CPE301 – SPRING 2019

Design Assignment 5

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>

The goal of the assignment is to develop the above code 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.

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

Components

• XPlained Mini Board

* ATmega328PB
* Programmer

• Atmel Studios

* Compiler
* Simulator

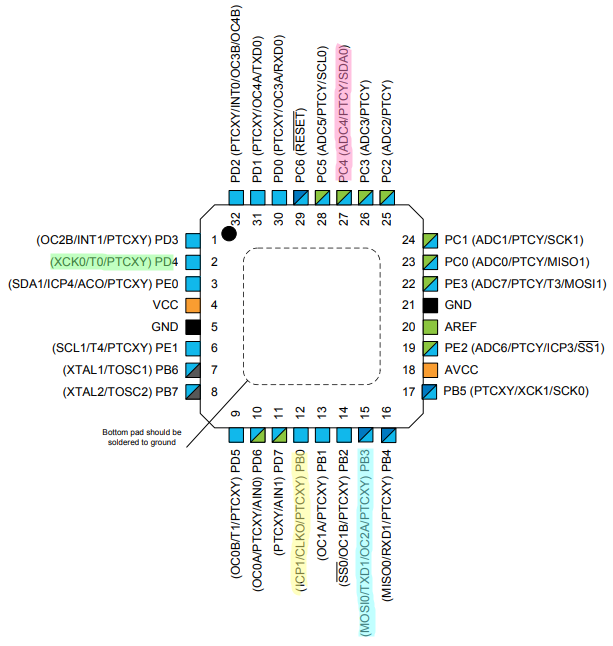
• LM35 Temperature Sensor

• DS18B20 Temperature Sensor

• Arduino Shield

* Seven-Segment Displays

Block Diagram



PC4 used as temperature analog input.

PB0 used as custom serial output.

PB3 used as SPI Master Out.

PD4 used as latch clock.

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

Task 1 Code:

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#define DDR\_SHIFT DDRB

#define PORT\_SHIFT PORTB

#define DDR\_CLK DDRD

#define PORT\_CLK PORTD

#define PIN\_SCLK 7

#define PIN\_LCLK 4

// Values for the seven-segment (common cathode configuration).

const *uint8\_t* sevenSegmentCode[] = {

0x03,

0x9F,

0x25,

0x0D,

0x99,

0x49,

0x41,

0x1F,

0x01,

0x09

};

// 'Custom Serial Transfer Protocol' functions.

void initializePort() {

// Initializes the direction.

DDR\_SHIFT = 0x01; // Sets lower most bit as output.

DDR\_CLK = (1<<PIN\_SCLK)|(1<<PIN\_LCLK); // Sets clock pin as output.

// Initializes output.

PORT\_SHIFT = 0x00;

PORT\_CLK = 0x00;

}

void clkShift() {

// Sets clock input to shift register as high.

PORT\_CLK |= (1<<PIN\_SCLK);

//\_delay\_ms(1);

PORT\_CLK &= ~(1<<PIN\_SCLK);

}

void clkLatch() {

PORT\_CLK |= (1<<PIN\_LCLK);

//\_delay\_ms(1);

PORT\_CLK &= ~(1<<PIN\_LCLK);

}

void sendData(*uint8\_t* data) {

// Places data into the port.

PORT\_SHIFT = data;

// Repeatedly clocks the value into the shift register,

// then shifts the next value into the position.

for (*uint8\_t* i = 0; i < 8; i++) {

clkShift();

PORT\_SHIFT >>= 1;

}

}

void displayDigits(*uint8\_t* digits[]) {

// For all digits in the array:

for (*uint8\_t* i = 0; i < 4; i++) {

sendData(sevenSegmentCode[digits[i]]); // Send the seven-segment value corresponding to the digit.

sendData(1<<(4+i)); // Send the 1-hot value corresponding to which 8-segment.

clkLatch(); // Clock in the value.

}

}

// Temperature functions.

void enableADC() {

// Sets PINC4 as input.

DDRC &= ~(1<<4);

// Sets up ADC settings.

ADMUX = 0xC4; // Reads from ADC4. Right justified. Internal 1.1V reference.

ADCSRB = 0x00; // Free running mode.

ADCSRA = 0b10000111; // Enables ADC with 128 pre-scaler.

}

void disableADC() {

ADCSRA &= 0x7F; // Disables the ADEN bit.

}

*uint16\_t* analogRead() {

ADCSRA |= (1<<6); // Enables ADSC to start AD conversion.

while (!(ADCSRA&(1<<4))); // Waits until ADIF is set indicating ADC is done.

ADCSRA |= (1<<4); // Clears ADIF flag.

return ADC; // Returns value.

}

*uint16\_t* analogToTemp(*uint16\_t* t, char unit) {

// Takes analog converted number and converts it into temperature.

float convertedTemp = t\*110.0/1023.0;

switch (unit) {

case 'C':

case 'c':

return convertedTemp; // Returns temperature converted to fahrenheit.

case 'F':

case 'f':

default:

return convertedTemp\*1.8 + 32; // Returns temperature converted to celsius.

}

}

void getDigits(*uint16\_t* value, *uint8\_t* arr[]) {

for (*uint8\_t* i = 0; i < 4; i++) {

arr[i] = ( (*uint8\_t*)(value/*pow*(10, i)) ) % 10;

}

}

int main(void) {

// Variables.

*uint16\_t* analogValue, temp; // Stores the analog value and the temp.

*uint8\_t* digits[4]; // Stores the digits.

// Sets all initial values.

initializePort();

enableADC();

while (1) {

analogValue = analogRead();

temp = analogToTemp(analogValue, 'c');

getDigits(temp, digits);

displayDigits(digits);

}

}

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

Task 2 Code (Modified functions from task 1 shown):

// SPI functions.

void initializeSPI() {

// Initializes ports.

DDRD |= (1<<PIND4); // Sets latch clock pin as output.

DDRB |= (1<<PINB3)|(1<<PINB5); // Sets MOSI pin as output.

PORTD &= ~(1<<PIND4); // Sets clock output to zero.

//Sets SPI settings.

SPCR0 = 0b01110000; // Enables SPI mode as master.

}

void clkLatch() {

PORTD |= (1<<PIND4);

*\_delay\_ms*(1);

PORTD &= ~(1<<PIND4);

}

void sendData(*uint8\_t* data) {

SPDR0 = data; // Sends data.

while ( !(SPSR0&(1<<SPIF)) ); // Waits until the data is sent.

SPSR0 |= (1<<SPIF); // Clears the flag.

//SPCR0 |= ();

}

void displayDigits(*uint8\_t* digits[]) {

// For all digits in the array:

for (*uint8\_t* i = 0; i < 4; i++) {

sendData(sevenSegmentCode[digits[i]]); // Send the seven-segment value corresponding to the digit.

sendData(1<<(4+i)); // Send the 1-hot value corresponding to which 8-segment.

clkLatch(); // Clock in the value.

}

}

Task 3 Code (Library not shown):

#define *F\_CPU* 16000000UL

#include <avr/interrupt.h>

#include <avr/io.h>

#include <util/delay.h>

#include "ds18b20.c"

#define UBBR\_VALUE 103

char myIntString[20];

// USART functions.

// Enables the UART and it's transmitter.

void enableTransmit() {

// Sets port.

DDRD |= 0x02;

// Sets baud rate.

UBRR0 = UBBR\_VALUE;

// Sets UART settings.

UCSR0C = 0b00000110; // to async mode with no parity

// 2 stop bits, and a frame of 8 bits.

UCSR0B = 0b00001000; // Enables transmit line.

}

// Disables the transmitter on UART.

void disableTransmit() {

UCSR0B &= ~(1<<TXEN0);

}

// Sends one byte of data out the TX line.

void sendByte(*uint8\_t* b) {

if (UCSR0B & (1<<TXEN0)) { // Checks whether transmit is enabled.

while (!(UCSR0A & (1<<UDRE0))); // Waits until the UDR0 register is ready.

UDR0 = b; // Sends the byte.

}

}

void sendString(char string[]) {

int i = 0;

while (string[i] != '\0')

sendByte(string[i++]);

}

int main(void) {

// Variables.

*uint16\_t* temp; // Stores the analog value and the temp.

// Sets all initial values.

enableTransmit();

sei();

while (1) {

temp = ds18b20\_gettemp();

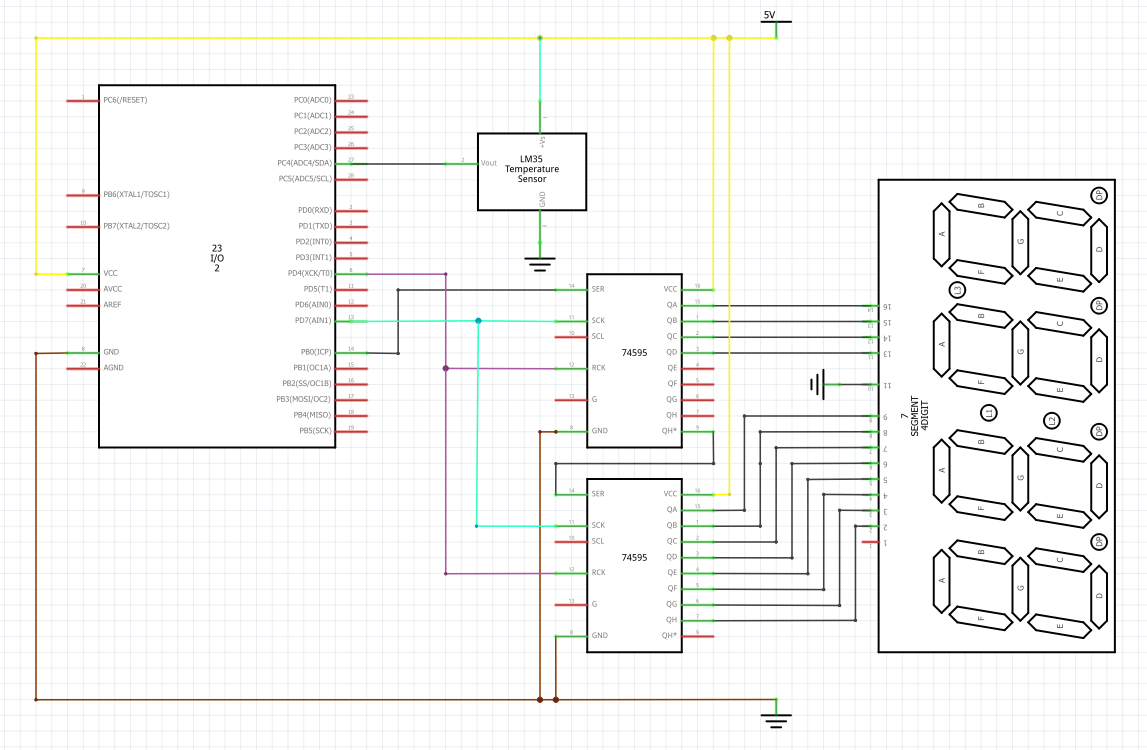
*snprintf*(myIntString, 20, "%u\n", temp);

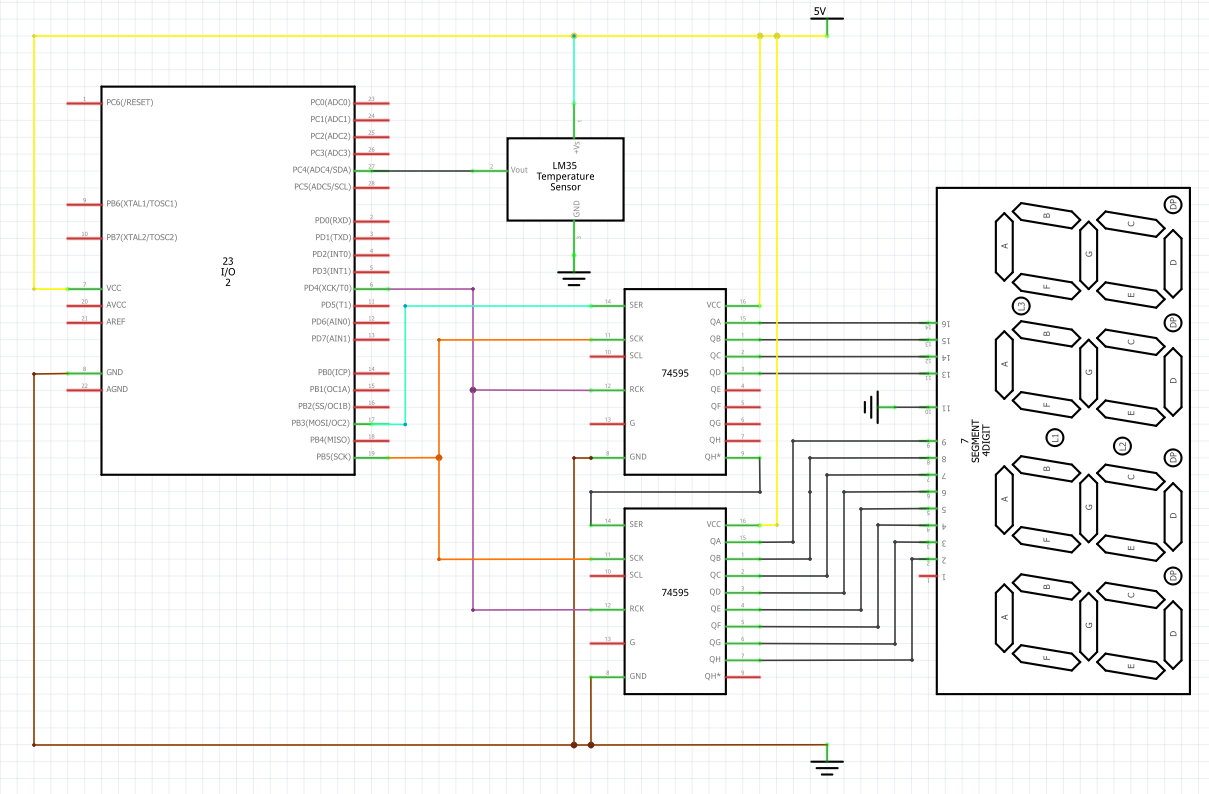
sendString(myIntString);

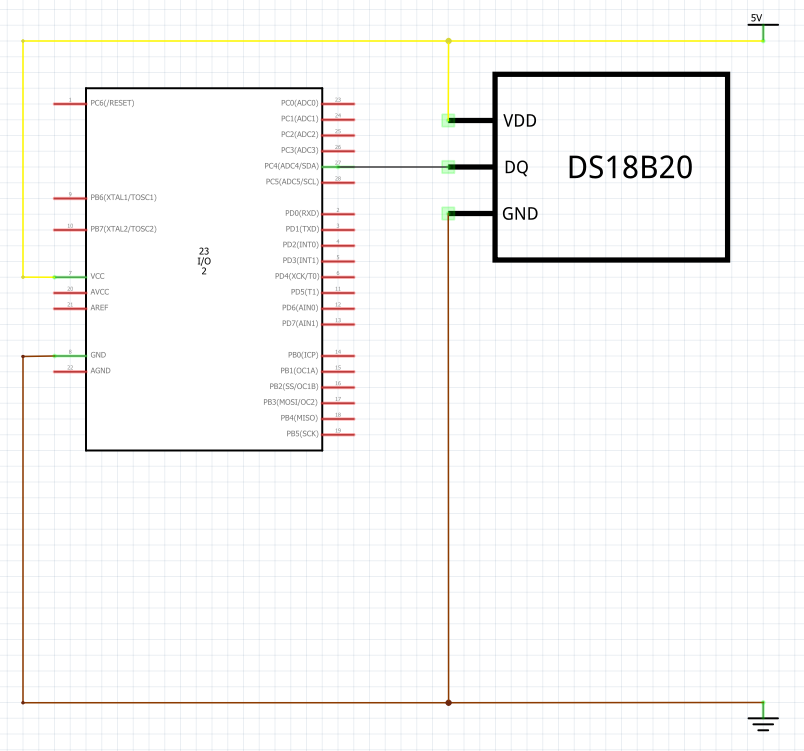
*\_delay\_ms*(500);

}

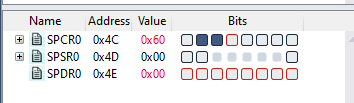
}

1. **SCHEMATICS**

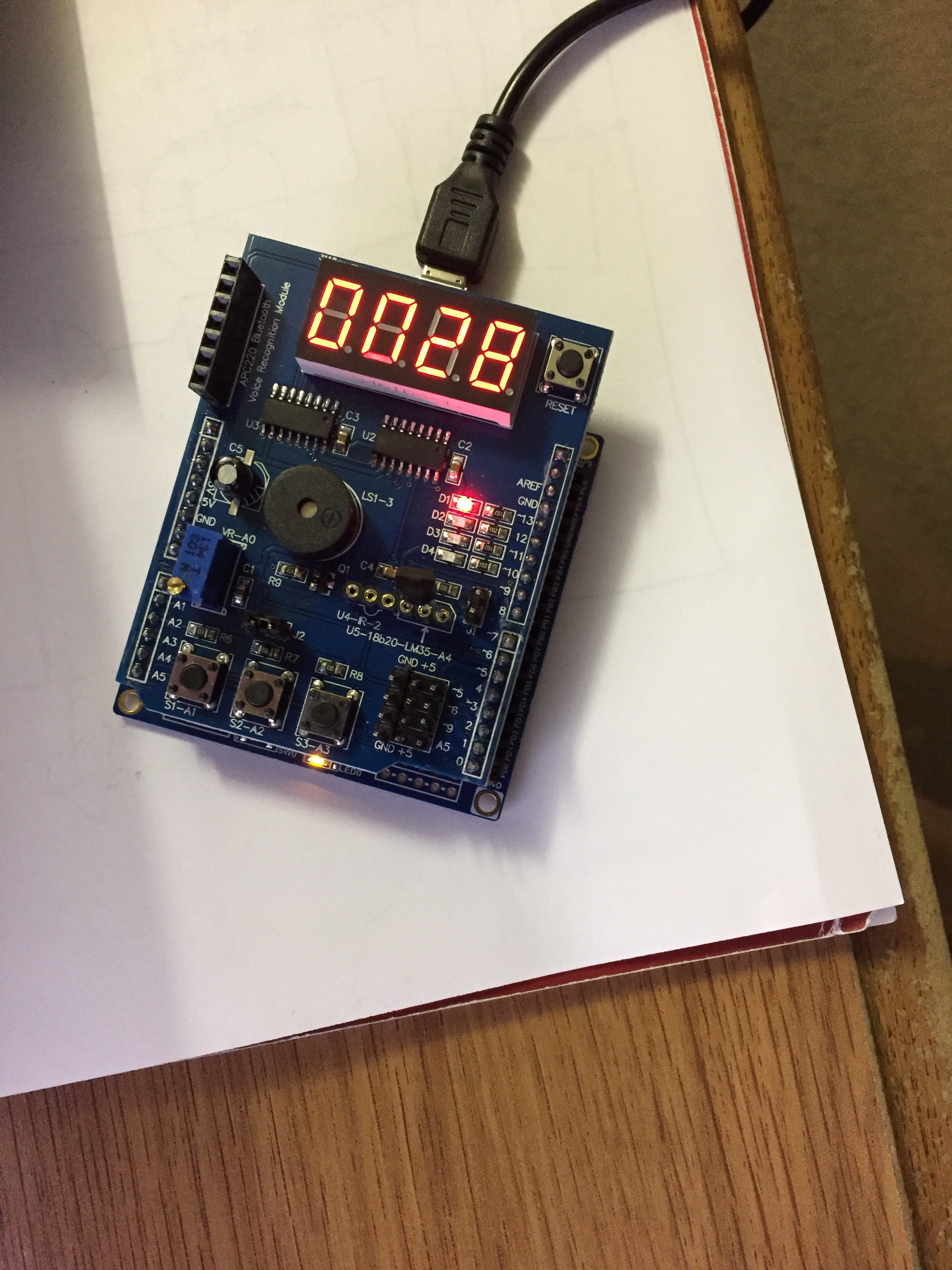


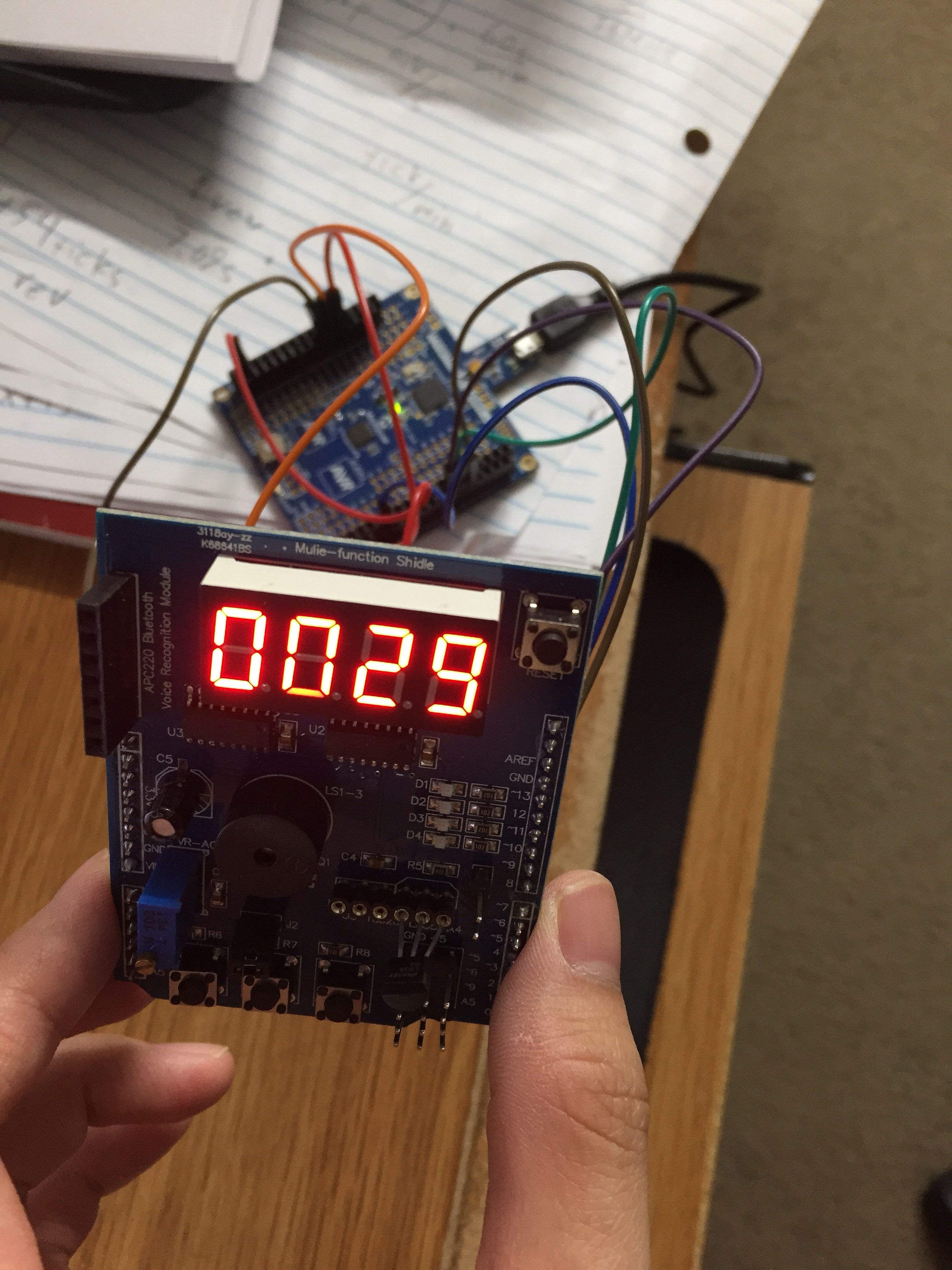


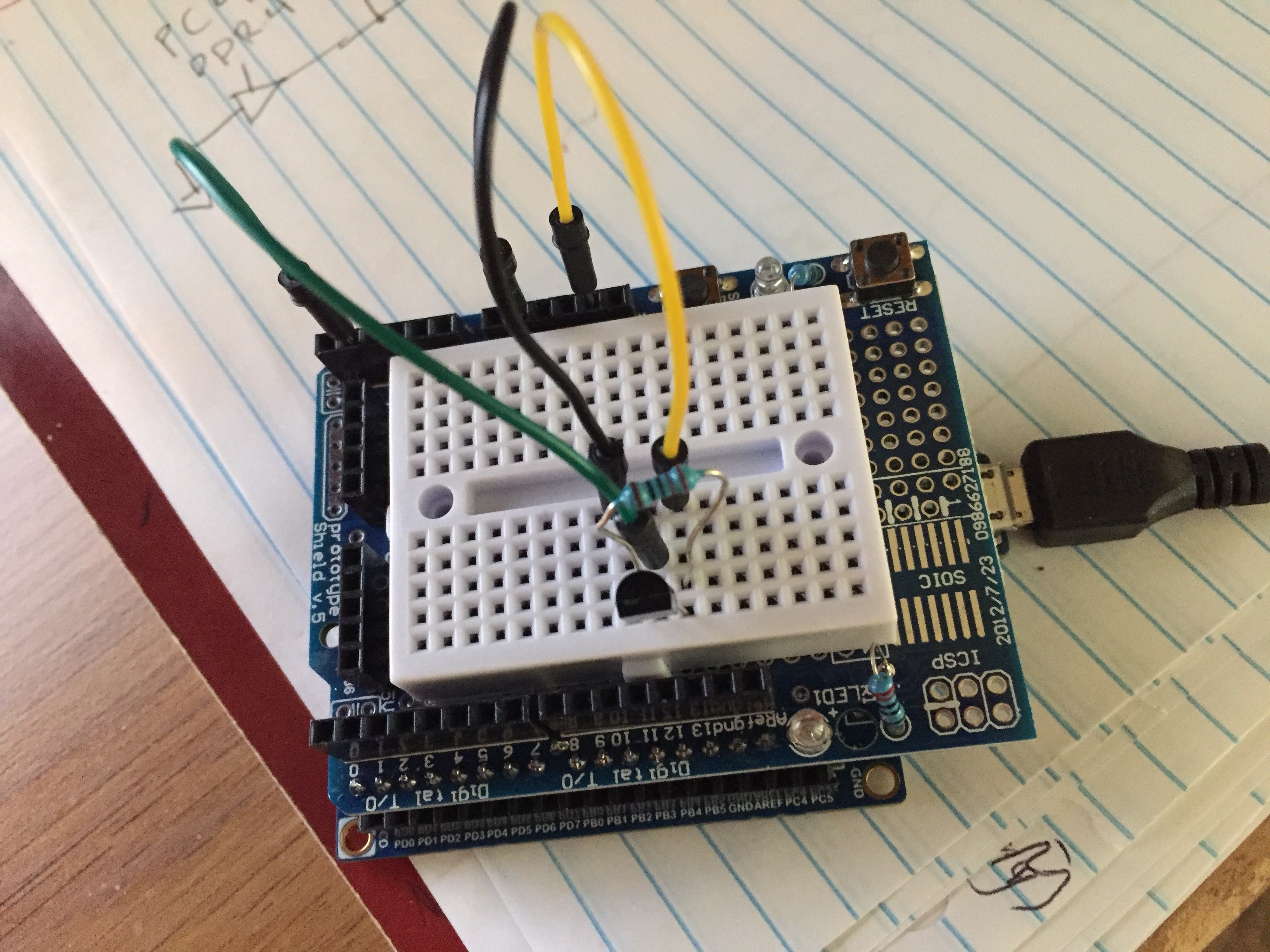
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**



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







1. **VIDEO LINKS OF EACH DEMO**

Task 1 Video

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

Task 2 Video

https://www.youtube.com/watch?v=TiWH5GA68fM

Task 3 Video

https://www.youtube.com/watch?v=w1iWFvSaYKU

1. **GITHUB LINK OF THIS DA**

https://github.com/DoVietLe/assignments/tree/master/ESD301/LAB05

**Student Academic Misconduct Policy**

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

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

Do Viet Le