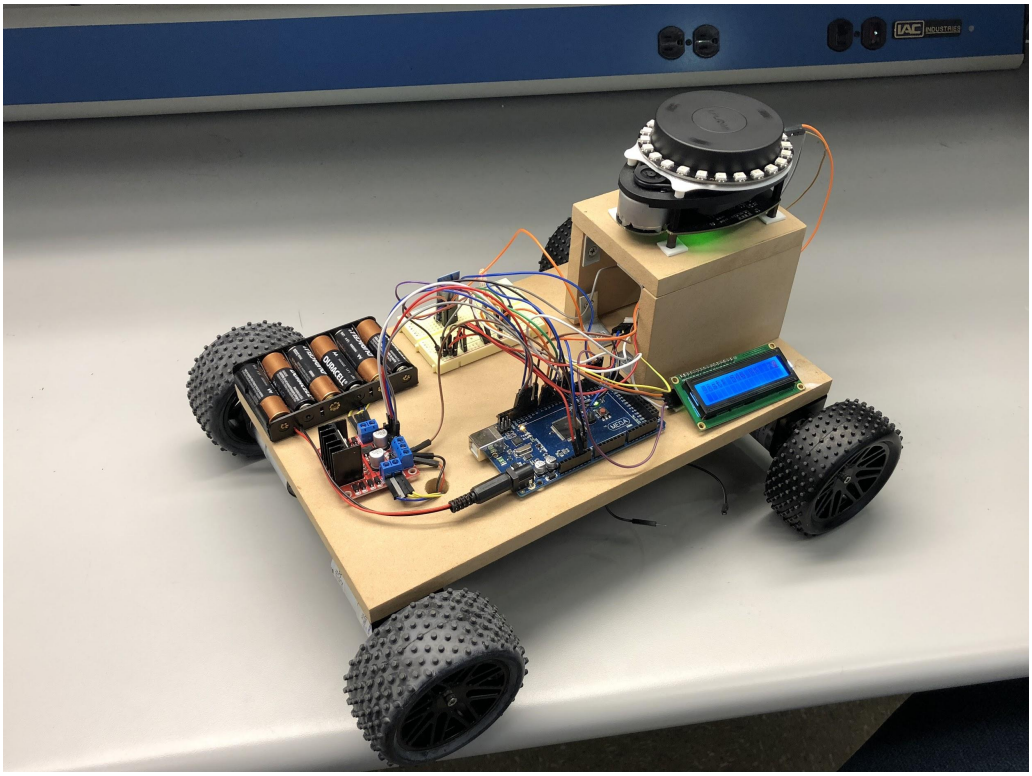
RC LIDAR Detection Vehicle

By: Yuta Nguyen, Alvin Tran, Hoang Nguyen

Faculty Advisor: Dr. David Cheng

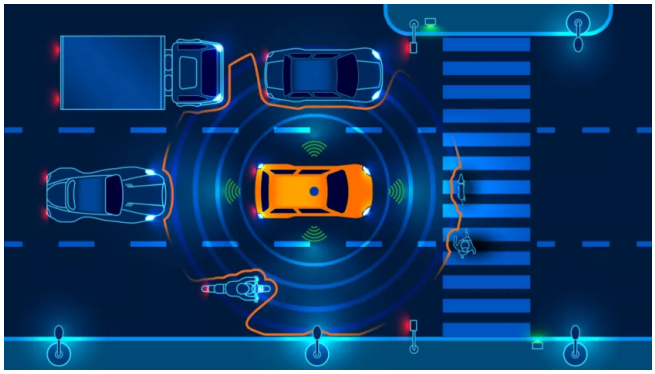


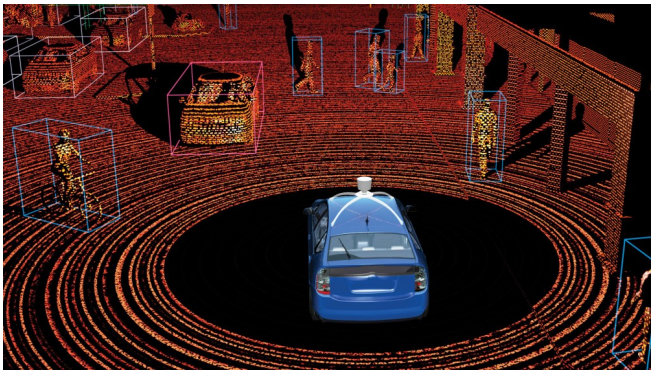
Electrical Engineering
Senior Design Project
SPRING 2019
Course Number: 13579

Table of Contents

Abstract

LIDAR, functions like a radar or sonar which utilizes light from laser energy to detect objects rather than using radio or sound waves. By calculating the amount of time it takes for light to reach the object and back to the LIDAR's receiver, this is used to collect data to create a 3D visualization of the target. Like many companies (Google, NASA, Tesla etc.), LIDAR is fairly common and is used to survey tasks such as human-made surroundings, different environments, and assist autonomous vehicles in navigation. Our project focuses to produce an RC car with a LIDAR Detection system that will allow the LIDAR to detect its surroundings, and allow the RC car to measure accurate distance readings through an RGB LED light and LCD screen. Furthermore, this RC based car will be utilized to show how modern cars in this age, use lidar to detect cars around them, objects, blind spots, and be utilized as a way to monitor surroundings. LIDAR serves as a sense to why this type of sensor is a necessity for public transportation to prevent more accidents and injuries for cars autonomously and man driven.





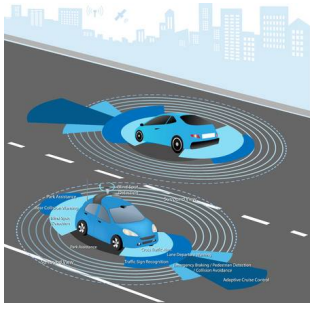
Project Description

_____In this project, we will be designing an RC based car; remote controlled via Bluetooth vehicle with a 360° LIDAR system in order to detect what objects are around the vehicle and the distance measuring between the LIDAR device and the closest object. First, we will be

implementing a low cost LIDAR based system provided by Slamtec, which gives a max range of 12 meters. Our design is an RC based car with a 360° LIDAR system in order to detect what objects are around the vehicle which measures distance in real time, which uses a three-dimensional visualization of the surroundings while the car is in motion. By implementing an RGB LED light and LCD screen display with an established code, we will detect range and direction of the cars surroundings and visually project this reading. An RGB LED light will contain a series of different colors to help signify how close or far the car is from an object, and project the distance through an LCD screen display. Through the use of an LED light ring, this will act as a sensor to determine the direction an object is projected from towards the RC car.

Objectives

- Build an RC based car with a mounted LIDAR device mounted on top of the car continuously rotating
- LED lights which would change color based on how close the object is, using a LIDAR device spinning continuously in a 360° rotation
- An LCD Screen will be displayed for distance measurements between objects and angle of projection of the nearest object
- This project is used to demonstrate how LIDAR is efficient and useful for avoiding accidents and used throughout autonomous driving vehicles or surveying
- Incorporate an LED light ring to showcase the directional movement and direction that an object is coming from while the car is in motion
- An electronic buzzer is used to tell the user that the object is too close up to 150mm

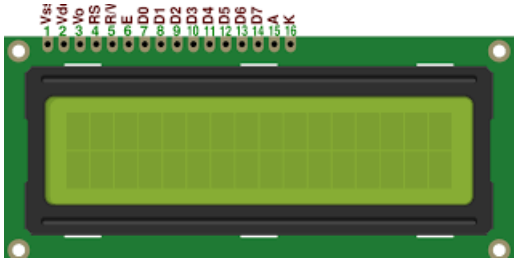


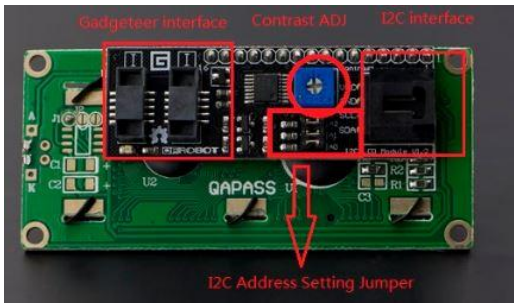
Design Specification and Constraints

Arduino Mega 2560

The Arduino Mega 2560 is microcontroller board that contains 54 digital input/output pins where 14 ports can be used as PWM outputs, 16 analog inputs, 4 UARTs (hardware serial ports). It contains everything needed to support the microcontroller; It's powered by either via a USB or battery to get it started. We will be powering the Arduino Mega with a 9V battery supply, and will be utilizing it as our mainframe to drive all of our components together.

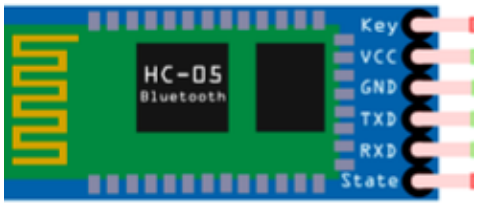
LCD Screen Display





This is a 16x2 LCD display screen with I2C interface. It is able to display 16x2 characters on 2 lines, white characters on a blue lit background. Usually, Arduino LCD display projects will run out of pin resources easily, and it's very complex when wiring the LCD screen properly. With a I2C communication interface, this means that the LCD screen needs only 4 pins: VCC, GND, SDA, and SCL. On top of that, it will save at least 4 digital/analog pins on the Arduino. In order to the LCD display to work properly with the interface, we would need to download the I2C library that supports this LCD screen. We will be utilizing the screen to display the distance from the closest object it reads.

HC-05 Bluetooth Module

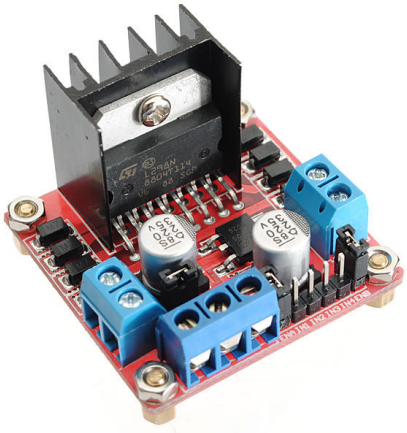


The HC-05 Bluetooth Module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. The communication is via serial communication which makes an easy way to interface with a smart device or PC. The module provides switching modes between the master and slave mode which means it is able to use neither receiving, nor transmitting data. For this project, we will be communicating with 4 DC motors that we attached on the driver (H-Bridge) by a tablet to allow us to control the car via the HC-05 towards the arduino mega.

RPLidar 360° Scanner

The RPLIDAR is a low cost 360° 2D laser scanner developed by SLAMTEC. The system can perform 360° scan within a 6 meter range. It produces 2D point cloud data, that can be used in mapping, localization, and object/environment scanning. This is a A1 model, which is the first model of this type. The lidar device is basically a laser triangulation measurement system, which emits a laser signal, senses an object, and is reflected back to the lidar receiver, and is then registered as a detected light pulse within the processor. The RPLidar can work excellent in all kinds of indoor and outdoor environments without any sunlight. For this project, this will be our main component for collecting data for both the distance and direction.

L298N H-Bridge

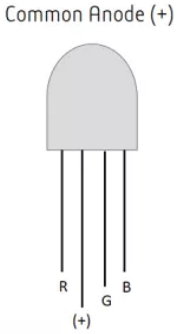


The L298N is an H Bridge Motor shield which is designed to drive inductive loads such as relays, solenoids, etc. The main purpose for this driver to to drive two drivers on each side of the output pins in parallel, to control the speed and direction of each of the pair independently.

NiMH Battery: 12V 4200mAh

The NiMH battery is a Nickel-metal hydride battery, which has the capability of having two-three times the capacity of an equivalent size Nickel-Cadmium battery. We will be using a 12V power supply to power up the L298N driver, since we need at least 12V to keep the DC motors spinning at a maximum rate of speed.

RGB LED LIGHT



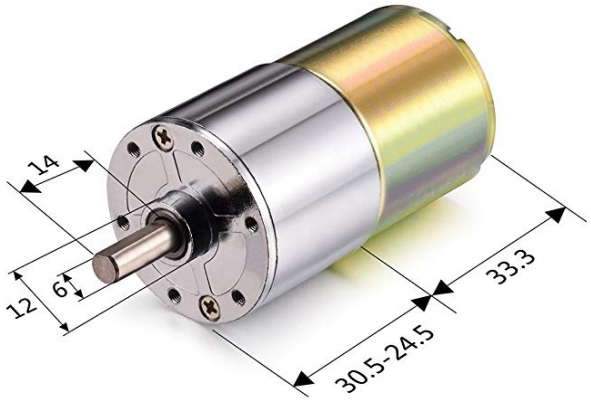
The RGB LED light, is a type of LED light which mainly produces red, blue, and green LED's. It combines these three colors to produce over 16 million hues of light. Note that not all colors are possible. Some colors are "outside" the triangle formed by the RGB LED's. Also, pigment colors such as brown or pink are difficult, or impossible, to achieve. For this project, we will be using a common anode LED, which means that the LED will turn on when logic is high and all three LED's share a positive connection. Furthermore, we will utilize its color technique as the strength of how close the object is going to be.

NeoPixel 24 bit LED RING

This is a 24 bit LED NeoPixel Ring, which has a full diameter of 96mm equipped with 5050 WS2812 RGB LEDs. Each NeoPixel Stick has about 18 mA constant current drive so the

color will be very consistent even if the voltage varies, and requires 5V input voltage. The ring is equipped with a single data line with a very timing-specific protocol requiring a real-time microcontroller with a 8 MHz or faster processor such as the arduino. Likewise, the LED ring will act as the directional component for the project, and data will be received from the LIDAR spinning relative to the output data received by the LIDAR system.

Uxcell 12V DC 200 RPM Gear Motor



The Uxcell 12V DC 200 RPM Gear Motor, is a small size dc gear motor with low noise and high torque. These motors can run up to 7.2 W, has a No-Load Current of 0.15 A, and a Load Current of 0.6 A. For these motors to operate, the positive and negative terminals will be connected to the L928N H-Bridge Motor Shield (Positive and negative can be switched since motor is reversible). The max speed, 200 RPM running at 12V, the motor can also be adjusted by a DC motor speed controller or an Arduino, to incorporate a delay or run at full load operation.

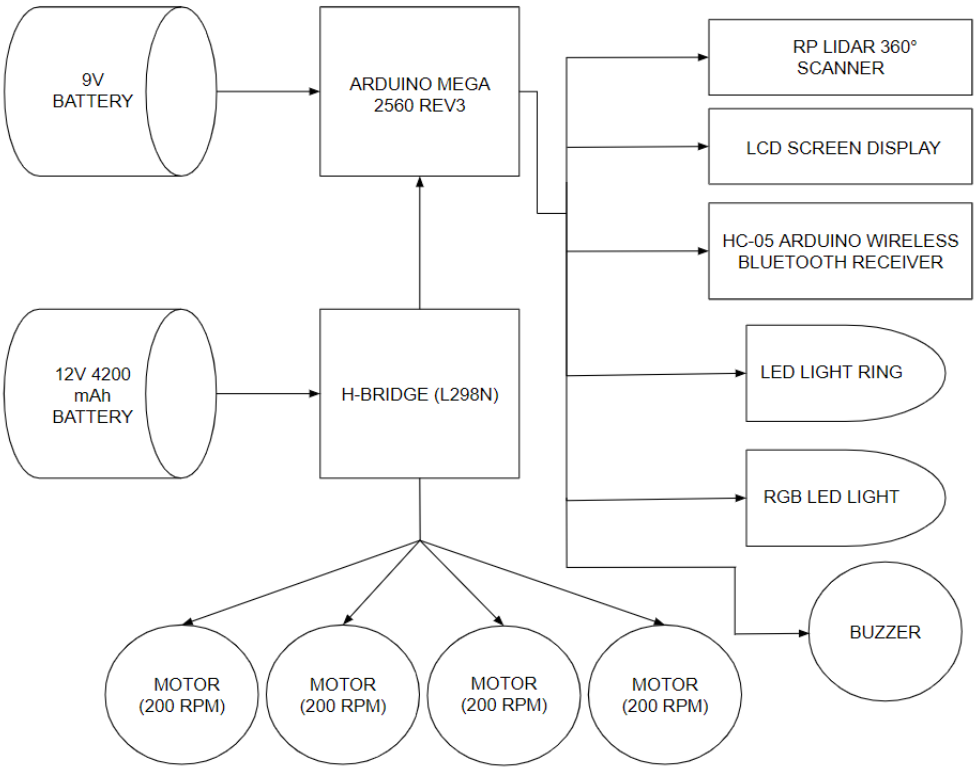
Electronic Tone Buzzer Alarm DC 3-24V

The Active Electronic Alarm Buzzer has an alarm diameter height of 22mm x 10mm that has an operating voltage from 3-24 Volts. This Active Electronic Alarm Buzzer is connected to an input of the Arduino Mega 2560 REV3 and coded to buzz once the object reaches approximately 150 mm within vicinity of the buzzer.

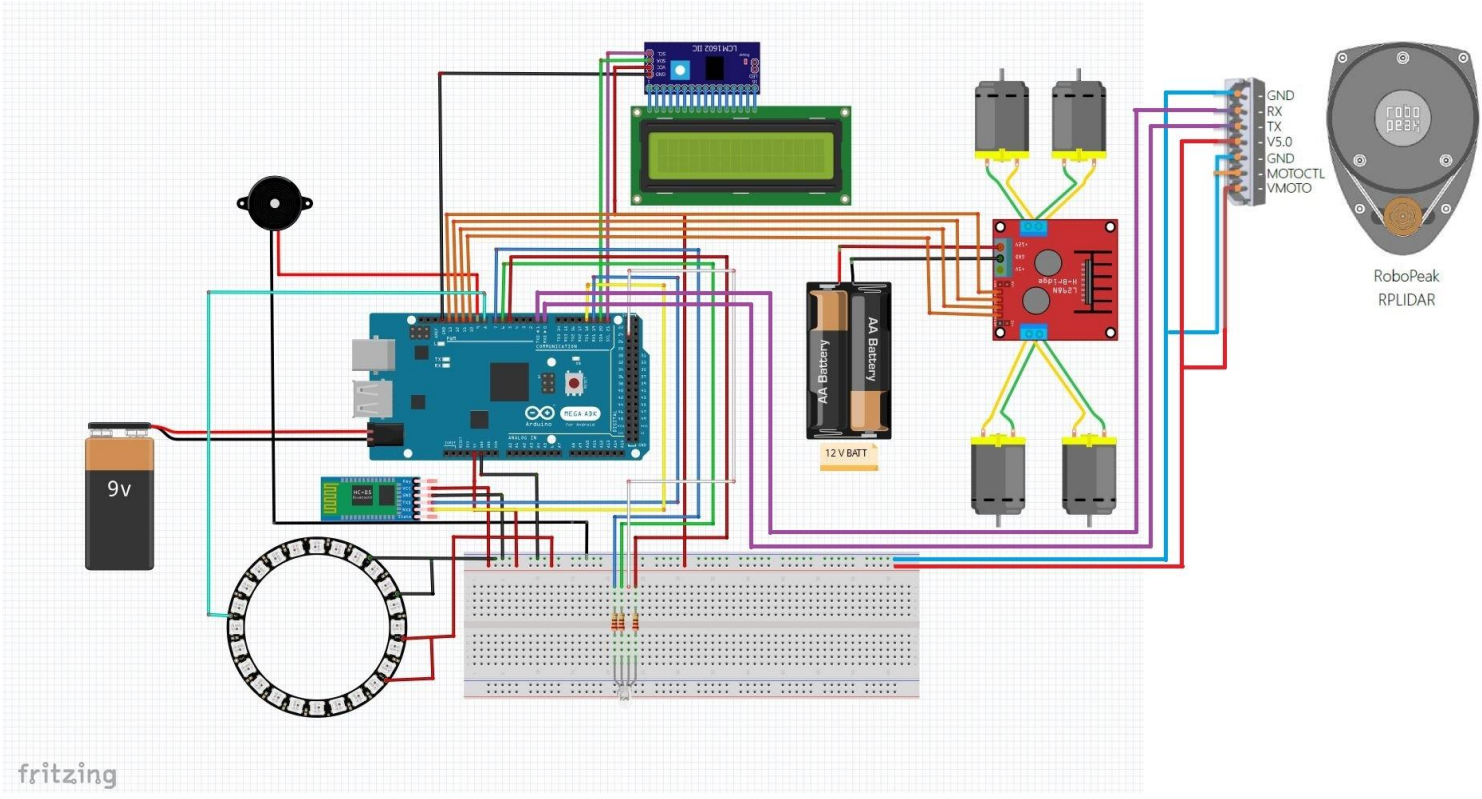
150 Ω 5% Tolerance Resistors

The 150 Ω 5% Tolerance Resistors are pull-up resistors used for digital logic circuits. The floating pin connected to the RGB LED light consists of two modes, high and low. Tying the pull-up resistor to a known voltage of 5V allows the floating pin to reach a specific state. We want the circuit to be certain of the state of all its inputs, and utilizing a pull-up resistor for the RGB LED light allows it work properly with no rapid pulsation.

BLOCK DIAGRAM AND FUNCTION DESCRIPTION



SCHEMATIC DIAGRAM



CIRCUIT DESCRIPTION

Using the program Fritzing, we were able to incorporate the full design detail. This program allows us to implement each component provided by a third party user. However, the RPLidar wasn't available in the program, so we had to use paint to crop an image and edit the wires to get the full design. As a result, we were able to organize each component individually and verify that the schematic works. Next for the block diagram, it gives more a rough sketch on how our final schematic will look like, and it gives us a template on each description of the component.

To start designing the board, we will be using the Arduino Mega 2560 REV3 as our main microcontroller to drive the whole project. We are using a bigger arduino so we can utilize more digital and PWM pins. Next starting with the RPLidar itself, we connected both the power pins and ground pins together and placed them on the breadboard. Since the VMOTO pin is constant, we placed it on a 5V power supply. For the MOTOCTL pin, we connected it into a pulse width modulation pin, so we can change the speed of the motor if we wanted to. Lastly, we connected both the transmitter and receiver pins to the mega in order to give and receive data from the LIDAR.

Likewise, the bluetooth module is also using both transmitter and receiver pins, so we can connect our RC car via tablet to be controlled. With multiple serial ports, we are able to connect our electrical components simultaneously with the LIDAR itself, since they are both using TX/RX pins. In addition, the modules needed to be powered with a 5V source in order for the bluetooth to operate properly.

For the DC motors, we needed an H-Bridge to drive all four motors. Two of the the DC motors are placed in parallel on each side of the output pins. Then, we connected the four input pins on the H-Bridge to four individual PWM pins to drive the motors to get the car moving. Lastly, we connected an external 12V battery supply to power up the H-Bridge itself, since the rectifier needs at least a minimum amount of voltage to power the motors to turn and rotate.

Now, in order to incorporate the distance of the LIDAR, we used a 16x2 LCD screen with an I2C driver, so it'll be easier to wire to the mainframe. Since it's using an I2C device, we needed to locate the LCD's address in order for the LCD screen display to work properly. So, we connected both Serial Data and Clock pins to the microcontroller. Then, we connected both the 5V pin and the ground pin to the breadboard itself.

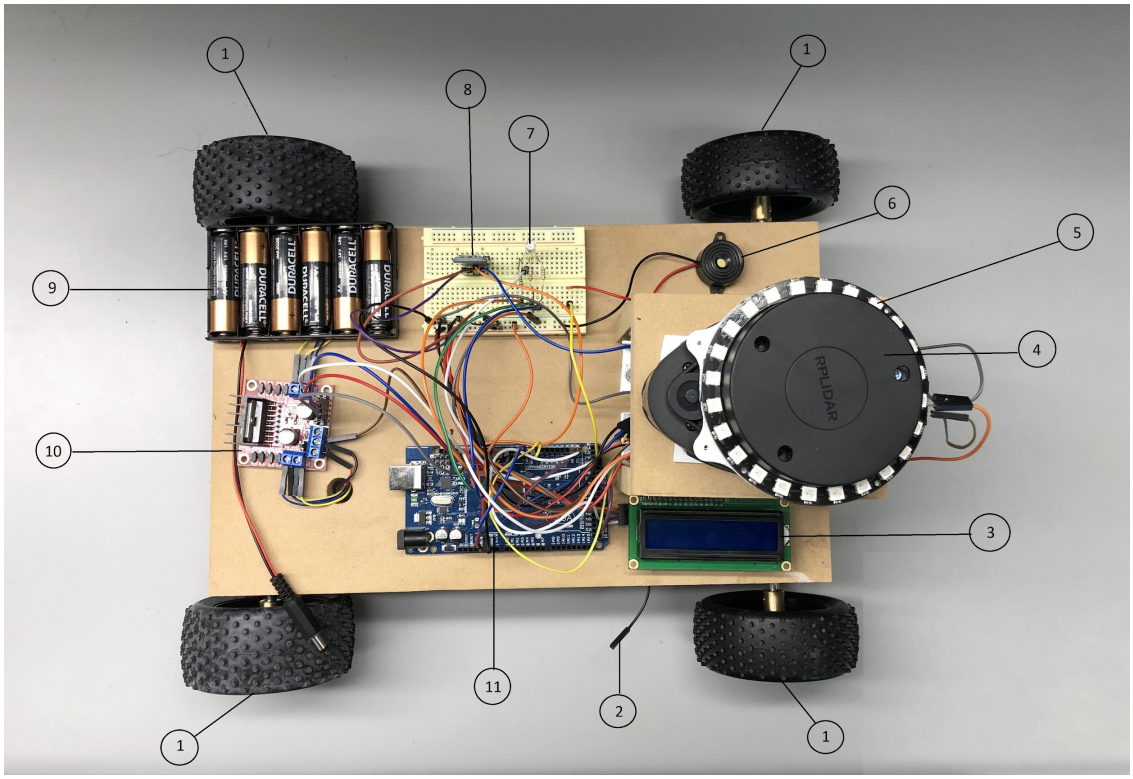
Finally, for the directional component. First we incorporated the RGB LED light. We connected 3 of the color pins: red, green, and blue to a PWM input signal to control the light flickering and speed. The main input pin is power digitally to enable the RGB LED light itself. Next, the LED light ring is a simple connection with just the data input pin, power 5V, and ground pins respectively. Lastly, we implement a buzzer using a digital input pin to operate it, and a ground pin to complete the connection.

PARTS LIST/BILL OF MATERIALS

ITEM	DESCRIPTION	QTY	COST	TOTAL COST
RPLiDAR	A1M8 360 Degree Laser Scanner Kit - 12M Rang	1	$ 99.00	$99.00
ARDUINO	MEGA 2560 REV3 Entire Kit	1	$ 85.56	$85.56
HC-05 Arduino Wireless Bluetooth Receiver	Receiver RF Transceiver Module Serial Port Tran	1	$ 10.57	$10.57
L298N	Dual H Bridge Stepper Motor Driver Controller Bo	1	$ 8.99	$8.99
LCD Screen Display	SunFounder IIC I2C TWI Serial 2004 20x4 LCD M	1	$ 12.99	$12.99
RBG LED Light	EDGELEC 100pcs 5mm RGB Tri-color (Red Gree	1	$ 8.99	$8.99
Hex bolt adapter nuts 12mm	DC Gear Motor Coupling Coupler Connector Ada	1	$ 12.98	$12.98
Servo Motors	Uxcell 200 rpm 12 v dc motor 37 mm	4	$ 15.98	$63.92
Batteries	12 V 4200 mAh	1	$ 52.95	$52.95
Wheels	(4-Pack) HobbyPark 1/10 Scale Off Road Buggy	1	$ 15.99	$15.99
Motors Mounting Brackets	To place the DC Motor of 37mm	1	$ 12.99	$12.99
Wires (Breadboard use)	SunFounder 120pcs Breadboard Jumper Wires 2	1	$ 7.68	$7.68
Wood Frame for RC Car (Chassis)	Body Frame made of Columbian Sanded Wood u	1	$ 6.00	$6.00
Buzzer	Electronic Buzzer Connected to LED	1	$ 7.59	$7.59
Poster Board (Printing and Board Itself)		1	$ 70.00	$70.00
				$476.20

Electrical Engineering (EE)	6 (T10)	RC LIDAR Detection	Alvin Tran Yuta Nguyen Hoang Nguyen	Alvin Tran	3	Dr. David Cheng	$510	$510/$440

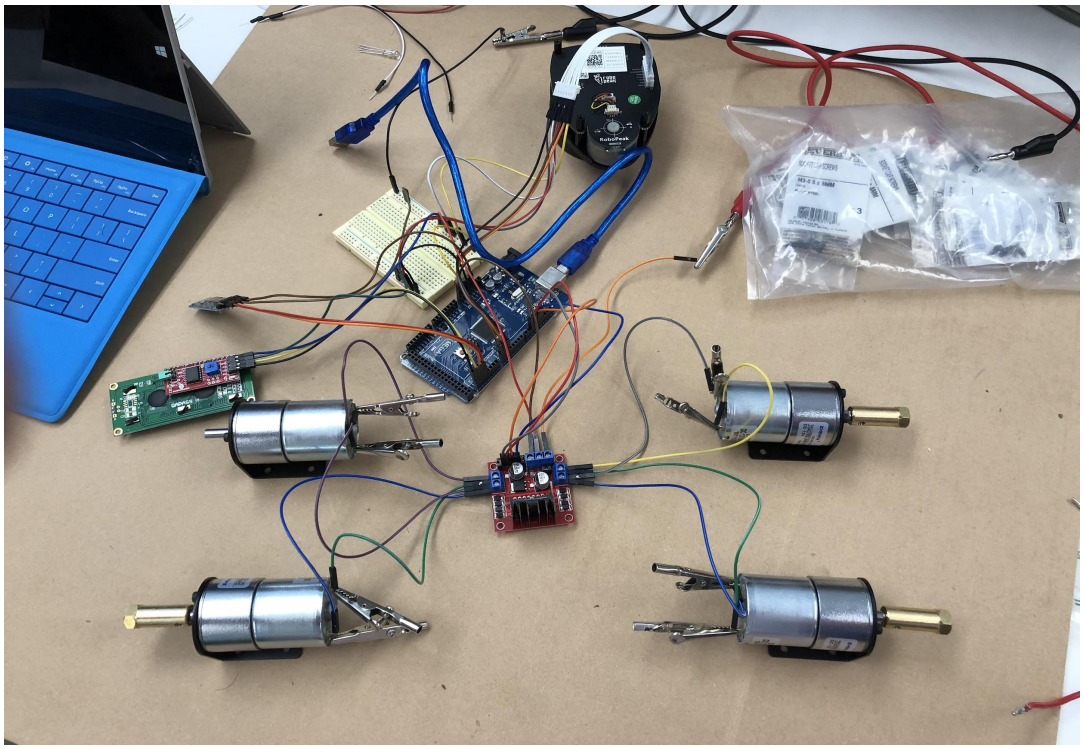
TOP ASSEMBLY DRAWING



#	Qty	Description
11	1	Arduino Mega 2560 REV 3
10	1	H-Bridge (L298N)
9	1	9V Battery
8	1	HC-05 Arduino Wireless Bluetooth Receiver
7	1	RGB LED Light
6	1	Buzzer
5	1	LED Light Ring
4	1	RPLiDAR 360° Scanner
3	1	LED Screen Display
2	1	12V 4200 maH Battery
1	4	Motors and Wheels

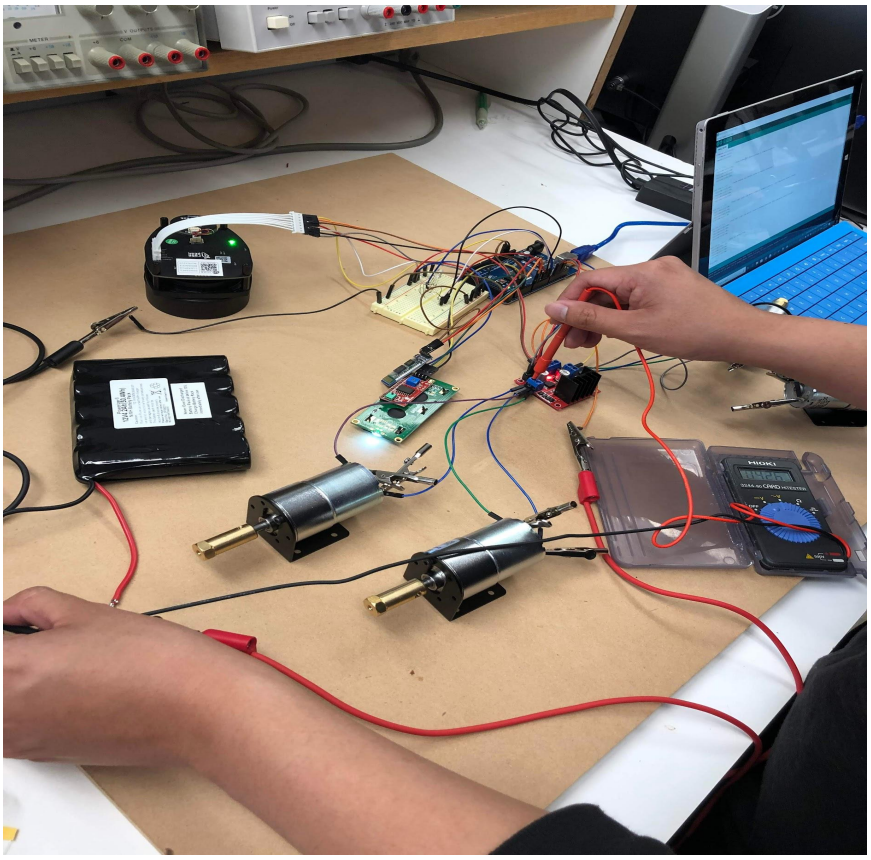
Test Set-Up and Procedures

Progression 1: Motors and Mounts Set-up



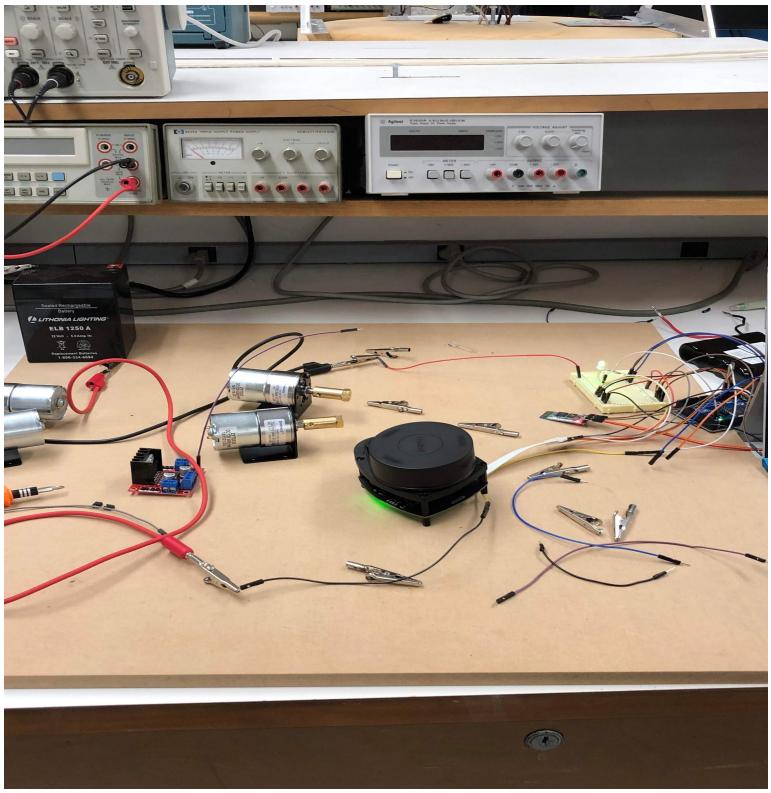
- We configured four 12V DC motors with one H-bridge, and then had to plug in the input drivers into four PWM signals to control the speed of the motor
- However, after further testing, there has been some slight complications with the left output side not responding to our code. This could be because of a faulty part or a cold solder joint in the pin connections. Possible errors that we think may include a faulty H-bridge, loose connections, faulty pin config. etc.

Progression 2: Testing of Potential Over Voltage and Over Current for the DC Motor

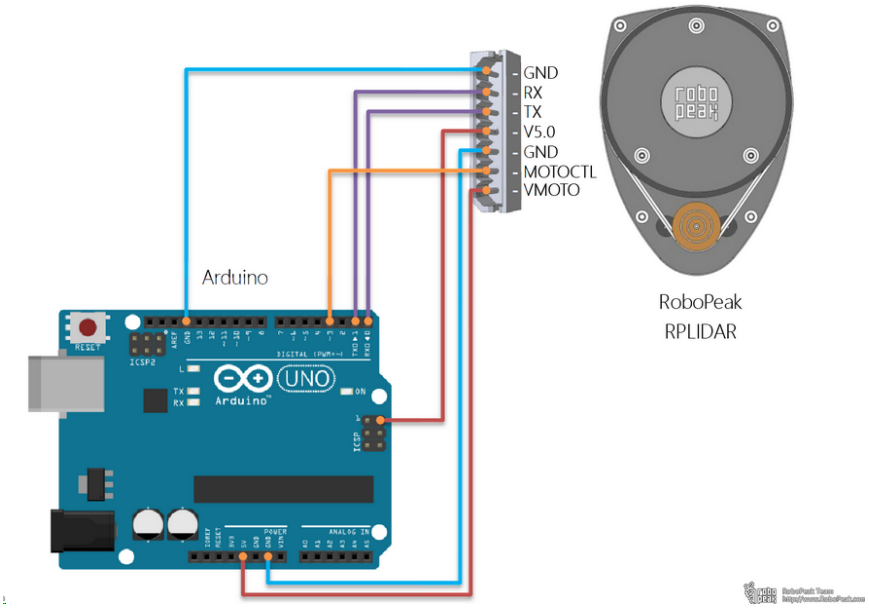


- With some of the motors not working properly, we had to reconfirm the voltages and current on the output side. We determined that no current was coming towards the left side of the motor. This concluded that it could have been a potential error in part faulty and not our circuitry.
- Based on our results, we get a voltage reading of approximately 10V DC initially from the 12V input, which is about a 2V drop due to the motors running but no current driving the motors. Each motor draws about 60 mA per motor as measured through a multimeter.
- **Problem Resolved:** The H-Bridge has a defect towards the PWM signals , therefore, no pulses were commanding one side of the driver towards the motors, possible solder problem, and we ordered new H-Bridges to be implemented

Progression 3: RP-LIDAR Set-Up



We need to Configure the Wire Diagram below



Next, we had to make sure that the RGB LED Light is at the correct location, using a 150 ohm resistor pulling about 20mA towards each leg of the RGB LED Light. Our RGB light is a common anode LED light with 4 pin connections of red, enable, blue, and green pin configurations.

Lastly, we tested the code and as a result, the RGB light works perfectly, but we do need to change the response time since the LIDAR is reacting slow reading data and outputting the distance. We also need to incorporate the LCD screen as well to read distance measurements.

Progression 4: Running Code and Testing the Vehicle

```
#include <RPLidar.h>     //include RPLidar library
#include <Wire.h>        //include wire library for the LCD screen
#include <LiquidCrystal_I2C.h>   //include liquid crystal library to read and write the LCD
#include <SoftwareSerial.h>      //include softwareserial library to define multiple serial ports
// You need to create an driver instance
RPLidar lidar;

LiquidCrystal_I2C lcd(0x27,20,4); //set the LCD Screen
char t;  //variable to turn the car

// Change the pin mapping based on your needs.
////////////////////////////////////////////////////////////////////////
#define LED_ENABLE  24 // The GPIO pin for the RGB led's common lead.
                       // assumes a common positive type LED is used
#define LED_R       7  // The PWM pin for drive the Red LED
#define LED_G       6 // The PWM pin for drive the Green LED
#define LED_B       5 // The PWM pin for drive the Blue LED

#define RPLIDAR_MOTOR 3 // The PWM pin for control the speed of RPLIDAR's motor.
                        // This pin should connected with the RPLIDAR's MOTOCTRL signal
////////////////////////////////////////////////////////////////////////////////


#define NEO_RGBSPACE_MAX (byte)(200L*255/360)
int _r, _g, _b;

//Set current RGB with the hue: HSV(hue, x, x)
void hue_to_rgb( _u8 hue)
{
/*`
    Hue(in Degree):  0 (RED) ----> 60 (Yello) ----> 120 (Green) --..... ----> 360
    Hue'(fit to _u8):0       ----> 60/360*255 ----> 120/260*255 --..... ----> 255
*/

    //convert HSV(hue,1,1) color space to RGB space
    if (hue < 120L*255/360)   //R->G
    {
        _g = hue;
        _r = NEO_RGBSPACE_MAX - hue;
        _b = 0;
    }else if (hue < 240L*255/360)  //G->B
    {
        hue -= 120L*255/360;
        _b = hue;
        _g = NEO_RGBSPACE_MAX - hue;
        _r = 0;
    }else //B->R
    {
        hue -= 240L*255/360;
        _r = hue;
        _b = NEO_RGBSPACE_MAX - _r;
        _g = 0;
    }
}

void displayColor(float angle, float distance)
{
    //1. map 0-350 deg to 0-255
    byte hue = angle*255/360;
    hue_to_rgb(hue);

    //2. control the light

    int lightFactor = (distance>500.0)?0:(255-distance*255/500);
    _r *=lightFactor;
    _g *=lightFactor;
    _b *=lightFactor;

    _r /= 255;
    _g /= 255;
    _b /= 255;

    analogWrite(LED_R, 255-_r);
    analogWrite(LED_G, 255-_g);
    analogWrite(LED_B, 255-_b);
}

void setup() {
  // bind the RPLIDAR driver to the arduino hardware serial
  lidar.begin(Serial);   //Serial0 port for Lidar Device itself
  Serial1.begin(9600);   //Serial1 port for Bluetooth device
```

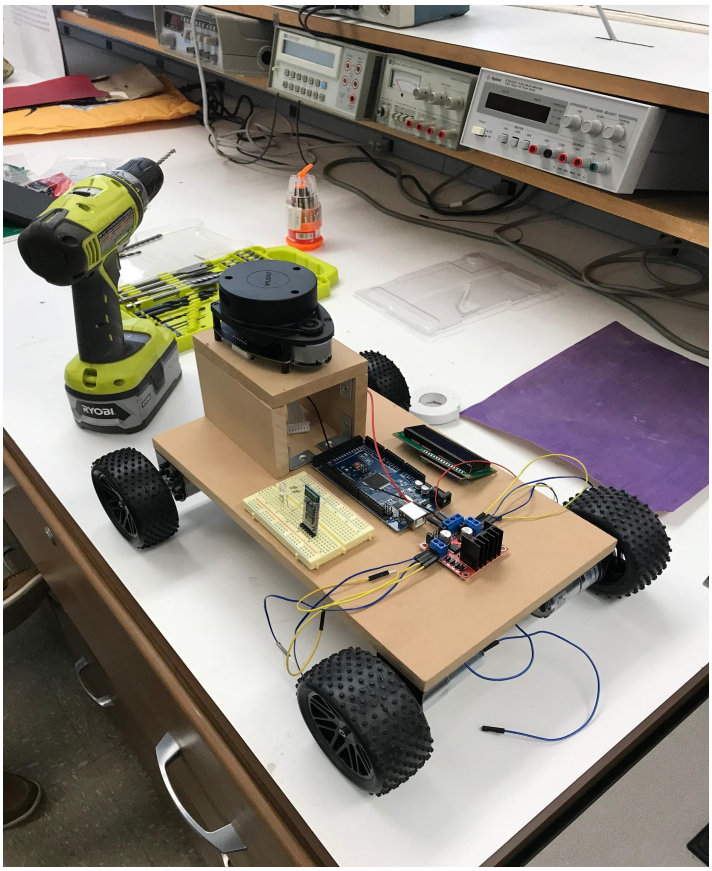
```
void setup() {
  // bind the RPLIDAR driver to the arduino hardware serial
  lidar.begin(Serial);   //Serial0 port for Lidar Device itself
  Serial1.begin(9600);   //Serial1 port for Bluetooth device

  // set pin modes
  pinMode(RPLIDAR_MOTOR, OUTPUT);   //define pwm motor signal for the lidar device

  pinMode(LED_ENABLE, OUTPUT);  //enable all the RGB lights
  pinMode(LED_R, OUTPUT);
  pinMode(LED_G, OUTPUT);
  pinMode(LED_B, OUTPUT);

pinMode(13,OUTPUT);   //left motors forward
pinMode(12,OUTPUT);   //left motors reverse
pinMode(11,OUTPUT);   //right motors forward
pinMode(10,OUTPUT);   //right motors reverse

  digitalWrite(LED_ENABLE, HIGH);

  analogWrite(LED_R,255);
  analogWrite(LED_G,255);
  analogWrite(LED_B,255);

  lcd.init();   //initialize the LCD screen to turn on
  lcd.init();

  lcd.backlight();  //initialize the backlight of the lcd screen
  lcd.blink_on();   //initialize the blinking function
}

float minDistance = 100000;
float angleAtMinDist = 0;

void loop() {
  if(Serial1.available()){    //check if serial1 port is available and read the data via serial monitor
  t = Serial1.read();
  Serial1.println(t);
}


if(t == '1'){            //move forward(all motors rotate in forward direction)
  digitalWrite(13,HIGH);
  digitalWrite(12,LOW);
  digitalWrite(11,HIGH);
  digitalWrite(10,LOW);
}

else if(t == '2'){      //move reverse (all motors rotate in reverse direction)
  digitalWrite(13,LOW);
  digitalWrite(12,HIGH);
  digitalWrite(11,LOW);
  digitalWrite(10,HIGH);
}

else if(t == '3'){      //turn right (left side motors rotate in forward direction, right side motors doesn't rotate)
  digitalWrite(13,LOW);
  digitalWrite(12,LOW);
  digitalWrite(11,HIGH);
  digitalWrite(10,LOW);
}

else if(t == '4'){      //turn left (right side motors rotate in forward direction, left side motors doesn't rotate)
  digitalWrite(13,HIGH);
  digitalWrite(12,LOW);
  digitalWrite(11,LOW);
  digitalWrite(10,LOW);
}

else if(t == '5'){      //STOP (all motors stop)
  digitalWrite(13,LOW);
  digitalWrite(12,LOW);
  digitalWrite(11,LOW);
  digitalWrite(10,LOW);
}
delay(100);


  if (IS_OK(lidar.waitPoint())) {
    //perform data processing here...
    float distance = lidar.getCurrentPoint().distance;
    float angle = lidar.getCurrentPoint().angle;
    String disp1 = String(distance);   //define a string variable to read the distance on the LCD screen
```

```
  if (IS_OK(lidar.waitPoint())) {
    //perform data processing here...
    float distance = lidar.getCurrentPoint().distance;
    float angle = lidar.getCurrentPoint().angle;
    String disp1 = String(distance);    //define a string variable to read the distance on the LCD screen
    String disp2 = String(angle);       //define a string variable to read the angle on the LCD screen
    lcd.clear();  //clear all unneccessary functions on the LCD screen
    lcd.setCursor(0,0);  //set the position on the LCD to the top left
    lcd.print("Dist: ");  //print the word Dist
    lcd.print(disp1);  //print the actual value of distance
    lcd.print(" cm");  //print the units
    lcd.setCursor(0,1);  //set the position on the position on the LCD to the bottom left
    lcd.print("Angle: ");  //print the angle
    lcd.print(disp2);  //print the actual value
    lcd.print(" deg");  //print the degrees

    delay(250);     //delay the speed in which the data is being collected

    if (lidar.getCurrentPoint().startBit) {
      // a new scan, display the previous data...
       displayColor(angleAtMinDist, minDistance);
       minDistance = 100000;
       angleAtMinDist = 0;
    } else {
       if ( distance > 0 &&  distance < minDistance) {
          minDistance = distance;
          angleAtMinDist = angle;
       }
    }
  } else {
    analogWrite(RPLIDAR_MOTOR, 0); //stop the rplidar motor

    // try to detect RPLIDAR...
    rplidar_response_device_info_t info;
    if (IS_OK(lidar.getDeviceInfo(info, 100))) {
       //detected...
       lidar.startScan();
       analogWrite(RPLIDAR_MOTOR, 255);
       delay(1000);
    }
  }
}
```

With the code up and running, we ran into some issues. First, the H-Bridge was not responding to any of the commands via bluetooth. Then, the RGB LED light kept flickering, and the LCD screen was only displaying units at one direction, when it's suppose to display a full cycle.

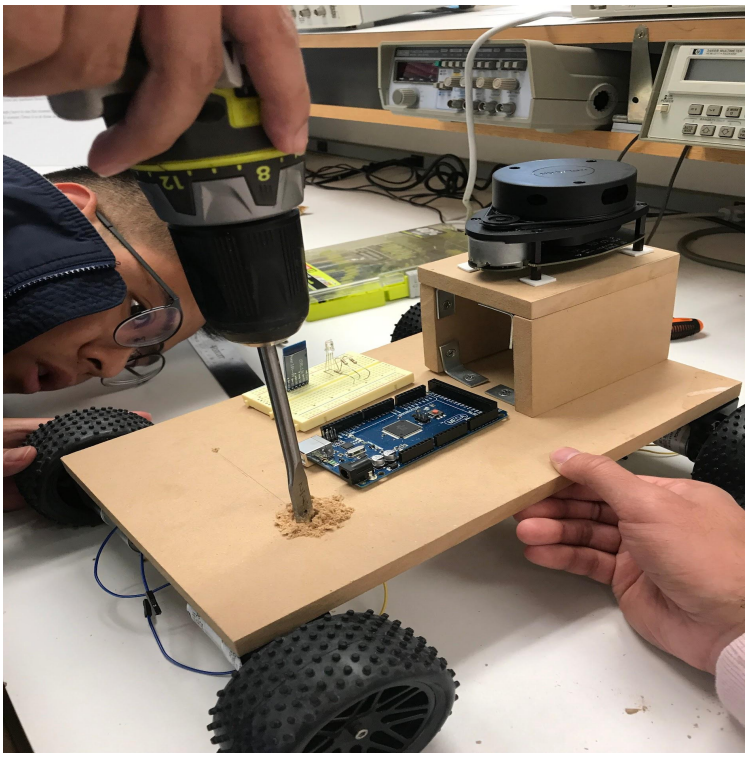
Problem Resolved: For the H-Bridge issue, we realized our input wiring was wrong, so we set the wiring into the correct format, thus all the motors were spinning in the right direction. Next, with the RGB LED light issue, we fixed the issue by simply moving the LCD code in a different loop, since it was interfering with the LIDAR's scanning process. In addition, we noticed that displaying the angle would not work simultaneously when the distance is being read. To fix it, we needed to remove it so the LED would not flicker as well. As a result, we fix both the H-bridge, RGB LED light, and LCD screen display issue by reorganizing our code, so we can get the correct measurements.

Progression 5: Mounting the RC Car Together



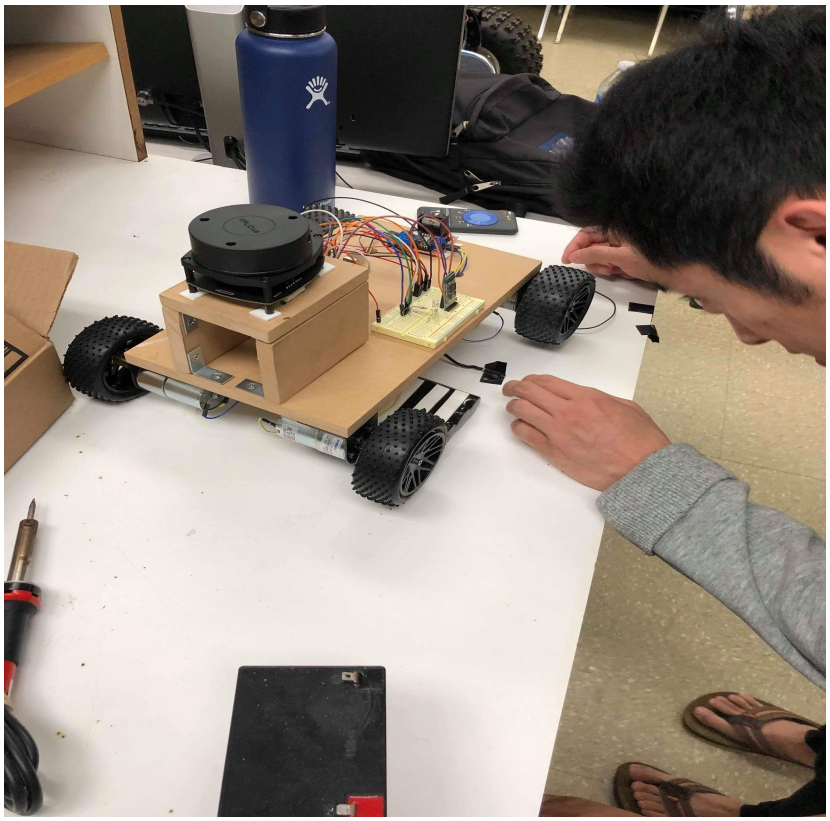
- With the holes and wire management figured out, more items are needed incorporated on the chassis
- Different resistor values need to be used as a pull up resistor to draw less current towards LCD and response time from LIDAR to screen and LIDAR to RGB
- For a better response time, reconfiguring the code has be done
- Duct tape, zip ties, and possibly super glue were in mind since the 12V battery is too heavy to be mounted on the bottom of the wood frame

Progression 6: Wiring the Motors



- First, we need to drill holes for the wires from the H bridge, so it can reach the 12V DC motors that we mounted under the chassis
- Next, the wires are reconfigured to reduce the risk of any wires hanging and intertwining with the motors when the car is in motion
- Lastly, we decided to utilize double sided tape to stick the 12V battery under the chassis, but the battery turned out to be too heavy and didn't manage to hold in place. The conclusion was to superglue the battery towards the wood frame

Progression 7: Wire Management and Glueing the Battery

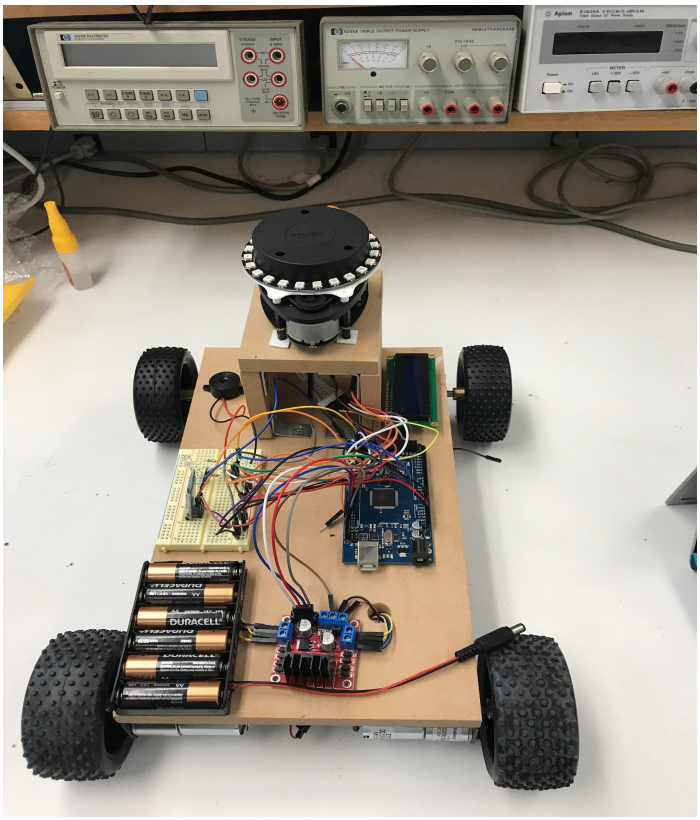


Our next step is to figure out the wiring management on the bottom of the board, since we do not want the wheels to get caught during the running process. As for the battery, we used gorilla glue to withstand the weight. Based on our results, we successfully glued the battery on the bottom and managed all the wires with electrical tape to prevent the wiring from touching the motors from spinning.

Progression 8: Implementing a Buzzer and an LED Light Ring

To compensate for the angle not displaying on the LCD screen, we bought the LED light ring to visualize where the direction is coming from the LED. In addition, we wanted to include an electronic buzzer to detect a close object nearby in a certain distance. Again, we now need to start researching and develop a brand new code, so that we can implement it with our current design.

Progression 9: Mounting the LED Light Ring and Buzzer



With mounting the LED light ring and the electronic buzzer successfully, we ran into some fitment issues with the LED ring. Inside the inner ring, there were two tiny bumps on each side. To fix it, we used sandpaper to smooth out the two bumps so it can fit through the Lidar without leaving any scratches on it. In addition, we 3-D printed a mounting bracket for the LED light ring so the ring wouldn't interfere with the rotor on the bottom of the Lidar as the device is continuously spinning. Finally, we superglued both the buzzer and the LED light ring to the bracket without any fitment issues for a smooth rotation.

Progression 10: Finalizing The Code

```
// This sketch code is based on the RPLIDAR driver library provided by RoboPeak
#include <RPLidar.h>
#include <Wire.h>
#include <NeoPixelBus.h>
#include <LiquidCrystal_I2C.h>   //include liquid crystal library to read and write the LCD
#include <SoftwareSerial.h>      //include softwareserial library to define multiple serial ports
LiquidCrystal_I2C lcd(0x27,16,2); //set the LCD Screen
// You need to create an driver instance
RPLidar lidar;

// Change the pin mapping based on your needs.
////////////////////////////////////////////////////////////////////////////////
#define LED_ENABLE  24 // The GPIO pin for the RGB led's common lead.
                       // assumes a common positive type LED is used
#define LED_R       7  // The PWM pin for drive the Red LED
#define LED_G       6 // The PWM pin for drive the Green LED
#define LED_B       5 // The PWM pin for drive the Blue LED

#define RPLIDAR_MOTOR 3 // The PWM pin for control the speed of RPLIDAR's motor.
                        // This pin should connected with the RPLIDAR's MOTOCTRL signal
////////////////////////////////////////////////////////////////////////////////

const uint16_t PixelCount = 24; // this example assumes 4 pixels, making it smaller will cause a failure
const uint8_t PixelPin = 8;  // make sure to set this to the correct pin, ignored for Esp8266
const int buzzer = 25;

NeoPixelBus<NeoRgbFeature, Neo800KbpsMethod> strip(PixelCount, PixelPin);

#define colorSaturation 64
RgbColor red(colorSaturation, 0, 0);
RgbColor green(0, colorSaturation, 0);
RgbColor blue(0, 0, colorSaturation);
RgbColor white(colorSaturation);
RgbColor black(0);




void blackStrip() {
  for (int i = 0; i < PixelCount; i++) {
    // strip.SetPixelColor(i, leds[i]);
    strip.SetPixelColor(i, black);
    // leds[i] = black(0);
  }
}

void pointangle(int angle) {
  // blackStrip();
  strip.SetPixelColor(map(angle, 0, 360, 0, 23), green);
  strip.Show();

}
#define NEO_RGBSPACE_MAX (byte)(200L*255/360)
int _r, _g, _b;

//Set current RGB with the hue: HSV(hue, x, x)
void hue_to_rgb( _u8 hue)
{
/*`
    Hue(in Degree):  0 (RED) ----> 60 (Yello) ----> 120 (Green) --..... ----> 360
    Hue'(fit to _u8):0       ----> 60/360*255 ----> 120/260*255 --..... ----> 255
*/

    //convert HSV(hue,1,1) color space to RGB space
    if (hue < 120L*255/360)   //R->G
    {
        _g = hue;
        _r = NEO_RGBSPACE_MAX - hue;
        _b = 0;
    }else if (hue < 240L*255/360)  //G->B
    {
        hue -= 120L*255/360;
        _b = hue;
        _g = NEO_RGBSPACE_MAX - hue;
```

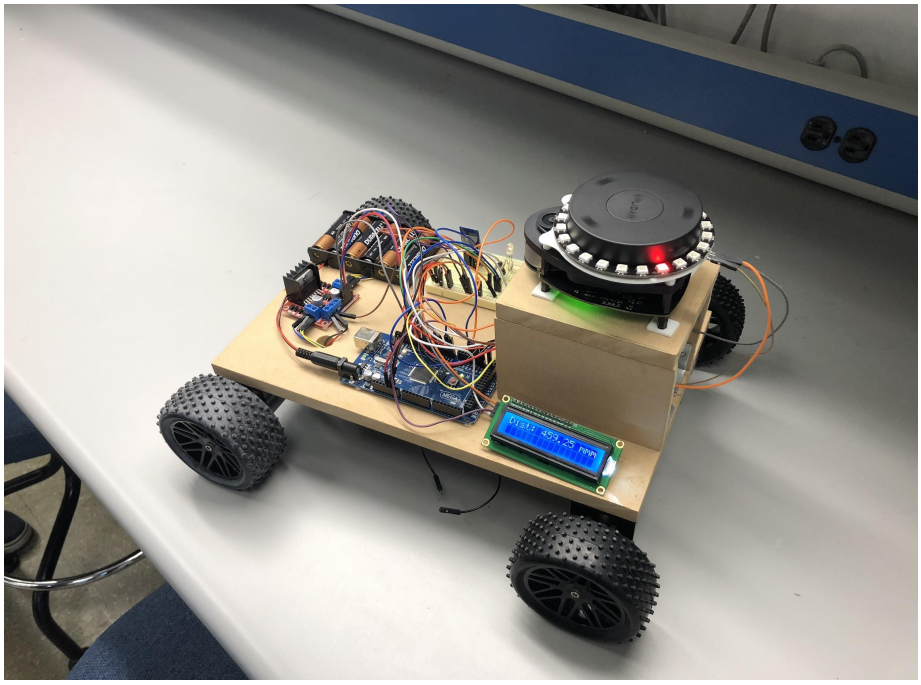
```
        _r = 0;
    }else //B->R
    {
        hue -= 240L*255/360;
        _r = hue;
        _b = NEO_RGBSPACE_MAX - _r;
        _g = 0;

    }
}

void displayColor(float angle, float distance)
{
    //1. map 0-350 deg to 0-255
    byte hue = angle*255/360;
    hue_to_rgb(hue);

    //2. control the light

    int lightFactor = (distance>500.0)?0:(255-distance*255/500);
    _r *=lightFactor;
    _g *=lightFactor;
    _b *=lightFactor;

    _r /= 255;
    _g /= 255;
    _b /= 255;

    analogWrite(LED_R, 255-_r);
    analogWrite(LED_G, 255-_g);
    analogWrite(LED_B, 255-_b);

    lcd.setCursor(0,0);  //set the position on the LCD to the top left
    lcd.print("Dist: ");  //print the word Dist
    lcd.print(distance);  //print the actual value of distance
    lcd.print(" mm");  //print the units




    if (distance < 150) {
      digitalWrite(buzzer, HIGH);
    }
    else {
      digitalWrite(buzzer,LOW);
    }
}


char t;
void setup() {
  // bind the RPLIDAR driver to the arduino hardware serial
pinMode(13,OUTPUT);   //left motors forward
pinMode(12,OUTPUT);   //left motors reverse
pinMode(11,OUTPUT);   //right motors forward
pinMode(10,OUTPUT);   //right motors reverse
  strip.Begin();
  blackStrip();
  strip.Show();
  lidar.begin(Serial);
  Serial1.begin(9600);
  lcd.init();


  lcd.backlight();
  lcd.blink_on();

  // set pin modes
  pinMode(RPLIDAR_MOTOR, OUTPUT);

  pinMode(LED_ENABLE, OUTPUT);
  pinMode(LED_R, OUTPUT);
  pinMode(LED_G, OUTPUT);
  pinMode(LED_B, OUTPUT);

  digitalWrite(LED_ENABLE, HIGH);
```

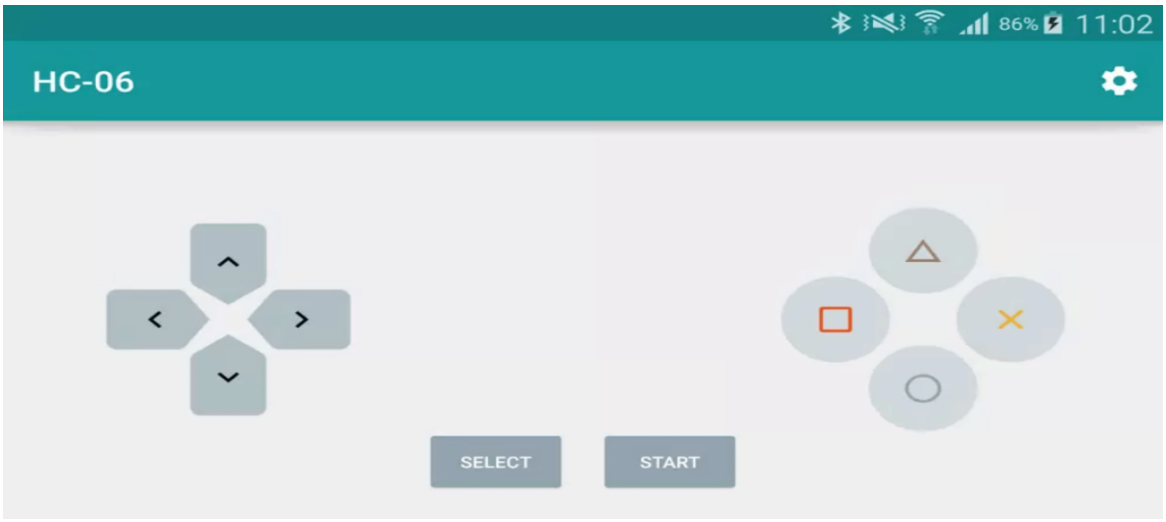
```
  analogWrite(LED_R,255);
  analogWrite(LED_G,255);
  analogWrite(LED_B,255);
}

float minDistance = 100000;
float angleAtMinDist = 360;
int nsbit = 0;
void loop() {
  if (IS_OK(lidar.waitPoint())) {
    //perform data processing here...
    float distance = lidar.getCurrentPoint().distance;
    float angle = lidar.getCurrentPoint().angle;

    if (lidar.getCurrentPoint().startBit) {
      nsbit++;

    }
    if (nsbit > 2) {
      // a new scan, display the previous data...
       displayColor(angleAtMinDist, minDistance);
      blackStrip();
      pointangle(angleAtMinDist);
      minDistance = 100000;
      angleAtMinDist = 0;
      nsbit = 0;

    } else {
      if ( distance > 0 &&  distance < minDistance) {
        minDistance = distance;
        angleAtMinDist = angle;
      }
    }
  } else {
    analogWrite(RPLIDAR_MOTOR, 0); //stop the rplidar motor




    // try to detect RPLIDAR...
    rplidar_response_device_info_t info;
    if (IS_OK(lidar.getDeviceInfo(info, 100))) {
      //detected...
      lidar.startScan();
      analogWrite(RPLIDAR_MOTOR, 255);
      //analogWrite(RPLIDAR_MOTOR, analogRead(A0));
      delay(1000);
    }
  }
if(Serial1.available()){
  t = Serial1.read();
  Serial1.println(t);
}

if(t == '1'){            //move forward(all motors rotate in forward direction)
  digitalWrite(13,LOW);
  analogWrite(12,120);
  analogWrite(11,120);
  digitalWrite(10,LOW);
}

else if(t == '2'){      //move reverse (all motors rotate in reverse direction)
  analogWrite(13,120);
  digitalWrite(12,LOW);
  digitalWrite(11,LOW);
  analogWrite(10,120);
}

else if(t == '3'){      //turn right (left side motors rotate in forward direction, right side motors doesn't rotate)
  digitalWrite(13,LOW);
  digitalWrite(12,LOW);
  digitalWrite(11,LOW);
  analogWrite(10,255);
}
```

```
else if(t == '4'){      //turn left (right side motors rotate in forward direction, left side motors doesn't rotate)
  analogWrite(13,255);
  digitalWrite(12,LOW);
  digitalWrite(11,LOW);
  digitalWrite(10,LOW);
}

else if(t == '5'){      //STOP (all motors stop)
  digitalWrite(13,LOW);
  digitalWrite(12,LOW);
  digitalWrite(11,LOW);
  digitalWrite(10,LOW);
}

}
```

For the coding process, we needed to add new libraries for the LED light ring in order for the complete design to operate. First, we needed to verify if the LED light ring itself was working without our old design code. Then, we needed to verify if the units were working properly as a directional sensor. Next, we implemented an if-else statement for the buzzer, stating that if any distance less than 150 millimeters, the buzzer will turn on, otherwise the buzzer will turn off. Finally, we incorporated both the buzzer and the LED ring into our code, giving us our final and finished product.

Progression 11: Designing and Testing Finalization





For the bluetooth controller setup, we used that specific template you see above called *Arduino Bluetooth Controller* which is downloadable application for android users only. When enabling the bluetooth to our Arduino, we've successfully connected and verified that the motors spin on all four corners. However, we felt like car was moving rather quick, and it was a risk that the LIDAR might not work properly if the car is going too fast as it is taking distance measurements as well as directional processing. So, in order to fix it, we slowed down the duty cycle for the motors to around 50% so that the car will go in a steady speed running at about 100 RPM. As result, this is our final product and our complete design for our project, and that we ran into zero issues when the car was mobile and when the lidar was detecting objects around its peripheral vision.

PERFORMANCE ANALYSIS

The RC LIDAR Detection Vehicle had some interesting performance when we finished designing our project. First was the response time on the LIDAR. Although the measurements are accurate, there is a slight quarter to half second delay when scanning a detected object. Even when we moved the delay functions on our code, it seemed that the LIDAR wasn't capable or performing well when it was being used by an RGB LED light to display the strength of the sensor. A LED light ring is incorporated into our system to display the direction of where the

object is coming from, and the LCD screen display to show the distance and how far the nearest object is from the vehicle in motion.

For the DC motors, the 12V supply worked perfectly with our H-bridge setup, having no lag whatsoever, and taking into account current drawn through our entire system. Likewise, our bluetooth module seemed to be working perfectly with our tablet when we control the car. There seem to be no input lag as well from the tablet to the drivers as our HC-05 bluetooth module is paired on the breadboard.

Finally, for the battery performance, the 12V rechargeable battery provided by Battery Space worked smoothly, without any voltage spikes as well as our double AA 9V battery setup for the Arduino towards the LIDAR system. In the future, we definitely want a bigger capacitance for the Arduino supply, since its feeding a lot of current to all the components for this project.

DISCUSSION AND CONCLUSION

The RC LIDAR Detection Vehicle meets our objective of measuring distance as well as detecting objects through components such as an LED light ring to display directional movement, an LED Screen to display distance, and our design meets requirements of nominal/maximum voltage and current. We have a series of RGB LED lights to signify the strength through color projections, as well as a buzzer to avoid accident collisions and possible detrimental accidents for cars in motion. Throughout the whole process, we definitely went through some bumps in the road, from the H-bridges not working properly, to adding more components to our design to make it more aesthetically pleasing. To reiterate, the main purpose of this project was to see how useful a LIDAR detection device can be, from determining the objects around its perimeter, to detecting if the object is too close for the car, to warning the operator that the object is too close. In a practical standpoint, this is how modern technologies works on every modern vehicle, with a series of sensors ranging from radar, lidar, to sonar. From blind spot monitoring to collision sensors, this project was designed to showcase how modern cars function in the real world. Hopefully in our future endeavors, we would get to design a car being driven autonomously from point A to point B without controlling it manually.

REFERENCES

"DF Robot LIDAR Sensors – Getting Started with LIDAR." *DroneBot Workshop*, 23 July 2018, dronebotworkshop.com/getting-started-with-lidar/.

Huang, Tony. "RPLIDAR-A1 360°Laser Range Scanner _ Domestic Laser Range Scanner." *SLAMTEC*, www.slamtec.com/en/lidar/a1.

"RPLidar." *Ros.org*, wiki.ros.org/rplidar.

Instructables. "How to Use the RPLIDAR 360° Laser Scanner With Arduino." *Instructables*, Instructables, 30 Oct. 2018, www.instructables.com/id/How-to-Use-the-RPLIDAR-360-Laser-Scanner-With-Ardu/.

"What Is Lidar and What Is It Used for?" *American Geosciences Institute*, 13 June 2018, www.americangeosciences.org/critical-issues/faq/what-lidar-and-what-it-used.

Admin. "100 Applications or Uses of LiDAR Technology." *LIDAR and RADAR Information*, 30 Dec. 2017, lidarradar.com/apps/100-applications-or-uses-of-lidar-technology.

"Help." *Advantages of Using Lidar in GIS-Help | ArcGIS for Desktop*, desktop.arcgis.com/en/arcmap/10.3/manage-data/las-dataset/advantages-of-using-lidar-in-gis.htm.