Project Code

```
// Garbage truck V0.1
#include <SPI.h>                /* to handle the communication interface with the modem*/
#include <nRF24L01.h>              /* to handle this particular modem driver*/
#include <RF24.h>
#include "printf.h"
// motor control pins
int motor_d0 = 14;
int motor_d1 = 15;
int motor_d2 = 16;
int motor_d3 = 17;

// IR sensor pins
int sensor_front_left =2;// 8;
int sensor_front_right =3;// 9;
int sensor_left_side = 4;//10;
int sensor_right_side =5;// 11;
int obstacle_sensor =6;// 12;
// status register
int sensor_front_left_status =0;
int sensor_front_right_status = 0;
int sensor_left_side_status = 0;
int sensor_right_side_status = 0;
int obstacle_sensor_status =0;

int rotor_current_status=0;

int robot_action =0;

//NRF
RF24 radio(9,10);

// LED  pin details
int led_orange = 7;
int led_red =8;

const uint64_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL };

void setup() {
// Serial monitor setup
Serial.begin(9600);
 printf_begin();
delay(500);



Serial.println("GarbageTruck V0.1");
Serial.println("================");
Serial.println("Started..........");
//Input pins
pinMode(sensor_front_left,INPUT);
pinMode(sensor_front_right, INPUT);
pinMode(sensor_left_side, INPUT);
pinMode(sensor_right_side, INPUT);
pinMode(obstacle_sensor, INPUT);
//Output pins
pinMode(motor_d0, OUTPUT);
pinMode(motor_d1, OUTPUT);
pinMode(motor_d2, OUTPUT);
pinMode(motor_d3, OUTPUT);
pinMode(led_orange, OUTPUT);
pinMode(led_red, OUTPUT);

setup_nrf();
rotor_current_status = 1; // assumming robot will always starting from initial position
//uturn();
}
int update_nrfdata()
{
   if ( radio.available() )
    {
      // Dump the payloads until we've gotten everything
      unsigned long data_r;

        // Fetch the payload, and see if this was the last one.
      radio.read( &data_r, sizeof(unsigned long) );
      Serial.print("Received:");
      Serial.println(data_r);
      delay(20);

if(data_r == 1)
{
  if(rotor_current_status == 1)
  {
  robot_action =1;
  rotor_current_status = 2;
  }
  Serial.println("1");
}
else if(data_r == 2)
{
  robot_action =2;

  Serial.println("2");
}
else if(data_r ==3)
{
  robot_action = 3;
  Serial.println("3");
}
else if(data_r == 4)
{
  robot_action = 4;
  Serial.println("4");
}
else
{
  Serial.println("unknown");
  return 0;
}
}
return 1;
}
void loop() {
update_nrfdata();
switch(rotor_current_status)
{
  case 1: // waiting for move command
```

```
    if(robot_action ==1)
{
  rotor_current_status = 2;
}
  break;
  case 2: // after reeceived movecommand
if(check_obstacle() == 0)
{
  do_line_follow();
}
else if(obstacle_sensor_status == 1)
{
   robot_stop();
   Serial.println("Obstacle detected");
}
  break;
  case 3: // wait for move command

  if(robot_action == 2)
  {
     uturn();
     delay(500);
  rotor_current_status =4;
  }
  else
  {
    digitalWrite(led_orange,HIGH);
  delay(500);
  digitalWrite(led_orange,LOW);
  delay(100);
  }
  break;
  case 4:
 if(check_obstacle() == 0)
{
  do_line_follow();
}
else if(obstacle_sensor_status == 1)
{
   robot_stop();
   Serial.println("Obstacle detected");
}
  break;
}
   delay(10);
}
int check_obstacle()
{
   obstacle_sensor_status = digitalRead(obstacle_sensor);
   return obstacle_sensor_status;
}
void do_line_follow()
{
  digitalWrite(led_red,HIGH);
    // Read sensor data
  sensor_front_left_status = digitalRead(sensor_front_left);
  sensor_front_right_status = digitalRead(sensor_front_right);
  sensor_left_side_status = digitalRead(sensor_left_side);
  sensor_right_side_status = digitalRead(sensor_right_side);
  if( sensor_front_left_status == 1 && sensor_front_right_status ==1)
  {
    // move forward
    move_forward();
  }
  else if( sensor_front_left_status == 0 && sensor_front_right_status ==1)
  {
    // move right
    move_right();
    Serial.println("move right");
  }
  else if( sensor_front_left_status == 1 && sensor_front_right_status ==0)
  {
    // move left
    move_left();
    Serial.println("move left");
  }
  else if( sensor_front_left_status == 0 && sensor_front_right_status == 0)
  {
    // stop
    Serial.println("Stop");
    switch(rotor_current_status)
    {
      case 1:
      break;
      case 2:
      rotor_current_status =3;
      robot_action=0;
      robot_stop();
      break;
      case 3:
      break;
      case 4:
      rotor_current_status =1;
      robot_action=0;
      uturn();
      delay(500);
      break;
    }

  }
  digitalWrite(led_red,LOW);
}
void setup_nrf()
{
  radio.begin();                    /* Activate the modem*/
 radio.setRetries(15,15);
  radio.openWritingPipe(pipes[0]);
  radio.openReadingPipe(1,pipes[1]);
  radio.startListening();
  radio.printDetails();
}
void move_forward()
{
  digitalWrite(motor_d0,HIGH);
  digitalWrite(motor_d1,LOW);
  digitalWrite(motor_d2,HIGH);
  digitalWrite(motor_d3,LOW);
}
void move_right()
{
  digitalWrite(motor_d0,HIGH);
  digitalWrite(motor_d1,LOW);
  digitalWrite(motor_d2,LOW);
  digitalWrite(motor_d3,HIGH);
  /*digitalWrite(motor_d0,HIGH);
  digitalWrite(motor_d1,LOW);
  digitalWrite(motor_d2,LOW);
  digitalWrite(motor_d3,LOW);*/
}
void move_left()
{
   digitalWrite(motor_d0,LOW);
  digitalWrite(motor_d1,HIGH);
  digitalWrite(motor_d2,HIGH);
  digitalWrite(motor_d3,LOW);
 /*  digitalWrite(motor_d0,LOW);
  digitalWrite(motor_d1,LOW);
  digitalWrite(motor_d2,HIGH);
  digitalWrite(motor_d3,LOW);*/
}
void robot_stop()
{
  digitalWrite(motor_d0,LOW);
  digitalWrite(motor_d1,LOW);
  digitalWrite(motor_d2,LOW);
  digitalWrite(motor_d3,LOW);
}
void uturn()
{
  move_left();
  delay(3200);
//    delay(1700);
  robot_stop();
  delay(3000); // wait for some time after turn
}
void uturn_s()
{
  move_left();
  delay(3400);
//    delay(1700);
  robot_stop();
  delay(3000); // wait for some time after turn
}
```