Build an Arduino Self-Driving Car

Lesson 7: Following a Lane

Estimated time: 2 hours

Introduction

In this lesson students will learn to use infrared light sensors to make their car automatically stay in a lane marked by two lane lines. They can also program additional behaviors like stopping at a crosswalk. The exact behaviors they program will depend on the road you have set up for them.

Learning Objectives

- Apply knowledge from previous lessons about motor control
- Program a car's steering behavior based on sensor readings

Preparation

- You will need a track or road set up for students to test their cars. See the "Track Creation Guide" document in the Welcome section for tips.
- Try this activity yourself before trying it with students. The process for calibrating the infrared sensors is important to get your cars to work properly. You can keep a working car as a demonstration.
- For clarity, the initial breadboard diagrams in this lesson just show the infrared sensors and not the H-bridge or ultrasonic sensor. However, students can add the sensors to their existing circuits (see Appendix 7C).
- Students will need craft materials (e.g. tape and popsicle sticks or cardboard) to mount their sensors to their cars.

Materials

Students will need all of the circuit materials from previous lessons. No additional 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	0:15	Remind students how the infrared sensors work.
3	2:36	Build a circuit with one sensor connected to a digital pin and one sensor connected to an analog input pin (see Appendix 7A).
4	5:16	Enter the example code (see Appendix 7A) and upload it to the Arduino. Test the two sensors and compare their performance with **`analogRead`** and **`digitalRead`**.
5	9:53	Move the **`analogRead`** sensor up and down over different surfaces and watch the output in the serial monitor. If needed, change the value for the **`threshold`** variable in the code so you can clearly differentiate between the surfaces.
6	10:39	Change the circuit and code so both sensors are connected to an analog input pin (see Appendix 7B, but let students try this on their own first).
7	13:29	Mount the sensors on the car using craft materials like tape and popsicle sticks or cardboard. Make sure the sensors are facing downward and not scraping against the ground. Program the Arduino so the car's steering behavior is determined by the sensors, for example, turning left to stay in the lane if the right sensor detects a line (see Appendix 7D for example code, but let students try this on their own first).

8	21:36	Test your car on the road and observe its behavior. You may need to adjust some of the parameters in your code (such as driving speeds or threshold values) to make sure it stays on the road.

Assessment

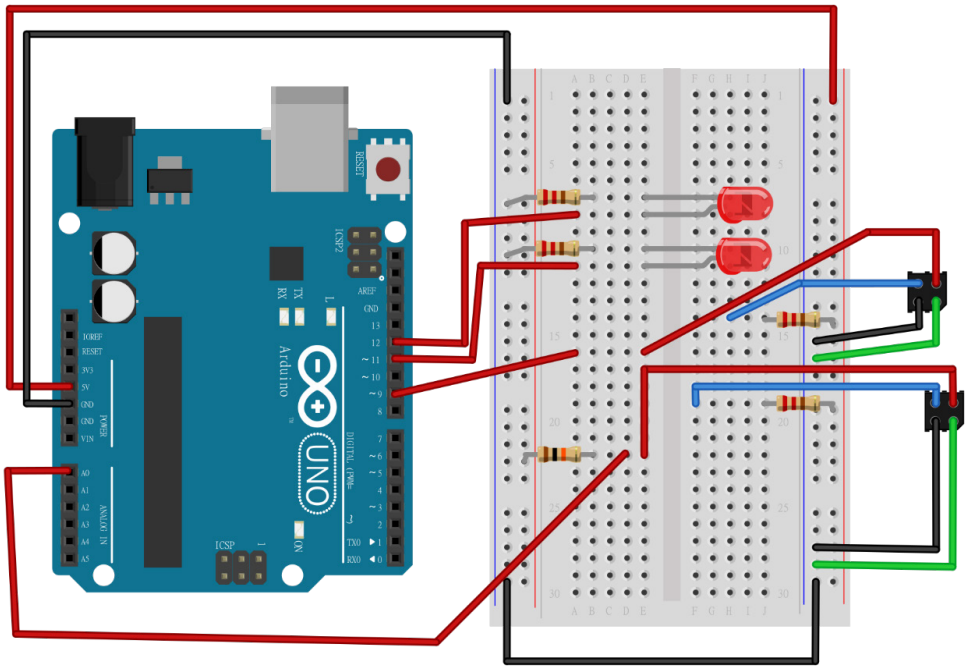
As you walk around the class:

- Confirm that students can get the initial circuit working with one **`analogRead`** and one **`digitalRead`** sensor.
- Make sure students can change the circuit and code to use both sensors with **`analogRead`**.
- Observe how students test their cars on the track and if they can make any needed changes to improve the car's behavior (e.g. overshooting turns).

Appendix 7A: One Analog and One Digital Input

The following diagram shows one sensor connected to an analog input and one sensor connected to a digital input.

Note: this diagram omits the H-bridge for clarity. If you already have the H-bridge circuit assembled, you may need to connect your sensors to different breadboard rows.



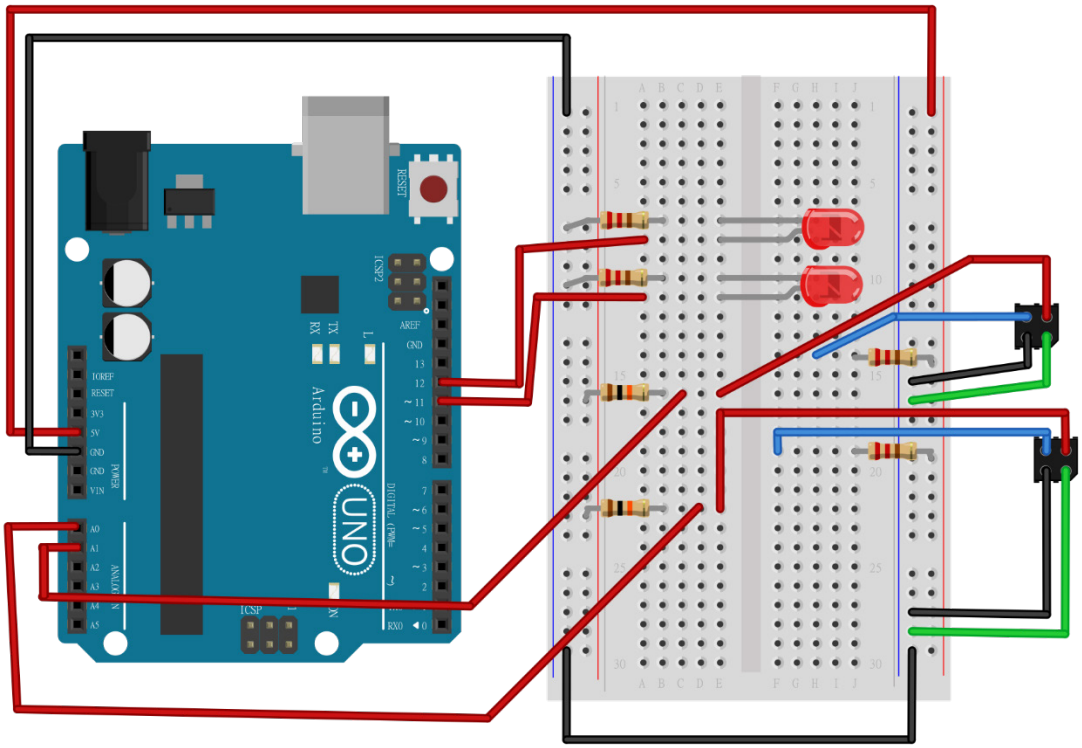
The following code will control the two LEDs based on the sensor readings.

```
// declare constants for pins
const int sensor1Pin = 7;
const int sensor2Pin = A0;
const int led1Pin = 12;
const int led2Pin = 11;

// variables for sensor readings
int sensor1;  // this will only be high or low
int sensor2;  // this will be an analog reading, 0-1023
```

```
// threshold for analog sensor reading
const int threshold = 200;

void setup() {  // setup code that only runs once
  // set sensor 1 pin as input with internal pullup resistor
  // this way you do not need an external resistor
  // note that you do not need to set analog input pins as inputs
  pinMode(sensor1Pin, INPUT_PULLUP);

  // set LED pins as output
  pinMode(led1Pin,OUTPUT);
  pinMode(led2Pin,OUTPUT);
  Serial.begin(9600);  // initialize serial communication
}

void loop() {  // code that loops forever
  // read the sensor pins
  sensor1 = digitalRead(sensor1Pin);
  sensor2 = analogRead(sensor2Pin);
  Serial.println(sensor2);  // print analog reading

  // control the LEDs
  if(sensor1 == HIGH){  // no IR light detected, sensor sees dark
                        // surface
    digitalWrite(led1Pin,LOW); // turn the LED off
  }
  else{ // IR light detected, sensor sees light surface
    digitalWrite(led1Pin,HIGH); // turn the LED on
  }

  if(sensor2 > threshold){  // higher analog value = less light
                            // reflected (darker surface)
    digitalWrite(led2Pin,LOW); // turn the LED off
  }
  else{ // IR light detected, sensor sees light surface
    digitalWrite(led2Pin,HIGH); // turn the LED on
  }
  delay(10);
}
```

Appendix 7B: Two Analog Inputs

The following diagram shows both sensors connected to analog inputs.



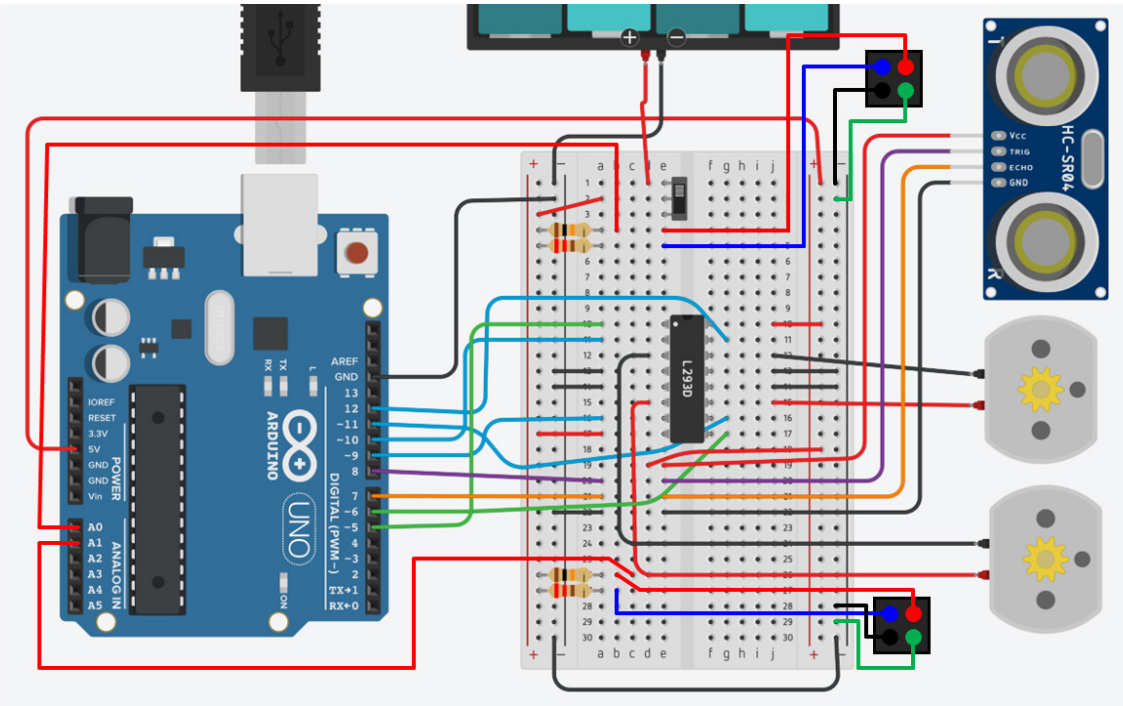
The following code uses analog inputs for both sensors.

```
// declare constants for pins
const int sensor1Pin = A1;
const int sensor2Pin = A0;
const int led1Pin = 12;
const int led2Pin = 11;

// variables for sensor readings
int sensor1;  // this will be an analog reading, 0-1023
int sensor2;  // this will be an analog reading, 0-1023
```

```
// threshold for analog sensor reading
const int threshold = 200;

void setup() {  // setup code that only runs once
  // note that you do not need to set analog input pins as inputs

  // set LED pins as output
  pinMode(led1Pin,OUTPUT);
  pinMode(led2Pin,OUTPUT);
  Serial.begin(9600);  // initialize serial communication
}

void loop() {  // code that loops forever
  // read the sensor pins
  sensor1 = analogRead(sensor1Pin);
  sensor2 = analogRead(sensor2Pin);
  // print both sensor readings
  Serial.print(sensor1);
  Serial.print(" ") ;
  Serial.println(sensor2);

  // control the LEDs
  if(sensor1 > threshold){  // higher analog value = less light
                            // reflected (darker surface)
    digitalWrite(led1Pin,LOW); // turn the LED off
  }
  else{ // IR light detected, sensor sees light surface
    digitalWrite(led1Pin,HIGH); // turn the LED on
  }

  if(sensor2 > threshold){  // higher analog value = less light
                            // reflected (darker surface)
    digitalWrite(led2Pin,LOW); // turn the LED off
  }
  else{ // IR light detected, sensor sees light surface
    digitalWrite(led2Pin,HIGH); // turn the LED on
  }
  delay(10);
}
```

Appendix 7C: Breadboard Diagram with H-bridge

This diagram shows one way to attach the sensors to a breadboard that already has an H-bridge circuit built.



Appendix 7D: Example Autonomous Steering Code

This code will make the car automatically steer to stay on a road marked by two lane lines, and stop at a crosswalk that goes across the road.

```
// 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;

const int sensor1Pin = A1;
const int sensor2Pin = A0;
// variables for sensor readings
int sensor1;  // this will be an analog reading, 0-1023
int sensor2;  // this will be an analog reading, 0-1023

// 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);   // set trigger pin as output
  pinMode(echoPin,INPUT);       // set echo pin as 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;

  sensor1 = analogRead(sensor1Pin);
  sensor2 = analogRead(sensor2Pin);

  // stop if distance is less than threshold, otherwise drive forward
  if((sensor1 > threshold) && (sensor2 > threshold)){
    // what should the car do if neither sensor sees a line?
    driveForward(0,255,255);
  }
  else if((sensor1 < threshold) && (sensor2 > threshold)){
    // what if only the left sensor sees a line?
    turnRight(0,255,255);
  }
```

```
  else if((sensor1 > threshold) && (sensor2 < threshold)){
    // what if only the right sensor sees a line?
    turnLeft(0,255,255);
  }
  else{
    // what if both sensors see a line?
    stopDriving(1000);
  }
}

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);
}
```