CPE301 – SPRING 2019

Design Assignment 2a

Student Name: Do Le

Student #: 2001183621

Student Email: led2@unlv.nevada.edu

Primary Github address: <https://github.com/DoVietLe>

Directory: <https://github.com/DoVietLe/assignments>

Design Assignment 2A: The goal of the assignment is use GPIO and delays:

1. Design a delay subroutine to generate a waveform on PORTB.3 with 55% DC and 0.75 sec period.

2. Connect a switch to PORTC.3 (active high - turn on the pull up transistor) to poll for an event to turn on the led at PORTB.2 for 2 sec after the event.

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

Components Used

• Atmel Studio 7.0

-Assembler

-Simulator

• Atmega328PB (Xplained Mini Board)

• Multi-Function Shield

-Switch

-LED

• Logic Analyzer

Block Diagram:

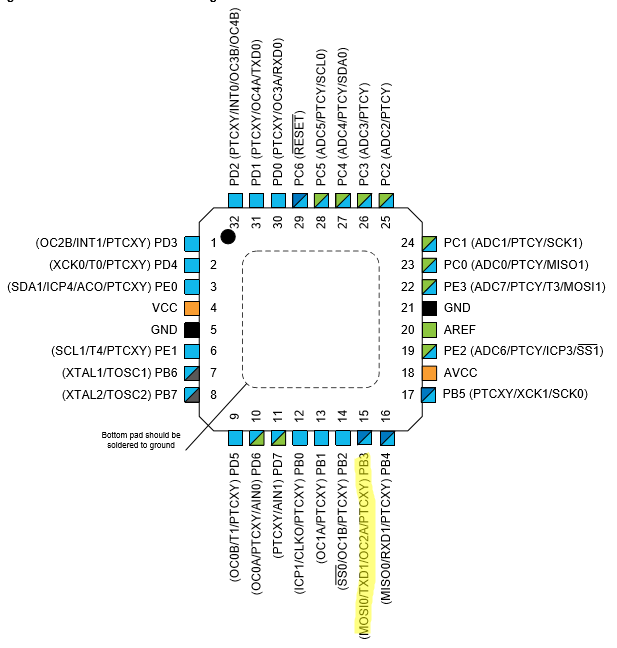


Figure 1: Pins used in task 1.

Task 1:

PORTB3: Used as output for the 55% duty cycle wave.

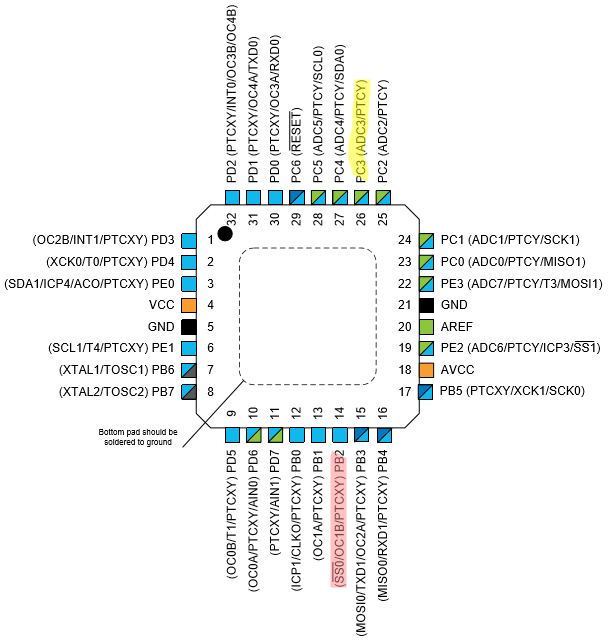
Task 2:

Figure 2: Pins used in task 2.

PORTC3: Used as input to read data from a switch.

PORTB2: Used as output to toggle the LED.

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

**Task 1 Codes**

**C Code:**

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRB = (0x01<<PORTB3);

while (1)

{

PORTB = (0x01<<PORTB3);

*\_delay\_ms*(412.5);

PORTB = 0x00;

*\_delay\_ms*(337.5);

}

}

**Assembly Code:**

.EQU A = 236

.EQU B = 254

start:

LDI R16, HIGH(RAMEND); Initialize the stack pointer.

OUT SPH, R16

LDI R16, LOW(RAMEND)

OUT SPL, R16

LDI R16, (0x01<<3) ; Sets PORTB.3 as an output.

OUT DDRB, R16

wave:

LDI R17, (0x01<<3) ; Sets the output waveform to high.

OUT PORTB, R17

LDI R16, 11

CALL delay ; Waits ~0.4125s.

LDI R17, 0x00

OUT PORTB, R17 ; Sets the output waveform to low.

LDI R16, 9

CALL delay ; Waits ~0.3375s.

RJMP wave

end:

RJMP end

; ~37.5ms delay

; Delay subroutine. The delay runs for about 600,000 cycles. The value of R16 determines

; how many times the delay is run (kind of).

delay:

PUSH R17 ; Pushes values onto the stack to save.

PUSH R18

delay\_again:

LDI R17, A ; Runs delay again.

loop0:

LDI R18, B ; Reinitialize the inner loop.

loop1:

NOP ; Do nothing.

NOP

NOP

NOP

NOP

NOP

NOP

DEC R18 ; Decrease inner most loop counter.

BRNE loop1 ; Branch if counter isn't up.

DEC R17 ; Decreases middle loop counter.

BRNE loop0 ; Branch if counter isn't up.

DEC R16 ; Decreases loop counter.

BRNE delay\_again; Branch if counter isn't up.

POP R18 ; Returns values from the stack.

POP R17

RET ; Exit the subroutine.

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

**Task 2 Codes**

**C Code:**

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRC = 0x00; // Sets PORTC3 as input port.

PORTC = (0x01<<PORTC3); // Activates the pull up resistor on PORTC3.

DDRB = (0x01<<PORTB2 | 0x01<<PORTB3); // Sets PORTB2 as an output port.

PORTB = (0x01<<PORTB2); // Output 1 on PORTB2 (Turns off LED since

// LED is connected to Vcc).

while (1)

{

// Waits for PINC3 to go low (the button is active high).

if ((PINC & (0x01<<PINC3)) == 0x00) {

PORTB &= ~(0x01<<PORTB2); // Turns on the LED by allowing current to sink.

*\_delay\_ms*(2000);

PORTB |= (0x01<<PORTB2); // Turns off the LED.

}

// Generates the waveform.

PORTB |= (0x01<<PORTB3); // Turns PORTB3 high for 55% duty cycle.

*\_delay\_ms*(412.5);

PORTB = ~(0x01<<PORTB3); // Turns PORTB3 low for 45% of the period.

*\_delay\_ms*(337.5);

}

}

**Assembly Code:**

.EQU A = 254

.EQU B = 252

.EQU C = 50

; Replace with your application code

start:

LDI R16, HIGH(RAMEND); Initializes the stack.

OUT SPH, R16

LDI R16, LOW(RAMEND)

OUT SPL, R16

LDI R16, 0x00

OUT DDRC, R16 ; Sets PORTC.3 as an input port.

LDI R16, (0x01<<3)

OUT PORTC, R16 ; Enabled the pull-up resistor on PORTC.3.

LDI R16, (0x01<<2 | 0x01<<3)

OUT DDRB, R16 ; Sets PORTB.2 and PORTB.3 as an output pin.

OUT PORTB, R16 ; Turns LED off (according to shield schematic,

; a value of 0 turns the LED on).

poll:

SBIS PINC, 3 ; If the value of PINC.3 is high,

CALL flash\_led ; skip the jump to flash the LED (switch is active high).

LDI R17, (0x01<<3 | 0x01<<2); Sets the output waveform to high.

OUT PORTB, R17

LDI R16, 11

CALL delay\_37500us ; Waits ~0.4125s.

LDI R17, (0x01<<2)

OUT PORTB, R17 ; Sets the output waveform to low.

LDI R16, 9

CALL delay\_37500us ; Waits ~0.3375s.

RJMP poll ; Keep polling.

end:

RJMP end

; Subroutine that flashes the LED on for 2 seconds, then turns it off.

flash\_led:

LDI R16, 0x00

OUT PORTB, R18 ; Sets the output of PORTB.2 to low (turns LED on).

CALL delay\_2s ; Waits ~2s.

RET

; ~2 delay

; Delay subroutine.

delay\_2s:

PUSH R16

PUSH R17 ; Stores the value of R17 and R18 AND R16 on the stack,

PUSH R18 ; so values aren't overwrited.

LDI R16, C

delay\_loop0:

LDI R17, A ; Reloads the value of A.

delay\_loop1:

LDI R18, B ; Reloads the value of B.

delay\_loop2:

NOP ; Do nothing.

NOP

NOP

NOP

NOP

NOP

NOP

DEC R18 ; Decrement nested loop counter.

BRNE delay\_loop2 ; Loop again when counter has not reached zero.

DEC R17 ; Decrement nested loop counter.

BRNE delay\_loop1 ; Loop again when counter has not reached zero.

DEC R16 ; Decrements counter.

BRNE delay\_loop0 ; Loop again when counter has not reached zero.

POP R18 ; Pops the values of R18 and R17 AND R16 back from the stack.

POP R17

POP R16

RET ; Exit the subroutine.

; ~37.5ms delay

; Delay subroutine. The delay runs for about 600,000 cycles. The value of R16 determines

; how many times the delay is run (kind of).

delay\_37500us:

PUSH R17 ; Saves values on the stack.

PUSH R18

delay\_again:

LDI R17, A ; Outter loop.

loop0:

LDI R18, B ; Nested loops.

loop1:

NOP ; Do nothing.

NOP

NOP

NOP

NOP

NOP

NOP

DEC R18

BRNE loop1

DEC R17

BRNE loop0

DEC R16

BRNE delay\_again

POP R18 ; Returns values from the stack.

POP R17

RET ; Exits the subroutine.

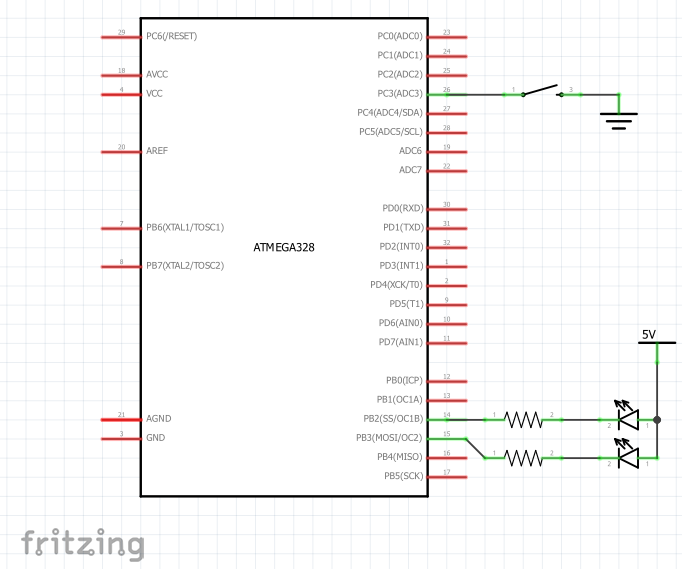
1. **SCHEMATICS**

Figure 3: Schematic for task 2.

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

**Simulation of waveform for task 1**

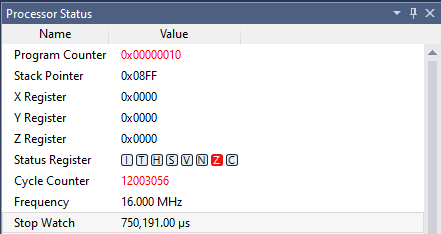


Figure 4: Atmel display of period.

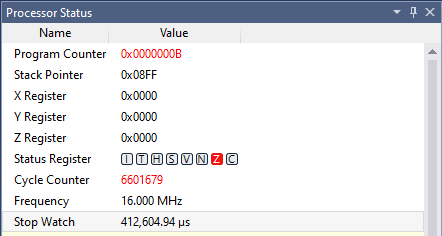


Figure 5: Atmel display of high time of wave.

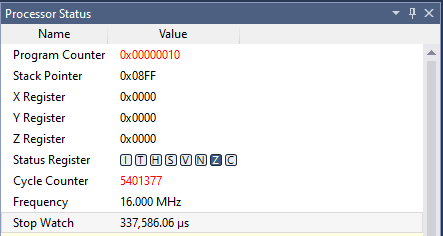


Figure 6: Atmel display of low time of wave.

**Waveform as recorded from the logic analyzer**

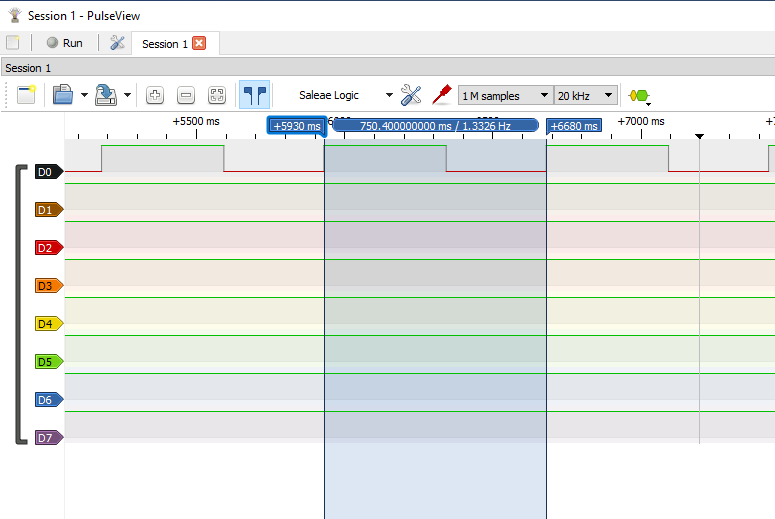
Assembly Code Implementation

Figure 7: Period (Assembly).

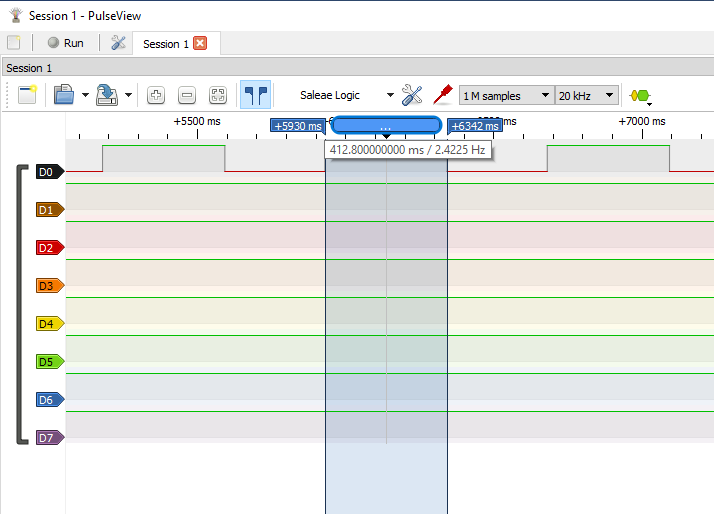
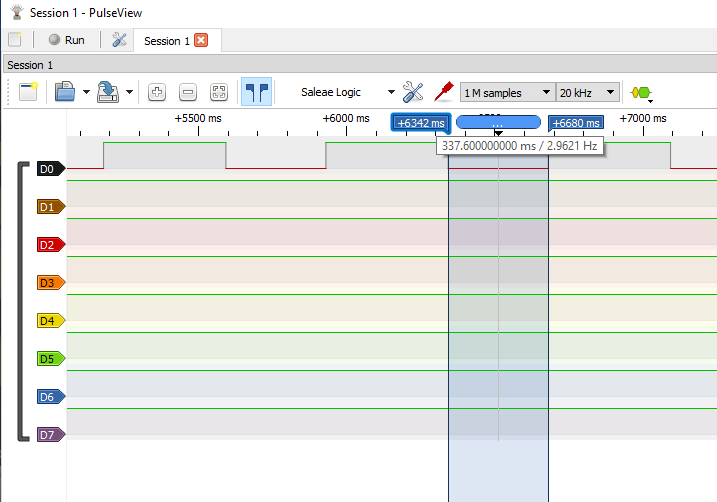


Figure 8: Low time (Assembly).

Figure 9: High time (Assembly).

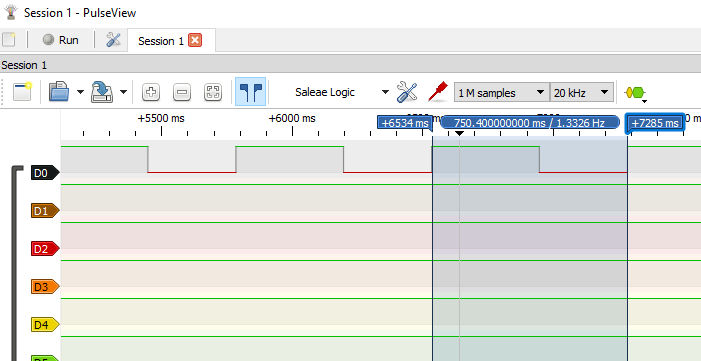
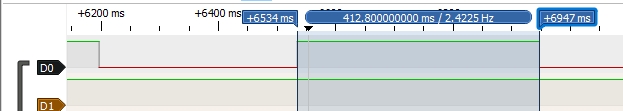
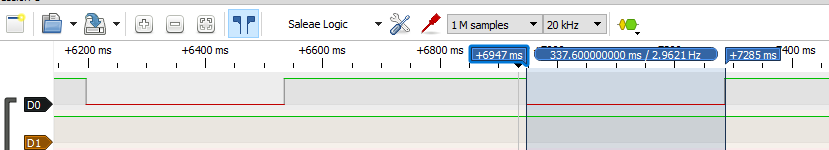
C Code Implementation

Figure 10: Low time (C).

Figure 11: High time (C).

Figure 12: Period (C).

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

**Task 1 Setup**



Figure 13: Task 1 set up.

**Task 2 Setup**

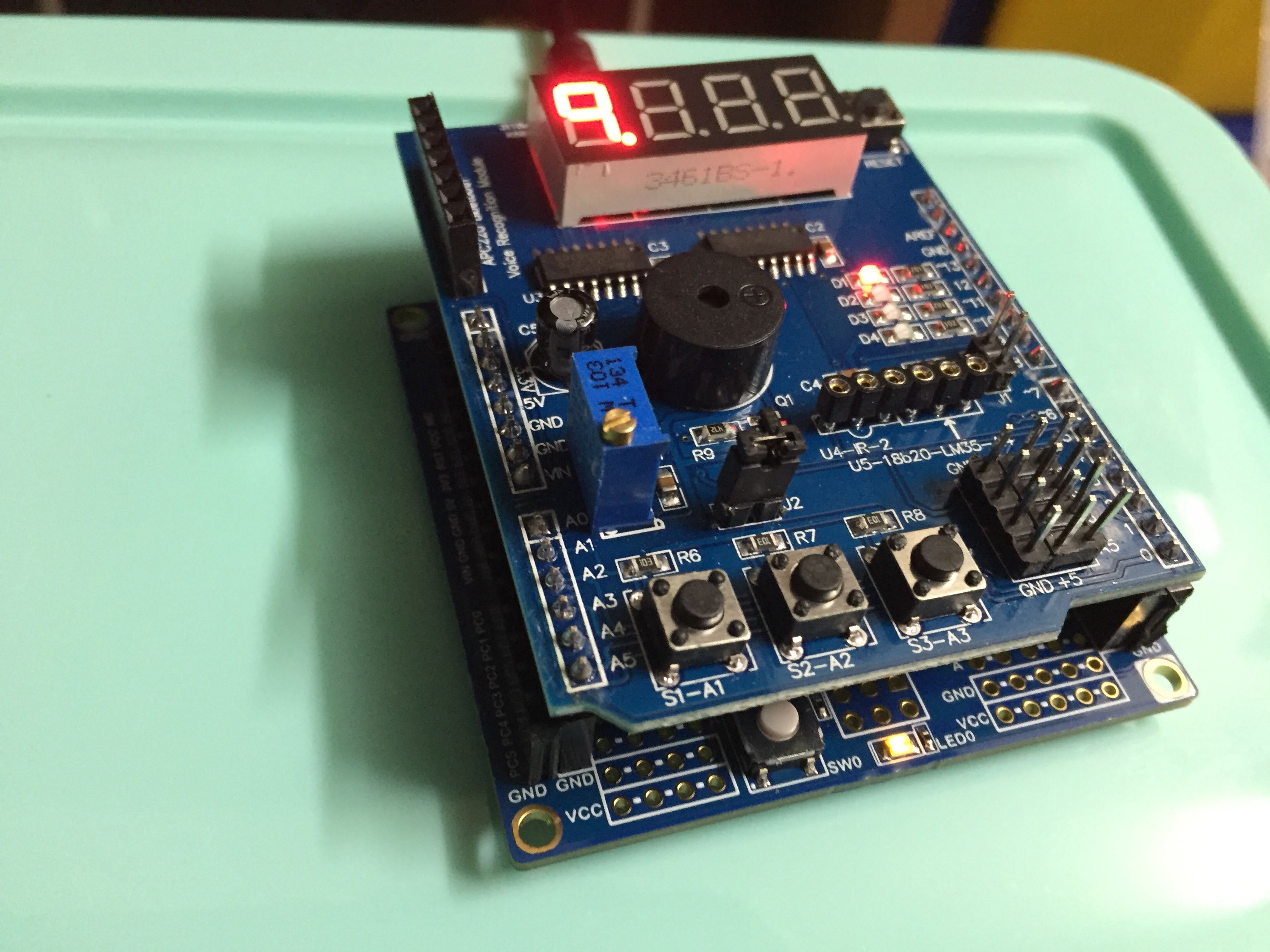


Figure 14: Task 2 set up.

1. **VIDEO LINKS OF EACH DEMO**

Task 1 in Assembly

https://youtu.be/epY2zkgyi4E

Task 1 in C

https://youtu.be/zJemcv6LIS8

Task 2 in Assembly

https://youtu.be/JazoxQC8Cms

Task 2 in C

https://youtu.be/ccZwBSx619c

1. **GITHUB LINK OF THIS DA**

**Student Academic Misconduct Policy**

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

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

Do Viet Le