Ultrasonic Theremin

Distance-to-Frequency Mapping and
PWM Volume Control

Krishna Karthikeya Chemudupati

ATmega328PB Embedded Systems Project

October 2025

Contents

Introduction

A theremin is one of the earliest electronic musical instruments, traditionally controlled without physical contact by the performer. This project implements a digital theremin using an ATmega328PB microcontroller, where hand distance from an ultrasonic sensor controls the pitch and ambient light intensity controls the volume.

The system maps distance readings from an HC-SR04 ultrasonic sensor to musical note frequencies in the C6–C7 octave range (1046–2093 Hz), while a photoresistor provides analog input for PWM-based volume control. The instrument supports two operating modes: a **discrete mode** that quantizes distance into eight distinct musical notes, and a **continuous mode** that smoothly maps distance to frequency.

Key Features

- Distance-controlled pitch using HC-SR04 ultrasonic sensor (3–152 cm range)
- Ambient light-controlled volume via photoresistor
- 8 discrete musical notes (C6–C7, 1046–2093 Hz)
- Dual operating modes with button toggle (discrete/continuous)
- Real-time UART status output for debugging

Hardware

- ATmega328PB Xplained Mini development board (16 MHz)
- HC-SR04 / US-100 ultrasonic distance sensor
- Photoresistor (light-dependent resistor)
- Piezo buzzer / speaker
- Tactile push button for mode selection

Waveform Generation

Before building the theremin, three Timer0 waveform generation modes were explored to understand how to produce a precise 440 Hz (A4) reference tone.

Timer Clock Configuration

The ATmega328PB system clock runs at $f_{\text{CPU}} = 16$ MHz. With a Timer0 prescaler of 256:

$$f_{\text{timer}} = \frac{16 \times 10^6}{256} = 62{,}500 \text{ Hz} \tag{1}$$

Normal Mode (Interrupt-Based)

In Normal mode, the timer counts from 0 to 255 and overflows. An Output Compare Match interrupt toggles the output pin when the counter reaches `OCR0A`. The compare value is derived from:

$$\text{OCR0A} = \frac{f_{\text{CPU}}}{2 \times f_{\text{out}} \times N} - 1 = \frac{16 \times 10^6}{2 \times 440 \times 256} - 1 \approx 70 \tag{2}$$

```
#include <avr/io.h>
#include <avr/interrupt.h>

void Initialize(void) {
    DDRD |= (1 << DDD6);          // PD6 as output
```

```
    TCCR0A = 0x00;                  // Normal mode
    TCCR0B = (1 << CS02);           // Prescaler = 256
    TIMSK0 |= (1 << OCIE0A);       // Enable compare match interrupt
    OCR0A = 70;                     // Compare value for ~440Hz
    sei();                          // Enable global interrupts
}

ISR(TIMER0_COMPA_vect) {
    PORTD ^= (1 << PORTD6);        // Toggle PD6
    TCNT0 = 0;                      // Reset counter
}

int main(void) {
    Initialize();
    while (1);
}
```

Listing 1: Normal mode 440 Hz generation

CTC Mode (Clear Timer on Compare)

In CTC mode, the timer automatically resets to zero upon matching `OCR0A`, and the hardware toggles the OC0A pin. The frequency is:

$$f_{\text{out}} = \frac{f_{\text{CPU}}}{2 \times N \times (1 + \text{OCR0A})} \tag{3}$$

Solving for $f_{\text{out}} = 440$ Hz yields OCR0A $= 71$.

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

int main(void) {
    DDRD |= (1 << DDD6);   // PD6 (OC0A) as output
    // Timer0 in CTC mode (WGM01 = 1)
    TCCR0A = (1 << COM0A0) | (1 << WGM01);
    TCCR0B = (1 << CS02);  // Prescaler = 256
    OCR0A = 71;             // Compare value for 440 Hz
    while (1);              // Hardware handles toggling
}
```

Listing 2: CTC mode 440 Hz generation

Phase-Correct PWM Mode

Phase-Correct PWM with TOP = `OCR0A` (Mode 5, Table 18-9) produces a symmetric waveform. With OCR0A $= 35$, the output frequency is approximately 440 Hz.

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

void Initialize(void) {
    DDRD |= (1 << DDD6);
    PORTD |= (1 << PORTD6);
    // Phase Correct PWM, TOP = OCR0A
    TCCR0A |= (1 << WGM00);
    TCCR0A &= ~(1 << WGM01);
    TCCR0B |= (1 << WGM02);
    // Toggle OC0A on compare match
    TCCR0A |= (1 << COM0A0);
    TCCR0A &= ~(1 << COM0A1);
    TCCR0B |= (1 << CS02);      // Prescaler = 256
    OCR0A = 35;
```

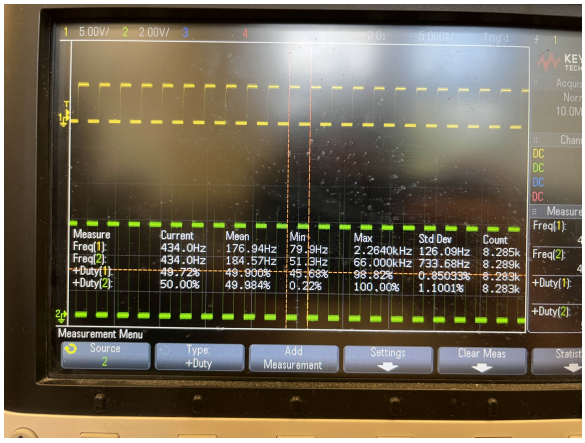
```
}

int main(void) {
    Initialize();
    while (1);
}
```

Listing 3: Phase-Correct PWM mode 440 Hz generation

Oscilloscope Validation

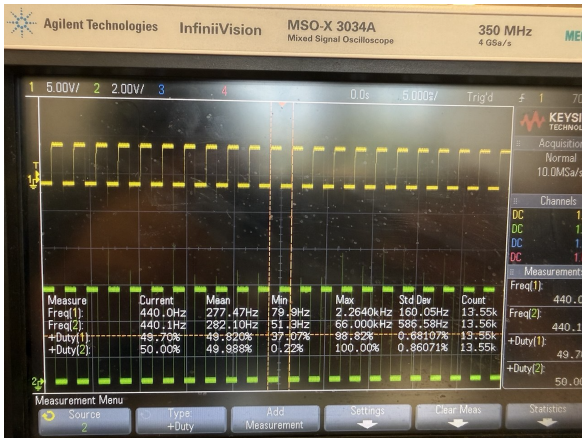
Each mode was validated using an oscilloscope to confirm the 440 Hz output:

(a) Normal mode waveform



(b) CTC mode waveform

(c) Phase-Correct PWM waveform



(d) PWM with variable duty cycle

Figure 1: Oscilloscope captures of 440 Hz waveforms across three timer modes

Ultrasonic Distance Measurement

HC-SR04 Sensor Operation

The HC-SR04 ultrasonic sensor measures distance by emitting a 40 kHz burst and timing the echo return. Operation proceeds as follows:

1. A 10 μs trigger pulse is sent on the Trig pin (PD2).
2. The sensor emits eight 40 kHz ultrasonic pulses.

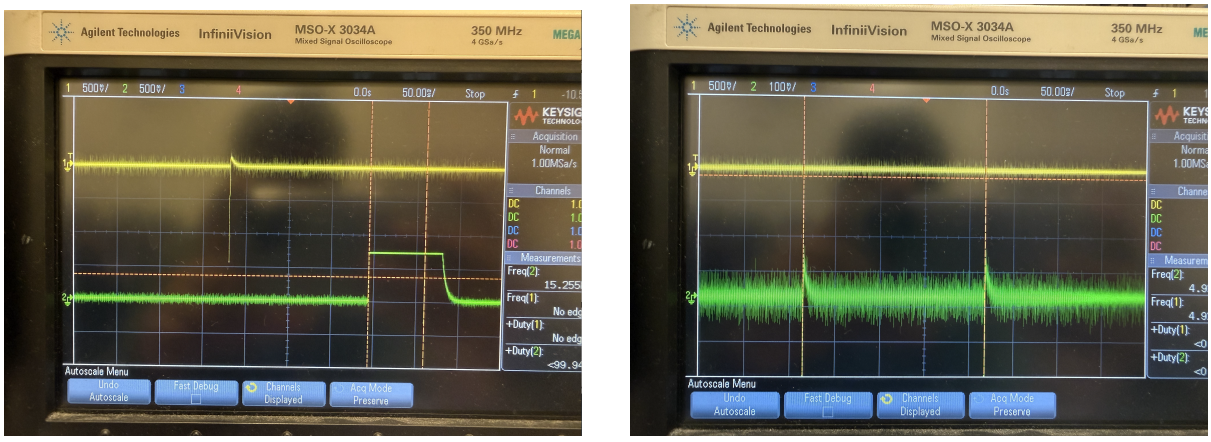
3. The Echo pin (PD0/PD3) goes high until the reflected signal is received.
4. Pulse width of the Echo signal is proportional to distance.

Distance is calculated from the echo pulse duration:

$$d_{\text{cm}} = \frac{t_{\text{echo}} \times 0.0343}{2} \tag{4}$$

Measured Range

- **Minimum distance:** 3 cm
- **Maximum distance:** 152 cm



(a) Maximum distance (152 cm)

(b) Minimum distance (3 cm)

Figure 2: Oscilloscope captures of ultrasonic echo pulses at extreme distances

Implementation

Timer1 serves as the timing reference for measuring the echo pulse width. Configured with a prescaler of 8 (yielding 2 MHz tick rate), it provides microsecond-level resolution for accurate distance computation.

```
void send_trigger(void) {
    PORTD &= ~(1 << TRIG_PIN);
    _delay_us(2);
    PORTD |= (1 << TRIG_PIN);
    _delay_us(10);
    PORTD &= ~(1 << TRIG_PIN);
}

uint16_t get_distance_cm(void) {
    uint16_t start, end;
    uint32_t timeout;

    timeout = 0;
    while (!(PIND & (1 << ECHO_PIN))) {
        if (timeout++ > 500000) return 0;
    }
    start = TCNT1;

    timeout = 0;
    while (PIND & (1 << ECHO_PIN)) {
        if (timeout++ > 500000) return 0;
    }
```

```
    end = TCNT1;

    if (end > start) {
        uint16_t ticks = end - start;
        uint32_t us = ticks / 2;
        return (uint16_t)(us / 58);
    }
    return 0;
}
```

Listing 4: Ultrasonic distance measurement

Distance-to-Frequency Mapping

Musical Note Table

Using CTC mode with a prescaler of 256, the `OCR0A` values for each note in the C6–C7 range are computed as:

$$\text{OCR0A} = \frac{f_{\text{CPU}}}{2 \times f_{\text{note}} \times N} - 1 = \frac{16 \times 10^6}{2 \times f_{\text{note}} \times 256} - 1 \tag{5}$$

Table 1: Musical note frequencies and corresponding OCR0A values

Note	C6	D6	E6	F6	G6	A6	B6	C7
Frequency (Hz)	1046	1174	1318	1397	1568	1760	1975	2093
OCR0A	30	26	23	21	19	17	15	14

Linear Mapping Formula

A linear relationship between the sensor distance (in cm) and the `OCR0A` register value was derived from two calibration points:

- At 3 cm (closest): OCR0A = 14 (C7, 2093 Hz — highest pitch)
- At 152 cm (farthest): OCR0A = 30 (C6, 1046 Hz — lowest pitch)

Using these two points $(3, 14)$ and $(152, 30)$:

$$m = \frac{30 - 14}{152 - 3} = \frac{16}{149} \approx 0.1074 \tag{6}$$

$$b = 14 - \frac{16}{149} \times 3 \approx 13.678 \tag{7}$$

$$\boxed{\text{OCR0A} = \frac{16}{149} \times d_{\text{cm}} + 13.678} \tag{8}$$

In firmware, the result is rounded to the nearest integer and clamped to the range $[14, 30]$.

Discrete Note Mode

In discrete mode, the 3–152 cm distance range is divided into eight equal segments of approximately 19 cm each. Each segment maps to one of the eight musical notes, with closer distances producing higher pitches:

```
const uint8_t discrete_notes[8] = {30, 26, 23, 21, 19, 17, 15, 14};

uint8_t map_distance_to_note_index(uint16_t distance) {
    if (distance < 3) return 7;   // Closest = C7
    if (distance > 152) return 0; // Farthest = C6

    uint16_t offset = distance - 3;
    uint8_t segment = offset / 19;
    if (segment > 7) segment = 7;
    return 7 - segment;  // Invert: closer = higher
}
```

Listing 5: Discrete note mapping

Continuous Frequency Mode

In continuous mode, the linear formula from Equation 8 is applied directly, providing a smooth frequency sweep as the hand moves:

```
float ocr_float = 0.10738255f * (float)dist + 13.67785235f;
uint8_t ocr_val = (uint8_t)(ocr_float + 0.5f);  // Round
if (ocr_val < 14) ocr_val = 14;
if (ocr_val > 30) ocr_val = 30;
OCR0A = ocr_val;
```

Listing 6: Continuous frequency mapping

Volume Control via Photoresistor

ADC Configuration

The photoresistor is connected to ADC channel 0 (PC0). The ADC is configured for 10-bit resolution with the following parameters:

- Reference voltage: AVCC (5V)
- ADC clock: 125 kHz ($f_{\text{CPU}}/128$)
- Mode: Free-running (continuous conversion)
- Measured range: 95 (bright) to 989 (dark)

Duty Cycle Mapping

The ADC reading is mapped to 10 discrete volume levels spanning 5%–50% duty cycle:

Table 2: ADC range to duty cycle mapping

ADC Range	Duty Cycle
95	5%
194	10%
294	15%
393	20%
492	25%
592	30%
691	35%
790	40%
890	45%
989	50%

Figure 3: Terminal output showing varying ADC readings and corresponding duty cycles

System Integration

Pin Conflict Resolution

A key design challenge arose from the dual use of Timer0. In the individual subsystems, PD6 (OC0A) was used for the buzzer output. However, since `OCR0A` determines the PWM *frequency* (TOP value), it cannot simultaneously control the duty cycle on the same pin.

The solution uses PD5 (OC0B) for the buzzer output instead. This separation allows:

- `OCR0A`: Controls the period/frequency (musical pitch)
- `OCR0B`: Independently controls the duty cycle (volume)

Circuit Diagram

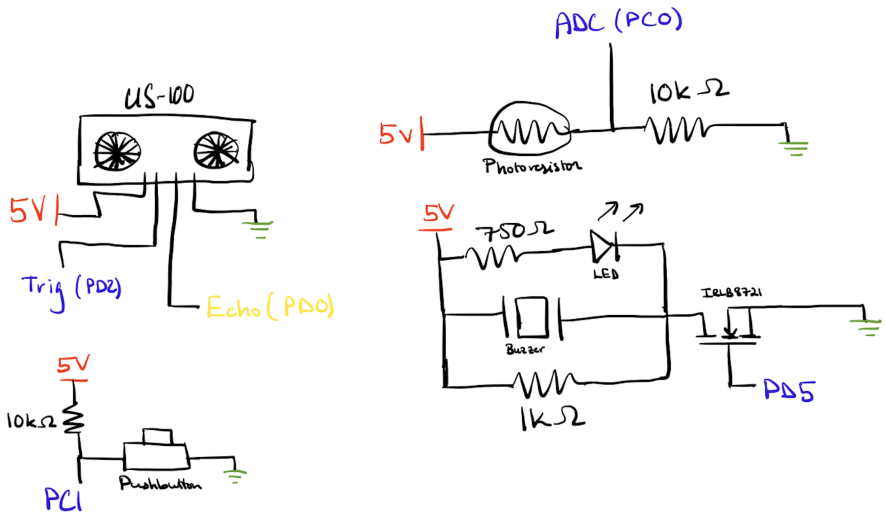


Figure 4: Complete theremin circuit schematic

Pin Assignments

Table 3: ATmega328PB pin assignments

Pin	Function	Description
PD2	Digital Output	Ultrasonic trigger
PD0	Digital Input	Ultrasonic echo
PD5 (OC0B)	PWM Output	Buzzer (frequency + volume)
PC0 (ADC0)	Analog Input	Photoresistor
PC1	Digital Input	Mode select button (pull-up)

Integrated Firmware

The final firmware combines all subsystems into a single main loop:

```
int main(void) {
    char status_buffer[100];

    // Configure I/O
    DDRD |= (1 << TRIG_PIN) | (1 << BUZZER_PIN);
    DDRD &= ~(1 << ECHO_PIN);
    DDRC &= ~(1 << BUTTON_PIN);
    PORTC |= (1 << BUTTON_PIN);  // Enable pull-up

    // Initialize peripherals
    UART_init(BAUD_PRESCALE);
    init_adc();
    timer0_pwm_init();

    while (1) {
        // Check button state (active low)
        if (!(PINC & (1 << BUTTON_PIN)))
            freq_mode = 1;  // Continuous
        else
            freq_mode = 0;  // Discrete
```

```
21
22          // Measure distance
23          ultrasonic_trigger();
24          _delay_us(50);
25          uint16_t distance = get_distance();
26          if (distance < 5) distance = 5;
27          if (distance > 30) distance = 30;
28
29          // Determine frequency based on mode
30          uint8_t frequency_value;
31          if (freq_mode == 0)
32              frequency_value = get_discrete_note(distance);
33          else
34              frequency_value = get_continuous_freq(distance);
35
36          // Read photoresistor for volume
37          uint16_t light_level = read_adc();
38          uint8_t duty_value = calculate_duty(light_level,
39                                              frequency_value);
40          // Update PWM registers
41          OCR0A = frequency_value;
42          OCR0B = duty_value;
43
44          _delay_ms(50);
45      }
46  }
```

Listing 7: Integrated theremin — main loop (part_f.c)

Timer0 PWM Configuration

The integrated system uses Phase-Correct PWM with non-inverting output on OC0B:

```
1  void timer0_pwm_init(void) {
2      // Phase Correct PWM, non-inverting on OC0B
3      TCCR0A = (1 << COM0B1) | (1 << WGM00);
4      TCCR0B = (1 << WGM02) | (1 << CS02);  // Prescaler 256
5      OCR0A = 20;   // Initial frequency
6      OCR0B = 10;   // Initial duty cycle
7  }
```

Listing 8: Timer0 PWM initialization for integrated system

Results and Discussion

Waveform Verification

All three timer modes (Normal, CTC, and Phase-Correct PWM) produced the target 440 Hz square wave, confirmed via oscilloscope. CTC mode was selected for the final design due to its simplicity—the hardware automatically toggles the output pin and resets the counter, requiring no interrupt service routine.

Distance Sensing Performance

The ultrasonic sensor reliably measured distances from 3 cm to 152 cm. The linear mapping formula (Equation 8) accurately converted this range to `OCR0A` values of 14–30, corresponding to musical notes C7 through C6.

Volume Control

The photoresistor ADC readings (95–989) mapped cleanly to a 5%–50% duty cycle range across 10 volume levels. Using OC0B for the PWM output allowed independent control of frequency and duty cycle through a single timer peripheral.

System Summary

Table 4: System specifications

Parameter	Value
CPU Clock	16 MHz
Timer Prescaler	256
Timer Frequency	62.5 kHz
Frequency Range	1046–2093 Hz (C6–C7)
Distance Range	3–152 cm
ADC Resolution	10-bit (0–1023)
ADC Sensor Range	95–989
Duty Cycle Range	5%–50%
Volume Levels	10
UART Baud Rate	9600

Skills and Technologies

- **C (Embedded):** Bare-metal firmware for the ATmega328PB with direct register manipulation
- **PWM:** Phase-Correct and CTC waveform generation for audio frequency synthesis
- **ADC:** 10-bit analog-to-digital conversion with free-running mode for real-time light sensing
- **Ultrasonic Sensors:** HC-SR04 time-of-flight distance measurement with echo pulse timing
- **Timer/Counter Peripherals:** Dual-timer architecture (Timer0 for PWM, Timer1 for pulse measurement)
- **UART:** Serial communication for real-time debugging and status monitoring