The goal of the assignment is use GPIO and delays using Timers and Interrupts:

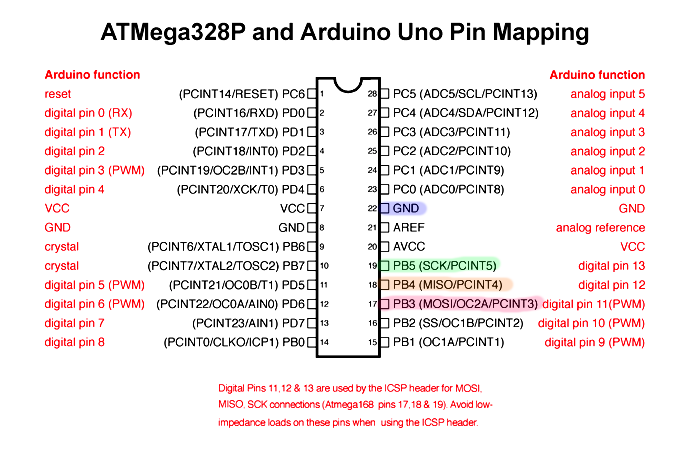
1. Generate three delays using three timers T0, T3, and T4.

a. Implement a delay of 0.125ms using Timer 0 in normal mode. Count OVF  
occurrence if needed. Do not use interrupts. Turn ‘on’ PB5 LED (also monitor and verify using logic analyzer) for approx. 1.5 sec and ‘off’ for 1.5 sec.

b. Implement a delay of 0.250ms using Timer 3 TIMER3\_COMPA\_vect interrupt mechanism in CTC mode. Count OVF occurrence if needed in the IRQ subroutine. Turn ‘on’ PB4 LED (also monitor and verify using logic analyzer) for approx. 2 sec and ‘off’ for 2 sec.

c. Implement a delay of 0.100ms using Timer 4 TIMER4\_OVF\_vect interrupt  
mechanism in normal mode. Count OVF occurrence if needed in the IRQ  
subroutine. Turn ‘on’ PB3 LED (also monitor and verify using logic analyzer) for approx. 1 sec and ‘off’ for 1 sec.

**Components Used/Connected**



1. **Timer0**

// Timer0 polling

void Normal\_Timer0(void)

{

TCNT0 = 225; // start count from 225

TIFR0 |= (1 << TOV0); // clear overflow flag

TCCR0B = (1 << CS01) | (1 << CS00); // prescaler 64

while (!(TIFR0 & (1 << TOV0)))

{

// wait

}

// stop timer

TCCR0B = 0x00;

}

int main(void)

{

// Set PB5 as output for Timer0 toggling

DDRB |= (1 << PB5);

// Timer0 poll loop for PB5 => 1.5s on/off

while (1)

{

// PB5 on

PORTB |= (1 << PB5);

for (uint16\_t i = 0; i < 12000; i++)

{

Normal\_Timer0();

}

// PB5 off

PORTB &= ~(1 << PB5);

for (uint16\_t i = 0; i < 12000; i++)

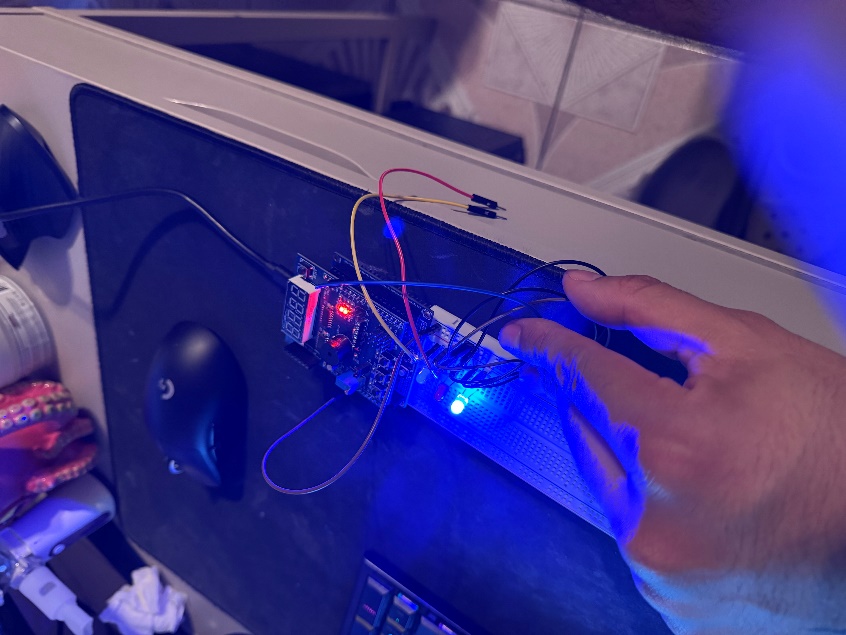
{

Normal\_Timer0();

}

}

A hand holding a circuit board

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

3 = 011

Timer0 is running in normal mode

1. **Timer3**

// Timer3

ISR(TIMER1\_COMPA\_vect)

{

t3\_count++;

if (t3\_count >= 8000)

{

PB4\_led = !PB4\_led;

if (PB4\_led)

PORTB |= (1 << PB4);

else

PORTB &= ~(1 << PB4);

t3\_count = 0;

}

}

void CTC\_Timer3(void)

{

DDRB |= (1 << PB4); // PB4 output

TCCR1B |= (1 << WGM12); // CTC mode => WGM12 = 1

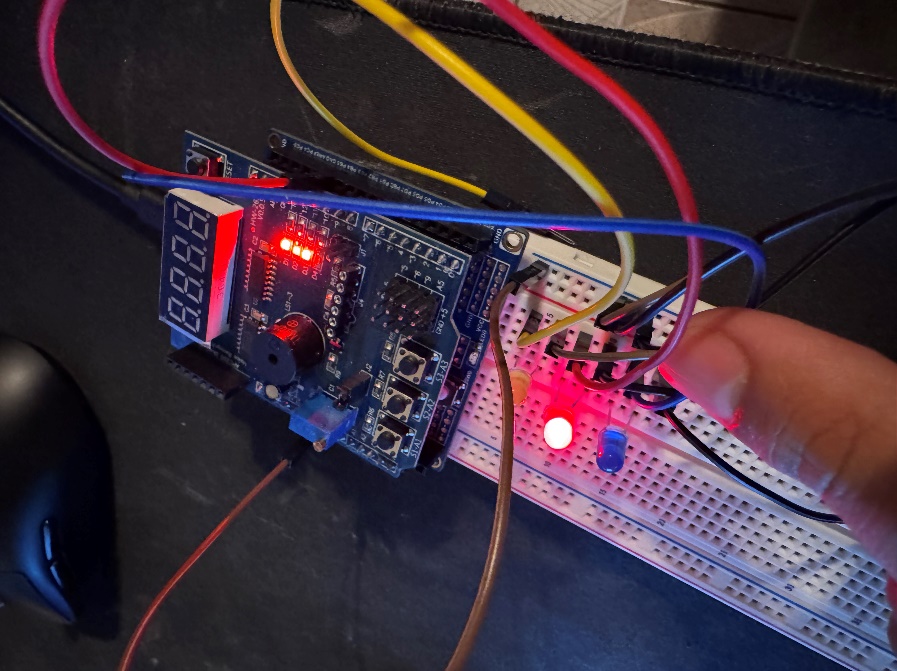
OCR1A = 61; // OCR1A = 61 => ~248us

TCCR1B |= (1 << CS11) | (1 << CS10); // Prescaler=64 => CS11=1, CS10=1

TIMSK1 |= (1 << OCIE1A); // Enable interrupt

}

A finger pointing at a circuit board

AI-generated content may be incorrect.

1. **Timer4**

// Timer4 ISR

ISR(TIMER2\_OVF\_vect)

{

// Reload for 100us

TCNT2 = 231;

t4\_count++;

if (t4\_count >= 10000) // ~1 second

{

PB3\_led = !PB3\_led;

if (PB3\_led)

PORTB |= (1 << PB3);

else

PORTB &= ~(1 << PB3);

t4\_count = 0;

}

}

void Normal\_Timer4(void)

{

DDRB |= (1 << PB3); // PB3 output

TCNT2 = 231; // preload for 100us

// Normal mode, prescaler=64 => TCCR2B = 0b100

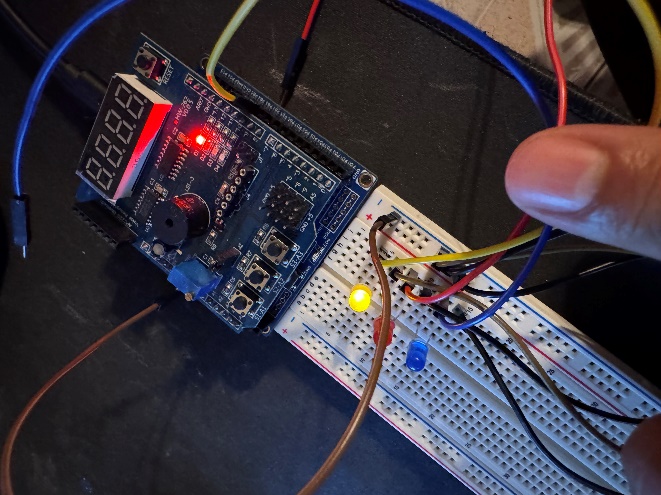
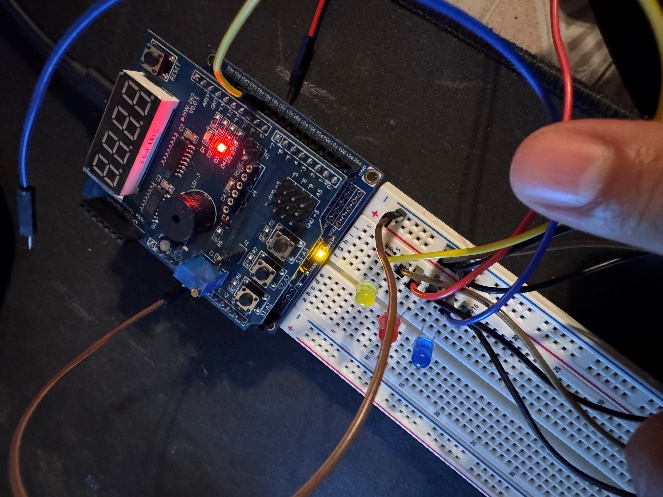
TCCR2A = 0x00;

TCCR2B = (1 << CS22);

// Enable OVF interrupt

TIMSK2 |= (1 << TOIE2);

}

****

**Final Code**

/\*

\* DA3.c

\*

\* Created: 3/28/2025 5:03:16 PM

\* Author : enriq

\*/

#define F\_CPU 16000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

// Timer4 Variables

volatile int t4\_count = 0;

volatile int PB3\_led = 0;

// Timer3 Variables

volatile int t3\_count = 0;

volatile int PB4\_led = 0;

// Timer0 polling

void Normal\_Timer0(void)

{

TCNT0 = 225; // start count from 225

TIFR0 |= (1 << TOV0); // clear overflow flag

TCCR0B = (1 << CS01) | (1 << CS00); // prescaler 64

while (!(TIFR0 & (1 << TOV0)))

{

// wait

}

// stop timer

TCCR0B = 0x00;

}

// Timer3

ISR(TIMER1\_COMPA\_vect)

{

t3\_count++;

if (t3\_count >= 8000)

{

PB4\_led = !PB4\_led;

if (PB4\_led)

PORTB |= (1 << PB4);

else

PORTB &= ~(1 << PB4);

t3\_count = 0;

}

}

void CTC\_Timer3(void)

{

DDRB |= (1 << PB4); // PB4 output

TCCR1B |= (1 << WGM12); // CTC mode => WGM12 = 1

OCR1A = 61; // OCR1A = 61 => ~248us

TCCR1B |= (1 << CS11) | (1 << CS10); // Prescaler=64 => CS11=1, CS10=1

TIMSK1 |= (1 << OCIE1A); // Enable interrupt

}

// Timer4 ISR

ISR(TIMER2\_OVF\_vect)

{

// Reload for 100us

TCNT2 = 231;

t4\_count++;

if (t4\_count >= 10000) // ~1 second

{

PB3\_led = !PB3\_led;

if (PB3\_led)

PORTB |= (1 << PB3);

else

PORTB &= ~(1 << PB3);

t4\_count = 0;

}

}

void Normal\_Timer4(void)

{

DDRB |= (1 << PB3); // PB3 output

TCNT2 = 231; // preload for 100us

// Normal mode, prescaler=64 => TCCR2B = 0b100

TCCR2A = 0x00;

TCCR2B = (1 << CS22);

// Enable OVF interrupt

TIMSK2 |= (1 << TOIE2);

}

int main(void)

{

CTC\_Timer3(); // Initialize Timer3

Normal\_Timer4(); // Initialize Timer4

// Set PB5 as output for Timer0 toggling

DDRB |= (1 << PB5);

// Enable global interrupts

sei();

// Timer0 poll loop for PB5 => 1.5s on/off

while (1)

{

// PB5 on

PORTB |= (1 << PB5);

for (uint16\_t i = 0; i < 12000; i++)

{

Normal\_Timer0();

}

// PB5 off

PORTB &= ~(1 << PB5);

for (uint16\_t i = 0; i < 12000; i++)

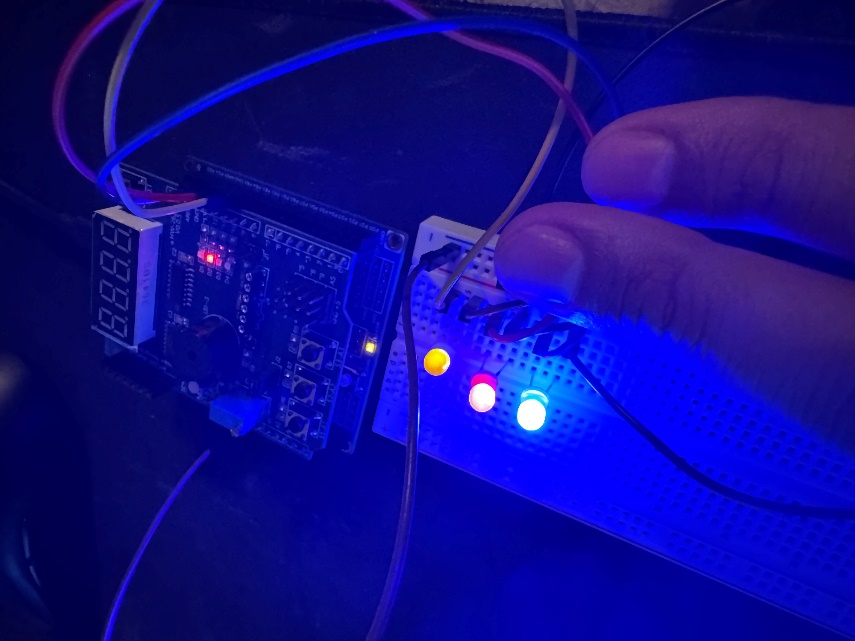
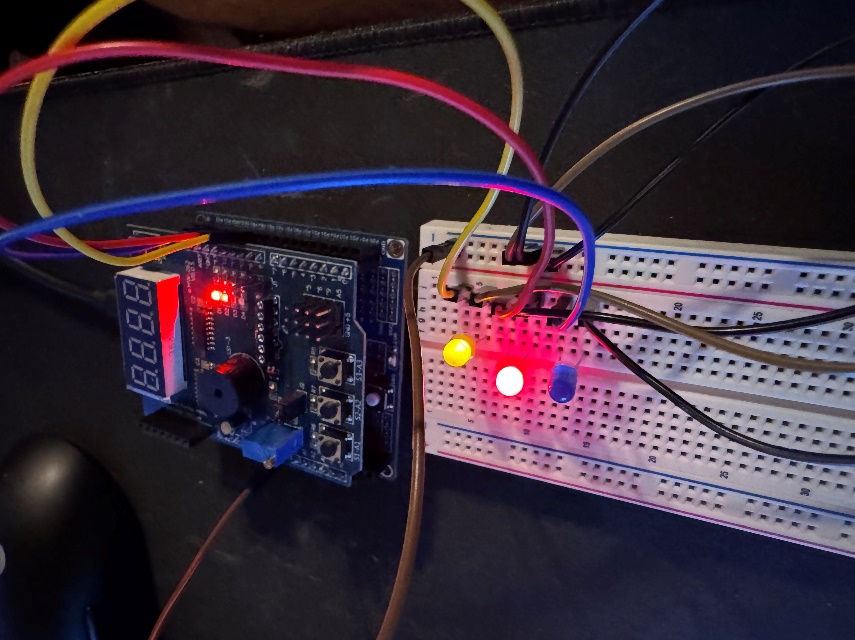
{

Normal\_Timer0();

}

}

}



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

At 10000, PB3 is triggered so the LED turns on. T4\_count is incremented every 100 us, so after 10000 we have a 1 second delay.

At 8000, PB4 is triggered to the LED turns on. T3\_count is incremented every 250 us, so after 8000 we have a 2 second delay.