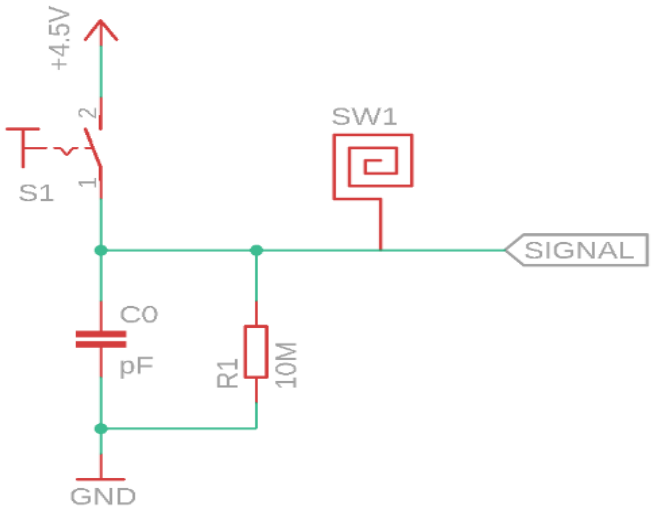
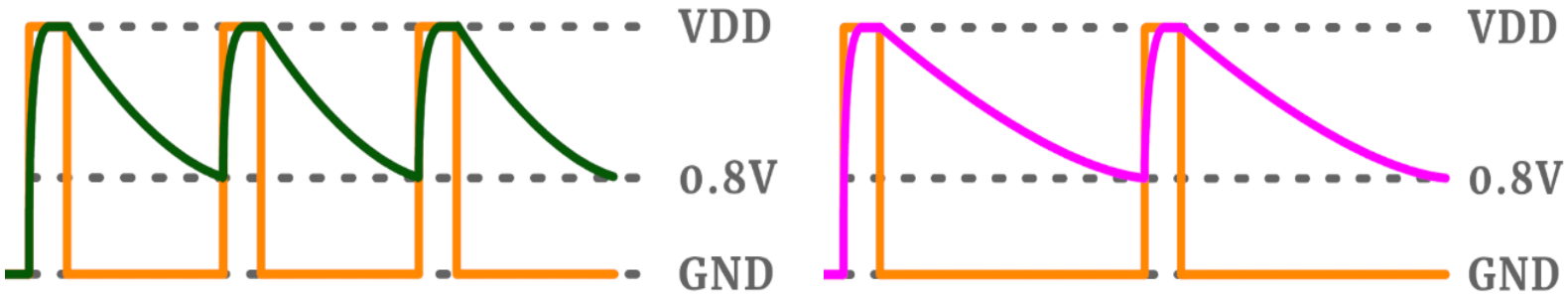
Capacitive Touch Sensor

How does a capacitive touch sensor work?





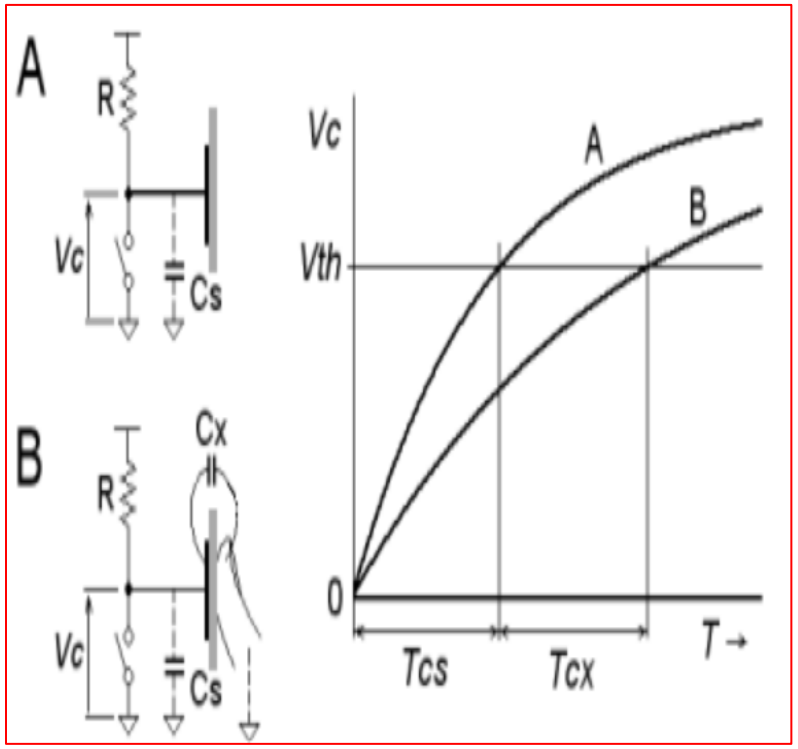
The main idea of capacitive touch sensor is the change of the capacitance in the circuit

First:

First idea:

- Turn on the switch until the main capacitor in the circuit charge
- Turn off the switch and calculate the discharge time of the capacitor in the resistor
- When we touch the capacitance of the circuit increases so the time of discharge increases
- At this this point by software , we detect a touch occur.

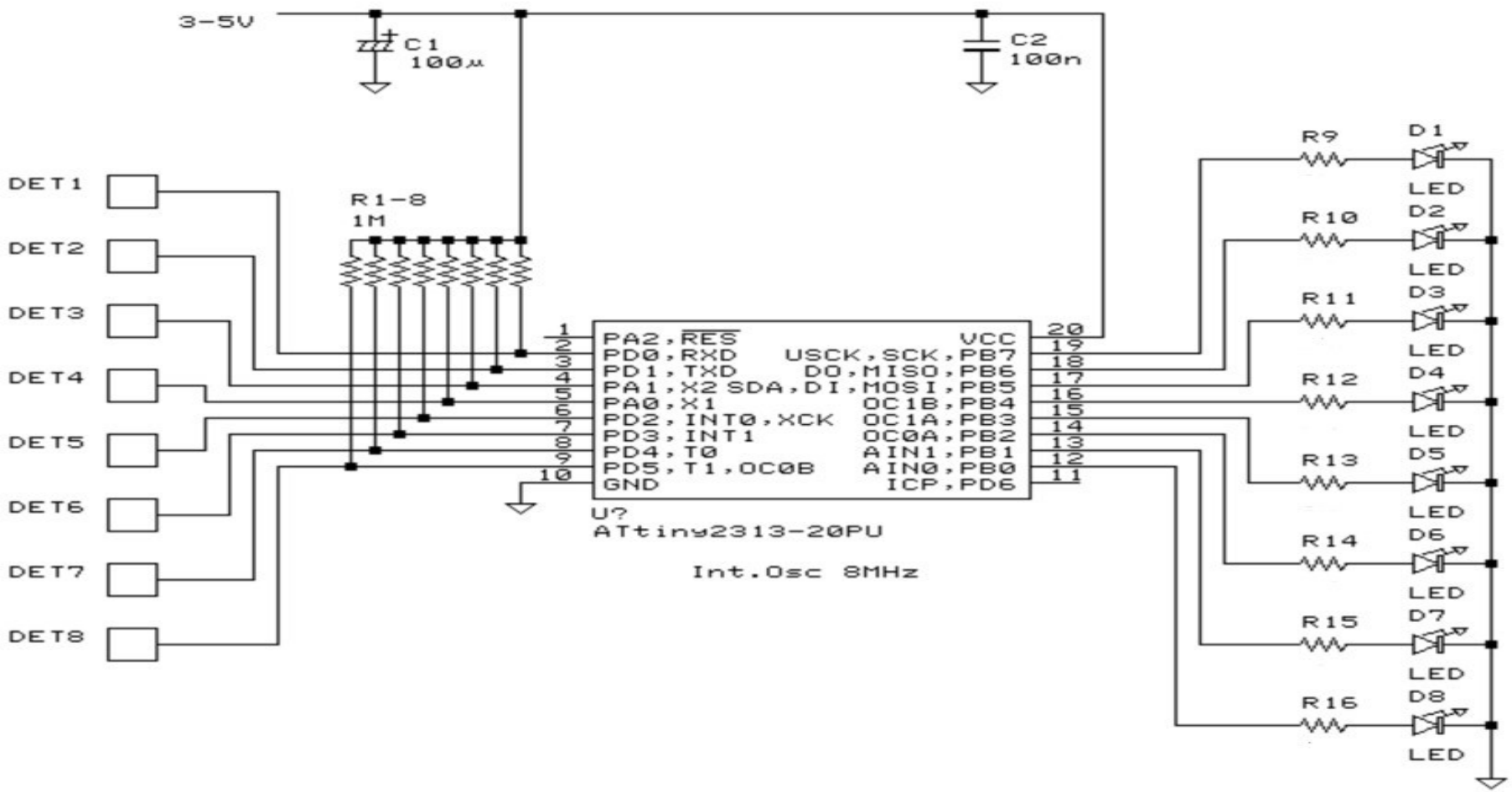
Continue: main idea



Second idea:

- In this idea we concern about charging time of the capacitor
- Without touching we calculate the time and then we calculate the time after touching
- We notice that the time after touching increases

Circuit Diagram



Software Code

In this project we use microcontroller ATTINY4313

```
#include <tiny4313.h>
#define SYSCLK    8000000UL

typedef unsigned long int uint32;
typedef unsigned short uint16;
typedef unsigned char uint8;

#define TIMER_CTRL  (*(volatile uint8 *)(0x4E)) // TCCR1B ==> Timer/Counter1 Control Register B
#define TIMER_CNTH  (*(volatile uint8 *)(0x4D)) // TCNT1H and TCNT1L ? Timer/Counter1
#define TIMER_CNTL  (*(volatile uint8 *)(0x4C)) // TCNT1H and TCNT1L ? Timer/Counter1

#define Pin0 0
#define Pin1 1
#define Pin2 2
#define Pin3 3
#define Pin4 4
#define Pin5 5
```

```
#define SREG     (*(volatile uint8 *)(0x5F))
#define CH0_OUT  (*(volatile uint8 *)(0x31)) // DDRD pin0
#define CH0_IN   (*(volatile uint8 *)(0x30)) // PIND pin0
#define CH1_OUT  (*(volatile uint8 *)(0x31)) // DDRD pin1
#define CH1_IN   (*(volatile uint8 *)(0x30)) // PIND pin1

#define CH2_OUT  (*(volatile uint8 *)(0x3A)) // DDRA pin1
#define CH2_IN   (*(volatile uint8 *)(0x39)) // PINA pin1
#define CH3_OUT  (*(volatile uint8 *)(0x3A)) // DDRA pin0
#define CH3_IN   (*(volatile uint8 *)(0x39)) // PINA pin0
#define CH4_OUT  (*(volatile uint8 *)(0x31)) // DDRD pin2
#define CH4_IN   (*(volatile uint8 *)(0x30)) // PIND pin2
#define CH5_OUT  (*(volatile uint8 *)(0x31)) // DDRD pin3
#define CH5_IN   (*(volatile uint8 *)(0x30)) // PIND pin3
#define CH6_OUT  (*(volatile uint8 *)(0x31)) // DDRD pin4
#define CH6_IN   (*(volatile uint8 *)(0x30)) // PIND pin4
#define CH7_OUT  (*(volatile uint8 *)(0x31)) // DDRD pin5
#define CH7_IN   (*(volatile uint8 *)(0x30)) // PIND pin5


uint16 touch(uint8 channelNumber);
void get_sens (uint16 *res, uint8 chs);
```

```
void main(void)
{
    uint16 reference[8],val_after_sense[8];
    volatile int Differance, c;
    PORTD = 0b1000000;
    DDRD =  0b0111111;     //port D output
    PORTA = 0b100;
    DDRA  = 0b011;        // port A output


    get_sens(reference,8);    /* Get reference count for each channel */

   PORTB = 0b00000000;
   DDRB  = 0b11111111;  //port B input

   while(1) {
       get_sens(val_after_sense, 8);
       for (c = 0; c < 8; c++) {  // 8 channels to sense
           Differance = val_after_sense[c] - reference[c];
           if (Differance < 3) PORTB &= ~(1 << (7 - c));
           if (Differance > 5) PORTB |= (1 << (7 - c));
       }
   }
}
```

```
void get_sens (uint16 *ref,uint8 chs)
{
      uint8 ch=0;
      uint32 loop=0;

    /* Clear count accumulator */
    for (ch = 0; ch < chs; ch++)
      ref[ch] = 0;

    /* Capture and accumulate integration time for 1/60 second */
    OCR0A = 8000000UL / 1024 / 60;  // Output Compare Register A
    TCNT0 = 0; // Timer/Counter Register
    TIFR |=(0b00000001); //Output Compare Flag 0 A write 1 to clear the flag
    TCCR0B = 0b101; // Timer/Counter Control Register B  clkI/O/1024 (From pre-scaler)
    loop = 0;
    do {
        for (ch = 0; ch < chs; ch++)
        ref[ch] += touch(ch);
        loop++;
    } while ((TIFR & (0b00000001)) == 0);
    TCCR0B = 0; //No clock source (Timer/Counter stopped)
    /* Average calculations */
    for (ch = 0; ch < chs; ch++)
      ref[ch] /= loop;
}
```

```
uint16 touch(uint8 channelNumber)
{
    uint8 init_val = 1;
    uint16 result=0;

    TIMER_CNTH = 0;
    TIMER_CNTL = 0;
    SREG &=(0b01111111); // Disable the Global Interrupt
    if (channelNumber == 0)
    {
        TIMER_CTRL = init_val;
        CH0_OUT &= ~(1 << Pin0); // make the Channel input
        while (!(CH0_IN & (1 << Pin0)));
        CH0_OUT |= (1 << Pin0);  // make the Channel output
    }
    else if (channelNumber == 1)
    {
        TIMER_CTRL = init_val;
        CH1_OUT &= ~(1 << Pin1);
        while (!(CH1_IN & (1 << Pin1)));
        CH1_OUT |= (1 << Pin1);
    }
    else if (channelNumber == 2)
    {
        TIMER_CTRL = init_val;
        CH2_OUT &= ~(1 << Pin1);
        while (!(CH2_IN & (1 << Pin1)));
        CH2_OUT |= (1 << Pin1);
    }
    else if (channelNumber == 3)
    {
        TIMER_CTRL = init_val;
        CH3_OUT &= ~(1 << Pin0);
        while (!(CH3_IN & (1 << Pin0)));
        CH3_OUT |= (1 << Pin0);
    }
    else if (channelNumber == 4)
    {
        TIMER_CTRL = init_val;
        CH4_OUT &= ~(1 << Pin2);
        while (!(CH4_IN & (1 << Pin2)));
        CH4_OUT |= (1 << Pin2);
    }
    else if (channelNumber == 5)
    {
        TIMER_CTRL = init_val;
        CH5_OUT &= ~(1 << Pin3);
        while (!(CH5_IN & (1 << Pin3)));
        CH5_OUT |= (1 << Pin3);
    }
    else if (channelNumber == 6)
    {
        TIMER_CTRL = init_val;
        CH6_OUT &= ~(1 << Pin4);
        while (!(CH6_IN & (1 << Pin4)));
        CH6_OUT |= (1 << Pin4);
    }
    else if (channelNumber == 7)
    {
        TIMER_CTRL = init_val;
        CH7_OUT &= ~(1 << Pin5);
        while (!(CH7_IN & (1 << Pin5)));
        CH7_OUT |= (1 << Pin5);
    }

    TIMER_CTRL =0x00; // to stop counter
    result=TCNT1;   // read the timer count register
    return result;
}
```