

ELEX 2120: Electronic Circuits I

LAB #7 – Robot Project Enhancements

|  |  |  |
| --- | --- | --- |
| Student Name: Faniel Yemane | Student Number: A01182890 | Set: C |
| Student Name: Ethan Bosiak | Student Number: A01314624 | Bench#: 15 |

# Introduction

Our enhancements included using 555 timers to implement blinking left and right LEDs for turns as well as sound effects for each direction.

We also used an Arduino Uno for controlling the robot electronic circuit. We programmed the Arduino to not only control the car but also incorporate additional sensor modules. These included ultrasonic sensors for collision avoidance, as well as an IR receiver diode for remote control.

# Final Project

Our final circuit design includes two additional red LED lights for turning, an IR receiver diode with remote, an Arduino Uno, and two ultrasonic sensors for collision avoidance.

Diagram

Description automatically generated

*Figure 1 – Circuit view of final robot car design*

*Diagram, schematic

Description automatically generated*

*A diagram of a house

Description automatically generated with low confidence*

*A picture containing text, whiteboard

Description automatically generated*

*Diagram, schematic

Description automatically generated*

*Figures 2-5 – Circuit schematics for robot car design*

# Blinking LED Indicators for Turns

We changed our LED system to have two green LEDs on the front, two red LEDs on the back, and the two separate red LEDs on either side of the car. Instead of having a single LED corresponding to a direction, we instead had the LEDs mimic that of car headlights and taillights.

To implement blinking LEDs for turns, we used a 555 timer to create a pulse with a low frequency which would cause an LED to blink. The two side LED lights will blink at the low frequency when the car is turning. We connected the 555-timer signals and the signal for the turning LEDs to another AND gate on a separate 74HC08 IC. This will cause the turning LEDs to blink on and off at the frequency determined by the 555-timer setup.

The LEDs turn on as follows:

* The green LEDs at the front of the car turn on when the car is moving forwards.
* The red LEDs at the back of the car turn on when the car is moving backwards.
* The centre left red LED blinks on and off when the car turns left.
* The centre right red LED blinks on and off when the car turns right.

# Sound Effects for Turns

In addition to sound effects when the robot is moving backwards and forwards, we also added sound effects when the robot is turning. We did this by using another 555 timer that emits a tone when the car is turning left or right.

The sound effects were done so that the robot can still respond to any new commands immediately while a sound is playing. Once it receives a command, the speaker will immediately switch to the corresponding tone.

The sound effects are as follows:

* Two unique frequencies when the car is moving forwards or backwards
* Another frequency when the car is turning either left or right
* No sound effects when the car is not in motion

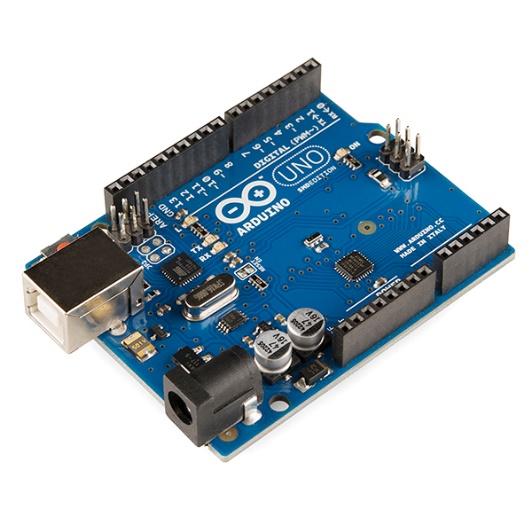
# Arduino Uno

To implement more complex enhancements, we decided to use an Arduino Uno to control the circuitry of our robot car. The Arduino Uno is a microcontroller board with multiple analog and digital pins that can be programmed to control an external circuit. It is easily powered by a +9V battery which we attached to the car. It also provides a +5V supply voltage which we used for our circuit.

The Arduino received infrared signals from the IR receiver transmitted from the IR remote as a digital input. We then programmed the circuit to output the following signals:

* 4 outputs controlled the inputs of the motor driver for movement
* 4 outputs controlled the echo and trigger pins of 2 ultrasonic sensors for collision avoidance

The code we uploaded into the Arduino is attached to the later appendix.



*Figure 6 – The Arduino Uno*

## IR remote control

By using the Arduino, we were able to enhance our robot car even further with IR remote control. This made controlling the car more fun and convenient than having to pick it up and flick a switch.

We wired an IR receiver diode to our Arduino. The IR receiver is a photodiode which converts the IR signal sent from a remote into an electrical signal. Each IR signal is unique depending on the remote button pressed. The diode will send the signals to the Arduino as the sole input to control the direction.

The IR commands were as follows:

* ON: The H-Bridge IC is enabled.
* OFF: The H-Bridge IC is disabled and all inputs to the H-Bridge are set low.
* STOP: All inputs to the H-bridge are set low.
* FORWARD, BACKWARD: The robot car moves forward and backward.
* LEFT, RIGHT: The car moves left or right.

By implementing IR remote control, the Grayhill switch used for prior builds became unnecessary and was removed. We were then left with a basic remote-controlled car.



*Figures 7-8 – The IR receiver diode and remote. We labelled specific remote buttons for user commands*

## Collision avoidance

Our last enhancement is using HC-SR04 ultrasonic sensors to detect the distance between the car and any potential obstacles to avoid.

The HC-SR04 ultrasonic sensor is a sensor module that uses echolocation to detect the distance from it and nearby objects. It contains a transmitter which converts an electrical signal to ultrasonic sound pulses. It also contains a receiver that listens to the transmitted pulses. When the receiver receives these pulses reflected from an object, it will output a pulse whose width is proportional to the distance of the object in front. The distance can then be calculated and used in the Arduino program.

With this feature, we made the car stop or go in the opposite direction if it got too close to a wall or obstacle. Two sensors were used to detect obstacles near the front and back of the car. For example, if the car is about to drive into an obstacle, it will automatically move in the opposite direction without any user input necessary. If the car is about to back up into a wall, it will go forward before contact.

A picture containing electronics

Description automatically generated

*Figure 9 – The HC-SR04 ultrasonic sensor*

# Conclusions

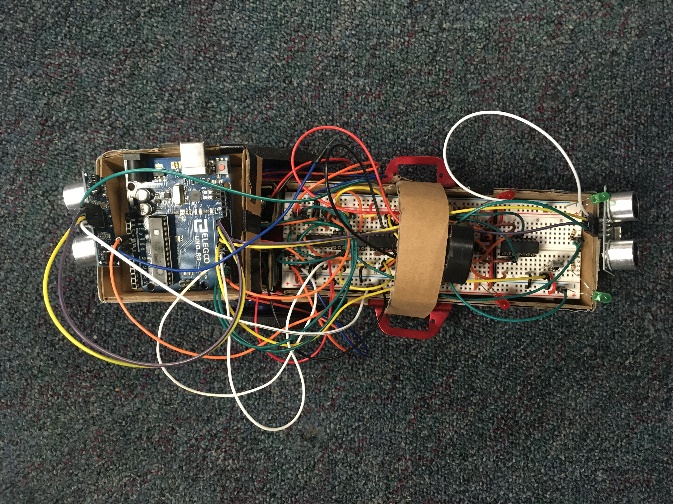
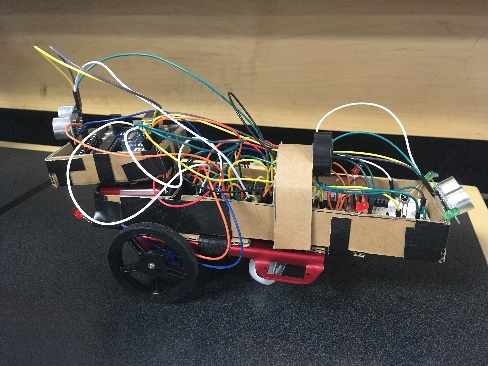
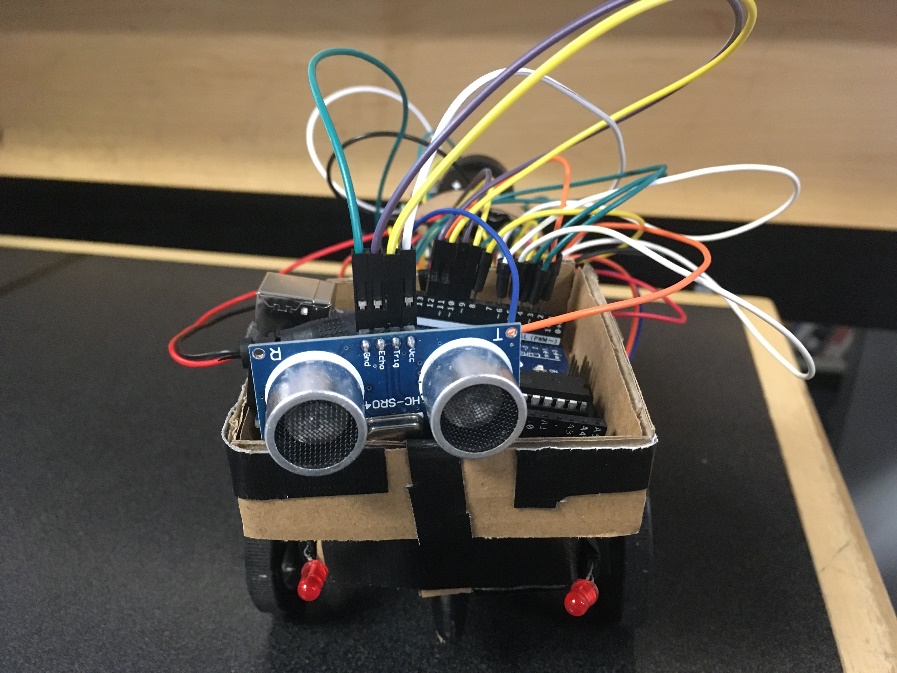
We learned how to add various enhancements to our robot car, both manually with 555 timers and through code. We implemented LED lighting and sound effects for turns as well as integrated an Arduino Uno to move the car when commanded by an IR remote, automatically avoiding obstacles in its way.

We initially used a physical Grayhill switch to directly control the robot car. We then transitioned to connecting the Grayhill switches to the Arduino to check if the Arduino could register the switch inputs and operate the motor driver. We then finally transitioned to using an IR remote for user input and abandoning the switches altogether. Step by step, we were able to troubleshoot the robot circuitry, debug the Arduino program, and fix design flaws.

The project was overall a testament to our ability to implement electronic and digital circuits as well as a fun experience in designing a cool and functioning remote-controlled car.

A picture containing carriage, cart

Description automatically generatedA picture containing indoor

Description automatically generated A picture containing electronics

Description automatically generated

*Figures 10-15 – Our final robot car design build!*

# Appendix

## Arduino code written in C

#include <IRremote.h> // include the IRremote library

// for motor outputs

#define MOTORS 2

#define LMOTOR\_A 3

#define LMOTOR\_B 4

#define RMOTOR\_A 5

#define RMOTOR\_B 6

#define ftrigPin 9

#define fechoPin 10

#define btrigPin 11

#define bechoPin 12

double backDistance, frontDistance;

// for IR receiver

const int RECV\_PIN = 8; // define the IR receiver pin

IRrecv irrecv(RECV\_PIN); // create a receiver object of the IRrecv class

decode\_results results; // create a results object of the decode\_results class

unsigned long key\_value = 0; // variable to store the key value

/\*

ethan remote - REMOTE FOR PROJECT

forward 0x9cb47

left 0xdcb47

right 0x3cb47

back 0x5cb47

stop 0xbcb47

on 0x16b47

off 0xe6b47

faniel remote

FORWARD = 0xff629d;

LEFT = 0xff22dd;

RIGHT = 0xffc23d;

BACKWARD = 0xffa857;

STOP = 0xff38c7;

tinkercad

forward - 0xfd8877

left - 0xfd28d7

right - 0xfd6897

back - 0xfd9867

pause - 0xfda857

on - 0xfd00ff

off - 0xfd807f

\*/

long readUltrasonicDistance(int triggerPin, int echoPin)

{

  pinMode(triggerPin, OUTPUT);  // Clear the trigger

  digitalWrite(triggerPin, LOW);

  delayMicroseconds(2);

  // Sets the trigger pin to HIGH state for 10 microseconds

  digitalWrite(triggerPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(triggerPin, LOW);

  pinMode(echoPin, INPUT);

  // Reads the echo pin, and returns the sound wave travel time in microseconds

  return pulseIn(echoPin, HIGH);

}

void setup(){

  // for IR remote

  Serial.begin(9600); // begin serial communication with a baud rate of 9600

  irrecv.enableIRIn(); // enable the receiver

  irrecv.blink13(true); // enable blinking of the built-in LED when an IR signal is received

  pinMode(ftrigPin, OUTPUT);

  pinMode(fechoPin, INPUT);

  pinMode(btrigPin, OUTPUT);

  pinMode(bechoPin, INPUT);

}

void loop(){

  frontDistance = 0.01723 \* readUltrasonicDistance(ftrigPin, fechoPin);

  backDistance = 0.01723 \* readUltrasonicDistance(btrigPin, bechoPin);

  // go backward if front too close

  if (frontDistance < 5) {

      digitalWrite(LMOTOR\_A, LOW);

      digitalWrite(LMOTOR\_B, HIGH);

      digitalWrite(RMOTOR\_A, LOW);

      digitalWrite(RMOTOR\_B, HIGH);

  }

   // go forward if front too close

  if (backDistance < 5) {

      digitalWrite(LMOTOR\_A, HIGH);

      digitalWrite(LMOTOR\_B, LOW);

      digitalWrite(RMOTOR\_A, HIGH);

      digitalWrite(RMOTOR\_B, LOW);

  }

  if (irrecv.decode(&results)){ // decode the received signal and store it in results

    Serial.println(results.value, HEX);

    Serial.print("front ");

    Serial.println(frontDistance);

    Serial.print("back ");

    Serial.println(backDistance);

    // TURN ON

    if (results.value == 0x16b47) {

      digitalWrite(MOTORS, HIGH);

    }

    // TURN OFF

    if (results.value == 0xe6b47) {

      digitalWrite(MOTORS, LOW);

      digitalWrite(LMOTOR\_A, LOW);

      digitalWrite(LMOTOR\_B, LOW);

      digitalWrite(RMOTOR\_A, LOW);

      digitalWrite(RMOTOR\_B, LOW);

    }

    // stop

    if (results.value == 0xbcb47) {

      digitalWrite(LMOTOR\_A, LOW);

      digitalWrite(LMOTOR\_B, LOW);

      digitalWrite(RMOTOR\_A, LOW);

      digitalWrite(RMOTOR\_B, LOW);

    }

    // MOVE FORWARD

    if (results.value == 0x9cb47) {

      digitalWrite(LMOTOR\_A, HIGH);

      digitalWrite(LMOTOR\_B, LOW);

      digitalWrite(RMOTOR\_A, HIGH);

      digitalWrite(RMOTOR\_B, LOW);

    }

    // MOVE BACKWARD

    if (results.value == 0x5cb47) {

      digitalWrite(LMOTOR\_A, LOW);

      digitalWrite(LMOTOR\_B, HIGH);

      digitalWrite(RMOTOR\_A, LOW);

      digitalWrite(RMOTOR\_B, HIGH);

    }

    // MOVE LEFT

    if (results.value == 0xdcb47) {

      digitalWrite(LMOTOR\_A, LOW);

      digitalWrite(LMOTOR\_B, HIGH);

      digitalWrite(RMOTOR\_A, HIGH);

      digitalWrite(RMOTOR\_B, LOW);

    }

    // MOVE RIGHT

    if (results.value == 0x3cb47) {

      digitalWrite(LMOTOR\_A, HIGH);

      digitalWrite(LMOTOR\_B, LOW);

      digitalWrite(RMOTOR\_A, LOW);

      digitalWrite(RMOTOR\_B, HIGH);

    }

    irrecv.resume(); // reset the receiver for the next code

  }

}