FALL 2016 CSE 325 Project 6
LIDAR Obstacle Avoidance Navigation

Deadlines

When to submit the files of this project on blackboard: Thursday, Oct 27, 2016, 23:59pm, AZ time.

When/where to demo this project: Friday Oct 28, 2016, 6:00 pm AZ Time at a space in front of Old Main which is in 425 E University Dr, Tempe, AZ 85281, 33.421164, -111.934012

Project Description

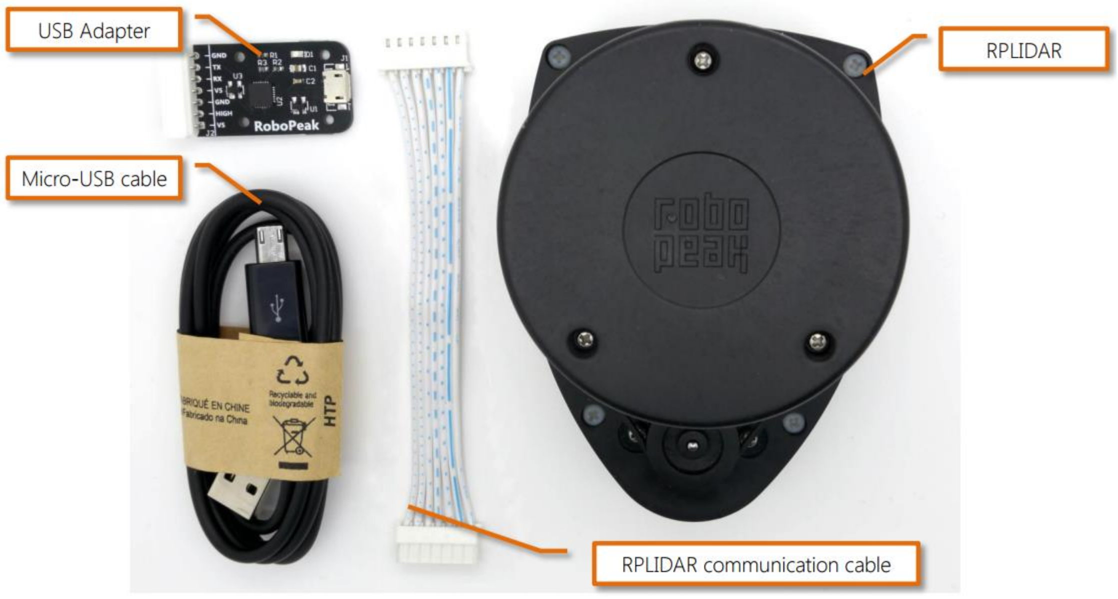
The purpose of project 6 is to introduce you to the RP LIDAR module. The sensor will need to be mounted into the top plate and wired up. Afterwards you can perform the same goal as project 5, however this time avoiding obstacles along the way.

Hardware:

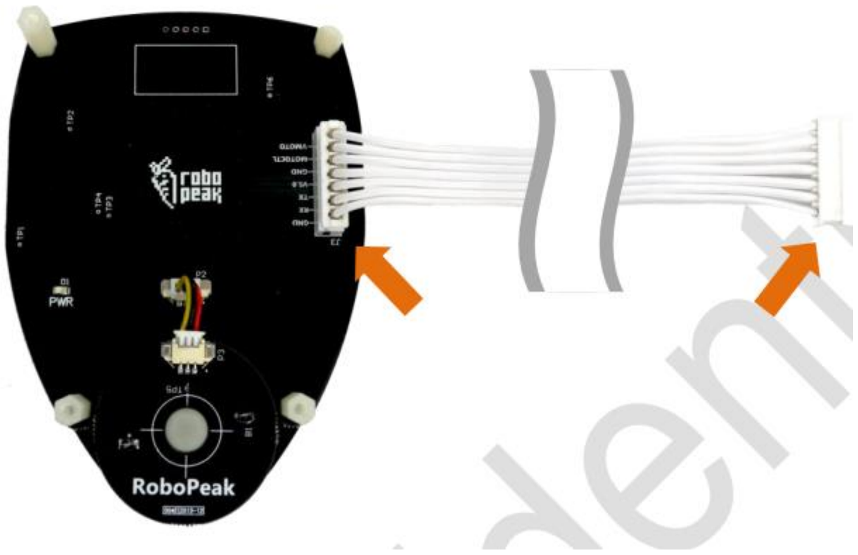
- Arduino Mega 2560 and cable.
- Arduino nano and cable
- Fully built robotic car from project 5.
- RPLIDAR

Part 1: Unpacking the RPLIDAR

Be careful as you unpack the LIDAR from the box. It should include the following parts:



Included with the box should be a bag of 4 small screws. DO NOT LOSE THESE!

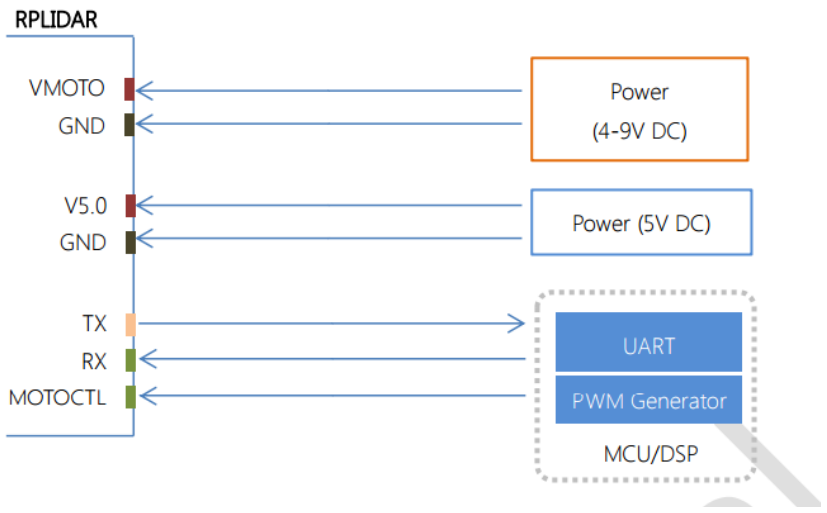


Flip the LIDAR over and plug in the RPLIDAR communication cable. Don't plug anything into the other side.

Part 2: Screwing in the RPLIDAR

Using the 4 included screws, gently screw in the RPLIDAR. Make sure that the cable stays attached and accessible so you can wire into it later. It may be helpful to perform step 4 before screwing the LIDAR into the plate.

Part 3: Wiring the RPLIDAR

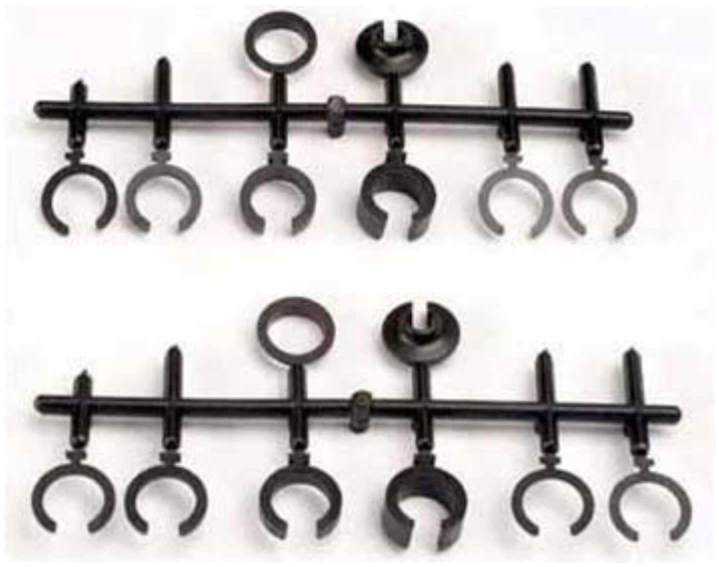


Use the manual to wire up the RPLIDAR. It may be helpful to wire it up before screwing it in. You can use jumper pins directly into the white RPLIDAR cable.

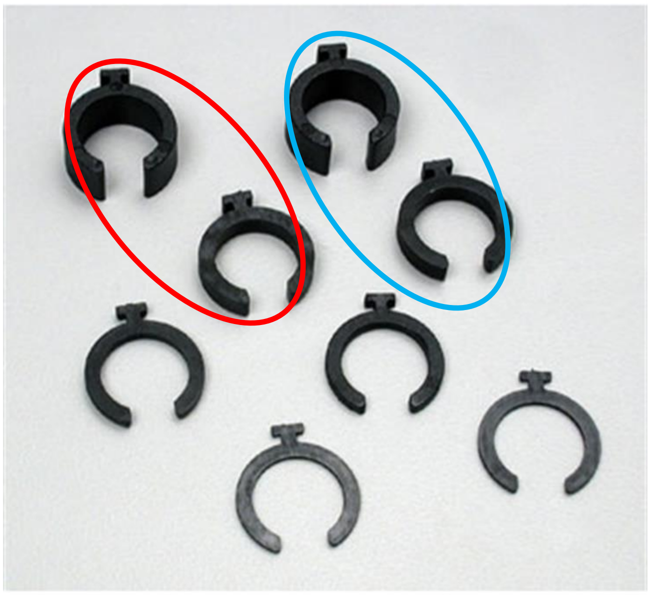
Part 4: Spring Spacers

You may have noticed that putting the LIDAR on the front of the robot reduced the ground clearance of the suspension on the front wheels of the car. In order to fix this, we need to add what are known as “spring spacers” to the front suspension and rear suspension so we do not bottom out while driving on rough terrain. If they have not been installed previously, TA’s will give you them to add to the suspensions.

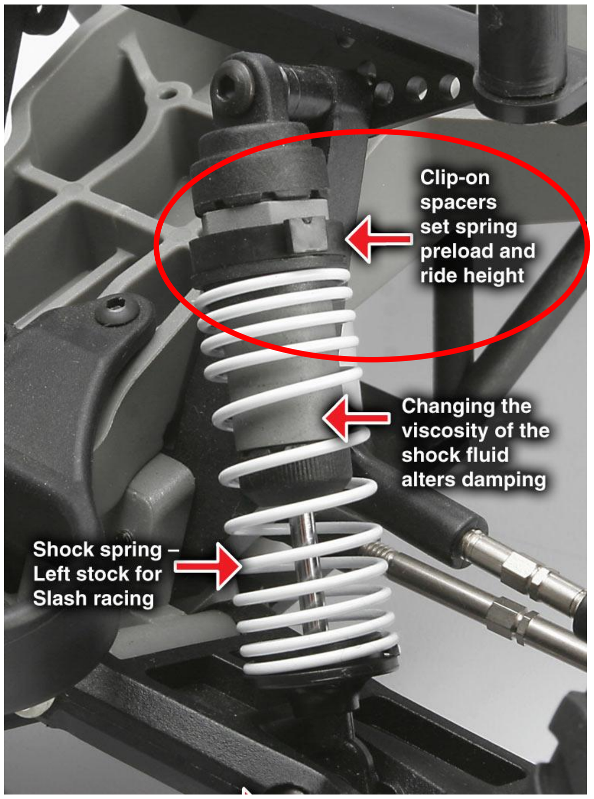
Included with the car was a set of extra spring boosters that should look something like this:



We need to break off the 2 largest ones as well as the 2 second largest ones. In the end we want to have 1 large and 1 of the second largest spring boosters on each shock tower.



See the following picture for how they are placed on the struts. You may notice that the back struts of the car already have 1 large booster on each. You are free to add and remove these boosters as you please to get better performance, however we suggest using 1 of the largest and 1 of the second largest boosters per strut for optimal performance.


Clip-on spacers set spring preload and ride height
Changing the viscosity of the shock fluid alters damping
Shock spring – Left stock for Slash racing

Part 5: Co-Processor Setup:

Because the LIDAR requires constant monitoring, in this project you are going to use the Arduino Nano as a co-processor. The Nano will read the data, parse it, and send it to the main processor the Arduino Mega upon request. The programming of this project is divided into three parts: First there is programing the Arduino Nano to read and parse the LIDAR data. The Nano will be connected to the LIDAR by serial port (Tx, Rx). It will gather the data from LIDAR. The second part is the communication between the Nano and the Mega. They will be connected via I^2C protocol, where the Nano is the slave, and the Mega is the master. The third part is the Arduino Mega 2560 which receives the LIDAR data from the Nano via the I^2C ports, and determine what that data means when navigating. Please be sure you follow these 4 guidelines carefully:

1- Get the correct library:
- Download RPLidar library from here:
    1. Go to https://github.com/robopeak/rplidar_arduino
    2. Click the green button (clone or download) and download the ZIP file.
    3. If you have any problem, unzip all and zip the folder that contains rplidar.cpp and header file.

2- Nano programming:
   - You need to disconnect TX and RX connected to RPLIDAR before programming otherwise it will not accept the new program! Reconnect them afterwards for testing.

3- GPS 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) SDA and (A5) SCL) to existing I^2C bus which is used for connecting Mega to IMU. (IMU and Nano are now sharing SDA and SCL bus with each other.) The Arduino Nano needs to be programmed as the slave device. The Arduino Mega must be the master. If this is not done, your IMU data is not going to work as it will be sharing the I2C bus with the Nano.

Part 6: Car Navigation

In the final part of this project, you will program the car to navigate from a start point to an end point using GPS, IMU and LIDAR to avoid obstacles. In order to initialize the program, the user will place the car down in the desired end location and wait for it to get a GPS fix. Once it has acquired a fix, the user would press the select button on the keypad to indicate this is the desired finish location. The car should read the GPS coordinates of the current location, store it and wait. The user would then walk to the desired start location, place the car on the ground (facing any direction), and press the up button to initialize the program. The car will then navigate back to the GPS coordinates it took in when the select button was pressed, which should be within 5 meters of the desired location. In the meantime, if any obstacle is encountered on the way, the car will avoid it. The car will then stop within the 5-meter circle of the finish point.

There are a couple of guidelines to keep in mind for this scenario. First, it is assumed there will be no tall buildings in the area of the robot that will interfere with the GPS. Second, you can assume there will be some major obstacles in between the starting and ending points of the navigation path. These obstacles will be tall enough and colored correctly 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 will be potential obstacles. Third, the robot may be pointed in any direction when it is set down at the start location, as well as the end location.

Hint: This project is the same as last project, except for the LIDAR. So just scavenge your code from P6 and add the LIDAR part to it. Keep in mind as you turn the robot, the obstacles will move in the LIDAR view. This may end up causing problems when trying to avoid incoming obstacles. The LIDAR also has problems seeing darker objects. Things like leather jackets are almost completely invisible. It is useful to connect the LIDAR to a PC and run the LIDAR data visualization program to wrap your head around how the LIDAR data looks.

Submission:

You have to submit 5 files

1. Mega.ino (This file is prepared to upload on the Arduino Mega)
2. Nano.ino (This file is prepared to upload on the Arduino nano)
3. Members.txt
4. Integrity.txt

Students will submit the Arduino codes and any library code you may have written for the GPS navigation project in project 5. All necessary codes need to be zipped into a file called project6.zip.

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

How project will be graded:

Grading for this project will be done through a demonstration to the TAs. On the demo day, the TA will bring you to the destination, and you should press the select button on the keypad to register the destination. Then the TA will ask you to take the car to the starting point, and then from there, the car must return to within $5m$ of the destination point, driving autonomously. The TA may set the orientation of the car at the start point. You will perform 3 demos with different TAs. Grades will be best 2/3 tests (we drop the lowest score of the 3 courses you perform). Grading scale is shown in the table below.

Points	**Description**
5	Car moves and goes up to halfway to the destination.
-5	Car hits any obstacle
3	Car reaches the obstacle, and reaches within 5 m of destination. The car should reach the destination and stop within 5 minutes of start. The destination will be less than 50 m away.
2	The communication between Mega and Nano is implemented well