

# **Development of a Radar System using Ultrasonic Sensor and Servo Motor Control with STM32 Microcontroller**

# **OUTLINE**

1. Objective
2. Introduction
3. Apparatus
4. Logic implemented
5. Pin Connection
6. Results
7. Discussion
8. Conclusion
9. References
10. Appendix

# **OBJECTIVE**

Develop a radar system using an STM32 microcontroller to detect nearby objects within a 180-degree range using an ultrasonic sensor mounted on a servo motor. When an object (enemy) is detected, the system will display the count of detected enemies on a 7-segment display, accompanied by a blinking red LED and a beep sound from a buzzer.

# **INTRODUCTION**

## **1. ULTRASONIC SENSOR**

An ultrasonic sensor is a device that uses ultrasound waves (sound waves with frequencies higher than the audible range of humans, typically above 20 kHz) to measure distance or detect the presence of objects.

### **1.1 BASIC PRINCIPLE OF ULTRASONIC SENSOR**

An ultrasonic sensor emits short bursts of high-frequency sound waves that travel through the air at the speed of sound. If these waves encounter an object, they bounce back as echo signals. By measuring the time, it takes for the emitted waves to return as echoes, the sensor can calculate the distance to the object. This is possible because the speed of sound in air is known, allowing for a straightforward calculation of distance based on the time delay between sending the signal and receiving the echo.

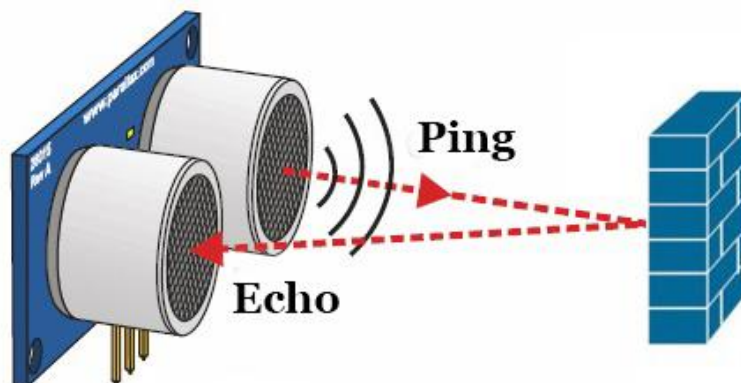


FIG 1: Ultrasonic Sensor Working

The sensor typically has four pins: Ground and VCC for power supply, Trigger for sending trigger pulses, and Echo for receiving echo pulses. To initiate a measurement, the Trigger pin is set to a high state for a brief period (e.g., 10  $\mu$ s), which triggers the sensor to emit ultrasonic waves. The sensor then listens for the echo signals and measures the time delay between sending the trigger pulse and receiving the echo, enabling precise distance determination based on the speed of sound. This basic principle allows ultrasonic sensors to accurately detect and measure distances without physical contact with objects.

## 2. FUTABA S3003 SERVO MOTOR

The Futaba S3003 servo is a popular and widely used standard servo motor manufactured by Futaba Corporation, a renowned brand in the field of radio control (RC) and robotics. The S3003 servo is designed for various applications where precise control of angular position is required, such as in RC models, robotics, animatronics, and other mechatronic systems.

### 2.1 PRINCIPLE OF SERVO MOTOR

The S3003 servo operates based on the principle of receiving electrical signals (typically PWM - Pulse Width Modulation) from a control system (e.g., microcontroller) to accurately position its output shaft to a desired angle within a specific range (typically 0 to 180 degrees).

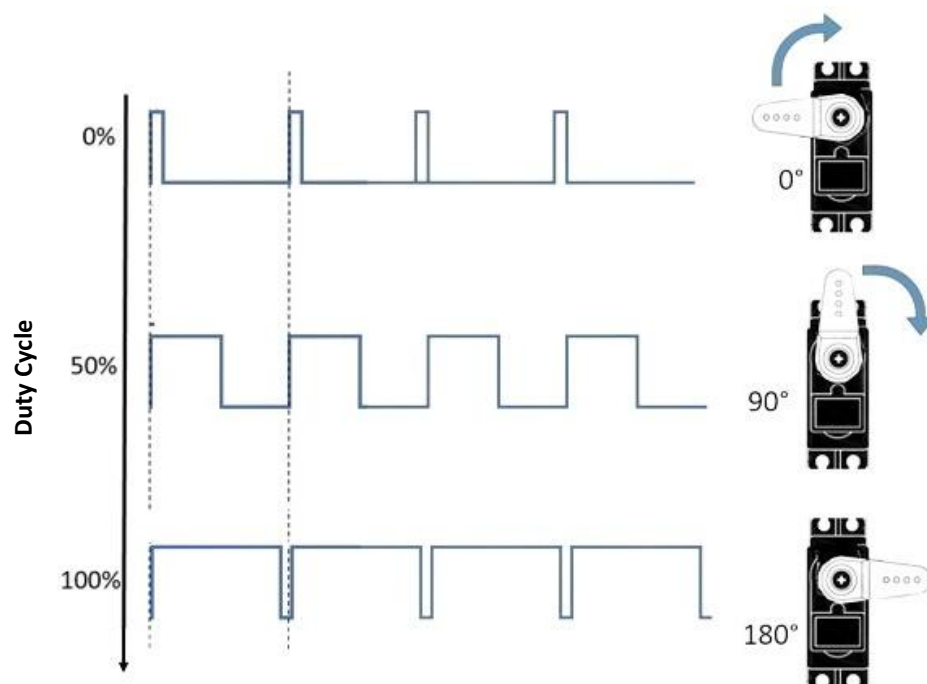





FIG 2: Servo Motor Calibration

## 2.2 CONTROL INTERFACE

It is controlled via a standard three-wire interface:

- **Ground (GND):** 
- **Power supply (5V or VCC):** 
- **Signal (PWM output):** 

## 3. BUZZER

The buzzer is an electromechanical device that generates sound when an electrical signal is applied to it. It is commonly used for audio alerting purposes in electronic circuits. The buzzer emits audible sound based on the electrical signal it receives.

### 3.1 CONTROL INTERFACE

It is controlled via two pins interface:

- **Signal (Control Pin):** Connects to a digital output pin of the microcontroller.
- **Voltage Level:** Apply a logic HIGH to activate the buzzer.

## 4. LED

An LED (Light-Emitting Diode) is a semiconductor device that emits light when current flows through it. LEDs are widely used in electronics for indication, illumination, and display purposes.

### 4.1 CONTROL INTERFACE

It is controlled via two pins interface

- **Signal (Control Pin):** Connects to a digital output pin of the microcontroller.
- **Voltage Level:** Apply a logic HIGH to illuminate the LED.
- **Current Limiting Resistor:** Ensure the LED is connected through a suitable current-limiting resistor (220 kilo-ohm in this case) to prevent excess current.

# APPARATUS



FIG 3: LEDs



FIG 4: Buzzer



FIG 5: Servo Motor

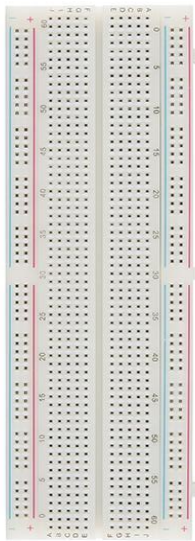


FIG 6: Breadboard



FIG 7: Jumper Wires



FIG 8: Resistors (220 kilo-ohm)

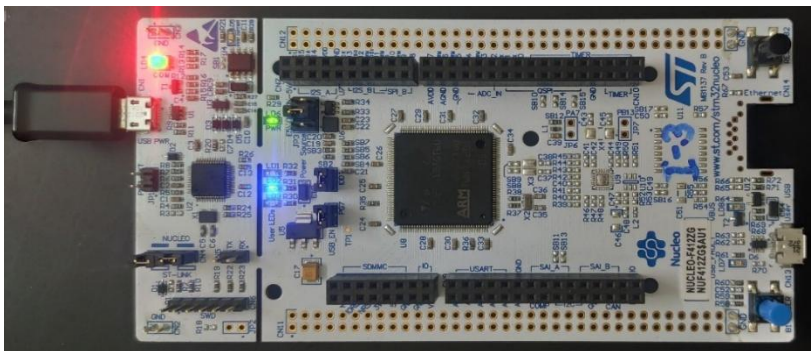


FIG 9: STM32



FIG 10: Ultrasonic Sensor

# **LOGIC IMPLEMENTED**

**1. Initialization:** Green LED is turned on to indicate the radar system is powered ON.

**2. Ultrasonic Sensor and Servo Motor:**

- Ultrasonic sensor is mounted on a servo motor.
- Servo motor continuously rotates the sensor in a range of 0 to 180 degrees, scanning for objects.

**3. Detection Logic:**

- When the ultrasonic sensor detects an object (enemy), the system triggers a response.

**4. Response:**

- Red LED: Blinks to visually indicate detection of an enemy.
- Buzzer: Emits a beep sound synchronized with the blinking of the red LED.

**5. Behaviour on Detection:** Upon detecting an enemy:

- Trigger the red LED to blink.
- Activate the buzzer to emit a beep sound synchronized with LED blinking.

**6. Continual Scanning:**

- The servo motor continues rotating, and the ultrasonic sensor continues scanning for enemies within its range.

## PIN CONNECTION

S.no	Device	STM32 Board(PIN )
1.	<b>Servo Motor</b>	
	Yellow (PWM)	PB5 (TIM3_CH2)
	Red (5V)	5 V
	Brown (GND)	GND
2.	<b>Ultrasonic Sensor</b>	
	VCC	5 V
	Trigger	PB7 (TIM4_CH2)
	Echo	PB3 (TIM2_CH2)
	Ground (GND)	GND
3.	<b>Green LED</b>	
	Positive	PD0
	Negative (GND)	GND
4.	<b>Red LED and Buzzer</b>	
	Positive	PA0
	Negative (GND)	GND

**\*NOTE:** All GND pins are shorted i.e. they are connected to common ground

## RESULTS

### **1. Scanning Mode:**

When the radar system is ON (indicated by the green LED), it enters the scanning mode where the ultrasonic sensor mounted on the servo motor continuously rotates from 0 to 180 degrees to detect any nearby objects (enemies).The servo motor rotates smoothly within its specified range (0 to 180 degrees), sweeping the ultrasonic sensor across this arc.



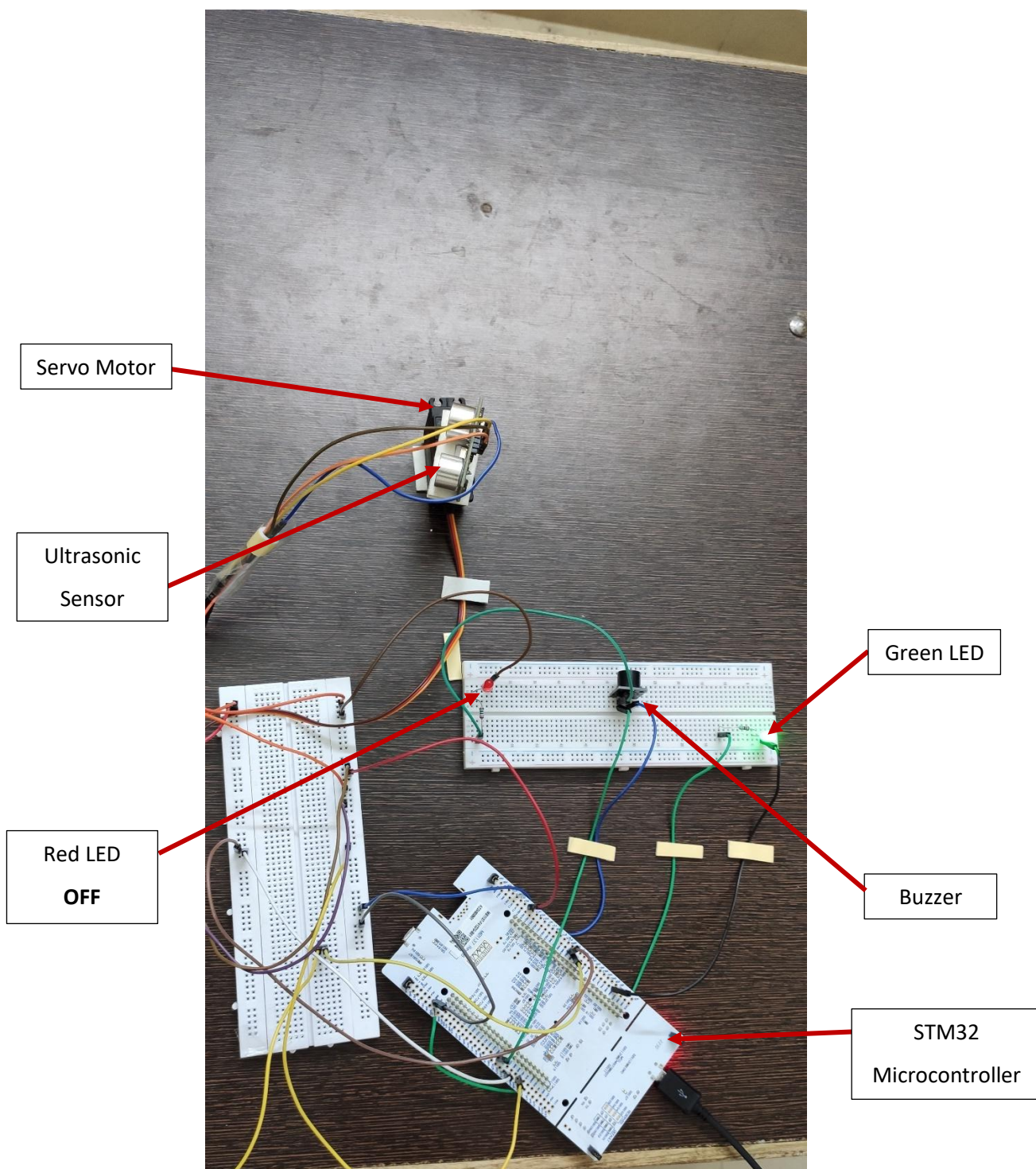


FIG 10: Setup in Scanning Mode

## 2. Object Detection:

In this mode, when an object is detected within the scanning range of the ultrasonic sensor, a **red LED** begins blinking to provide a clear indication of the detection event. Additionally, a **buzzer** emits a **beep sound** to complement the visual indicator, ensuring that the user receives both visual and auditory cues upon object detection.

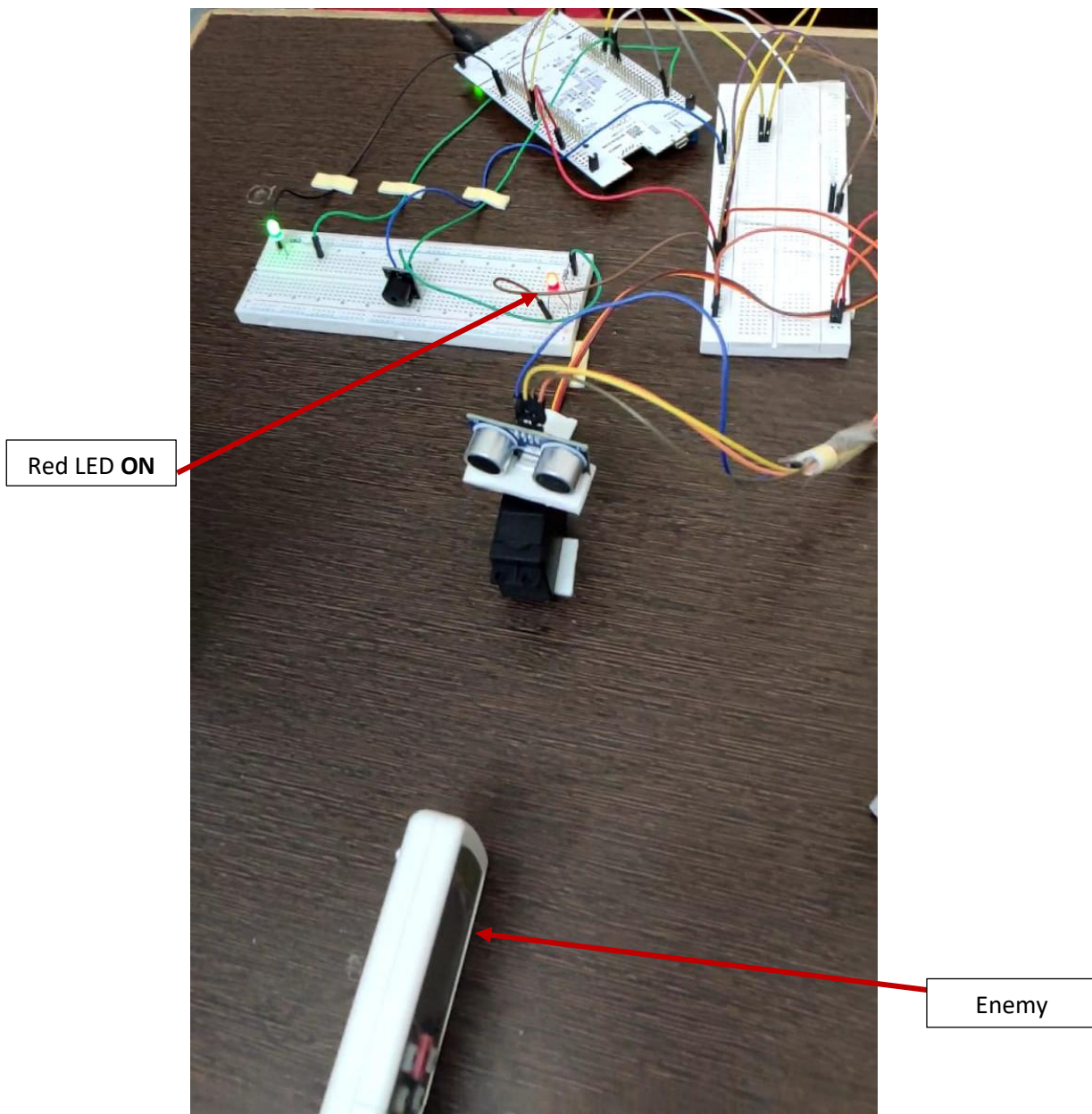


FIG 11: Setup in Detection Mode

## DISCUSSION

### **SETUP MOTOR:**

To control a servo motor with a required frequency of 50 Hz using an STM32 microcontroller running at a clock frequency of 16 MHz (from the HSI source), we utilize the Timer 3 (**TIM3**) by configuring its Prescaler (**PSC**) and Auto-Reload Register (**ARR**).

By calculating and setting appropriate values for **PSC** and **ARR** using the formula,

$$\frac{f_{\text{clk}}(\text{HSI})}{\text{Required Frequency}} = \text{PSC} \times \text{ARR}$$

$$\frac{16,000,000}{50} = 320,000$$

TIM3->PSC = 320 - 1;  
TIM3->ARR = 1000 - 1; // 50Hz

Here, we ensure that Timer 3 generates a 50 Hz signal suitable for servo motor control.

### **SETUP ULTRASONIC SENSOR:**

To operate the ultrasonic sensor effectively, we need to generate a PWM (Pulse Width Modulation) pulse with a high duration of 10 $\mu$ s at the trigger pin. This pulse is crucial for triggering the ultrasonic sensor to start measuring distances.

TIM4->PSC = 160 - 1;  
TIM4->ARR = 10001 - 1; //10Hz  
TIM4->CCR2 = 1; //GENERATE PWM PULSE WITH 10  $\mu$ s HIGH

Additionally, when we receive the echo signal bounced back from an object, we need to accurately measure the time duration in  $\mu$ s.

TIM2->PSC = 16 - 1; //1Mhz or 1  $\mu$ s scaling

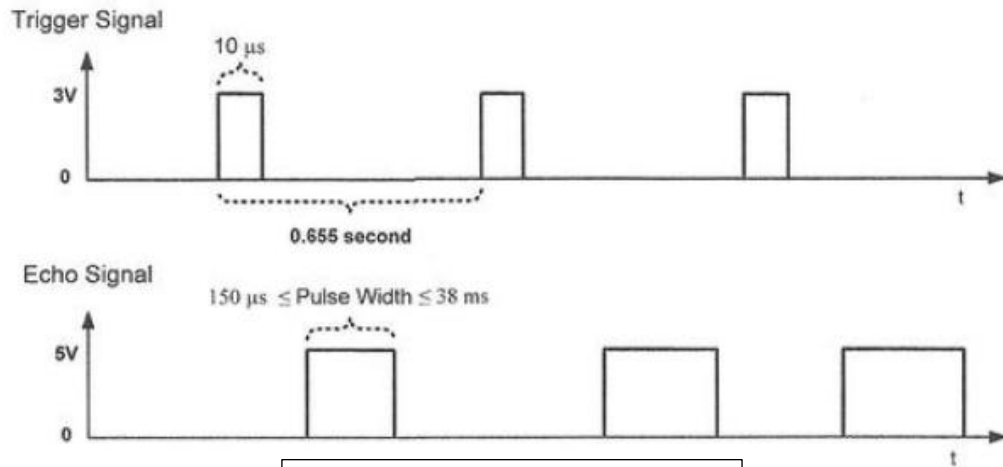


FIG 12: Trigger and Echo Signals

After receiving echo signals, we measure the pulse width to calculate the distance in cm:

$$\text{Distance} = \frac{\text{Pulse Width } (\mu\text{s})}{58} \text{ cm}$$

Based on the above distance estimate, we set the **range to 25 cm** for object (enemies) detection.

### SETUP PERIPHERALS:

We configured 2 GPIO ports:

- GPIOA port: This port is used for Red LED and Buzzer
- GPIOD port: This port is used for green LED

## **CONCLUSION**

This project demonstrates the implementation of a radar system using an STM32 microcontroller. The system utilizes an ultrasonic sensor mounted on a servo motor to detect nearby objects. When an object is detected within the scanning range, the system provides visual and auditory feedback to indicate the presence and count of enemies.

## **REFERENCES**

1. <https://www.circuitgeeks.com/hc-sr04-ultrasonic-sensor-with-arduino/>
2. <https://www.meccanismocomplesso.org/en/arduino-servo-motors-how-they-work-and-how-to-control-them/>
3. Embedded Systems with Arm Cortex-M Microcontrollers in Assembly Language and C: Third Edition by Yifeng Zhu (Author)

## **APPENDIX** (code...)

```
#include "stm32f412zx.h" // Device header
```

```
static uint32_t Val1 = 0;
```

```
static uint32_t Val2 = 0;
```

```
static uint32_t pulse = 0;
```

```
static uint8_t First_captured = 0;
```

```
static int distance;
```

```
void DELAY (int dd)
```

```
{
```

```
    for (;dd>0;dd--)
```

```
    {
```

```
        for(int i=0;i<3000;i++);
```

```
    }
```

```
}
```

```
void PERIPHERAL_SETUP(void)
```

```
{
```

```
    //LED AND BUZZER SETUP
```

```
    RCC->AHB1ENR |=0x1; // Enables Clock of PORT A
```

```
    GPIOA->MODER |= 0x1; //BUZZER, RED LED as OUTPUT
```

```
    //GREEN LED FROM START
```

```
    RCC->AHB1ENR |=0x8; // Enables Clock of PORT D
```

```
    GPIOD->MODER |=0x1; // making PD0 as ouput
```

```
    GPIOD->ODR=0x1;//TURNING ON GREEN LED FROM START
```

```
}
```

```

void MOTOR_SETUP(void)
{
// Enable GPIOB for generating pulse
RCC->AHB1ENR |= 0x2; // Enable GPIOB
GPIOB->MODER |= 0x800; // Set PB5 to AF2 mode
GPIOB->AFR[0] |= 0x200000; // AF2 for TIM3_CH2

// Generating PWM pulse in TIM4_CH2
RCC->APB1ENR |= 0x2; // Enable TIM3
TIM3->CCMR1 |= 0x6000; // PWM mode 1 for CH2
TIM3->CCER |= 0x10; // Enable compare mode for CH2
TIM3->SR = 0;

// Setup TIMER4 CH2
TIM3->PSC = 320 - 1;
TIM3->ARR = 1000 - 1; // 50Hz
TIM3->CR1 = 1; // Enable TIM3
}

```

```

void ULTRASONIC_SETUP(void)
{
//ENABLE GPIOB FOR GENERATING PULSE
//RCC->AHB1ENR |= 0x2; //Enable GPIOB for generating pulse
GPIOB->MODER |= 0x8000; //AF mode for PB7
GPIOB->AFR[0] |= 0x20000000; //AF2 for TIM4

//GENERATING PWM PULSE IN TIM4_CH2
RCC->APB1ENR |= 0x4; //ENABLE TIM4
}

```

**TIM4->CCMR1 |= 0x6000; //FOR CH2, PWM MODE 1**

**TIM4->CCER |= 0x10; //ENABLE COMPARE**

**TIM4->SR = 0;**

**//SETUP TIMER CH2**

**TIM4->PSC = 160 - 1; //0.1MHz**

**TIM4->ARR = 10001 - 1;**

**TIM4->CCR2 = 1; //GENERATE PWM PULSE WITH 10us HIGH**

**TIM4->CR1 = 1;**

**//-----//**

**//RECIEVING ECHO PULSE**

**//PB3--TIM2-CH2--AF1**

**//RCC->AHB1ENR |= 0x2; //ENABLE GPIOB**

**GPIOB->MODER |= 0x80; //AF MODE FOR PB3**

**GPIOB->AFR[0] |= 0x1000; //AF1 for TIM2\_CH2 WITH PB3**

**//ENABLE TIMER FOR CAPTURE MODE**

**RCC->APB1ENR |= 0x1; //ENABLE TIM2**

**TIM2->CCMR1 |= 0x100; //INPUT CAPTURE MODE IC1 => TI1**

**TIM2->CCER |= 0xB0; //CCE1 input enabled and 00 for measuring rising  
and falling edge**

**TIM2->DIER |= 0x4;**

**TIM2->SR |= 0x0;**

**//SETUP TIMER CH1**

**TIM2->PSC = 16 - 1;**

**TIM2->ARR = 65536 - 1; // 15.25 Hz**

**TIM2->CR1 |= 0x1; //TURN ON TIMER**

**NVIC\_EnableIRQ(TIM2\_IRQn);**



```
}
```

```
void ENEMY_DETECTED(int X)
```

```
{
```

```
    switch(X)
```

```
    {
```

```
        case 1: GPIOA->ODR |= 0x1; //BUZZER and RED LED TOGGLE
```

```
ON
```

```
        break;
```

```
        case 0: GPIOA->ODR = 0x00000000; // Set all bits of GPIOA ODR
```

```
to 0
```

```
        break;
```

```
    }
```

```
}
```

```
//-----MAIN LOGIC-----//
```

```
int main(void)
```

```
{
```

```
    PERIPHERAL_SETUP();
```

```
    MOTOR_SETUP();
```

```
    ULTRASONIC_SETUP();
```

```
    while(1)
```

```
    {
```

```
        // Increasing duty cycle from 0 to 60
```

```
        for (int duty = 20; duty <= 120; duty++)
```

```
        {
```

```
            TIM3->CCR2 = duty;
```

```

        DELAY(55); // Adjust delay time as needed for servo
response
    }

    // Decreasing duty cycle from 60 back to 0
    for (int duty = 120; duty >= 20; duty--)
    {
        TIM3->CCR2 = duty;
        DELAY(55); // Adjust delay time as needed for servo
response
    }
}

```

```

void TIM2_IRQHandler(void)
{
    // Check if the capture/compare 1 interrupt flag is set
    TIM2->SR &= ~TIM_SR_CC2IF; // Clear the interrupt flag

    if (First_captured%2==0)
    {
        Val1 = TIM2->CCR2;
        First_captured++;
    }

    else if (First_captured%2==1)
    {
        Val2 = TIM2->CCR2;
    }
}

```

```

        if (Val2 > Val1)
        {
            pulse = Val2 - Val1;
        }
        else
        {
            pulse = (TIM2->ARR - Val1) + Val2 + 1;
        }

        distance = pulse * (0.034/2) ;

        if (distance <= 25)
        {
            ENEMY_DETECTED(1);
            DELAY(7);
            ENEMY_DETECTED(0);
        }
        First_captured++;
    }

}

```