CPE301 – SPRING 2023

Design Assignment 4

Student Name: David Lenzin
Student #: 2001654470
Student Email: lenzin@unlv.nevada.edu
Primary Github address: https://github.com/dlenzin15/submissions
Directory: submissions/DA4

Submit the following for all Labs:

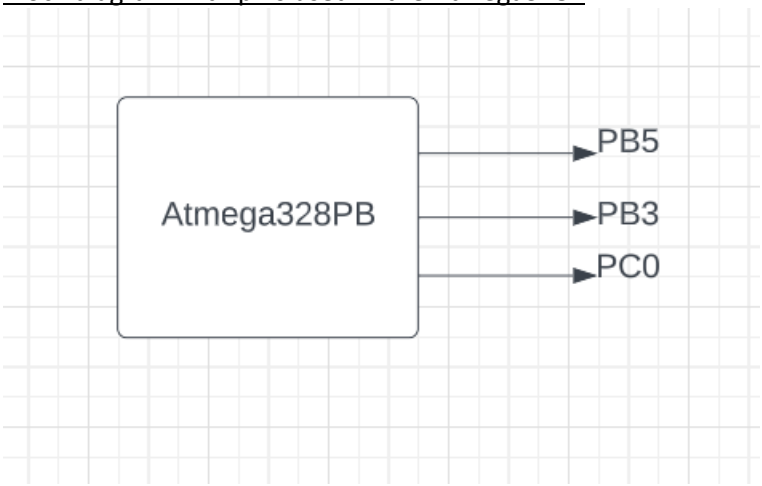
1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder 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:

- Atmega328PB
- Internal LEDs on pins PB3 and PB5
- Internal potentiometer on pin AC0/PC0

Block diagram with pins used in the Atmega328P



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

Insert initial code here

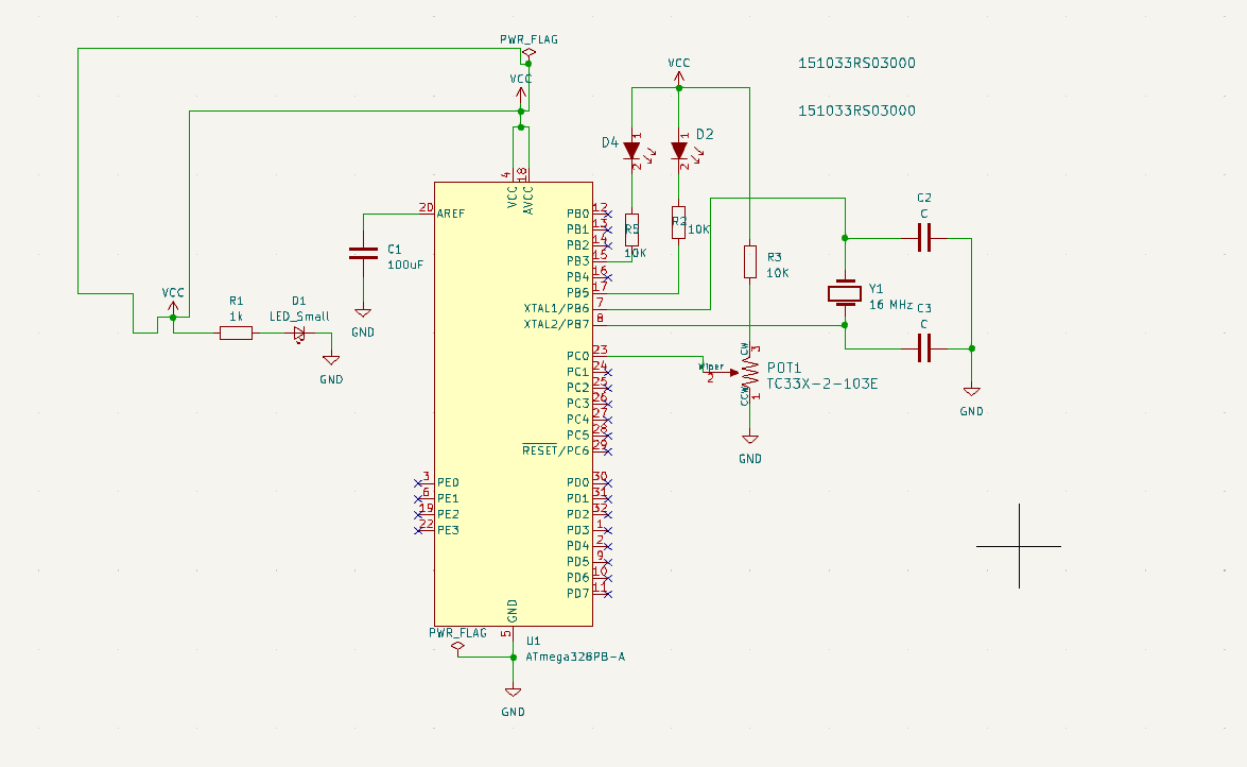
```
/*
 * DA4.c
 *
 * Created: 3/27/2023 10:52:05 AM
 * Author : David Lenzin, 2001654470
 */

/*****************************************************************/
/* Calculations for blink timer:                                 */
/* Desired blinking time is 1 kHz, and 50% duty cycle            */
/* OCR0A_VALUE = (16 MHz / (2*64*1000))-1 = 124.                 */
/* 1 kHz = 1 ms. 1s = 1000 ms. OVERFLOW_MAX = 1000 for a 1s period */
/*****************************************************************/

//Macros
#define BAUD 9600
#define MYUBRR F_CPU/16/BAUD-1
#define OCR0A_VALUE 124
#define OVERFLOW_MAX 1000
#define F_CPU 16000000UL   //Define clock speed at 8 MHz
#define ADC_RESOLUTION 1024      // 10-bit ADC resolution
#define VOLTAGE_REF 1100   // 1.1 V internal ref = 1100 mV
#define ADC_CONVERSION (VOLTAGE_REF/ADC_RESOLUTION)   // Formula to convert ADC raw data to voltage

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

```
// Global Variables
unsigned int blink_enable = 0;          // Variable to enable the blinking LED at PB3
volatile uint16_t adc_temp = 0;         // Variable to track the raw ADC readings

//Function declarations:
void UART_init(unsigned int);
void timer_init(void);
void led_init(void);
void adc_init(void);
void turn_off_led(void);
void turn_on_led(void);
void help_menu(void);
void UART_transmit_string(char *);
void read_adc(void);

void UART_init(unsigned int ubrr)
{
	//Set baud rate
	UBRR0H = (unsigned char)(ubrr>>8);
	UBRR0L = (unsigned char)ubrr;

	//Enable transmitter and receiver and reciever interrupt
	UCSR0B = (1<<RXEN0) | (1<<TXEN0) | (1<<RXCIE0);

	//Set frame format: 8 bits data, 1 stop bit
	UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01);

	sei();
}

void timer_init(void)
{
	TCCR0A |= (1<<WGM01);                           //Set Timer1 to CTC mode
	TCCR0B |= (1<<CS01) | (1<<CS00); //Set prescaler to 1024
	OCR0A = OCR0A_VALUE;                            //Load compare register value
	TIMSK0 |= (1 << OCIE0A);                //Set interrupt on compare match
}

void led_init(void)
{
	DDRB |= (1<<3) | (1<<5);   //Set PB5 and PB3 to outputs for LED
	PORTB |= (1<<3) | (1<<5);  //Turn off LEDs initially
}

void adc_init(void)
{
	// Use AVcc with external capacitor. Right adjusted result. ADC0 selected for PC0
	ADMUX = (0<<REFS1) | (1<<REFS0) | (0<<ADLAR) | (0<<MUX2) | (0<<MUX1) | (0<<MUX0);

	// Set the prescaler to 32. Don't enable the ADC until it is needed
	ADCSRA = (0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (1<<ADPS2) | (0<<ADPS1) |
(1<<ADPS0);
}

void turn_on_led(void)
{
	PORTB &= ~(1<<5); //Turn on PB5 LED
}

void turn_off_led(void)
{
	PORTB |= (1<<5); //Turn off PB5 LED
}
```

```
void help_menu(void)
{
	UART_transmit_string("**********************************DA4 Help
Screen:**********************************\n");
	UART_transmit_string("'h'\t Displays a help screen to lists all keys and functionalities\n");
	UART_transmit_string("'o'\t Turns on LED connected to PB5\n");
	UART_transmit_string("'O'\t Turns off LED connected to PB5\n");
	UART_transmit_string("'b'\t Blinks the LED connected to PB3 continuously every second\n");
	UART_transmit_string("'P'\t Turns off the LED connected to PB3\n");
	UART_transmit_string("'a'\t Reads and displays the ADC value from the potentiometer connected to
AC0/PC0\n");
	UART_transmit_string("\t ^ Please note that the ADC must be stopped before taking other inputs\n");
	UART_transmit_string("'A'\t Stops reading the ADC value from the potentiometer connected to
AC0/PC0\n");
	UART_transmit_string("**********************************************************************************
******\n\n");
}

ISR(USART0_RX_vect)
{
	// Get data from the USART data register
	unsigned char data = UDR0;
	switch(data)
	{
		case 'h':
			if (!(ADCSRA & (1 << ADEN)))	// Break if ADC is being read
				help_menu();
			break;
		case 'o':
			if (!(ADCSRA & (1 << ADEN)))	// Break if ADC is being read
				turn_on_led();
			break;
		case 'O':
			if (!(ADCSRA & (1 << ADEN)))	// Break if ADC is being read
				turn_off_led();
			break;
		case 'b':
			if (!(ADCSRA & (1 << ADEN)))	// Break if ADC is being read
				blink_enable = 1;
			break;
		case 'P':
			if (!(ADCSRA & (1 << ADEN)))	// Break if ADC is being read
				blink_enable = 0;
			PORTB |= (1<<3); //Turn LED off
			break;
		case 'a':
			ADCSRA |= (1<<ADEN);		// Enable the ADC to start reading potentiometer
			break;
		case 'A':
			ADCSRA &= ~(1<<ADEN);		// Disable the ADC to stop reading potentiometer
			break;
		default:
			break;
	}
}

ISR(TIMER0_COMPA_vect)
{
	static int overflows = 0;  // Variable to track timer overflows
	if (blink_enable == 1){			// Check if blink has been enabled
		overflows++;				// Increment overflows until we reach the value that gives us 1
second
```

```
            if (overflows == OVERFLOW_MAX) {
                PORTB ^= (1<<3);    //Toggle the PB3 LED
                overflows = 0;
            }
        }
}

void UART_transmit_string(char *data) {
        while ((*data != '\0')) {  // Check if NULL char
            while (!(UCSR0A & (1 <<UDRE0))); // Wait for register to be
            UDR0 = *data; // Store data in the data register
            data++;
        }
}

void read_adc(void)
{
        ADCSRA |= (1<<ADSC); // Start a conversion
        adc_temp = ADC;            // Read the raw ADC value
}


int main(void)
{
        UART_init(MYUBRR);  //Initialize USART
        UART_transmit_string("Connected!\n");   //Welcome message
        timer_init();       //Initialize Timer
        led_init();             //Initialize LEDs
        adc_init();             //Initialize ADC
        help_menu();        //Print the help menu upon startup/reboot

        while (1) {
            if (ADCSRA & (1 << ADEN))  //Check if ADC has been enabled
            {
                char buffer[100];          //Buffer to read ADC
                read_adc();
                uint16_t adc_voltage = adc_temp * ADC_CONVERSION;
                sprintf(buffer, "%d mV\r\n", adc_voltage);     //Read the adc value into the buffer
                UART_transmit_string(buffer);     //Send the adc value to the terminal
                _delay_ms(100);     //Delay for 0.10 seconds
            }
        }
        return 0;
}
```

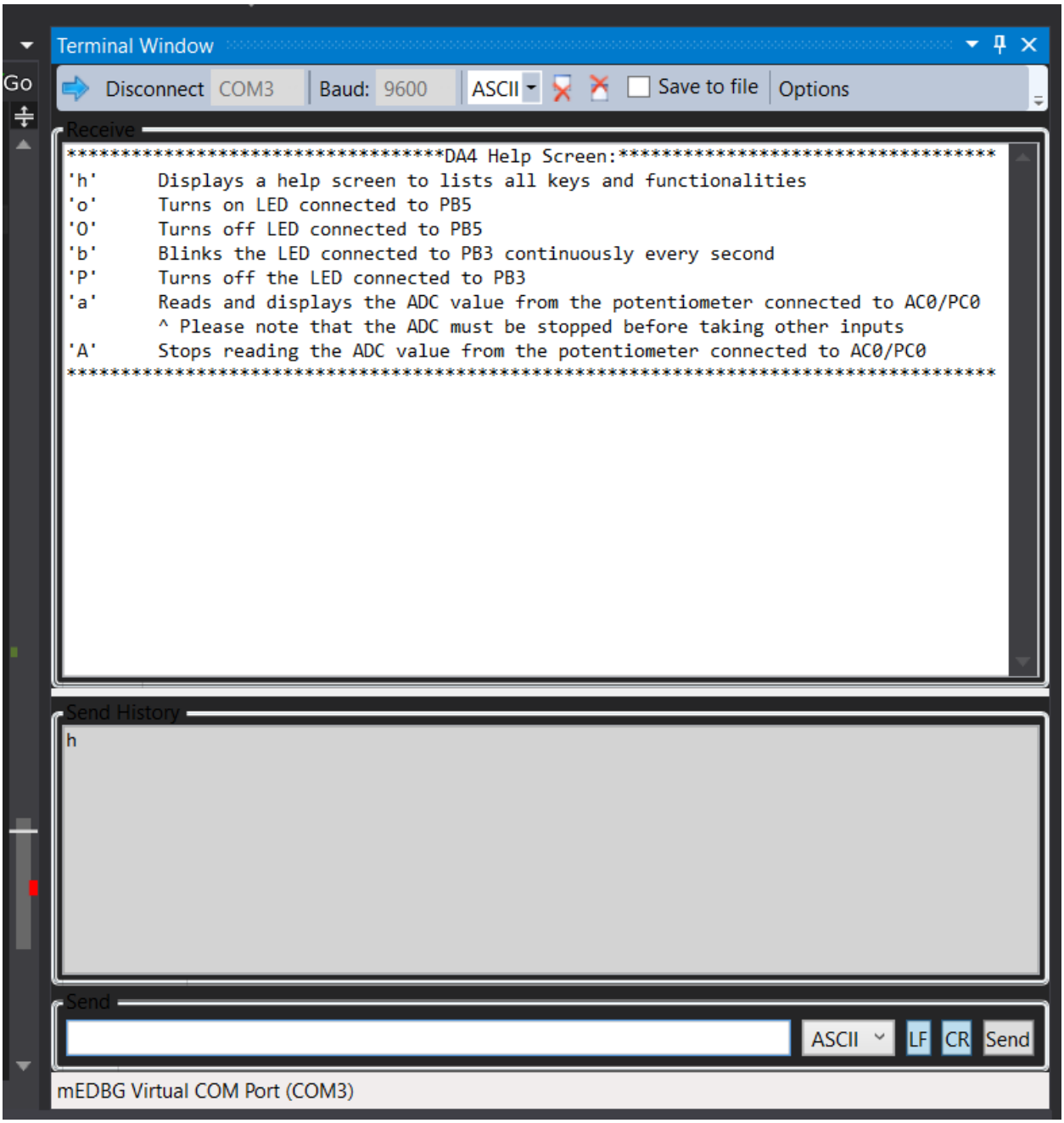
3. SCHEMATICS



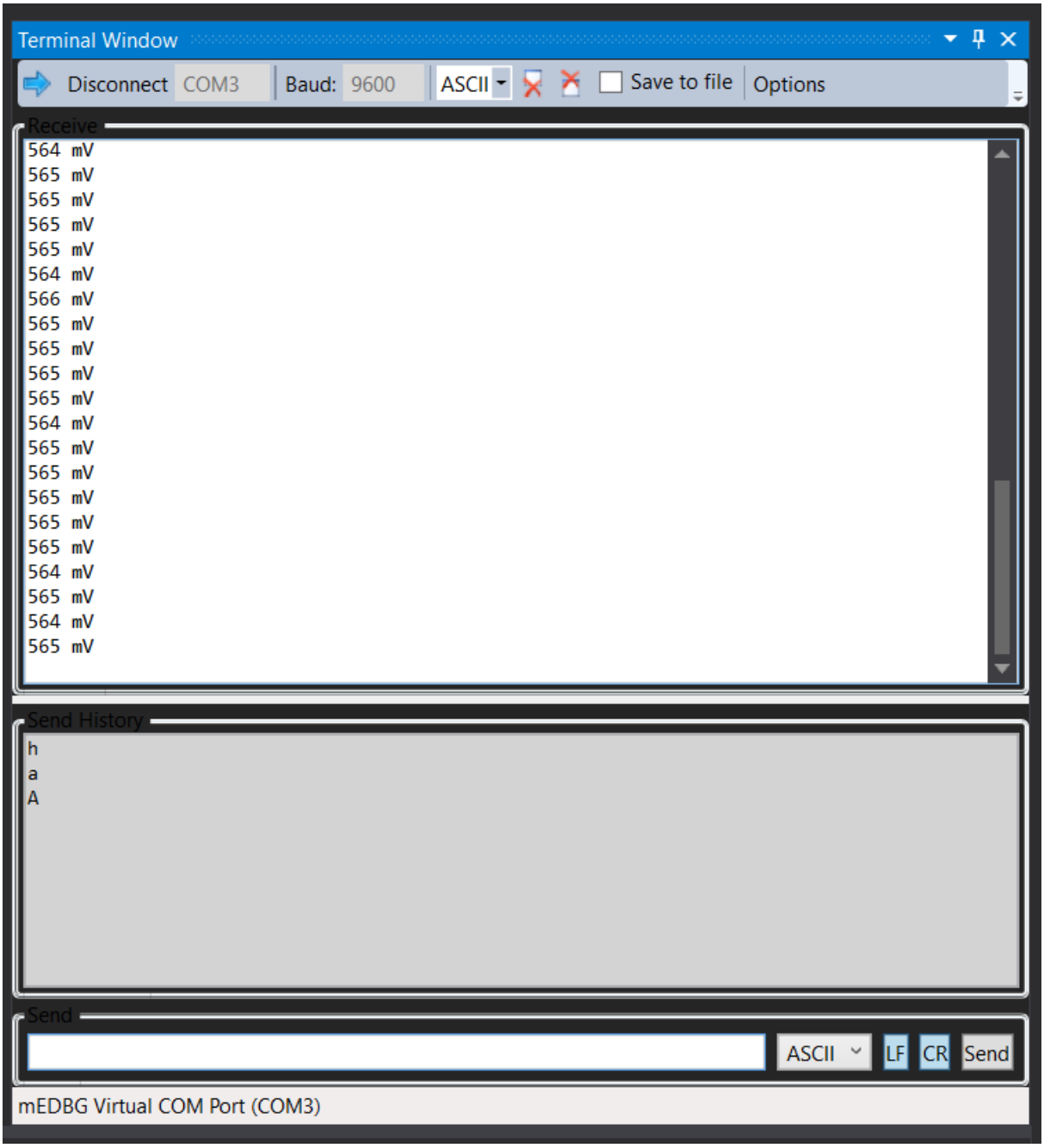
4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

Note: Only the help screen and ADC readings had terminal output (Tasks 1.1 and 1.4):

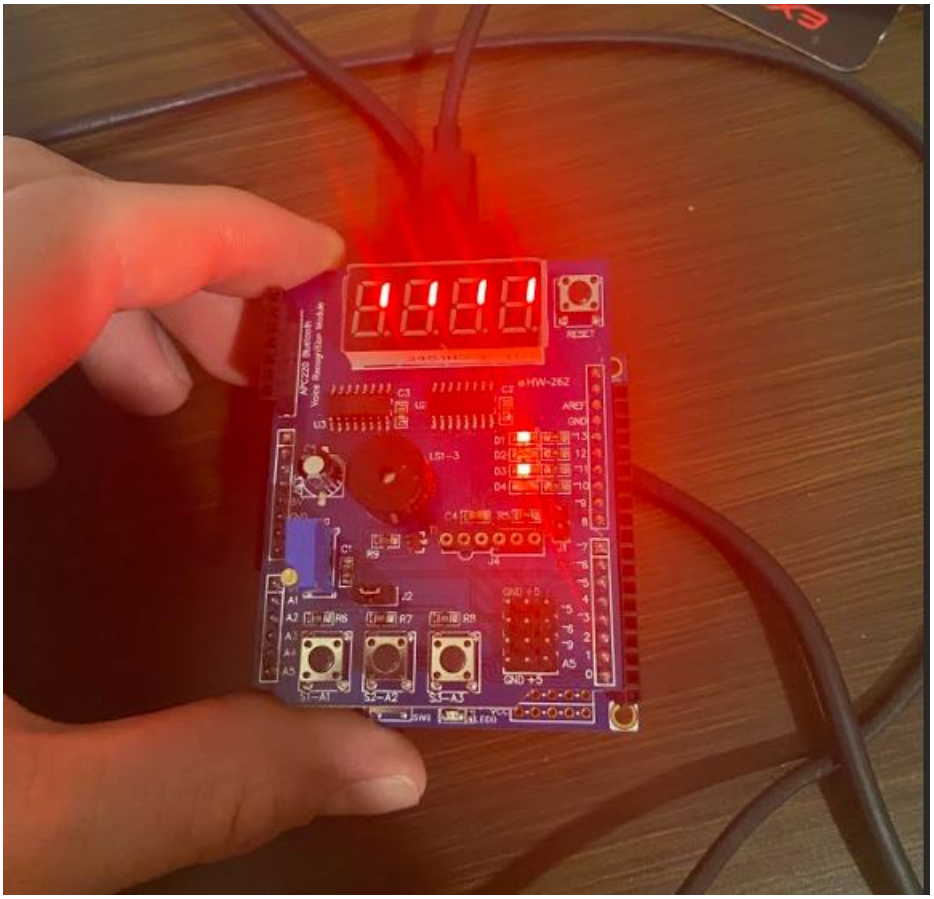
Help Menu:



ADC Readings:



5. SCREENSHOT OF EACH DEMO (BOARD SETUP)



- The board was set to turn on the PB5 LED and blink the PB3 LED in this screenshot

6. VIDEO LINKS OF EACH DEMO

Link to complete demonstration: https://youtu.be/YM8_BLL70RQ

7. GITHUB LINK OF THIS DA

https://github.com/dlenzin15/submissions/tree/main/DA4

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
David Lenzin