



Automatic door system

Using Ultrasonic Sensor and Servo Motor

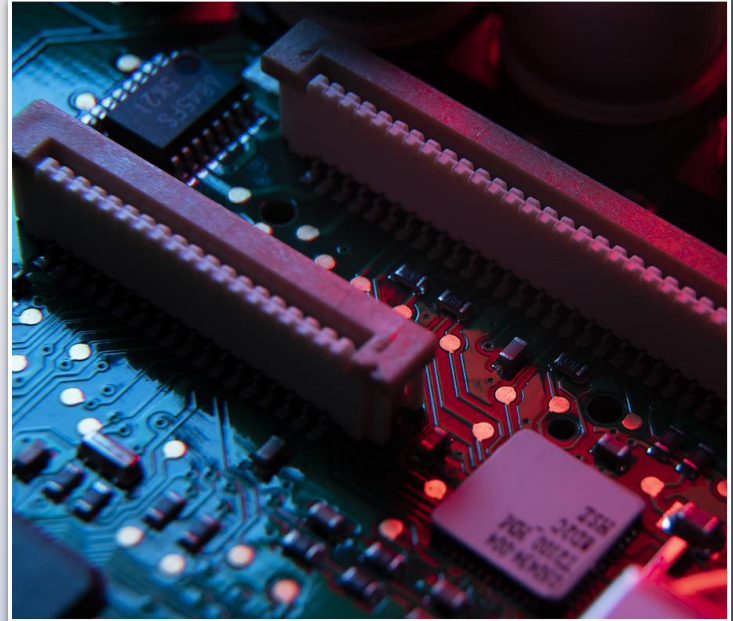


Table of contents

01

Introduction

02

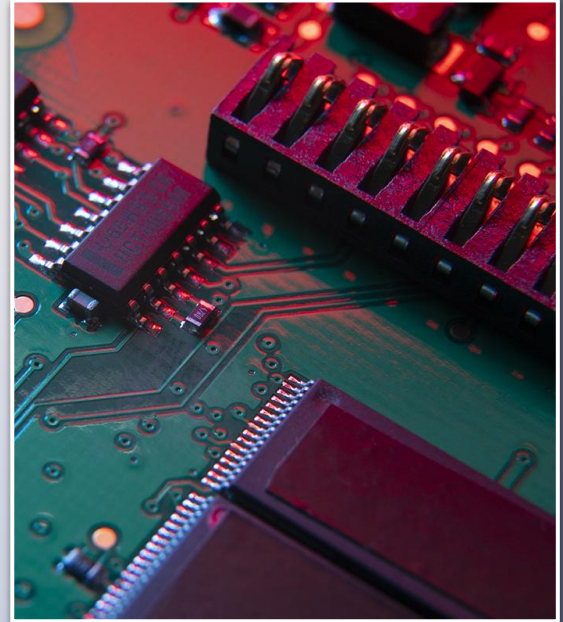
Circuit & Components

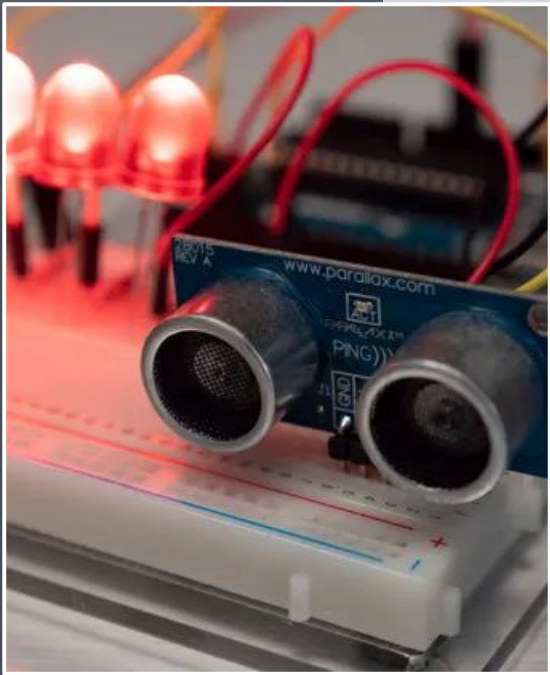
03

Code

04

Simulation & Demo





...

01

Introduction

Overview

Purpose : To design an automatic door control system using an ultrasonic sensor to detect the presence of an object and a servo motor to open or close the door accordingly.

Abstract: This project uses distance measurement to control the door's opening and closing, providing a hands-free operation for convenience and safety.





02

Circuit & Components





Components List

Microcontroller

Arduino Uno

Servo Motor

Controls the door
movement

Ultrasonic Sensor

Measures the distance

Jumpers

Used to connect
components

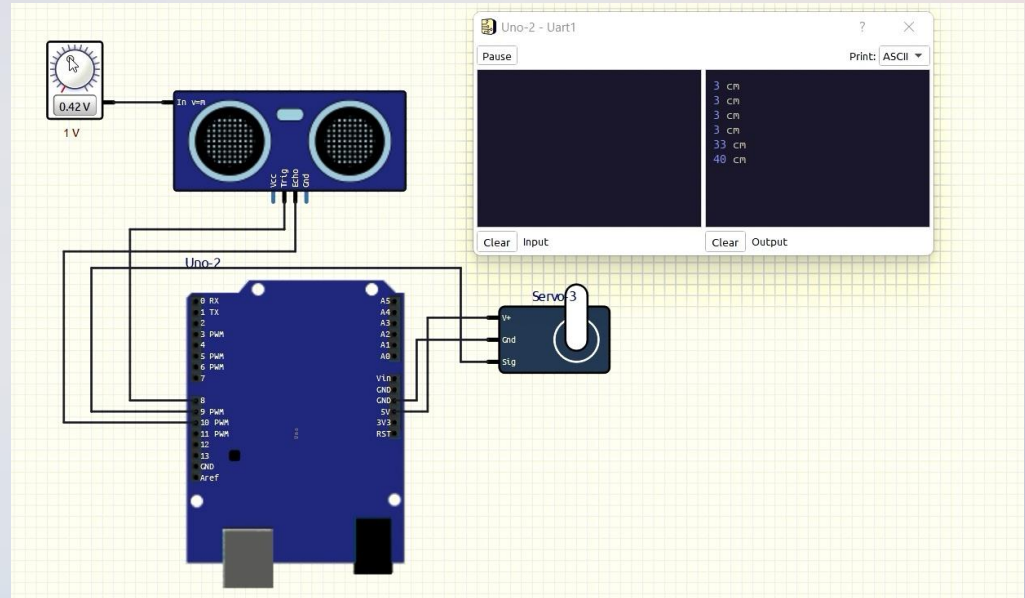
Usb cable

Works as power supply
and running code



Connection

- **Servo motor connected to PB1**
- **Ultrasonic sensor trigger pin connected to PB0**
- **Ultrasonic sensor echo pin connected to PB2**
- **Ground and Vcc standard as Shown in the schematic**



Ultrasonic Sensor

Principle

The ultrasonic sensor works by emitting an ultrasonic wave and measuring the time it takes for the echo to return after reflecting off an object.

Operation

1. Triggering the Pulse:

1. A 10-microsecond pulse is sent to the TRIG pin.
2. The sensor emits an 8-cycle burst of ultrasonic sound waves at 50 kHz.

2. Receiving the Echo:

1. The ECHO pin goes high when the pulse is sent and stays high until the echo is received back.



Measuring distance

The speed of sound in air at room temperature (around 20°C) is approximately 343 m/s or 34300 cm/s.

However, since we are measuring the time for the ultrasonic pulse to travel to the object and back (round-trip), we need to halve this value.

$$\text{Effective speed} = \frac{34300 \text{ cm/s}}{2} = 17150 \text{ cm/s}$$

Accounting for the prescaler:

$$\text{Effective clock frequency} = \frac{16 \text{ MHz}}{8} = 2 \text{ MHz}$$

Overall, dividing the duration in microseconds by approximately 116.7 gives us the distance in centimeters.

$$\text{division factor} = \frac{17150 \text{ cm/s}}{2} \times \frac{1 \text{ s}}{10^6 \mu\text{s}}$$

$$\text{division factor} = \frac{17150}{2 \times 10^6}$$

$$\text{division factor} = \frac{17150}{2000000}$$

$$\text{division factor} \approx 0.008575 \text{ cm}/\mu\text{s}$$

Servomotor

Principle

- A servo motor is a rotary actuator that allows for precise control of angular position.
- It consists of a motor coupled with a sensor for position feedback.






Operation of Servomotor

1. Pulse Width Modulation (PWM):

1. The position of the servo motor is determined by the duration of the PWM signal.
2. Typical pulse duration:
 1. $1\text{ }\mu\text{s}$: 0 degrees (minimum position)
 2. $1.5\text{ }\mu\text{s}$: 90 degrees (mid position)
 3. $2\text{ }\mu\text{s}$: 180 degrees (maximum position)

2. Timer Configuration:

1. Timer1 on the Atmega328P is configured for Fast PWM mode.
 2. The ICR1 register is set to define a 20ms period (50Hz frequency).
 3. The OCR1A register is used to set the pulse width corresponding to the desired angle.
- 

03

Code

```
#include <avr/io.h>
#include <util/delay.h>

#define TRIG_PIN PB0 // Ultrasonic sensor trigger pin connected to PB6 (digital pin 8)
#define SERVO_PIN PB1 // Servo motor connected to PB1 (OC1A, digital pin 9)
#define ECHO_PIN PB2 // Ultrasonic sensor echo pin connected to PD7 (digital pin 10)

void initServo() {
    // Set the Servo pin as output
    DDRB |= (1 << SERVO_PIN);

    // Configure Timer1 for Fast PWM mode, 8-bit
    TCCR1A |= (1 << WGM11) | (1 << COM1A1);
    TCCR1B |= (1 << WGM13) | (1 << WGM12) | (1 << CS11); // Prescaler = 8

    // Set ICR1 to define the top value for the 20ms period (50Hz PWM)
    ICR1 = 39999; // ICR1 = 16 MHz / (8 * 50 Hz) - 1

    // Start with the servo at 0 degrees
    OCR1A = 999; // 1.5 ms pulse width for 0 degrees
}

void setServoPosition(uint8_t angle) {
    // Map the angle (0-180) to pulse width (1ms - 2ms)
    uint16_t pulseWidth = 999 + (angle * 22.22);

    // Set the pulse width to move the servo
    OCR1A = pulseWidth;
}

void initUltrasonicSensor() {
    // Set the TRIG_PIN as output and ECHO_PIN as input
    DDRA |= (1 << TRIG_PIN);
    DDRB &= ~(1 << ECHO_PIN);
}

long measureDistance() {
    // Send a trigger pulse to the ultrasonic sensor
    PORTB |= (1 << TRIG_PIN);
    _delay_us(10);
    PORTB &= ~(1 << TRIG_PIN);

    // Measure the duration of the echo pulse
    float duration = 0;
    TON1 = 0; // Reset Timer1
    while (!(PINB & (1 << ECHO_PIN))) { // Wait for the echo to start
    }
    TON1 = 0; // Reset Timer1
    while (PINB & (1 << ECHO_PIN)) { // Wait for the echo to end
        duration = TON1;
    }

    // Convert the duration to distance in cm speed
```

Code

Including necessary libraries and
defining used pins for

```
#include <avr/io.h>
#include <util/delay.h>

#define TRIG_PIN PB0    // Ultrasonic sensor trigger pin connected to PD6 (digital pin 8)
#define SERVO_PIN PB1   // Servo motor connected to PB1 (OC1A, digital pin 9)
#define ECHO_PIN PB2    // Ultrasonic sensor echo pin connected to PD7 (digital pin 10)
```

Code con't

Measuring distance using division
factor of 116.62 as calculated previously

```
36
37 long measureDistance() {
38     // Send a trigger pulse to the ultrasonic sensor
39     PORTB |= (1 << TRIG_PIN);
40     _delay_us(10);
41     PORTB &= ~(1 << TRIG_PIN);
42
43     // Measure the duration of the echo pulse
44     float duration = 0;
45     TCNT1 = 0; // Reset Timer1
46     while (!(PINB & (1 << ECHO_PIN))) { // Wait for the echo to start
47     }
48     TCNT1 = 0; // Reset Timer1
49     while (PINB & (1 << ECHO_PIN)) { // Wait for the echo to end
50         duration = TCNT1;
51     }
52
53     // Convert the duration to distance in cm speed
54     float distance = (duration / 116.62); // d = (34300 / 2) * ticks * (1/16 MHz)
55
56     return distance;
57 }
```

Code con't

Initialization for both the
Servomotor and the Ultrasonic sensor

```
8 void initServo() {
9     // Set the Servo pin as output
10    DDRB |= (1 << SERVO_PIN);
11
12    // Configure Timer1 for Fast PWM mode, 8-bit
13    TCCR1A |= (1 << WGM11) | (1 << COM1A1);
14    TCCR1B |= (1 << WGM13) | (1 << WGM12) | (1 << CS11); // Prescaler = 8
15
16    // Set ICR1 to define the top value for the 20ms period (50Hz PWM)
17    ICR1 = 39999; // ICR1 = 16 MHz / (8 * 50 Hz) - 1
18
19    // Start with the servo at 0 degrees
20    OCR1A = 999; // 1.5 ms pulse width for 0 degrees
21 }
```

```
void initUltrasonicSensor() {
    // Set the TRIG_PIN as output and ECHO_PIN as input
    DDRB |= (1 << TRIG_PIN);
    DDRB &= ~(1 << ECHO_PIN);
}
```

Code con't

The logic of the main function is as shown in the pseudocode

```
//psudocode for the logic of the main function
Initialize servo motor
Initialize ultrasonic sensor
while (1) {
    Measure distance using ultrasonic sensor

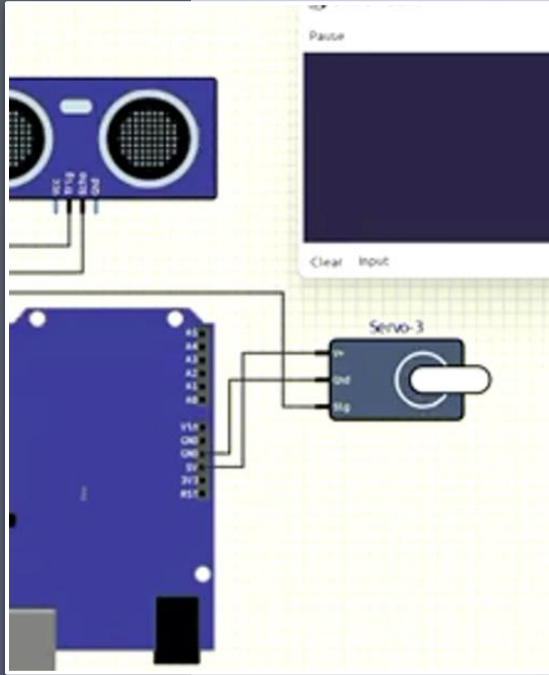
    if (distance > 0 cm and distance < 30 cm) {
        Set servo to 90 degrees (open position)
    } else {
        Set servo to 0 degrees (closed position)
    }

    Delay for 1 second
}
```


Code con't

Actual implementation of
the main function

```
67 int main(void) {
68     // Initialize the servo, ultrasonic sensor, and UART
69     initServo();
70     initUltrasonicSensor();
71
72     while (1) {
73         // Measure the distance
74         float distance = measureDistance();
75
76         // Control the servo based on distance
77         if (distance > 0 && distance < 30) {
78             setServoPosition(90); // Open position
79         } else {
80             setServoPosition(0); // Closed position
81         }
82
83         _delay_ms(1000); // Wait for 1 second before the next measurement
84     }
85 }
```

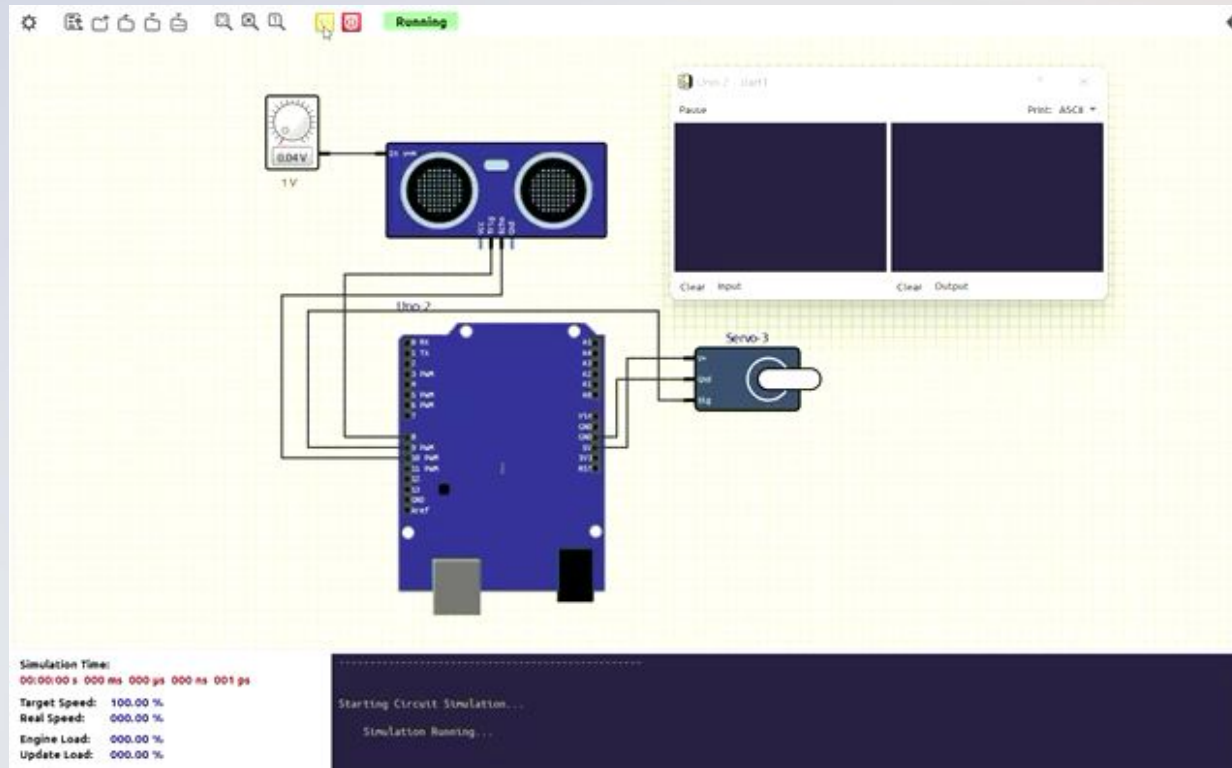


04

Simulation & Demo



Simulation



Demo





Thanks!

Preparation of :

Sally Zeineldeen - 120210008

Khaled gad - 120210024

