Maze Solver Control Program: Line Following + Sonar Sensing

Neel Choudhary
Nicole Danuwidjaja

Project Proposal

Goal: Write a robot control program to successfully solve a maze at a given point using line following and sonar sensing.

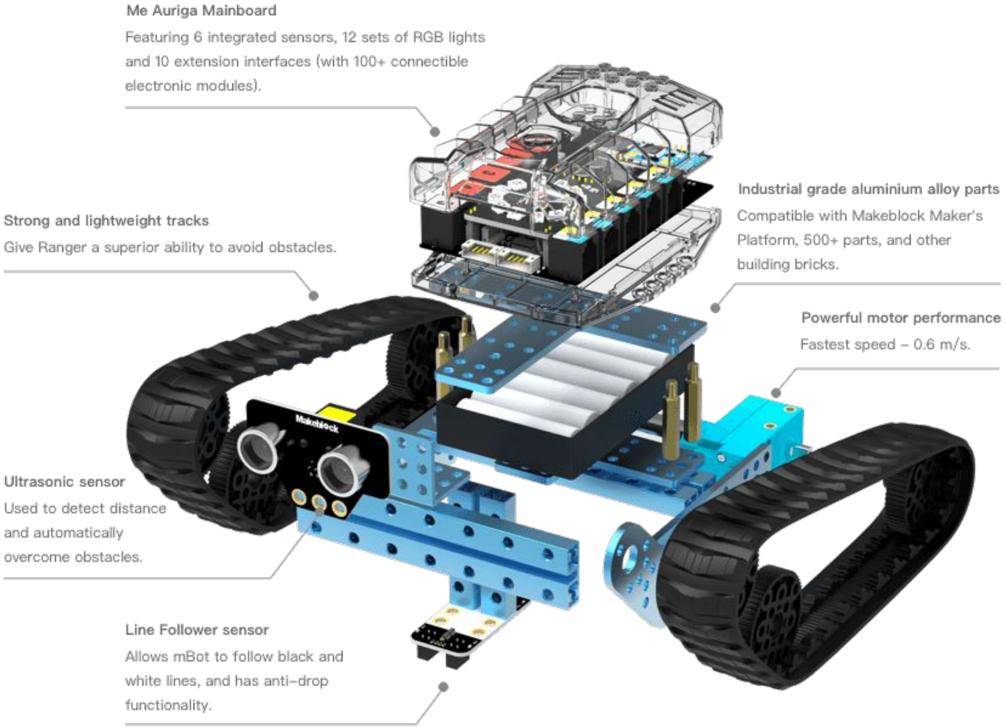
- Robot moves autonomously using line following techniques to navigate through maze
- Identify surroundings using sonar sensor to find next best path
- Use sonar sensor to identify the goal.

Project Breakdown

1. **Construct physical mBot Ranger Robot**
2. **Establish computer/mBot serial communication**
3. **Connect firmware and write functional code**
4. **Determine maze layout and build physical course**
5. **Create models to replicate maze in Gazebo**
6. **Profit!**

mBot Ranger (Physical Robot)

- Modified robot to have additional line following sensor
- Utilized ultrasonic sensor and sonar sensor data



Me Auriga Mainboard

Featuring 6 integrated sensors, 12 sets of RGB lights and 10 extension interfaces (with 100+ connectible electronic modules).

Strong and lightweight tracks

Give Ranger a superior ability to avoid obstacles.

Industrial grade aluminium alloy parts

Compatible with Makeblock Maker's Platform, 500+ parts, and other building bricks.

Powerful motor performance

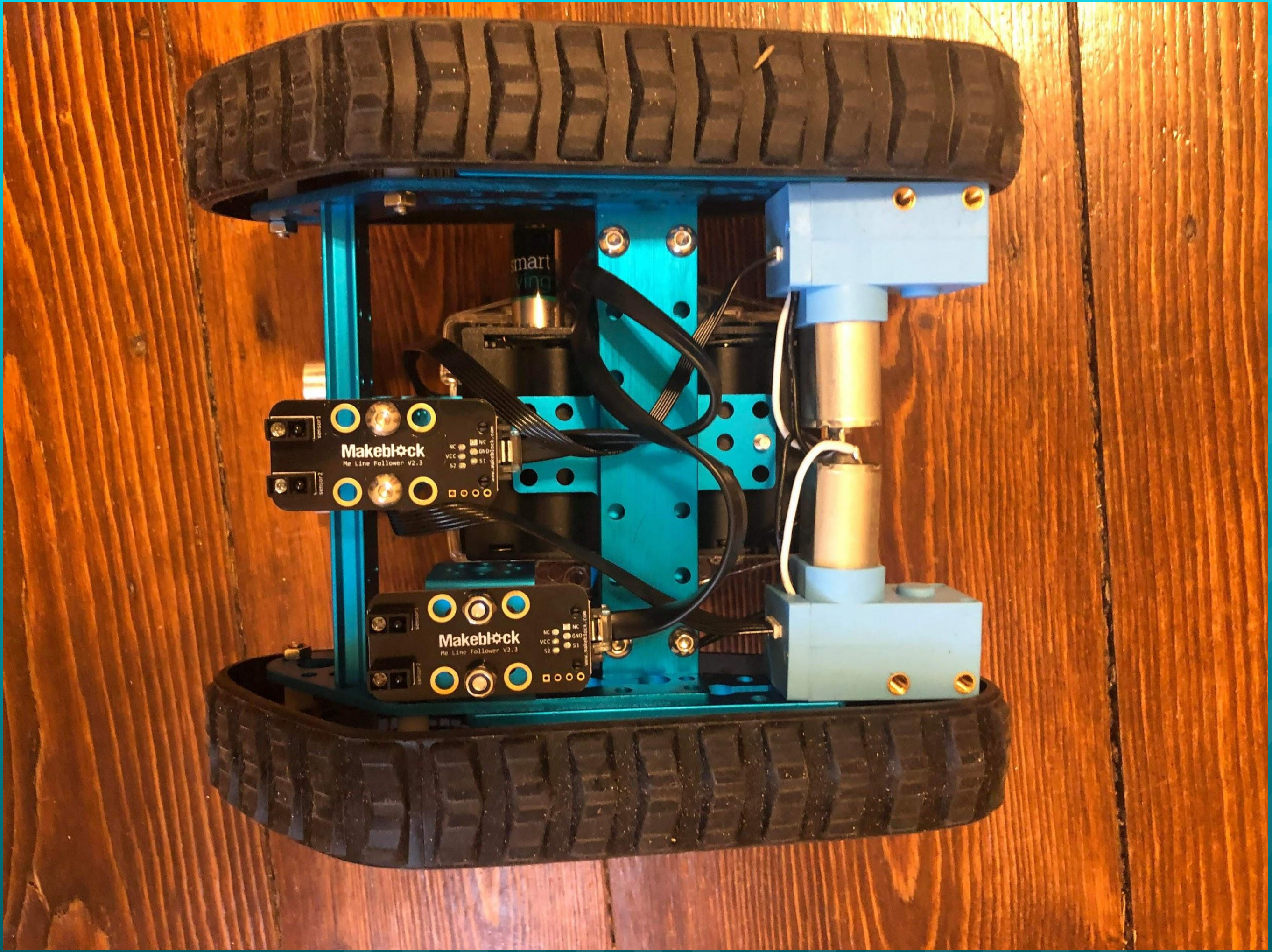
Fastest speed – 0.6 m/s.

Ultrasonic sensor

Used to detect distance and automatically overcome obstacles.

Line Follower sensor

Allows mBot to follow black and white lines, and has anti-drop functionality.

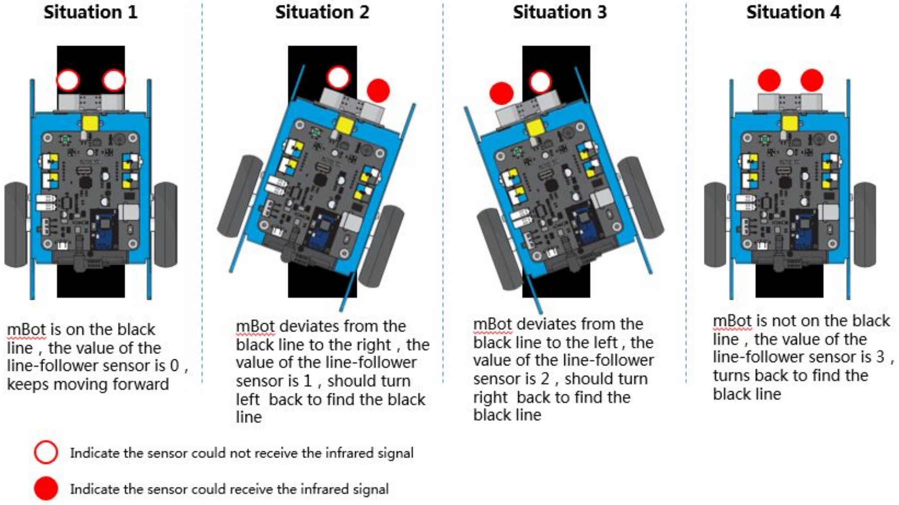

Makeblock
Me Line Follower V2.3
Makeblock
Me Line Follower V2.3
sensor1
sensor2
NC
VCC
S2
GND
S1
smart

Maze Construction

Used black duct tape with a 1.88” width for line following

Had to use an empty room to avoid sonar sensor interference.

Had a large object in the middle to signify the end of the maze.



Situation 1

mBot is on the black line , the value of the line-follower sensor is 0 , keeps moving forward

Situation 2

mBot deviates from the black line to the right , the value of the line-follower sensor is 1 , should turn left back to find the black line

Situation 3

mBot deviates from the black line to the left , the value of the line-follower sensor is 2 , should turn right back to find the black line

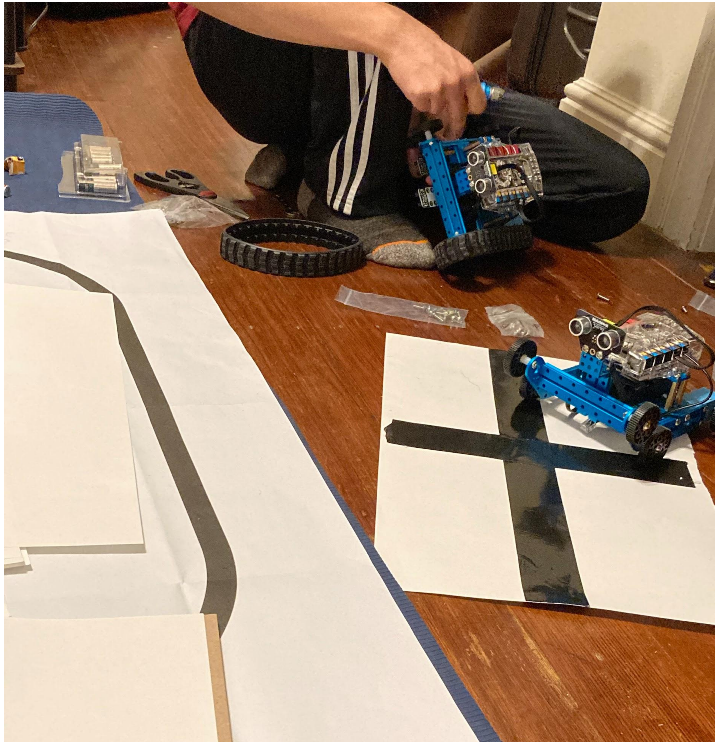
Situation 4

mBot is not on the black line , the value of the line-follower sensor is 3 , turns back to find the black line

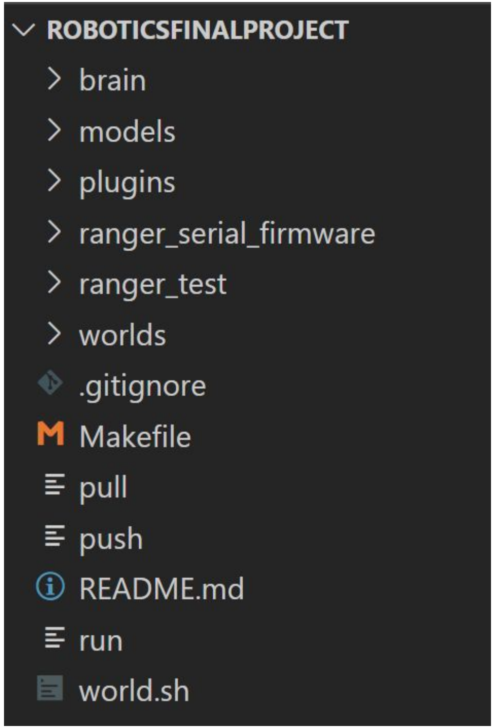
○ Indicate the sensor could not receive the infrared signal

● Indicate the sensor could receive the infrared signal

Progress



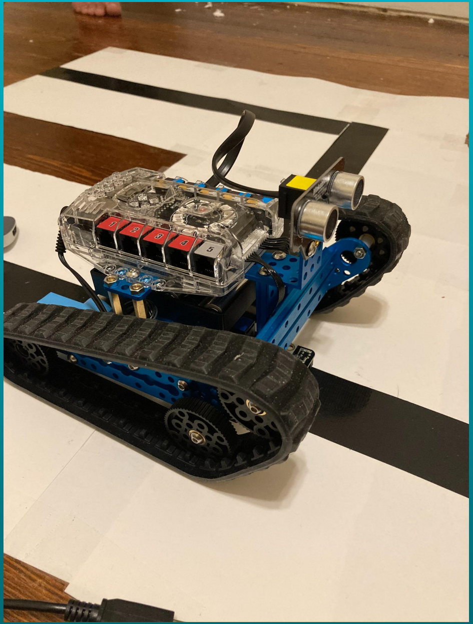
Repository

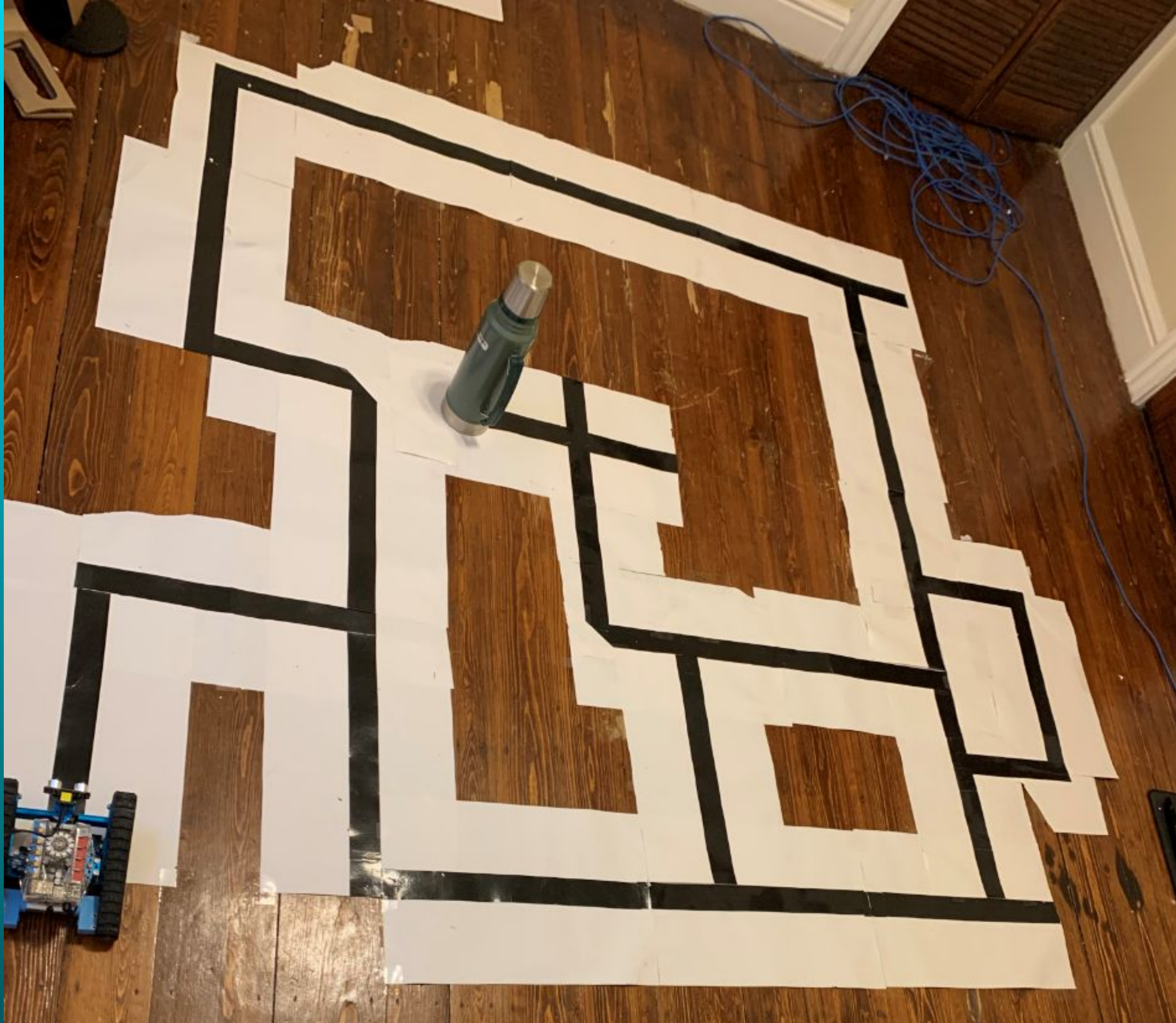


ROBOTICSFINALPROJECT
- brain
- models
- plugins
- ranger_serial_firmware
- ranger_test
- worlds
- .gitignore
- Makefile
- pull
- push
- README.md
- run
- world.sh

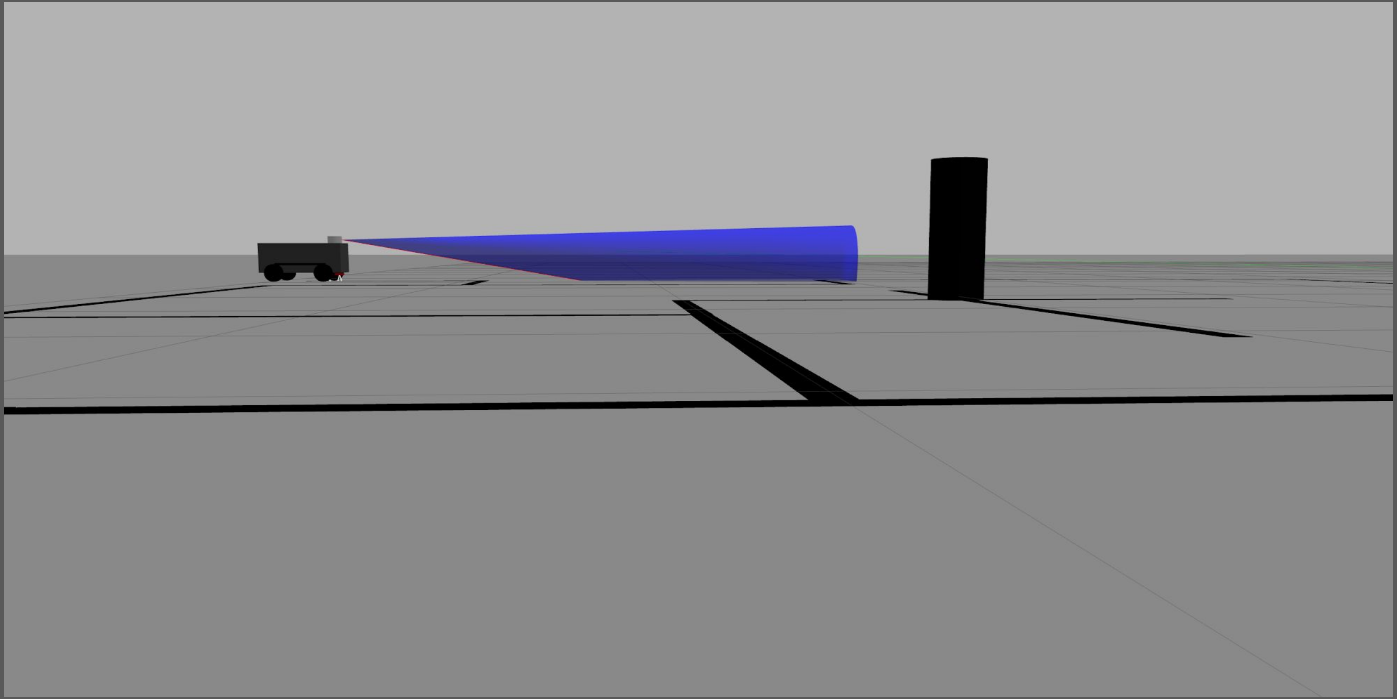
- **/brain:** contains mBot Ranger libraries, Robot classes, and control program for virtual/physical bots
- **/models:** Gazebo model files with plugins to represent objects virtually
- **/plugins:** Shared Gazebo libraries for models and elements
- **/ranger_serial_firmware:** installation of MeAuriga firmware Sketch
- **/ranger_test:** control program + firmware Sketch

Physical Maze

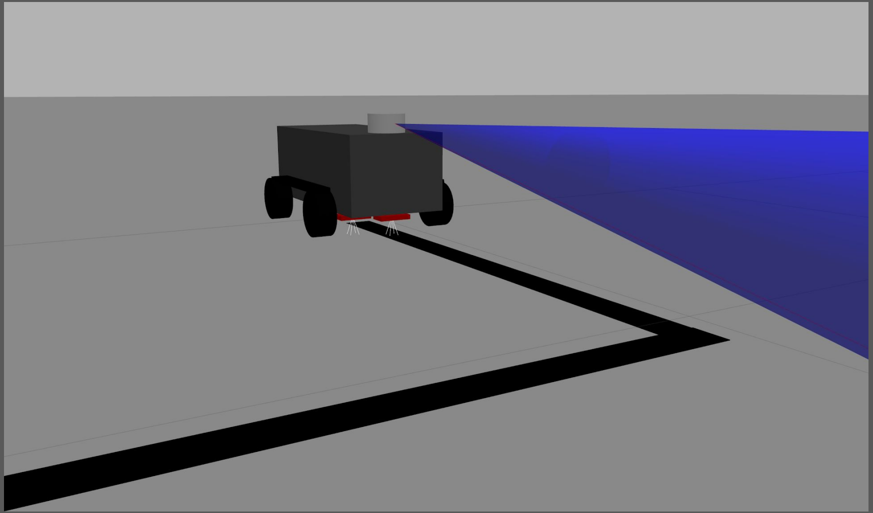


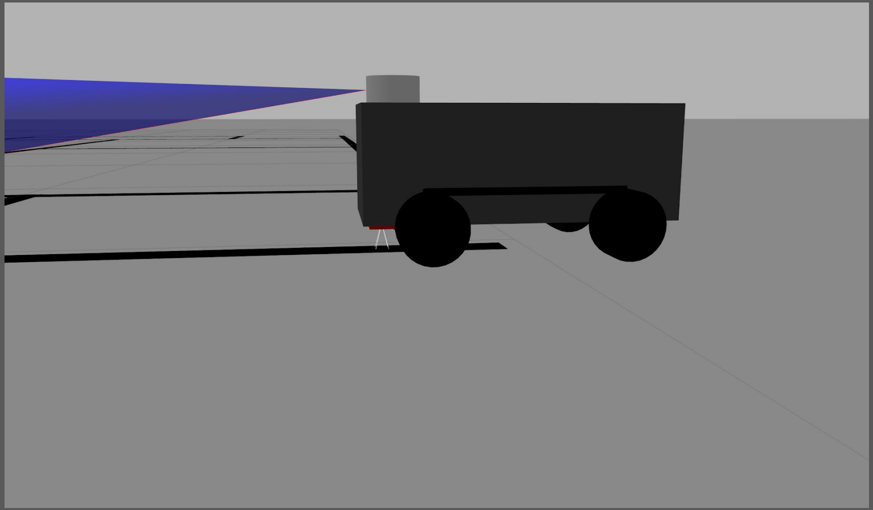


Gazebo Simulation

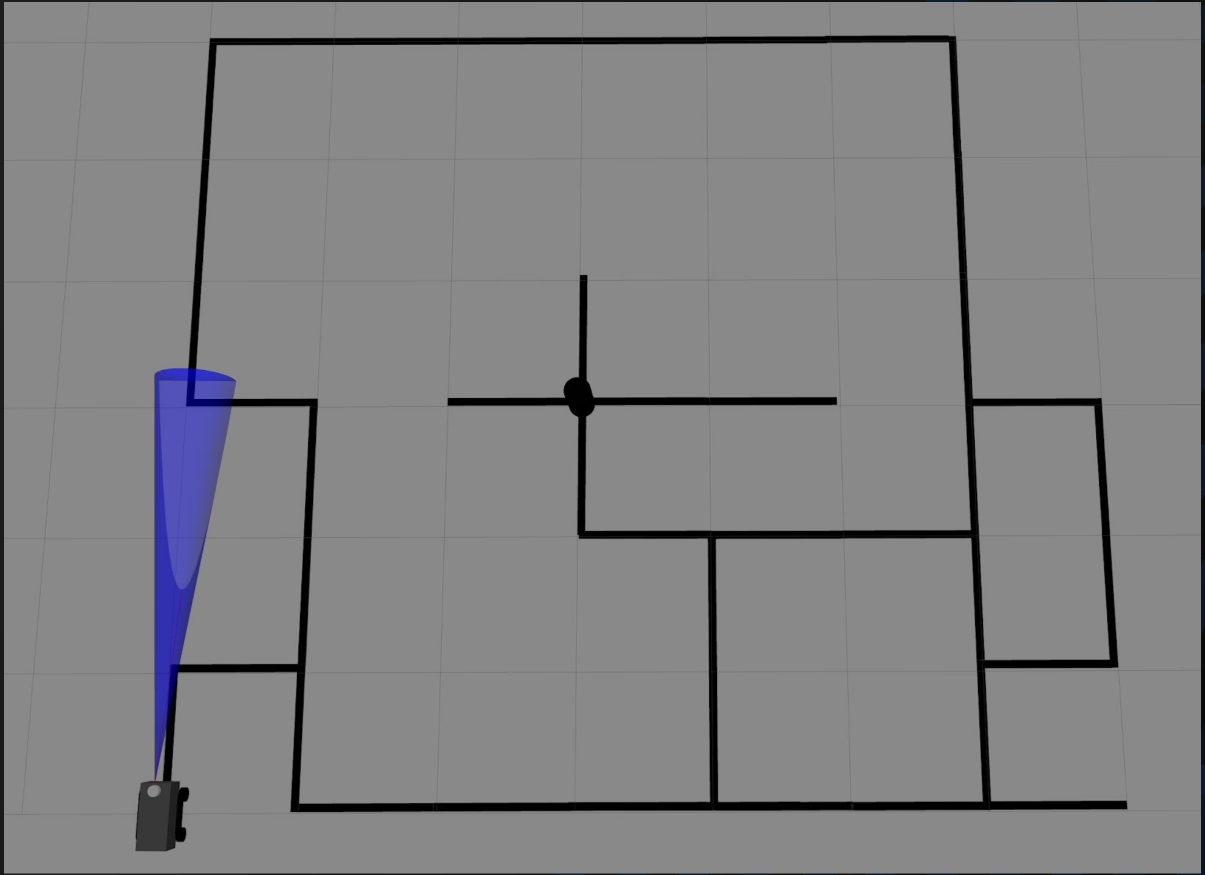


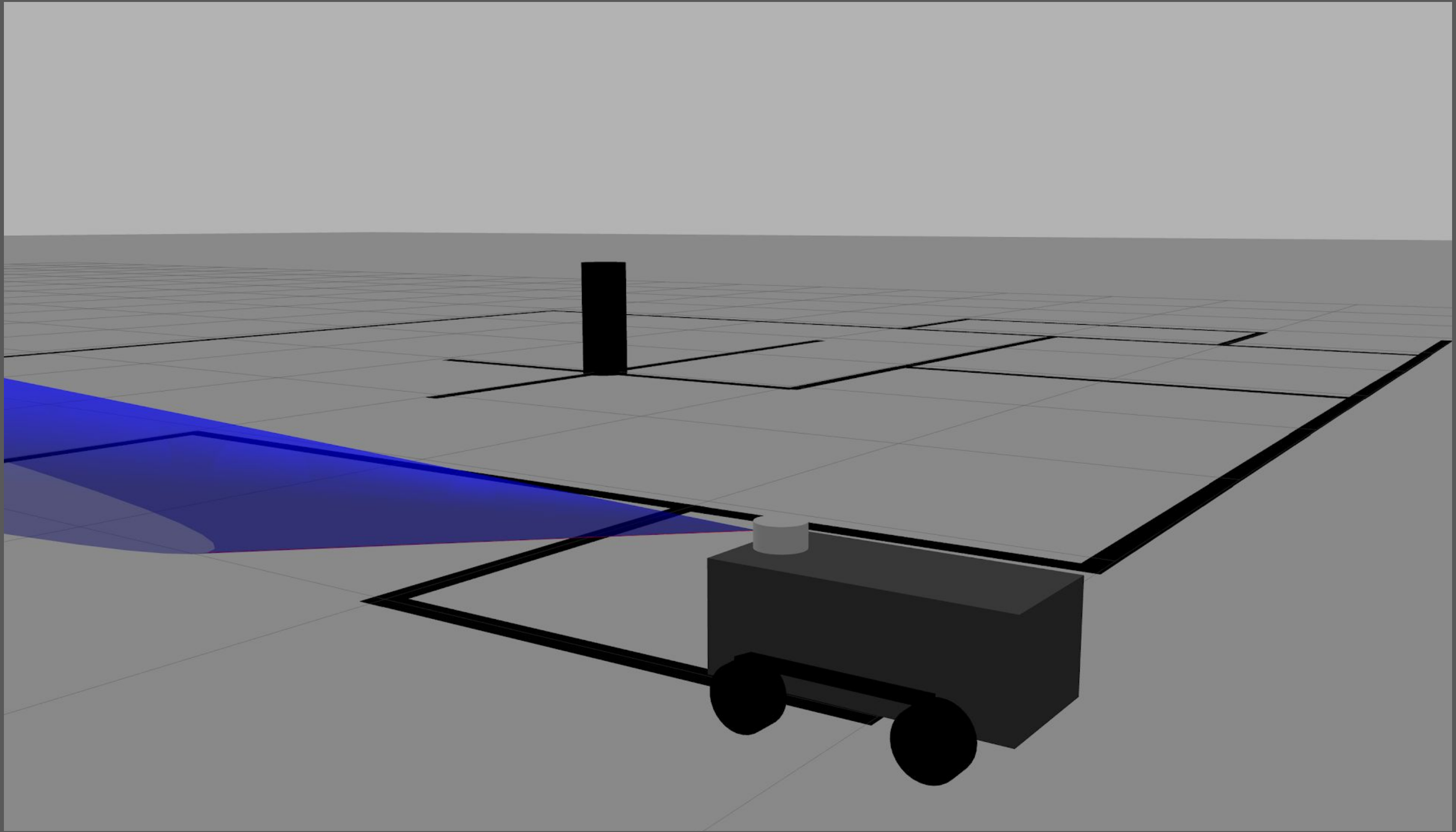
Gazebo mBot Model

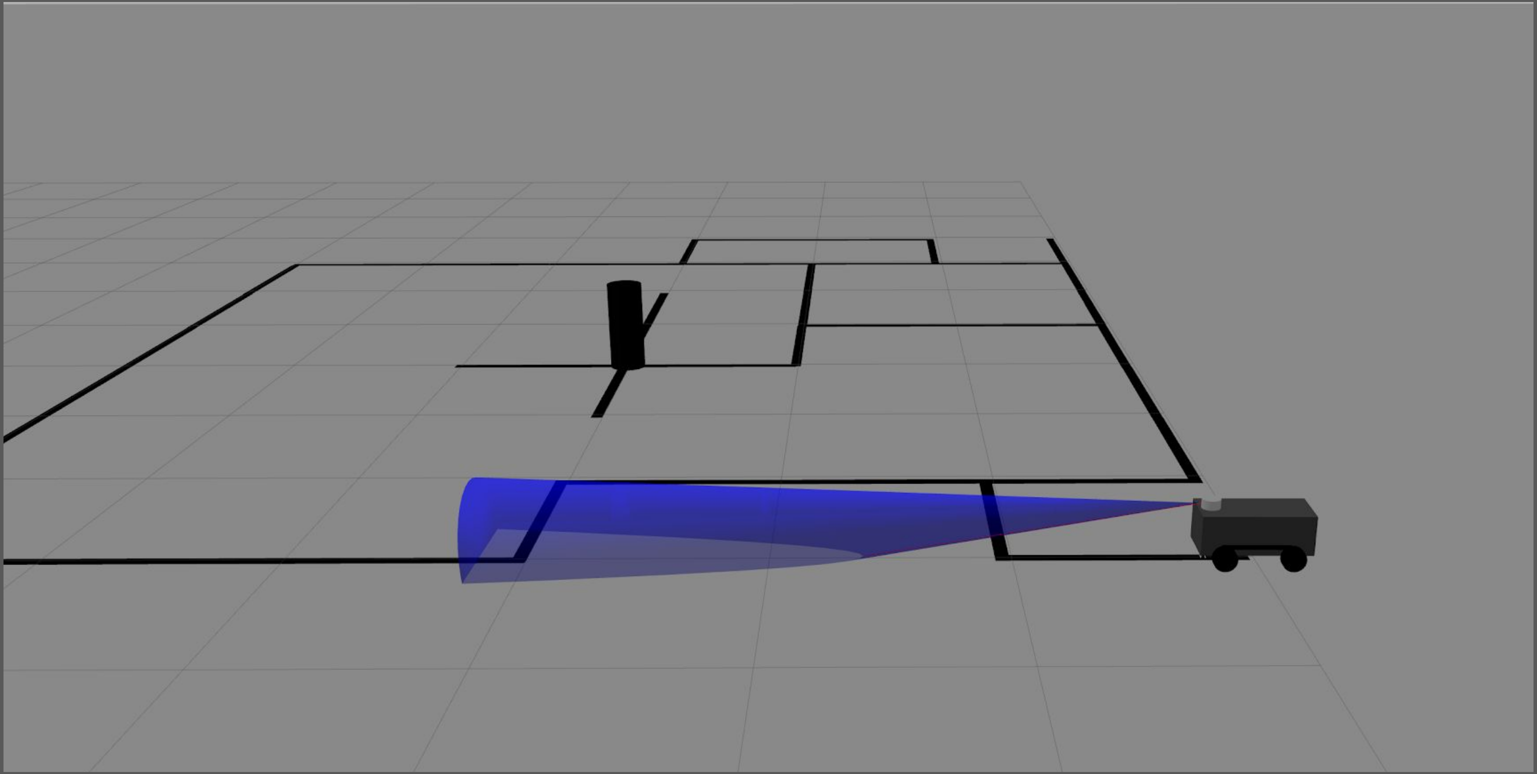


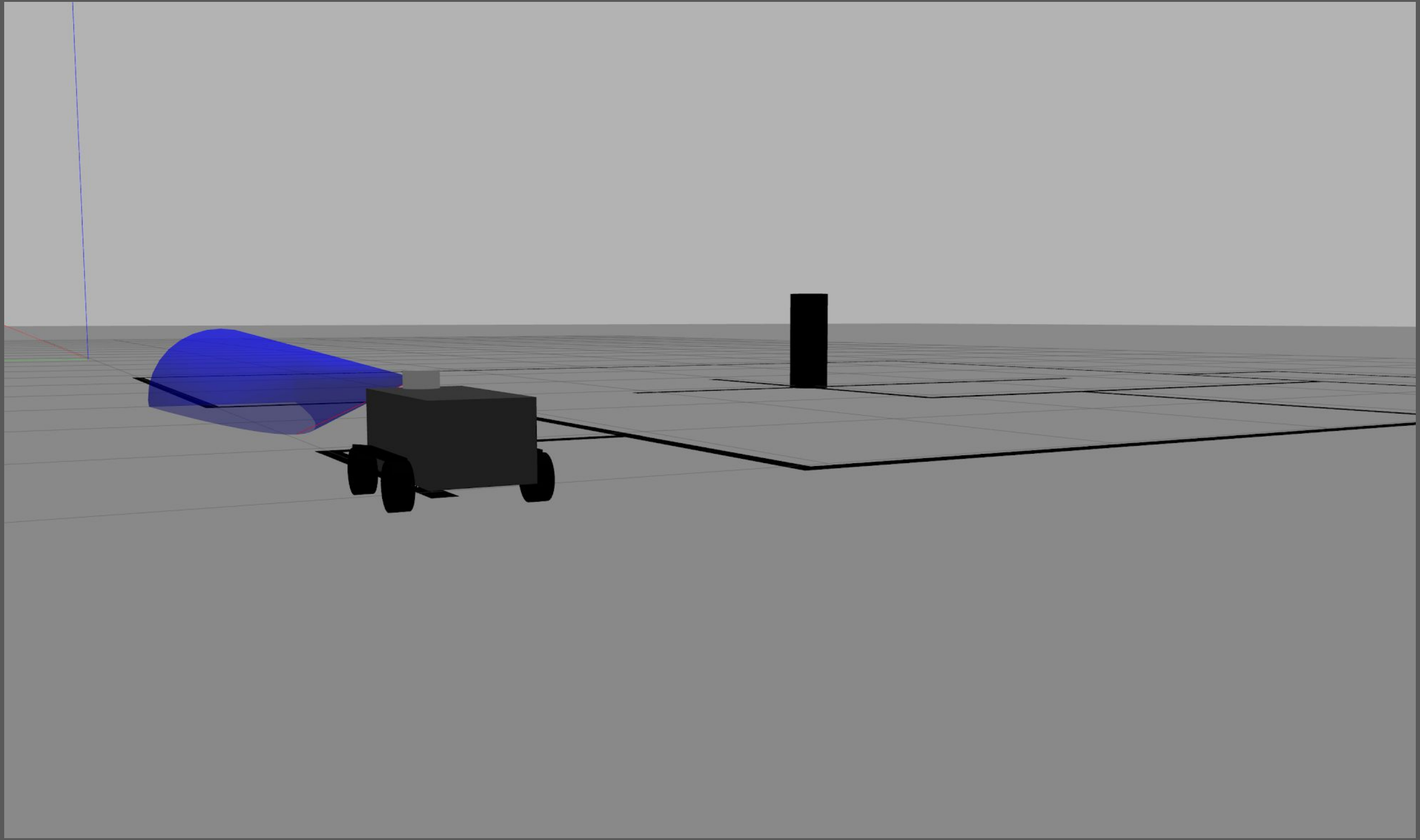


Virtual Maze



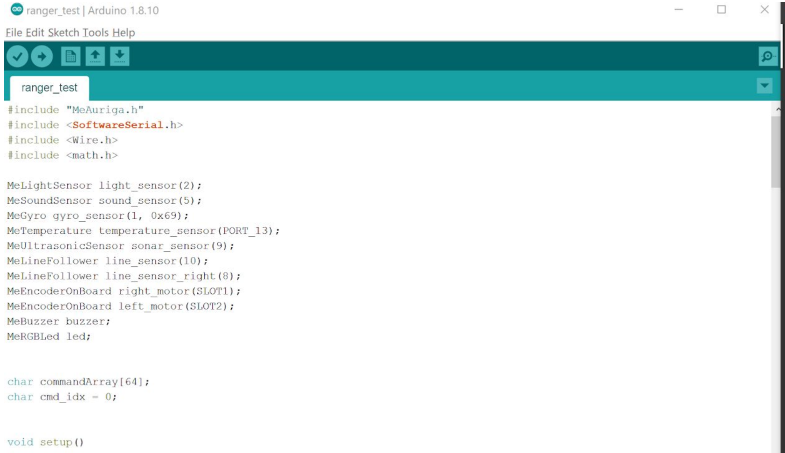






Arduino Serial Communication

- Based our serial communication of Nolan’s code.
- Modified code to send all the sensor data as one string instead of sending each data separately.
- Ended up not using serial communication at all.

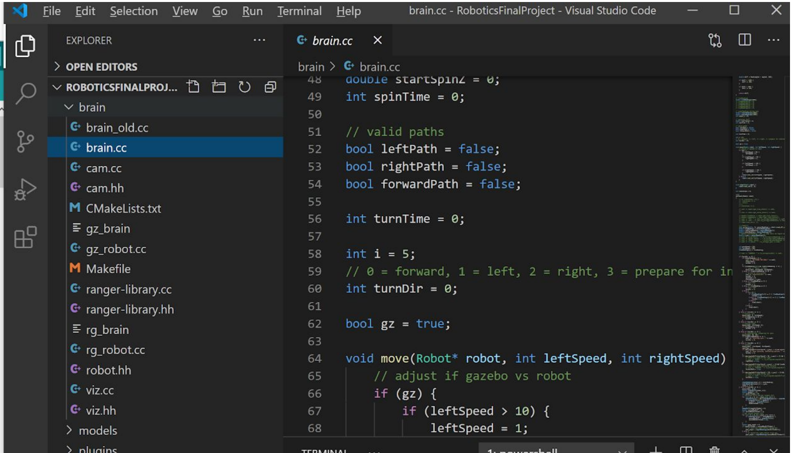


```
#include "MeAuriga.h"
#include <SoftwareSerial.h>
#include <Wire.h>
#include <math.h>

MeLightSensor light_sensor(2);
MeSoundSensor sound_sensor(5);
MeGyro gyro_sensor(1, 0x69);
MeTemperature temperature_sensor(PORT_13);
MeUltrasonicSensor sonar_sensor(9);
MeLineFollower line_sensor(10);
MeLineFollower line_sensor_right(8);
MeEncoderOnBoard right_motor(SLOT1);
MeEncoderOnBoard left_motor(SLOT2);
MeBuzzer buzzer;
MeRGBLed led;


char commandArray[64];
char cmd_idx = 0;


void setup()
```



```
double startSpinZ = 0;
int spinTime = 0;

// valid paths
bool leftPath = false;
bool rightPath = false;
bool forwardPath = false;

int turnTime = 0;

int i = 5;
// 0 = forward, 1 = left, 2 = right, 3 = prepare for in
int turnDir = 0;

bool gz = true;

void move(Robot* robot, int leftSpeed, int rightSpeed)
    // adjust if gazebo vs robot
    if (gz) {
        if (leftSpeed > 10) {
            leftSpeed = 1;
```

Control Program

The control program was set up to interact with the firmware.

Implemented line following and error correction when the robot deviates from the black line.

Implemented logic for detecting an intersection and deciding where to proceed based on all possible paths and the angle to the goal.

Interesting Components

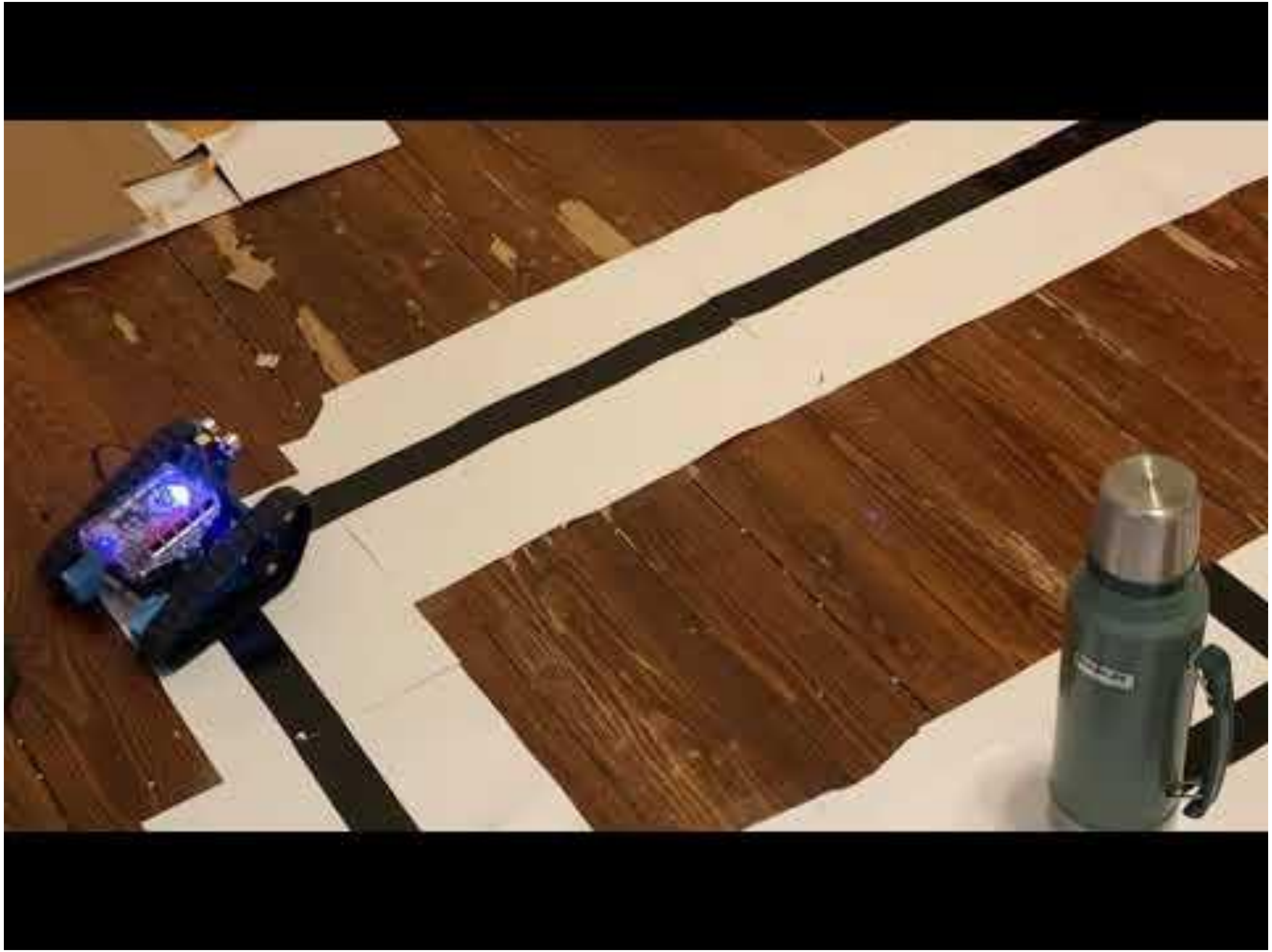
- Control program worked across different mBot subsystems
- Firmware compression enabled us to load entire control program onto mBot Arduino
- mBot Modding for multiple line following sensors

Expected Challenges

- Implementing maze solver algorithm
  - Backtracking? Additional sensor data?
- Porting code for both physical and digital robot
  - Tracking expected behaviors both digitally and physically

Unexpected Challenges

- Setting up firmware
- Difficulties with testing code on mBot
- Short battery life led to limited firmware capabilities
- Difficulties with establishing communication



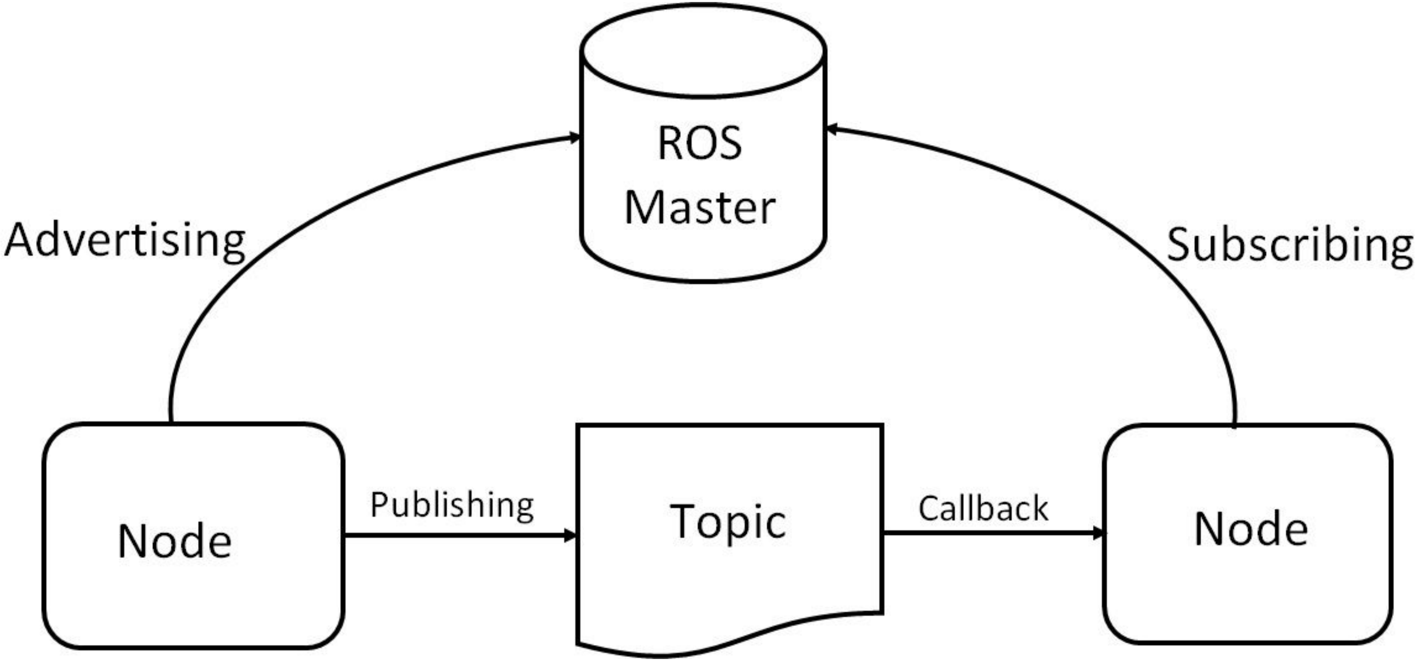
Thank you!

Neel Choudhary
Nicole Danuwidjaja

Shoutout to...

- Nolan’s team for the ranger firmware library
- https://www.mblock.cc/example/primary-line-patroling-program/
- https://github.com/senestone/A_Gentle_Introduction_To_Robotics_With_mBlock_and_mBot

Gazebo System Components


ROS Master
Advertising
Subscribing
Node
Publishing
Topic
Callback
Node