

Lesson

4

Obstacle Mode



Introduction:

There are three parts in this lesson. You will learn how to use your car to avoid obstacles, so that the car can change the direction of motion when encountering obstacles.

Tumbler Obstacle Mode

- 1.1 Software Design
- 1.2 Upload Validation

Preparations:

- one car (with a battery)
- one USB cable

1.1 Software Design

The obstacle avoidance function of the car is also realized through the ultrasonic module and photoelectric sensor, so let's directly look at the idea of the program: (As shown in figure 1.1.1)

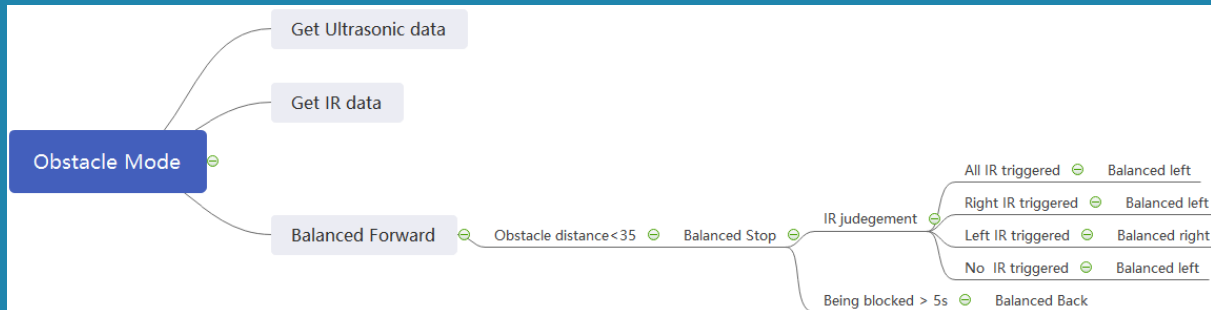


Figure 1.1.1 Programming ideas

Please open [ELEGOO Tumbler Self-Balancing Car Tutorial -> Lesson 4 Obstacle Mode -> Balanced_Car -> Balanced_Car.ino](#) in the current folder.

First of all, let's take a look at the declaration of the obstacle avoidance pattern class.

```
//in Obstacle.h
class Ultrasonic
{
public:
    void Pin_init();
    void Get_Distance();
    void Check();
    static void Distance_Measure();

public:
    static char measure_flag;
    static unsigned long measure_prev_time;
    static unsigned long get_distance_prev_time;
    static double distance_value;
    static unsigned long ir_send_time;
private:
#define ECHO_PIN 17
#define TRIG_PIN 11
#define Distance_MIN 3
#define Distance_MAX 35
#define Obstacle_MIN 40
#define DISTANCE_JUDGEMENT (distance_value > Distance_MIN && distance_value < Distance_MAX)
#define OBSTACLE_JUDGEMENT (Ultrasonic.distance_value < Obstacle_MIN)
#define OBSTACLE_JUDGEMENT2 (Ultrasonic.distance_value < 10)
};
```

Then the realization of obstacle avoidance mode.

1. Data Acquisition

```
//in Obstacle.cpp
void Function::Obstacle_Mode()
{
    IR.Send();
    Ultrasonic.Get_Distance();
    .....
}
```

2. Balanced Forward

```
//in Obstacle.cpp
void Function::Obstacle_Mode()
{
    .....
    if (millis() - follow_prev_time >= 100)
    {
        follow_prev_time = millis();
        Balanced.Motion_Control(FORWARD);
    }
    .....
}
```

3. Obstacle Judgement

```
//in Obstacle.cpp
void Function::Obstacle_Mode()
{
    .....
    while (OBSTACLE_JUDGEMENT)
    {
        .....
    }
}
```

4. Balanced Stop

```
//in Obstacle.cpp
void Function::Obstacle_Mode()
{
    .....
    Balanced.Motion_Control(STOP);
    .....
}
```

5. Blocked Or Not

```
//in Obstacle.cpp
void Function::Obstacle_Mode()
{
    .....
    if (millis() - Obstacle_time > 5000)
    {
        Obstacle_time = millis();
        Back_time = millis();
        while (millis() - Back_time < 500)
        {
            Balanced.Motion_Control(BACK);
        }
    }

    Turning_Time = millis();.....
}
```

6.IR Judgement

```
//in Obstacle.cpp
void Function::Obstacle_Mode()
{
    .....
    Turning_Time = millis();
    while (millis() - Turning_Time < 750)
    {
        if (turn_flag)
        { turn_flag = 0;
          IR.Check();
        }
    }
    turn_flag = 1;
    .....
}
```

```
//in Obstacle.cpp
void IR::Check()
{
    int motion = left_is_obstacle + right_is_obstacle;
    switch (motion)
    {
        case Meet_OBSTACLE: Balanced.Motion_Control(LEFT);
            left_is_obstacle = 0; break;

        case FOLLOW_RIGHT:
            Balanced.Motion_Control(RIGHT);
            left_is_obstacle = 0; break;

        case FOLLOW_LEFT:
            Balanced.Motion_Control(LEFT);
            right_is_obstacle = 0; break;

        case FOLLOW_BACK:
            Balanced.Motion_Control(LEFT);
            right_is_obstacle = 0;
            left_is_obstacle = 0; break;
    }
}
```

It should be noted that, considering that it is possible to touch the obstacles when moving forward, the left and right turns should be changed to left rear and right rear turns.

```
//in Balanced.cpp
void Balanced::Left(int speed)
{
    setting_car_speed = -speed;
    setting_turn_speed = speed;
}

void Balanced::Right(int speed)
{
    setting_car_speed = -speed;
    setting_turn_speed = -speed;
}
```

1.2 Upload Validation

Upload the program to the Nano board. After downloading the program, put the car on the ground. When there is no obstacle in front of the car, it will go straight at a constant speed. When the car encounters an obstacle, it will avoid it.