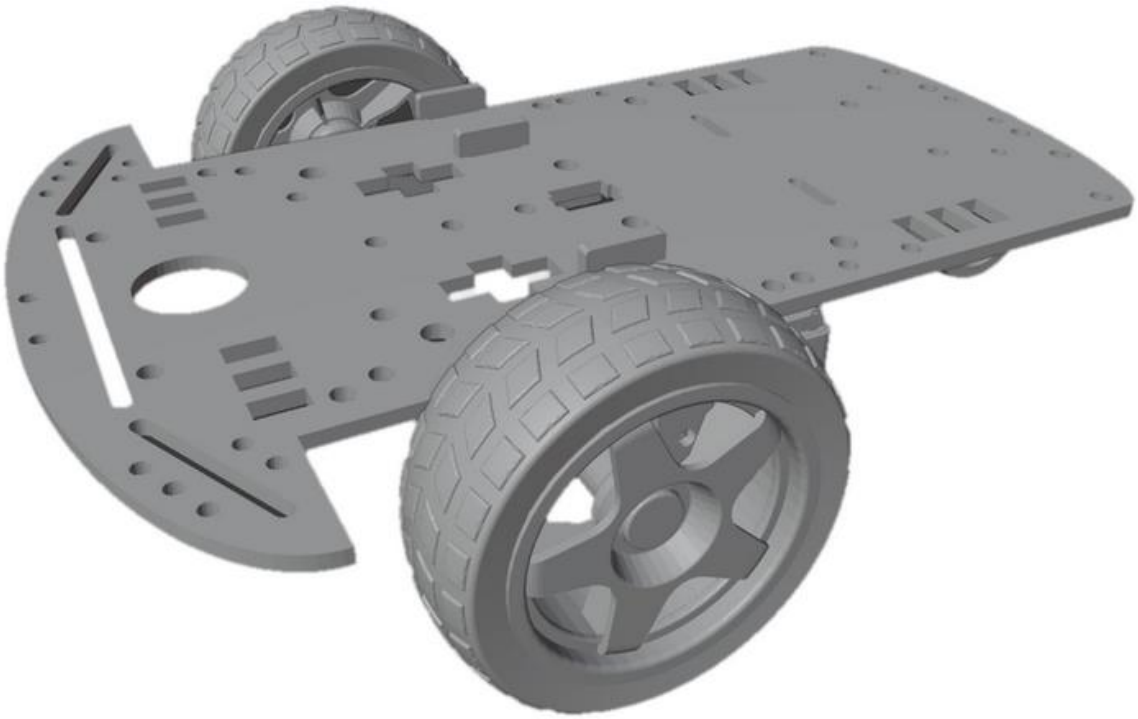




Our main code enables the rover to stay on track by detecting inputs from the left and right IR sensors. Our sensors read the path information, and when they detect that the rover is off track, the code adjusts the motors to correct the direction.

Chassis references:





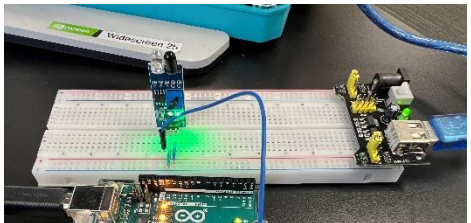
The final chassis design will be like the reference photos above, I have yet to get the modified version put together, and therefore our prototype is a breadboard ridden hardware implementation for now

Hardware components

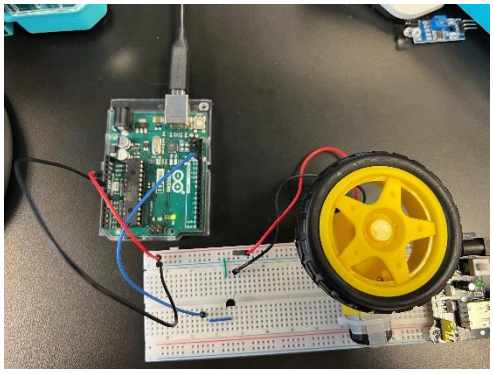
Component	Model #	Quantity	Price	Inclusion status	Order status
6V DC geared Motors	AE19522	4	9.99	YES	On Hand
Prototyping kit: Breadboard + Basic Circuit Elements	N/A	1	15.77	YES	On Hand
IR Obstacle Avoidance + Detection sensor	TS-US-070-CA	5	6.99	YES	On Hand
Arduino uno R3	ARD_A000066	1	27.6	YES	On Hand
NPN BJT for motor control	2N3904	2	N/A	YES	On Hand
1.5-6V DC Motors	B07SQXRSNR	6	9.68	insufficient	On Hand
Bluetooth base board for arduino uno rev3	ML-HM-10	1	10.99	Not likely	
Chassis to be modified and printed	N/A	1	N/A	YES	Not yet printed
Smaller breadboard for rover chassis	N/A	1	5.99	YES	Not yet ordered
5 pin Photoelectric color sensor	C41	1	9.99	Most likely	Not yet ordered
360 degree swivel wheel and bearing					Not yet ordered

Circuit configurations

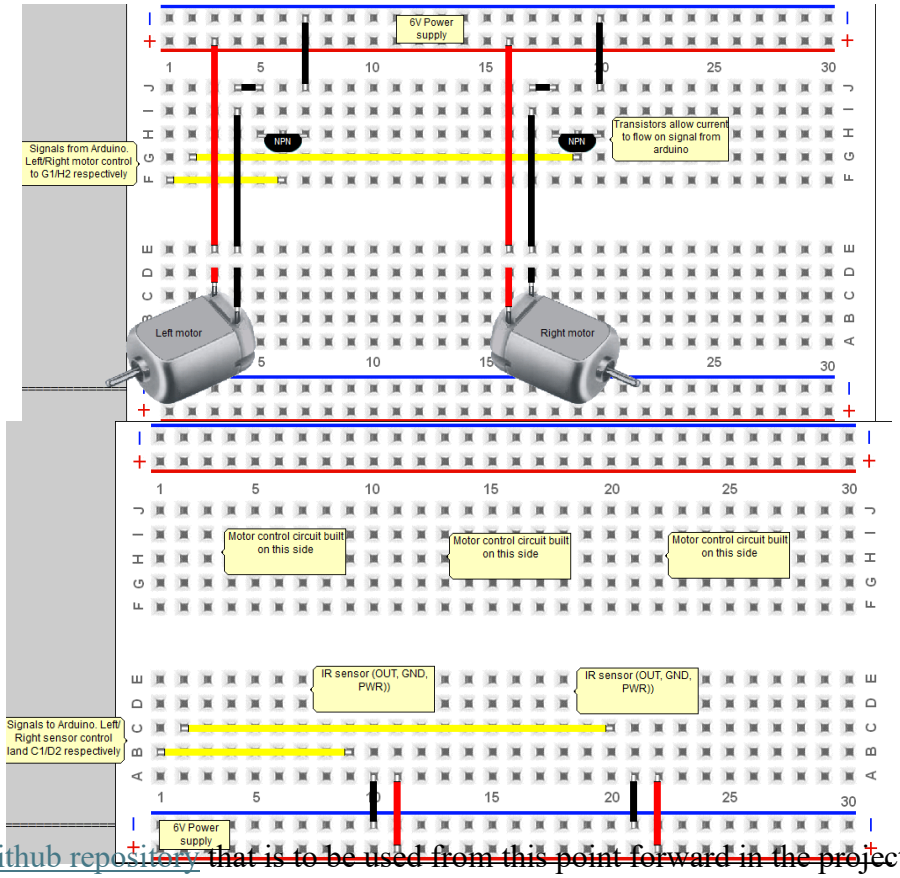
IR-sensor testing:



Motor testing:



Full circuit w/ control board:



Source code:

Link to the Github repository that is to be used from this point forward in the project progression

Calibrating the IR sensors:

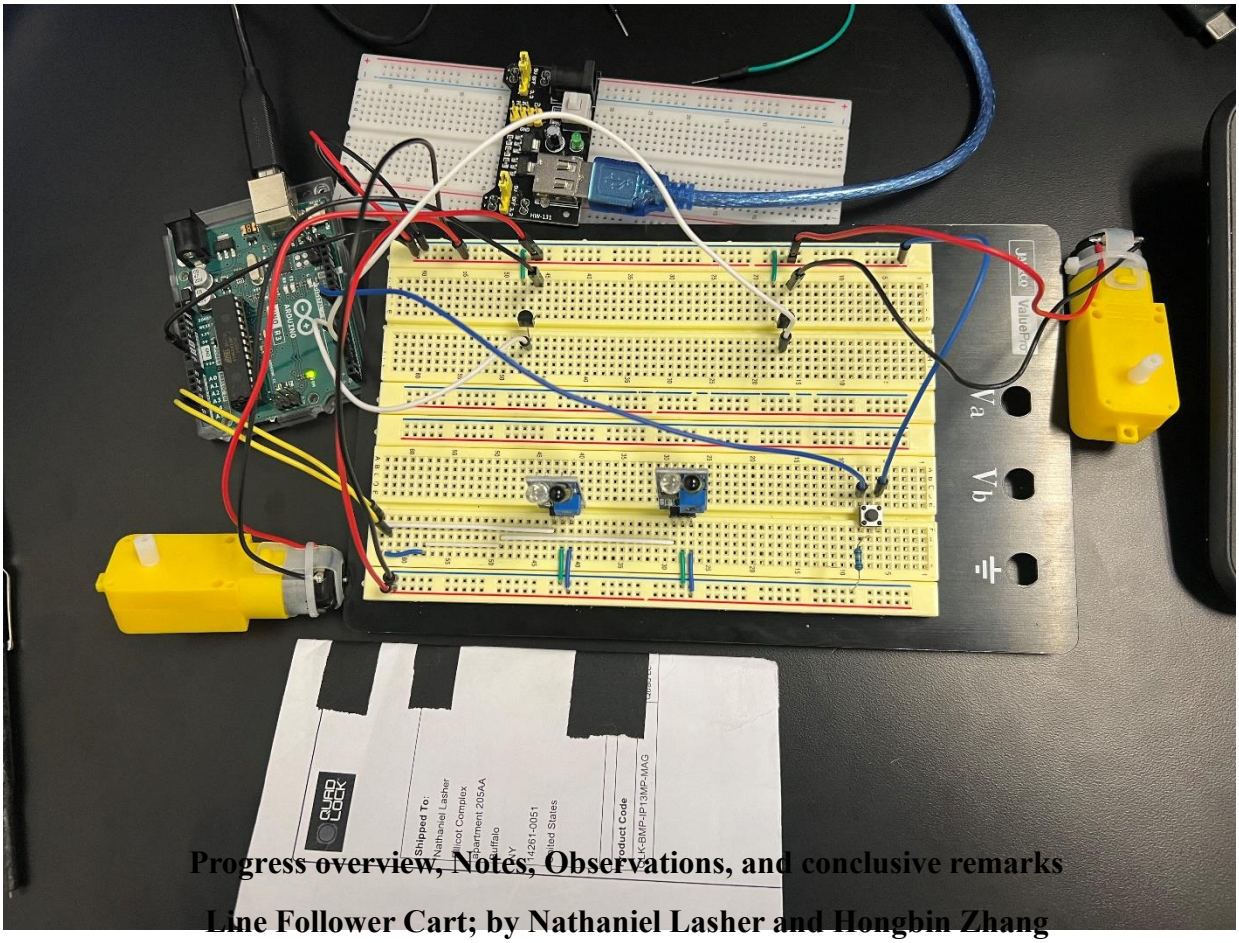
The IR sensors were calibrated individually using the “IR sensor test” code and the circuit configuration labeled as IR-sensor testing on the last page during early development. The serial monitoring built into the Arduino will be used for this on the final cart design for ease of calibration/troubleshooting. The potentiometer on each IR sensor is adjusted to ensure the output stream reads >1000 when idle and < 35 when triggered. This ensures the IR sensors output a value that is within the range expected by the software. Each IR sensor has an onboard led to indicate when the device is triggered, and a set of redundant LEDs are to be added to the cart to indicate the control board has received and acknowledged the input correctly.

Testing the motors:

The motors were tested using the “Motor test” code and the circuit configuration labeled

“Motor testing” on the last page during early development. The final design of the software includes a simple proximity trigger for both motors in which both motors spin up when the cart is held up to a test paper. This adds a simple method of checking that each motor not only functions but that the control board is sending control signals accordingly.

Base hardware circuit:



Progress overview, Notes, Observations, and conclusive remarks

Line Follower Cart; by Nathaniel Lasher and Hongbin Zhang

I. Conceptual changes

My original idea was an autonomous rover that can map out an environment such as a maze or a fabricated obstacle course and be able to calculate and take the safest or fastest route through it with consideration of environmental conditions, some represented as colored tiles. However, to adhere to a tighter schedule and the fact that this semester has left little time for extensive research I decided to scale back to a much simpler line following rover and basic representations of the main functionality to avoid memory intense calculations like path finding. Included in this document are the updated versions of the early design diagrams to reflect the direction we plan to take the project from here

II. Where were at

The current state of the project is a semi-autonomous rover (currently breadboard ridden) that can follow a line course. This course can be made with any reflective or dark matte tape as well as any dark colored marker. My test paper with different thicknesses and materials is pictured with the full circuit. Once we certify that it can indeed follow a line once the circuit is fitted to a chassis the plan is to program extra behaviors that consider environmental conditions represented by colored lines or walls along the course. Even though at its simplest the "mapping" function would only return a string of data that can then be processed by an external device, that will have to be saved for a future iteration. Mostly because we would rather spend time fine thoroughly testing and implementing another layer of environmental interaction with the colored lines and color sensor. Bluetooth communication is a project of its own as I have no experience with any APIs or libraries of which I thought I would have a lot more specific guidance on in the beginning. The LCD readout becomes redundant in the new scope of the project, so it will also be sidelined for a future iteration. The rover will not be able to be piloted manually as of this but will run off the interactions between it and the environment alone. The current working codebase can be found at our GitHub repository, the main file is available in both a pure ".c" in the main folder or as the native ".ino" within the "Main_code_prototype" folder. I still thing the original problem I was trying to solve with the line follower is interesting in that a more advanced version of this system could make use of different hardware and be applied to the exploration of areas inaccessible to emergency personnel with a more complex and robust vehicle to map utilities that are hazardous or unsafe in certain scenarios such as underground gas, water, or power lines.

III. Challenges and solutions

This being the first time using Arduino for my partner and I the progress on our basic functionality has been slow but significant. We now have a working hardware-software interface and plan to move forward towards the final product with everything we've learned about the board and the hardware components we've worked with so far. Many of the slowdowns we had when interfacing the code with physical hardware were caused by the (electrical) current limitations of the Arduino. The Rev3 control board has a maximum current draw of 50 mA which was a huge problem for the geared motors that finally arrived just to require 150mA…

However, we were able to use an external power supply that, in combination with a NPN Bipolar Junction Transistor (BJT) made it possible to use the Arduino to simply offer a control signal that would flip the transistor to allow current from the external supply to power the motors! in the final product this power supply will be supplemented with 1 or 2 AA nickel-metal

battery packs each containing 4 cells. Supplying around 6-10V to power both DC motors and the IR sensors separately from the Arduino. We made this decision to avoid overloading the control board as I saw this affects the capability of the board to function properly. Even if there was no sign of load, in some cases the circuit would do things that were unexpected considering the code that was uploaded to it. Another issue that I resolved within a week was that the documentation for both the original DC motors that we ordered AND the IR sensors was slightly incorrect. The DC motors did not meet the current specifications of the Arduino, and the IR sensors sent the *opposite* values when triggered than what was mentioned in the description on amazon.

IV. Future testing

As for future testing plans for the unit, I've implemented serial monitoring for quick calibration of the IR sensors, as well as a proximity trigger to spin the motors on contact with the test paper. This may later be tied to the color sensor to incorporate it into the test i.e. a blue paper will be excluded from the conditions list and instead become a hardcoded "test" color. Some simple test-course concepts are pictured below. The triangle represents the entrance to the course and the square represents the "End tile" that will trigger the rover/cart to stop. These will be expanded upon to gauge what our current project is capable of and what we need to improve upon prior to the final phase.