Build an Arduino Self-Driving Car

Lesson 5: Detecting Obstacles

Estimated time: 2 hours

Introduction

In this lesson students will combine what they have learned about motor control and the ultrasonic sensor to make their car automatically stop when it detects an obstacle. They will design and program behaviors for their car to react to the obstacle, such as backing up or driving around it.

Learning Objectives

- Apply knowledge from previous lessons about motors and sensors
- Make a car detect an obstacle
- Program a behavior to autonomously react to the obstacle

Preparation

- Try this activity yourself before you try it with students. You can keep a working car assembled for a demonstration.
- You will need open floor space for students to test their cars and solid objects to use as obstacles. Small cardboard boxes work well. Driving the cars on tabletops is **not** recommended; they will break if they fall off the edge.
- Students will need to mount their ultrasonic sensors to the front of their cars using tape. Make sure tape does not cover the front of the sensor.

Materials

Students will need their assembled car chassis with an H-bridge circuit and ultrasonic sensor from previous lessons. No additional circuit materials are required.

Procedure

Walk your students through the following steps or have them watch the video and follow along.

Step	Timestamp	Description
1	0:00	Demonstrate a working car for students.
2	2:52	Connect the H-bridge and ultrasonic sensor circuits to the breadboard and Arduino if you have not already (see Appendix 5A).
3	8:08	If needed, review the code for the ultrasonic sensor and H-bridge individually.
4	10:12	Starting with the H-bridge code, modify the program to include code for the ultrasonic sensor and stop the motors when an obstacle is detected closer than a certain distance in front of the sensor (see Appendix 5B, but let students try this on their own first – the video contains hints if needed).
6	14:03	Run the code and test the cars by aiming them straight at an obstacle. Observe the effect of the delay time in the **`driveForward`** function. Change this time to 0, re-upload the code, and test again.

7	17:05	Program an autonomous behavior to react to an obstacle. This is an open-ended challenge. Students can decide how their car should respond to the presence of an obstacle. You can encourage them to plan their reaction based on the environment you will set up for their cars. For example, if they know the size of the obstacle, can they program their car to drive around it? Would that require driving off the "road"? Or can they program their car to do a 180 degree turn and drive in the opposite direction? Does this create a problem if you have a one-way "road" and other students' cars are driving at the same time? You can also discuss what behaviors would be realistic for a real autonomous car. For example, a real autonomous car might want to simply wait for a pedestrian or other temporary obstacle to get out of the way, but it might need to turn around if a tree has fallen across the road (note that the ultrasonic sensor in this project cannot differentiate between different types of obstacles, only whether an obstacle is present or not).

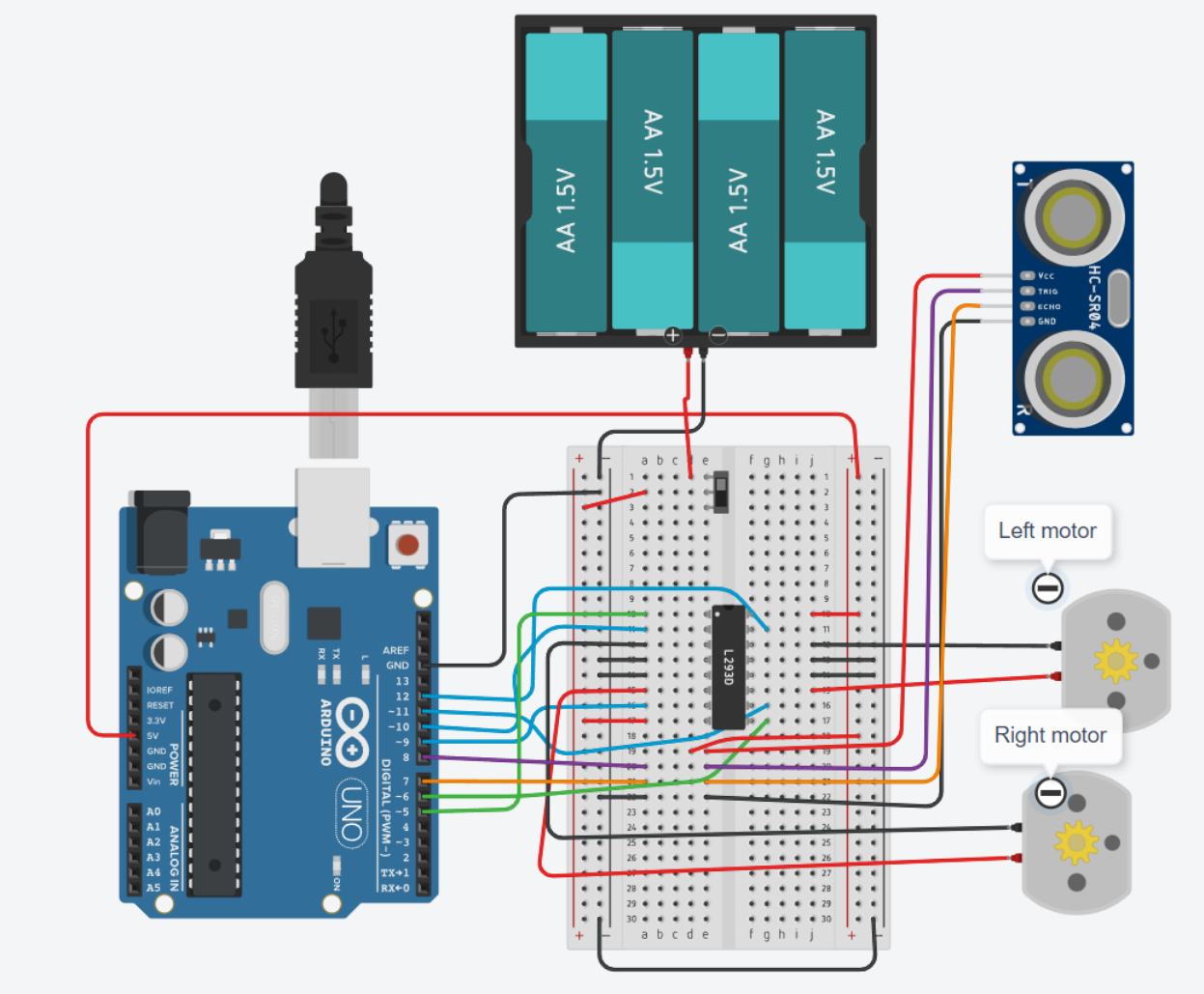
Assessment

As you walk around the class:

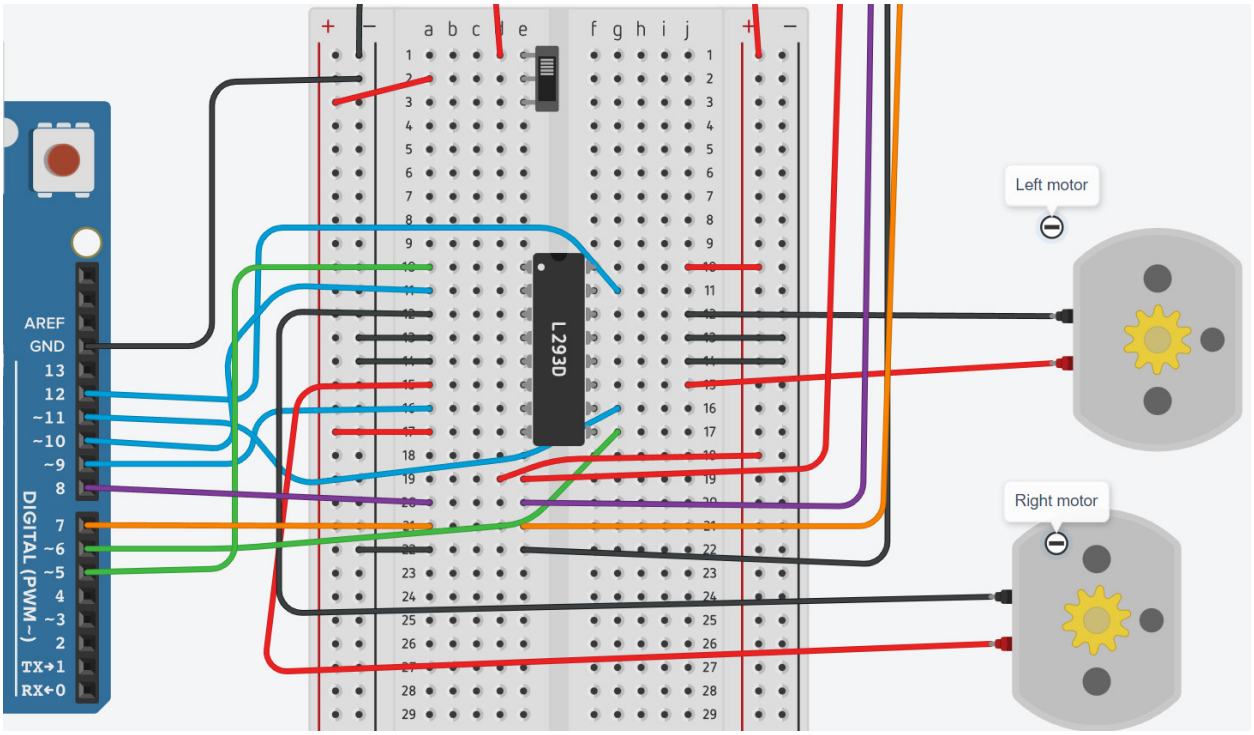
- Confirm that students can connect both the H-bridge and ultrasonic sensor on their breadboard without excessively messy wiring.
- Observe whether students can merge the code from the two programs and make their car automatically react to an obstacle in its path.
- Observe the different behaviors that students program for their cars. You can encourage groups to share and show their car's behavior to the class.

Appendix 5A: Breadboard Diagram

The following diagram shows the H-bridge and ultrasonic sensor connected to the Arduino at the same time. Note that students do not have to use the same Arduino pins shown here, but this diagram matches the example code given in this lesson.



This diagram shows a zoomed-in view of the breadboard and Arduino connections.



Appendix 5B: Example Code

The following code will make the car drive forward until it detects an obstacle closer than the threshold distance; then it will stop until the obstacle is removed. The delay times can be changed to 0 to prevent delayed reactions to obstacles. Note that the code contains additional functions for steering that are not used in this example.

```
// declare variables for motor control pins
const int leftMotorForwardPin = 11;
const int leftMotorBackwardPin = 12;
const int rightMotorForwardPin = 9;
const int rightMotorBackwardPin = 10;
const int leftMotorSpeedPin = 6;
const int rightMotorSpeedPin = 5;

// declare variables for sensor pins
const int triggerPin = 8;
const int echoPin = 7;

// define variables for duration of the ping and distance
long duration;  // time for echo to return in microseconds
long cm;        // distance to target in centimeters

// define threshold distance for stopping
const long threshold = 20;
```

```
void setup(){ // code that only runs once
  // set motor control pins as outputs
  pinMode(leftMotorForwardPin,OUTPUT);
  pinMode(leftMotorBackwardPin,OUTPUT);
  pinMode(rightMotorForwardPin,OUTPUT);
  pinMode(rightMotorBackwardPin,OUTPUT);
  pinMode(leftMotorSpeedPin,OUTPUT);
  pinMode(rightMotorSpeedPin,OUTPUT);

  // set sensor pin modes
  pinMode(triggerPin,OUTPUT);
  pinMode(echoPin,INPUT);

}

void loop(){  // code that loops forever
  // take sensor reading and convert to centimeters
  // generate trigger pulse
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  // measure return pulse
  duration = pulseIn(echoPin, HIGH);
  cm = duration/58;

  // stop if distance is less than threshold, otherwise
  // drive forward
  if(cm<threshold){
    stopDriving(1000);
  }
  else{
    driveForward(1000,255,255);
  }
}
```

```
void driveForward(int delay_time, int leftSpeed, int rightSpeed){
  // set pins to make car drive forward then wait for delay_time ms
  analogWrite(leftMotorSpeedPin,leftSpeed);
  analogWrite(rightMotorSpeedPin,rightSpeed);
  digitalWrite(leftMotorForwardPin,HIGH);
  digitalWrite(leftMotorBackwardPin,LOW);
  digitalWrite(rightMotorForwardPin,HIGH);
  digitalWrite(rightMotorBackwardPin,LOW);
  delay(delay_time);
}

void stopDriving(int delay_time){
  // set pins to make car stop then wait for delay_time ms
  digitalWrite(leftMotorForwardPin,LOW);
  digitalWrite(leftMotorBackwardPin,LOW);
  digitalWrite(rightMotorForwardPin,LOW);
  digitalWrite(rightMotorBackwardPin,LOW);
  delay(delay_time);
}

void driveBackward(int delay_time, int leftSpeed, int rightSpeed){
  // set pins to make car drive backward then wait for
  // delay_time ms
  analogWrite(leftMotorSpeedPin,leftSpeed);
  analogWrite(rightMotorSpeedPin,rightSpeed);
  digitalWrite(leftMotorForwardPin,LOW);
  digitalWrite(leftMotorBackwardPin,HIGH);
  digitalWrite(rightMotorForwardPin,LOW);
  digitalWrite(rightMotorBackwardPin,HIGH);
  delay(delay_time);
}
```

```
void turnRight(int delay_time, int leftSpeed, int rightSpeed){
  // set pins to make car turn right then wait for delay_time ms
  analogWrite(leftMotorSpeedPin,leftSpeed);
  analogWrite(rightMotorSpeedPin,rightSpeed);
  digitalWrite(leftMotorForwardPin,HIGH);
  digitalWrite(leftMotorBackwardPin,LOW);
  digitalWrite(rightMotorForwardPin,LOW);
  digitalWrite(rightMotorBackwardPin,HIGH);
  delay(delay_time);
}

void turnLeft(int delay_time, int leftSpeed, int rightSpeed){
  // set pins to make car turn left then wait for delay_time ms
  analogWrite(leftMotorSpeedPin,leftSpeed);
  analogWrite(rightMotorSpeedPin,rightSpeed);
  digitalWrite(leftMotorForwardPin,LOW);
  digitalWrite(leftMotorBackwardPin,HIGH);
  digitalWrite(rightMotorForwardPin,HIGH);
  digitalWrite(rightMotorBackwardPin,LOW);
  delay(delay_time);
}
```