b) Prelab Questions

None.

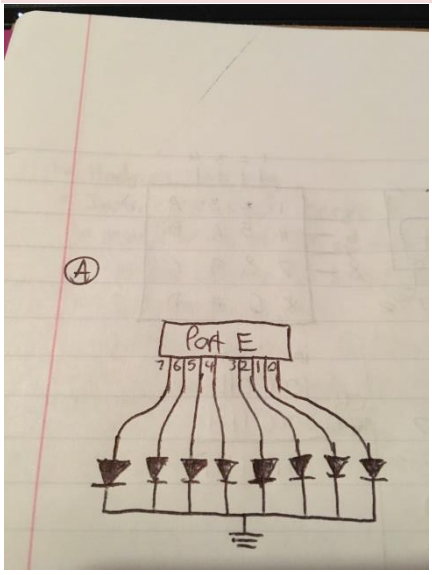
c) Problems Encountered

I lost some hours understanding the pin configuration of the keypad, and then more figuring out how to properly handle waiting until the key is released.

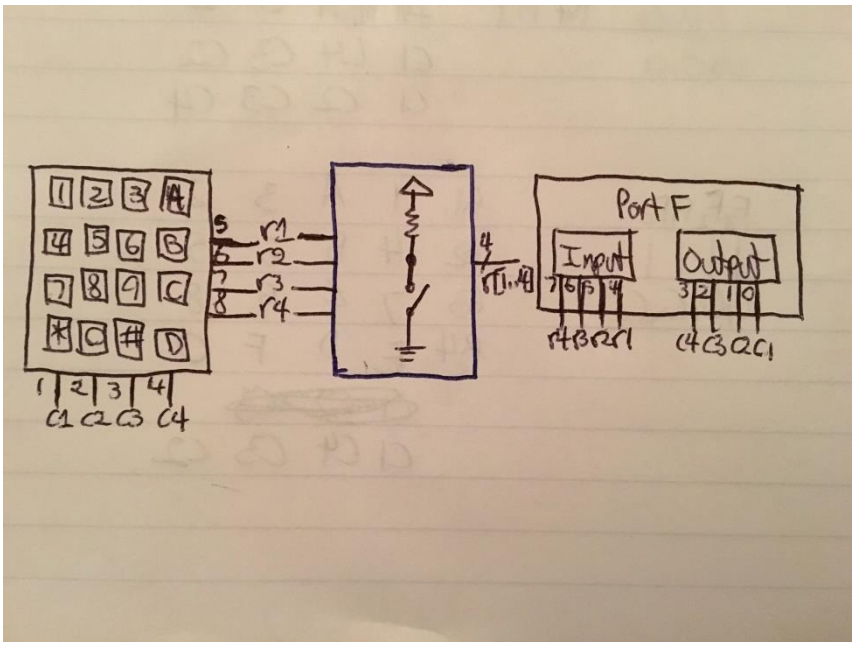
d) Future Work/Applications

The most profound application from this lab is the use of the keypad as an input device. Since we have full control over how the input is handled, we can configure it as an interface to any program we can support on the processor.

e) Schematics



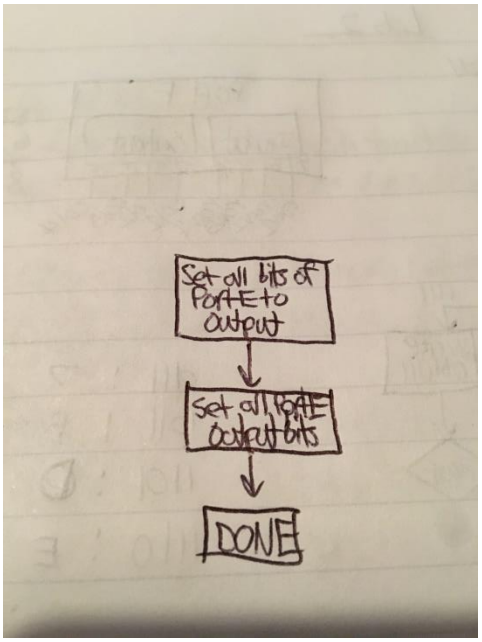
Port E connection to LED schematic



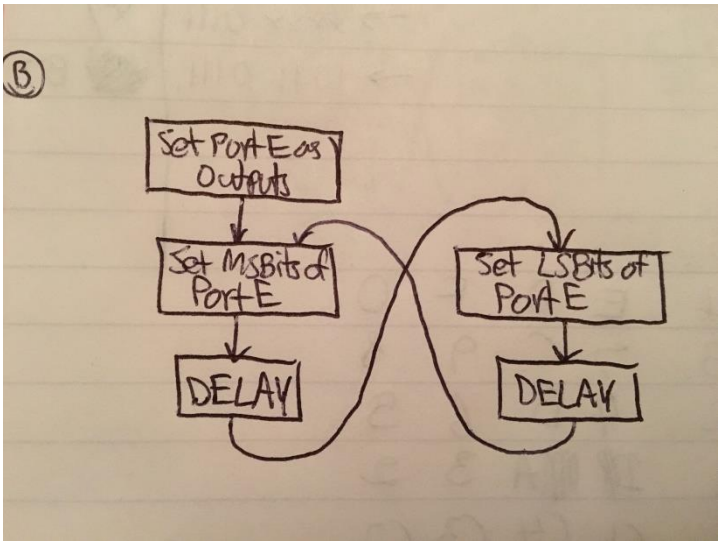
Keypad -> Pull-up Resistors -> Port F

f) Decoding Logic

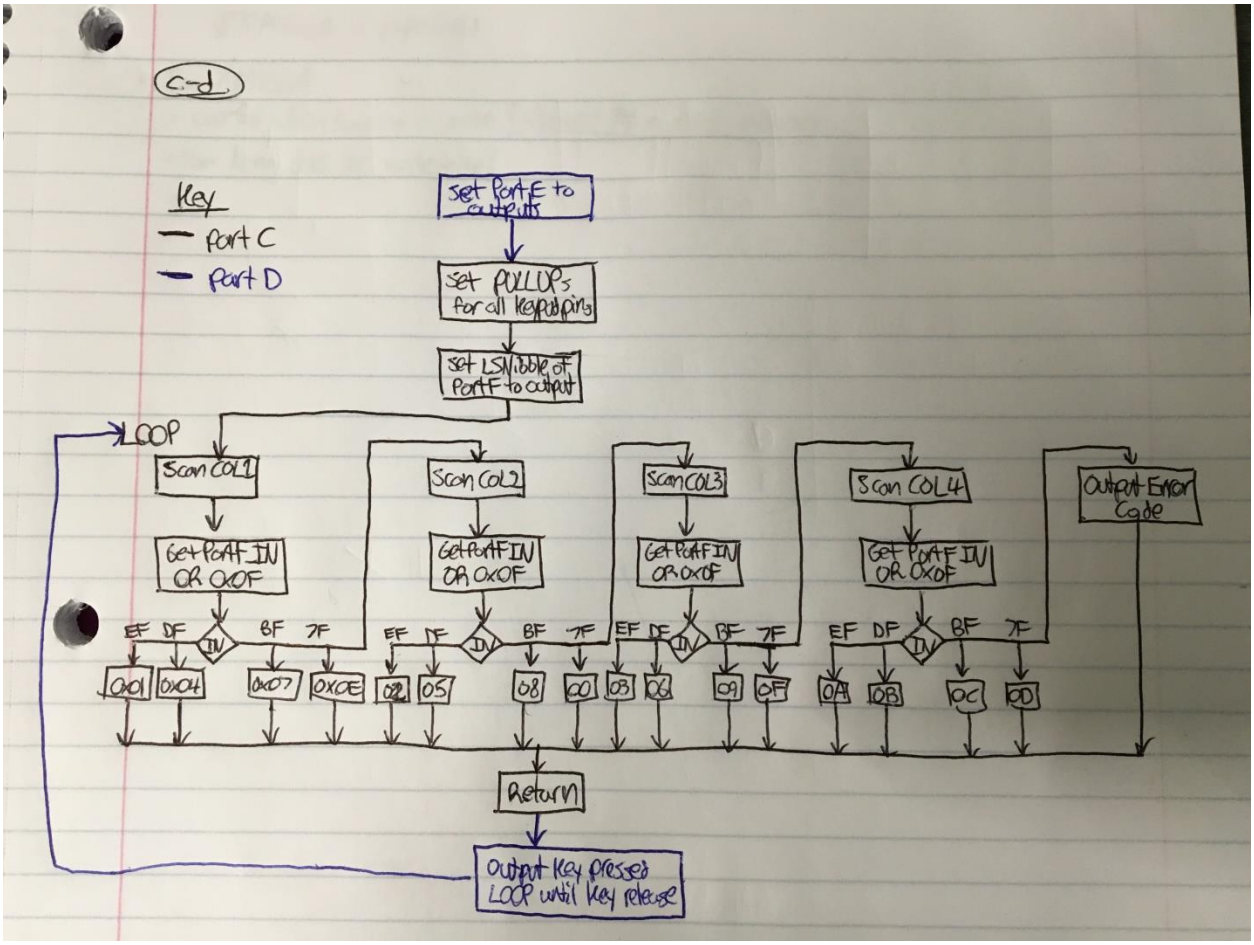
g) Pseudocode/Flowcharts



Part (a) LED test program flowchart



Part (b) Alternating LEDs flowchart



Part (c-d) Keypad interface flowchart and wrapping subroutine call

h) Program Code

Part A

```
/*
 * lab2a.asm
 *
 * Lab 2 Part A
 * Name: Nicholas Imamshah
 * Section: 6957
 * TA Name: Daniel Gonzalez
 * Description: The purpose of this program is to test the LED circuits of the uPAD.
 */

.nolist ; This works, but the below file can't be removed for lss file.
.include "ATxmega128A1Udef.inc"
.list

.org 0x0000
        rjmp MAIN

.org 0x0100
MAIN:
        ldi r16, 0xFF                   ;load 0xFF to r16
        sts PORTE_DIRSET, r16    ;set all of PortE to outputs
        sts PORTE_OUT, r16              ;set all bits of PortE (Turn on all LEDs)
```

```
LOOP:
        rjmp LOOP
```

Part B

```
/*
 * lab2c.asm
 *
 * Lab 2 Part B
 * Name: Nicholas Imamshah
 * Section: 6957
 * TA Name: Daniel Gonzalez
 * Description: The purpose of this program is to alternate the nibbles of the LED array.
 */

.nolist ; This works, but the below file can't be removed for lss file.
.include "ATxmega128A1Udef.inc"
.list

.def lower = r18
.def upper = r19

.org 0x0000
        rjmp MAIN

.org 0x0100
MAIN:
        ldi r16, 0xFF                   ;set all bits of PortE to outputs
        sts PORTE_DIRSET, r16
        ldi lower, 0x0F         ;load named registers with corresponding bits
        ldi upper, 0xF0

MSB:
        sts PORTE_OUT, upper    ;set MSBits and delay
        rcall DELAY

LSB:
        sts PORTE_OUT, lower    ;set LSBits and delay
        rcall DELAY
        rjmp MSB

;Contents of delay.inc
;DELAY:
;       ldi r16, 111            ;load the count(r16) with the number of cycles we need to loop
;
;DELAY_LOOP:
;       dec r16                         ;decrement our count
;       brne DELAY_LOOP         ;if count != 0, repeat
;
;       ret
.include "delay.inc"
```

Part C

```
/*
 * lab2c.asm
 *
 * Lab 2 Part C
 * Name: Nicholas Imamshah
 * Section: 6957
 * TA Name: Daniel Gonzalez
 * Description: The purpose of this program is to interface with a keypad.
 */

KEYPAD:
        ldi r16, 0xFF
        sts PORTE_OUT, r16
        rcall KEYSCAN                   ;call the Keypad Scanning subroutine
```

```
        sts PORTE_OUT, r16              ;output result of KEYSCAN to LEDs

        rjmp KEYPRESSED
        rjmp KEYPAD

KEYPRESSED:
        lds r16, PORTF_IN               ;check PortF's input again
        cpi r16, 0xF0                   ;if it is < 0xF0, then one of the keys are pressed
        brlo KEYPRESSED                 ;loop until this is not the case

        rjmp KEYPAD

KEYSCAN:
        ldi r19, 0x18                   ;Need OPC set to PULLUP for all Keypad pins
        sts PORTF_PIN7CTRL, r19
        sts PORTF_PIN6CTRL, r19
        sts PORTF_PIN5CTRL, r19
        sts PORTF_PIN4CTRL, r19
        sts PORTF_PIN3CTRL, r19
        sts PORTF_PIN2CTRL, r19
        sts PORTF_PIN1CTRL, r19
        sts PORTF_PIN0CTRL, r19

        ldi r16, 0x0F                   ;set the LSNibble of PORTF to output
        sts PORTF_DIRSET, r16
        rcall COL1                              ;call to the first column to scan

        ret

INIT:
        sts PORTF_OUT, r16              ;initiates the bits for each columns scan
        nop
        lds r17, PORTF_IN               ;get the input bits from PortF
        ori r17, 0x0F                   ;bit mask the input to simplify code

        ret

COL1:
        ldi r16, 0x0E                   ;column 1 is 0b1101
        rcall INIT                              ;check for pressed key

        cpi r17, 0xEF                   ;check if row 1
        breq PRESS_1
        cpi r17, 0xDF                   ;check if row 2
        breq PRESS_4
        cpi r17, 0xBF                   ;check if row 3
        breq PRESS_7
        cpi r17, 0x7F                   ;check if row 4
        breq PRESS_ST

        rjmp COL2                               ;move on to column 2

        PRESS_1:                        ;load value corresponding to key pressed
                ldi r16, 0x01
                ret
        PRESS_4:
                ldi r16, 0x04
                ret
        PRESS_7:
                ldi r16, 0x07
                ret
        PRESS_ST:
                ldi r16, 0x0E
                ret

COL2:
        ldi r16, 0x0D                   ;column 2 is 0b1011
        rcall INIT                              ;check for pressed key
```

```
        cpi r17, 0xEF                   ;check if row 1
        breq PRESS_2
        cpi r17, 0xDF                   ;check if row 2
        breq PRESS_5
        cpi r17, 0xBF                   ;check if row 3
        breq PRESS_8
        cpi r17, 0x7F                   ;check if row 4
        breq PRESS_0

        rjmp COL3                               ;move on to column 3

        PRESS_2:                        ;load value corresponding to key pressed
                ldi r16, 0x02
                ret
        PRESS_5:
                ldi r16, 0x05
                ret
        PRESS_8:
                ldi r16, 0x08
                ret
        PRESS_0:
                ldi r16, 0x00
                ret

COL3:
        ldi r16, 0x0B                   ;column 3 is 0b0111
        rcall INIT                              ;check for pressed key

        cpi r17, 0xEF                   ;check if row 1
        breq PRESS_3
        cpi r17, 0xDF                   ;check if row 2
        breq PRESS_6
        cpi r17, 0xBF                   ;check if row 3
        breq PRESS_9
        cpi r17, 0x7F                   ;check if row 4
        breq PRESS_NUM

        rjmp COL4                               ;move on to column 4

        PRESS_3:                        ;load value corresponding to key pressed
                ldi r16, 0x03
                ret
        PRESS_6:
                ldi r16, 0x06
                ret
        PRESS_9:
                ldi r16, 0x09
                ret
        PRESS_NUM:
                ldi r16, 0x0F
                ret

COL4:
        ldi r16, 0x07                   ;column 4 is 0b1110
        rcall INIT                              ;check for pressed key

        cpi r17, 0xEF                   ;check if row 1
        breq PRESS_A
        cpi r17, 0xDF                   ;check if row 2
        breq PRESS_B
        cpi r17, 0xBF                   ;check if row 3
        breq PRESS_C
        cpi r17, 0x7F                   ;check if row 4
        breq PRESS_D

        rjmp COL1
```

```
PRESS_A:                        ;load value corresponding to key pressed
        ldi r16, 0x0A
        ret
PRESS_B:
        ldi r16, 0x0B
        ret
PRESS_C:
        ldi r16, 0x0C
        ret
PRESS_D:
        ldi r16, 0x0D
        ret
```

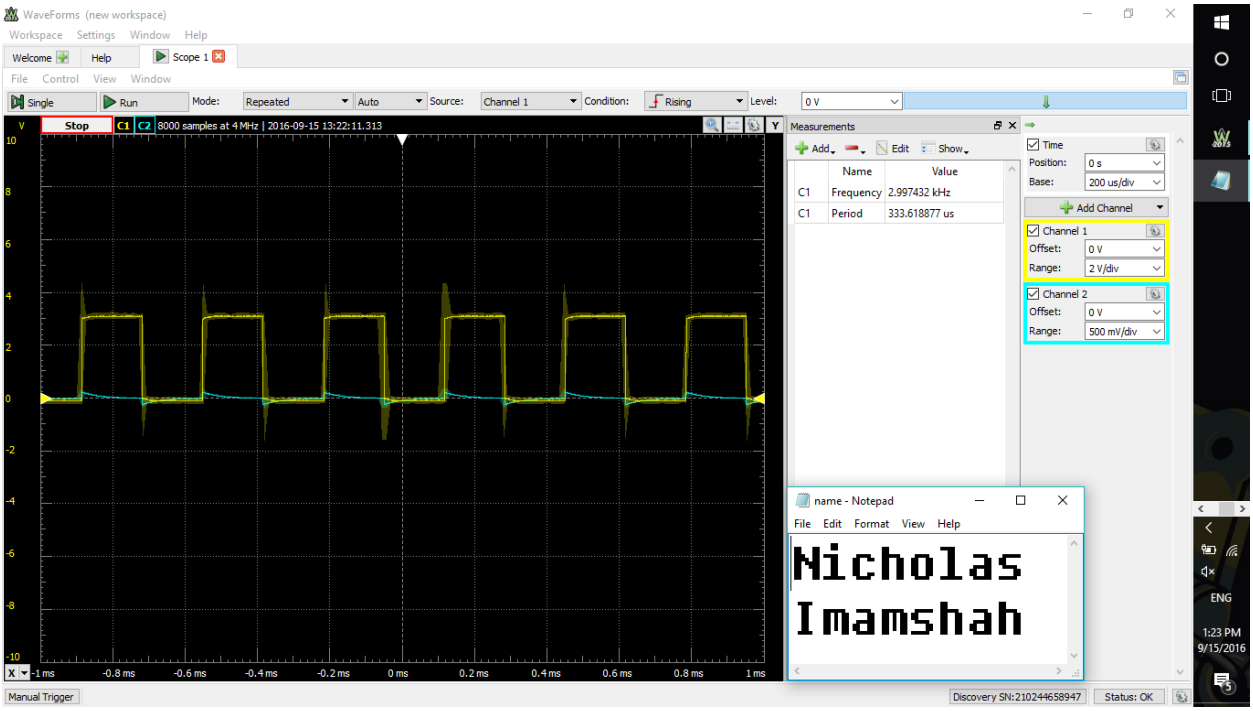
Part D

```
/*
 * lab2d.asm
 *
 * Lab 2 Part D
 * Name: Nicholas Imamshah
 * Section: 6957
 * TA Name: Daniel Gonzalez
 * Description: The purpose of this program is to use the keypad interface developed previously.
 */

.nolist
.include "ATxmega128A1Udef.inc"
.list

.org 0x0000
        rjmp MAIN

.org 0x0100
.include "keypad.inc"
MAIN:
        ldi r17, 0xFF                   ;set PortE as all OUTPUT
        sts PORTE_DIRSET, r17
        rcall KEYPAD                            ;jump to LOOP to begin repeated scanning
```

i) Appendix

Part B Screenshot

Lab: 2
Nicholas Imamshah
Section: 6957



Delay

```
/*
 * delay.asm
 *
 * Lab 2 Part B
 * Name: Nicholas Imamshah
 * Section: 6957
 * TA Name: Daniel Gonzalez
 * Description: The purpose of this program is to create a delay.
 */

.equ cycles = 111
.def count = r17

DELAY:
        ldi count, cycles       ;load the count(r17) with the number of cycles we need to loop

DELAY_LOOP:
        dec count                       ;decrement our count
        brne DELAY_LOOP         ;if count != 0, repeat

        ret
```