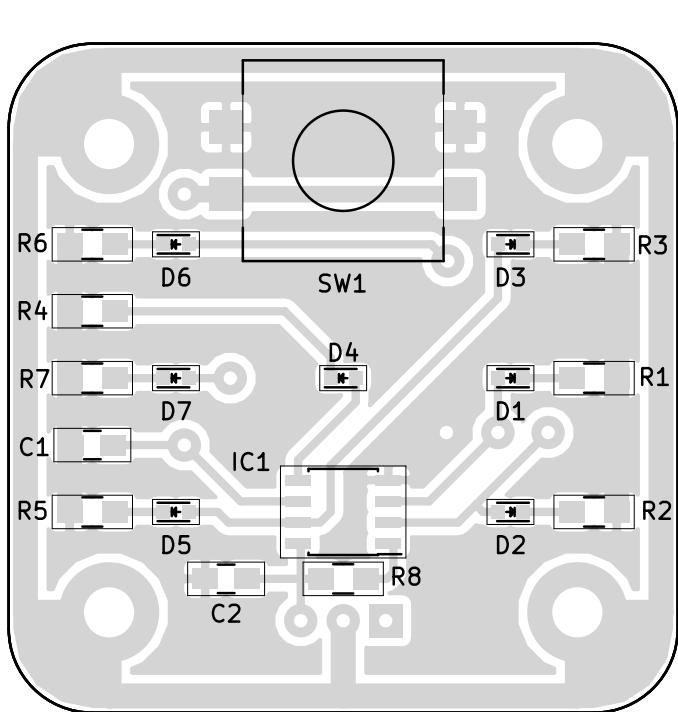
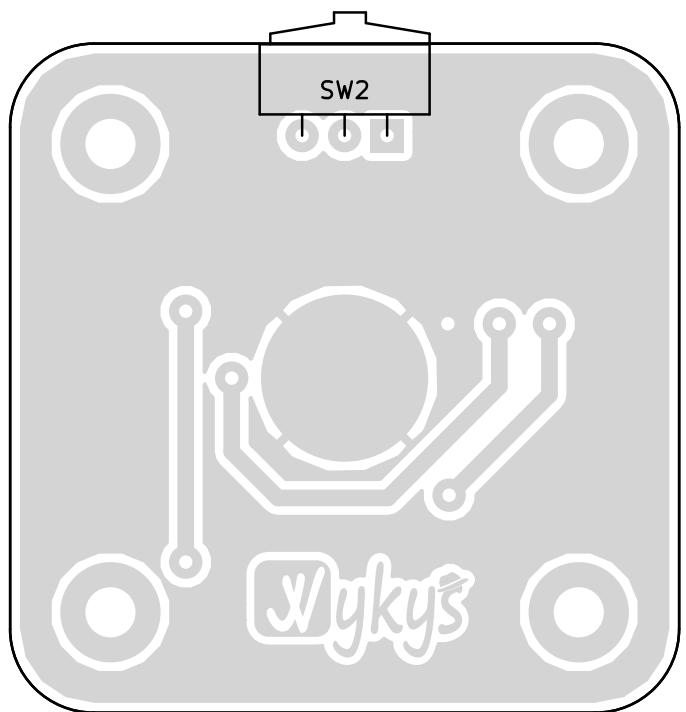


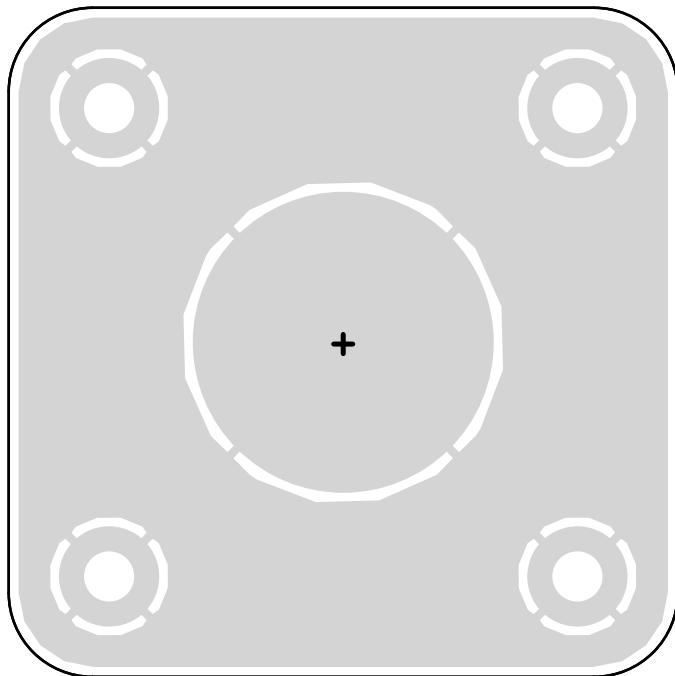
Obrázek 1: Schéma zapojení



Obrázek 2: Osazovací plán strana součástek



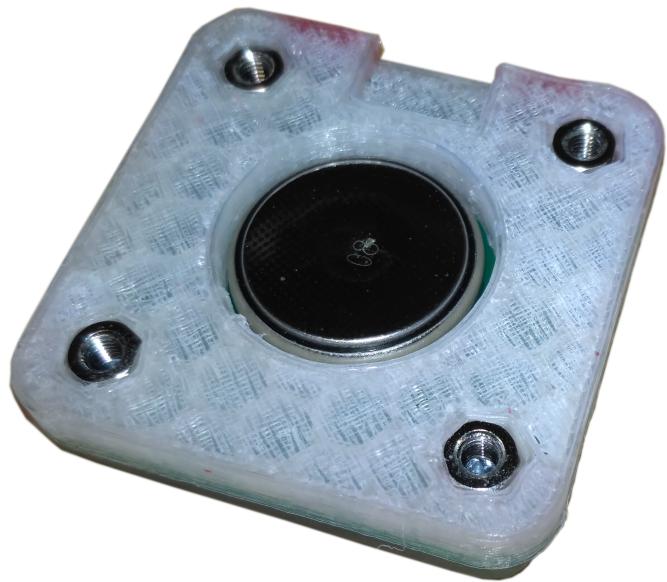
Obrázek 3: Osazovací plán strana spojů



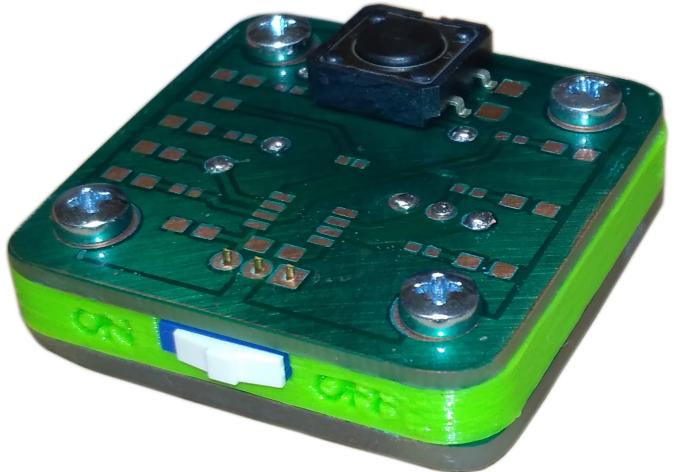
Obrázek 4: Spodní destička, slouží jako přívod kladného pólu baterie k hlavní desce

Tabulka 1: Seznam součástek

$R_1 - R_7$	100 Ω	rezistor
R_8	10 $k\Omega$	rezistor
C_1, C_2	100 nF	kondenzátor
$D_1 - D_7$	LTW-C190DA5	LED
IC_1	ATTINY13A-SSU	mikrokontrolér
SW_1	B3FS-4002P	tlačítko
SW_2	ESP2010	přepínač
BT_1	CR2032	baterie
$M_1 - M_4$	2· šroub a sloupek M3	mechanické díly



Obrázek 5: ukázka uložení distančních sloupků a baterie CR2032



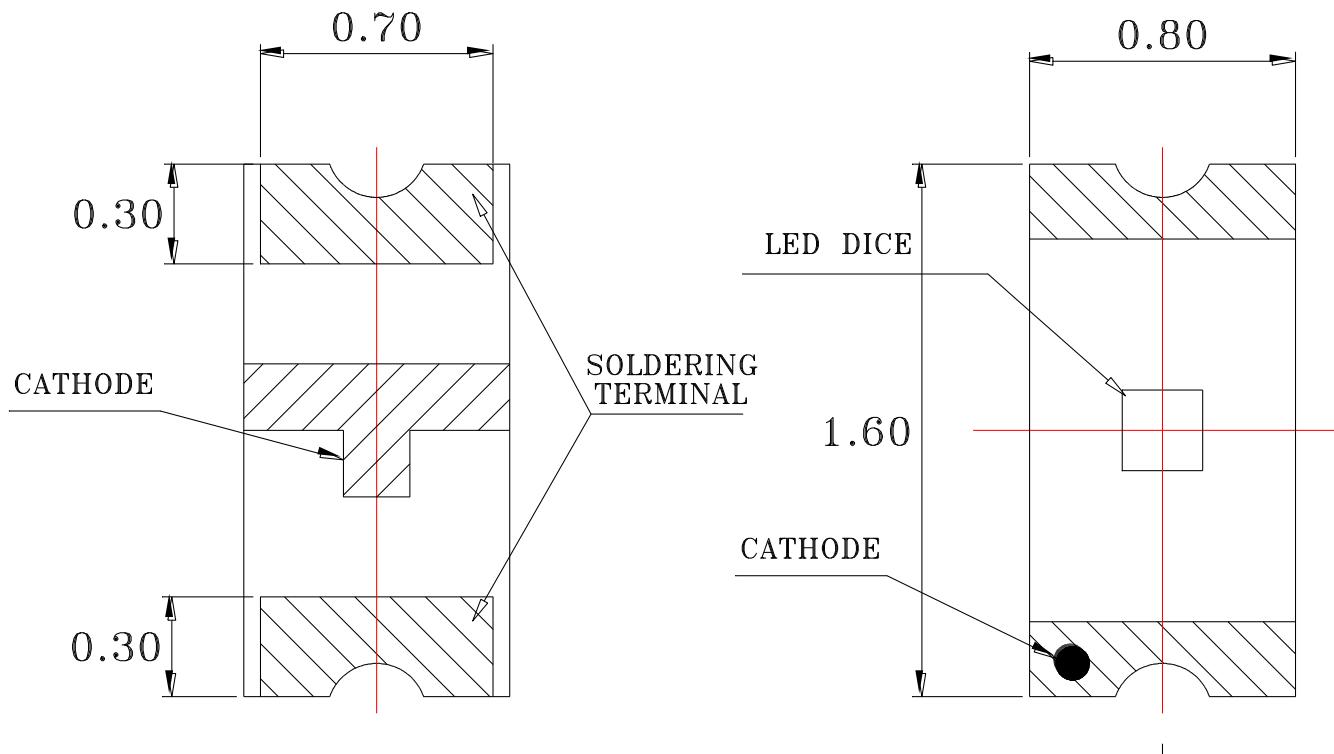
Obrázek 6: smontovaná kostka bez osazených součásťek

Osazování

Osazování je nejlepší provádět od nejmenších součástek po ty největší. Je tedy vhodné napřed osadit LED diody v pouzdře 0603, poté rezistory a kondenzátory v pouzdrech 0805. Následně mikrokontrolér v pouzdře SO-8 a nakonec tlačítko. Vypínač je vhodné přiletovat až když je kostka smontovaná, tak aby dobře pasoval do předchystaného otvoru.

Pozor na správnou polarizaci LED diod! Dioda je polarizovaná součástka, která má anodu a katodu. Katoda je označena malým kolečkem s horní strany a zelenou značkou ze strany spodní. Pokud budou diody nesprávně otočeny nebudou svítit. Správně otočit je třeba i mikrokontrolér, který při nevhodném otočení shoří. na osazovacím plánu je jednička mikrokontroléru označena čárkou. Poslední věcí, na kterou je třeba dávat obzvláště

pozor je správné otočení baterie. Kladný pól, ten větší na kterém je napsáno obvykle +, musí být položen na velkou plošku, sodní destička, na které nejsou součástky. Záporný pól, ten menší je třeba připojit k malé ploše, která je na straně součástek na hlavní destičce. Při špatném otočení baterie je kostka zničena! Tak hodně štěstí.



Obrázek 7: Označení katody

Jak to funguje?

Srdcem kostky je malíčký mikrokontrolér ATTINY13, což je osmibitový čip s architekturou AVR. Jeho větší bratříčky lze nalézt třeba v ARDUINU. Mikrokontrolér nastaví svůj interní čítač tak, aby čítal od nuly do pětky. Dále nastaví přerušení na sestupnou hranu, které nastane při stisku tlačítka. Poté co vše nastaví, tak skočí do nekonečného cyklu.

Když je stisknuto tlačítko, tak nastane přerušení. Program je přerušen a po uložení adresy s programového čítače (to je místo kde se v programu právě nacházíme) do zásobníku (což je paměť typu Last In Last Out) skočí na obslužný podprogram přerušení. Dokud je tlačítko stisknuto, tak je na výstupních pinech generováno napětí takovým způsobem, který způsobuje postupné rozsvěcování LED a vyvolává efekt, že se točí.

Po uvolnění tlačítka je s registru čítače vyzvednuta aktuální hodnota. Tato hodnota se dekóduje pomocí tabulky a rozsvítí se příslušné LED, které zobrazí výslední číslo. Při zobrazování čísla ještě číslo třikrát zabliká, během doby kdy bliká najde losovat další číslo. To je kvůli omezení podvádění několika rychlými stisky za sebou.

Když skončí podprogram přerušení, tak je ze zásobníku opět vyzvednuta adresa a program se opět vrátí do nekonečné smyčky.

Rezistory $R_1 - R_7$ tu jsou kvůli nastavení pracovního bodu LED diody. Bez nich by LED shořely. Rezistor R_8 tu je kvůli nastavení vhodné úrovně na RESETu. Bez něj by se zařízení mohlo restartovat. Kondenzátor C_1 slouží k eliminaci kmitů napětí při stisku či uvolnění tlačítka. Kondenzátor C_2 tu je kvůli odfiltrování rušení a možnosti v případě potřeby poskytnout mikrokontrolérovi krátkodobě potřebný proud.

Zdrojový kód

¹ `#include <avr/io.h>`

² `#include <avr/interrupt.h>`

```
4 #define A (1<<0)
5 #define B (1<<2)
6 #define C (1<<3)
7 #define D (1<<4)
8 #define T (1<<1)
9
10#define delay() for (t=0; t<300; t++) asm("nop")
11
12const uint8_t tabulka [] = {
13    T | A,           // 1
14    T | B,           // 2
15    T | A | B,       // 3
16    T | B | C,       // 4
17    T | A | B | C,   // 5
18    T | B | C | D,   // 6
19    T | A | B,       //
20    T | A | C,       // rotace
21    T | A | D       //
22};
23
24ISR(INT0_vect)
25{
26    uint8_t i, cislo;
27    uint16_t t;
28
29    cli();
30    PORTB = T;
31    while (!(PINB & T))
32    {
33        PORTB = tabulka [6];
34        delay();
35        PORTB = tabulka [7];
36        delay();
37        PORTB = tabulka [8];
38        delay();
39    }
40    cislo = tabulka [TCNT0];
41    for (i=0; i<3; i++)
42    {
43        PORTB = T;
44        delay();
45        PORTB = cislo;
46        delay();
47    }
48    sei();
49}
50
51int main(void)
52{
53    DDRB = A | B | C | D;
54    PORTB = T;
55
56    TCCR0A = 1<<WGM01; // nastav citac co CTC modu
57    OCR0A = 5;           // nastav porovnavaci registr
58    TCNT0 = 0x00;         // vynuluje citaci registr
59    TCCR0B = (1<<CS00); // nastav frekvenci citani clk/1024
60    MCUCR = 0x00;         // nastav preruseni na sestupnou hranu
61    GIMSK = 1<<INT0;     // povol preruseni od INT0
62    sei();               // globalni povoleni preruseni
63
64    while (1)
65    {
66        asm("nop");
67    }
68}
```