**CPEN 291   
Project 1 Report**

**A. Group info**

Lab section: *L2B* Group #: B\_G11 Group’s Lab Bench #s: 11 & 12

Student names:

|  |  |
| --- | --- |
| **Sanjeev Krishnan** | **Parsa Riahi** |
| **Arnold Ying** | **Amir Ali Barkam** |
| **Stella Wang** | **Rain Zhang** |
| **Manek Gujral** |  |

In Sections B and C of the report:

* Explain the design and implementation procedures, and thoroughly provide documentation for the circuitry and software. During the project, you must have selected any method over another for some steps, describe the alternative (e.g. the second best) you considered. Include block diagrams or drawing to identify the main components and their interactions.
* You may include code segments in this part of the report only whenever needed for the explanations of the software design and approach. Your code must include comment statements, so do not repeat what is already included in the comment statements. As usual you will need to submit the complete code file separately, and also to include the complete code as an appendix to this report. The code must be readable in the first place and include sufficient comments (per code segment and per line, when needed) for documentation.

**B. Technical documentation for the main functionality**

* The hardware including the circuit for the reflective optical sensors, and why you chose this configuration (i.e. the number of sensors you used and the way you arranged them relative to each other)

For the reflective optical sensors, we used a 100 Ohm resistor connected to the Digital in pin and a 4100 Ohm resistor connected to the 5V pin. This ratio was chosen to maintain the recommended circuit but to increase the power to the circuit at a safe level that will operate well under our otherwise low power drawing conditions. In total, we used 4 reflective optical sensors for our robot. They are used as sensors to support our PID controller that detects whether the robot is aligned with the track.

The four sensors are configured in two pairs that are situated tape-width apart - approximately 1 cm (See Appendix G-1 for diagram). This way, if the inner sensors detect black while the outer detect white, the robot is following the line. If both sensors in the right pair detect black while both sensors in the left pair detect white, then the robot is straying left, and vice versa for straying right.

Used in conjunction with a PID controller, we are able to generate a stream of error values based on the robot’s alignment with the track to output adjustment values to the motor, so the robot will turn smoothly and without significant oscillation, having used all three terms of the PID.

* The algorithm for the line tracking functionality

The algorithm for the line tracking functionality is a Proportional Derivative Integral (PID) controller that takes into account the current error value, its derivative and its integral over time according to binary input from the four sensors. Each sensor outputs a bit, with 0 signifying white and 1 signifying black. Therefore, we get a 4-bit input from four sensors with the most significant bit from the leftmost sensor and least significant bit from the rightmost sensor.

Using the input, we calculate an error value that is either positive, negative, or zero which signifies an adjustment to the right, left, or none, respectively. For example, if the input is 4b’0110, then our PID outputs a value of 0, namely no adjustment is required and the robot continues to move forward.

Sensors are sampled at a rate of 3000 Hz, namely approximately every 1/3000 seconds. The sample rate was decided upon after trial and error and fine tuning to allow for sufficient time to see a sharp turn and react to it.

* The headless Pi use, implementation, and challenges

The headless Pi is used as the main controller for the PID algorithm and the motor speeds. It is attached to a portable battery pack and the Motor Hat and is situated on the 2WD Mobile Platform robot to allow autonomous function and can be controlled via ssh if needed after powering on.

It is at times difficult to control the RPi headless, as it can only be communicated via terminal. In the beginning, it was difficult to get used to coding through the terminal, but this issue was solved by the team’s desire to always learn more and now each member can use the terminal to start the Pi. We also purchased a touchscreen LCD for our demo so this will allow easier use of the Pi going forward.

* Battery-operated robot implementation and challenges

The main body of our robot is the 2WD Mobile Platform, which consists of 5 AAA batteries and 2 DC motors. The two DC motors connect to pins on the Motor Hat to allow control of voltage supply via the MotorKit (software). The battery source is attached to a power switch on the 2WD Mobile Platform before connecting to the Motor Hat. The motors can independently go forwards and backwards due to the individual H-Bridges existing in the motor hat, removing the need for our team to use relays or another large hardware component that would hinder the robot’s overall speed or else draw more power to move due to the added weight.

The greatest challenge that we ran into with the robot implementation was during fine tuning of the motor and PID control. Many times, we will fine tune our code to working condition and come back the next day to have the robot not following the line properly again. This is due to a number of factors: battery pack drains slowly, ambient lighting and shadows in testing area, different track shape, etc. The general method we used to determine accurate Kd, Kp, and Ki values (in that order) was similar to the Ziegler-Nichols method described here:

1. Set all gains to 0.
2. Increase Kd until the system oscillates.
3. Reduce Kd by a factor of 2-4.
4. Set Kp to about 1% of Kd.
5. Increase Kp until oscillations start.
6. Decrease Kp by a factor of 2-4.
7. Set Ki to about 1% of Kp.
8. Increase Ki until oscillations start.
9. Decrease Ki by a factor of 2-4 and the PID values are approximately set.
10. Do some small fine tuning.

**C. Technical documentation for the additional functionality**

* What the additional functionalities are
  + Mobile app
  + LCD Display
  + Live camera feed
  + twitter bot
  + wall detection
  + Live path graphing on LCD
* Include the list of the additional components you used
* ultrasonic sonar sensor
* 3.5” Touchscreen LCD
* Pi camera module
* How camera is used as a part of an additional feature
* The camera is used to take snapshots of the robot’s environment to be posted to Twitter as an update tweet.

* The hardware implementation
  + Infrared Sensor protoboard
  + Vehicle weight balancing
  + Power control
* The software implementation
  + LCD Path Graphing
    - Since the robot always determines its own direction and speed, we can plot the path it travels, and essentially draw a picture of the tape line that it sees.
    - We assume an initial orientation of moving up in the y direction, and use the last known point to calculate the coordinates of the next point through some trigonometric identities and formulae
    - This image is shown on the LCD as the robot traverses its path
  + Mobile App to control Robot
  + Wall detection
  + Twitter bot

**D. Test and evaluations**

Since each of us started off by writing separate code in separate files for each different component, we did initial testing for each component separately. For example, one person wrote ‘motor.py’ for software controlling the DC motors, while another person wrote ‘camera.py’ for testing and controlling the camera. After ensuring hardware and software for each component was working, we started integrating two components at a time and ensured the integration was functioning before integrating additional components. For example, the first two components that we integrated together was the motor and PID controller.

To test the motors, we tried both stepper motor and motor.throttle from the MotorKit library. We started by connecting one motor to the MotorHat on the Raspberry Pi and ensured that the motor moved according to the code. We chose to use motor.throttle over stepper motor because the output movement on the motor was more consistent. Then, we attached both motors to the Motor Hat and ensured both motors moved simultaneously and according to code.

After ensuring motor control via software, we tested the integration of our PID controller. This part required lots of iteration and fine tuning from trial and error, as various factors affected the output on our robot. For example,

Finally, after ensuring the basic functionality of robot following a line, we moved on to implementing and integrating additional functionalities, starting with the camera and LCD display.

**E. Conclusions and Reflections**

This was a very open-ended project, which allowed more freedom and creativity, but also required more responsibility and communication. In earlier stages, it was difficult to figure out equal tasks for each team member, especially because we were not fully set on all of our ideas yet. The process was an iterative one and there were no formulas or guidelines for us to follow closely, and this lead to lots of communication - for large or small steps - during the design process, to ensure that everybody is on the same page and being productive with their time.

In the end, each member on our team rotated taking leadership depending on the component that they focused mainly on and where that fit into the design process. For example, teammates in charge of the motor and PID took charge in the earlier stages of the process, while the teammates in charge of the LCD display displayed more leadership towards the later stages in the process.

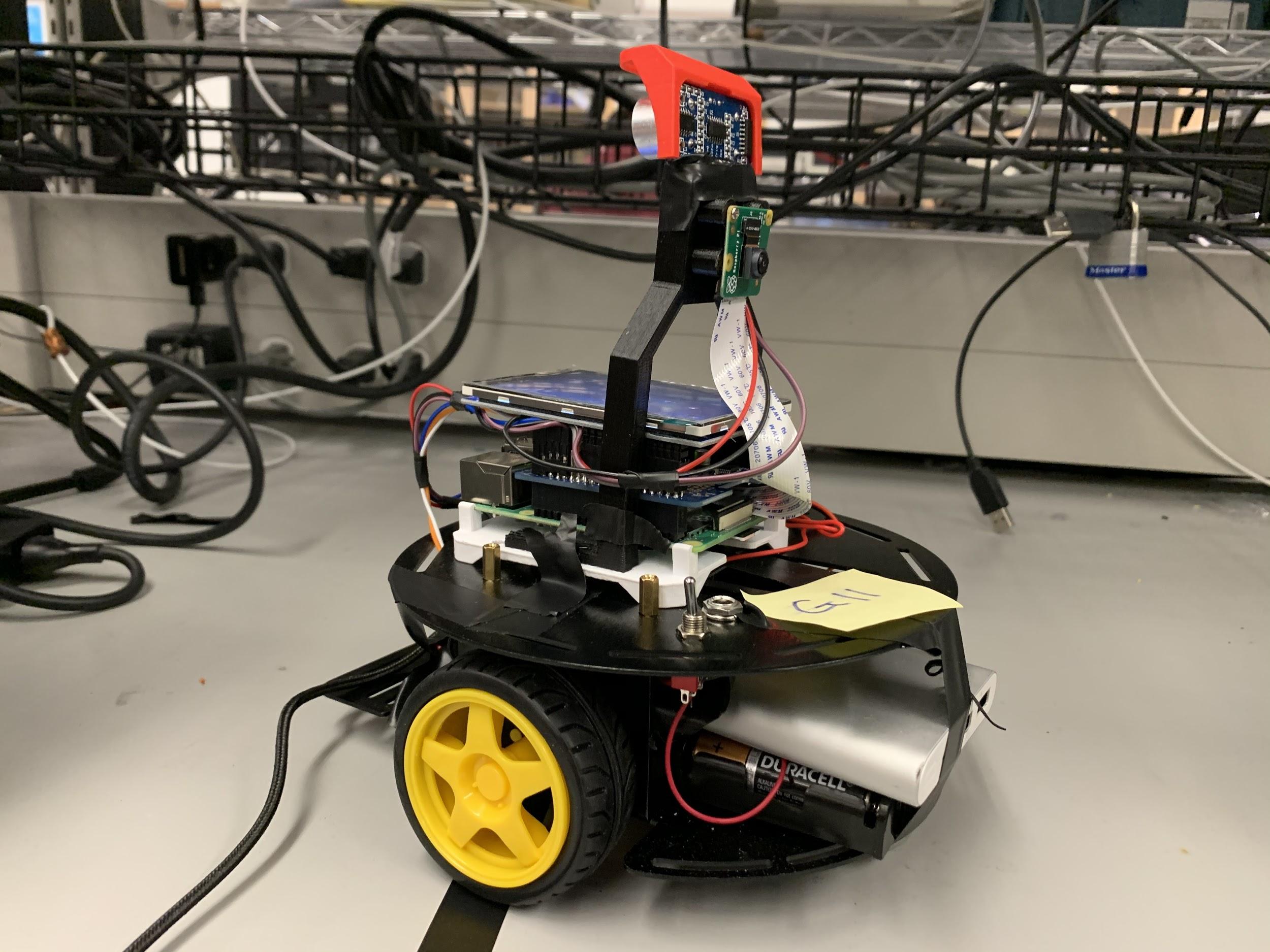
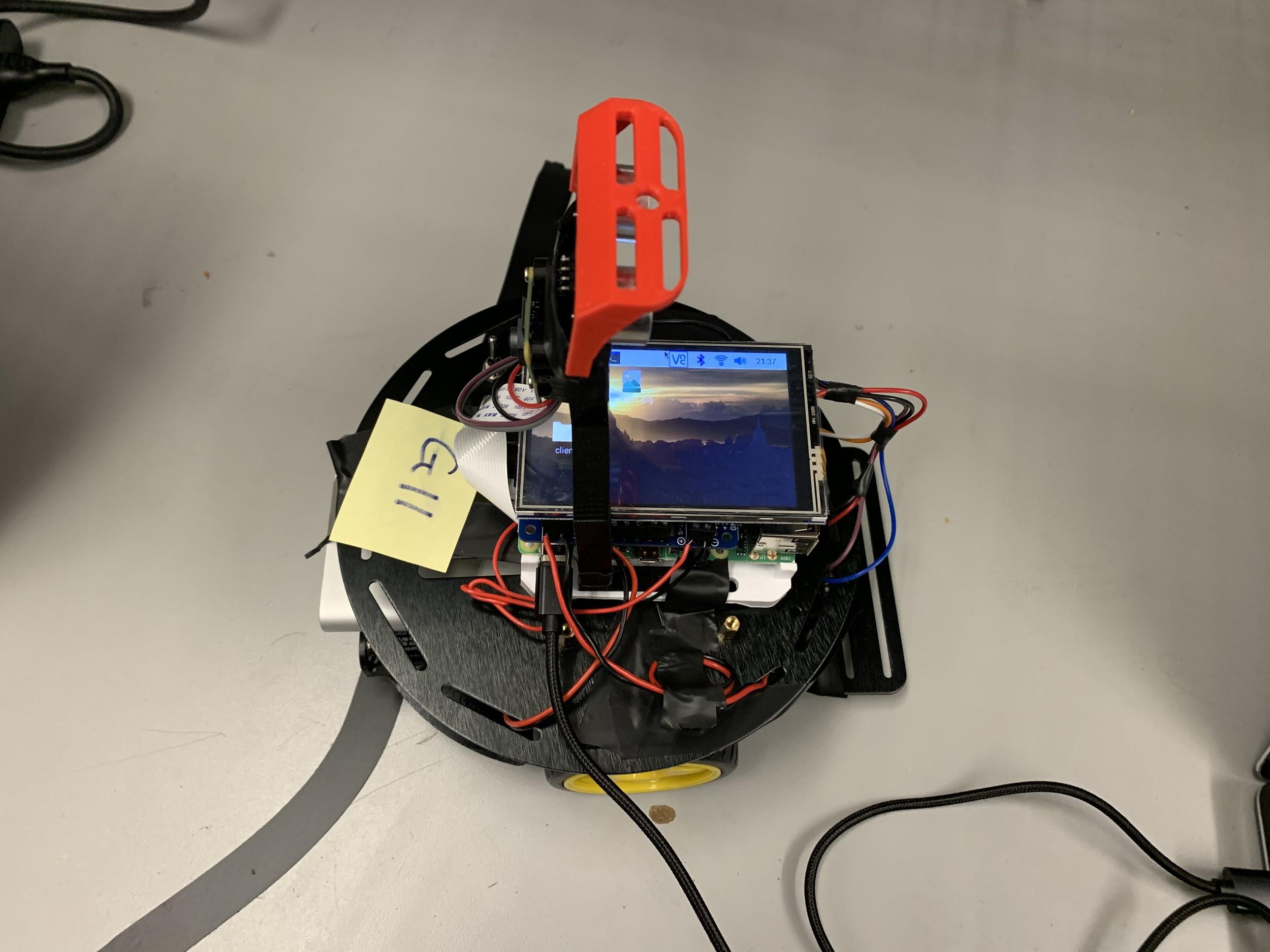
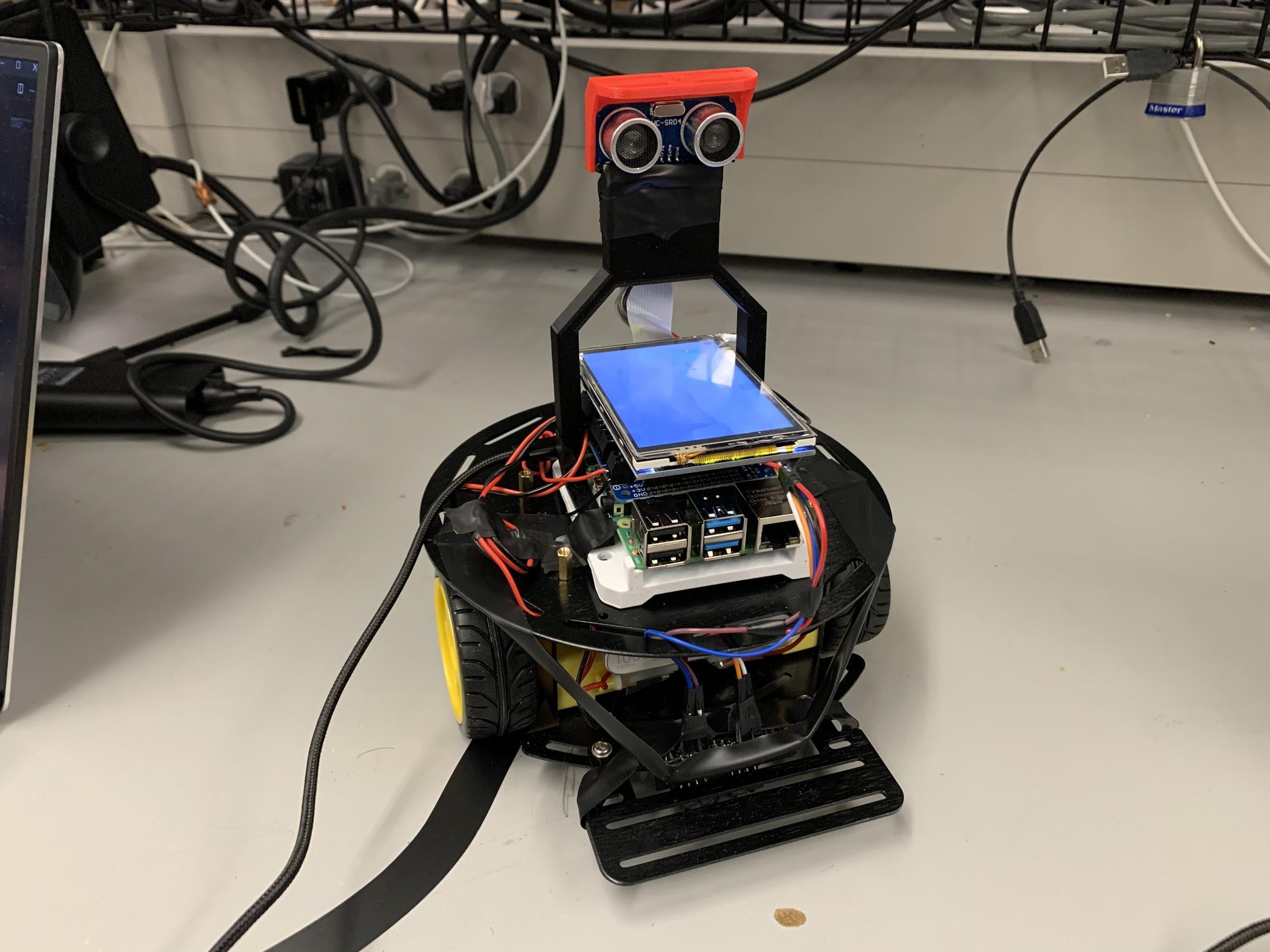
**F. References and bibliography**

Provide any relevant references.

Also include the list and description of the files submitted for this lab (including code and Fritzing breadboard view)

**Appendix A – Robot pictures**

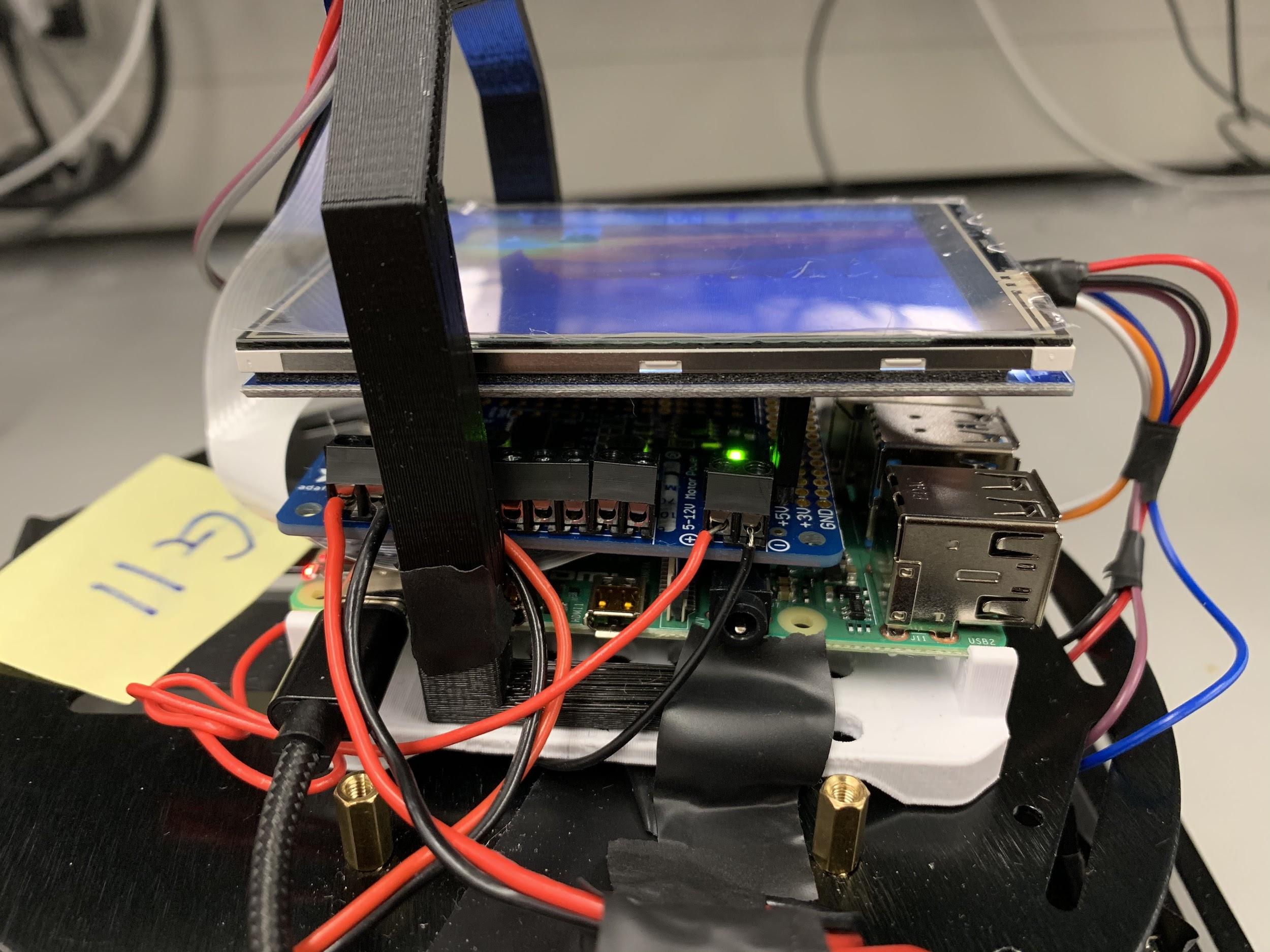
A-1) Front, top, and side view of our robot



A-2) wiring and position of our optical sensors



A-3) wiring and placement of the Motor Hat and Raspberry Pi on the robot



A-4) Back view of our robot with camera and sonar sensor wiring

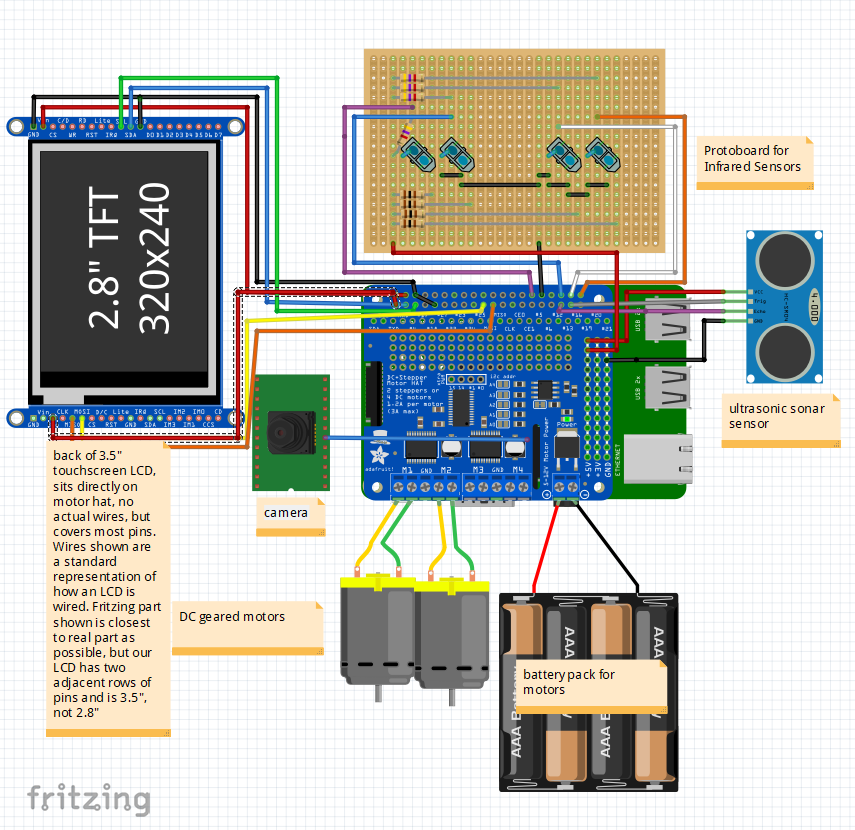


**Appendix B - Code**

Include the complete Python code with comment statements. This code must be the same code as the files you demo and submit. Clearly identify the portion of the code for the main functionality and the Additional functionality.

The code must be readable, with proper indentation, syntax highlighting (that is, copy with colour coding), and on white background. The code must be in text (that is, absolutely no snapshots of the code).

**Appendix C - Fritzing**

****

**Appendix D - GitHub**

There was an even spread of the workload amongst our group members, with some of the work being done together on one computer while debugging and testing directly with the Pi, and a good segment of the work, especially during the portion of the project where we used separate files, that the group members all worked on their own computers. This process was not perfect, as sometimes all the commits would be from a single account for a period of time, and this also doesn’t account for the time spent on the hardware and fritzing implementation.

**Appendix E – Complete Component list**

Include the list of all the components used for the project.

If you have used any component you have purchased on own, include full info, a link to datasheet, and cost.

* Raspberry Pi
* 2WD Mobile Platform robot kit
* Motor Hat
* 2x DC Motors
* 3.5” TFT LCD Display for final demo, 1” TFT LCD for first demo.
  + <http://www.kumantech.com/kuman-35-inch-touch-screen-tft-lcd-display-spi-with-touch-pen-for-raspberry-pi-3b-pi-2b-pi-zero-w-pi-a-b_p0442.html?fbclid=IwAR3yYOxRVp23q_tMWj9pfTLT0T5kVnOPfuhFs4RfwVZ2mXSb20nyRVzxNJ0>
  + <http://www.lcdwiki.com/MHS-3.5inch_RPi_Display>
  + Approximately CAD $30.00
* RPi Camera attachment
* Ultrasonic Sensor
* 5x AAA Batteries
* Android phone

**Appendix F – Answer the following questions:**

Q1 – Teamwork: Explain in detail the methods your group has used to communicate effectively among team members.

To streamline the development process, we clearly delegated tasks to each teammate both through a README on Github as well as in person, wherever possible. We developed a group culture of openness and honesty so that every team member was able to contribute and test their ideas effectively. This also allowed for extensive constructive criticism to occur so that we could rapidly prototype effectively and develop and polish a large breadth of ideas fast into our core additional functionalities and core features.

To maintain a consistent and efficient level of communication, our team used a messenger group chat to stay in touch while not meeting face to face. This was especially practical during the reading break since many of our team members were off-campus at this time, and yet we were still able to coordinate meetings, delegate tasks, and to complete the project during this time.

Another way we ensure that everyone is working equitably for the project is through the analysis of our Github commits in terms of lines added and removed, past peer reviews, and qualitative milestones reached for the hardware and for the report. This allows for a simple diagnosis of everyone’s workload, and gives the opportunity to each teammate to see where they may be falling behind, and thus incentivizing consistent self-improvement for the teammate as well as ensuring we meet deadlines.

Q2 – Design Process for the additional functionalities: Describe clearly the process you used for the following design aspects of your own additional functionalities. Please spend time to carefully answer each of them.

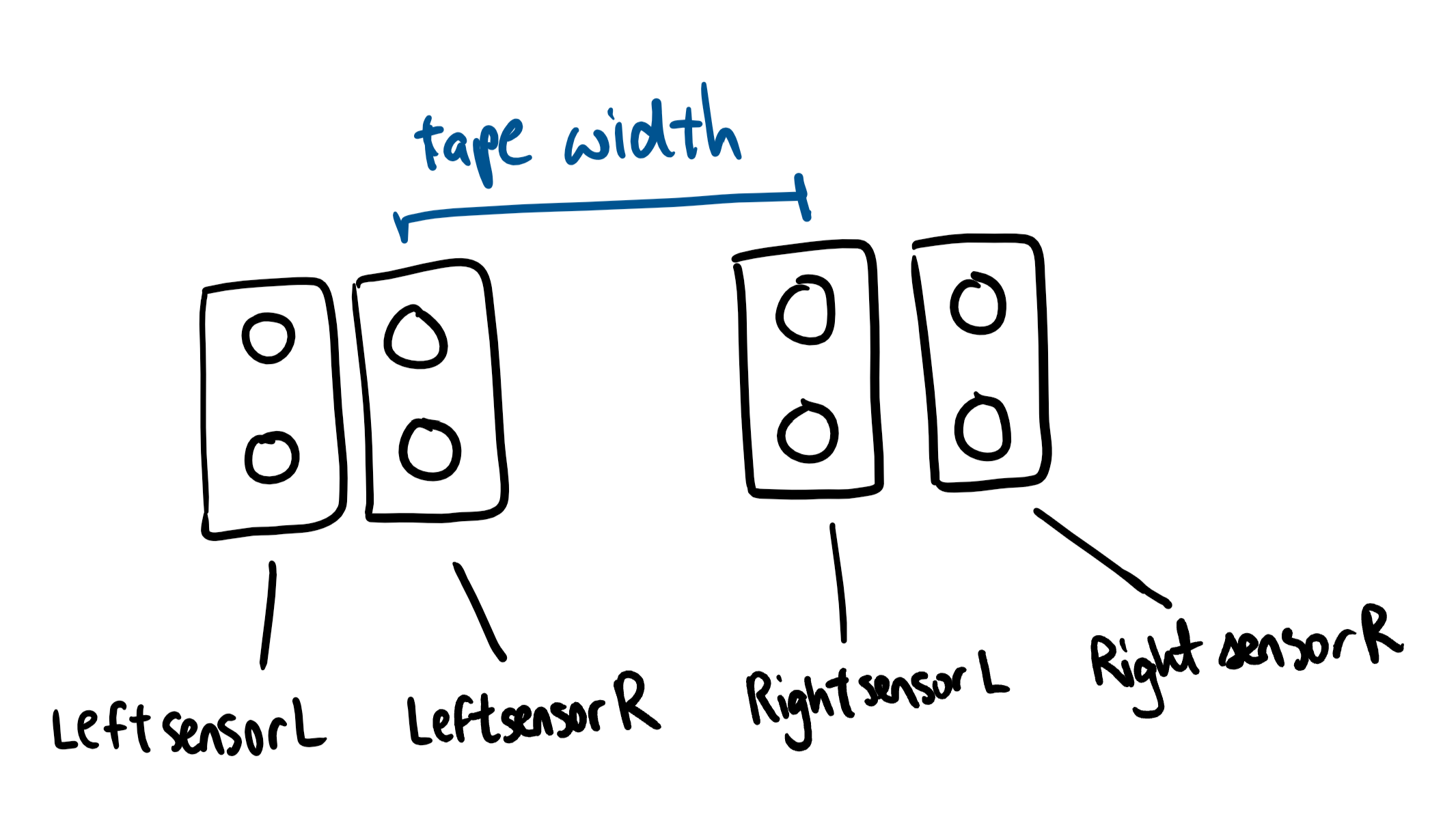
1. **Use of process**: Describe your approach to adapt and apply a general design process for any additional feature. What was your approach?

We used an iterative rapid prototyping process to incrementally add new features to our line tracking robot. We began with the base requirements given to us by the lab description, then went on to address the open-ended problems we still saw with the system, adding multiple new features to improve client engagement and to be able to fully visually display the data we received from the RPi.

1. **Constraint identification**: Explain the constraints that you must consider in the design of the additional functionalities.
   1. Hardware constraints: The main hardware constraint with this project was limiting the pins used by all of our smaller components since the LCD uses so many pins. Another factor that added to this was how to design a circuit that is clean and safe in a dense space, and which will not short with the metal casing of the robot or within itself. This was made more critical as we switched away from the breadboard to a more permanent protoboard solution.
   2. Software constraints: There were three very large software constraints we faced, getting reliable PID values, developing an app which was compatible with the RPi and our other hardware/software, and integrating additional features into the robot while maintaining the relative timing of the robot.
      1. The PID values were determined according to the Ziegler-Nichols method as described earlier, but since the battery level, ambient light, angle of the sensors, shadows, and other power drawing components were added and removed as we tested each day, the values we determined for the PID were effectively obsolete whenever we made a change. This allowed us to become more efficient in determining new PID values which can turn quickly, smoothly, without oscillation, while still being able to stop fast and make sharp turns.
      2. We decided to make an android app which controls the robot via bluetooth. We began developing on android studio but realised MIT App Inventor had a better interface for new app developers. We made an app which connects to a bluetooth server and sends text when a button is pressed. The text was decoded on the server side. Due to some technical anomalies, when the app client connects to a bluetooth server, the app stops responding and closes.
      3. Since the reliability of the robot movement is heavily dependant on getting very accurate readings as quick as possible, we had issue adding features like dynamic track plotting and wall detection, as those increased the latency of our robot either directly in the case of the sonar, while has to have sleep function calls to align its readings, or in the case of the tkinter graph, which requires too much time to virtually add points to the plot and make complex calculations at the rate of 3000Hz we were using for the IR sensors.
   3. Quality vs Quantity Tradeoff consideration: A consideration our group always keeps in mind is the feasibility of our ambitions for a project given its deadline and requirements. One limitation our team sometimes has, and one which we are consistently fixing, is that we try to add to many additional features into a project before we can get the base functionality off the ground. While this allows for a more holistic view of a project even from its early stages, it can overshadow the main goal of the project until the last portion of the project.
2. **Solution generation**: Explain at least two possible alternative additional features that your group rejected due to technical reasons and explain why.
3. Bluetooth App with multiple features like live stream of pi camera, default mode for line tracking, joystick control for moving the robot and controlling the sensors connected to the raspberry pi. The live streaming was rejected because the feature required web development which was not recommended for project 1. The other features of the app were not tested because we could not get the basic functionality of sending information from the client to the server in bluetooth due to technical anomalies.
4. Using servos to control the Ultrasonic Sensor angle for wall detection. This feature was supposed to be implemented for the wall detection mode of our line following robot but it was rejected because of space limitations on that robot and code conflicts while trying to move, detecting wall and torating the sensor.
5. **Solution Assessment**: Explain how you tested and assessed the viability and then correctness of your group’s additional features.
   1. The correctness of the sonar, twitter, and track plotter were all assessed by the accuracy of their response to their predefined function conditions. We can easily measure if the sonar is measuring X cm with a ruler or by eye, and the track plotter final image has to look like the tape below it or else it would need refining. The twitter bot can easily be verified with print statements and checking our team’s twitter account for updates.
   2. The mobile app control’s correctness was verified when it was able to connect to and turn the robot at the user’s will.
   3. The robot’s ability to make turns, speedup, and stop was tested by how well it adhered to the track at its defined speed (which speeds up on straight patches), how little oscillation and jittering there is in its adjustments, and how sharp of a turn it can handle. This correctness was wholly dependent on the accuracy of the PID value determination process outlined before.

**Appendix G - Other**

G-1) Reflective optical sensor configuration diagram



G-2) Graphical representation of GUI graph algorithm

