VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Mikroprocesorové a vestavěné systémy

Měření srdečního tepu

19. prosince 2018

Petr Marek

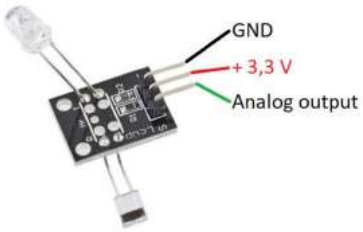
1 Úvod

Cílem tohoto projektu bylo navrhnout a implementovat vestavnou aplikaci v jazyce C pro Kinetis K60. Základní funkcí aplikace je měření frekvence srdečního tepu. Tuto frekvenci (počet tepů za minutu) aplikace zobrazuje na displeji.

2 Detektor srdečního tepu

K získání srdečního tepu aplikace využívá modul s detektorem srdečního tepu. Jeho dokumentace je k dispozici *zde*. Dle této dokumentace funguje na principu průchodu infračerveného světla skrze přiložený prst. Pro správné použití je tedy třeba nasměrovat diodu na přijímací tranzistor a vložit prst do cesty světelného paprsku. Ten je infračervený, tedy pro oko neviditelný (lze ho detekovat například digitálním fotoaparátem). Analogový výstup je zpracován ADC převodníkem. Ten je nastaven na 12-bitový výstup, tedy měřené hodnoty jsou v intervalu 0-4095.

2.1 Zapojení detektoru



Obrázek 1: Detektor

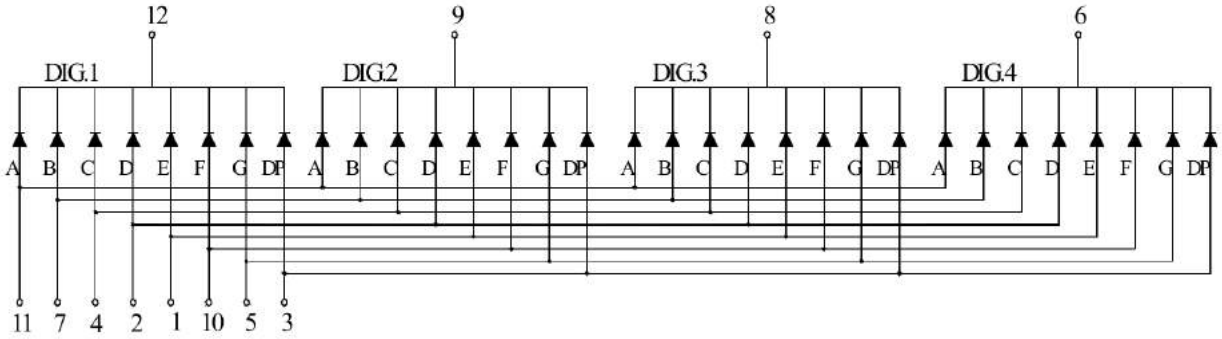
Detektor je zapojen v bloku P1. Přesné rozmístění vývodů na piny (resp. porty) je popsáno v následující tabulce.

Vývod	**Pin**	**Port**
GND	49	GND
3.3V	1	V3.3
Analog output	5	ADC0 SE16

Tabulka 1: Zapojení detektoru

3 Displej

K zobrazení tepu byl použit sedmisegmentový displej. Dokumentace je opět k dispozici *zde*. Jedná se o displej se společným vývodem na katodu. Jeho schéma je na obrázku 2.



Obrázek 2: Schéma Displeje

Ze schéma je patrné, že k rozsvícení segmentu displeje je třeba přivést proud na některý z vývodů 11,7,4,2,1,10,5,3. Na číslici, kterou chceme rozsvítit (vývody 12,9,8,6), přivedeme zem. Prakticky se jedná o čtyři displeje propojené společnou katodou. Schéma na obrázku 2 je převzato z dokumentace. Čísla udávaná u jednotlivých vývodů tak nekorespondují s čísly portů ani pinů v zapojení.

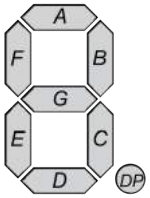
3.1 Zapojení Displeje

K zapojení displeje i senzoru byl použit blok pinů P1. Piny 17-24 ovládají segmenty displeje, piny 25-28 jeho číslice. Mapování pinů na porty je znázorněno tabulkou 2.

Pin	**Port**	**Port**	**Pin**
17	PTD8	PTD9	18
19	PTD12	PTD13	20
21	PTD15	PTD14	22
23	PTA8	PTA10	24
25	PTA6	PTA11	26
27	PTA7	PTA9	28

Tabulka 2: Zapojení pinů

Zapojení jednoho segmentového displeje vypadá následovně:



Obrázek 3: Segmenty

Mapování segmentů na porty:

Segment	**Port**
A	PTD12
B	PTD8
C	PTA8
D	PTD15
E	PTD13
F	PTD9
G	PTA10
DP	PTD14

Tabulka 3: Mapování segmentů na porty

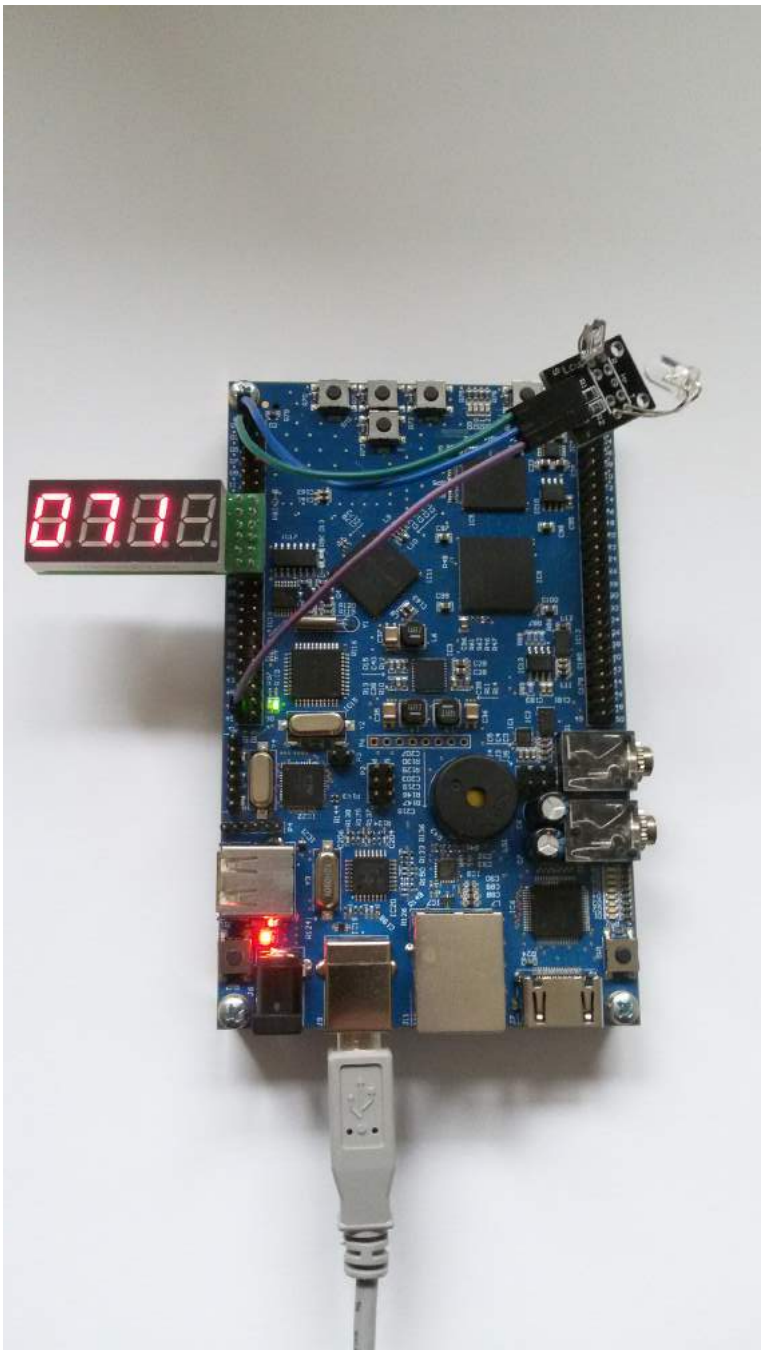
Mapování číslic na porty:

Číslice	**Port**
DIG1	PTA9
DIG2	PTA6
DIG3	PTA11
DIG4	PTD7

Tabulka 4: Mapování číslic na porty

4 Měření

Po zapnutí aplikace budou na displeji svítit tři nuly. Znamená to, že aplikace čeká na spuštění měření. Vložte prst mezi diodu a detektor. Dioda by měla stále mířit co nejpřesněji na detektor. Druhou rukou stiskněte tlačítko v pravém horním rohu displeje s nápisem SW6. Displej přestane svítit a aplikace začne měřit. Pro nejpřesnější měření nehybně vyčkejte půl minuty, než se zobrazí výsledný údaj na displeji. Jinak znovu stiskněte tlačítko SW6 a údaj o tepu se zobrazí dříve. Aplikace spustí nové měření opětovným stisknutím tlačítka. Během zobrazování hodnoty aplikace neměří. Je tedy možné odejmout prst z detektoru.



Obrázek 4: Výsledek měření

5 Hlavní funkce *main*

```c
int main(void)
{
    MCUInit();
    PortsInit();
    ADC0_Init();

    int pressed = 0;      // button is pressed
    int delayMsec = 60;  // delay between two samples (in miliseconds)
    double alpha = 0.75; // average ratio variable
    double oldValue = 0; // previous average value
    int beatMsec = 0;

    save_value(0); // set value to be displayed to 0
    while(1) {
       // display BPM on display
       // on startup display 0
        while(1) {
            index = 0;
            // Display value, changes index variable
            DisplayValue(result);
            // on SW6 button press, initiate new measuring by turning LEDs off
            if (!pressed && !(GPIOE_PDIR & BTN_SW6)) {
                pressed = 1;
                break;
            } else if (GPIOE_PDIR & BTN_SW6) {
                pressed = 0;
            }
        }
        // measure BPM, LEDs are off
        int timer = 30000; // 30 seconds
        while(1) {
            // average 75% from old value and 25% from current value
            double value = alpha * oldValue + (1 - alpha) * currentValue;
            oldValue = value;
            int heartRateBPM = 0;

            if (heartbeatDetected(delayMsec)) {
                // compute BPM from beats per ms (beatMsec)
                heartRateBPM = 60000 / beatMsec;
                save_value(heartRateBPM);
                beatMsec = 0; // restart measuring
            }
            time_delay_ms(delayMsec);
            beatMsec += delayMsec;
            timer -= delayMsec;
            if (timer <= 0) break;

            // on SW6 button press leave measuring
            if (!pressed && !(GPIOE_PDIR & BTN_SW6)) {
                pressed = 1;
                break;
            } else if (GPIOE_PDIR & BTN_SW6) {
                pressed = 0;
            }
        }
    }
    return 0;
}
```

Z funkce *main* je patrné, že program má dva stavy. První je zobrazení výsledku na displeji, druhým je měření. Ze stavu zobrazení do stavu měření se přechází stisknutím tlačítka SW6. Zpět lze přejít buď opětovným stisknutím daného tlačítka, či vyčkáním na časovač *timer*.

Měření BPM funguje na principu měření doby mezi jednotlivými maximálními hodnotami napětí. Tyto hodnoty získáme ze senzoru. K měření času je použit LPTRM modul, který je naprogramován ve funkci *time_delay_ms* tak, aby zajistil čekání mezi dvěma měřeními. Takto přesně víme, jaká je doba mezi dvěma vzorky. Dokud nenarazíme na další maximum, tyto doby sčítáme. Zároveň doby čekání použijeme pro odpočet optimální doby měření.

5.1 Funkce detekce tepu

```
// detect heartbeats
int maxValue = 0;    // current maximum value measured
bool isPeak = false; // peak detected
bool heartbeatDetected(int delay)
{
    int rawValue = 0;
    bool result = false;
    rawValue = currentValue;
    rawValue *= (1000/delay);

    // If sensor shifts, then max is out of whack.
    // Just reset max to a new baseline.
    if (rawValue * 4L < maxValue) {
        maxValue = rawValue * 0.8;
    }

    // Detect new peak
    if (rawValue > maxValue - (1000/delay)) {
        if (rawValue > maxValue) {
            maxValue = rawValue;
        }
        // Only return true once per peak.
        if (isPeak == false) {
            result = true;
        }
        isPeak = true;
    } else if (rawValue < maxValue - (3000/delay)) {
          isPeak = false;
          // Decay max value to adjust to sensor shifting
          // Note that it may take a few seconds to re-detect
          // the signal.
          maxValue-=(1000/delay);
    }
    return result;
}
```

Funkce *heartbeatDetected* zajišťující měření tepu byla převzata z dokumentace detektoru. Tato funkce detekuje srdeční tep na základě aktuální změřené hodnoty v porovnání s maximální naměřenou. Také detekuje vychýlení senzoru a reaguje na něj snížením maximální hodnoty na 80% vychýlené hodnoty. V případě naměření maxima dojde k detekování vrcholu a funkce vrací *true*. Pokud k detekování maxima nedojde, funkce vrací *false* a snižuje maximální hodnotu, graf funkce tepu klesá.

6 Závěr

Přesnost naměřených hodnoty je především závislá na přesnosti a citlivosti detektoru. Dále jakýkoliv nepatrný pohyb prstu může zapříčinit získání zcela nepřesných výsledků. Při měření vyššího tepu je obzvláště obtížné udržet prst zcela nehnutě. Také diodu je třeba namířit přesně na detektor a prst nesmí na detektor příliš tlačit, ani být příliš volný. Nicméně v ideálním případě se chyba měření většinou pohybovala v rozmezí 20%, což vzhledem okolnostem považuji za uspokojivý výsledek.