“Parallel Parking R/C Car”

EECS 129B Senior Project Report
Prof. Klefstad

Junghoon Ha aka Eric Ha
Stevanes Hermawan
Willie Pramono

Using the infrared sensor, we are building an application for future vehicle i.e. car. The main function of the sensor primarily is to guide the car to do parallel parking by itself. With the vision that people do not need to drive their car anymore (automated driving) in the future, the use of this sensor is useful to help the car park in parallel.

Table of Content

2. Project Description

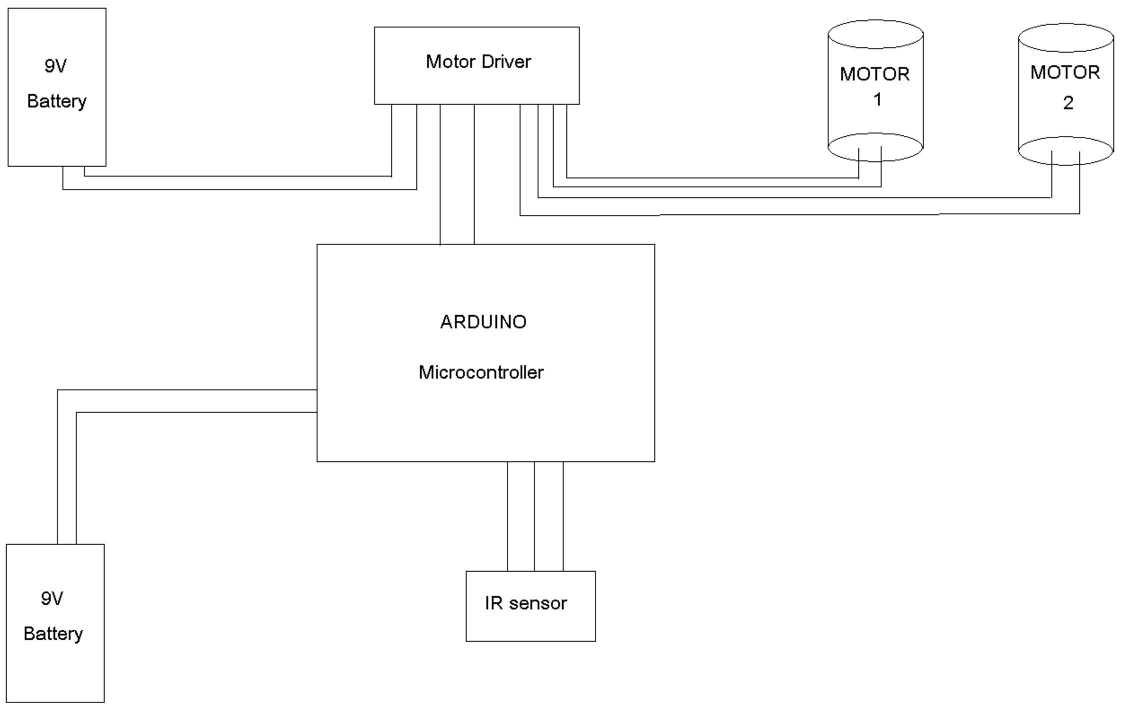
We as engineers may have thought about building a dream car. How is an automated car possible? Well, it’s simple. At least the idea is. First, we need to make it clear that we are not building a car. Furthermore, we are not saying we are going to make a car that moves without a driver. We simply want to create a system that will help the car to parallel park by itself.

This project is inspired by the new Lexus LS, which has a feature of self parallel parking. Since it is too expensive to experiment with a real car, we sized down our object of experiment to an RC car.

Using the microcontroller that we used in the previous term, Arduino, we control the car. On the stripped down RC car, we attach a breadboard on top of it. The only parts we used from the original RC car are the motors, the gears (to rotate the wheels) and the wheels. Using the motor driver to control the motors based on the instructions sent by the Arduino. Arduino simultaneously move the car forward while the IR sensor feed the distance information to Arduino. The Arduino then keep moving the car forward until some distance condition is met. If there is enough space to park and the conditions are met, the car will start to self-park by using pre-defined movements. With this method, it is possible to apply this system for different type of parking.

This project is designed with the view that future car does not require people to actually drive it. With this in mind, this system would be able to assist the car in parallel parking. And since this system only use one sensor and predefined movements to park the car; we can apply this kind of system to any kind of parking.

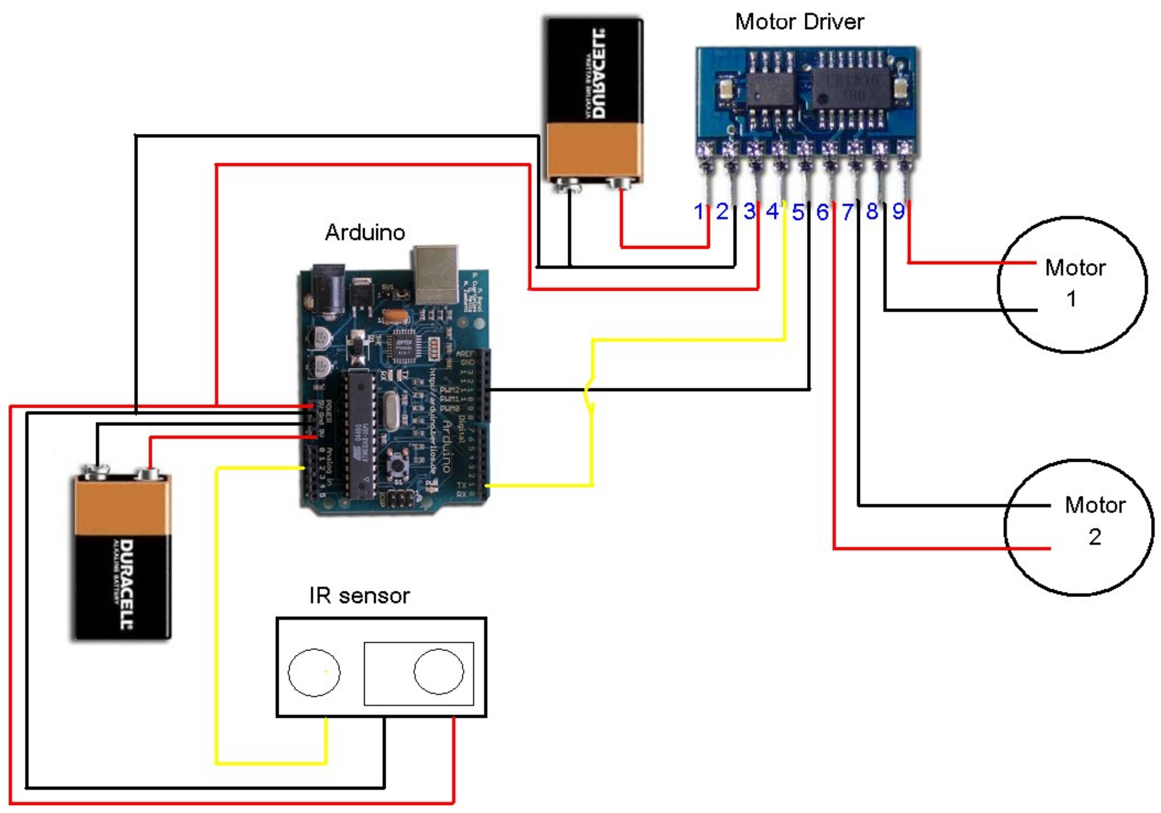
<u>3. System Level Block Diagram</u>



Description:

- 9V Batteries
  - Used to power up arduino and the motor driver.
- Motor Driver
  - Used to control the two motors: motor 1 and motor 2.
- Arduino Microcontroller
  - The brain of the project. The code that we wrote will be uploaded to this microcontroller. This microcontroller will then control the motor driver based on the inputs received from the IR sensor.
- IR Sensor
  - Used to detect distance, which is essential in positioning the car for parallel parking.
- Motor 1
  - This motor will drive the RC Car forward or reverse, controlling the rear wheels.
- Motor 2
  - This motor will function as the servo for the RC Car, controlling the front wheels i.e. making the RC turn left or right.

4. Circuit Level Block Diagram



The measurements used for the descriptions bellow are assumed ideal conditions and not mounted using a breadboard.

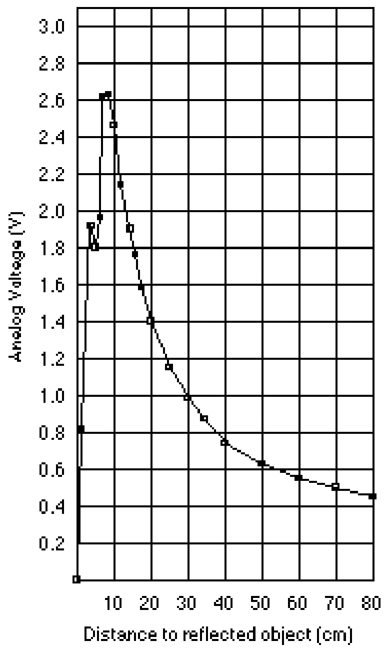
<u>Arduino</u>

- The 9V and GND are connected to the positive and negative terminals of the 9V battery respectively. This will power up Arduino.
- The 5V is connected to the Vcc of the IR sensor and the pin 3 of the Motor Driver. For the IR sensor, the 5V should provide the power for the IR to do the distance measurement. As for Motor Driver, the 5V will provide the logic power supply which will be used in determining the direction of the motors.
- GND is also connected to the GND of the IR sensor and pin 2 of the Motor Driver as depicted using black wires.
- Analog pin 2 is connected to V_o of the IR sensor as depicted using yellow wires. This pin will read in the values for distance reading done by the sensor.
- Digital pin 11 is connected to the motor driver as the reset pin. The function of the reset pin is to stop all the motors' activities.

- Digital pin 1 (TX) is connected to the motor driver as a mean to send instruction to the motor driver. Using a specific format, we control the motor driver using Arduino. The serial instruction sent through this pin enable us to control the direction of both motors used in this experiment.

IR Sensor

- V_{cc} is the input voltage for the IR sensor. This pin is connected to the 5V source from the Arduino as depicted using red wires.
- V_o is the output voltage from the IR sensor. The output voltage is in analog which result will vary depending on the distance of the object with to the IR sensor. The output voltage is expected to be varying from 0.7V to 2.6V depending on the distance measured. As the distance measured becomes larger, the output voltage will decrease from 2.6V. The minimum and maximum range this IR sensor can measure is 10 cm and 24 cm respectively with an error margin of 3 cm.
- GND is connected to the GND of the Arduino. This value should be absolute 0V.



Motor Driver

- Pin 1 is used as *motor supply*. The possible input voltage for this pin ranges from 1.8V to 9V. In this case, we use 9V.
- Pin 2 is the ground. We connect this to the GND of the 9V and the GND of the Arduino.
- Pin 3 is the *logic supply*. This pin should be connected to the 5V pin of Arduino. This will drive the logic gates inside the Motor Driver.
- Pin 4 is the *serial control input*. Using this pin, we control the logic of the motor drivers in determining in what directions the motors should be. This pin is connected to the Digital pin 1 of Arduino.

- Pin 5 is the *reset pin*. This pin is only used if we want to stop all the motor activities immediately. By default, this pin should be kept to high at all time (i.e. 5V). Hence, this is connected to Digital Pin 11 of Arduino which steadily supply 5V unless we want to reset then we set the output of Digital Pin 11 to be 0V.
- Pin 6 and Pin 7 are connected to Motor 1. The values for both pin 6 and 7 should be either Hi or Lo (with an exception that both should not be Hi). This will drive the bipolar motor to either function in clockwise or counter clockwise direction. Pin 6 and 7 could be both Lo when we are not using the motor. Another case is when the *reset pin* is set to zero.
- Pin 8 and Pin 9 are connected to Motor 2. The values for both pin 8 and 9 should be either Hi or Lo (with an exception that both should not be Hi). This will drive the bipolar motor to either function in clockwise or counter clockwise direction. Pin 8 and 9 could be both Lo when we are not using the motor. Another case is when the *reset pin* is set to zero.

RC Car

We used a fully functioned RC car and strip the whole components down except the motors, which control the movement of the car. Using the motor driver, we are able to manipulate the motors to achieve our objectives in parallel parking. The motors are directly connected to the motor driver and are specified to be as follow:

Motor 1

- This motor control the forward or reverse movement (i.e. the rear wheels)

Motor 2

- This motor control the left or right movement of the car (i.e. the front wheels)

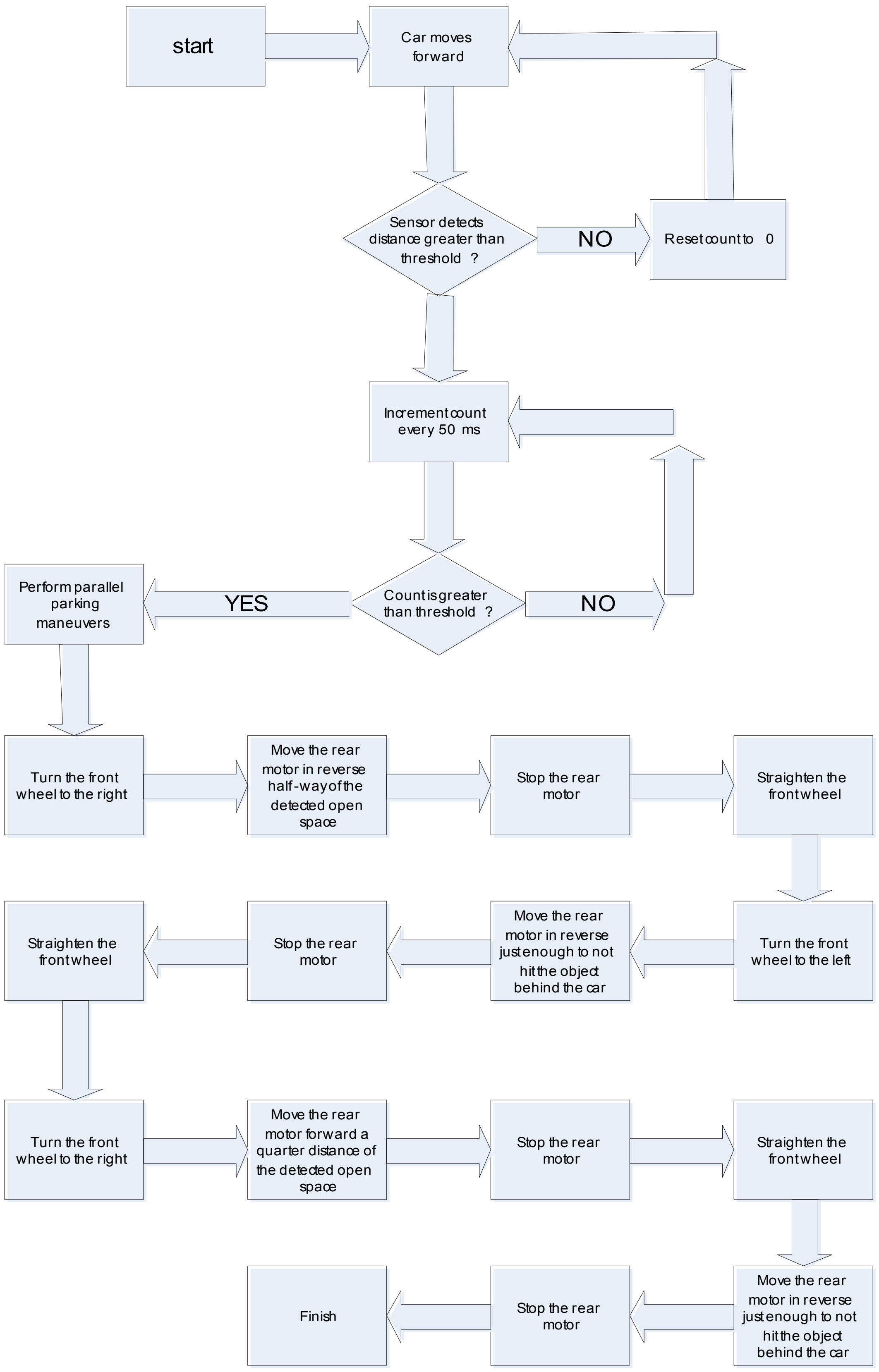
5. Software description

The software that we used is provided from www.arduino.cc . This software is used to load the code into the Arduino, our microcontroller. Using our knowledge in C language, we can write the code (Arduino uses C language); Like in C, the use of header file is permissible in Arduino. The only difference is Arduino keep looping the main function, whereas in C, we need to define the loop for the main function.

By our specification, the IR (InfraRed) sensor will always be on. This will feed the input to Arduino. When we start the system, the car will go forward. While the car is moving forward, the IR "measures" the usable distance.

To park the car, we have two scenarios. First, we want to park between two cars. The sensors, reading distance over 10 cm, will start counting by seconds. The distance is measured by finding the time it took from one object to the next. We use the simple formula of distance = time x speed. The speed is known; the time is the one we look for here. When the sensor found the second object, the time calculation stopped. If the distance measured is enough, then we start parking; otherwise, do nothing.
The second scenario is we set a threshold time. If the time that we measured is greater than the threshold time, then the car has enough space to park. The threshold time is the time required for the car to cover the distance, which enable us to park the car.
Now, using the inputs from IR, the Arduino process the data then output through the Motor Driver. Before using the motors, we need to reset the motor driver. To do this, we set the RESET to LO then to HIGH. First, we sent 0x80 then 0x00. Then, we send an 8 bits data to the motor driver from the arduino. The most significant bit is always zero, the 2nd most significant to the 7th define motor number and the least significant bit defines the direction, 1 for forward and 0 for reverse. Next, we send the speed, from 0 to 127, which is converted to binary. These bits are sent one after the other in the order mentioned by the serial connections.

<u>Flow Chart</u>



start
Car moves forward
Sensor detects distance greater than threshold ?
NO
Reset count to 0
Increment count every 50 ms
Count is greater than threshold ?
NO
YES
Perform parallel parking maneuvers
Turn the front wheel to the right
Move the rear motor in reverse half-way of the detected open space
Stop the rear motor
Straighten the front wheel
Turn the front wheel to the left
Move the rear motor in reverse just enough to not hit the object behind the car
Stop the rear motor
Straighten the front wheel
Turn the front wheel to the right
Move the rear motor forward a quarter distance of the detected open space
Stop the rear motor
Straighten the front wheel
Move the rear motor in reverse just enough to not hit the object behind the car
Stop the rear motor
Finish

6. System Test Plan

Individual Component Test

- Arduino
  - There are several ways to test whether the arduino work or not. The basic test to see whether the Arduino is functioning well is to plug the USB connection to the Arduino Board from the computer. Using a voltmeter, we test if there is a voltage coming out of the 5V pin. If there is, then Arduino is powered up nicely.
  - To test whether the pins work or not, we have to load a code into Arduino which basically set the entire pins into OUTPUT with values to High. Using voltmeter, we need to find the voltage to be around 5V.
  - The software to write the code in Arduino can be found on the website www.arduino.cc. The code is based on C language.
- IR sensor
  - The required input voltage for the IR sensor ranged from 4.5V to 5.5V. So, without using the micro controller, we can use 3 x 1.5V batteries to power up the IR sensor. With the Vcc and GND properly connected to the batteries, we can use the voltmeter again to measure the values for Vo. With the voltmeter properly attached to Vo and GND, we can test if the IR work or not by placing an object in front of the sensor. With the object becoming more distant to the sensor, the reading on the voltmeter should be decreasing.
- Motor Driver
  - Unfortunately, in order to test if this component properly works or not, we have to use the microcontroller.

    To test this component individually, additionally,we need a voltmeter, 9V battery and Arduino. We set the serial port to 9600 for communication line between the Motor Driver and Arduino.

    There are four parts for the input that we need to send through this communication line. The first byte should be 0x80, next should be the motor number either defined as 0x00 or 0x01 (first or second motor). Next is motor number, 0 or 1; next is direction of rotation, 0 or 1; last is the speed, varied form 0 to 127 (stop to fastest).

- RC Car
  - Before we strip down the RC car down to the motors, we can play with the RC car to make sure that the car functions well (i.e. forward, reverse, turn left or right)
  - After we strip it down to the motors, we can apply a DC voltage directly to each motor with the maximum range up to 4V (for safety reason). Regardless the positive and negative terminals, the motor should rotate the wheels in one direction. When the voltages applied to the opposite motor terminals, the motor should now rotate in the counter direction.

    Since the motors are bidirectional, you should not worry about applying the wrong voltages to the motor terminal.

<u>Test the whole project</u>

Since our objective is to parallel park, ideally, we could use two RC cars to help the simulation. By placing the cars in a distance, we will attempt to park the car in between. The two cars must be separated by at least 66 cm (our distance threshold). Although it is possible to park in smaller distance threshold, we use this value instead because this way, we can compensate any possible error that might occur in either the distance calculating or the parking movements.

Using the IR sensors placed on the side, we will measure the distance between the cars. Instead of using RC Car, we could use any object with the height of at least 15 cm and the width is at least 4 cm.

Start by placing the car next to the object and press the button to start the simulation. If there is enough distance to park, the car will start to park as soon as the IR sensor detects the second object (i.e. car in front). Using a predefined movement, the car will park it self in parallel.

<u>Test Case</u>

Case 1: Place a car/object next to our car. Make sure that is aligned correctly. Run the car. By default, it will park itself if the distance threshold is met.

Case 2: Place some cars/objects in a line and put them apart by some distance. If there's not enough space for the car to park, it will proceed forward. When the distance threshold is met (enough space), the car will parallel park.

Case 3:
When there's no obstacle / reference for the sensor, the car will start doing the pre defined movements by itself.

7. Cost Analysis

- Arduino USB Board
  - From www.sparkfun.com priced at $31.95 (shipping and tax excl).
- IR Sensor: GP2D15
  - From www.acroname.com priced at $12.50 (shipping and tax excl).
  - Backup available from www.sparkfun.com

- 2x 9V Batteries
  - We bought ours from Radio Shack for $9.99 for 4-pack (shipping and tax excl).
  - For this project, the cost for the batteries used is $4.99
- Motor Driver: Micro Dual Serial Motor Controller
  - From www.pololu.com priced at $23.00 (shipping and tax excl).
- RC Car
  - From Fry's Electronic priced at $21.95 (tax excl).
  - Backup available at Radio Shack or any hobby shop.
- Breadboard (#194271)
  - From www.jameco.com priced at $15.95 (shipping and tax excl).
  - Backup available at JK Electronics at Westminster.
- 9V Battery connector
  - From www.jameco.com priced at $0.34 (shipping and tax excl).
  - Backup available at Radio Shack.
- 100' reel 22awg, solid black, red, and yellow wires (#36792, #36856, #36920)
  - From www.jameco.com priced at $5.49 each (shipping and tax excl).
  - We do not use that much wires for this project and the wires that we used are the left over from EECS 129A projects.
  - Our estimation for the cost of the wires for this project is about $2.50.

The base cost comes to US$ 116.18.

8. Summary

We met quite a lot of problems doing this project. Some problems that we met are related with the size of the RC car, the distance measurement sensor and power problem. Originally, we wanted to use several sensors to make the parking measurement more accurate. Unfortunately, the range of these sensors is far too large to be used in the RC car. So, we decided to use one sensor to measure the distance and when the conditions, for parallel parking, permits, we apply the pre defined movements.

The ultra sonic sensor that we originally plan to use has a range of measurement 10-60 cm from the sensor. This creates a far less accurate measurement. Another problem is met when we wanted to use the ultra sonic sensor, it requires AC voltage source, which we cannot provide using Arduino. Using IR sensor solves this problem. This sensor does not require AC voltage to operate and is smaller in size compared with the ultrasonic sensor. These sensors that we tested on will work well when applied on actual car because 10 cm difference is very close for an actual car. In real car application, ultrasonic sensor is more effective because it utilize sound to measure the distance. IR relied on light, so it may not be accurate in different conditions. In our case, IR works well for us.

Although the hardware may seem simple, the programming and the fine-tuning of this project take a lot of time. It took us a lot of time to configure and understand how each component works. When creating the pre-defined movement, we have to do it by trials.

Challenges:

1. Power consumption, constantly not enough power/voltage to power the circuit.
2. Motor driver constantly overheats.
3. RC car did not go straight.
4. Time constraint.
5. RC did not stop at the right position.
6. Writing the C language.

Solution:

1. We use two batteries, one to power up Arduino, one to power up the motor driver.
2. Between tests, we have to rest the motor driver for a period of time (about 2 minutes). And we tried to run the car less than two minutes.
3. We align both the steering and the wheels; originally the car is heading a bit to the right.
4. We spent extra hours finding the suitable algorithm (late hours).
5. We set the distance threshold to be higher than the actual space required to park the car.
6. Writing the code for the fine tuning take lots of trial and error calculation. When taking the reading from the IR sensor, we cannot use centimeter(which is more

accurate count) because the formula requires floating point (which arduino would not compile); instead, we used inch (less accurate but do the work well).

Despite all the problems that we met working on this project, we are pleased that we have a successful project and hope this project will be useful for everyone.