MAS 863
How To Make (almost) Anything
2009

Microcontroller Programming

001. Assembler is a challenging language for me. The first thing I did was to add comments where I had difficulties in understanding codes. Here is the code that I modified to the board to be reacted to the button swithch. For the instuctions, I consulted with the ATMEL instructions nomenclature here

http://www.atmel.com/dyn/resources/prod_documents/doc0856.pdf

For the general instructoin, I checked the Introduction to Gerd's AVR Assembler website

http://www.avr-asm-tutorial.net/gavrasm/GAVRASMI.html

I appreciated this week's guru David and Jonathan. David's intuitive and easy explanations were the key part of my understanding of the assembler and hardware.

```
;
; hello.echo.44.asm;
; 115200 baud serial echo and LED hello-world program
;
; Neil Gershenfeld
; CBA MIT 10/12/09
;
; (c) Massachusetts Institute of Technology 2009
; Permission granted for experimental and personal use;
; license for commercial sale available from MIT.
;
;I added comments for my understadning and future reference here
;John Juhong Park
;MAS 863 2009 Fall
;PhD in Architecture
;Design Computation Group

;General Instruction 01
;To compile ASM into HEX
;Use "gavrasm"
;http://www.avr-asm-tutorial.net/avr_en/beginner/index.html
;example
;"gavrasm hello.echo.44.asm"

;General Instruction 02
;After write a program into ATTiny44
;Open a terminal,
;Type : "screen /dev/ttys0 115200"
;It will open a screen terminal in which whenever a keyboard is hit, LED blink
;
;To exit
;Ctrl+A
;k
;y
;
;If there is already opened screen
;I need to kill the operation
;To do that
;screen -r
;it will find any opreating screen
;In my case, I found these
;There are several suitable screens on:
;       9803.pts-1.fab-desktop      (Detached)
;       9674.pts-0.fab-desktop      (Detached)
;
;type
;fab@fab-desktop:~$ screen -r 9803.pts-1.fab-desktop
;[screen is terminating]

;General Instruction 03
;avrdude -p t44 -c bsd -U lfuse:w:0x7E:m
;this command initiate tye crystal
;avrdude -p t44 -c bsd -U flash:w:hello.echo.44.hex
;this runs the program
;order does not, but without running both, the board is not working


.include "tn44def.inc"
;Text file containing declarations, headers, functions, or other data referenced by a program's source code. (JP)

.equ led_pin = PB2
; LED pin
```

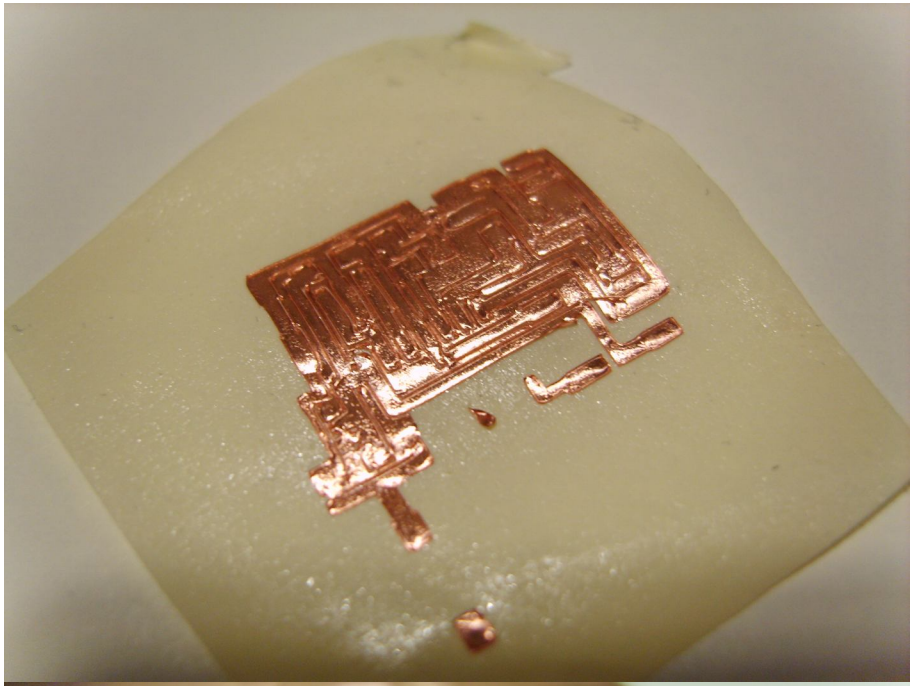
```
;(PCINT10/INT0/OC0A/CKOUT) PB2   (JP)
.equ led_port = PORTB
; LED port
;Just to identify which port it is (JP)

.equ led_dir = DDRB; LED dir
;PB2 is assigned to led_pin

.equ txpin = PA6; transmit pin
.equ rxpin = PA7; receive pin

.equ comm_port = PORTA; comm port
.equ comm_dir = DDRA; comm direction
;DDRA ??Port A Data Direction Register
.equ comm_pins = PINA; comm pins
;PINA ??Port A Input Pins

.equ button1 = PA2; button
.equ button1_port = PORTA; button
.equ button1_dir = DDRA; button
.equ button1_pins = PINA;
;I defined button1 and it is assigned to PA0

.def bitcnt = R16; bit counter
.def temp = R17; temporary storage
.def temp1 = R18; temporary storage
.def txbyte = R19; transmit byte
.def rxbyte = R20; receive byte
;Registor definitions

;
; print
;
.macro print
    ldi zl,low(@0*2)
    ;Loads an 8 bit constant directly to register 16 to 31.
    ldi zh,high(@0*2)
    rcall print_db
    .endmacro
```

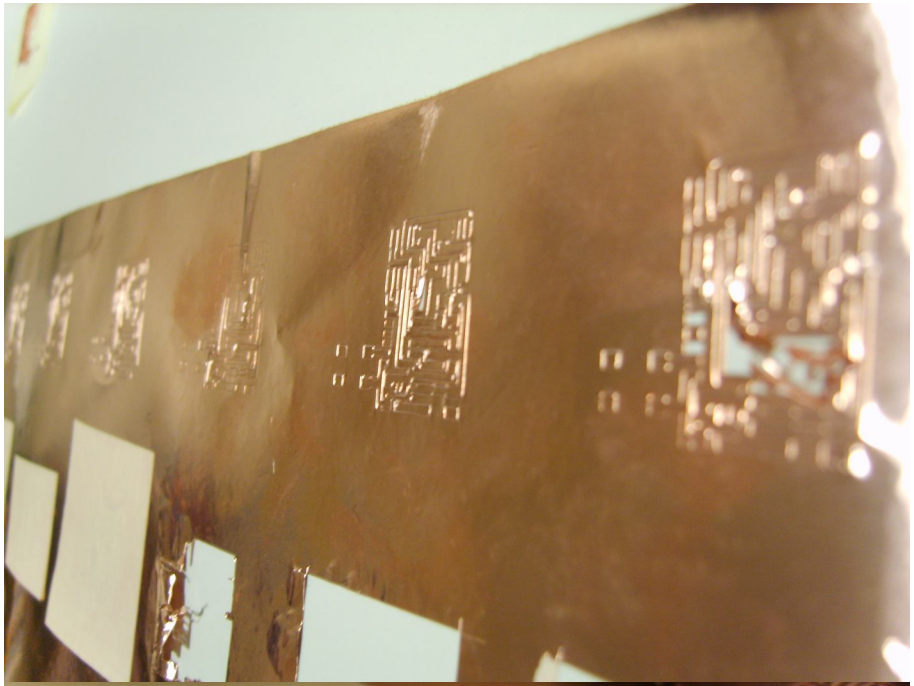
```
.cseg
.org 0
rjmp reset

;
; half_bit_delay
; serial half bit delay
;
half_bit_delay:
    ldi temp, 25; 115200 baud (20 MHz clock /1)
    half_bit_delay_loop:
        dec temp
        brne half_bit_delay_loop
    ret
;
; putchar
; assumes no line driver (doesn't invert bits)
;
putchar:
    ldi bitcnt, 10; 1 start + 8 data + 1 stop bit
    com txbyte; invert everything
    ;One Complement - This instruction performs a One Complement of register Rd.
    sec
    ; set start bit
    ;Sets the Carry Flag (C) in SREG (Status Register).
    putchar0:
        brcc putchar1; if carry set
        ;Branch if Carry Cleared
        sbi comm_port, txpin; send a '0'
        rjmp putchar2; else
        ;Relative jump to an address within PC - 2K +1 and PC + 2K (words).
    putchar1:
                cbi comm_port, txpin            ; send a '1'
                  ;Clears a specified bit in an I/O Register.
                nop; even out timing
                  ;This instruction performs a single cycle No Operation.
    putchar2:
          rcall half_bit_delay; bit delay
```

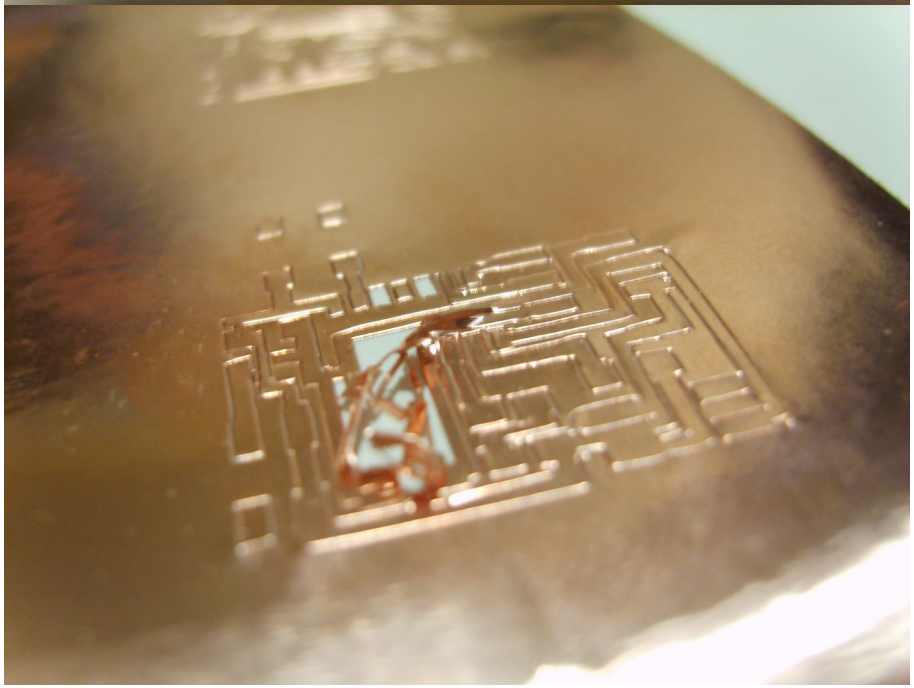
```
        ;Relative call to an address within PC - 2K + 1 and PC + 2K (words).
         rcall half_bit_delay
         lsr txbyte; get next bit
         ;Logical Shift Right
         ;-Shifts all bits in Rd one place to the right. Bit 7 is cleared.
         dec bitcnt; if not all bits sent
         ;Subtracts one -1- from the contents of register Rd and
         ;places the result in the destination register Rd.
         brne putchar0; send next bit
         ;Branch if Not Equal
         ;Conditional relative branch. Tests the Zero Flag (Z) and

;branches relatively to PC if Z is cleared.
    ret;
;
; getchar
; assumes no line driver (doesn't invert bits)
;
getchar:
    ldi bitcnt, 9
    ; 8 data + 1 stop bit
    ;Loads an 8 bit constant directly to register 16 to 31.
    ;
    getchar1:
        sbis comm_pins, rxpin; wait for start bit
        rjmp getchar1
    rcall half_bit_delay; delay to middle of bit
    getchar2:
        rcall half_bit_delay; bit delay
        rcall half_bit_delay; "
        clc; clear carry
        sbis comm_pins, rxpin; if RX pin high skip
            sec; otherwise set carry
        dec bitcnt
        breq getchar3; return if all bits read
        ror rxbyte; otherwise shift bit into receive byte
        rjmp getchar2; go get next bit
    getchar3:
        ret
;
; blink_delay
; LED blink delay
;
blink_delay:
    ldi temp, 255
    blink_delay_loop:
        ldi temp1, 255
        blink_delay_loop1:
            dec temp1
            brne blink_delay_loop1
        dec temp
        brne blink_delay_loop
    ret
;
; blink
; blink the LED
;
blink:
    sbi led_port, led_pin
    rcall blink_delay
    cbi led_port, led_pin
    ret
;
; print_db
; prints a null-terminated .db string
;
print_db:
    print_loop:
        lpm
        ;Load Program Memory-loads one byte pointed to by the Z-register into the destination
        register Rd
        mov txbyte,R0

        cpi txbyte,0
        ;Compare with Immediate - This instruction performs a compare between register Rd and
```

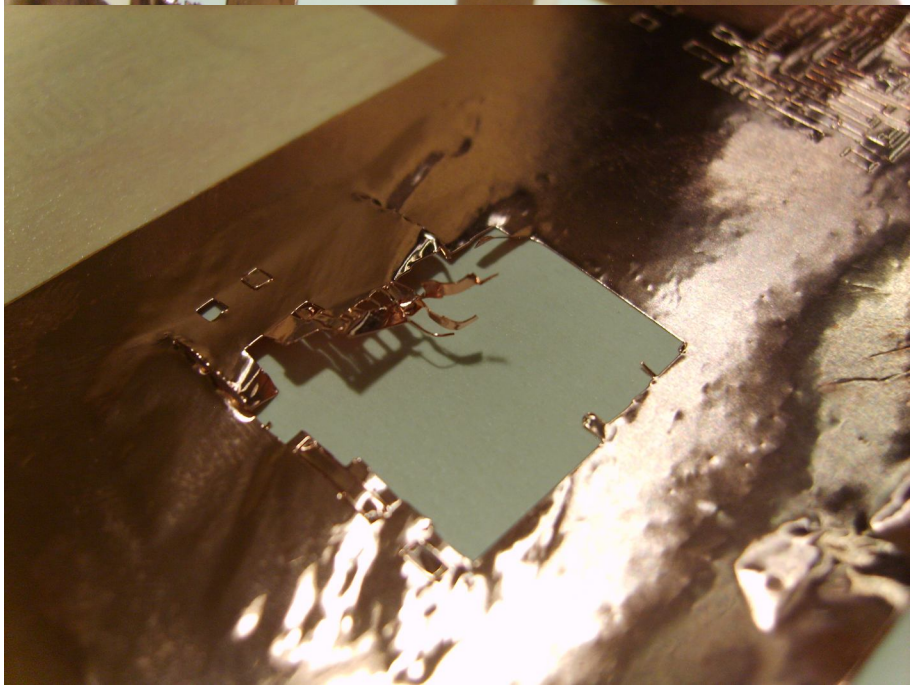
```
        ;constant.   The register is not changed. All conditional branches can be used after this
        ;instruction.
        breq return
        ;Branch if Equal - Conditional relative branch. Tests the Zero Flag (Z) and branches relatively
        ;to PC if Z is set. If the instruction is executed immediately after any of the instructions CP,
        ;CPI, SUB or SUBI, the branch will occur if and only if the unsigned or signed
        ;binary number represented in Rd was equal to the unsigned or signed binary number
        ;represented in Rr. This instruction branches relatively to PC in either direction (PC –
        ;63 destination PC + 64).
        rcall putchar
        adiw zl, 1
        ;Add Immediate to Word - Adds an immediate value (0 - 63) to a register pair and places the
        ;result in the register pair. This instruction operates on the upper four register pairs, and is well
        ;suited for operations on the pointer registers.
         rjmp print_loop
         ;Relative Jump -Relative jump to an address within PC - 2K +1 and PC + 2K (words).
    return:
        ret
        ;Return from Subroutine
;
; main program
;
reset:
    ;
    ; set fuse low byte to 0x7E for 20 MHz resonator
    ;
    ; set clock divider to /1
    ;
    ldi temp, (1 << CLKPCE)
    ;Load Immediate -Loads an 8 bit constant directly to register 16 to 31.
    ldi temp1, (0 << CLKPS3) | (0 << CLKPS2) | (0 << CLKPS1) | (0 << CLKPS0)
    out CLKPR, temp
    ;Store Register to I/O Location -
    out CLKPR, temp1
    ;
    ; set stack pointer to top of RAM
    ;
    ldi temp, high(RAMEND)
    ;Load Immediate -Loads an 8 bit constant directly to register 16 to 31.
    out SPH, temp
    ;Store Register to I/O Location -
    ldi temp, low(RAMEND)
    ;Load Immediate -Loads an 8 bit constant directly to register 16 to 31.
    out SPL, temp
    ;Store Register to I/O Location -
    ;
    ; init comm pin
    ;
    sbi comm_port, txpin                ;Set Bit in I/O Register
    sbi comm_dir, txpin                 ;Set Bit in I/O Register
    ;
    ; init LED pins
    ;
    cbi led_port, led_pin
    ;Clear Bit in I/O Register
    sbi led_dir, led_pin
    ;Set Bit in I/O Register
    ;
    ; init Button
    ;
    sbi button1_port, button1
    ;Clear Bit in I/O Register
    cbi button1_dir, button1
    ;Set Bit in I/O Register

    ;
    ; start main loop
    ;
    loop:
      sbic button1_pins, button1
            ;Skip if Bit in I/O Register is Cleared
      rjmp loop
            ;Relative jump to an address within PC - 2K +1 and PC + 2K (words)
      rcall blink
    rjmp loop
```

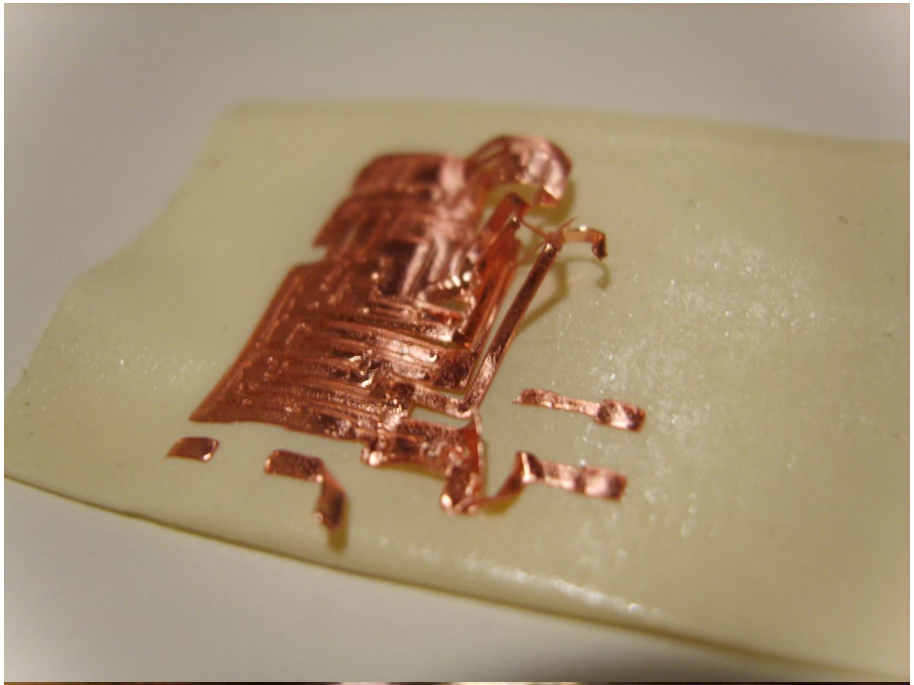
002.I used the Vynilcutter to make a circuit board. I tried three times with the original design and four times. I failed to get the original design circuit board continously.

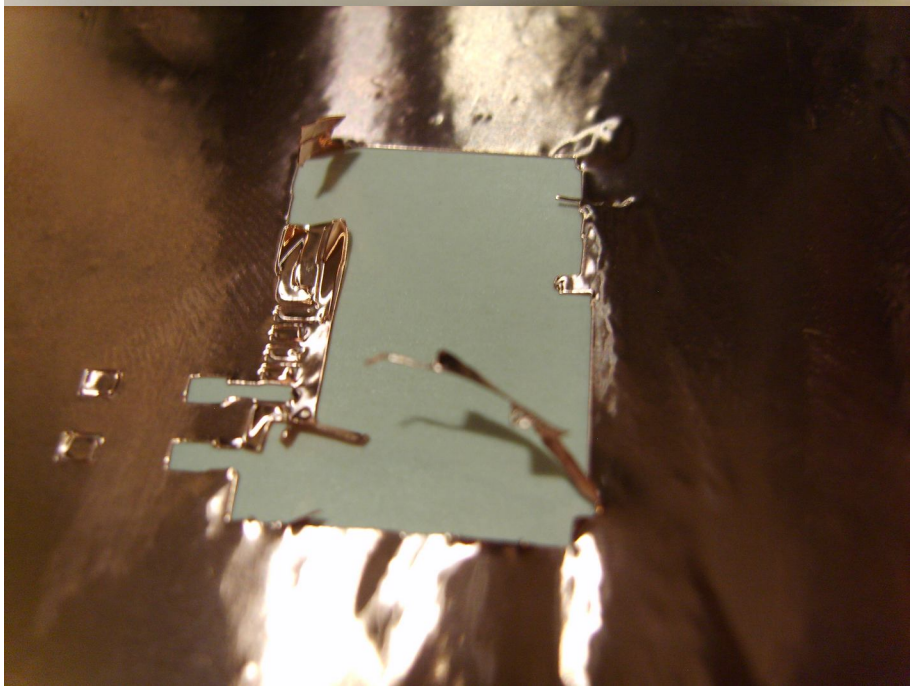




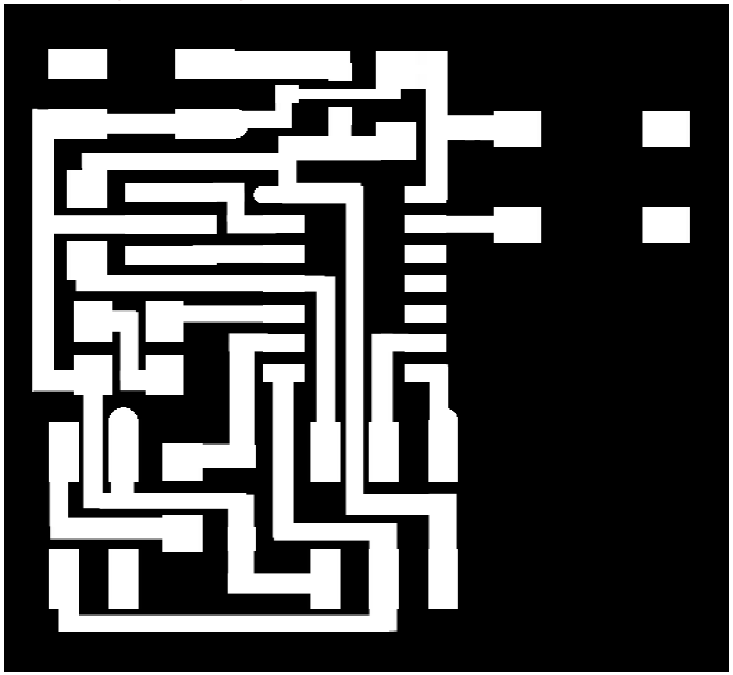




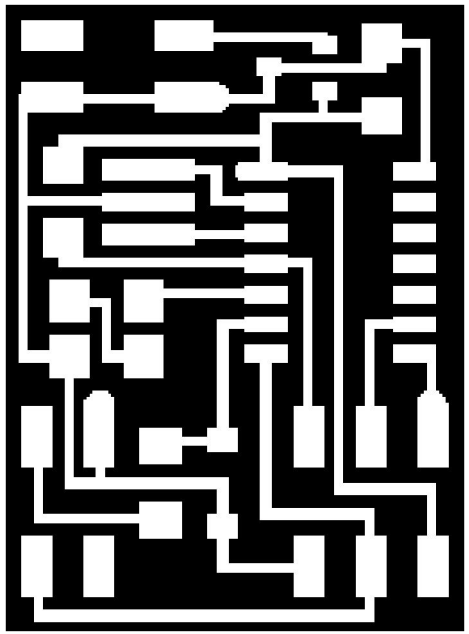


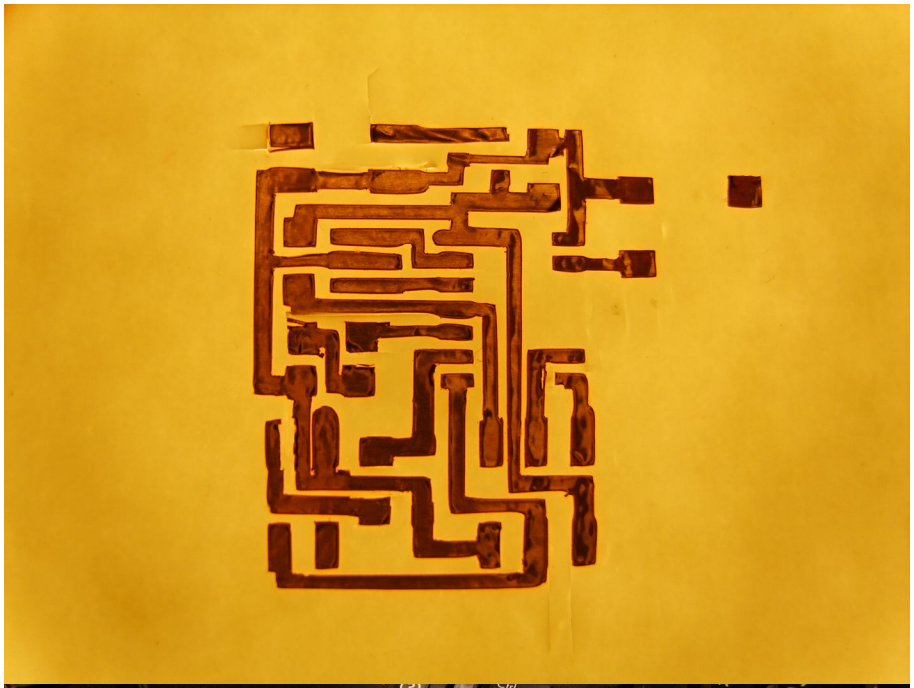


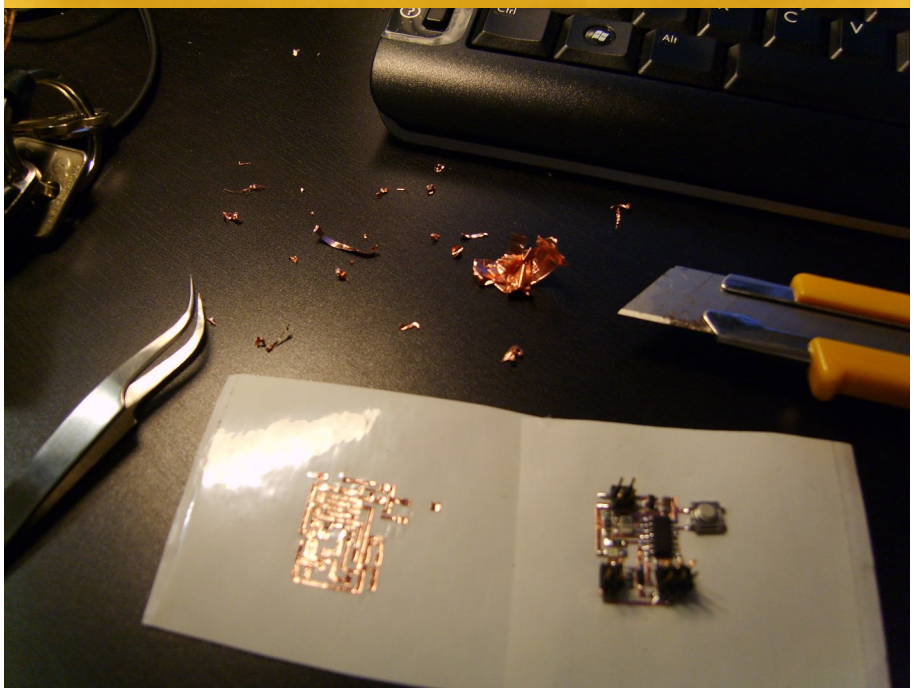
003. I changed the design – I made all the connection lines thicker.

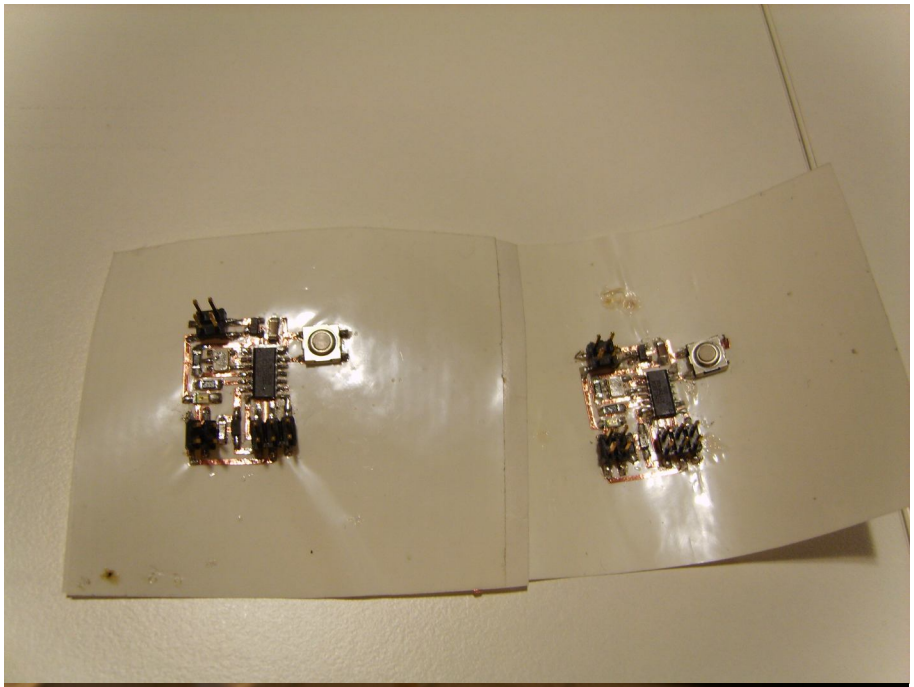


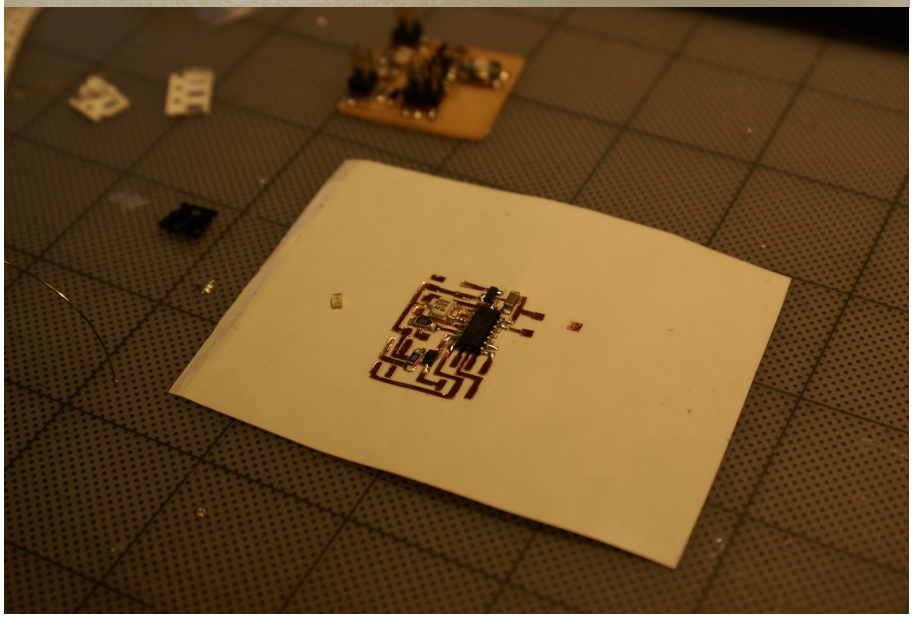
Changed Design (Above), Original Design (Below)



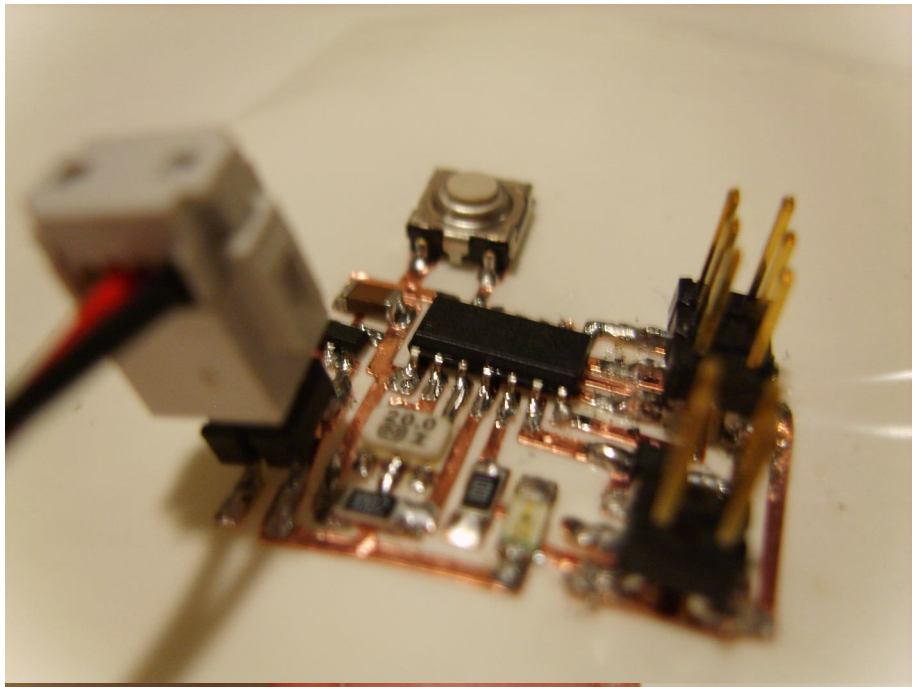


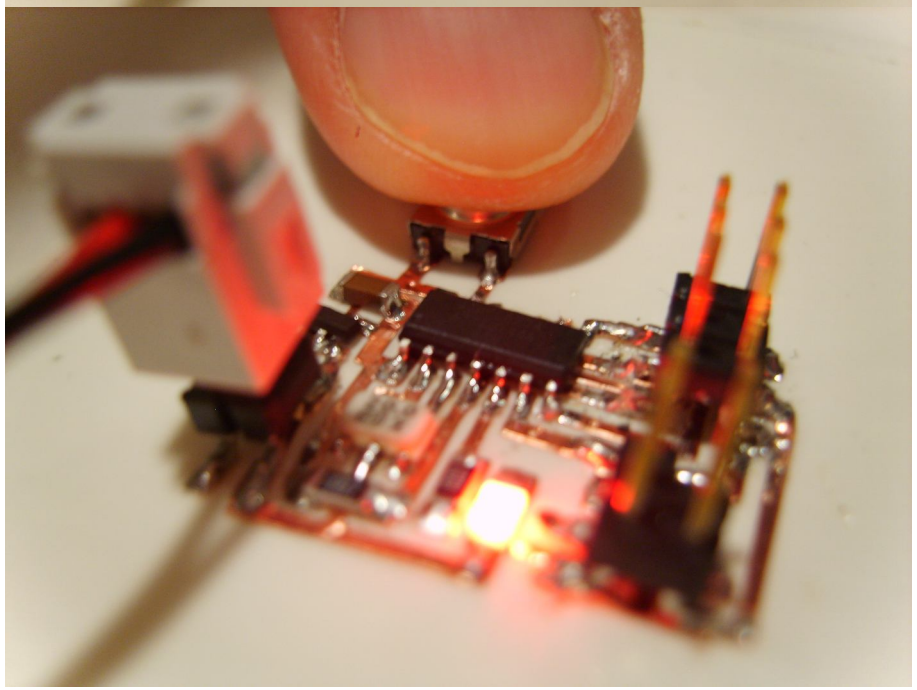






004. I finally made board with working button.





005. While I had a trouble with my board, Eric generously borrowed his susscessful board. When I explained how to connenct cables between the board and computer to Florence, I accidently connected the power cable into the communcation connector, then we saw some smoke from the board and burned some resistors, zener diode and LED . I need to figure out what broken parts are, replaced burned parts and need to reprogram his board. Eric's button was connected to PA2 (my button was connected to PA0). Fixing the assembly code this time was reletatively easy process.