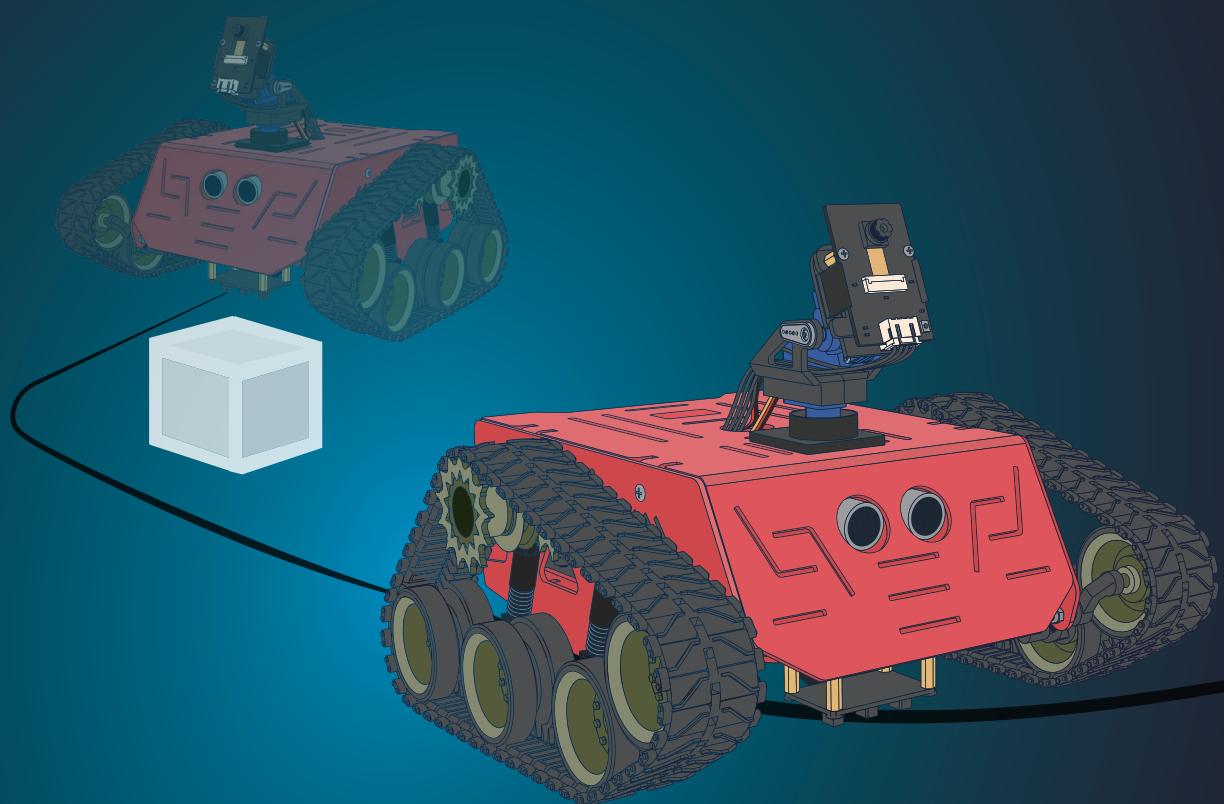


Lesson

Obstacle-avoidance Function



Introduction:

In this lesson, we will teach you how to achieve the Obstacle-avoidance Function and make the car change its moving direction when encounter obstacles.

Material Preparation:

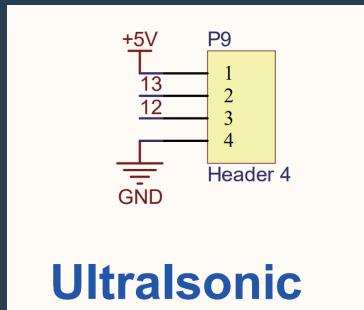
A Conqueror Robot Tank Car (with battery)

A USB Cable



In our kit, the Obstacle-avoidance Function is implemented by the ultrasonic module. The principle of ultrasonic distance measurement is that the transmitter emits ultrasonic waves in a certain direction and starts timing at the same time of the launch moment. The ultrasonic waves propagate in the air, and on the way to meet the obstacle immediately returns. Then the ultrasonic receiver stops timing immediately when receives the reflected wave. According to the recorded time and the known condition that ultrasonic propagation speed in the air is 340m/s, the distance between the ultrasonic module and the obstacle can be obtained.

Please open the last folder for details:Related chip information -> Conqueror-Shield



Ultrasonic

It can be seen from the figure that the ultrasonic module is connected to D12 and D13 on the UNO board. Then please open [Demo1](#):

```
// in DeviceDriverSet_xxx0.h

class DeviceDriverSet_ULTRASONIC
{
public:
    void DeviceDriverSet_ULTRASONIC_Init(void);
    void DeviceDriverSet_ULTRASONIC_Test(void);
    void DeviceDriverSet_ULTRASONIC_Get
    (uint16_t *ULTRASONIC_Get /*out*/);

private:
#define TRIG_PIN 13
// Arduino pin tied to trigger pin on the ultrasonic sensor.
#define ECHO_PIN 12
// Arduino pin tied to echo pin on the ultrasonic sensor.
#define MAX_DISTANCE 200
// Maximum distance we want to ping for (in centimeters).
// Maximum sensor distance is rated at 400-500cm.
};
```

Then, initial the pins of ultrasonic.

```
// in DeviceDriverSet_xxx0.cpp

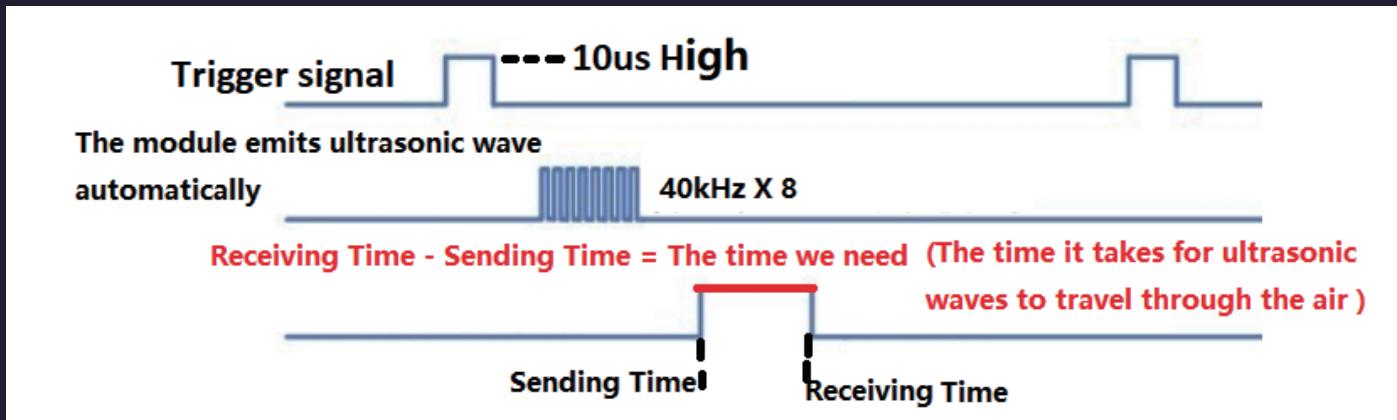
void
DeviceDriverSet_ULTRASONIC::DeviceDriverSet_ULTRASONIC_Init
(void)
{
    pinMode(ECHO_PIN, INPUT); //Ultrasonic module initialization
    pinMode(TRIG_PIN, OUTPUT);
}
```

And then call it in the Setup() function:

```
//in Demo1
void setup()
{
    Serial.begin(9600);
    myUltrasonic.DeviceDriverSet_ULTRASONIC_Init();
}
```

Next, let's look at the flow of acquiring the measurement data through ultrasonic module.

- 1、Use Arduino to employ digital pin to give at least 10 μ s high level signal to Trig pin of SR04 to trigger the distance measurement function of SR04 module.
- 2、After triggering, the module will automatically send eight 40KHz ultrasonic pulses and detect whether there is a signal return. This step is done internally by the module automatically.
- 3、If a signal returns, the Echo pin will output high level, and the duration of the high level is the time from launch to return of the ultrasound. At this point, we can use the `pulseIn()` function to obtain the distance measurement data, and calculate the actual distance from the object being measured.



We will use the `pulseIn(pin,value)` function during the programming.

`pulseIn(pin, value)`: The pulse width used to detect the high and low level output of the pin, i.e., **the time we need**.

Pin: The pin used to obtain the pulse.

Value: Type of pulse, HIGH or LOW

Since the default unit of the `pulseIn()` function is us, and the propagation velocity of ultrasonic waves in the air is about 340m/s, unit conversions is needed.

Because,

the distance traveled by the ultrasonic wave = the distance from the launch point to the end + the distance from the end back to the launch point.

Therefore,

the distance traveled by the ultrasonic wave

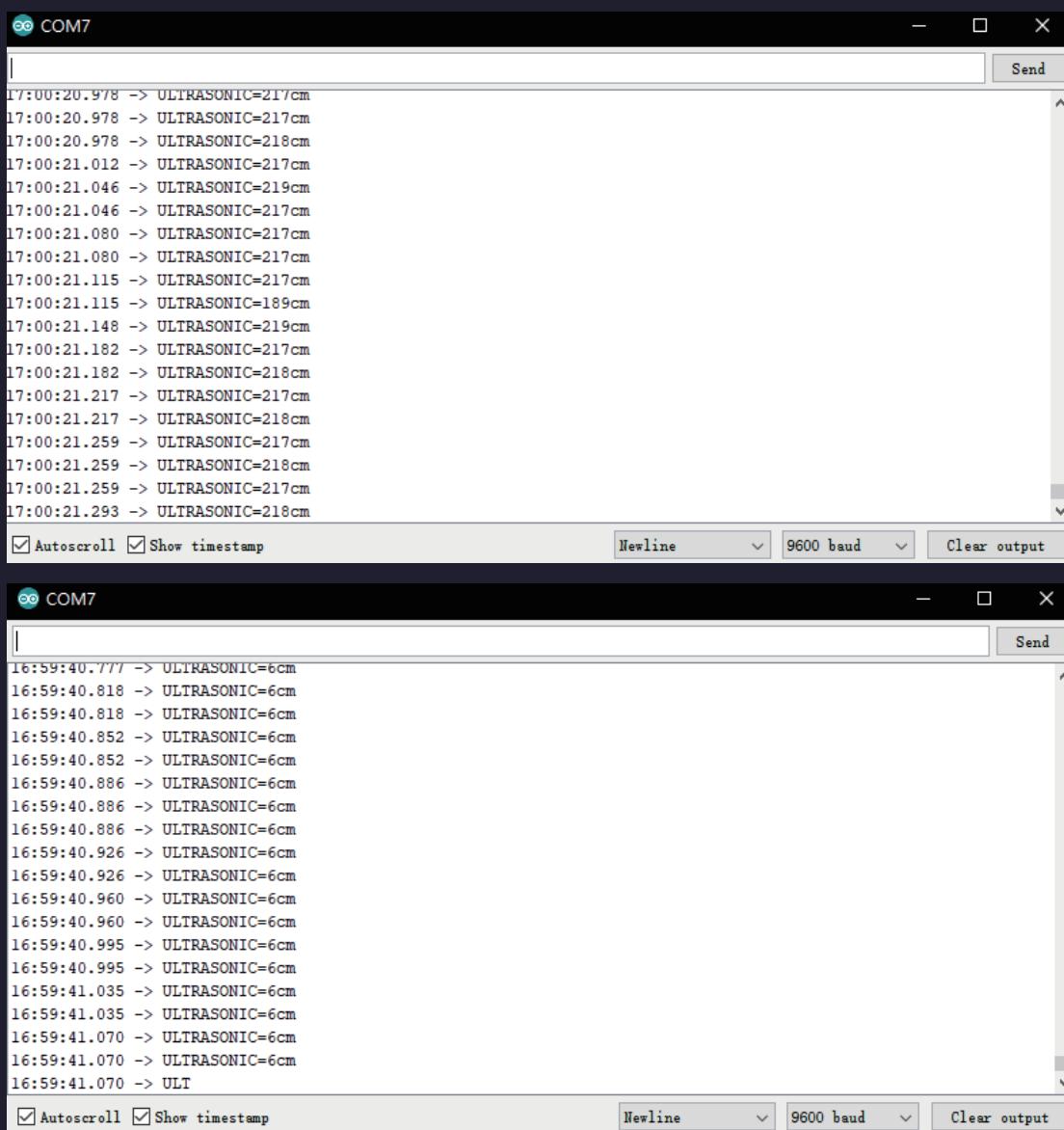
= `pulseIn() / 29.15 / 2` \approx `pulseIn() / 58`

```
// in DeviceDriverSet_xxx0.cpp
```

```
void DeviceDriverSet_ULTRASONIC::DeviceDriverSet_ULTRASONIC_Test(void)
{
    unsigned int tempda = 0;
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    tempda = ((unsigned int)pulseIn(ECHO_PIN, HIGH) / 58);

    Serial.print("ULTRASONIC=");
    Serial.print(tempda); // Convert ping time to distance and print result
    (0 = outside set distance range, no ping echo)
    Serial.println("cm");
}
```

Upload program. (Please toggle the “Upload-Cam” button to “Upload” when uploading the program.) Please open the serial port after the program has been uploaded successfully. If there is no obstacle in front, the ultrasonic distance will be shown as more than 200. And if you try to put your hand in front of the ultrasonic module, it will show the distance between your hand and the ultrasonic module.



```
17:00:20.978 -> ULTRASONIC=217cm
17:00:20.978 -> ULTRASONIC=217cm
17:00:20.978 -> ULTRASONIC=218cm
17:00:21.012 -> ULTRASONIC=217cm
17:00:21.046 -> ULTRASONIC=219cm
17:00:21.046 -> ULTRASONIC=217cm
17:00:21.080 -> ULTRASONIC=217cm
17:00:21.080 -> ULTRASONIC=217cm
17:00:21.115 -> ULTRASONIC=217cm
17:00:21.115 -> ULTRASONIC=189cm
17:00:21.148 -> ULTRASONIC=219cm
17:00:21.182 -> ULTRASONIC=217cm
17:00:21.182 -> ULTRASONIC=218cm
17:00:21.217 -> ULTRASONIC=217cm
17:00:21.217 -> ULTRASONIC=218cm
17:00:21.259 -> ULTRASONIC=217cm
17:00:21.259 -> ULTRASONIC=218cm
17:00:21.259 -> ULTRASONIC=217cm
17:00:21.293 -> ULTRASONIC=218cm
```

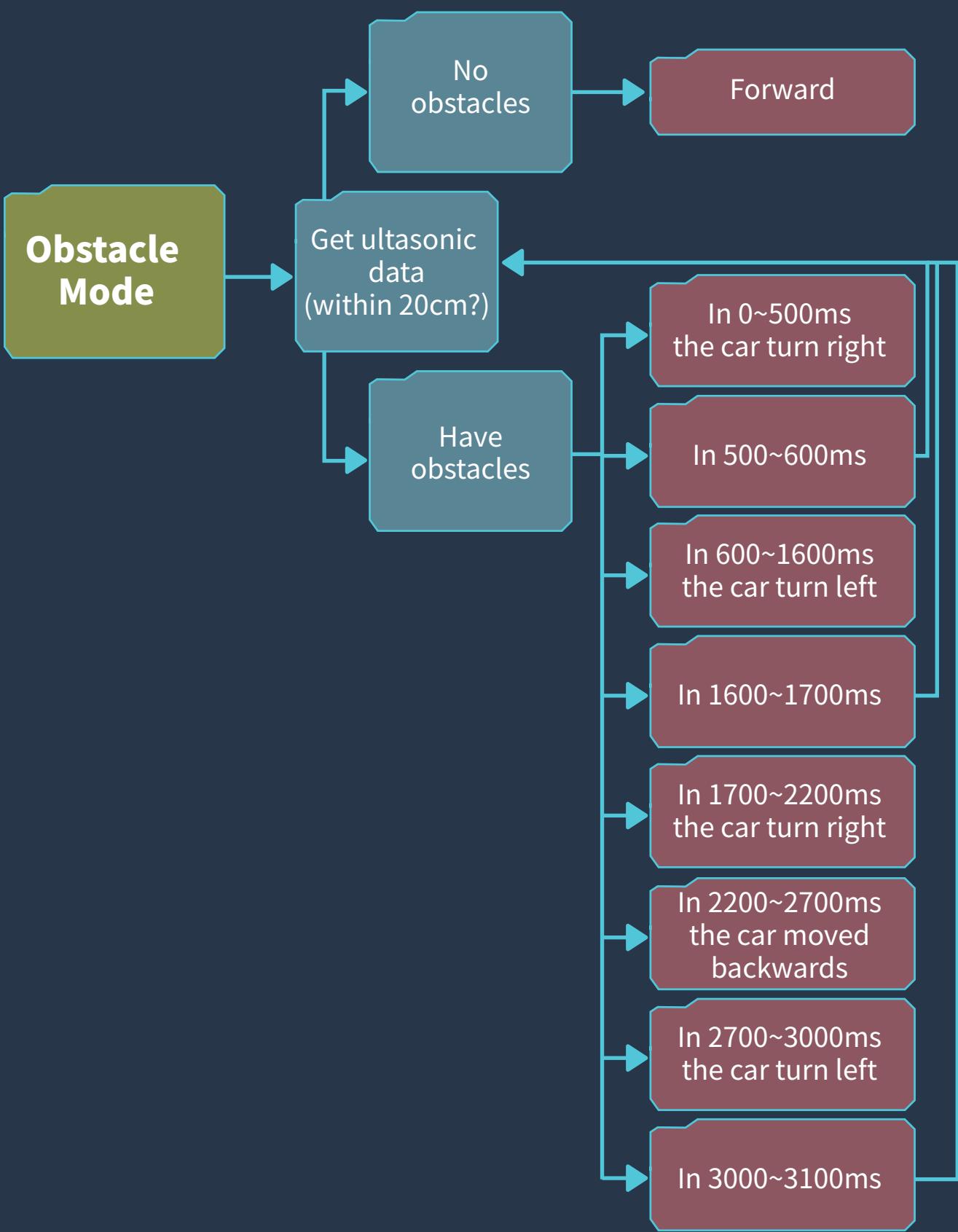
Autoscroll Show timestamp Newline 9600 baud Clear output


```
16:59:40.777 -> ULTRASONIC=6cm
16:59:40.818 -> ULTRASONIC=6cm
16:59:40.818 -> ULTRASONIC=6cm
16:59:40.852 -> ULTRASONIC=6cm
16:59:40.852 -> ULTRASONIC=6cm
16:59:40.886 -> ULTRASONIC=6cm
16:59:40.886 -> ULTRASONIC=6cm
16:59:40.926 -> ULTRASONIC=6cm
16:59:40.926 -> ULTRASONIC=6cm
16:59:40.960 -> ULTRASONIC=6cm
16:59:40.960 -> ULTRASONIC=6cm
16:59:40.995 -> ULTRASONIC=6cm
16:59:40.995 -> ULTRASONIC=6cm
16:59:41.035 -> ULTRASONIC=6cm
16:59:41.035 -> ULTRASONIC=6cm
16:59:41.070 -> ULTRASONIC=6cm
16:59:41.070 -> ULTRASONIC=6cm
16:59:41.070 -> ULT
```

Autoscroll Show timestamp Newline 9600 baud Clear output

At this point, the ultrasonic module driver is complete. Now, let's take a look at the implementation principle of Obstacle-avoidance Function using the ultrasonic module, and then analyze the program:

implementation principle of Obstacle-avoidance Function



Please open **Demo2** in the current folder:

Let's look at the definition of the relative function and variables of Obstacle-avoidance Function:

```
// in ApplicationFunctionSet_xxx0.h
#ifndef _ApplicationFunctionSet_xxx0_H_
#define _ApplicationFunctionSet_xxx0_H_

#include <arduino.h>

class ApplicationFunctionSet
{
public:
    void ApplicationFunctionSet_Init(void);
    void ApplicationFunctionSet_Obstacle(void);

private:
    volatile uint16_t UltrasoundData_mm;
    volatile uint16_t UltrasoundData_cm;
    boolean UltrasoundDetectionStatus = false;
public:
    boolean Car_LeaveTheGround = true;
    const int ObstacleDetection = 20;
};

extern ApplicationFunctionSet Application_FunctionSet;
#endif
```

Then, as for writing the program of Obstacle-avoidance Function, the situation described on the flow chart when encountering obstacles can be listed down one by one.

```
// in ApplicationFunctionSet_xxx0.cpp
void
ApplicationFunctionSet::ApplicationFunctionSet_Obstacle(void)
{.....}
```

Upload program. (Please toggle the “Upload-Cam” button to “Upload” when uploading the program.) After the program has been uploaded successfully, please place the car on the ground steadily and then turn on the switch. The Conqueror Robot Car will moving forward by default if there is no obstacle in front. And it will implement the Obstacle-avoidance Function intelligently depends on different situations when encounters the obstacle.