Auto Nav Bot Project Document

By: Zeeshan Bombaywala
January 17, 2023
TEJ4MU

Idea:

Create a self navigation robot that can collect data about the environment it's in using sensors.

Physical Materials Used:

1. Arduino
2. MQ-2 Gas sensor
3. 9V Battery
4. 12V Battery
5. Robot/Car Kit (Anything with 2 separate motors)
6. Castor Wheel
7. DHT11 Temperature & Humidity
8. HC-SR04 Ultrasonic Distance Sensor Module
9. L298N motor driver
10. Servo motor
11. 16x2 LCD
12. LCD Display I2C Adapter
13. 9V Battery Button Clip
14. Box to contain electronics

Skills Needed:

1. C++ programming on Arduino IDE
2. Circuit Design
3. App creation in MIT app inventor
4. Research Skills

Initial Process:

Brainstorming and Planning:

Assembly and Testing:

- Assemble the robot kit as per the provided instructions.

Bluetooth Module Integration:

- Connect the HC-05 Bluetooth module to an app for remote control.
- Reference: Using an HC-05 Bluetooth Module for Arduino Medium Article

Navigation Robot Development:

- Develop a navigation system using Arduino.
- Incorporate sensors for gas, temperature, and humidity measurements.
- Integrate an LCD display for real-time information.

Air Quality Monitoring System:

- Create a DIY Arduino air quality monitor using Tinkercad.
- Experiment with various sensors for accurate air quality measurements.
- Research the practical applications of the selected sensors.

Purpose and Application:

- Define the purpose of the robot: Exploration of spaces with potential hazards or inconvenience for humans.

Iterations and Improvements after a few days:

- Instead of using a motor shield, I'll be using motor drivers and connecting them directly to the Arduino
- LCD display: Displays the direction the robot will be turning.
- Improve physical aesthetic and layout design by integrating the LCD display into the box created.

Project Notes and New Brainstorming Ideas:

- Check-In Date: Nov 13th:
  - Review progress and assess any challenges faced during the initial stages.
- Development:
  - Create a breadboard based prototype from the tutorial video here.
  - Develop the first prototype on a breadboard before moving to the PCB.
- Sensor Functionality:
  - Code the functionality of one sensor before moving to the next.

- Container Creation:
    - Utilize a 3D printer or laser cutter to create a container for the entire system.
    - Aim for a final product appearance, resembling a professional project rather than a prototype. Hiding the arduino and its wiring and integrating a place for the LCD display to go.
    - Create a container for the L298N that also functions as the attachment to the car body.
- Greenhouse Application:
    - I considered the sensor's application in a greenhouse environment.
    - Assess how the project aligns with greenhouse monitoring and control needs.
- Final Application Idea (Hazardous Workspace):
    - In envisioning the final application, I explored the potential application of my project in a hazardous workspace environment. The integration of gas sensors, such as the MQ-2, can be valuable in monitoring and ensuring the safety of industrial or laboratory spaces where the presence of hazardous gasses is a concern before sending in human workers, thus the auto navigation feature.
    - The project could be adapted to detect and measure various types of gasses relevant to the specific workspace, providing real-time data on air quality. (Carbon Monoxide, Liquefied petroleum gas, Smoke)

Documenting Process and Learnings

Research on Gas sensors:

Process and research done on different sensors:

- SGP30 Gas Sensor: This sensor from Sensirion measures indoor air quality and provides accurate readings of total volatile organic compounds (TVOCs) and equivalent carbon dioxide (eCO2) concentrations. It communicates with Arduino through I2C and provides calibrated readings.

- CCS811 Air Quality Sensor: Another option from AMS is the CCS811, which measures eCO2 and total volatile organic compounds (TVOCs). It features a digital I2C interface and provides calibrated readings.

- MiCS5524 Carbon Monoxide, Methane, and LPG Gas Sensor: This sensor provides an analog output, but it also has a digital interface for easy integration with Arduino. It measures carbon monoxide, methane, and liquefied petroleum gas (LPG).

- Alphasense CO-B4 Carbon Monoxide Sensor: Alphasense produces high-quality gas sensors for industrial applications. The CO-B4 sensor provides precise measurements of carbon monoxide and has a digital output.

After doing research on the different types and use cases of sensors I settled on the Mq-2 gas sensor as I learned that regardless of the sensor calibration will be necessary and the MQ-2 has a more simple process in comparison to the others.

Sensitivity Characteristics:

Understanding the MQ-2 gas sensor's sensitivity involves interpreting information found in its datasheet. The sensitivity graph illustrates how the sensor's resistance changes with varying gas concentrations. During my project, I studied these graphs and observed the relationship between resistance (Rs/Ro) and gas concentration for LPG, CO, and Smoke.

For our specific application, I extracted meaningful parameters—like the slope, intercept, and power—from the sensitivity graphs. These parameters enable the creation of gas curves, forming the basis for accurate gas concentration calculations.

Gas Percentage Calculation:

The gas percentage calculation formula, derived from the sensitivity equation, became a key aspect of my project. Using the normalized sensor resistance, I could accurately determine gas concentrations. The parameters obtained from the gas sensitivity curve fitting process—like *pcurve*[0], *pcurve*[1], and *pcurve*[2]—played a crucial role in this calculation.

This practical application of the sensitivity equation allowed me to translate sensor resistance data into meaningful gas percentage readings for LPG, CO, and Smoke in real-time.

Gas Sensor Calibration In My Code:

In the presented Arduino code for gas sensing using the MQ2 gas sensor, a calibration process is implemented to enhance the accuracy of gas concentration measurements. Calibration is imperative to establish a baseline resistance value in clean air, allowing for the compensation of sensor readings and ensuring reliability in gas concentration assessments.

MQResistanceCalculation Function:

```
float MQResistanceCalculation(int raw_adc) {
  return ((float)RL_VALUE * (1023 - raw_adc) / raw_adc);
}
```

The MQResistanceCalculation function is crucial in determining the resistance of the MQ sensor. Utilizing the raw analog-to-digital converter (ADC) value (raw_adc) from the MQ sensor, the function employs the formula (RL_VALUE * (1023 - raw_adc)) / raw_adc. Here, RL_VALUE represents the load resistance of the sensor. The result is the calculated resistance of the MQ sensor, a fundamental parameter for subsequent calibration steps.

MQCalibration Function:

```
float MQCalibration(int mq_pin) {
```

```
  int i;
  float val = 0;

  for (i = 0; i < CALIBRATION_SAMPLE_TIMES; i++) {
    val +=
MQResistanceCalculation(analogRead(mq_pin));
    delay(CALIBRATION_SAMPLE_INTERVAL);
  }
  val = val / CALIBRATION_SAMPLE_TIMES;

  val = val / RO_CLEAN_AIR_FACTOR;

  return val;
}
```

The MQCalibration function manages the calibration procedure by systematically collecting multiple resistance readings from the MQ sensor. Averaging these readings mitigates potential noise and yields a stable baseline resistance value (val). Subsequently, this baseline value undergoes normalization by dividing it by the clean air factor (RO_CLEAN_AIR_FACTOR). The clean air factor, a predetermined constant (9.83), serves as a critical calibration parameter specific to the MQ sensor model. The calibrated resistance value obtained from this process forms the basis for precise gas concentration calculations.

HC-05 Bluetooth Module:

Setup:

To set up an HC-05 Bluetooth module using AT commands and change the baud rate to a value faster than 9600:

Prepare Hardware:

- I connected my HC-05 module to the Arduino using a USB-to-Arduino cable for code uploading.
- I ensured that the HC-05 was adequately powered, and I connected its TX/RX pins to the RX/TX pins of the Arduino.

Enter AT Mode:

- Since my HC-05 module doesn't have a physical button, I followed an alternative method to enter AT mode. Before powering up the module, I ensured that the "Enable" pin was LOW. After powering it up, I pulled the "Enable" pin HIGH, creating the condition needed to enter AT mode.

Open Serial Terminal:

- I used a serial terminal software like Arduino IDE's Serial Monitor to first test it to send AT commands, however it wasn't working, so then I tried with PuTTY terminal software that allowed me to send AT commands to the module.

Set Baud Rate to 38400:

- Sent the following AT command to set the baud rate to 38400 bps:

```
AT+UART=38400,0,0
```

- Wait for the module to respond with "OK."

Change Serial Terminal Baud Rate:

- I changed the baud rate of your serial terminal to 38400 bps. This is necessary to communicate with the HC-05 at the new baud rate.

Restart the HC-05 Module:

- Power off the HC-05 module and then power it on again.

Verify New Baud Rate:

- After restarting, the HC-05 module should now be operating at the new baud rate (38400 bps). I can verify this by entering AT mode again and checking the baud rate:

AT+UART// returns baud rate

- The module will respond with the current UART configuration.

Exit AT Mode:

- Once everything is working correctly, exit AT mode. I had to pull the "Enable" pin LOW

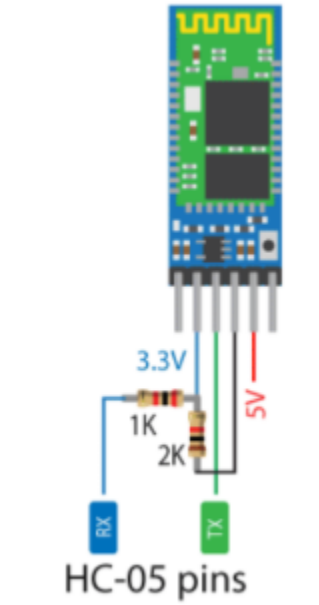
Update Baud Rate in Your Application:

- Since I'm using the HC-05 module with an arduino, I had to update the baud rate in the code to match the new baud rate (38400 bps).

Test Communication:

- Tested the communication between the microcontroller and the HC-05 module at the new baud rate to ensure proper functionality.

Voltage divider:

	The HC-05 Bluetooth module typically operates at 3.3V, while the Arduino uses 5V logic levels. To ensure compatibility and prevent potential damage to the HC-05 module, I had to use a voltage divider for the RX (receive) pin of the HC-05. An issue I encountered was with the first HC-05 I had used. The first Bluetooth Module I used seemed to have stopped working. This can be possibly due to someone prior not using the voltage divider. I know it was never working as the red LED never turned on. After testing another HC-05 with the same circuit configuration I had, I knew there was an issue with the previous module.

Calculations to find what resistor values to use for the voltage divider

A voltage divider is needed to step down the 5V signal to a safe level for the HC-05.

The voltage divider is made using two resistors. The formula for the output voltage ($\mathbf{V}_{out}$) in a voltage divider is:

$\mathbf{V}_{out} = V_{in} \times (R2/ R1+R2)$

Where V_{in} is the input voltage (5V from the arduino), R1 is the resistor connected to V_{in}, and R2 is the resistor connected to ground.

To step down 5V to 3.3V, I need to choose R1, and R2 such that:

3.3V = 5V x (R2 /R1+ R2)

A common choice is to use a 1k ohm resistor for R1 and a 2k ohm resistor for R2. Next is to check it with Ohm's law:

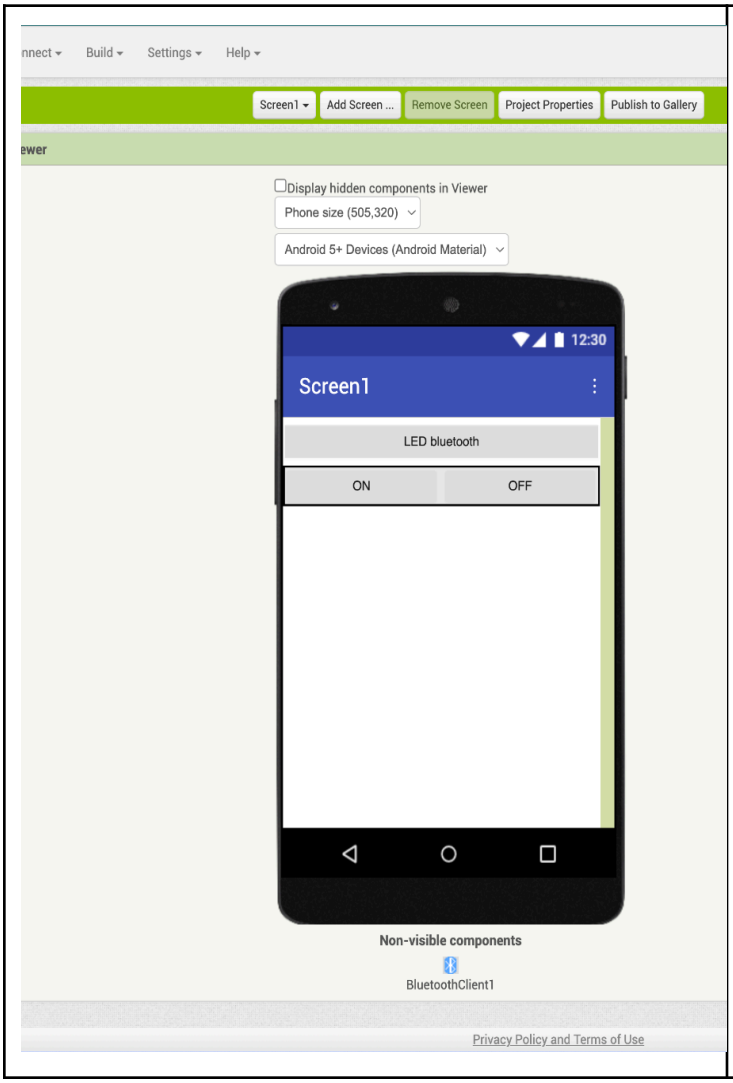
$\mathbf{V}_{out}$ = 5V X (2000 ohms/1000 ohms + 2000 ohms)

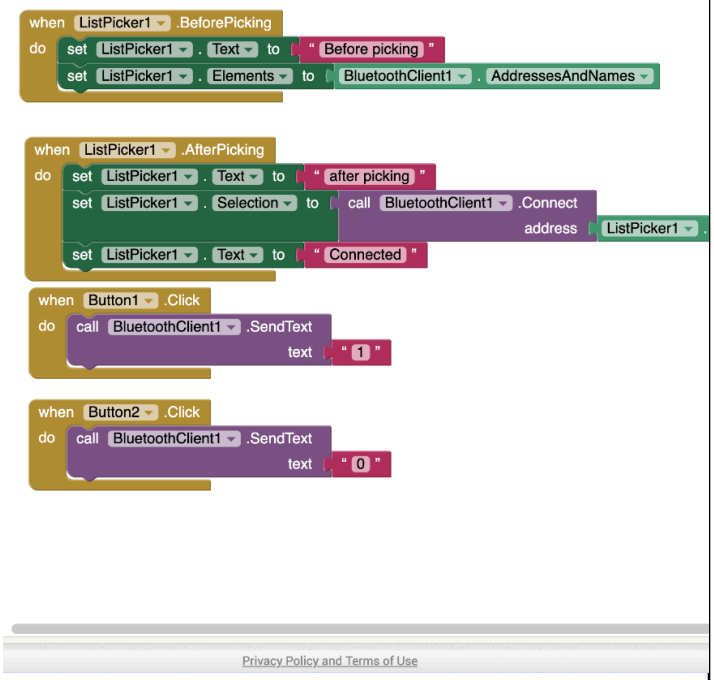
= 3.33333V

Using a 1k resistor for R1, and a 2k resistor for R2 in the voltage divider, the output voltage $\mathbf{V}_{out}$ is approximately 3.33V. This is close enough to the required 3.3V for the HC-05's RX pin, making these resistor values a suitable choice for safely interfacing a 5V microcontroller with the HC-05 Bluetooth module.

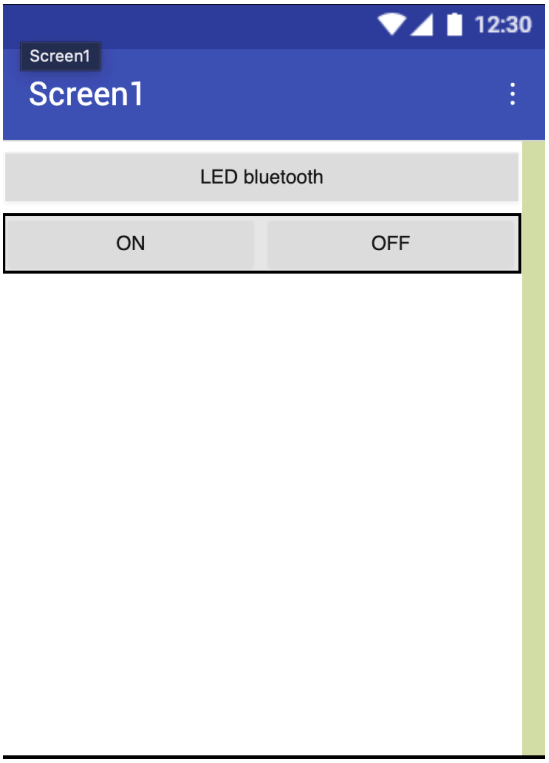
Challenges With HC-05:

The application created for the bluetooth module is not displaying the option to connect to the HC-05. Due to this issue I had to remove the use of the bluetooth module and move on with the project due to the nearing deadline.

	Display in MIT app inventor preview screen



Code created



Issue faced in the application: When pressing connect to pair to the HC-05 no bluetooth devices appear. In the gif to the left you can see when clicking the LED Bluetooth button a screen appears with no devices displaying

Expected Behaviour: When you press connect all bluetooth devices connected to your device should appear. You then click on the HC-05.

Once paired it will turn On and Off a LED. This same function would be used to turn the auto nav robot from the IDLE state to moving.

L298N Motor Driver:

Info on it:

1. H-Bridge Configuration:
It consists of four switches arranged in the shape of an "H." The switches can be either transistors (typically MOSFETs or BJTs) or electronic switches like relays.

Here's a detailed explanation of how an H-bridge works:

Basic Configuration:

- An H-bridge has four switches labeled as Q1, Q2, Q3, and Q4.
- Q1 and Q2 are on one side, and Q3 and Q4 are on the other, forming the legs of the "H."

Motor Connections:

- The motor is connected between the common points of Q2 and Q3 (the center of the "H").
- The power supply voltage is applied across the terminals of the motor.

Switching States:

- By controlling the states (ON or OFF) of the switches, the H-bridge can change the direction of current flow through the motor.
- When Q1 and Q4 are ON, and Q2 and Q3 are OFF, current flows from the power supply, through Q1, the motor, and then through Q4 to ground. This configuration causes the motor to rotate in one direction (let's call it forward).
- When Q2 and Q3 are ON, and Q1 and Q4 are OFF, the current flows in the opposite direction, causing the motor to rotate in the opposite direction (reverse).

Braking and Freewheeling:

- To quickly stop the motor, both sides of the H-bridge can be turned ON simultaneously. This creates a short circuit across the motor terminals, inducing a braking effect.
- To allow the motor to freely spin (freewheeling), both sides of the H-bridge can be turned OFF, letting the motor coast.

PWM for Speed Control:

- Pulse Width Modulation (PWM) is often used to control the speed of the motor.
- By rapidly switching the H-bridge's switches ON and OFF, the effective voltage applied to the motor is varied, controlling its speed.

Protection Diodes:

- Freewheeling diodes (also known as flyback diodes or catch diodes) are connected in parallel with each switch to protect against voltage spikes generated by the motor's inductive nature when the switches are turned off.

Motor Controls:

Motor A Connections:

- **enA** (Pin 11): This is the enable pin for Motor A. Connecting this pin to a PWM (Pulse Width Modulation) capable pin (like pin 11) allows you to control the speed of Motor A. The higher the PWM signal, the faster the motor rotates.
- **in1** (Pin 12): This is one of the input pins for Motor A. It determines the direction of rotation for Motor A. High (5V) and Low (GND) on this pin will make the motor rotate in one direction, and reversing these states will make it rotate in the opposite direction.
- **in2** (Pin 10): The second input pin for Motor A. Its state in combination with **in1** controls the direction of rotation for Motor A.

Motor B Connections:

- **enB** (Pin 3): Similar to enA, this is the enable pin for Motor B, and it's connected to a PWM capable pin (pin 3). By adjusting the PWM signal on this pin, you control the speed of Motor B.
- **in3** (Pin 9): This is one of the input pins for Motor B, determining its rotation direction based on its state (High or Low).
- **in4** (Pin 5): The second input pin for Motor B, working in conjunction with **in3** to control the rotation direction.

Direction Control:

- The pairs of input pins (in1 and in2 for Motor A, and in3 and in4 for Motor B) control the direction of rotation. By manipulating the states of these pins, you can make the motors rotate forward, backward, or stop.

Example:

- To make Motor A move forward: Set **in1** to High, **in2** to Low.
- To make Motor A move backward: Set **in1** to Low, **in2** to High.
- To stop Motor A: Set both **in1** and **in2** to either High or Low (depending on the desired braking method).
- Similar logic applies to Motor B using pins **in3** and **in4**.

Challenge I faced:

- The motor driver when powered by an external power source required different wiring configurations to work. I didn't know this and was facing issues as the motor driver wasn't working with the arduino. After doing more research I was able to find the different ways to power the motor driver. This website is where I found the solution to my problem.
  https://www.hackster.io/ryanchan/how-to-use-the-l298n-motor-driver-b124c5

- I also faced an issue within the code. The project tutorial is controlling the motor motor driver using a library. I never fully understood this, so I was having trouble figuring out why my project was not working. I decided to instead control each individual pin in a function. This resolved the issue and it was easier for me to understand and create myself.
-

```
//Function to turn right
void right() {
 digitalWrite(in1, HIGH);
 digitalWrite(in2, LOW);
 analogWrite(enA, 200);

 digitalWrite(in3, LOW);
 digitalWrite(in4, HIGH);
 analogWrite(enB, 200);

 delay(200);
 stop();
}
```

All pin connections:

Keeping this allowed me to plan for when rewiring the entire project to put the project in my finished enclosure of a box, and to keep track of how many pins are used. As well as where all the pins are so that it is easier to implement into the code.

Arduino to Motors and Motor Driver (L298N):

- Motor A:
  - enA (ENA) → Arduino pin 11
  - in1 (N1) → Arduino pin 12
  - in2 (N2) → Arduino pin 10
- Motor B:
  - enB (ENB) → Arduino pin 3
  - in3 (N3) → Arduino pin 9
  - in4 (N4) → Arduino pin 5

Arduino to Servo Motor:

- servoMainPin → Arduino pin 6

Ultrasonic Sensor (HC-SR04):

- trigPin → Arduino pin 7
- echoPin → Arduino pin 8

MQ-2 Gas Sensor:

- A0 → Analog pin A1

DHT11 Sensor:

- DHTPIN → Analog pin A0

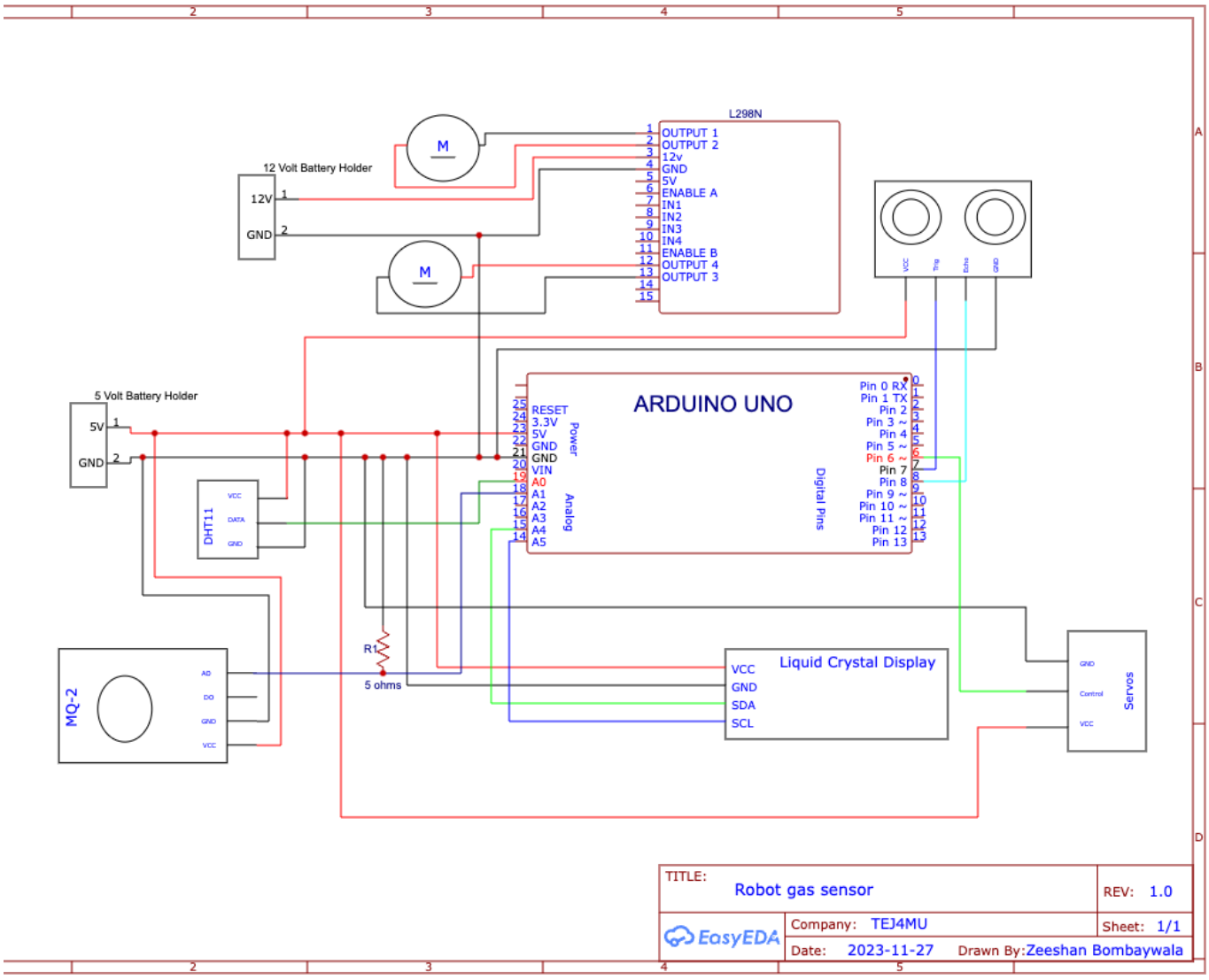
LCD (assuming I2C module is used):

- SDA → A4 on Arduino (for data)
- SCL → A5 on Arduino (for clock)

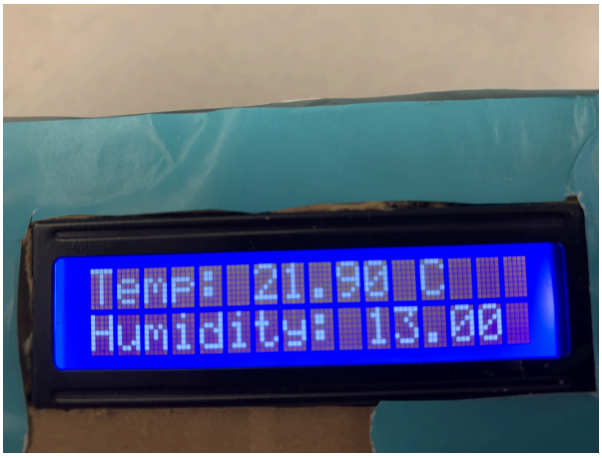
Power Supply:

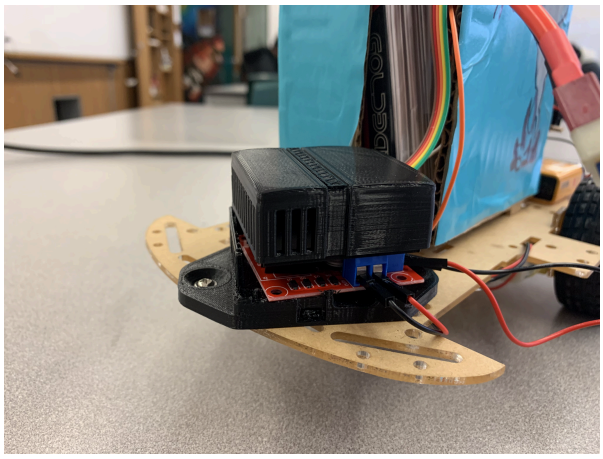
- Provide a suitable power supply for the motors and the motor driver. Using a 12 volt battery. Ensure the common ground between Arduino, motors, and sensors.
- A 9 volt battery to power the arduino as there was significant voltage drop while using a buck converter. Consequently, there was a lack of power resulting in the servos motor unable to turn.

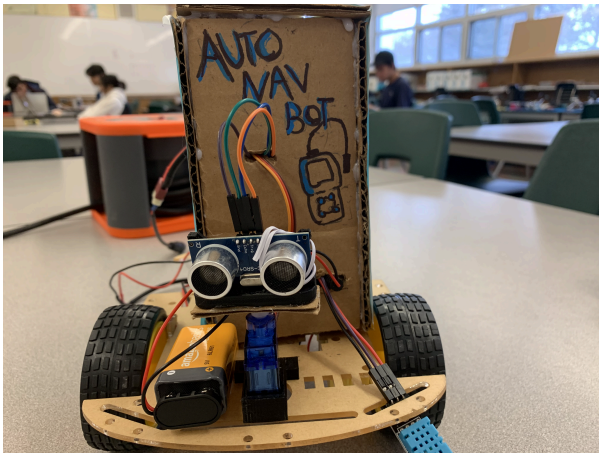
Schematic diagram:

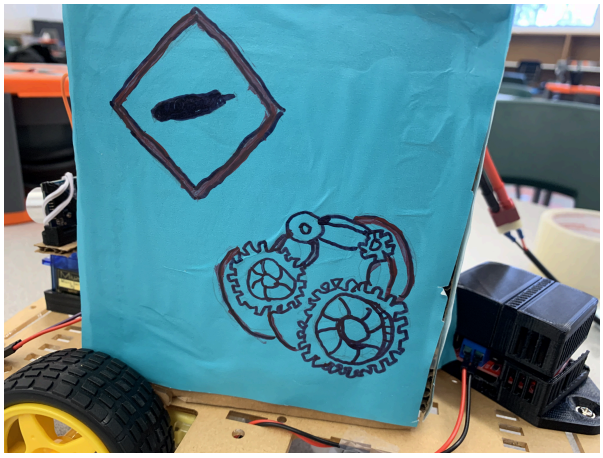


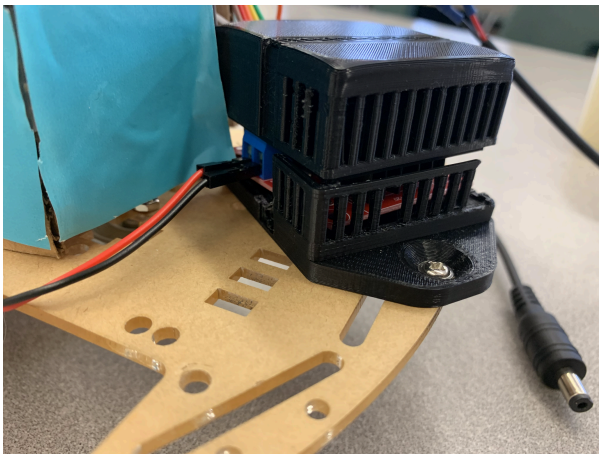
Finished Project Pictures:











Project Resources:

Remote-Controlled Car Setup

- Instructables - Remote-controlled car using Arduino and TV remote

Automated Navigation Robot with Gas, Temperature, and Humidity Sensors

- Instructables - Automated Navigation Robot With Gas, Temperature, Humidity Sensors

Grove Air Quality Sensor v1.3

- Seeed Studio - Grove Air Quality Sensor v1.3

MQ135 Air Quality Sensor

- Elprocus - MQ135 Air Quality Sensor

DHT 11 Temperature and Humidity Sensor

- Instructables - How to Interface DHT 11 Temperature and Humidity Sensor

Bluetooth HC05:

- Getting Started With HC05 Bluetooth Module & Arduino [Tutorial]
- How Do You Change Baud Rate
- Smartphone LED Controller Using Bluetooth With Own Application
- Arduino and HC-05 Bluetooth Module Complete Tutorial
- Add a Voltage divider in the HC-05 Arduino circuit
- https://github.com/binaryupdates/arudino-hc05-bluetooth/blob/master/bluetooth_arduino.ino
- Arduino Bluetooth Control | HC-05 with Android App | Led Control

Gas sensor:

- MQ-2 Semiconductor Sensor for Combustible Gas
- Arduino And MQ2 Gas Sensor
- How does work MQ2 sensor - MQ2 sensor with Arduino UNO [Code and circuit diagram]
- How to use MQ2 Gas Sensor with Arduino and Serial Monitor
- https://github.com/harshkzz/Arduino-with-MQ2-Sensor-and-Serial-Monitor blob/main/MQ2_SERIALMONITOR/MQ2_SERIALMONITOR.ino
- Calibration MQ2 Sandbox
- MQ2 Gas Sensor Tutorial: Measuring Gas Concentration with PictoBlox.

Motor Driver:

- L298n Motor driver Arduino | Motors | Motor Driver | L298n
- How to use the L298N Motor Driver