Fall 2017 - CSE 325 Project 7
LIDAR Obstacle Avoidance Navigation

Deadlines

When to submit the files of this project on blackboard:

Friday, Nov 3 2017, 6:00pm, AZ time.

When/where to demo this project:

Friday Nov 3, 2017, 6:00 pm AZ Time

The lawns in front of Old Main which is at 425 E University Dr, Tempe, AZ, GPS coordinates: 85281, 33.421164, -111.934012

Project Description

The purpose of this project is to introduce the RPLIDAR module to you. The LIDAR needs to be mounted on the top plate and wired up to one of the serial port of the Arduino Nano. Afterwards, your goal is to navigate and avoid the obstacles at the same time.

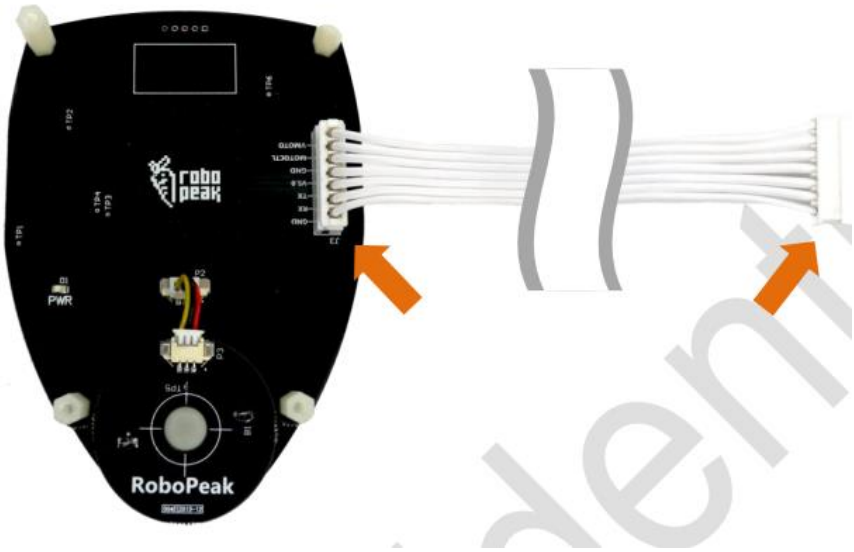
Required Hardware:

- Arduino Mega 2560 and cable.
- Arduino Nano and cable
- Fully built robotic car from previous project
- RPLIDAR

Part 1: Unpacking the RPLIDAR

The LIDAR package includes the following parts:

There is a bag with 4 small screws in the box. DO NOT LOSE THEM!



Flip the LIDAR and plug in the RPLIDAR communication cable. Don't plug anything into the other side.

Part 2: Mounting the LIDAR

Mark four points on the black top plate according to LIDAR dimension and drill four holes. Using the included screws, gently install the RPLIDAR on the plate. Make sure the cable is attached and accessible so that you can connect it to Arduino Nano.

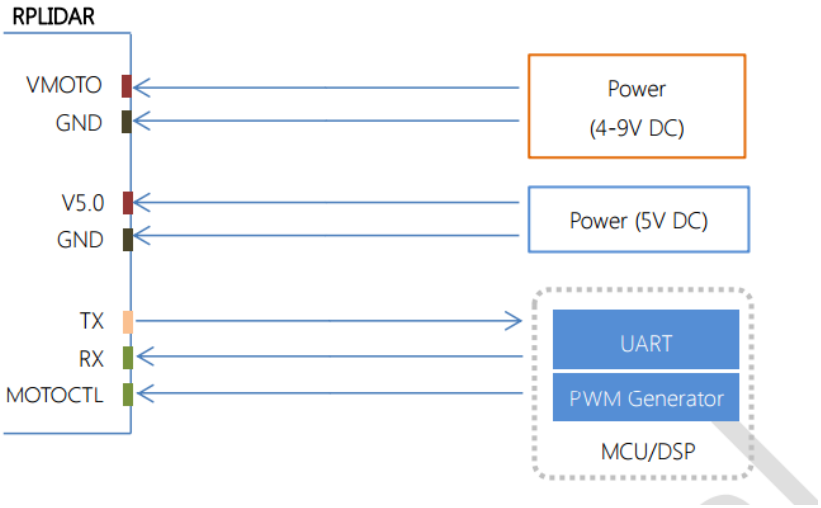
Part 3: Wiring the RPLIDAR

Use the manual to wire up the RPLIDAR. It's better to wire up the LIDAR before mounting it. You can use jumper wire directly into the white RPLIDAR cable. The following figure shows the wiring.

ATTENTION: VMOTO and V5.0 should be connected to the 5V from voltage converter, not the 5V from Arduino board!

TX pin of the LIDAR should be connected to RX pin of the Arduino and RX pin of the LIDAR should be connected to the TX pin of the Arduino.

The MOTOCTL is the motor speed control pin and should be connected to one of the Arduino pins with PWM output.



Part 4: Spring Spacers

You might have noticed that mounting the LIDAR on the front side of the car reduces the clearance of the suspension on the front wheels. In order to fix this, you can add spacing parts to the suspension

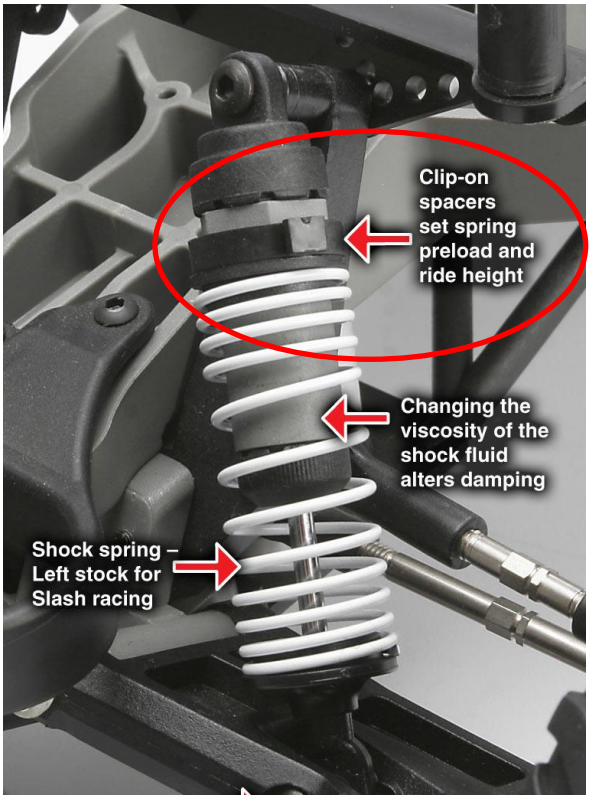
system what are known as “spring spacers” so that the LIDAR does not detect the ground as an obstacle. If they have not been installed previously, ask TAs to give it to you.

Extra spring boosters should look like this:

You need to detach the 2 largest ones as well as the 2 second largest ones. At the end, you should have 1 large and 1 second largest spring boosters on each shock.

See the following picture to know how spacers are placed on the struts. You may have noticed the rear struts of the car have already 1 large booster on each. You are free to add and remove these

boosters as you like. However, we suggest using 1 of the largest and 1 of the second largest boosters per strut for optimal performance.



Part 5: Co-Processor Setup:

Because the LIDAR requires a high rate of monitoring, you are going to use the Arduino Nano as a co-processor to read data from LIDAR. The objective of Arduino Nano is reading data from LIDAR, filter and parse it, and, send it to the main processor (Arduino Mega) upon request. The programming of this project is divided into three parts: First, you should program the Arduino Nano to read, filter and parse the LIDAR data. The Nano should be connected to the LIDAR by serial port (TX, RX). The second part is the communication between the Nano and the Mega. They will be connected via I^2C communication, where the Arduino Nano is slave, and the Arduino Mega is master in the communication. The third part is programming the Arduino Mega 2560 which receives the LIDAR data from the Nano via the I^2C ports, heading data from IMU and positioning data from GPS. The

Arduino Mega then determines how to navigate and avoid obstacles at the same time. Please make sure you follow these 4 guidelines:

1- Get the correct library:
   - Download RPLidar library from here:
     i. Go to https://github.com/robopeak/rplidar_arduino
     ii. Click the green button (clone or download) and download the ZIP file.
     iii. Install the library using the Arduino IDE by going to Sketch->include Library-> Add .ZIP library
     iv. If you have any problem installing the library, unzip all the file and zip the folder that contains rplidar.cpp and header file and attempt to install again.
     v. If problem persists, ask TAs to help you.
2- Nano programming:
   - You need to disconnect TX and RX pins which are connected to RPLIDAR before uploading the code on Arduino Nano, otherwise, it will not upload the new program.
   - Reconnect TX and RX pins after uploading.
3- Acquiring GPS data in Degree
   - Make sure you use GPS.latitudeDegrees and GPS.longitudeDegrees to read GPS data in decimal degrees.
4- Communication with Arduino Nano
   - Connect Nano I^2C pins (A4 is SDA and A5 is SCL) to the existing I^2C bus which was used for connecting Mega to IMU (BNO055 sensor). The IMU and Arduino Nano are now sharing SDA and SCL bus with each other and they are both slaves in the communication and the Arduino Mega is the master in the communication.
   - Test the communication by writing a simple program for both Arduino Mega and Nano. You can find good material here:
     https://www.arduino.cc/en/Tutorial/MasterWriter
     https://www.arduino.cc/en/Tutorial/MasterReader

Part 6: Car Navigation

In order to save the destination position, the user will place the car on the ground and wait for it to get a GPS fix position. Once it has acquired a fix position, the user would press the select button on the keypad to indicate this is the desired destination. By pressing the select button, the Arduino Mega should read the GPS coordinates of the location and store it. Then, the user will take the car to an arbitrary start point, place the car on the ground facing any arbitrary direction, and press the up

button to start driving. The car should then navigate and reach within 5 meters of the destination and then stop. The car should avoid any obstacle on its way to destination.

There are a couple of guidelines to keep in mind for implementation. First, it is assumed there will be no tall buildings in the testing area. Second, there will be some detectable obstacles of the field. These obstacles will be tall enough for your robot to see them with the LIDAR. Since the test would be taken in front of Old Main building, the car will be driving on grass and the trees in that area are potential obstacles. Third, the robot may be pointed towards any arbitrary direction at start point.

Hints:

Since communication with LIDAR and monitoring the data on serial port is not possible for Arduino Nano, you can use Arduino Mega to check if your program (the on which will be uploaded on Nano) outputs the right data.

Keep in mind, as you turn, the obstacles will move from the LIDAR perspective, so the program should be dynamic.

The LIDAR has problem detecting darker objects. Textures like leather are almost invisible from LIDAR perspective, so test it with lighter objects.

Submission files:

You have to submit following files:

1. Mega.ino (Arduino Mega program)
2. Nano.ino (Arduino Nano program)
3. Integrity.txt
4. Group's_members.txt

All files should be zipped into a single file named CSE325_Fall2017_Project7.zip and be uploaded to the blackboard before the due date. You should submit any library code you may have written for this project. All necessary files need to be zipped into a single file called project6.zip.

Submit on time. Late submissions will be awarded 0 points.

How the project is graded:

Grading of the project is done by TAs. On the demo day, the TA will take you to the destination, and ask you to save the location by pressing the select button on the keypad. Then, the TA will ask you to take the car to a starting point with an arbitrary heading. The car should return to saved destination point and stop within $5m$ of it. The car should drive smoothly and avoid any obstacles on its way. You may perform 3 different demos with different paths. Grades will be best 2 out of 3 tests (we ignore the lowest score).

Grading Rubric

The Grading rubric is shown in the table below

Points	Description
5	Car drives up to the halfway
-5	Car hits any obstacle
3	Car reaches within 5 m of destination and avoid any obstacle in less than 5 minutes. (The destination will be less than 50 m away).
2	The communication between Mega and Nano is implemented well
-5	If the Arduino Mega implementation is polling based instead of interrupt based
	If you do not upload your code up to the deadline, you will be graded 0.