Real Time Embedded Systems, Final Project Report

Guyu Liu gl1813, Naisheng Zhang nz862
Group 10, Team 4

***Abstract*—This report is intended to introduce our final project details, the performance of our car, the code we write, the materials we bought and the method we use to assemble our car will be introduced in this report.**

I. Project details

This project allows us to assemble a kit car, and every student need to let the car to have two main functions, the first function is to make the car avoid the obstacles, and second is to make the car find the Wi-Fi beacon, and students need to combine these two functions together, in the final part, this car will be left in the area, which contains obstacles and one Wi-Fi beacon, the car needs to go to the position of the beacon while walking to the beacon.

A. The materials we use

Here are the materials we use:

- Motor Driver: Pololu Dual DC Motor Driver 1A, 4.5V-13.5V- TB6612FNG
- Car kit: EMO Smart Robot Car Chassis Kit with Motors, Speed Encoder and Battery Box for DIY
- Micro controller: Adafruit Feather HUZZAH with ESP8266 - Loose Headers
- Battery: Lithium Ion Polymer Battery - 3.7v 1200mAh
- Ultrosonic sensor: ELEGOO 5PCS HC-SR04 Ultrasonic Module Distance Sensor for Arduino UNO MEGA2560 Nano Robot XBee ZigBee
- Servo: 4Pcs SG90 9g Micro Servos for RC Robot Helicopter Airplane Controls Car Boat
- Wi-Fi Beacon: iPhone 11 Pro personal Hot Pot

B. Implementation

In this project, we use one ultrasonic sensor to measure the distance between car and obstacles, and one motor driver to transfer the signal to power energy to enable the car to move, we use one servo to control the direction the sensor face so that we can measure the distance from different directions, which can help the car to decide which direction it can go, one Wi-Fi beacon to help the car to fix the direction. We use the Jumper Wire to connect the microcontroller and the motor driver, and we give the pin with High and Low voltage to make our car move, meanwhile, we use the Jump Wire to connect the ultrasonic sensor and the microcontroller, to make it output the wave, we use the tape and glue to combine the servo and the sensor. We decided to use the 3.7V battery to charge our motor driver first, but we found that this voltage cannot satisfy the motor driver, so we change our battery to four 4AA, and use the 3.7V battery to charge the microcontroller.

II. Algorithm Description

A. Algorithm description

Our task is to find the beacon and arrive position of our beacon, and the car needs to avoid the obstacles the distance is so close to the car, so we design an algorithm to help our car to perform better. The general idea is to make the car stop at first and when there is no obstacles near car, make the car to find beacon in the default case. But when the car detects obstacles, it avoids them.

```
switch (action) {
    // try to avoid obstacle at first.
    case 'TURNLEFT':
      motorRun(TURNLEFT, 600, 600);
      delay(700);
      break;
    case 'TURNRIGHT':
      motorRun(TURNRIGHT, 600, 600);
      delay(700);
      break;
    case 'BACKWARD':
      motorRun(BACKWARD, 700, 550);
      delay(700);
      motorRun(TURNRIGHT, 600, 600);
      delay(600);
      break;
    default:
      /*the default is to forward which
      *means there is no obstacle around.
      *So in this case,
      *let car try to find beacon.*/
      char action_find = find();
      if (action_find == 'STOP') {
        motorRun(STOP, 0, 0);
        delay(100000);
      }
      else if (action_find == 'FORWARD') {
        motorRun(FORWARD, 700, 550);
        delay(600);
      }
      else if (action_find == 'TURNLEFT'){
        motorRun(TURNLEFT, 600, 600);
        delay(600);
      }
      else {
        motorRun(TURNRIGHT, 600, 600);
        delay(600);
      }
```

```
  }
```

In the default case, we design an algorithm to find the beacon. First, we keep receiving ten RSSIs from our beacon, these represent the distance between the beacon and car. Then, we take the average from these ten RSSIs, we create a one-dimensional array to store the average value we compute, comparing them with the previous average, the current average is stored as the fourth element, and the previous value is stored in the third position of the array. if the current average is greater than previous, which means the car is closer to the beacon, so the car will continue forward, but if our current average value is less than previous, which means the car is farther and farther away from the beacon, so it will turn left.

```
char find()
{
  char action;
  if (RSSIs[4] > -50) {
    //The car is close to the beacon,stop
    action = 'STOP';
  }
  else if (RSSIs[4] >= RSSIs[3]) {
    /*when the value of RSSI become more
    *and more larger
    *(i.e.the car is closer to the beacon)
    *let car forward.*/
    action = 'FORWARD';
  }
  else {
    /*when the value of RSSI
    *become more and more smaller
    *(i.e. the car is farther and farther
    *away from the beacon)
    *let car turnleft.*/
    action = 'TURNLEFT';
  }
  return action;
}
```

```
int getRSSI() // get the value of RSSI.
{
  int sum = 0;
  int maxVal = -100;
  int minVal = 0;
  // read 10 times RSSI
  for (int i = 0; i < 10; i++) {
    int temp = WiFi.RSSI();
    sum += temp; // sum
  }
  // get the average value of 10 RSSIs
  return sum / 10;
}
```

In the avoidance algorithm, we define four actions the car will use: Forward, Turn Left, Turn Right, and Backward. When the loop begins, we make our car stop firstly and make the sensor begin to scan. The distance information has three parts: front, left and right, we compare these data each other, if one of the data in this part is less than 35cm, we make our car back, and fix the direction by turning left or right, and forward again until our car bypass the obstacle. code as following

```
char scan()
{
  char action;
  int left, right, front;

  // scan left
  myServo.write(LEFT);
  delay(500);
  left = getDistance();

  // scan right
  myServo.write(RIGHT);
  delay(800);
  right = getDistance();

  // scan straight ahead
  myServo.write(FRONT);
  delay(500);
  front = getDistance();
  if (right<DIS||left<DIS||front<DIS) {
    // if one of the distances are less
    //than the safe distance 35,
    //making car back.
    action = 'BACKWARD';
  }
  else if(left>right&&left>front) {
    /*if there are obstacbles at the front
    *and left, make car turn left.*/
    action = 'TURNLEFT';
  }
  else {
    /*if there are obstacles at
    *the front and right,
    *making car turn right.*/
    action = 'TURNRIGHT';
  }
  return action;
}
```

```
int getDistance()
{
  /*Making the interface send out
  *the ultrasonic wave that time < 2 s*/
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  /*Making the interface send out the
  *ultrasonic signal < 10 s,
  *here is at least 10 s*/
  digitalWrite(trig, HIGH);
```

```
  delayMicroseconds(10);
  // keep trig low
  digitalWrite(trig, LOW);
  // read the length of the pulse
  int distance = pulseIn(echo, HIGH);
  /*calculate distance based on
  *the length of the pulse*/
  distance = distance / 58;
  //print distance
  Serial.println(distance);

  if (distance >= 50)
  {
    //return 50 if distance > 50
    return 50;
  }
  else {
    return distance;
  }
}
```

In our algorithm, our car will keep turning left when it is far away from the beacon until the direction is right. when the car facing the right direction(i.e. the car is walking toward the beacon), it will never turn left and just move forward to the beacon if there is no obstacle between the car and beacon. When the car arrive at the beacon, it will stop.

III. Performance evaluation

Actually, when we upload our code to the microcontroller, we found that because the Wi-Fi strength is unstable, so the car cannot exactly determine whether it is further away or closer to the beacon. So sometimes it will turn left when it moves toward the beacon, which means it will walk into wrong direction, sometimes opposite to the beacon. But when it realizes it is far away from the beacon, it will keeping turning left and moving forward until the direction is right. Our car will move from a big circle to a small circle and then move to the position of the beacon at the end. However, because of the unreliable RSSI values, the car will take a long time to find the right direction. But it always performs successfully if we give it enough time.

In our demo, our car avoids the obstacles successfully while walking to the beacon, and find the beacon and stop on the top of our cellphone finally!

Summary

This project is a very challenging project, it takes us about a half month to finish it. This project helps us to put theoretical knowledge we learned from lectures into practice and help us to understand the knowledge deeply. We implement the avoidance first, and then we design an algorithm to find the beacon, in the final state, we combined these two algorithms together and test the car's performance. Our performance is great and we finish all the task when we demo our project, I hope this report can help the reader to understand our whole project progress and details.