Appendix A- Arduino code

This appendix presents the code created to run the car autonomously with RPLIDAR. This code contains a few libraries such RPLIDAR library, this library can be found in the attached folder of the zip file. This library is used to preconfigure the RPLIDAR's proper functioning. Only with inclusion of these libraries and the necessary drivers installed will make the RPLIDAR run properly.

```
//Now we are defining the motor connections

int EnA = 21; // the jumper pin is on on EnA so no need to define the pin number

int In1 = 20;

int In2 = 19;


#define RPLIDAR_MOTOR 6 // The PWM pin for control the speed of RPLIDAR's motor.

// This pin should connected with the RPLIDAR's MOTOCTRL signal



//Here we include the servo library

#include <Servo.h>

#include <RPLidar.h>

#include <limits.h>



Servo servo;  // create servo object to control a servo

RPLidar lidar; // Create lidar object for lidar sensor


float OldAngle = 0;
```

```
int i = 0;

int max_dist;

float max_angle;

int max_k;

unsigned long servoMillis = 0;

unsigned long motorMillis = 0;

unsigned long currentMillis;

const unsigned long servo_interval = 500; // other interval is 1000

const unsigned long motor_interval = 3000; // other intervals are 2000 and 1500

void setup() {


  Serial.begin (115200); //we're going to use arduino serial monitor to monitor the
//results, so we setup the serial monitor at 115200 baud

  lidar.begin(Serial1);

  // setup the motor control pins

  //------------------------------------------------------------

  pinMode(EnA, OUTPUT);

  pinMode(In1, OUTPUT);

  pinMode(In2, OUTPUT);

  // set pin modes

  pinMode(RPLIDAR_MOTOR, OUTPUT);

  //------------------------------------------------------------


  servo.attach(23);  // attaches the servo on to pin 23 to the servo object
  // Print Values
  //-------------------------------------------------------
  Serial.print("\nDist");
  Serial.print("\t");
  Serial.print("Angle");
  Serial.print("\t");
  Serial.print("Servo");
  Serial.print("\t");
  Serial.print("\n");
  //-------------------------------------------------------
}


//Now we are going to create a set of functions that will be used to determine the movement of the car
//-------------------------------------------------------
void goForward()   //run motor forward
{
  digitalWrite(In1, LOW);
  digitalWrite(In2, HIGH);
  digitalWrite(EnA, HIGH);
}

void goBackward()   //run motor backwards
{
  digitalWrite(In1, HIGH);
  digitalWrite(In2, LOW);
  digitalWrite(EnA, HIGH);
}

void goNothing()
{
  digitalWrite(In1, LOW);
  digitalWrite(In2, LOW);
}
//-------------------------------------------------------

//Here we define the processing logic function for the movement of the car.
//-------------------------------------------------------
void turnSides(int givenAngle) {
  int koefL = -28; //Edit Coefficent as -28 as the offset angle of the front wheels for
the right turn orientation
  if ( givenAngle >= 0.0 && givenAngle < 90.0 ) {
    int totalAngle = givenAngle + 90 + koefL; // turn right
    currentMillis = millis();
    if (currentMillis - servoMillis >= servo_interval)
    {
      servo.write(totalAngle);
      servoMillis = currentMillis;
      if (currentMillis - motorMillis >= motor_interval)
      {
        goForward();
        motorMillis = currentMillis;
      }
    }
  }

  else if (givenAngle >= 270.0 && givenAngle < 0.0) {
    int totalAngle = givenAngle - 270 + koefL; // turn left
    currentMillis = millis();
    if (currentMillis - servoMillis >= servo_interval)
    {
      servo.write(totalAngle);
      servoMillis = currentMillis;
      if (currentMillis - motorMillis >= motor_interval)
      {
        goForward();
        motorMillis = currentMillis;
      }
    }
  }

  else if (givenAngle >= 90.0 && givenAngle < 180.0) {
    int totalAngle = givenAngle + koefL; // turn right
    currentMillis = millis();
    if (currentMillis - servoMillis >= servo_interval)
    {
      servo.write(totalAngle);
      servoMillis = currentMillis;
      if (currentMillis - motorMillis >= motor_interval)
      {
        goBackward();
        motorMillis = currentMillis;
      }
    }
  }

  else if ( givenAngle >= 180.0 && givenAngle < 270.0 ) {
    int totalAngle = givenAngle - 180 + koefL ; // turn left
    currentMillis = millis();
    if (currentMillis - servoMillis >= servo_interval)
    {
      servo.write(totalAngle);
      servoMillis = currentMillis;
      if (currentMillis - motorMillis >= motor_interval)
      {
        goBackward();
        motorMillis = currentMillis;
      }
    }
  }
}
//------------------------------------------------------------

// Put the code here to repeat in a loop
void loop() {

  float angle = 0;
  int k=0;
  if (IS_OK(lidar.waitPoint())) {
    float distance = lidar.getCurrentPoint().distance; //distance value in mm unit
    angle    = lidar.getCurrentPoint().angle; //anglue value in degree
    bool  startBit = lidar.getCurrentPoint().startBit; //whether this point is belong to
//a new scan
    byte  quality  = lidar.getCurrentPoint().quality; //quality of the current
//measurement
    k= servo.read(); //Gets the Servo Motor angle

    if (distance > max_dist)
    {
      max_dist = distance;
      max_angle = angle;
      max_k = k;
  }
}

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();

    // start motor rotating at max allowed speed
    analogWrite(RPLIDAR_MOTOR, 255);
    delay(1000);
  }
}

if (angle - OldAngle < 0) {
  //perform data processing here...
  Serial.print(max_angle);
  Serial.print("\t");
  Serial.print(max_dist);
  Serial.print("\t");
```

```
    Serial.println(max_k);
    float anglerequired = max_angle;
    turnSides(anglerequired);
    max_dist = 0;
    max_angle = 0;
  }

  OldAngle = angle;
}
```