MARMARA UNIVERSITY

Microprocessor Systems
EE2004

PROJECT REPORT

CEM ÜNSALAN

Ömer Sabri EMEKSİZ	150721841

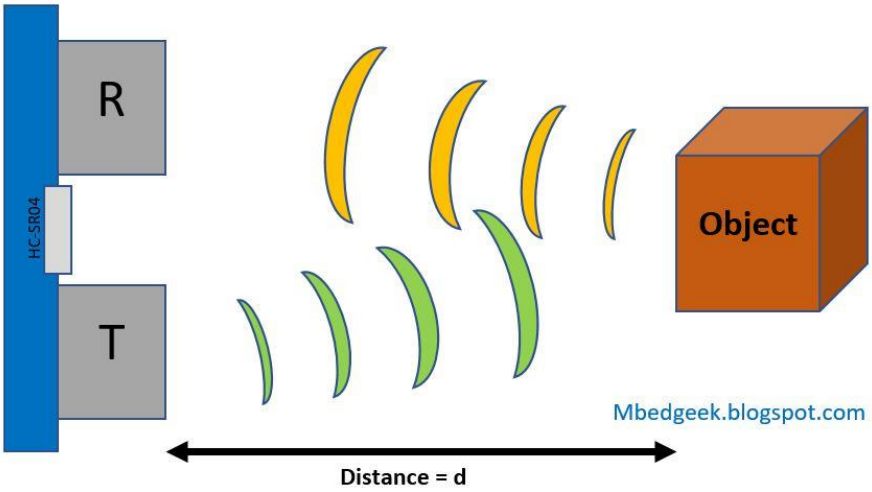
CONTENTS

How the Ultrasonic Distance Sensor Works?

In this project, I will use the HC-SR04 ultrasonic distance sensor. The features and working principles that I will describe in the following sections may vary in different distance sensors.

The Working Principle of HC-SR04 ultrasonic distance sensor is as follows:

It emits an ultrasound at 40 000 Hz that circulates through the air and returns to the module if there is an object or obstacle in its path. You can calculate the distance by considering the travel time and the speed of sound.



The Distance Calculation can be done as follow:

$$Time = \frac{Distance}{Speed\ of\ Sound}$$

By using this formula, we can obtain.

$$Distance = Time \times Speed\ of\ Sound$$

As you can see here, we can calculate the distance if we can measure the time taken for the ultrasound to go and come back. Therefore, we use time here to represent the measured time difference for the ultrasound waves to travel back and forth.

As we know the Speed of Sound is 340 m/s = 34 cm/s = 0.034 cm/µs.

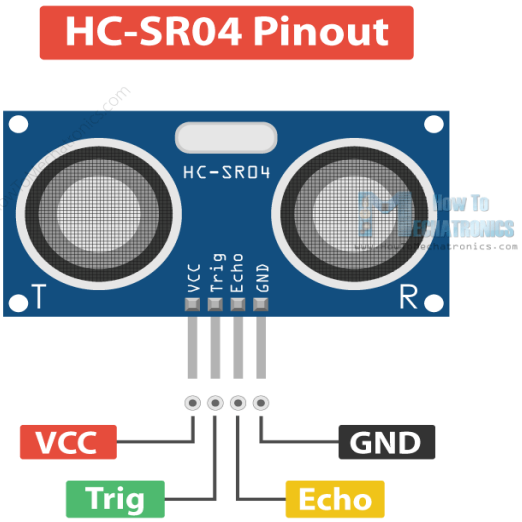
We convert the speed of sound in terms of cm/µs because of we need to measure microseconds for the sensor.

There is a trick to consider at this point, dividing the time by 2 because we measure the time it takes for the sound wave to travel to the object and back, but we only need one time between the sound waves' travels to calculate the distance. So, the result formula can be written as follow:

$$Distance = (Time \times Speed\ of\ Sound)/2$$

The HC-SR04 Ultrasonic Sensor Pinout:

The sensor has 4 pins. These are VCC, GND, Trig and Echo. Here's the pinout of the sensor:



The VCC pin is connected to 5V or 3V on the STM32L073RZ microcontroller and the GND pin is connected to the ground pin.

We should connect to Trick pin any pin on the board, but we should set the connected pin as GPIO_OUTPUT.

Also, we should connect to Echo pin any pin on the board, but we should set the connected pin as GPIO_INPUT.

The reason why we connect these pins as GPIO_INPUT/GPIO_OUTPUT as follow:

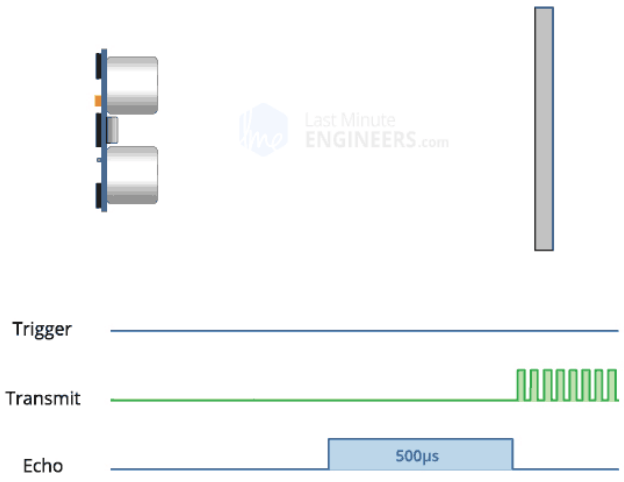
How the HC-SR04 Ultrasonic Sensor Generate/Capture Ultrasound:

In order to generate the ultrasound, we need to set the Trig pin on a High State for 10 µs. As you simply understand using Trig pin, we send the ultrasonic wave from the transmitter as a result we need to use Trig pin as a output.

Setting Trig pin on a High State for 10 µs will send out an 8-cycle ultrasonic burst which will travel at the speed of sound.

At this stage, the Echo pins goes high right away after that 8-cycle ultrasonic burst is sent, and it starts listening or waiting for that wave to be reflected from an object. As you can see we are using the Echo pins as a input for our system.

The ultrasound generation can be seen by following animation:



STM32CubeIDE Settings

The HC-SR04 Ultrasonic Sensor Settings:

- Pinout and Configuration System Core > RCC > Set High Speed Clock (HSE) as Crystal/Ceramic.
- Clock Configuration > Choose PLLCLOCK >> Set HCLK (MHz) to 24.
- Pinout and Configuration > Timer > TIM2 > Set Clock Source as Internal Clock > Set Prescaler as 23.
- Set PA9 as GPIO_Output (Trig Pin).
- Set PA8 as GPIO_Input (Echo Pin).

The Servo Motor Settings:

- Pinout and Configuration > Timer > TIM3 > Set Clock Source as Internal Clock > Set Prescaler 355 >> Set Channel3 PWM Generation CH3.
- Connect Servo Motor pin to A3 by manually.
- Connect Servo Motor GND and VCC to GND and VCC on the board respectively.

The Other Settings:

- Set PB10 as GPIO_Output (LED1).
- Set PB4 as GPIO_Output (LED2).
- Set PB5 as GPIO_Output (LED3).
- Set PC7 as GPIO_EXTI (BUTTON).
- Pinout and Configuration System Core > System Core > GPIO > Set PC7 GPIO mode as External Interrupt Mode with Rising edge trigger detection > Set GPIO Pull-up/Pull-down as Pull-down.

Why We Set 23 as a Prescaler Value?

We need to measure microseconds for the sensor.

$$Time = 1/Frequency$$

$$1\mu s = 1/1MHz$$

As simple seen in order to have a microsecond we need timer frequency as 1MHz. As you know from above, we set the timer frequency as 24 MHz. We need to divide it by 24 which should be the value of prescaler but prescaler always one less of it so in this case, it's value should be 23.

Aim of the Program

The purpose of the program is to toggle the LEDs and enable the servo motor to rotate at certain angles according to the distance calculated using the HC-SR04 Ultrasonic Sensor and the number of button presses. This setup can be thought of as a simulation of automatic opening and closing door systems. A system similar to the automatic door system can be designed with minor changes to the algorithm.

The working principle of the program is as follows:

- If the distance measured by the sensor is between 0 cm and 10 cm and the number of button presses is 1, the first led will start toggle and continue in an infinite loop. At the same time, our engine turns 75 degrees counterclockwise from the first angle. If the distance goes beyond 0 to 10 or if the number of button presses is different than 1, the led stops toggling.
- If the distance measured by the sensor is between 10 cm and 20 cm and the number of button presses is 2, the second led will start toggle and continue in an infinite loop. At the same time, our engine turns 45 degrees counterclockwise from the previous case angle. If the distance goes beyond 10 to 20 or if the number of button presses is different than 2, the led stops toggling.
- If the distance measured by the sensor is between 20 cm and 30 cm and the number of button presses is 3, the third led will start toggle and continue in an infinite loop. At the same time, our engine turns 60 degrees counterclockwise from the previous case angle. If the distance goes beyond 20 to 30 or if the number of button presses is different than 3, the led stops toggling.
- As the last case, if the number of button presses is equal to or more than 4, the number of button presses and the motor rotation angle are reset, at the same time the motor angle returns to the starting condition.

Code Explanation

Necessary Variable Declarations:

```
/* USER CODE BEGIN PV */
uint32_t pMillis;
uint32_t Value1 = 0;
uint32_t Value2 = 0;
int Distance  = 0;  // cm
int counter = 0;
/* USER CODE END PV */
```

Motor Angle Changing and Callback Functions:

```
/* USER CODE BEGIN 0 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_7){
        counter++;
    }
}
void change_angle(int angle){
        angle=angle+45;
        __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_3,angle);
}
/* USER CODE END 0 */
```

Starting Timers and Setting Trig Pin of Sensor Low:

```
/* USER CODE BEGIN 2 */
HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);
HAL_TIM_Base_Start(&htim2);
 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);  // pull the TRIG pin low
/* USER CODE END 2 */
```

Codes Necessary for the Operation of the Sensor and the Program:

```
/* USER CODE BEGIN WHILE */
while (1)
{
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET);  // pull the TRIG pin HIGH
        __HAL_TIM_SET_COUNTER(&htim2, 0);
        while (__HAL_TIM_GET_COUNTER (&htim2) < 10);  // wait for 10 us
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);  // pull the TRIG pin low

        pMillis = HAL_GetTick(); // used this to avoid infinite while loop  (for timeout)
        // wait for the echo pin to go high
        while (!(HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_8)) && pMillis + 10 >  HAL_GetTick());
        Value1 = __HAL_TIM_GET_COUNTER (&htim2);

        pMillis = HAL_GetTick(); // used this to avoid infinite while loop (for timeout)
        // wait for the echo pin to go low
        while ((HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_8)) && pMillis + 50 > HAL_GetTick());
        Value2 = __HAL_TIM_GET_COUNTER (&htim2);

        Distance = (Value2-Value1)* 0.034/2;
        HAL_Delay(50);

        int angle = 0;

        if(Distance>=0 && Distance<10 && counter==1){
            HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_10);
            angle = angle + 30;
            change_angle(angle);
        }
        if(Distance>=10 && Distance<20 && counter==2){
                    HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_4);
                    angle = angle + 45;
                  change_angle(angle);
                }
        if(Distance>=20 && Distance<30 && counter==3){
                    HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_5);
                    angle = angle + 60;
                    change_angle(angle);
                }
        if(counter>=4){
            counter = 0;
            angle = 0;
            change_angle(angle);
        }
  /* USER CODE END WHILE */
```