ECE 4500 Laboratory 3 Prelab

Instructor: Dr. Ashraf Ali Khan

Members: Aaron Oates
Sam Martino

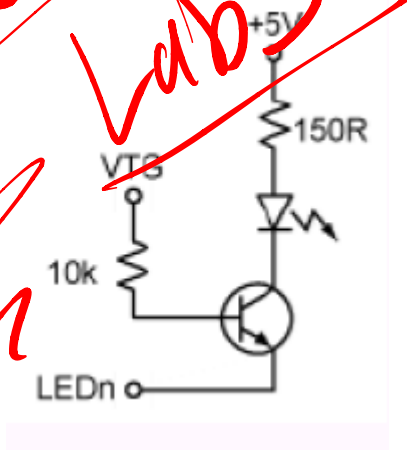
Group #: 18

The objective of the pre-lab is to understand how to use input and output ports in AVR ATMEGA 32. First, carefully understand the notes provided then answer the questions. If you do not complete the pre-lab, you will lose marks and will face difficulties in your lab tasks.

Understanding the Built-in LEDs in STK600 Board

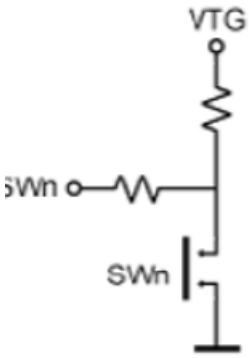
- The LEDs are labelled LED0 to LED7. The corresponding pins on the LEDs header have the same labels.
- The transistor circuit ensures the LED brightness is independent of the target voltage.
- To light one of the LEDs, the corresponding pin found on the LEDs header must be pulled to GND.
- To control the LEDs from the AVR, connect a cable between the LEDs header and one of the PORT headers. Use a 10-wire cable to connect to all eight LEDs.

Demo Done in Lab3 w. Sudhir



Understanding the Built-in Switches in STK600 Board

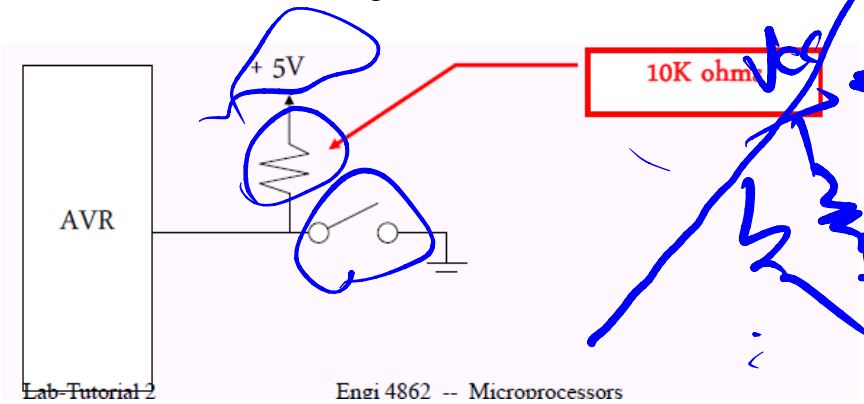
- The switches are labelled SW0 to SW7, and are available on the SWITCHES header.
- When pressing one of the switches, the corresponding SW pin on the SWITCHES header will be pulled low. When the switch is released, the switch's 10k pull-up will pull the line to VTG.
- Connect a cable between the SWITCHES header and one of the PORT headers. Use a 10-wire cable to connect to all eight switches.



Note: On most AVR pins configured as input, you can enable an internal pull-up, removing the need for an external pull-up on the push button. In the STK600 design, an external 10K pull-up is present to give all users a logical '1' on SWn when the push button is not pressed, even if the internal pull-up is not enabled.

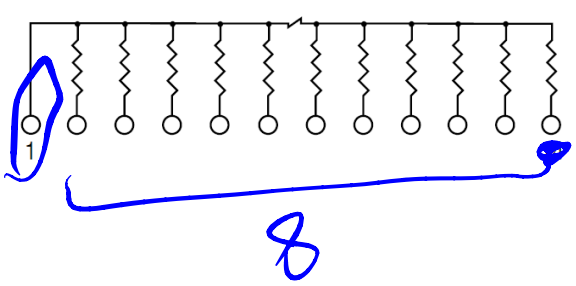
Understanding how to Set up External Switches

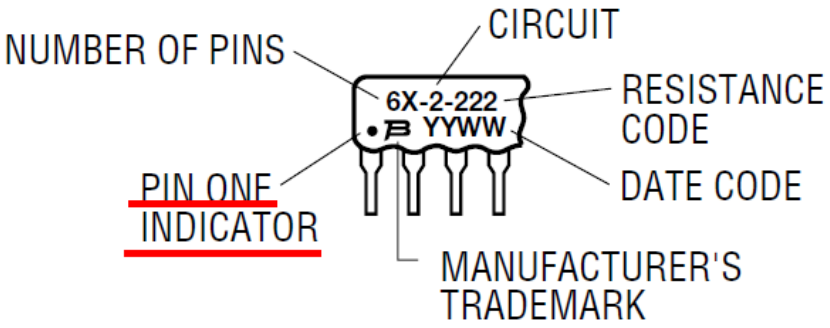
- How do we set up external switches?
  - When open, one logic level
  - When closed, the other logic level



Bussed Resistors

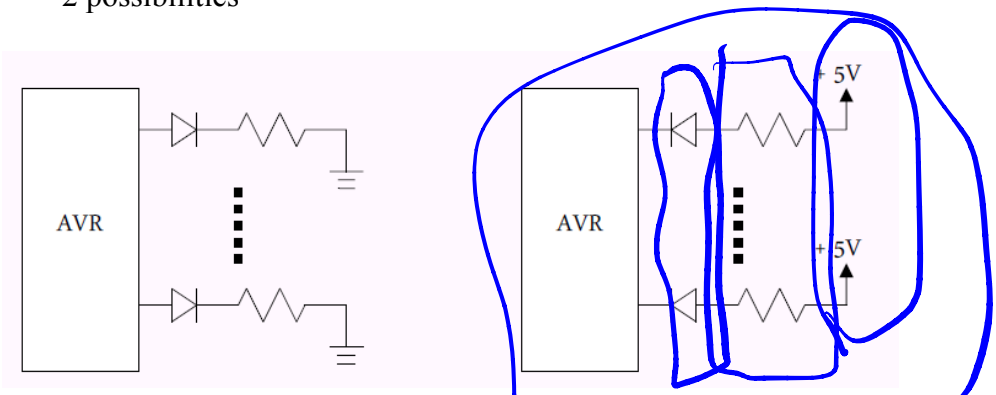
There are commonly two ways the resistors are connected in the array, isolated and bussed where one end of all the resistors share a common connection as shown in figure 8. The bussed arrangement is **commonly used for pull up, pull down or bus termination on logic signals**





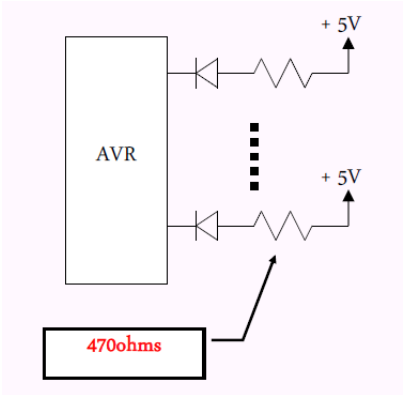
Understanding How to Set up External LEDs

- How do we connect the LEDs?
  - 2 possibilities

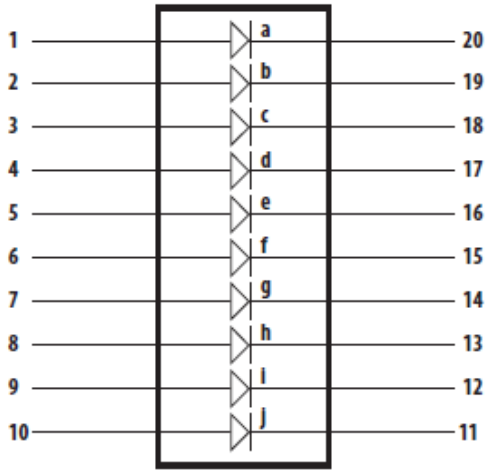


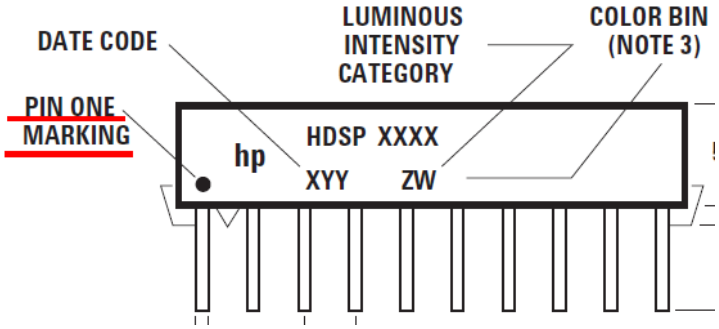
(Active high) (Active low)

- In active low, LED illuminates for logic 0.
- In active high, LED illuminates for logic 1.
- The AVR port may not have enough power to drive an LED, therefore active low is preferred as shown below.
- The device can sink enough current for the LED to light (15 to 20 mA)
- You will use bussed resistors to set up external LEDs.



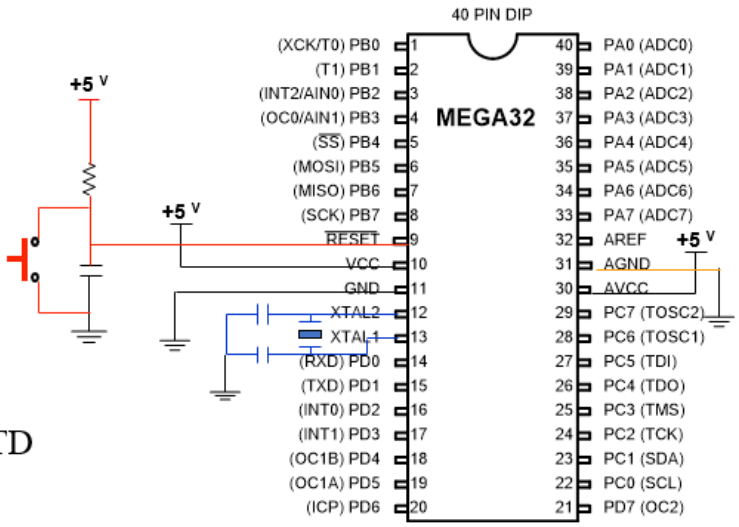
You will use an array of external LEDs as shown below.





Understanding the Pinout of the AVR ATMEGA 32

1. Vital Pins:
   1. Power
      - VCC
      - Ground
   2. Crystal
      - XTAL1
      - XTAL2
   3. Reset
2. I/O pins
   - PORTA, PORTB, PORTC, and PORTD
3. Internal ADC pins
   - AREF, AGND, AVCC

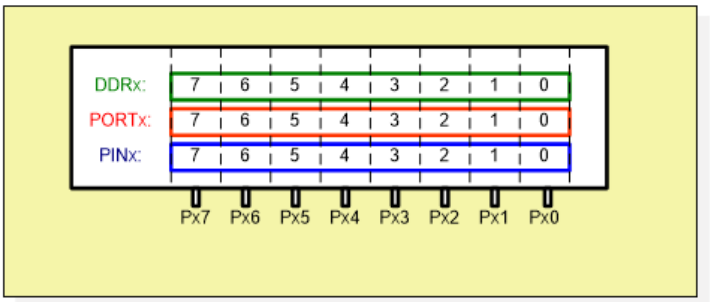


Watch pinout tutorial and notes

Understanding Registers to Make a Port Input or Output

Each port has three registers associated in special function registers block.

Port	Address	Usage
PORTA	$3B	output
DDRA	$3A	direction
PINA	$39	input
PORTB	$38	output
DDRB	$37	direction
PINB	$36	input
PORTC	$35	output
DDRC	$34	direction
PINC	$33	input
PORTD	$32	output
DDRD	$31	direction
PIND	$30	input



- **PORT Register:** It is used to write anything to physical pin of microcontroller
- **DDR Register:** It is called data direction register used to set a port as input or output. As the name suggests it gives the direction to data. 0 in DDR register makes a port as input and 1 makes a port as output.
- **PIN Register:** It is used to transfer data from input pins.

- The number of ports in the AVR family varies depending on the number of the pins on the chip.

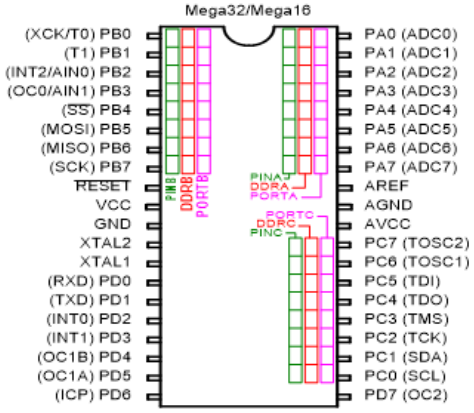
- Each of the ports A-D in ATmega32 can be used for input or output. To use any of these ports as an input or output port, it must be programmed.
- Each port has three I/O registers associated with it: DDRx, PORTx, PINx. Each bit of the I/O registers affects one of the pins.
- DDRx is to make a given port as an input or output port. 1 for output and 0 for input.

Examples

- Write a program that makes all the pins of PORTA one.

DDRA:	1	1	1	1	1	1	1	1
PORTA:	1	1	1	1	1	1	1	1

```
.INCLUDE "M32DEF.INC"
LDI  R20,0xFF  ;R20 = 11111111 (binary)
OUT  DDRA,R20  ;DDRA = R20
OUT  PORTA,R20  ;PORTA = R20
```



- Write a program that gets the data present at the pins of port C and sends it to port B indefinitely, after adding the value 5 to it:

```
        .INCLUDE "M32DEF.INC"
        LDI     R16, 0x00       ; R16 = 00000000 (binary)
        OUT     DDRC, R16       ; make Port C an input port
        LDI     R16, 0xFF       ; R16 = 11111111 (binary)
        OUT     PORTC R16       ; To activate Internal Pull up
        OUT     DDRB, R16       ; make Port B an output port(1 for Out)
L2:     IN      R16, PINC       ; read data from Port C and put in R16
        LDI     R17, 5
        ADD     R16, R17        ; add 5 to it
        OUT     PORTB, R16      ; send it to Port B
        RJMP    L2              ; continue forever
```

After reading the above section answer the following questions.

Question 1: Mention the names of three registers associated with each port.

DDR, PORT, PIN

Question 2: Describe the duty of each register in the previous question.

PIND 350 input

- **PORT Register:** It is used to write anything to physical pin of microcontroller
- **DDR Register:** It is called data direction register used to set a port as input or output. As the name suggests it gives the direction to data. 0 in DDR register makes a port as input and 1 makes a port as output.
- **PIN Register:** It is used to transfer data from input pins.

Vasad Prelab 3

Question 3: How to configure an input port?

```
OUT     DDRC, R16        ; make Port C an input port
```

Question 4: How to configure an output port?

```
OUT     DDRB, R16        ; make Port B an output port(1 for Out)
```

Question 5: What is the difference between push-button and slide-style switches?

- The switches are labelled SW0 to SW7, and are available on the SWITCHES header.
- When pressing one of the switches, the corresponding SW pin on the SWITCHES header will be pulled low. When the switch is released, the switch's 10k pull-up will pull the line to VTG.
- Connect a cable between the SWITCHES header and one of the PORT headers. Use a 10-wire cable to connect to all eight switches.

Question 6: How does the unsigned multiply instruction of ATmega32 work? Refer to the instruction manual.

MUL Rd, Rr; R1:R0 ← Rd x Rr (UU)

Question 7: Write an assembly language program to implement an 8-bit unsigned number multiplier (watch uploaded prelab 3 video).

```
.include "M32DEF.INC" ;

.org 0x0000 ;

RESET:
    LDI R16, 0x12 ;
    LDI R17, 0x34 ;

    MUL R16, R17

    MOV R18, R0

    MOV R19, R1

LOOP:
    rjmp LOOP
```

Aaron Oates, Sam Martino ECE 4300 Lab 3

Question 1: DDRA stores the input byte of the slide switches, DDRB outputs the byte from port A to the LEDs, DDRC stores two 8-bit numbers (two different inputs from A) and the arithmetic operation and then DDRD outputs the numbers from C and then their product once SW7 is pressed.

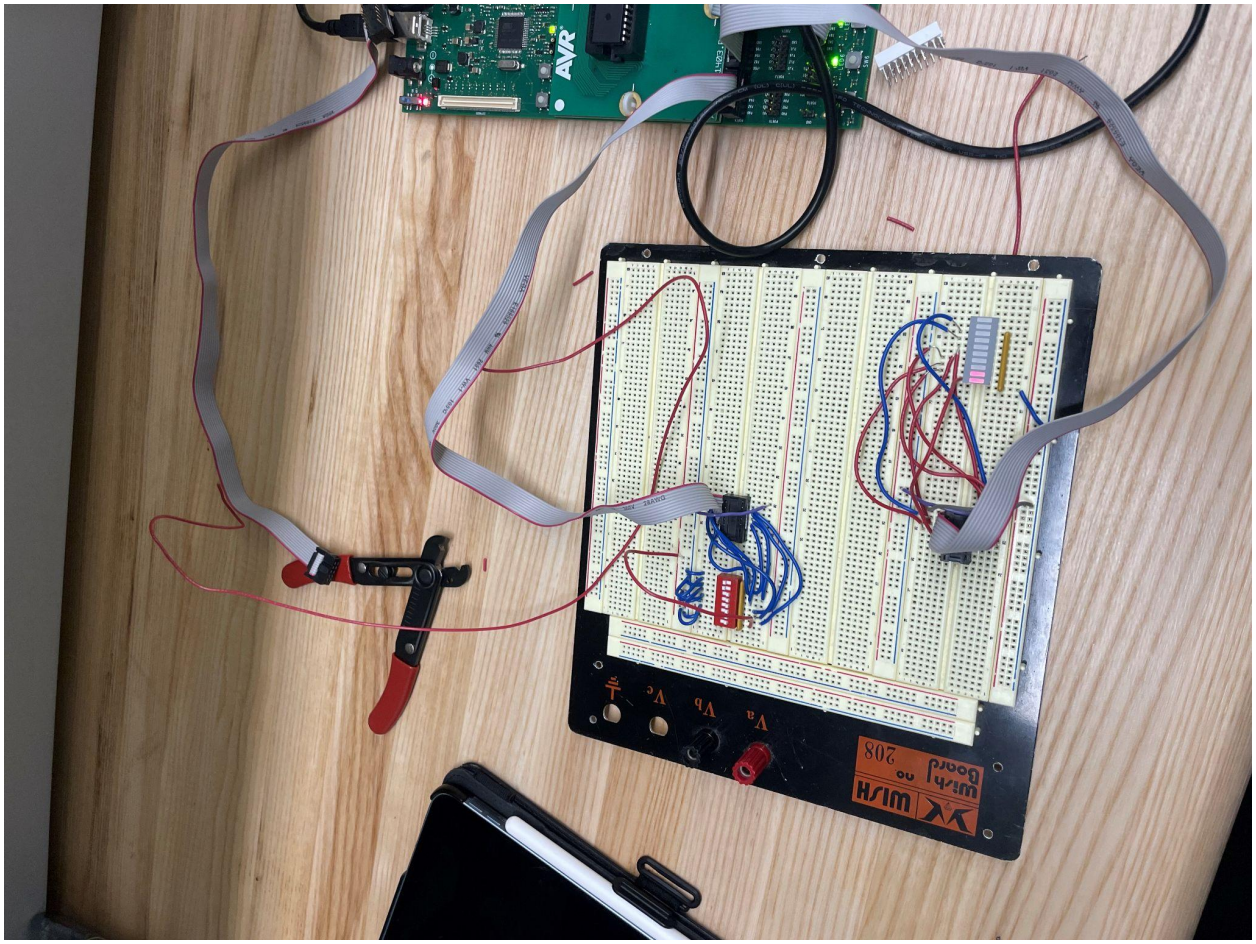
Question 2: R16(low) and R17(high).

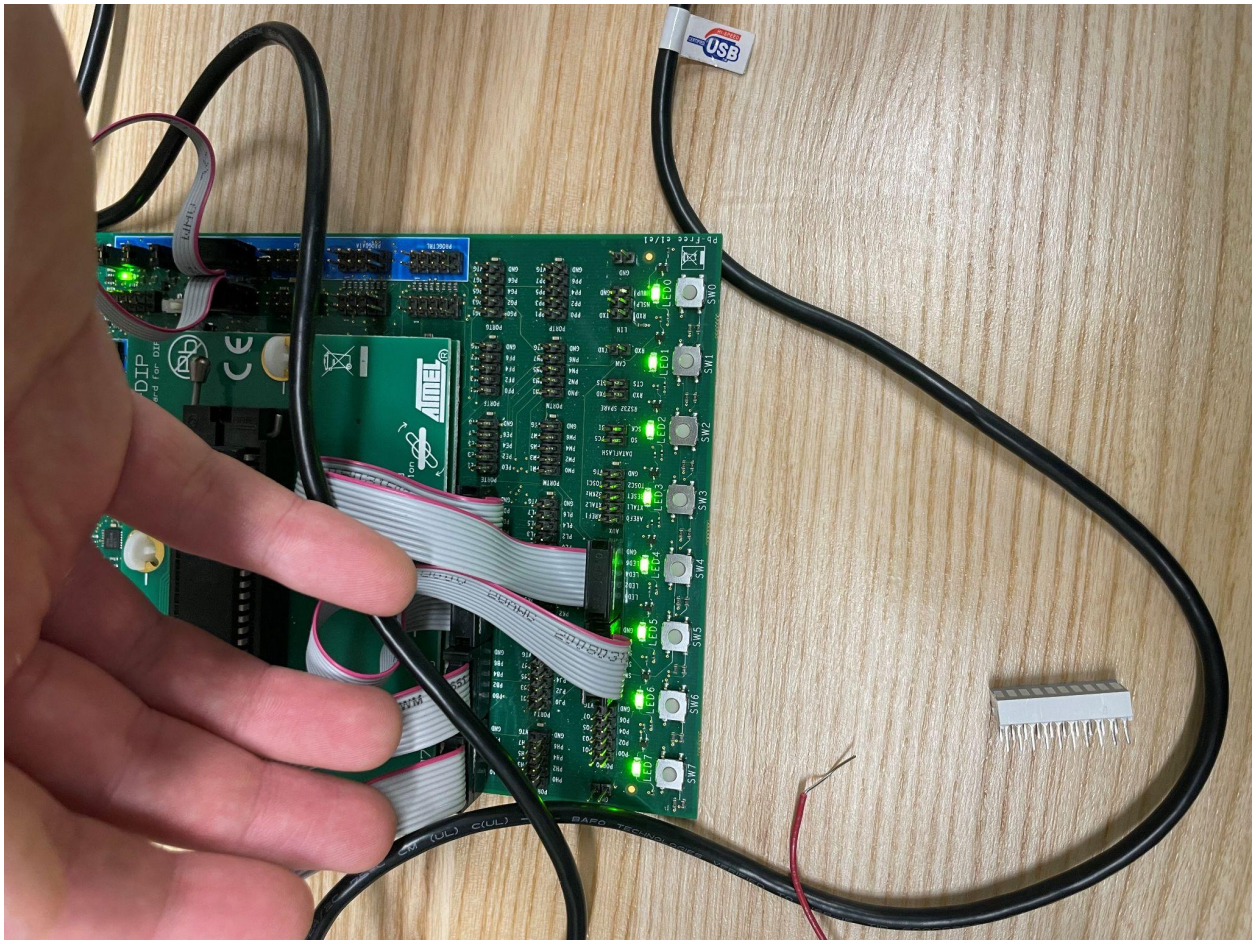
Question 3:

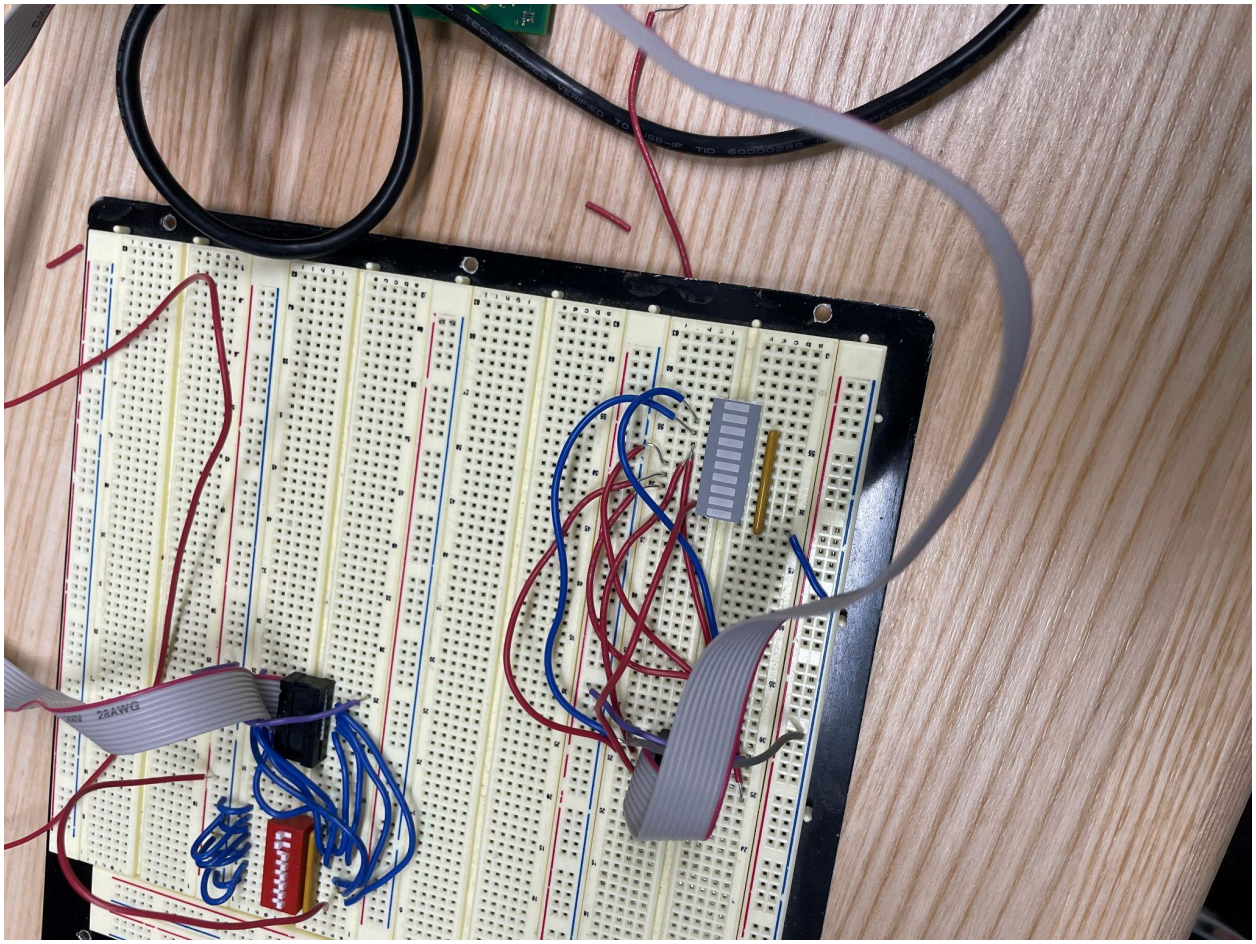
```
.INCLUDE "M32DEF.INC"
        SER R16 // Set all the bits of Register R16 to be all ones
        OUT DDRB, R16 //Set PORTB as  output
        OUT DDRD, R16 //Set PORTD as  output

        CLR R17 //Set all the bits of Register R17 to be 0
        OUT DDRA, R17 //Set PORTA as  input
        OUT DDRC, R17 //Set PORTC as  input

K1: //Labelling instruction code
        SBIC PINC,0 //condition to check if SW0 IS PRESSED, IF PRESSED THEN THIS PIN
IS CLEARED (GND), then SKIP NEXT INSTRUCTION
        JMP K1
        IN R20, PINA //copy the value from input register PINA and store it in R20
        OUT PORTB, R20 //copy the value stored in R20 and output it to PORTB
K2: //Labelling instruction code
        SBIC PINC,1 //condition to check if SW7 IS PRESSED, IF PRESSED THEN THIS PIN
IS CLEARED (GND), then SKIP NEXT INSTRUCTION
        JMP K2
        IN R21, PINA //copy the value from input register PINA and store it in R21
        OUT PORTD, R21 //copy the value stored in R21 and output it to PORTD
M1: //Labelling instruction code
        SBIC PINC,6 //condition to check if SW3 IS PRESSED, IF PRESSED THEN THIS PIN
IS CLEARED (GND), then SKIP NEXT INSTRUCTION
        JMP M1
        MUL R20,R21 //Multiplication of values in Registers R20 and R21
        OUT PORTB, R0 //Output the lower byte of the multiplication result to PORTB
        OUT PORTD, R1 //Output the higher byte of the multiplication result to PORTD
        JMP K1 //loop back
```

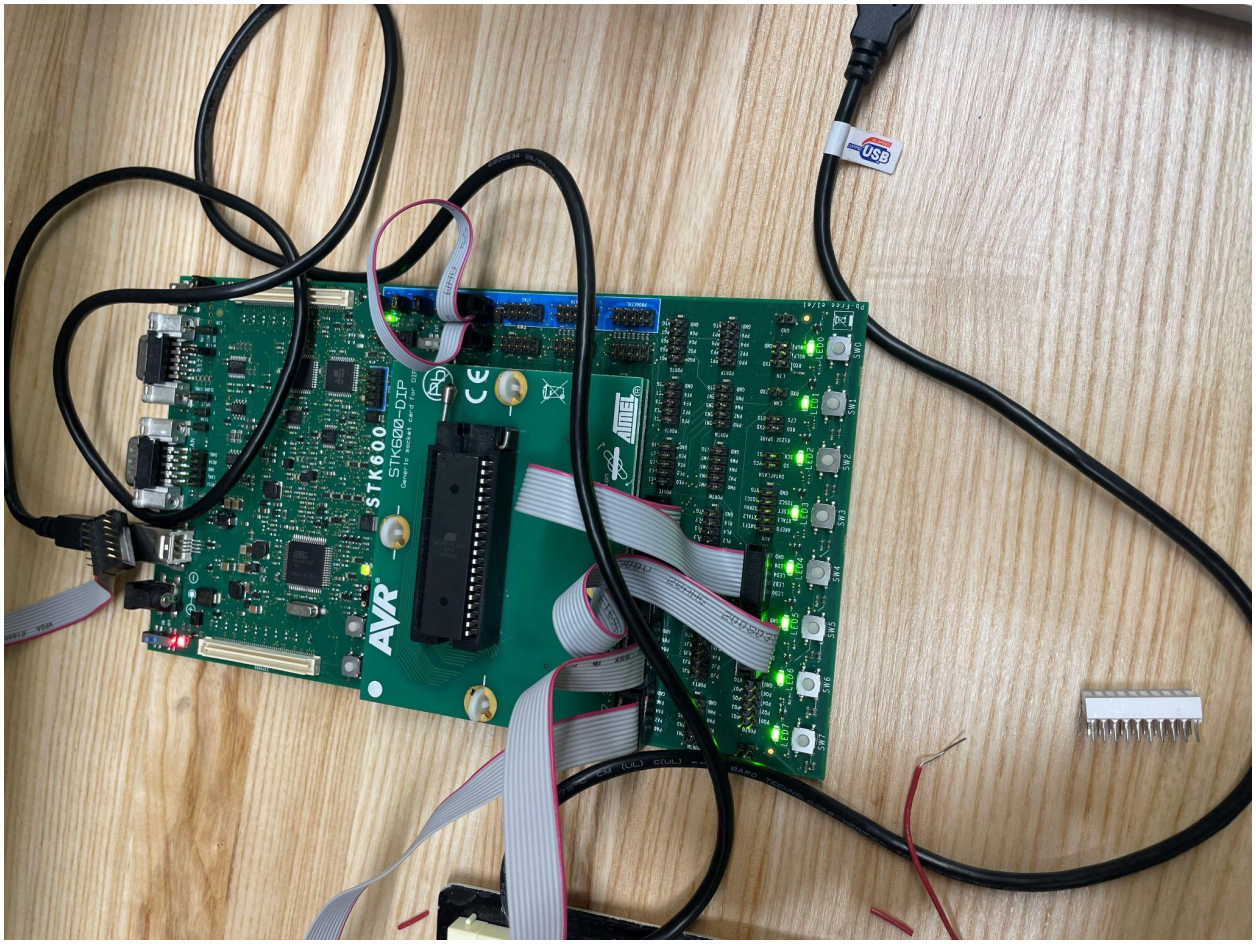
Question 4: Overall this lab was a success. The ATmega32 was able to store and output the correct mathematical operations and results. In the debug process, none of our LEDS were outputting logic 1, so to solve this problem we tried building the code from Lab 1 to make sure that it was a problem with the ATmega32 and not the code. The code worked, meaning that it

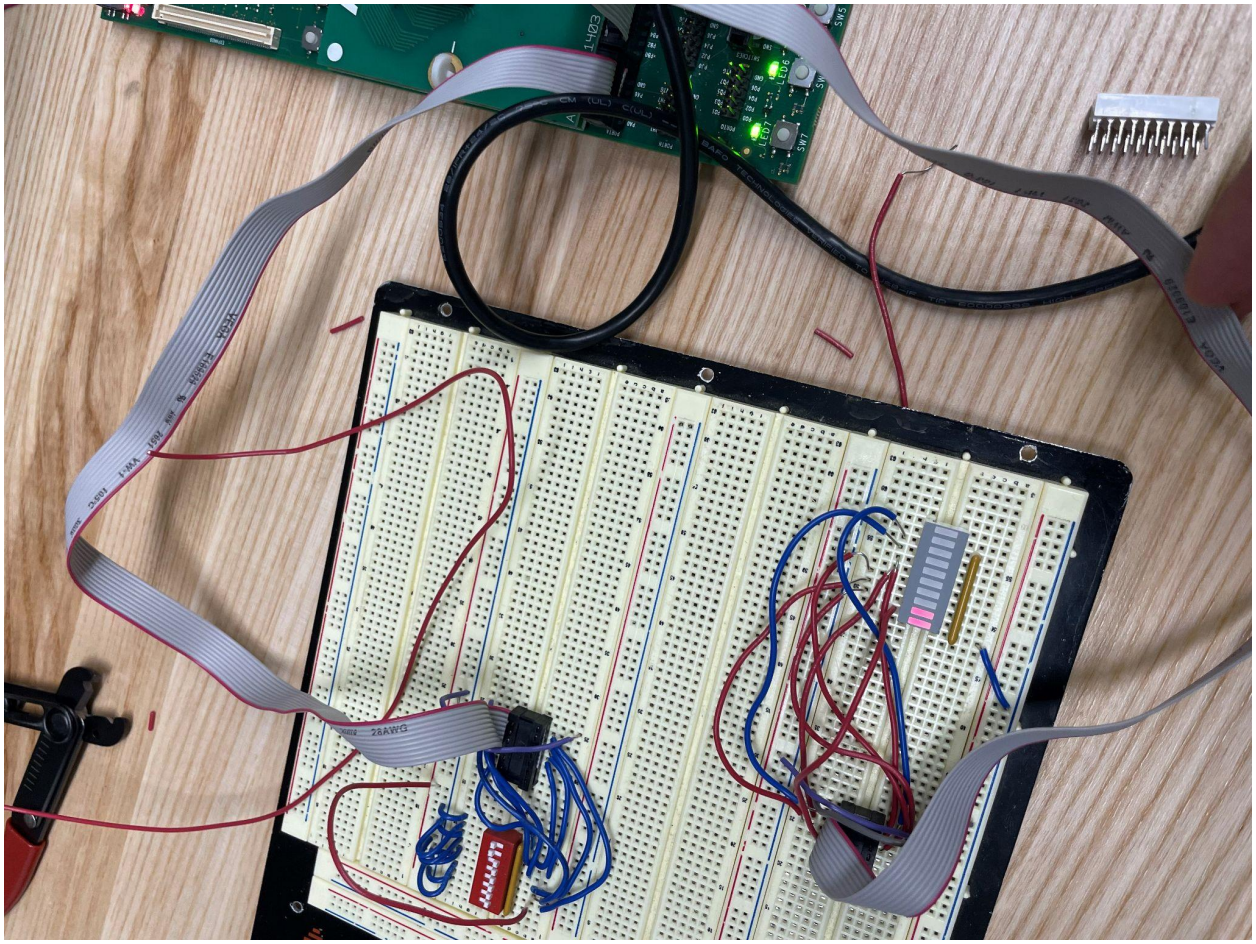
was a problem with our circuit setup. After much time looking at the circuit, we discovered that there was a single floating pin at the LED which was the root of all our problems. Once we replaced the LED, the circuit board gave us the proper results and we were able to complete the lab demo.

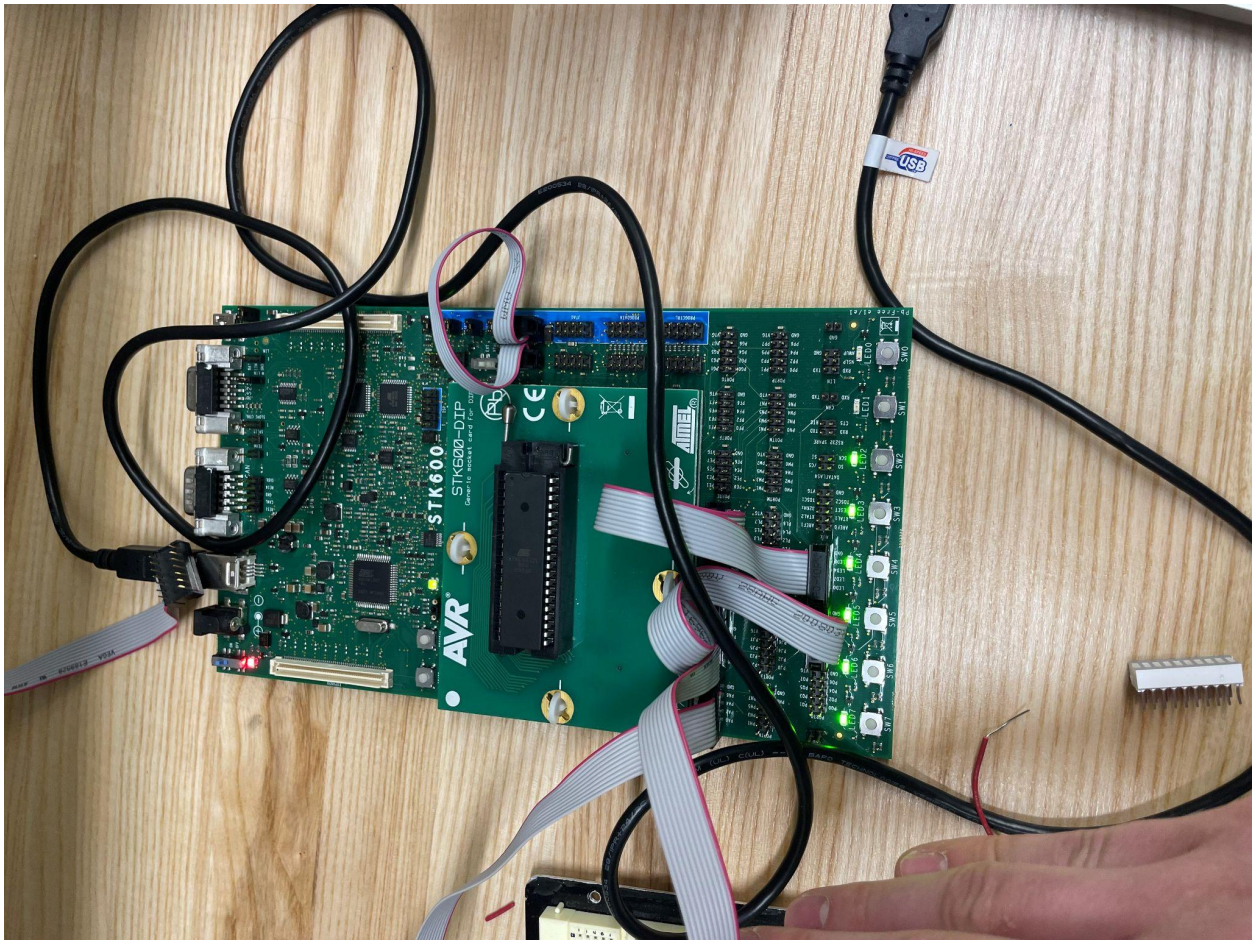

AVR
Va
Vb
Vc
WISH
Wish Board no 208

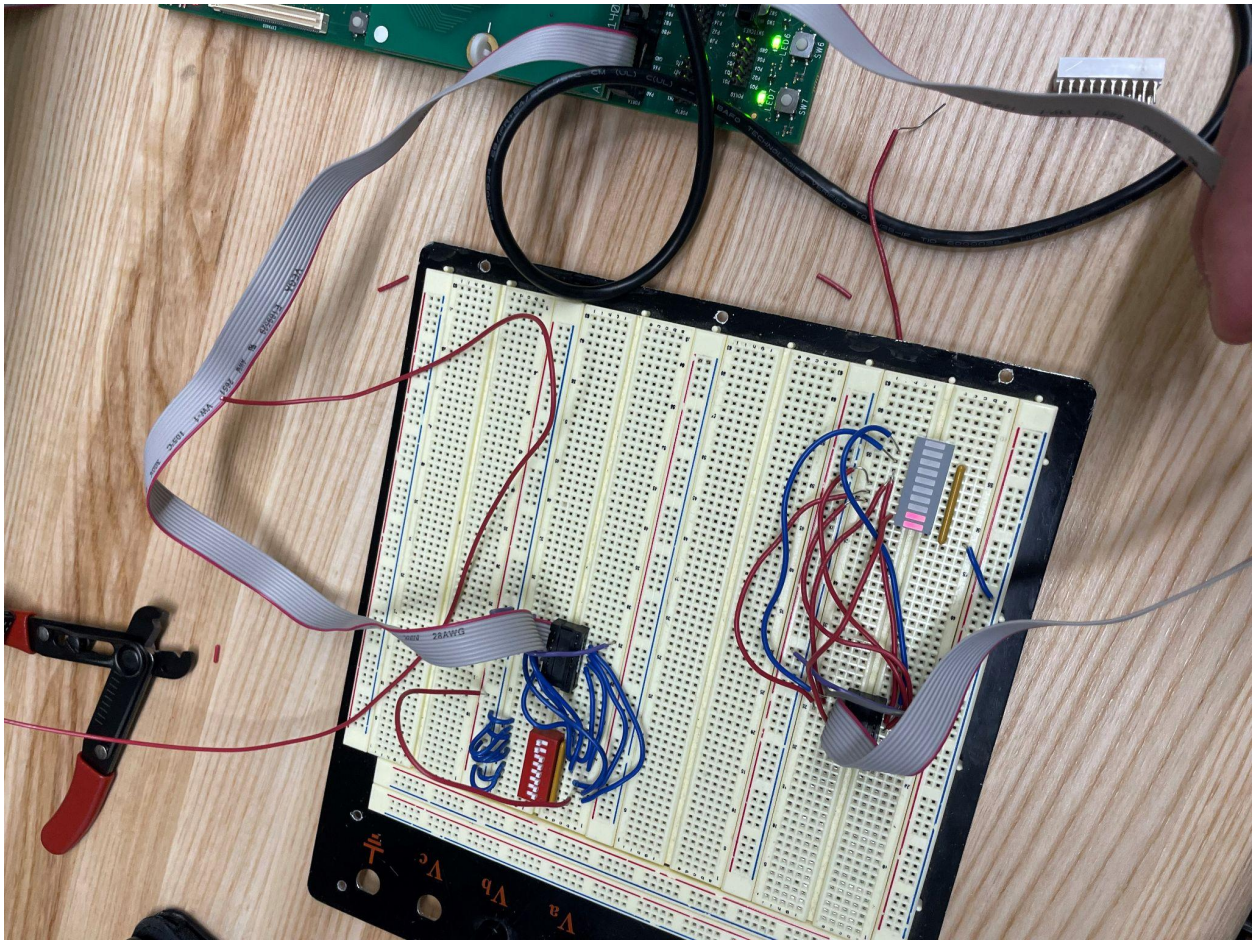

USB
LED0
SW0
LED1
SW1
LED2
SW2
LED3
SW3
LED4
SW4
LED5
SW5
LED6
SW6
LED7
SW7

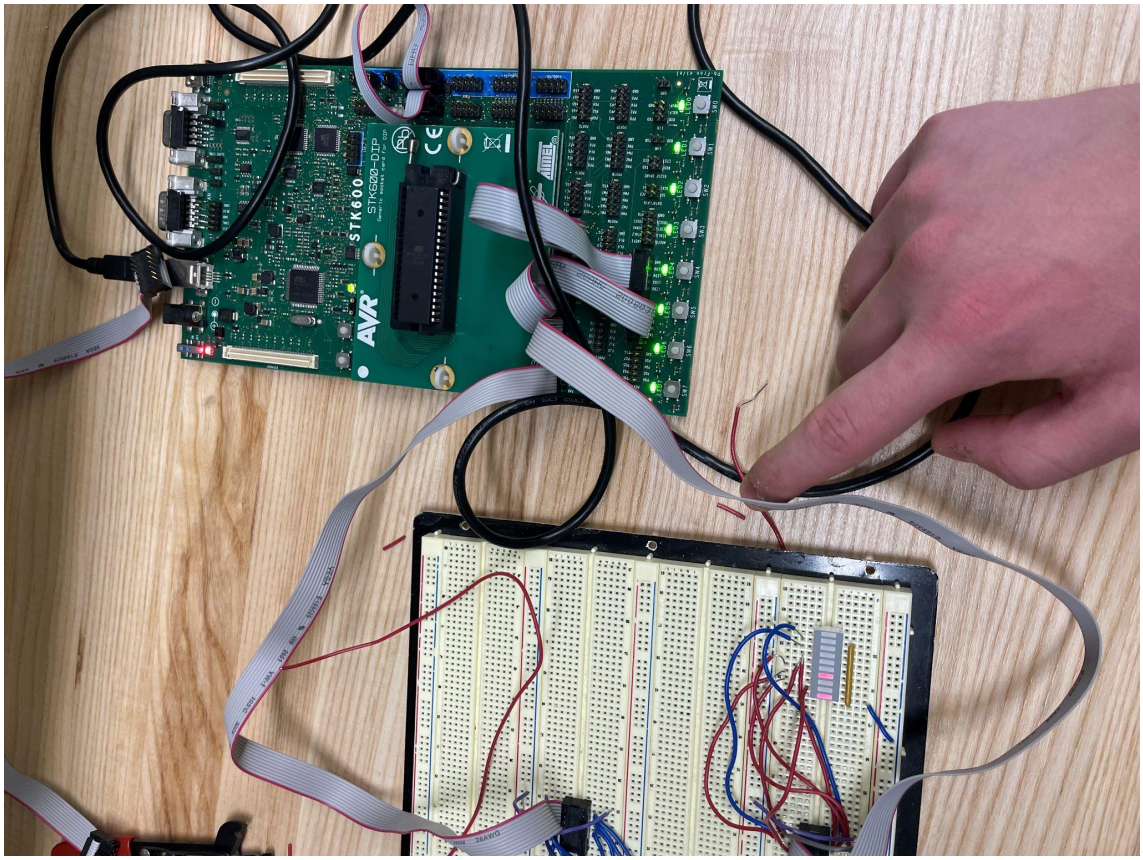



STK600
STK600-DIP
AVR
ATMEL
USB


LED7
SW7
LED6
SW6


STK600
STK600-DIP
AVR
ATMEL
USB


Va
Vb
Vc


STK600
STK600-DIP
AVR