CPE301 – SPRING 2019

Design Assignment X

Student Name: Chris Barr

Student #: 2000682859

Student Email: barrc1@unlv.nevada.edu

Primary Github address: https://github.com/BarrChris

Directory: https://github.com/BarrChris/submission\_da.git

Submit the following for all Labs:

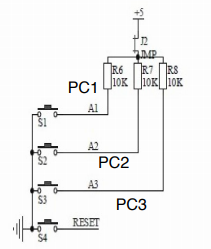
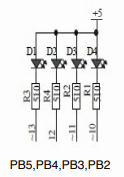
1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

List of Components used

* Atmega328P
* Multi-Function Shield
* LED (PB2)
* Pushbutton (PC2)

Block diagram with pins used in the Atmega328P

These are block diagrams for the 2 main components I used, PC2 (pushbutton), PB2 (LED)

1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 2/A**

This code was used in the prelab and used as a basis for my code

;Square wave with a period of 4 seconds

.org 0

LDI R16,32 ;representing PB5

OUT DDRB, R16 ;enable PB5 as output

LDI R17,0 ;used to set or reset PB5

LDI R20,5 ;set clock prescaler to 1024

STS TCCR1B,R20

begin:

LDI R20,0x00 ;resetting the counter to 0

STS TCNT1H,R20

STS TCNT1L,R20

RCALL delay ;calling timer to wait for 1 sec

EOR R17,R16 ;XOR to toggle led

OUT PORTB,R17

RJMP begin ;repeat main loop

delay:

LDS R29, TCNT1H ;loading upper bit of counter to R29

LDS R28, TCNT1L ;loading lower bit of counter to R28

CPI R28,0x08 ;comparing if lower 8 bits of timer is 0x08

BRSH body ;if lower bits of timer have reached desired amount, check the upper

bits

RJMP delay ;otherwise, keep checking lower bits

body:

CPI R29,0x1E ;check to see if upper timer bits have reached the desired value

BRLT delay ;if not, recheck the lower bits

RET ;once the timer reached the desired value, toggle the LED

1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 2/A**

This is the modified code to fit the DA2A requirements

Task 1:

AVR Code

.CSEG

.org 0x0000

SBI DDRB, 5 ;Make sure PB5 stays turned off

SBI PORTB, 5

SBI DDRB, 2 ;Sets PB2 as an output

LDI R16,0x04 ;Represents PB2

LDI R17,0x00 ;Used to toggle LED

LDI R20,5 ;set clock prescaler to 1024

STS TCCR1B,R20 ;Clock prescaler now set to 1024

begin:

LDI R20,0x00 ;Used to reset counter

STS TCNT1H,R20 ;

STS TCNT1L,R20 ;Counter high and low reset to 0

SBI PORTB, 2 ; Turns on PB2

RCALL delay ;calling timer to wait for 0.435 sec turned on

LDI R20,0x00 ;Used to reset counter

STS TCNT1H,R20

STS TCNT1L,R20

CBI PORTB, 2 ; Turns off PB2

RCALL delay2 ;calling timer to wait for 0.29 sec turned off

RJMP begin ;repeat main loop

delay:

LDS R29, TCNT1H ;Loads upper bits into register R29

LDS R28, TCNT1L ;Loads lower bits into register R28

CPI R28,0x8C ;comparing lower 8 bits of timer to 0x8C

BRSH body ;if lower bits are the same or higher to 0x8C, jump to body and start counting there

RJMP delay ;else, keep checking lower bits

body:

CPI R29,0x1A ;comparing upper 8 bits of timer to 0x1A

BRLT delay ;If lower, then go back and check the lower bits

RET ;When the timer is over, return to turn off the lead

delay2:

LDS R29, TCNT1H ;Loads upper bits into register R29

LDS R28, TCNT1L ;Loads lower bits into register R28

CPI R28,0xB2 ;comparing lower 8 bits of timer to 0xB2

BRSH body2 ;if lower bits are the same or higher to 0xB2, jump to body and start counting there

RJMP delay2 ;else, keep checking lower bits

body2:

CPI R29,0x11 ;comparing upper 8 bits of timer to 0x11

BRLT delay2 ;If lower, then go back and check the lower bits

RET ;When the timer is over, return to turn off the lead

C Code

#include <avr/io.h>

#include <stdio.h>

int main(void)

{

DDRB = (1<<2); //Sets PB2 as an output

TCCR1B = 5; //Set prescaler to 1024

while(1) //infinite loop to keep the LED's to blink

{

TCNT1 = 0; //Reset counter

while (TCNT1 != 4530){} //Delay to keep LED off

PORTB ^= (1<<2); //LED turns on

TCNT1 = 0; //Reset counter

while (TCNT1 != 6796){} //Delay to keep LED on

PORTB ^= (1<<2); //LED turns off

}

return 0;

}

Task 2:

AVR Code

.CSEG

.org 0x0000

LDI R16, 0xFF ;Sets PB5 as output

OUT DDRB, R16

CBI DDRC, 2 ;Sets PC2 as input

LDI R20,5 ;set clock prescaler to 1024

STS TCCR1B, R20 ;send prescaler

OUT PORTB, R16

LOOP:

;Loops back here when nothing is pressed

SBI PORTB, 2 ;Keeps LED turned off

SBIC PINC, 2 ;Skips next instruction if pushbutton is pressed

RJMP LOOP ;If pushbutton isn't pressed, continues through this loop

;Goes here once button is pressed

LDI R18, 0b1110\_00 ;Turns on LED

OUT PORTB, R18

LDI R20, 0 ;Clears counter

STS TCNT1H, R20

STS TCNT1L, R20

RCALL DELAY ;Delay for 1.25 seconds

RJMP LOOP ;Loops back until pushbutton is pressed again

delay:

LDS R29, TCNT1H ;Loads upper bits into register R29

LDS R28, TCNT1L ;Loads lower bits into register R28

CPI R28,0x4A ;comparing lower 8 bits of timer to 0x8C

BRSH body ;if lower bits are the same or higher to 0x8C, jump to body and start counting there

RJMP delay ;else, keep checking lower bits

body:

CPI R29,0x4C ;comparing upper 8 bits of timer to 0x1A

BRLT delay ;If lower, then go back and check the lower bits

RET ;When the timer is over, return to turn off the lead

C Code

#include <avr/io.h>

#include <stdio.h>

int main(void)

{

DDRB = (1<<2); //PB2 as an output

PORTB = (1<<2); //Sets PB2

DDRC = (0<<2); //PC2 as an input

PORTC = (0<<2); //Sets PC2

TCCR1B = 5; //Sets prescaler to 1024

while (1) //loop forever to keep program running

{

if (!(PINC & (1 << PINC2))) //If the pushbutton is pressed, go through this if statement

{

PORTB &= ~(1<<2); //Keeps LED on until otherwise

while (TCNT1 != 19530){} //Delay to keep LED off

TCNT1 = 0; //Reset counter

}

else //else, if the pushbutton is not pressed, go through this else statement

{

PORTB |= (1<<2); //Keeps LED off until otherwise

TCNT1 = 0; //Reset counter

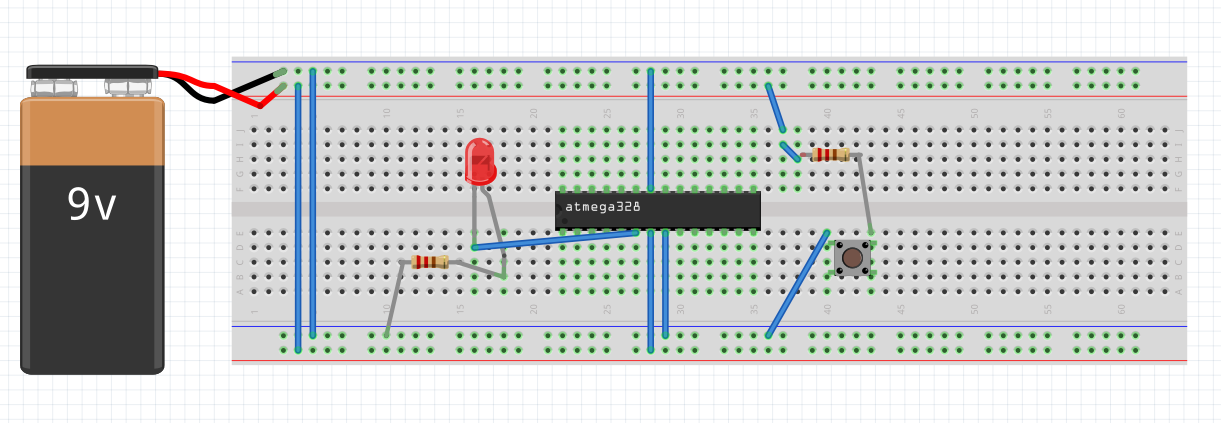
}

}

return 0;

}

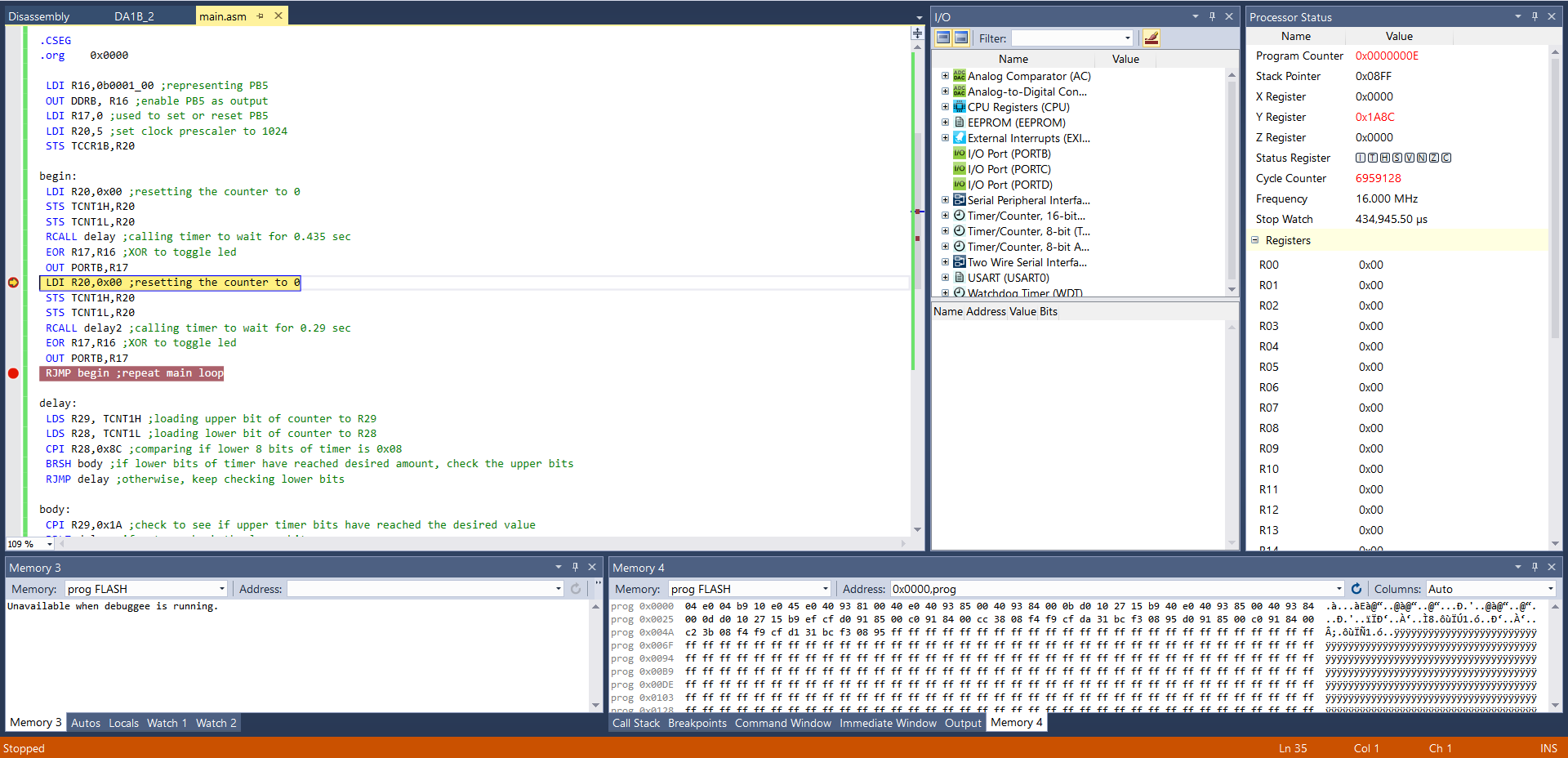
1. **SCHEMATICS**



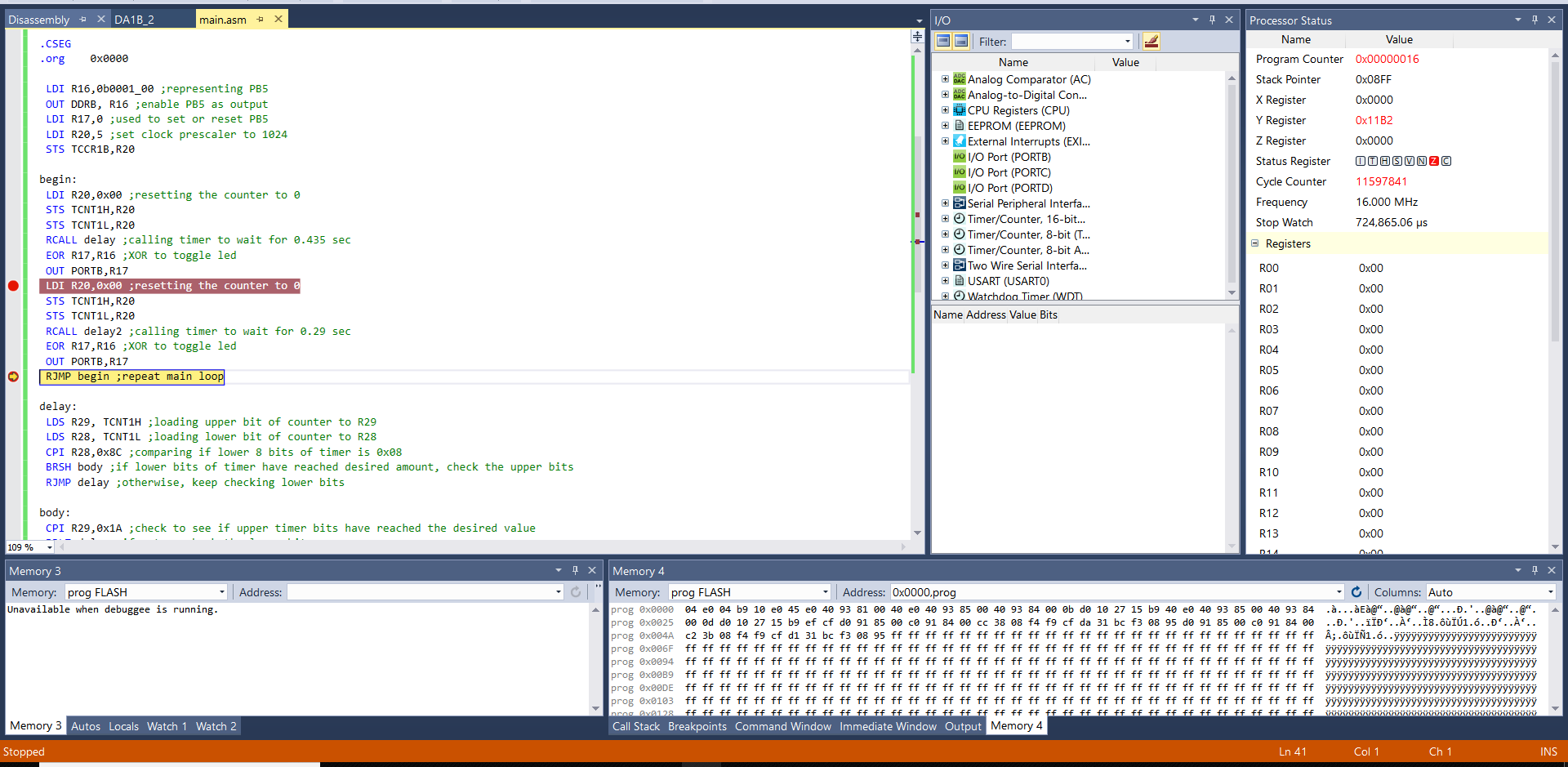
Note: It’s 5V not 9V, also assume the resistors/pushbutton/LED is the correction version on the board

1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

Task 1 (AVR):

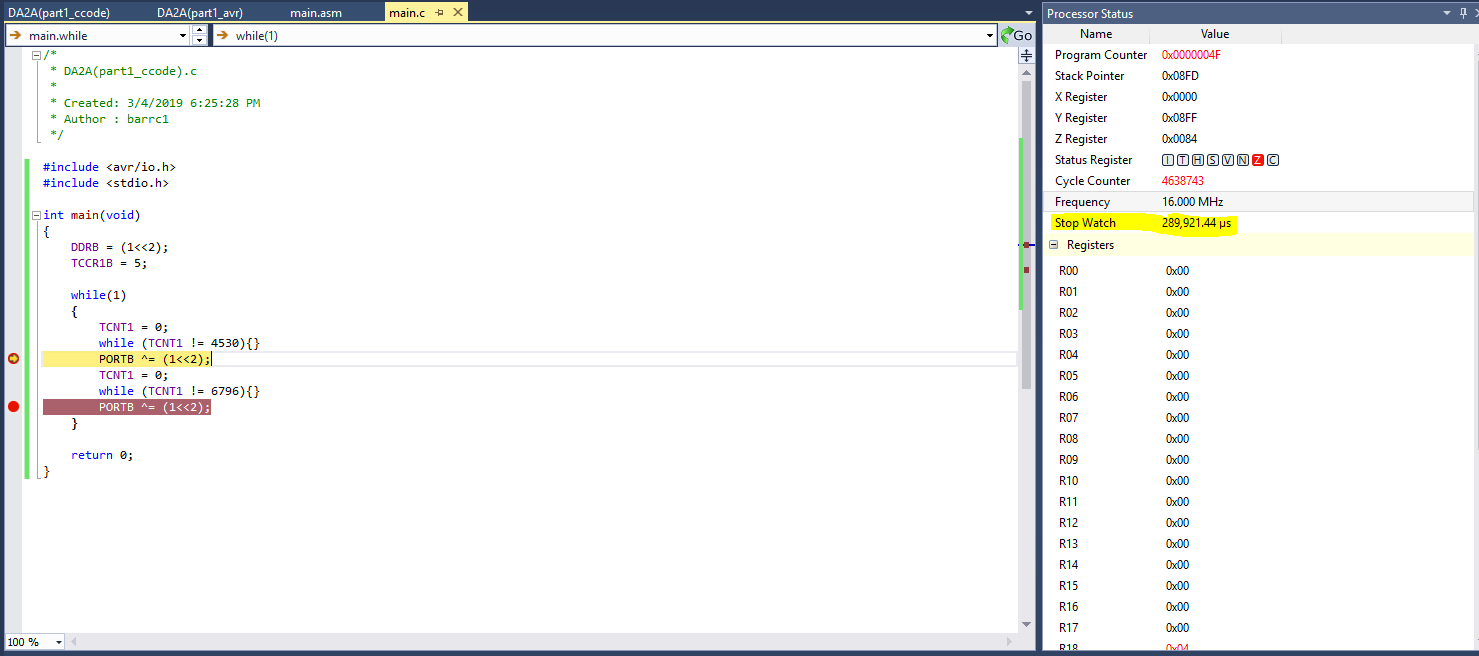


This shows the LED staying on for 435000 us (0.435 seconds) after going through the first delay loop

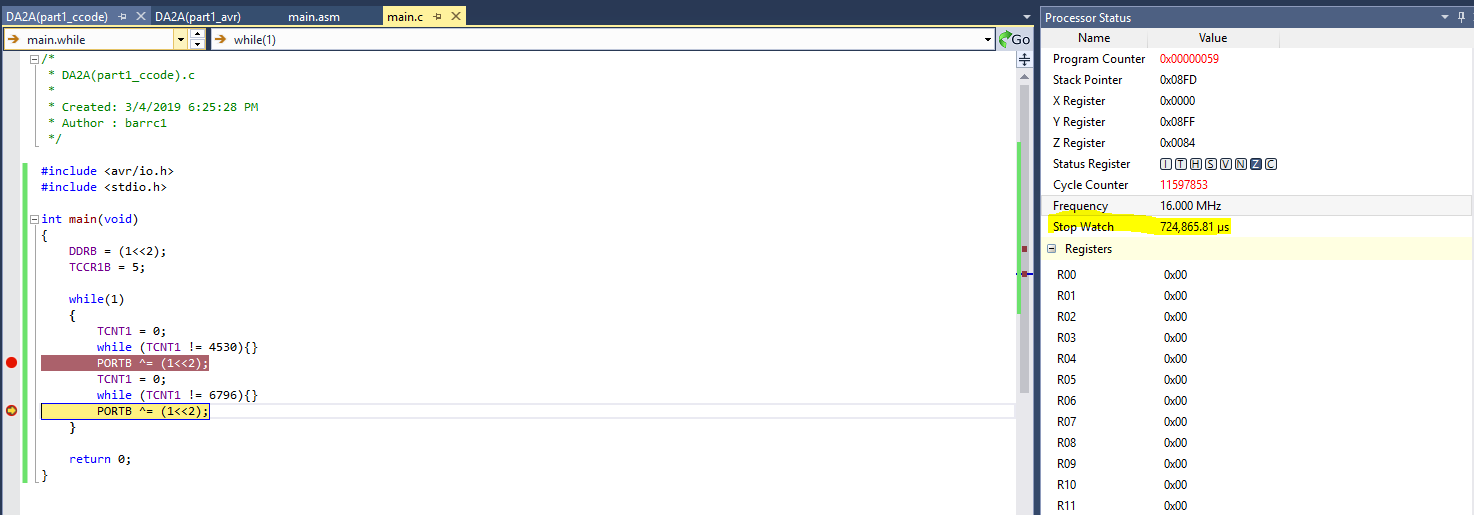


This shows the LED staying off for 290000 us (0.29 seconds) after going through the second delay loop, but the stop watch shows the accumulation of 725000 us (0.725 seconds) which proves the LED’s stay on and off at the correct time

Task 1 (C Code):

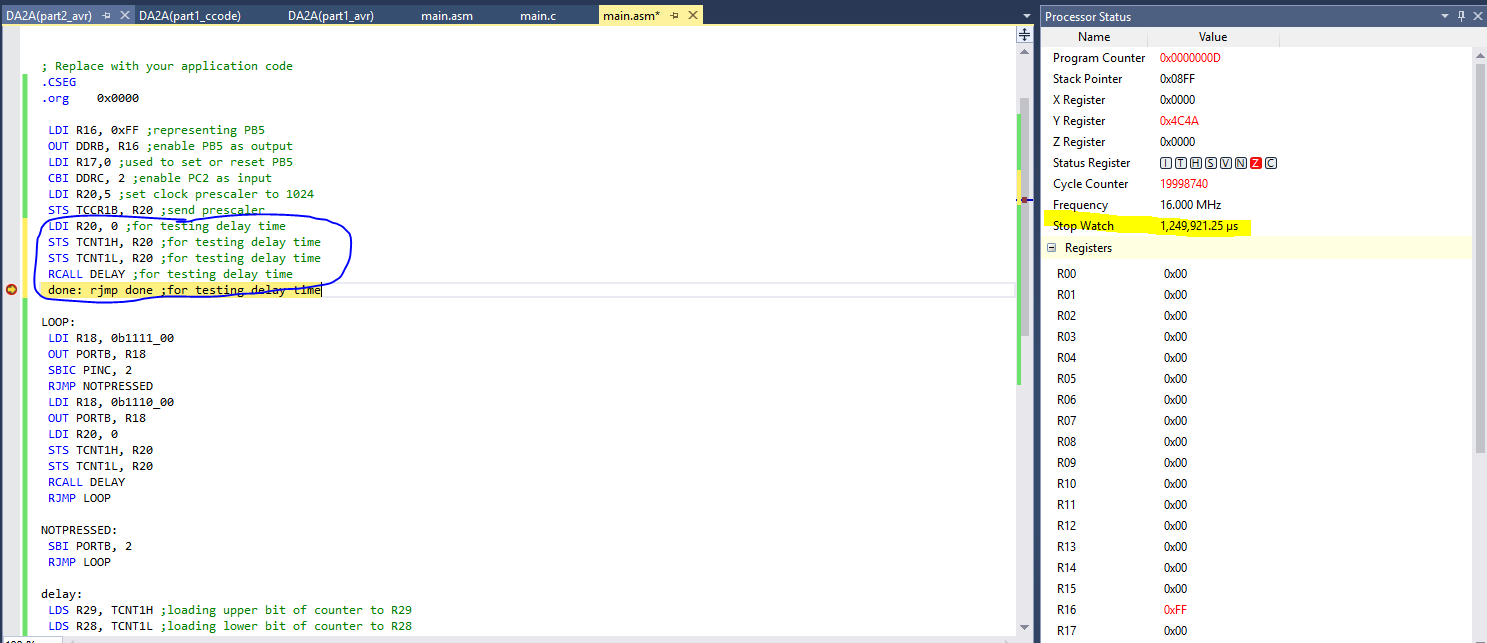


This shows the LED staying on for 290000 us (0.29 seconds) after going through the first delay loop



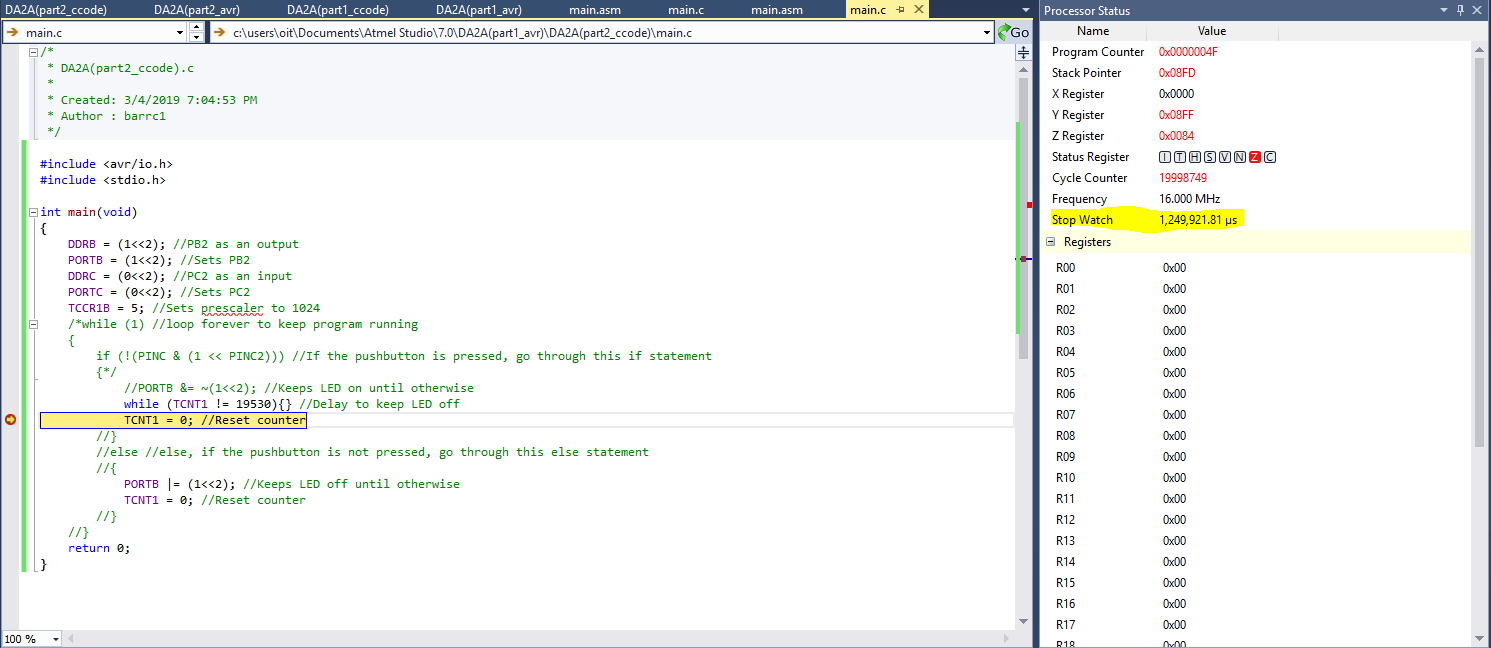
This shows the LED staying off for 435000 us (0.435 seconds) after going through the second delay loop, but the stop watch shows the accumulation of 725000 us (0.725 seconds) which proves the LED’s stay on and off at the correct time

Task 2 (AVR):



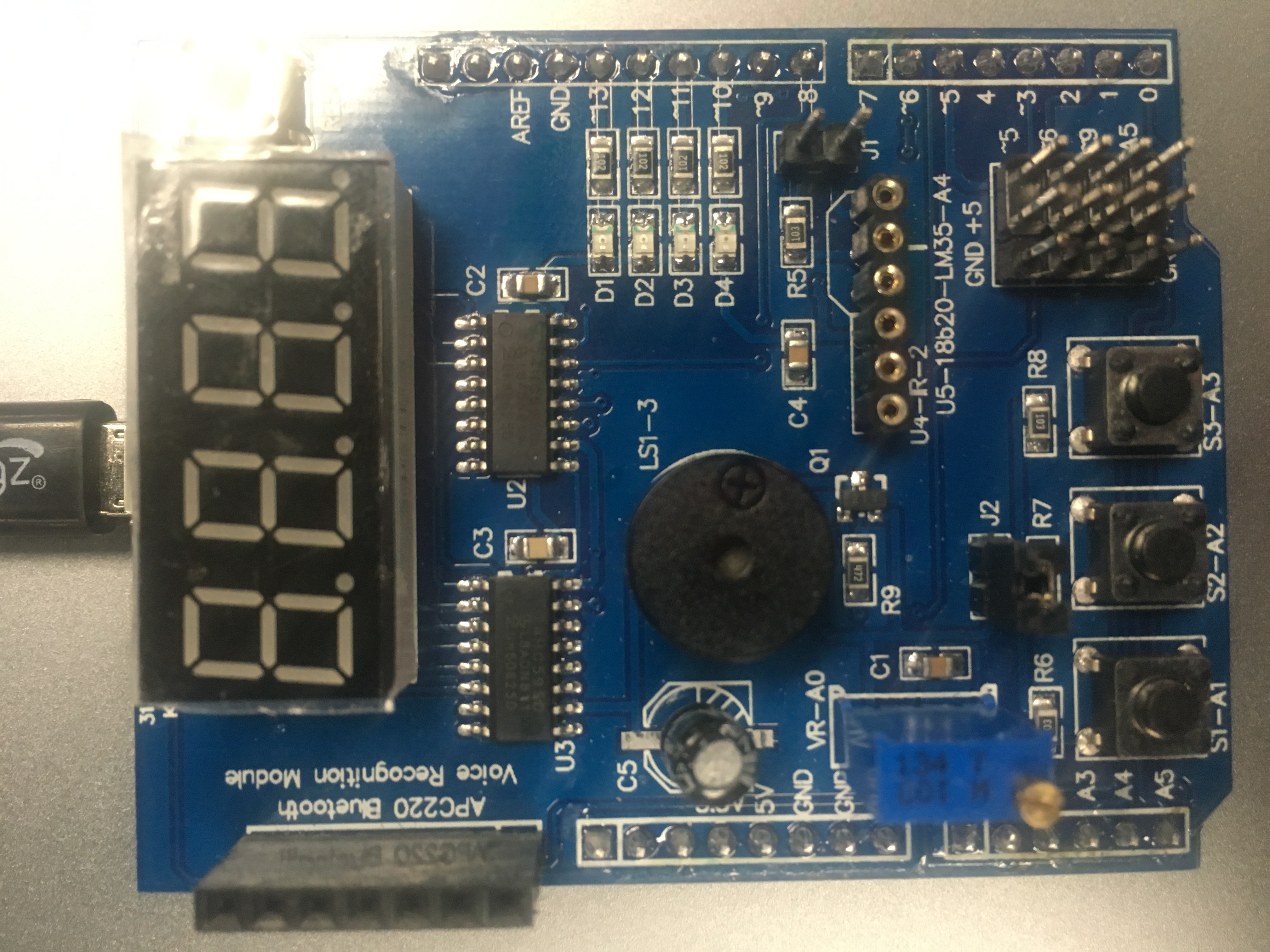
I implemented the code to just go straight through the delay loop to test the timing. The stop watch proves that 1250000 us (1.25 seconds) will pass, so it will go through the correct amount of delay

Task 2 (C Code):

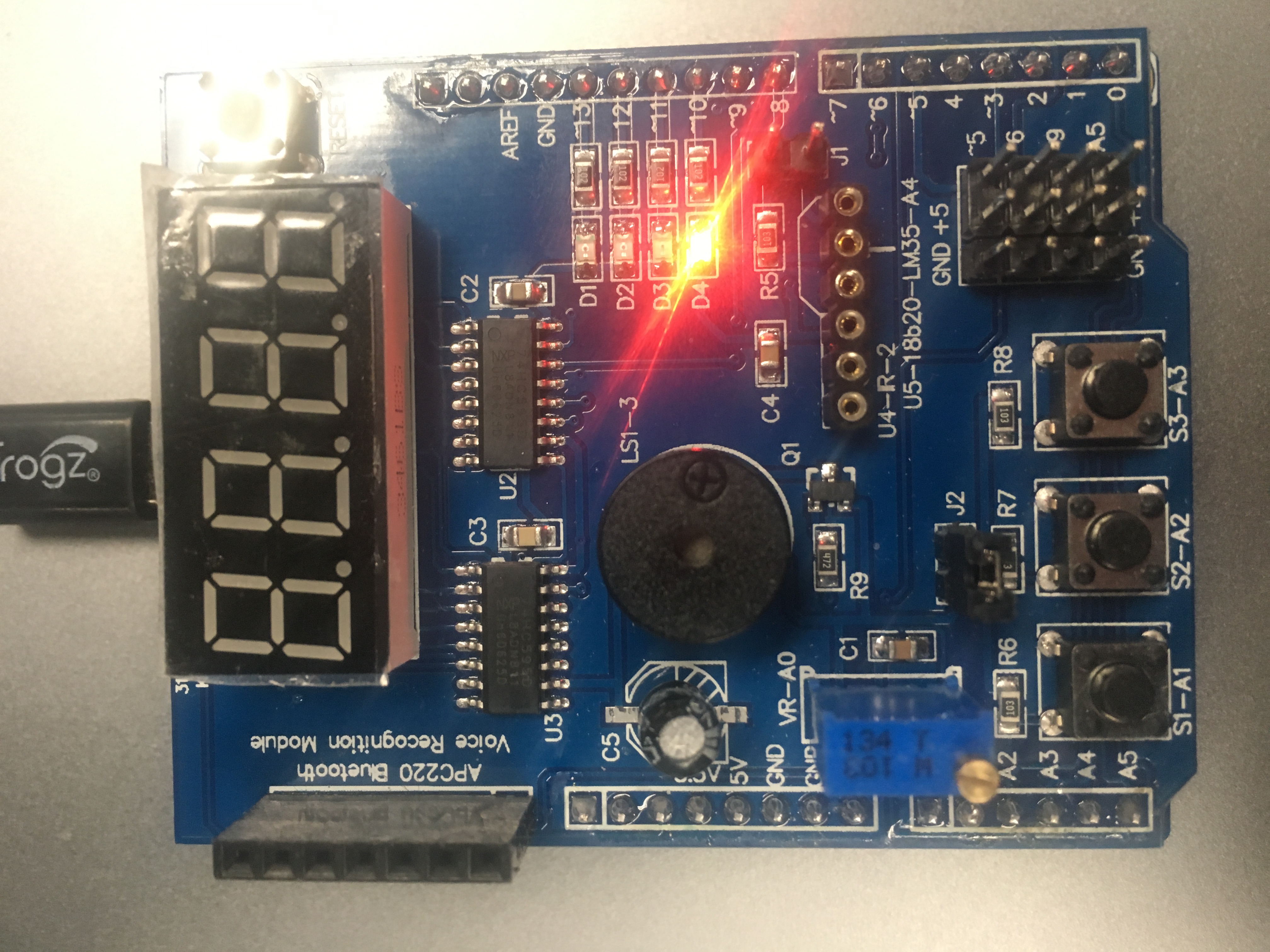


I commented out the code so it can test the delay, and this proves that it will go through 1250000 us (1.25 seconds).

1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**



For all tasks, the LED will be turned off and then turned on at the specific D4 LED



Here’s where the LED will be turned on from either the blinking during task 1 or after the pushbutton is pressed during task 2

1. **VIDEO LINKS OF EACH DEMO**

https://www.youtube.com/watch?v=SC6bMbpIVlA

1. **GITHUB LINK OF THIS DA**

https://github.com/BarrChris/submission\_da/tree/master/DesignAssignments/DA2B

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Chris Barr