CPE301 – SPRING 2020

Design Assignment 5

The goal of the assignment is to modify the above codes to do the following

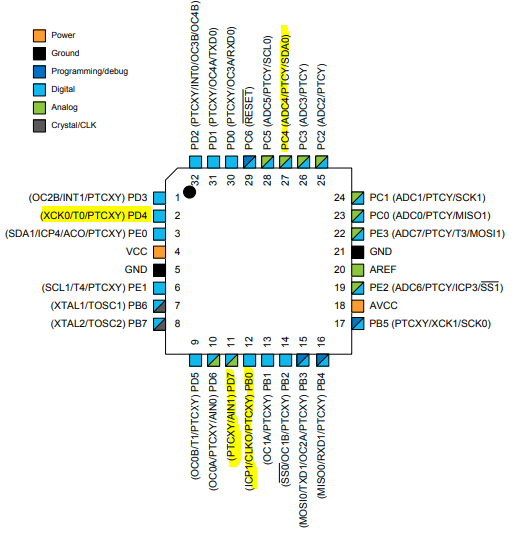
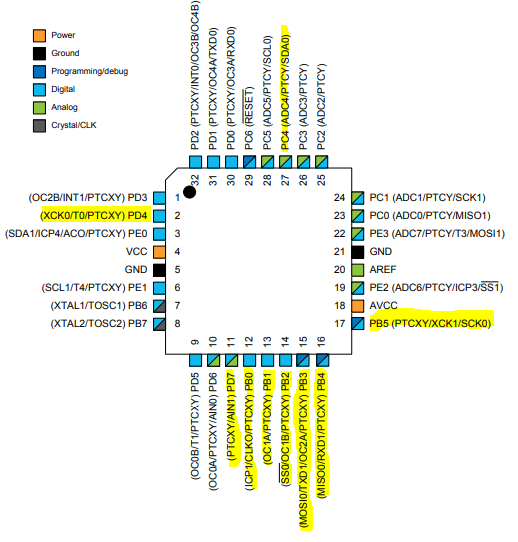
1. Using the Multi-function shield, using a custom serial transfer protocol display the temperature sensor value (int only) of LM34/35 on to the seven-segment display.
2. Using the Multi-function shield, using the SPI protocol display the temperature sensor value (int only) of LM34/35 on to the seven-segment display.
3. Using one-wire interface display the temperature sensor for DS18B20 on the UART terminal
4. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

ATmega328PB Xplained mini USB cable Atmel Studio 7

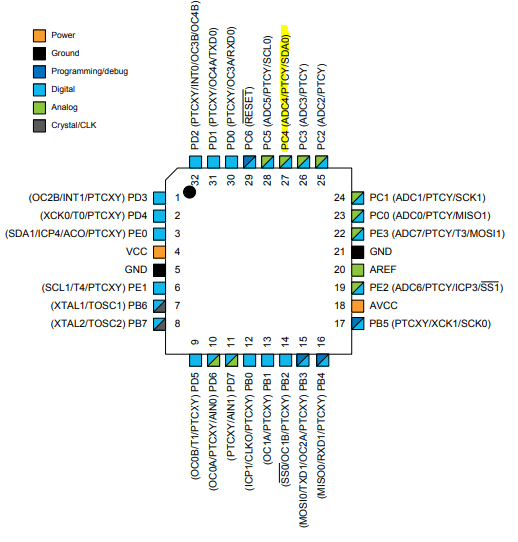
Data Visualizer 7 segment display LM35

Multifunction Shield DS18B20

**TASK 1:** **TASK 2:**

**TASK 3:**



Notes

* The lm35 is not working, I talk to Dr. Venki about it, and he told me to just leave a note here.
* For the schematics, I separated the part for the 7 segment because it doesn’t change between task 1 and 2.
* The result of task 1 and 2 are the same: 156 (because lm35 is broken), the demo vids will show that.
* Just to check that my code works, I used an temperature sensor using mux 1000, just to show that code works as it should and its just that the component lm35 is broken.
* There’s no DS18B20 component on fritz so I substitute temp sensor as to represent the part.

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

**Ads**

/\*

\* DA5\_T1.c

\*

\* Created: 4/21/2020 10:09:41 AM

\* Author : John Paulo Lumbres

\*/

/\*LM35 is broken, tried applying direct heat, yet value still doesn't change\*/

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <stdio.h>

#include <stdlib.h>

#define UBRR\_9600 103

#define LATCH\_DIO 4 //PD4

#define CLK\_DIO 7 //PD7

#define DATA\_DIO 0 //PB0

//declaring variable for controlling and storing values

volatile unsigned int adc\_val;

//number 0 to 9

const *uint8\_t* segment\_map[] ={0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0X80, 0X90};

//digit select

const *uint8\_t* segment\_select[] = {0xF1,0xF2,0xF4,0xF8};

void adc\_init(void)

{

//AVcc reference, left adjust, ADC4 = PC4 = LM35,ADC enable

ADMUX =(1<<REFS0)|(1<<ADLAR)|(1<<MUX2);

ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1); //prescaler 64

}

void read\_adc(void)

{

unsigned char i=4;

adc\_val = 0; //initial value

while(i--){ //get 4 values of ADC

ADCSRA |= (1<<ADSC);

while((ADCSRA & (1<<ADIF))==0);

ADCSRA |= (1<<ADIF);

adc\_val+=ADCH;

*\_delay\_ms*(50);

}

adc\_val = adc\_val/4; //have the average

}

void display(*uint8\_t* sel, *uint8\_t* light){

for(*uint8\_t* i=0; i<8; i++){ //writing data of segment map

if (0 == (light & \_BV(7 - i))) {

PORTB &= (0<<DATA\_DIO);

} else {

PORTB |= (1<<DATA\_DIO);

}

//send pulse to clk dio

PORTD |=(1<<CLK\_DIO);

PORTD &=(0<<CLK\_DIO);

}

for (*uint8\_t* i=0; i<8; i++){ //writing data of segment select

if (0 == (sel & \_BV(7 - i))) {

PORTB &= (0<<DATA\_DIO);

} else {

PORTB |= (1<<DATA\_DIO);

}

//send pulse to clk dio

PORTD |=(1<<CLK\_DIO);

PORTD &=(0<<CLK\_DIO);

}

}

int main(void)

{

DDRC &= ~(1<<PC4); //pc4 as input

PORTC |=(1<<PC4); //pc4 enabled

DDRB |= (1<<DATA\_DIO); //data\_dio is output for portb

DDRD |= (1<<CLK\_DIO)|(1<<LATCH\_DIO);

//clk\_dio and latch\_dio are output for portd

adc\_init(); //initialize adc

int val1, val2, val3; //declare int variables

while (1)

{

read\_adc(); //read adc val from sensor

//mod adc val to get one's value

val1 = (adc\_val)%10;

//divide and mod adc val to get ten's place value

val2 = (adc\_val/10)%10;

//divide and mode val to get the hundred's place value

val3 = (adc\_val/100)%10;

for(int i=0; i<101;i++){ //repeat 100 times

PORTD &= ~(1<<LATCH\_DIO); //latch to get values

//display one's value on the last 7 segment display

display(segment\_select[3],segment\_map[val1]);

PORTD |= (1<<LATCH\_DIO); //close latch

*\_delay\_ms*(5);

PORTD &= ~(1<<LATCH\_DIO);

//display ten's value on the 2nd to the last 7 segment display

display(segment\_select[2],segment\_map[val2]);

PORTD |= (1<<LATCH\_DIO);

*\_delay\_ms*(5);

PORTD &= ~(1<<LATCH\_DIO);

//display hundred's value on the first 7 segment display

display(segment\_select[1],segment\_map[val3]);

PORTD |= (1<<LATCH\_DIO);

*\_delay\_ms*(5);

}

}

}

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

**As**

/\*

\* DA5\_T2.c

\*

\* Created: 4/21/2020 3:36:49 PM

\* Author : John Paulo Lumbres

\*/

/\*LM35 is broken, tried applying direct heat, yet value still doesn't change\*/

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#define UBRR\_9600 103

#define LATCH 2 //PD4

#define CLK 5 //PD7

#define DATA 3 //PB0

#define SHIFT\_REGISTER DDRB

#define SHIFT\_PORT PORTB

//declaring variable for controlling and storing values

volatile unsigned int adc\_val;

//number 0 to 9

const *uint8\_t* segment\_map[] ={0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0X80, 0X90};

//digit select

const *uint8\_t* segment\_select[] = {0xF1,0xF2,0xF4,0xF8};

void adc\_init(void)

{

//AVcc reference, left adjust, ADC4 = PC4 = LM35,ADC enable

ADMUX =(1<<REFS0)|(1<<ADLAR)|(1<<MUX2);

ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1); //prescaler 64

}

void read\_adc(void)

{

unsigned char i=4;

adc\_val = 0; //initial value

while(i--){ //get 4 values of ADC

ADCSRA |= (1<<ADSC);

while((ADCSRA & (1<<ADIF))==0);

ADCSRA |= (1<<ADIF);

adc\_val+=ADCH;

*\_delay\_ms*(50);

}

adc\_val = adc\_val/4; //have the average

}

void spi\_init(){

//SPI enabled, master mode

SPCR0 |= (1<<SPE)|(1<<MSTR);

//open latch

SHIFT\_PORT &= ~(1<<LATCH);

}

void store(){

//opens and close the latch to store

SHIFT\_PORT |= (1<<LATCH);

SHIFT\_PORT &= ~(1<<LATCH);

}

void display(*uint8\_t* data){

//store data to SPDR0 register

SPDR0 = data;

//send the data to slave

while(!(SPSR0 &(1<<SPIF)));

}

int main(void)

{

DDRC &= ~(1<<PC4); //PC4 as input

PORTC |=(1<<PC4); //enable pc4

//have data, latch and clk outputs on portb

SHIFT\_REGISTER |= (1<<DATA)|(1<<LATCH)|(1<<CLK);

//initialize portb to 0;

SHIFT\_PORT &= ~((1<<DATA)|(1<<LATCH)|(1<<CLK));

adc\_init();

spi\_init();

int val1, val2, val3;

while (1)

{

read\_adc();

//mod adc val to get one's value

val1 = (adc\_val)%10;

//divide and mod adc val to get ten's place value

val2 = (adc\_val/10)%10;

//divide and mode val to get the hundred's place value

val3 = (adc\_val/100)%10;

for (int i=0; i<35000; i++) { //repeat to see display

display(segment\_map[val1]); //send one's value

display(segment\_select[3]); //send which led to turn on

store();

display(segment\_map[val2]); //send ten's value

display(segment\_select[2]); //send which led to turn on

store();

display(segment\_map[val3]); //send hundered's value

display(segment\_select[1]); //send which led to turn on

store();

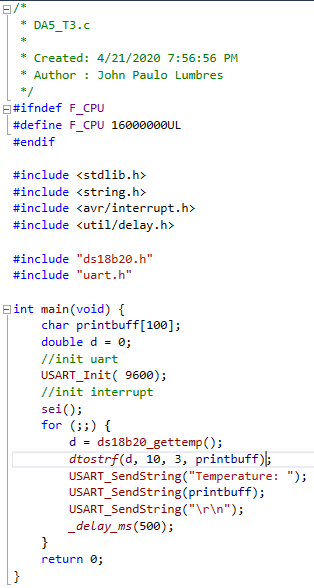
}

}

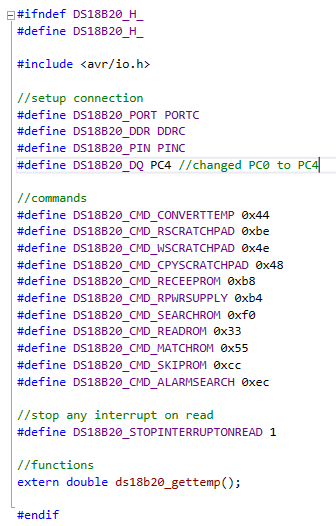
}

1. **DEVELOPED MODIFIED CODE OF TASK 3/A**

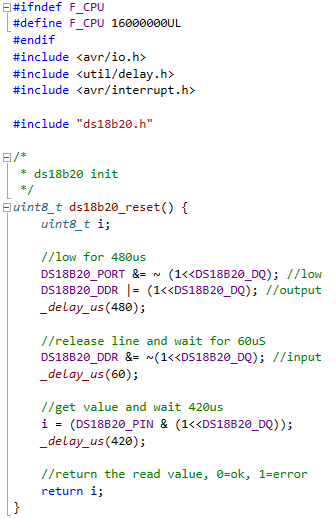
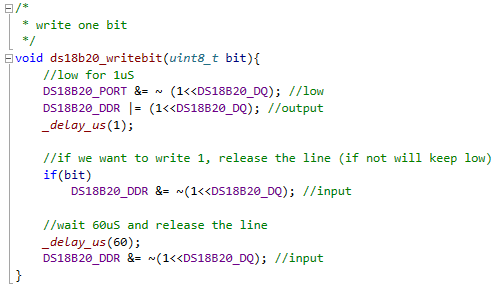
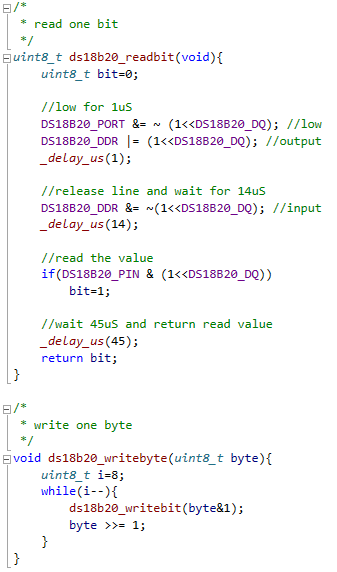
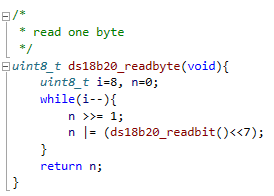
**Main.c**



**Ds18b20.h**

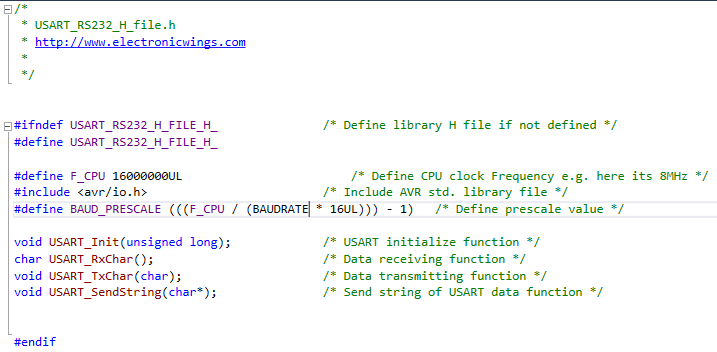


**Ds18b20.c**

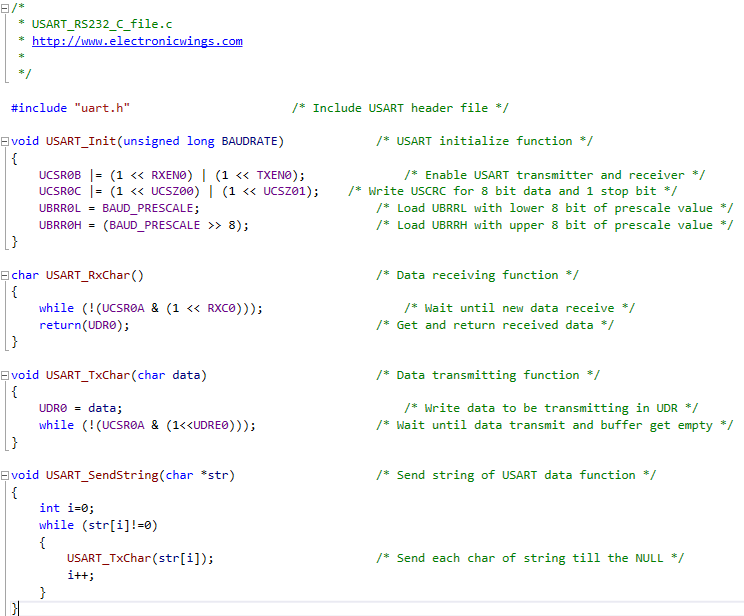
     




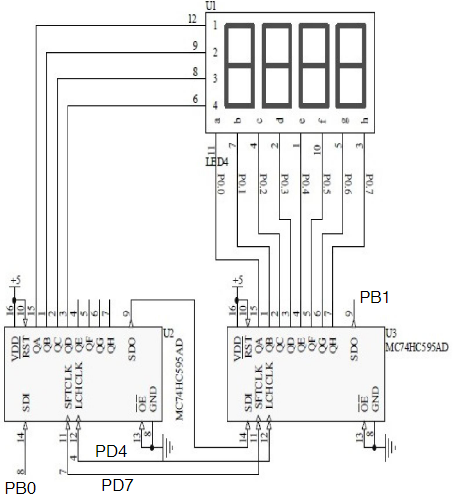
**Uart.h**



**Uart.c**

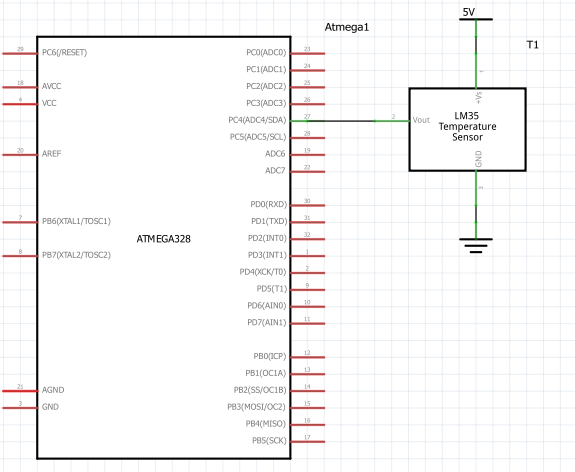


1. **SCHEMATICS**

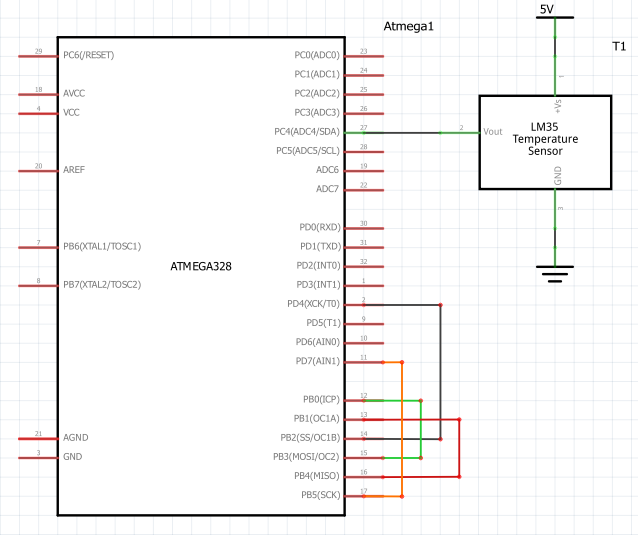


**I’ll be posting the schematic for operating the 7-segments separately, but the connections for sensors and SPI will be done on fritzing.**

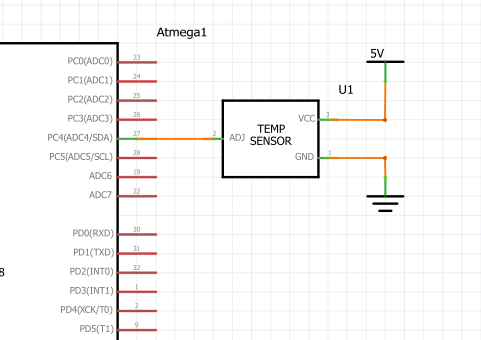
**TASK 1:**



**TASK 2:**



**TASK 3:**

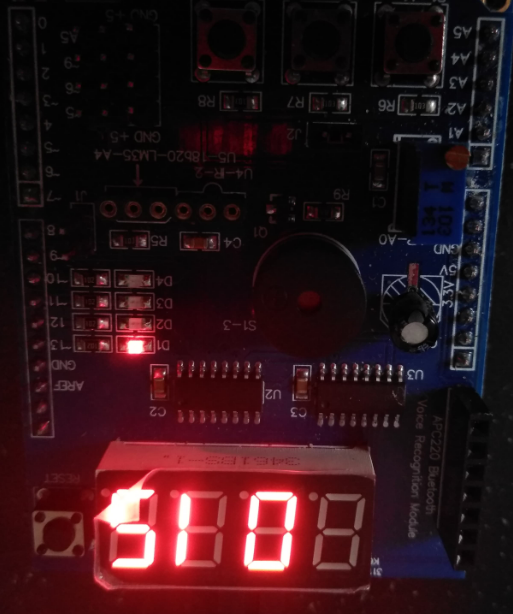
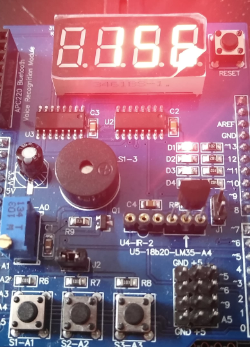


1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

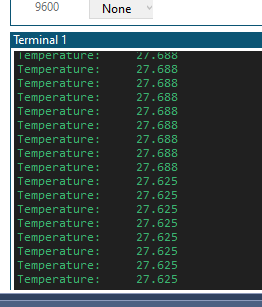
Both Task 1 and Task two display value of broken sensor, which is 156. On demo, you will see that both tasks outputs 156. I used internal sensor just to see if it works, and it works, just that the value is in centigrade.

**TASK 1 and 2:**

**using broken sensor using internal sensor**

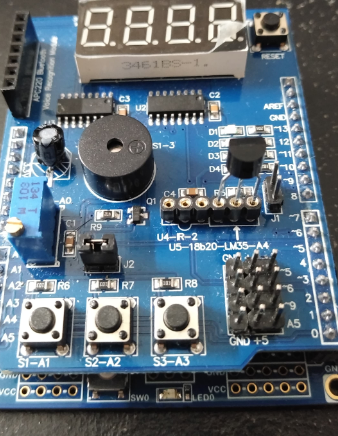
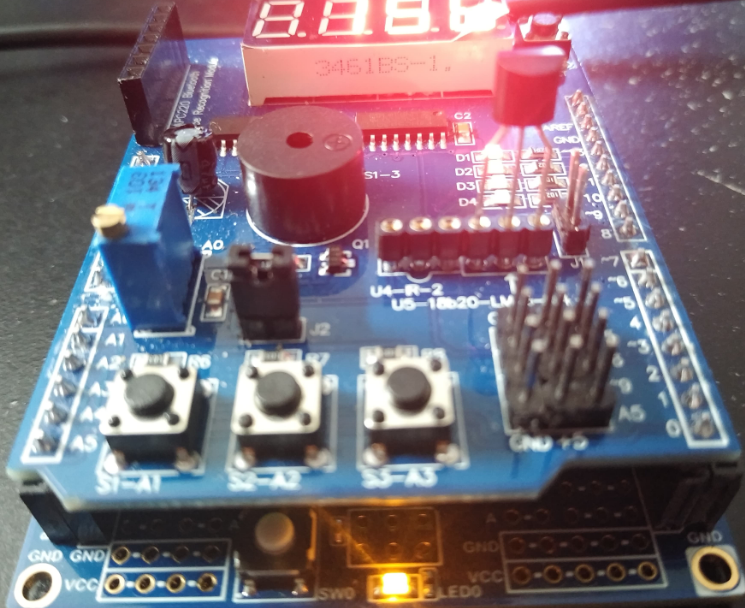


**TASK 3:**



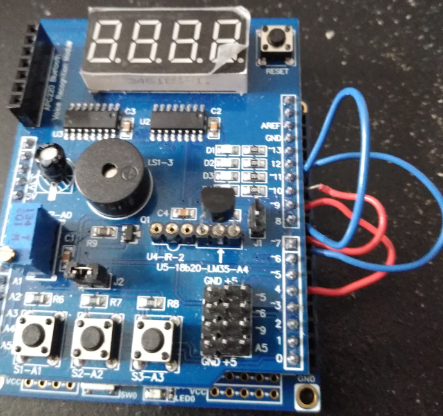
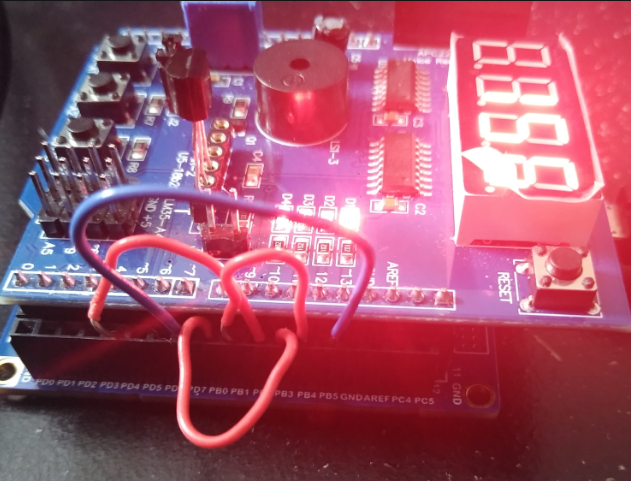
1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

**TASK 1:**



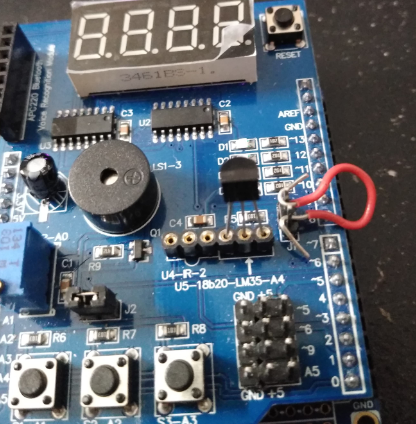
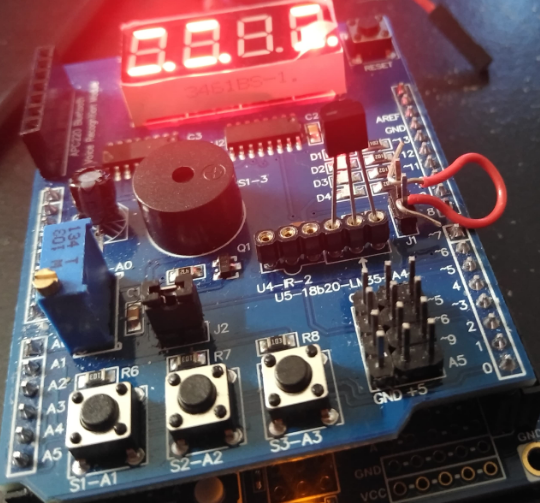
**TASK 2:**

**Recreated the circuit for a top view look**



**TASK 3:**

Wire is there to connect jumper.



1. **VIDEO LINKS OF EACH DEMO**

Task 1 – <https://youtu.be/WoEyfLpDsu4>

Task 2 – <https://youtu.be/hJMT37sSHkQ>

Task 3 – <https://youtu.be/5OdyAWBk9A4>

1. **GITHUB LINK OF THIS DA**

<https://github.com/lumbrj1/submission/tree/master/DesignAssignments>

**Student Academic Misconduct Policy**

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

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

John Paulo Lumbres