CPE301 – SPRING 2021

MIDTERM 1

Student Name: Elmer Mejia

Student #: 5003824808

Student Email: mejiae4@unlv.nevada.edu

Primary Github address: https://github.com/cpemejia/design\_assignments.git

Directory: design\_assignments

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

-Atmega328pb - Multi-function shield

-Microchip Studio -LM35

Table

Description automatically generated with medium confidence



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

/\*

\* Mid1.c

\*

\* Created: 3/28/2021 8:00:23 PM

\* Author : ElmerOMejia

\* References includes modules and lecture videos, github

\*/

#define *F\_CPU* 16000000UL // 16M w/ 9600 baud rate

#define BAUD 9600

#define BAUD\_PRESCALE (((*F\_CPU* / (BAUD \* 16UL))) - 1)

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/atomic.h>

#include <string.h>

// declaration of variables

volatile unsigned char in\_data[8];

volatile unsigned char in\_cmd[8];

volatile unsigned char count;

volatile unsigned char command;

*uint16\_t* adc\_value; //Variable used to store the value read from the ADC

char buffer[5]; //Output of the itoa function

// Variables to hold current settings

unsigned int freq = 0;

float duty\_cycle;

void usart\_putc (char send)

{

// send data

while ((UCSR0A & (1 << UDRE0)) == 0) {};

UDR0 = send;

}

void usart\_puts (const char \*send)

{

// send each character in string

while (\*send) {

usart\_putc(\*send++);

}

}

void usart\_ok()

{

usart\_puts("OK\r\n"); // OK confirm

}

void copy\_command ()

{

*ATOMIC\_BLOCK*(*ATOMIC\_FORCEON*) { // copy content and use atomic so no info lost

*memcpy*(in\_cmd, in\_data, 8);

*memset*(in\_data[0], 0, 8);

}

}

unsigned long parse\_assignment ()

{

char \*pch;

char cmdValue[16];

// search for equal sign

pch = *strchr*(in\_cmd, '=');

// when found copy the following chars

*strcpy*(cmdValue, pch+1);

// convert to int and return

return *atoi*(cmdValue);

}

void init\_timer1(unsigned int freq)

{

DDRB |= (1<<PB2);

TCNT1 = 0;

ICR1 = freq;

OCR1B = 5000;

TCCR1B |= (1 << WGM12) | (1 << WGM13);

// Mode 12, CTC on ICR1

TIMSK1 |= (1 << OCIE1B) | (1 << ICIE1);

//Set interrupt on compare match

TCCR1B |= (1 << CS12);

// set prescaler to 256 and starts the timer

}

void init\_timer3(unsigned int dc)

{

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

TCNT3 = 0;

ICR3 = 50000;

OCR3B = dc\*ICR3;

TCCR3B |= (1 << WGM32) | (1 << WGM33);

// Mode 12, CTC on ICR3

TIMSK3 |= (1 << OCIE3B) | (1 << ICIE3);

//Set interrupt on compare match

TCCR3B |= (1 << CS32);

// set prescaler to 256 and starts the timer

}

//f\_out = 16M/(2\*64\*63) =

void adc\_init(void){

ADCSRA |= ((1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)); //16Mhz/128 = 125Khz the ADC reference clock

ADMUX |= (1<<REFS0); //Voltage reference from Avcc (5v)

ADCSRA |= (1<<ADEN); //Turn on ADC

ADCSRA |= (1<<ADSC); //Do an initial conversion because this one is the slowest and to ensure that everything is up and running

}

*uint16\_t* read\_adc(*uint8\_t* channel){

ADMUX &= 0xF0; //Clear the older channel that was read

ADMUX |= channel; //Defines the new ADC channel to be read

ADCSRA |= (1<<ADSC); //Starts a new conversion

while(ADCSRA & (1<<ADSC)); //Wait until the conversion is done

ADCSRA|=(1<<ADIF);

return ADCW; //Returns the ADC value of the chosen channel

}

void process\_command()

{

usart\_puts("Help Menu:\n");

usart\_puts("h - help screen\n");

usart\_puts("t - T in C of LM34/35\n");

usart\_puts("f - T in F of LM34/35\n");

usart\_puts("o - ON LED @PB5\n");

usart\_puts("O - OFF LED @PB5\n");

usart\_puts("d - DC for LED @PB3\n");

usart\_puts("i - Frequency for LED @PB2\n");

usart\_puts("'=' after d or i for change\n");

usart\_puts("0<d<100\n");

usart\_puts("5k<i<65k\n");

usart\_puts("\n");

switch (in\_cmd[0]) {

case 'h': // h for help menu

usart\_puts("Help Menu:\n");

usart\_puts("h - help screen\n");

usart\_puts("t - T in C of LM34/35\n");

usart\_puts("f - T in F of LM34/35\n");

usart\_puts("o - ON LED @PB5\n");

usart\_puts("O - OFF LED @PB5\n");

usart\_puts("d - DC for LED @PB3\n");

usart\_puts("i - Frequency for LED @PB2\n");

break;

case 't':

usart\_puts("Reading channel "); // read temp on adc

usart\_puts(" : \n");

adc\_value = read\_adc(0);

*itoa*(adc\_value, buffer, 10);

usart\_puts(buffer);

usart\_puts(" \n ");

break;

case 'f':

usart\_puts("Reading channel "); // read temp on adc convert to F

usart\_puts(" : \n");

adc\_value = read\_adc(0);

adc\_value = ((adc\_value \* 9) / 5) + 32;

*itoa*(adc\_value, buffer, 10);

usart\_puts(buffer);

usart\_puts(" \n ");

break;

case 'o':

PORTB &= ~(1<<PB5); // output PB5 on

break;

case 'O':

PORTB |= (1<<PB5); // output PB5 off

break;

case 'd':

if (in\_cmd[1] == '=') {

// Do the query action for S

duty\_cycle = parse\_assignment();

init\_timer3(duty\_cycle);

}

else{

init\_timer3(50);

}

break;

case 'i':

if (in\_cmd[1] == '=') {

// Do the query action for S

freq = parse\_assignment();

init\_timer1(freq);

}

else{

init\_timer1(10000);

}

break;

default:

usart\_puts("INCORRECT 'h' for list\r\n");

break;

}

}

void UART0\_init(void)

{

// Turn on USART hardware (RX, TX)

UCSR0B |= (1 << RXEN0) | (1 << TXEN0);

// 8 bit char sizes

UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01);

// Set baud rate

UBRR0H = (BAUD\_PRESCALE >> 8);

UBRR0L = BAUD\_PRESCALE;

// Enable the USART Receive interrupt

UCSR0B |= (1 << RXCIE0 );

}

void out\_init(void)

{

DDRB |= (1<<PB5); // set PB5 output

DDRC |= (1<<PC0); // set PC0 as output

}

int main(void)

{

UART0\_init();

adc\_init();

out\_init();

// Globally enable interrupts

*sei*();

while(1) {

if (command == 1) {

copy\_command(); // copy command

process\_command(); // process the command

command = 0;

usart\_ok(); // OK for confirmation

}

}

}

ISR (USART0\_RX\_vect)

{

// get data from uart

in\_data[count] = UDR0;

if (in\_data[count] == '\n') {

command = 1;

// reset count to 0 at end of line

count = 0;

} else {

count++;

}

}

ISR (TIMER1\_COMPB\_vect)

{

PORTB &= ~(1<<PB2); // PB2 on on comp B match

}

ISR (TIMER1\_CAPT\_vect)

{

PORTB |= (1<<PB2); // PB2 off on Capture

TCNT1 = 0;

}

ISR (TIMER3\_COMPB\_vect)

{

PORTB &= ~(1<<PB3); // PB3 on on comp B match

}

ISR (TIMER3\_CAPT\_vect)

{

PORTB |= (1<<PB3); // PB3 off on Capture

TCNT3 = 0;

}

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

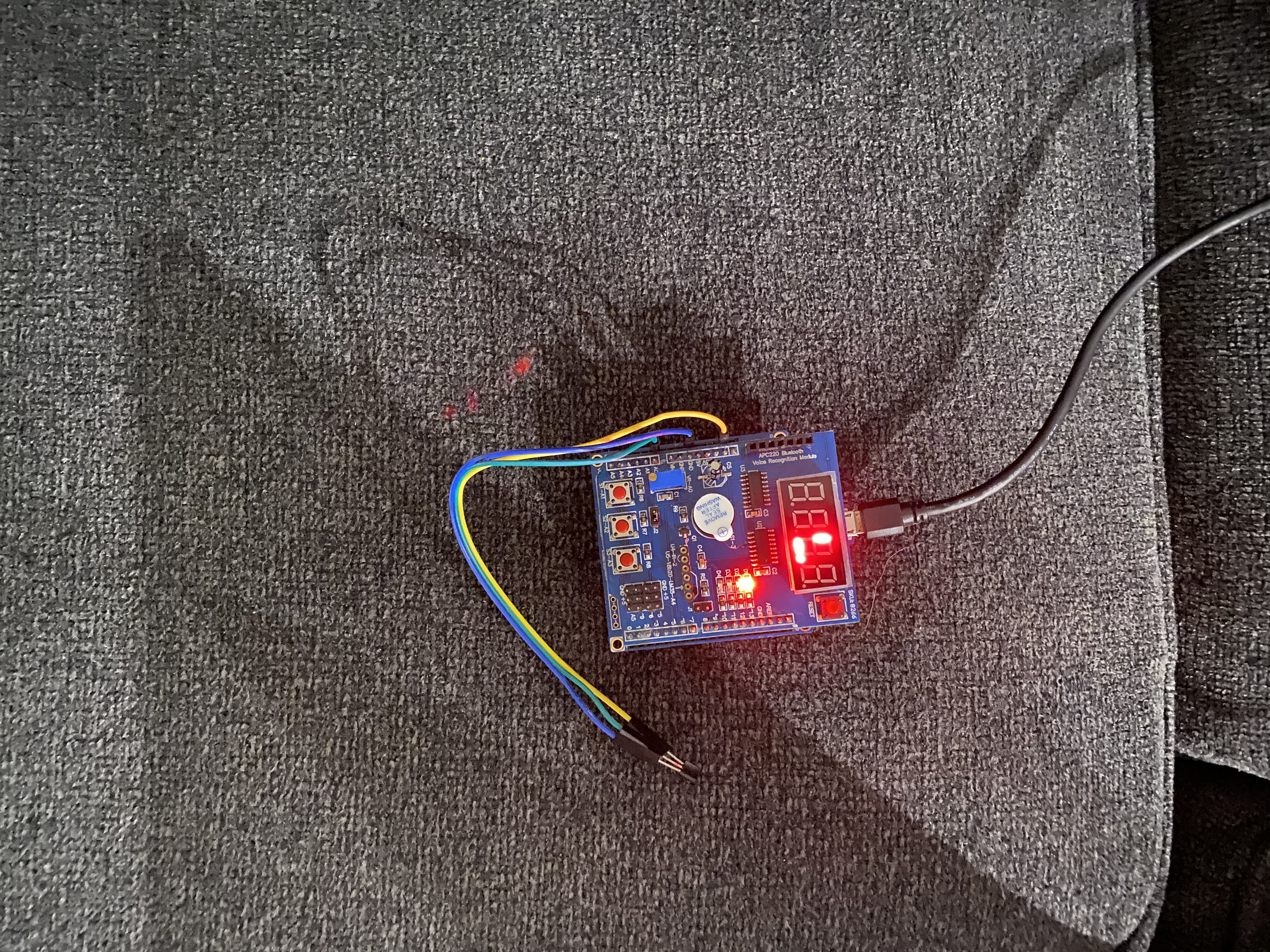
Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

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



1. **VIDEO LINKS OF EACH DEMO**

[MID\_1 - YouTube](https://youtu.be/mTgwk5wOknI)

1. **GITHUB LINK OF THIS DA**

https://github.com/cpemejia/design\_assignments.git

**Student Academic Misconduct Policy**

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

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

Elmer Mejia