CS LAB BOT

By: Alan Deuchert

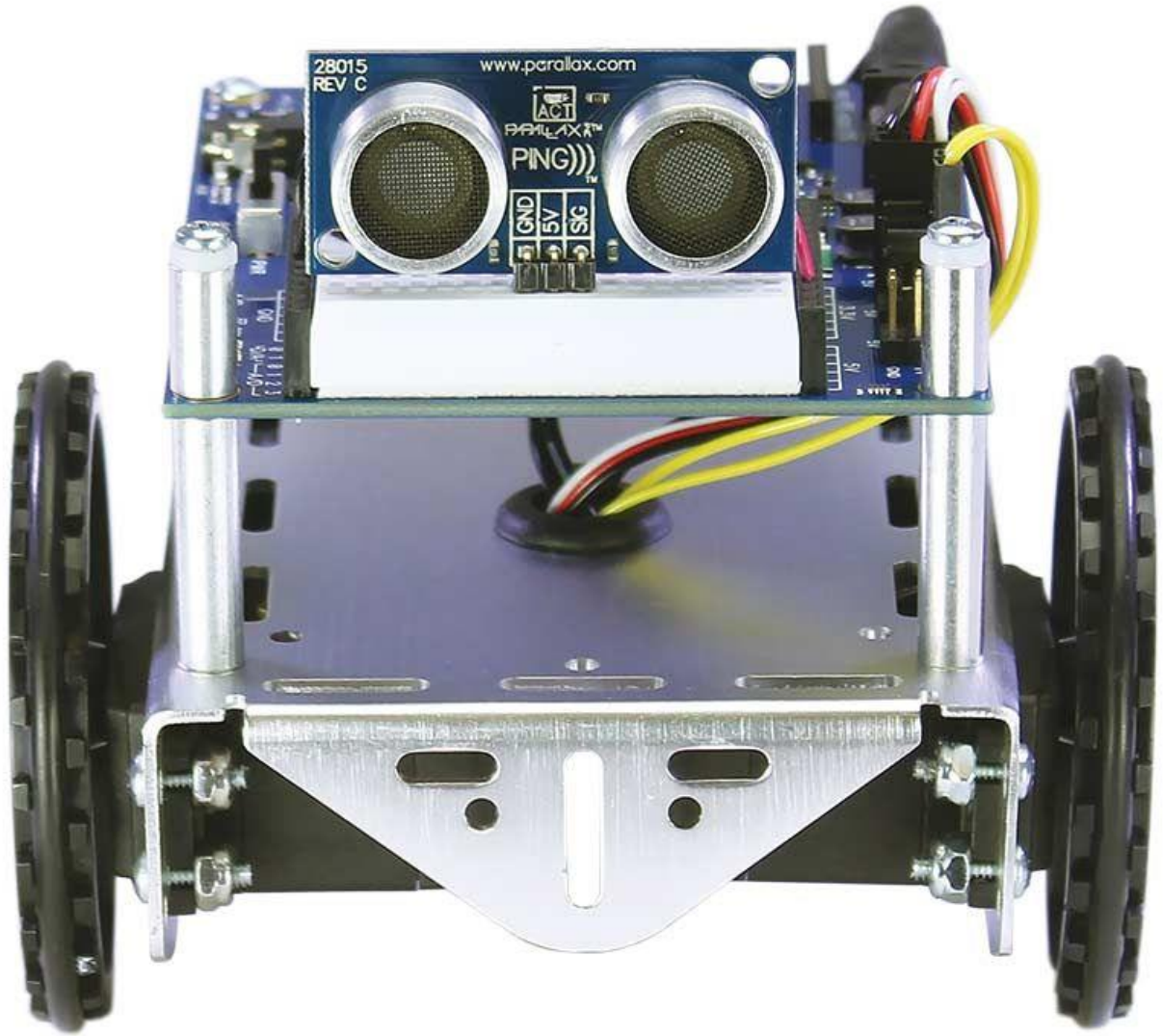
DEFINITION

- Design activities for a mobile robot mascot that will hang out and "socialize" with students in the Computer Lab.

REQUIREMENTS

1. Users can instruct the robot much like a RC vehicle.
2. Users direct the mascot to move toward a target while avoiding obstacles.
3. The lab robot should be able to perform some preprogrammed activities such as finding a target while avoiding fixed obstacles.
4. The bot should understand its environment and know the location of the contents, probably using a modified geo-mesh to map the lab layout.
5. The lab robot should avoid obstacles that appear unexpectedly.
6. Multiple users could give the robot instructions handling racing issues if necessary.
7. The bot needs a name and a personality/attitude.

THE BOT (EDDIE)



WHAT EDDIE OFFERS

- PING))) Ultrasonic Distance Sensor
- (2) wheels with a circumference of 208mm
- Feedback 360° High Speed Servos with built-in encoders

- Flavor of C Language Programming with SimpleIDE
- Parallax WX ESP8266 WiFi Module

PING))) ULTRASONIC DISTANCE SENSOR

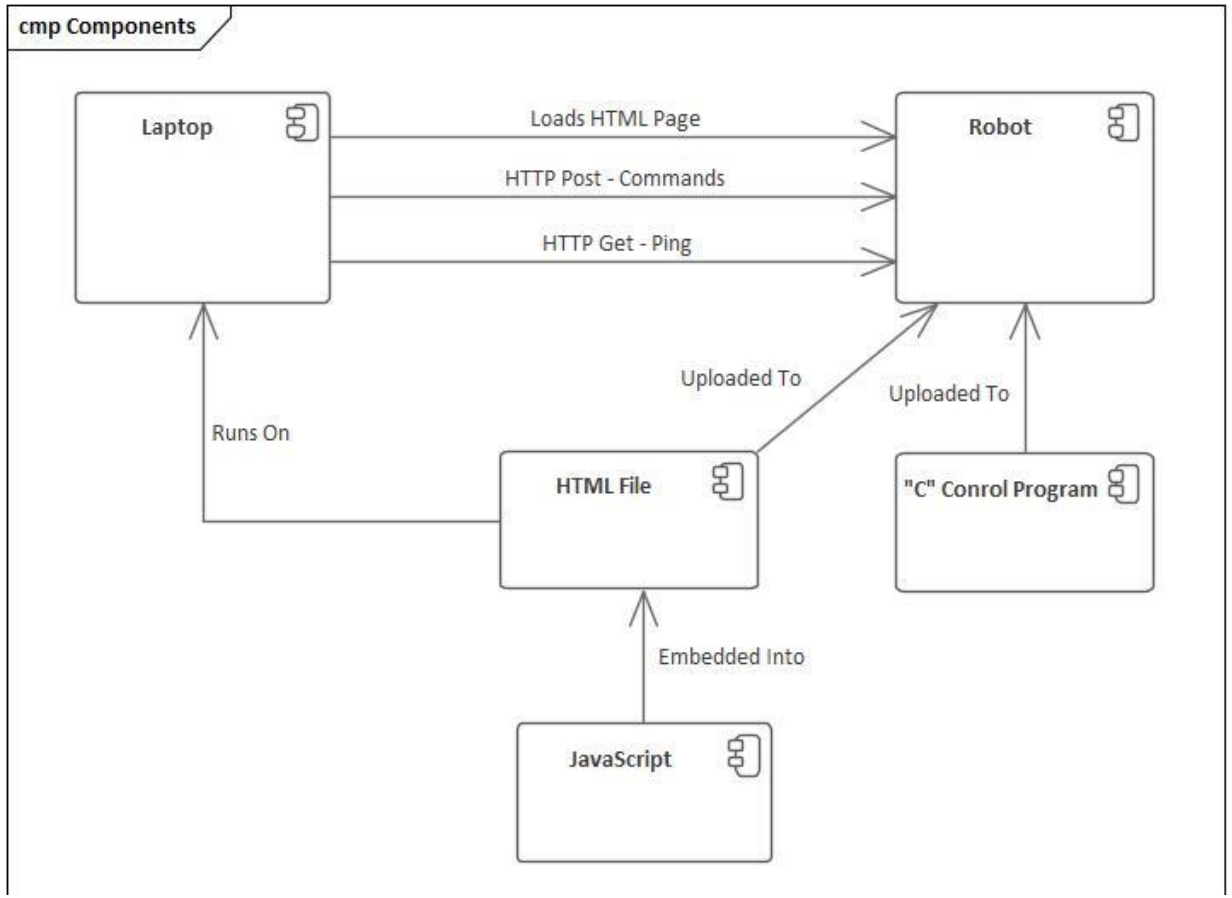
- The "eyes" of Eddie
  - Triggers an ultrasonic burst from one eye and then "listens" for the echo return pulse with the other; uses the time of this to gauge distance.
    - It is very sensitive to variation in the temperature.
    - It has more difficulties in reading reflections from soft, curved, thin and small objects.
    - It requires its mounting bracket to turn around see; can only see "head-on."

THE SERVOS AND THE WHEELS

- The function for travel distance is in units of wheel encoder ticks.
  - There are 64 wheel encoder ticks per revolution.
    - Given the circumference of the wheels, one tick = 3.25mm.
  - If the distance is not a whole number of ticks, round off to the nearest whole number; you can't specify fractions of a tick.
    - Depending on distance traveled, there is an offset.

WX WI-FI MODULE

- Creates a Wi-Fi network for external devices to connect to.
- Can upload user-created web pages that then make it possible for devices to interact with the bot through the Wi-Fi module itself.
  - In turn, the bot uses C, but also HTML, CSS, and JavaScript as well.



OVERVIEW OF COMPONENTS

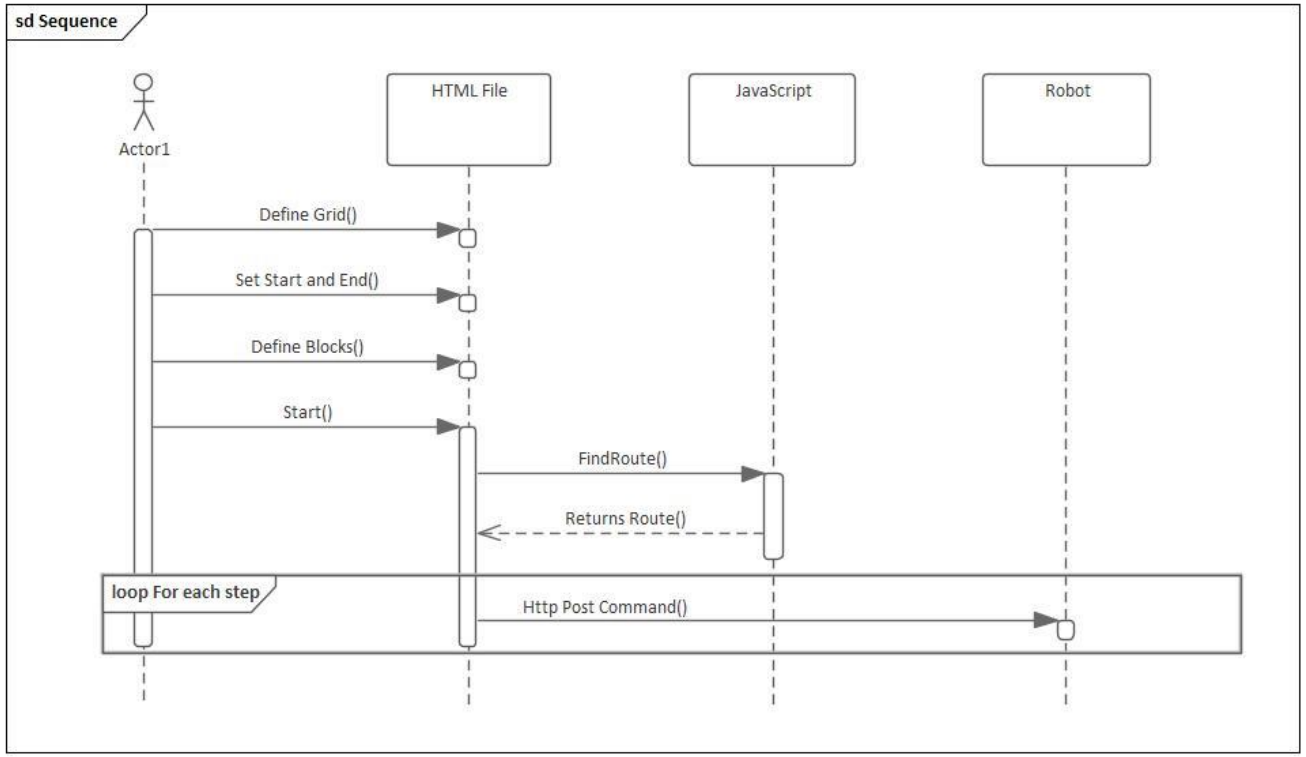
MANUAL DRIVE

- Required C and some HTML/CSS/JavaScript.
- Utilized previous senior capstone code (thanks Nathan Labott!)
  - Webpage → Bot and Bot → Webpage.
- Most important modification made was “press and hold” driving controls.
  - Focused heavily on simplicity.

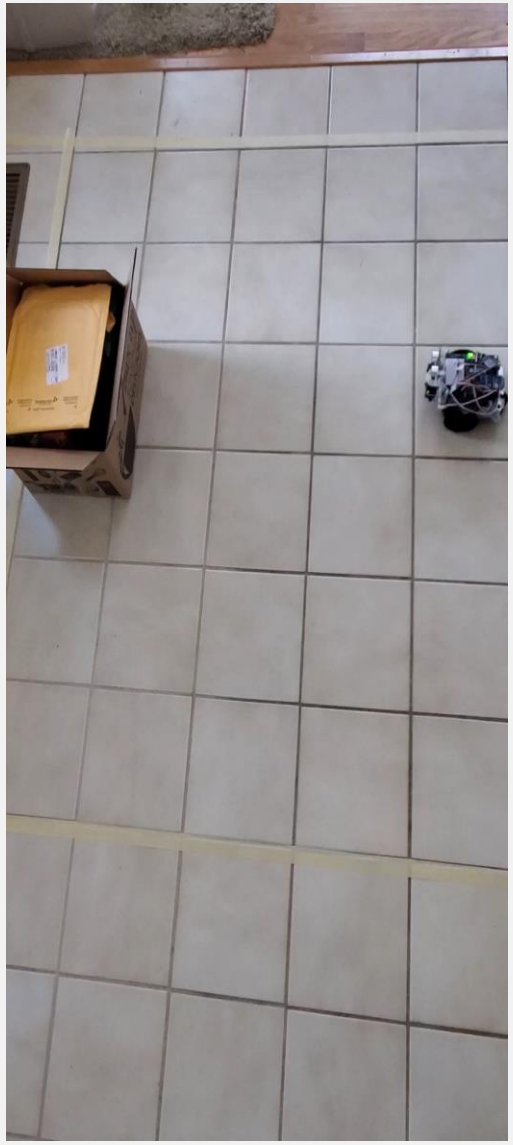
AUTOMATED DRIVE

- Required C, HTML, CSS, and *A LOT* of JavaScript
- Procedure:
  1. User inputs on a grid (the modified geo-mesh) where Eddie is, what cell it's facing, where to go, and any pre-established blocks.
  2. JavaScript creates an optimal path with recursion (in a "theoretical sense" – more on that later).
  3. Take the path given and execute it step-by-step; tell Eddie when to turn and move forward (for simplicity, Eddie does not move diagonally).
- The computer does all the heavy lifting; the bot goes where it's told to go.

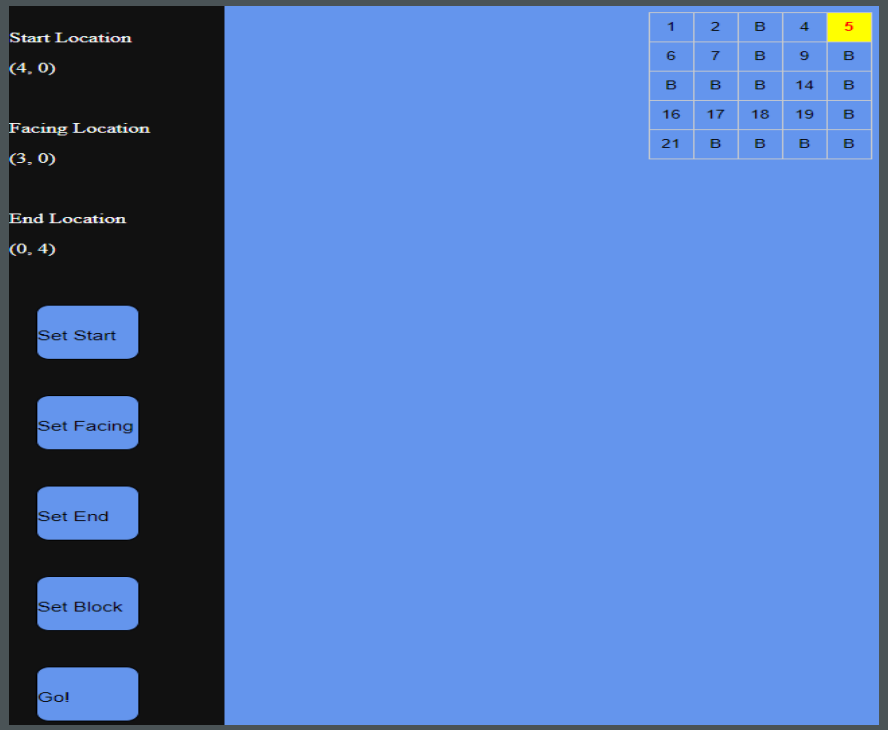
OVERVIEW OF SEQUENCING (AUTOMATED)

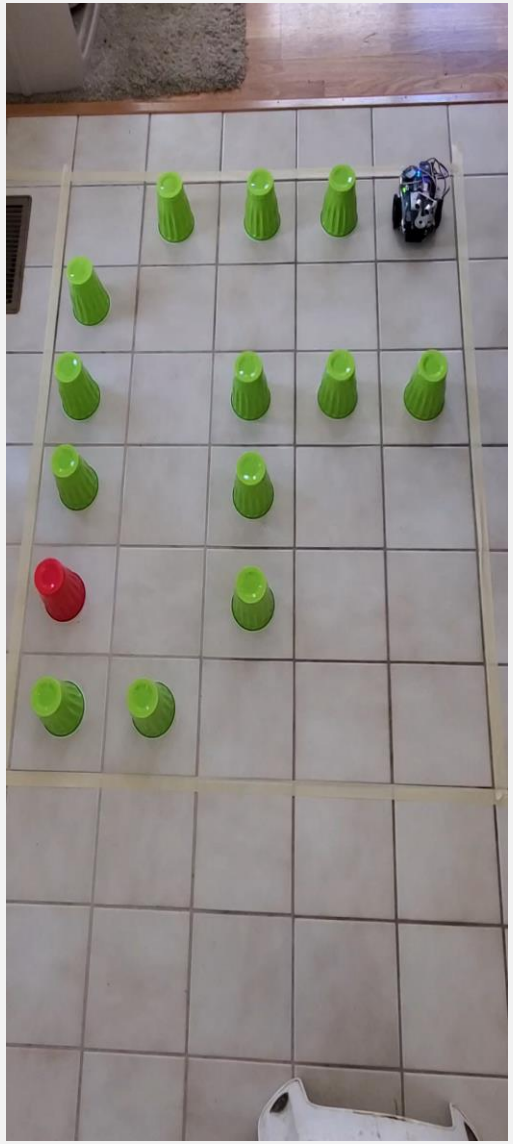


DEMONSTRATION (MANUAL)

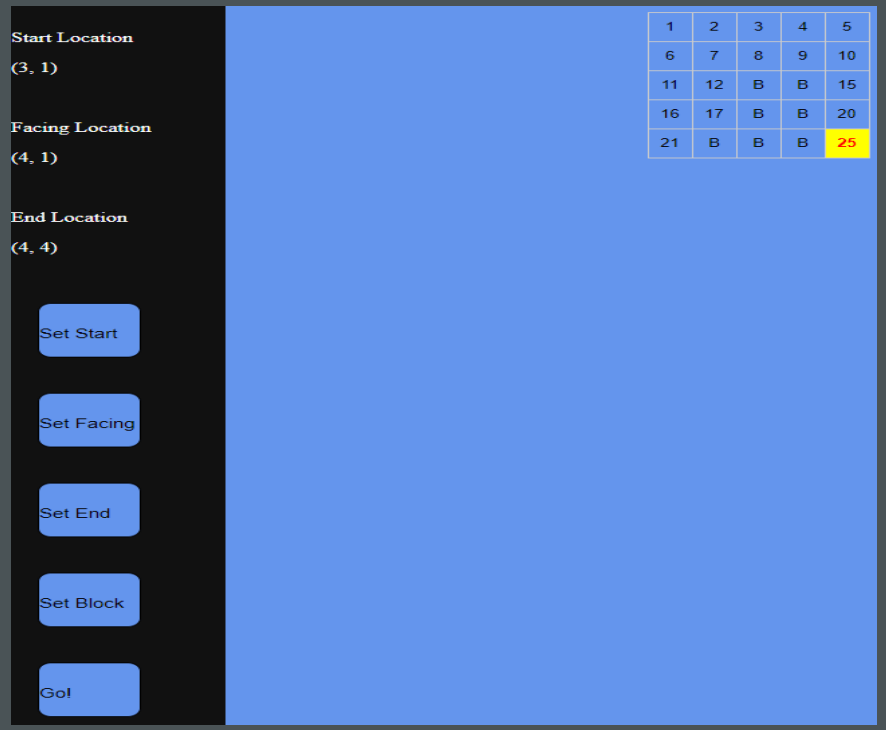


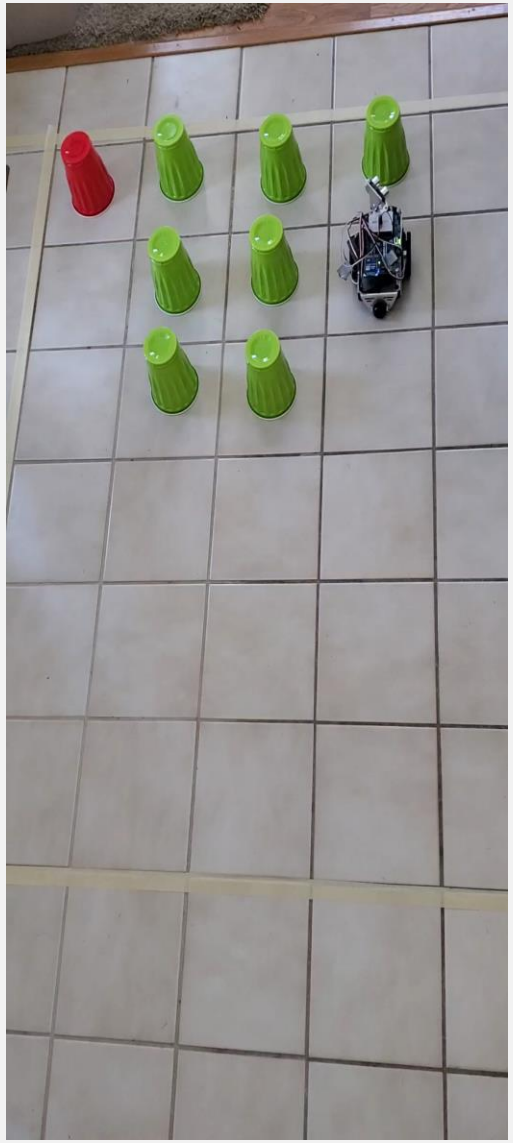
DEMONSTRATION (AUTOMATION)





DEMONSTRATION (AUTOMATION)

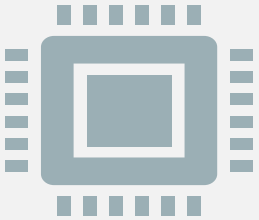




EXCEPTIONS

- "The bot should … know the location of the contents, probably using a modified geo-mesh to map the lab layout."
  - Technically, Eddie doesn't need to know where things are – its controller does it for him.
- "The lab robot should avoid obstacles that appear unexpectedly."
  - Unfortunately, I ran out of time regarding this (for automation).
    - A suggestion would be to use the PING))) sensor, have it move back to the last cell it was at, and then recalculate the route with the unexpected block as an added pre-existing one (this is easier said than done, however).
- "Multiple users could give the robot instructions handling racing issues if necessary."
  - Unfortunately, I ran out of time regarding this as well (not sure how I'd handle this with Eddie).
- "The bot needs … personality"
  - Allow the user to upload a file and make Eddie dance!

LEARNING TECHNIQUES



Trial and Error

Manual drive (calculating distances, using the PING))) sensor)

Automation (building the classes and making the bot go where I want it to go)

- Always be learning from your mistakes.

Patience

Takes a while to connect with the module, load the webpage, and start the C code up.

- Break up your work into bite-sized chunks and keep a steady pace.

RESOURCES USED

Caitlin and David Deuchert

- Helped with refreshing my JavaScript and gave much-needed perspective.

Nathan Labott's previous code and documentation

- Made manual drive quick and painless; helped me understand how the webpage and the bot talk with one another.

Pre-existing Parallax documentation

- How the bot measures distance, how to do specific turns, how to use SimpleIDE.

Previous CS Courses

- Programming Languages helped by teaching me a basic level of HTML and JavaScript.
- Operating Systems helped by teaching me a basic level of C.

Stack Exchange (of course)

EXTENSIONS

Merge manual and automatic drive into one webpage.

Utilize 8-directional movement instead of 4-directional movement.

Find a way for the bot to know *where it is*, rather than be only told *where to go*.

Create a *truly* optimal path (my optimal path doesn't consider a bot, and thus does some turns where it wouldn't be necessary – my path is optimal inasmuch as it doesn't naturally zigzag).