Fall 2017 CSE 325 Project 5
GPS Navigation

Deadlines

When to submit the files of this project on blackboard:

Friday, Oct 6, 2017, 6:00pm, AZ time.

When/where to demo this project:

Friday Oct 6, 2017, 6:00 pm AZ Time

At the lawn in front of Old Main which is located on 425 E University Dr, Tempe, AZ

33.421164, -111.934012

Project Description

The purpose of this project is to become familiar to the GPS module. First, you should solder the pin header to the sensor. You should acquire position data from GPS and heading data from IMU sensor. Using these data, you should be able to drive toward specific destination.

Required Hardware:

- Arduino Mega 2560 and cable.
- Fully built robotic car from last project
- LCD + keypad
- BNO055 IMU sensor
- GPS Module.

Part 1: Plugging in the Keyboard (If is not already done)

The first step is to mount the keypad on the Arduino Mega similar to previous project.

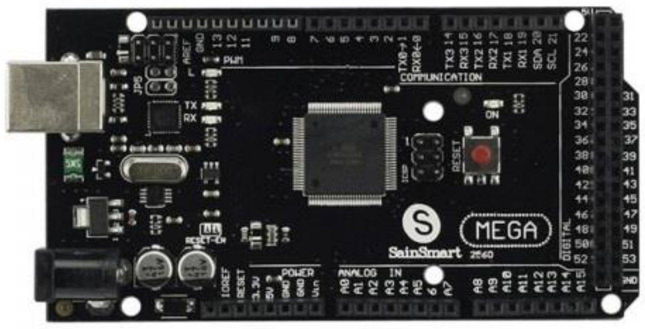


Figure 1 : Arduino Mega Top-Down

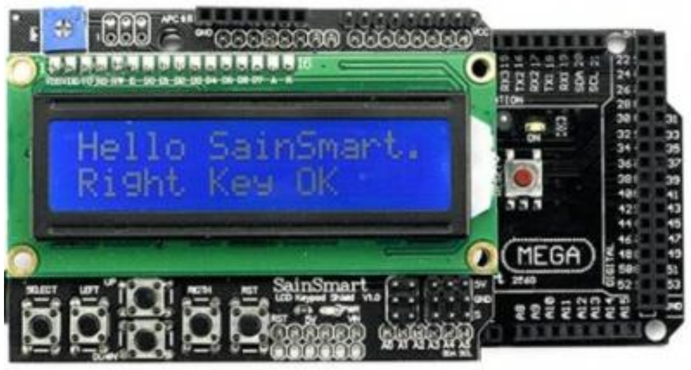


Figure 2 : Arduino Mega With Keypad Top Down

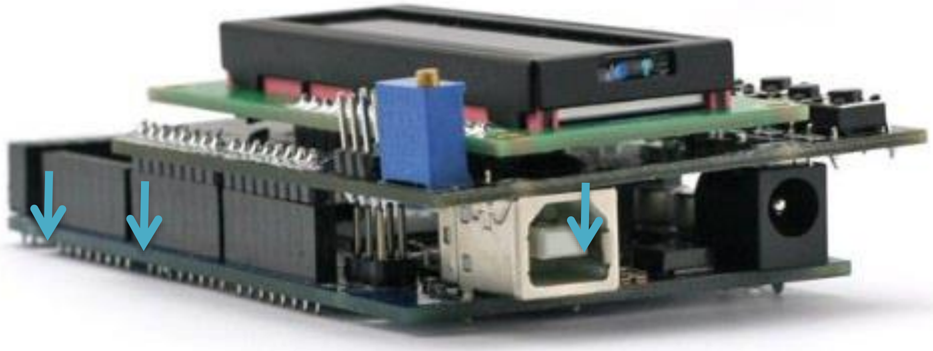


Figure 3 : Side View of Completed Assembly. The 2 pin headers on this side should match up.

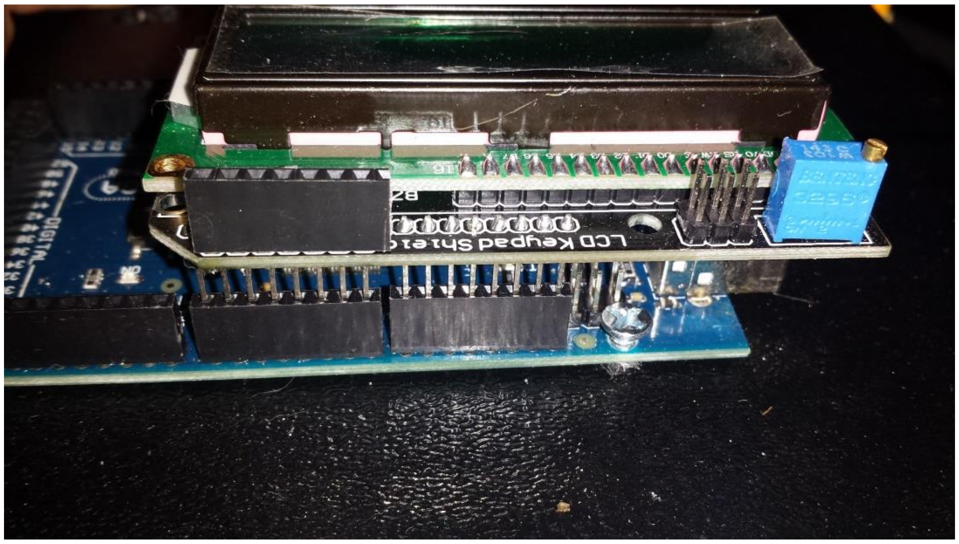
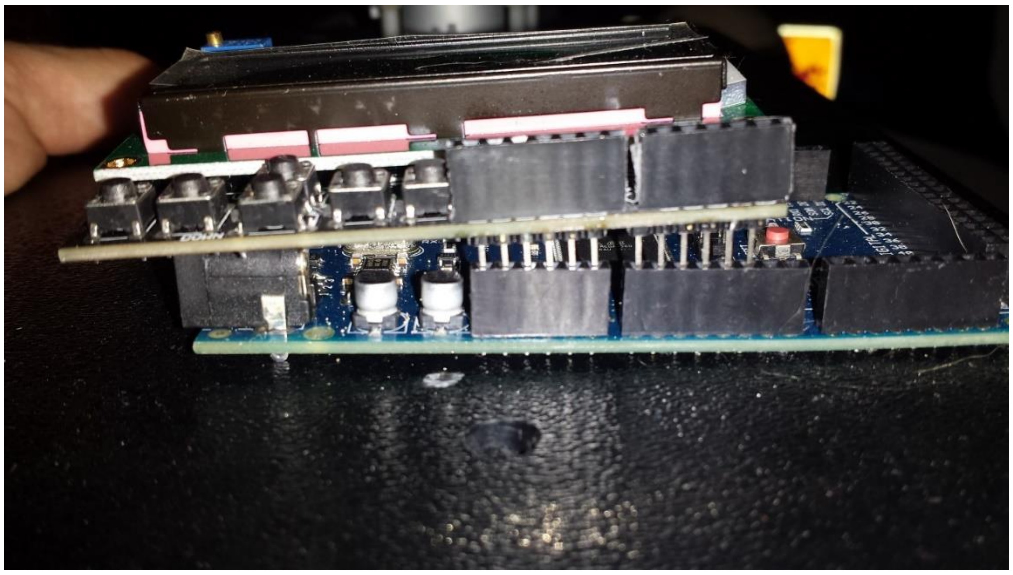
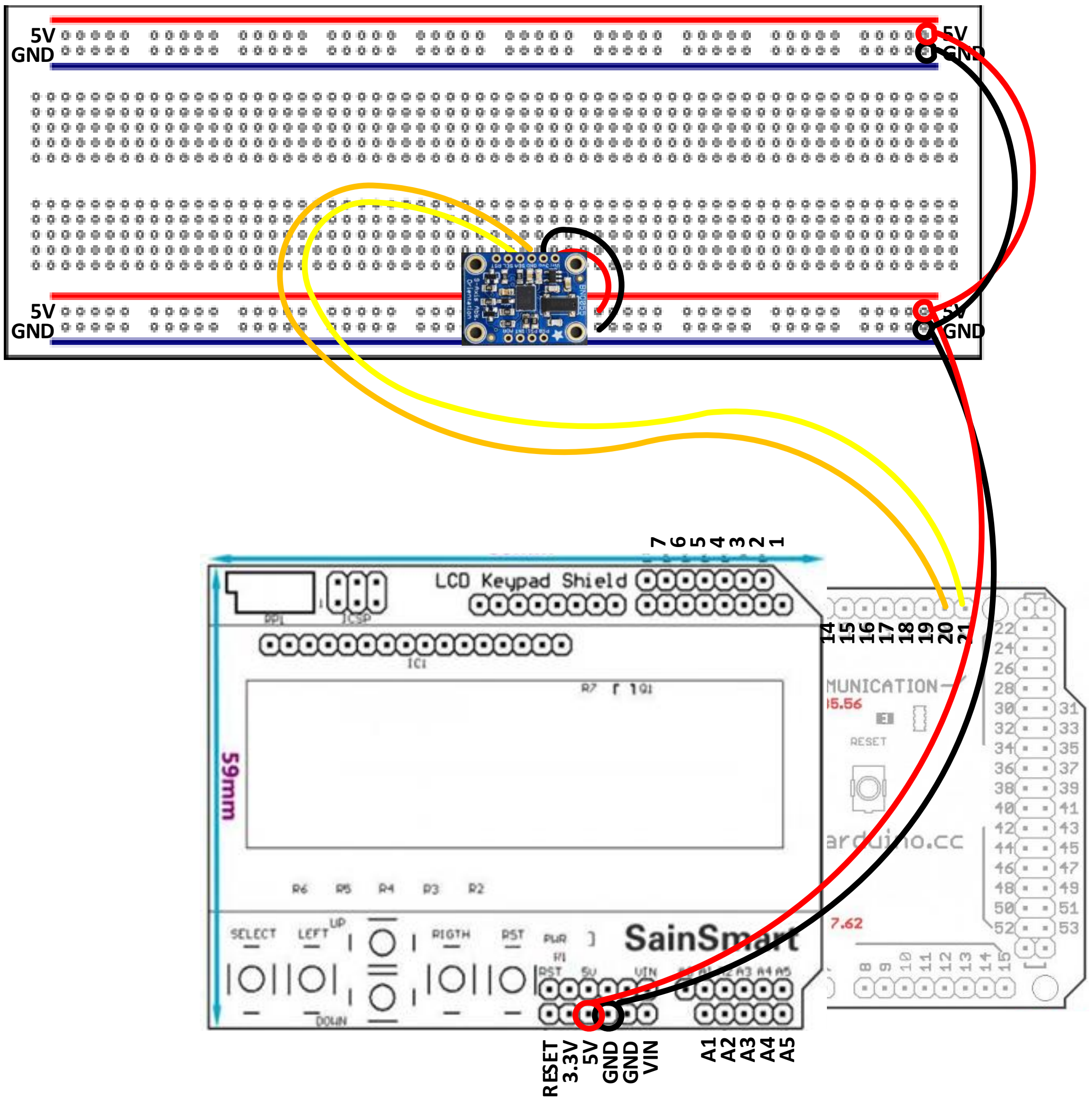


Figure 4 : Side view of completed assembly (other side). The furthest left pin header should line up. The Pin header to the right will have 2 vacant pin spots.



You need to wire up the IMU to the Arduino with the keypad installed. Wire up the GND to the ground line, the VCC to the 5V line, and the SDA and SCL to the SDA and SCL pins of the Arduino Mega. See the following Figure:

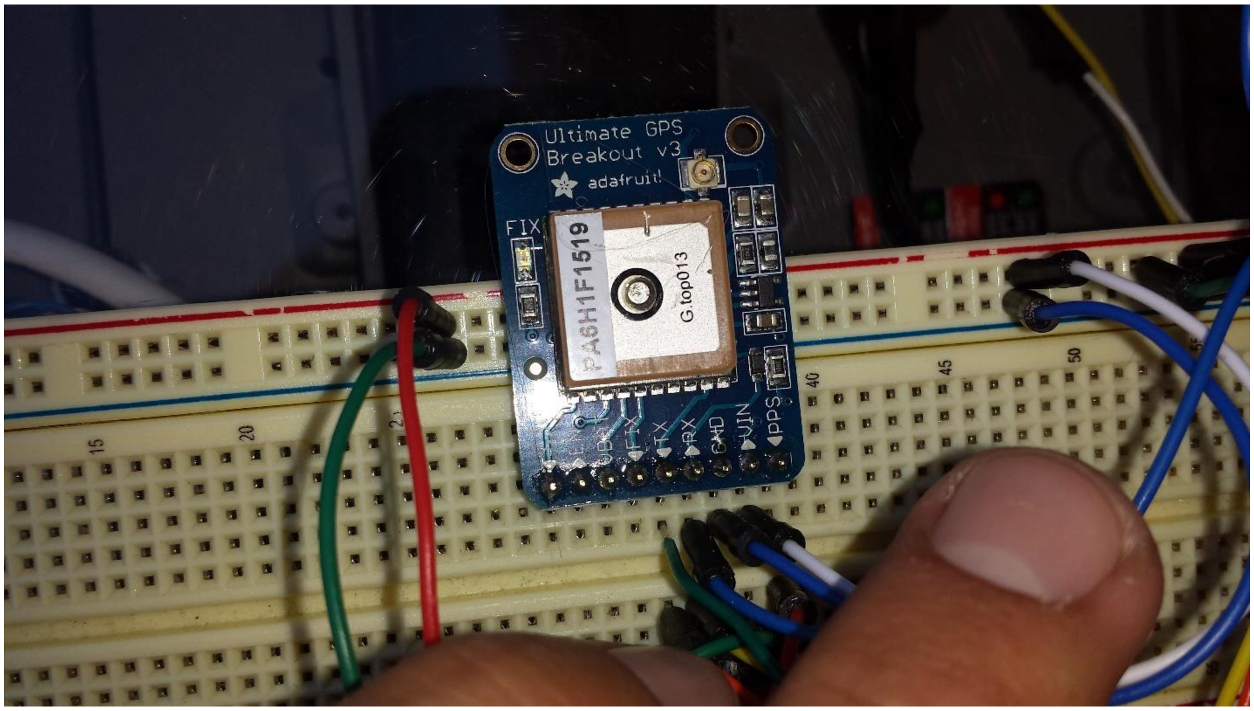


It’s better to power up the IMU sensor and GPS module with voltage converter’s 5V and GND and then power up the Arduino (Vin) from the 5V.

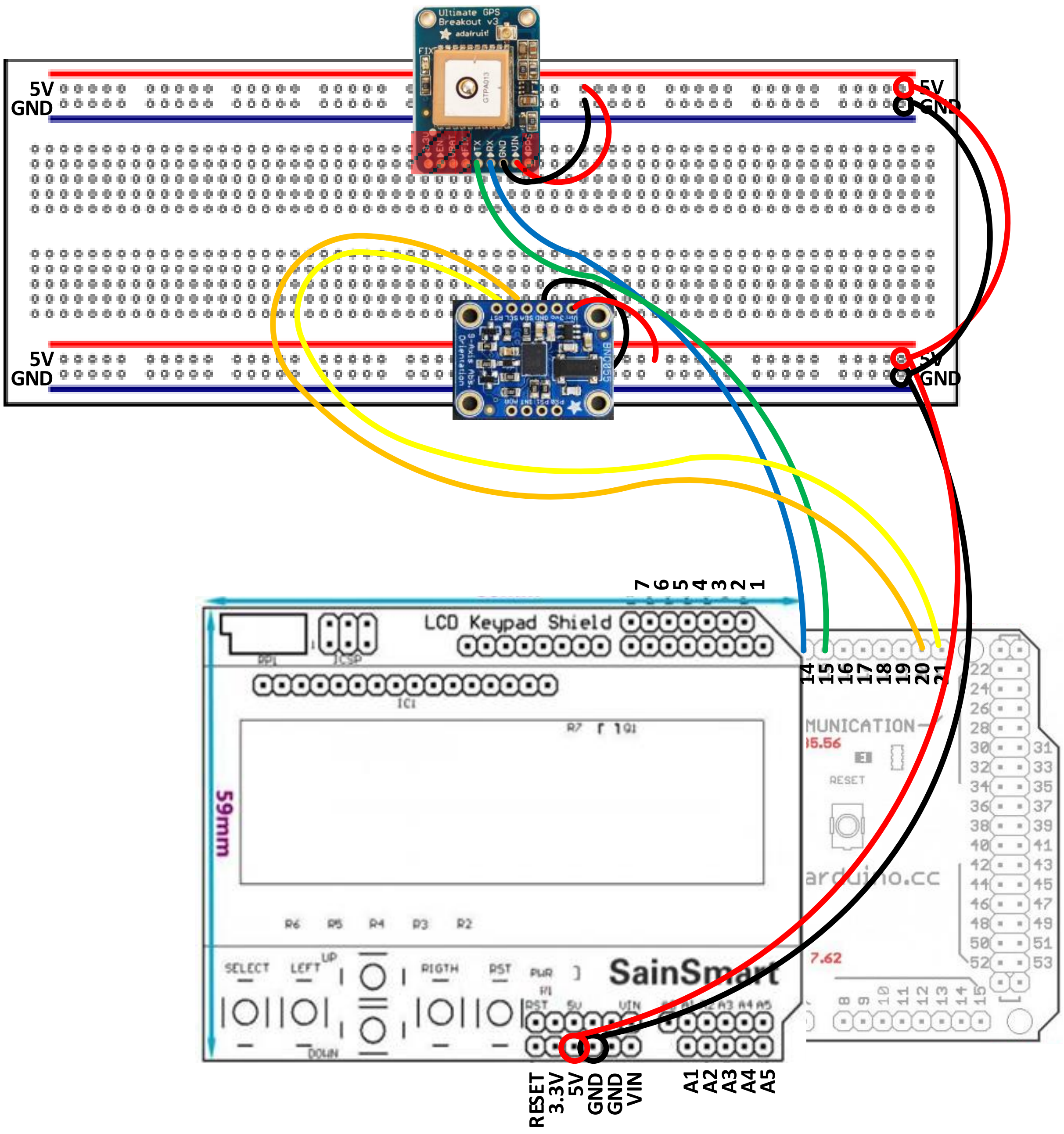
Part 2: Soldering the GPS (If not already soldered)

Now it is time to unpack the GPS chip from its packaging. You will notice an in header for the GPS and the chip itself. Use the male header and solder it under the GPS module.

Next we need to install the GPS and wire it up to the Arduino. The following image shows how to mount the GPS module on the breadboard:



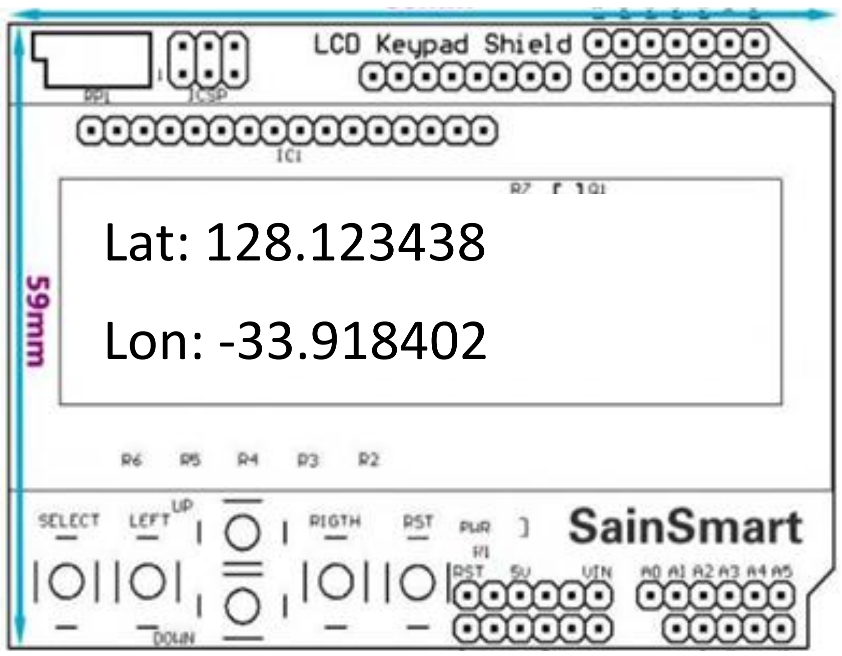
Connect the RX pin of the GPS to the TX pin of the Serial 3 and the TX pin of the GPS to RX pin of the Serial 3 on Arduino mega. **Don't use TX0 and RX0, it's already used for serial monitoring on PC.** The following figure depicts the wiring of the GPS module.



Part 3: operating the LCD

The next step is to utilize the GPS and monitor the location on LCD. The GPS module uses NMEA format to report the position. It sends NMEA sentences which contains latitude and longitude. Use the GPS at 1Hz. You can read the Latitude and Longitude data using “GPS.latitudeDegrees” and “GPS.longitudeDegrees” functions.

Try to print the latest latitude and longitude data on the LCD with 6 decimal values. (similar to the following image)



Hints:

The GPS module needs to track at least 4 satellites in order to output a valid coordinate. We recommend to operate the GPS after tracking at least 5 or 6 satellites. You can accomplish this by using "GPS.satellites" function.

The GPS will not work inside a building and it works inaccurately near tall buildings.

The GPS has a red light on it that will blink about once a second while it is trying to track satellites (this means it will NOT output any coordinates).

Once a fixed coordinate is achieved, the light will blink every 15 second. (this can take up to 5 minutes).

Removing power will reset the GPS tracking and it needs to re-track again.

You can test the achieved coordinates by a cell phone or Google maps.

The GPS is accurate to 5 meters in normal condition, you can improve it by operating when there are more GPS satellites available (6 or more).

Part 5: Programming

You need to program the Arduino to navigate from an arbitrary position to a set destination using GPS and the IMU. In order to save the destination position, the user needs to move the car to the desired destination and save the position by **pressing a button** (Note that: the GPS should have a fixed position before saving the position). Once the position is acquired, the user moves the car to an arbitrary position indicated by TA and the car will start driving **when the button is pressed**. The car shouldn't drive before pressing the button. The car should navigate towards the set destination and drive smoothly. When the car reaches the destination (within 5 meters of the location), it should stop and it shouldn't drive again.

You can assume there will be no major obstacles in between the starting and ending points of the navigation path. So you can avoid the obstacles by picking up the car.

Hints:

The GPS can output latitude and longitude data at the rate of 1Hz, 5Hz, and 10Hz. the faster the rate of output, the less accurate the coordinates will be. Then, use 1Hz.

You will probably find that creating a feedback loop using the 1Hz update rate is too slow and results in choppy and inaccurate turns. In order to address this problem, you need define another loop (e.g. 100Hz) for navigation which use the GPS data with 1 Hz rate and IMU.

Submission files:

You have to submit the following files

1. Project5.ino
2. Integrity.txt
3. Group's_members.txt

Outlines for all the files are provided with the project document.

All files should be zipped into a single file named CSE325_Fall2017_Project5.zip and be uploaded to the blackboard before the due date. Students have to submit the Arduino code and any library code you may have written and used for the GPS navigation project. All necessary codes need to be zipped into a single file "project5.zip".

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

How the project will be graded:

Grading for this project will be done by TAs. On the demo day, the TA will bring you to the destination point and asks you to press the select button on the keypad to save the destination. Then, the TA will ask you to take the car to a starting point with an arbitrary direction. The car should drive smoothly towards the destination and stop within 5m of the destination point. You may be required to test your car on multiple source and/or destination points.

Note that even if you have worked on one of the aspects of the program, you have to be able to answer the questions about all aspects of the application.

Grading:

The grading rubric is as follows:

Points	**Description**
1	Car is able to drive.
5	Car stops within 5 meters of the destination. It should reach the destination in less than 5 minutes after the car starts moving. (The distance is less than 50 m.)
3	Car takes a reasonable trajectory to the destination, e.g., does not wander aimlessly.
1	The group member is able to answer the TA's questions
-5	If the implementation is polling based, rather than interrupt based
	If you do not upload your code up to the deadline, you will be graded 0.