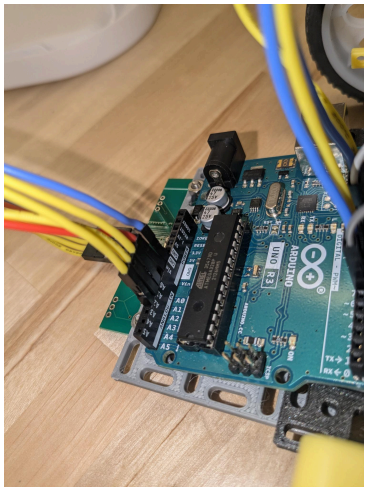
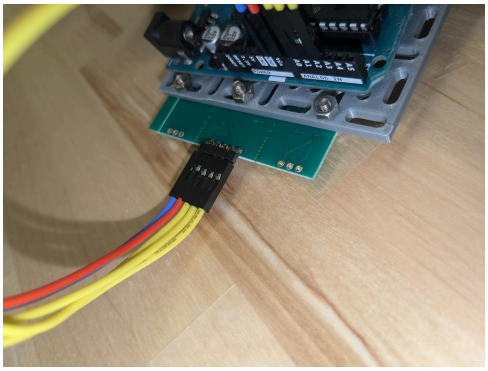
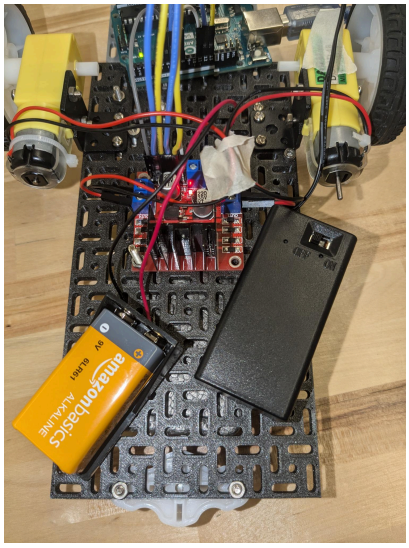
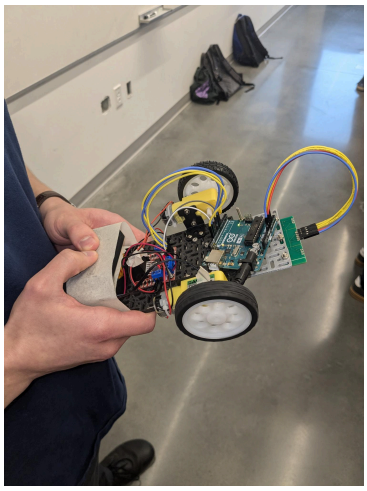
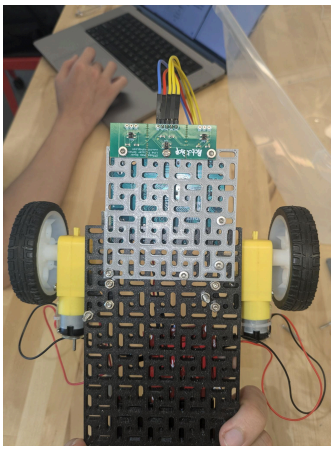
Photos

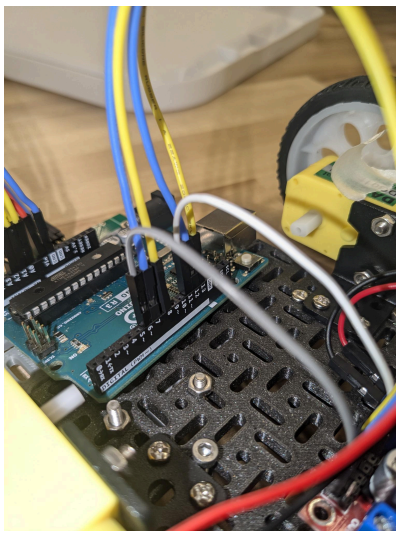


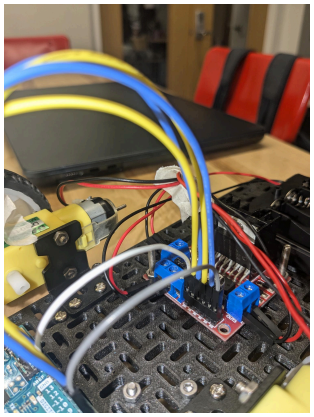


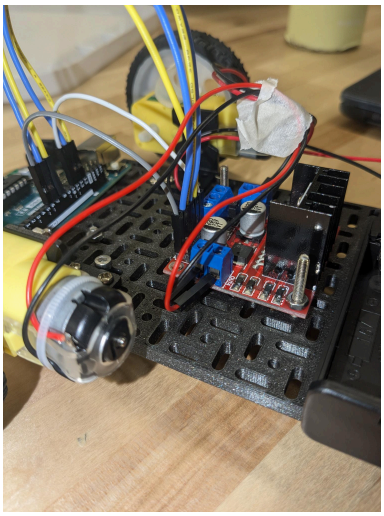




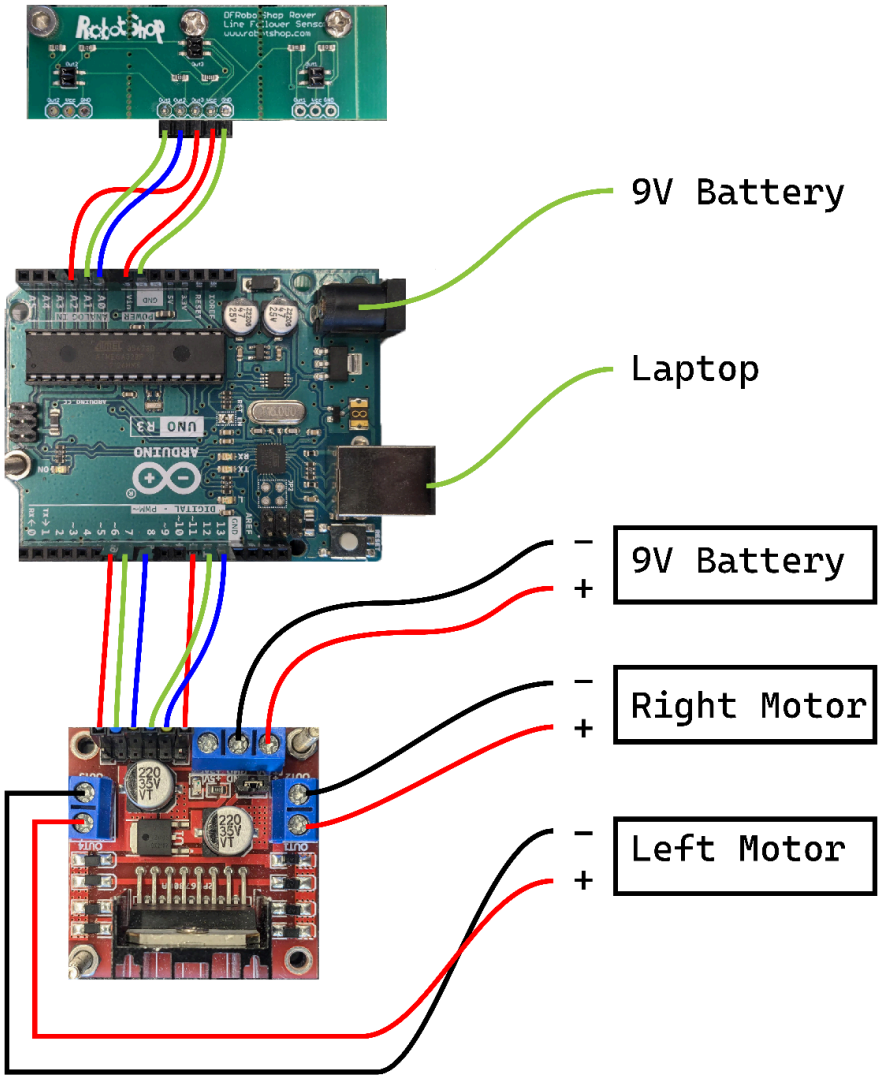








Layout



Sensor to arduino

The sensor outputs (out1,out2,out3) are connected to the analog input pins on the Arduino(A0,A1,A2)

The Vcc pin of the sensor is connected to the Vin pin of the Arduino. Vin is the power input for Arduino; it accepts the 9 voltage from the power source.

The Grd(ground) pin of the sensor is connected to the GND pin of the Arduino, This completes the electrical circuit.

Arduino to Motor driver

Arduino Pin13 to IN1 and Pin12 to IN2 on the motor driver. These two pin sets the direction of the motor or enable/disable the motor on the left Arduino Pin 11~(Pulse width modulation). This allows for variable speed control instead of just on/off

The same case goes for Pin8 to IN3 and Pin7 to IN4 to control the direction or enable/disable the motor on the right Motor and Pin 6~ (Pulse width modulation) allows for variable speed control instead of just on/off

Motor driver to motor

Out 1,Out 2,Out 3,Out 4 are the outputs from the motor driver and are connected to the two motors, these connections let the motor driver regulate power to the motors, controlling their direction and speed.

Battery to motor driver

The 9V battery is connected to the +12V input on the motor driver. WIth +(positive voltage) connected to+12v on the motor driver and -(GND) to GND on the motor driver

Parts List

1. Line follower sensor (x1)
2. Arduino Uno (x1)
3. L298N motor driver (x1)
4. Ball caster (x1)
5. AA battery (x2)
6. AA battery holder (x2)
7. Plastic chassis pieces (x2)
8. DG01D 48:1 motor (x2)
9. Wheel (x2)
10. M3 threaded L bracket (x2)
11. Male to male jumper wire (x5)
12. Male to female jumper wire (x6)
13. M3 nuts (x20)
14. M3 screws (x20)
15. Masking tape

How it works & design decisions

The chassis is constructed of an 18x9cm flat plastic plate with a grid of holes, allowing for m3 screws to pass through it. Most parts are fastened to the plate using 2 or 3 screws in order to prevent wobbling. The two motors and wheels are attached near the front of the robot via the L brackets, and the ball caster is attached near the back. This causes the robot to be angled such that the front approaches the ground. A second smaller flat plate is attached below, extending out in front of the robot. The line follower sensor is attached below the smaller plate. At this point the position of the sensor was adjusted until it was within 3 mm of the ground, because the documentation on the sensor claims more accurate readings at that distance. It's important that the screws used to fasten the sensor have a low profile to prevent them from scraping the ground. The H bridge is fastened to the middle of the large plate because there's space available, and two 9 volt battery holders are fastened near the back by masking tape. Masking tape is easily removed for battery replacement. Having an on/off switch on 1 of the battery holders helps to control the robot when something goes wrong during testing. The arduino is fastened at 1 corner by a single screw. This allows it to rotate, making the usb port more easily accessible.

The motors use a 9 volt battery instead of the suggested AA batteries because of limited availability of charged AA batteries. The reduced current means that the robot can only power 1 wheel at once. To overcome this issue, the robot will rapidly switch between the two motors to imitate moving in a straight line. When the sensors don't detect the line, we assume that the line must be within a few centimeters of the sensors. In this situation, one motor turns forward while the other turns backward and the robot rotates around its axis. The direction of rotation is determined by which sensor most recently detected black.

Arduino sketch with comments

```
/* Motor Control */
int enA = 11; //Enable A, connected to pin 11, analogWrite() PWM
int enB = 6; //Enable B, connected to PWM pin 6, analogWrite() PWM
int A_backward = 12; //Digital pin 12, If HIGH && A_forward LOW, left wheel spins backward
int A_forward = 13; //Digital pin 13, If HIGH && A_backward LOW, left wheel spins forward
int B_backward = 7; //Digital pin 7, if HIGH && B_forward LOW, right wheel spins backward
int B_forward = 8; //Digital pin 8, if HIGH && B_backward LOW, right wheel spins forward

//These variables are used for flipping which motor gets powered, due to only one of them being able to be powered at a time
int counter = 0;
int motorSelect = 0;

/* Infrared Sensor */
int rsensor = 1; // Left Sensor on Analog Pin 1
int lsensor = 0; // Right Sensor on Analog Pin 0
int msensor = 2; // Middle Sensor on Analog Pin 2
const int sensor_threshold = 418; // The sensor outputs around 800 over black and around 35 over white. 418 sits in the middle of this range for ample detection padding
int last_sensor = 0; //Initial placeholder value before either right or left sensors have been activated.

long time = millis(); //Initiate time counter


void setup() {
  Serial.begin(9600); //Set baud rate
  motorOff(); //Initiate all control pins to low
}

void loop(){
  if(millis() - time >= 15){
    counter += 1;
    motorSelect = counter % 2; //If at least 15 milliseconds have passed, toggle motorSelect between 0 and 1
    motorSwitch(210, 185, motorSelect); //Either right or left motor is powered; the speeds 210 and 185 were found to be the most optimal for staying on the line
    time = millis(); //Reset timer
  }
  
  if (analogRead(lsensor) < sensor_threshold && analogRead(msensor) > sensor_threshold && analogRead(rsensor) < sensor_threshold)
  {
    goForward(); //If middle sensor is on black while the two side sensors are on white, drive forward
  }
  else if(analogRead(lsensor) > sensor_threshold){
    turnLeft(); //Otherwise, if the left sensor is on black, turn left
    last_sensor = lsensor; //Track that the last seen sensor was the left sensor
  }
  else if(analogRead(rsensor) > sensor_threshold){
    turnRight(); //Otherwise, if the right sensor is on black, turn right
    last_sensor = rsensor; //Track that the last seen sensor was the right sensor
  }
  else{ //No sensors are on black: spin in the direction last sensed
    if(last_sensor == rsensor){
      spinLeft();
    }
    else{
      spinRight(); //Default behavior is to spin in a clockwise direction
      
    }
  }
}


/* Function which selects which motor gets power. Takes three parameters.
speedA: The voltage to be supplied to motor A. Expected values are integers between 0 and 255, inclusive.
speedB: The voltage to be supplied to motor B. Expected values are integers between 0 and 255, inclusive.
motorSelect: Which motor to be powered. Expected values are 0 (motor A) or 1 (motor B).
*/
void motorSwitch(int speedA, int speedB, int motorSelect){
  if(motorSelect == 0){
    analogWrite(enB, 0); //Turn opposite motor off before turning selected motor on
    analogWrite(enA, speedA);
  }
  if(motorSelect == 1){
    analogWrite(enA, 0);
    analogWrite(enB, speedB);
  }
}

//Sets all control pins to LOW; no movement.
void motorOff(){
  digitalWrite(A_forward, LOW);
  digitalWrite(A_backward, LOW);
  digitalWrite(B_backward, LOW);
  digitalWrite(B_forward, LOW);
}

//Sets forward control pins to HIGH, backward control pins to LOW; both wheels spin forwards, car moves forward(ish).
void goForward(){
  digitalWrite(A_forward, HIGH);
  digitalWrite(A_backward, LOW);
  digitalWrite(B_forward, HIGH);
  digitalWrite(B_backward, LOW);
}

//Sets right wheel's forward control pin to HIGH, all others LOW; right wheel spins forward, car turns leftward.
void turnLeft(){
  digitalWrite(A_forward, HIGH);
  digitalWrite(A_backward, LOW);
  digitalWrite(B_forward, LOW);
  digitalWrite(B_backward, LOW);
}

//Sets left wheel's forward control pin to HIGH, all others LOW; left wheel spins forward, car turns rightward.
void turnRight(){
  digitalWrite(A_forward, LOW);
  digitalWrite(A_backward, LOW);
  digitalWrite(B_forward, HIGH);
  digitalWrite(B_backward, LOW);
}

//Sets left wheel's forward control pin to HIGH, right wheel's backward control pin to HIGH, all others LOW; car spins in a clockwise direction.
void spinRight(){
  digitalWrite(A_forward, LOW);
  digitalWrite(A_backward, HIGH);
  digitalWrite(B_forward, HIGH);
  digitalWrite(B_backward, LOW);
}

//Sets right wheel's forward control pin to HIGH, left wheel's backward control pin to HIGH, all others LOW; car spins in a counter-clockwise direction.
void spinLeft(){
  digitalWrite(A_forward, HIGH);
  digitalWrite(A_backward, LOW);
  digitalWrite(B_forward, LOW);
  digitalWrite(B_backward, HIGH);
}
```

```
/* Motor Control */

int enA = 11; //Enable A, connected to pin 11, analogWrite() PWM

int enB = 6; //Enable B, connected to PWM pin 6, analogWrite()
PWM

int A_backward = 12; //Digital pin 12, If HIGH && A_forward LOW,
left wheel spins backward

int A_forward = 13; //Digital pin 13, If HIGH && A_backward LOW,
left wheel spins forward

int B_backward = 7; //Digital pin 7, if HIGH && B_forward LOW,
right wheel spins backward

int B_forward = 8; //Digital pin 8, if HIGH && B_backward LOW,
right wheel spins forward


//These variables are used for flipping which motor gets
powered, due to only one of them being able to be powered at a
time

int counter = 0;

int motorSelect = 0;


/* Infrared Sensor */

int rsensor = 1; // Left Sensor on Analog Pin 1

int lsensor = 0; // Right Sensor on Analog Pin 0

int msensor = 2; // Middle Sensor on Analog Pin 2
```

```
const int sensor_threshold = 418; // The sensor outputs around 800 over black and around 35 over white. 418 sits in the middle of this range for ample detection padding

int last_sensor = 0; //Initial placeholder value before either right or left sensors have been activated.


long time = millis(); //Initiate time counter




void setup() {

  Serial.begin(9600); //Set baud rate

  motorOff(); //Initiate all control pins to low

}


void loop(){

  if(millis() - time >= 15){

     counter += 1;

     motorSelect = counter % 2; //If at least 15 milliseconds have passed, toggle motorSelect between 0 and 1

     motorSwitch(210, 185, motorSelect); //Either right or left motor is powered; the speeds 210 and 185 were found to be the most optimal for staying on the line

     time = millis(); //Reset timer

  }
```

```
  if (analogRead(lsensor) < sensor_threshold &&
analogRead(msensor) > sensor_threshold && analogRead(rsensor) <
sensor_threshold)

  {

     goForward(); //If middle sensor is on black while the two
side sensors are on white, drive forward

  }

  else if(analogRead(lsensor) > sensor_threshold){

     turnLeft(); //Otherwise, if the left sensor is on black,
turn left

     last_sensor = lsensor; //Track that the last seen sensor
was the left sensor

  }

  else if(analogRead(rsensor) > sensor_threshold){

     turnRight(); //Otherwise, if the right sensor is on black,
turn right

     last_sensor = rsensor; //Track that the last seen sensor
was the right sensor

  }

  else{ //No sensors are on black: spin in the direction last
sensed

     if(last_sensor == rsensor){

     spinLeft();

     }
```

```
     else{
     spinRight(); //Default behavior is to spin in a clockwise
direction


     }
  }
}


/* Function which selects which motor gets power. Takes three
parameters.
speedA: The voltage to be supplied to motor A. Expected values
are integers between 0 and 255, inclusive.
speedB: The voltage to be supplied to motor B. Expected values
are integers between 0 and 255, inclusive.
motorSelect: Which motor to be powered. Expected values are 0
(motor A) or 1 (motor B).
*/
void motorSwitch(int speedA, int speedB, int motorSelect){
  if(motorSelect == 0){
     analogWrite(enB, 0); //Turn opposite motor off before
turning selected motor on
     analogWrite(enA, speedA);
  }
```

```
  if(motorSelect == 1){
     analogWrite(enA, 0);
     analogWrite(enB, speedB);
  }
}

//Sets all control pins to LOW; no movement.
void motorOff(){
  digitalWrite(A_forward, LOW);
  digitalWrite(A_backward, LOW);
  digitalWrite(B_backward, LOW);
  digitalWrite(B_forward, LOW);
}

//Sets forward control pins to HIGH, backward control pins to
LOW; both wheels spin forwards, car moves forward(ish).
void goForward(){
  digitalWrite(A_forward, HIGH);
  digitalWrite(A_backward, LOW);
  digitalWrite(B_forward, HIGH);
  digitalWrite(B_backward, LOW);
}
```

```
//Sets right wheel's forward control pin to HIGH, all others
LOW; right wheel spins forward, car turns leftward.

void turnLeft(){

  digitalWrite(A_forward, HIGH);

  digitalWrite(A_backward, LOW);

  digitalWrite(B_forward, LOW);

  digitalWrite(B_backward, LOW);

}


//Sets left wheel's forward control pin to HIGH, all others LOW;
left wheel spins forward, car turns rightward.

void turnRight(){

  digitalWrite(A_forward, LOW);

  digitalWrite(A_backward, LOW);

  digitalWrite(B_forward, HIGH);

  digitalWrite(B_backward, LOW);

}


//Sets left wheel's forward control pin to HIGH, right wheel's
backward control pin to HIGH, all others LOW; car spins in a
clockwise direction.

void spinRight(){
```

```
  digitalWrite(A_forward, LOW);
  digitalWrite(A_backward, HIGH);
  digitalWrite(B_forward, HIGH);
  digitalWrite(B_backward, LOW);
}


//Sets right wheel's forward control pin to HIGH, left wheel's backward control pin to HIGH, all others LOW; car spins in a counter-clockwise direction.
void spinLeft(){
  digitalWrite(A_forward, HIGH);
  digitalWrite(A_backward, LOW);
  digitalWrite(B_forward, LOW);
  digitalWrite(B_backward, HIGH);
}
```