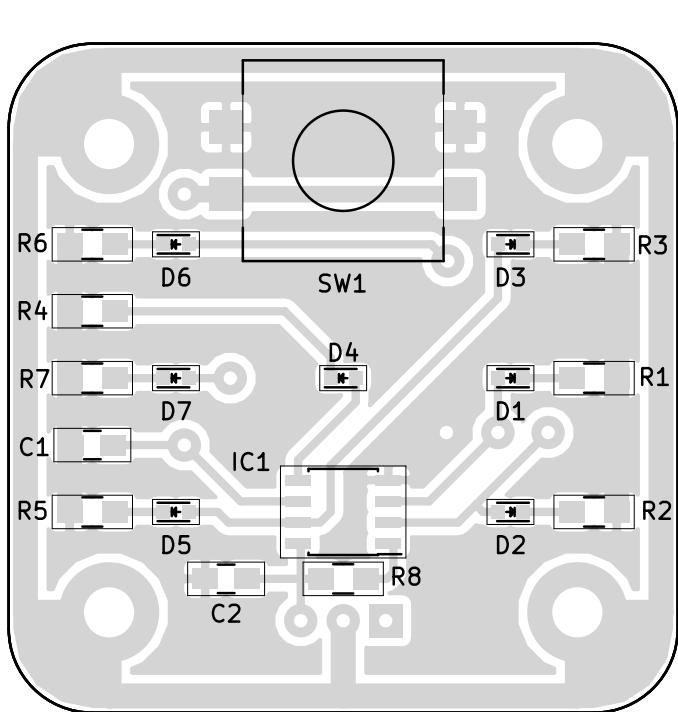
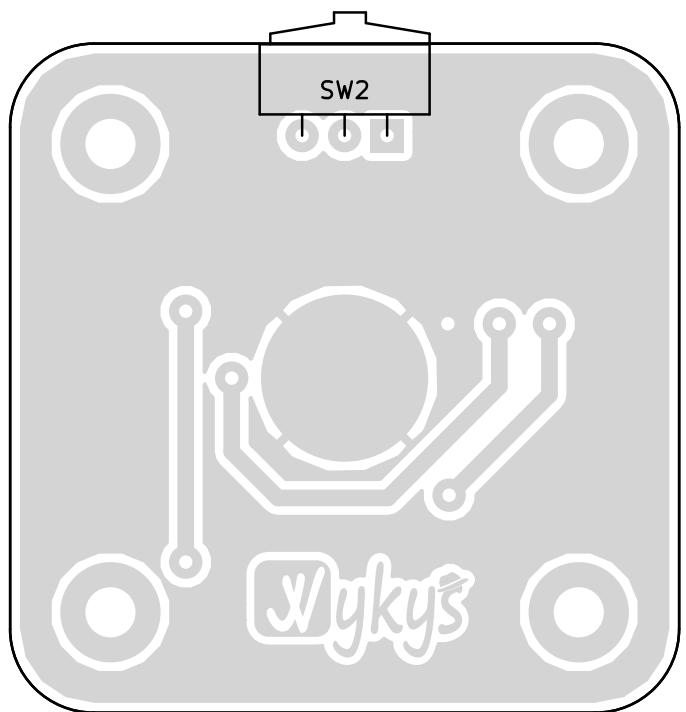


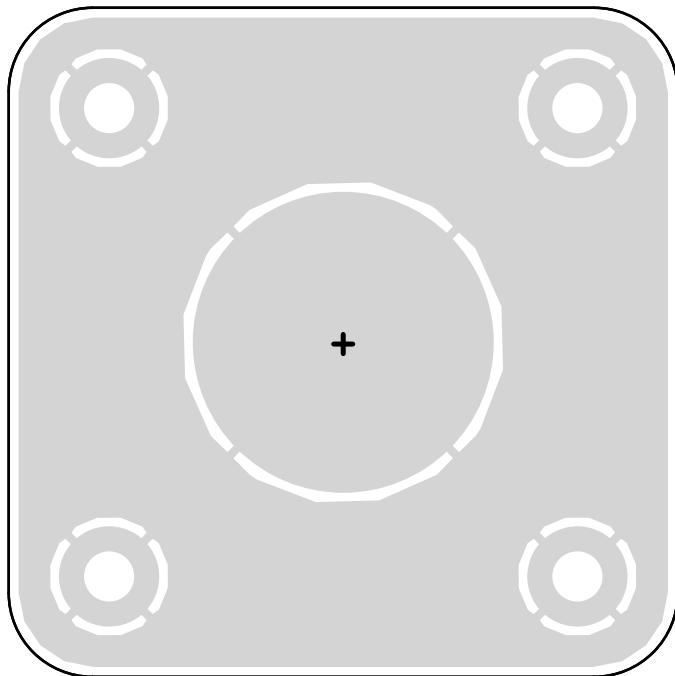
Obrázek 1: Schéma zapojení



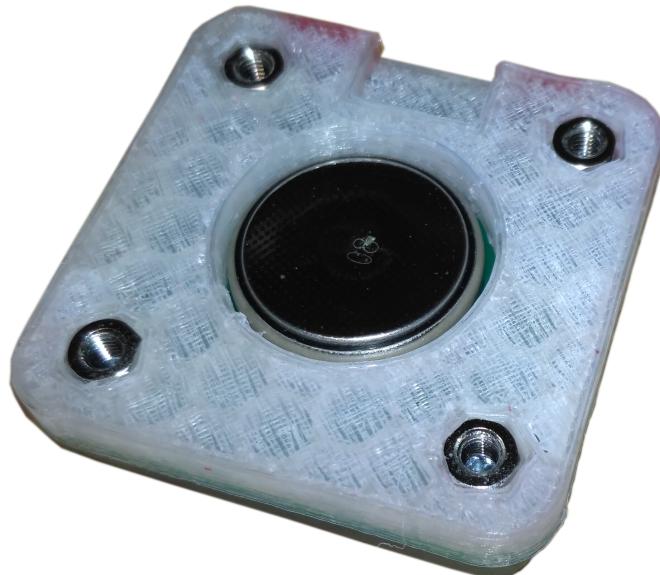
Obrázek 2: Osazovací plán strana součástek



Obrázek 3: Osazovací plán strana spojů



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



Obrázek 5: ukázka uložení distančních sloupek a baterie CR2032

Tabulka 1: Seznam součástek

$R_1 - R_7$	100 $\Omega$	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



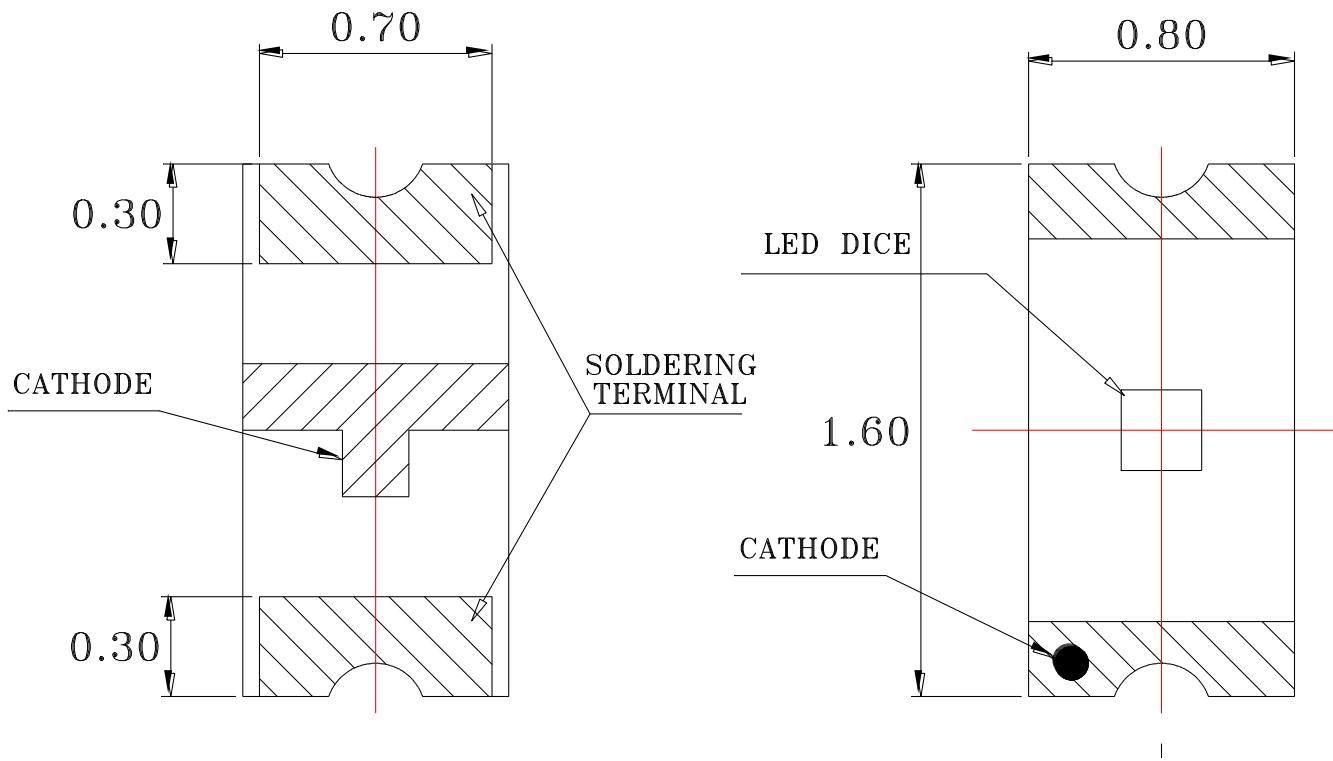
Obrázek 6: smontovaná kostka bez osazených součástek

## 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 LED diody LTW-C190DA5

## Jak to funguje?

Srdcem kostky je maličký 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 First In Last Out) skočí na obslužný podprogram přerušení. Dokud je tlačítko stisknuto, tak je na výstupní piny 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 nejde 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 vstupu RESET. 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 odfiltraci rušení.

## Zdrojový kód

```
1 // firmware pro elektronickou kostku v 1.0 wykys 8.12.2016
2 #include <avr/io.h>
3 #include <avr/interrupt.h>
4
5 #define A (1<<0)
6 #define B (1<<2)
7 #define C (1<<3)
8 #define D (1<<4)
9 #define T (1<<1)
10
11 #define delay() for (t=0; t<300; t++) asm("nop")
12
13 const uint8_t tabulka [] = {
14     T | A,           // 1
15     T | B,           // 2
16     T | A | B,       // 3
17     T | B | C,       // 4
18     T | A | B | C,   // 5
19     T | B | C | D,   // 6
20     T | A | B,       //
21     T | A | C,       // rotace
22     T | A | D       //
23 };
24
25 ISR(INT0_vect)
26 {
27     uint8_t i, cislo;
28     uint16_t t;
29
30     cli();
31     PORTB = T;
32     while (!(PINB & T))
33     {
34         PORTB = tabulka[6];
35         delay();
36         PORTB = tabulka[7];
37         delay();
38         PORTB = tabulka[8];
39         delay();
40     }
41     cislo = tabulka[TCNT0];
42     for (i=0; i<3; i++)
43     {
44         PORTB = T;
45         delay();
46         PORTB = cislo;
47         delay();
48     }
49     sei();
50 }
51
52 int main(void)
53 {
54     DDRB = A | B | C | D; // nastaveni I/O
55     PORTB = T;             // natvaveni pull-up odporu na vstup s tlacitkem
56     TCCR0A = 1<<WGM01;    // nastav citac co CTC modu
57     OCR0A = 5;             // nastav porovnavaci registr
58     TCNT0 = 0x00;           // vynuluj 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     while (1) asm("nop"); // nekonecny cyklus
64 }
```