Vyšší odborná škola a Střední průmyslová škola elektrotechnická Olomouc,
Božetěchova 3

DLOUHODOBÁ MATURITNÍ PRÁCE

Programování mikrokontroléru ATmega

2016 Jan Doležel

Na druhé stránce bude vloženo originální zadání.

Prohlašuji, že jsem seminární práci vypracoval samostatně a všechny prameny jsem uvedl v seznamu použité literatury.

……

jméno a příjmení studenta

Chtěl bych vyslovit poděkování panu Ing. Marku Nožkovi za odborné konzultace a poskytnuté informace.

……

jméno a příjmení studenta

Prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé práce nebo její části se souhlasem školy.

……

jméno a příjmení studenta

OBSAH

Úvod

Mikrokontroléry jsou v dnešní době nedílnou součástí prakticky každého elektronického zařízení. Od elektroniky v letadlech, přes robotiku, až po bezpečnostní systémy je jimi vše řízeno. Jsou využívány k vyhodnocování dat z různých čidel jako například teploměry či senzory pohybu, ovládají hromadnou dopravu a denně ulehčují život velkému množství lidí.

Cílem této práce je propojení mikrokontroléru ATmega16 s počítačem přes sériové rozhraní a dosažení toho, že výsledné zařízení bude možné ovládat pomocí počítačové aplikace. Počítačovou aplikací bude možné měnit údaje na LCD displeji a dále bude také možné počítačovou klávesnicí generovat tóny pomocí „piezo pípače“. K zařízení bude připojen LCD displej, na kterém budou vykreslovány hodiny, běžící text, zapisované znaky.

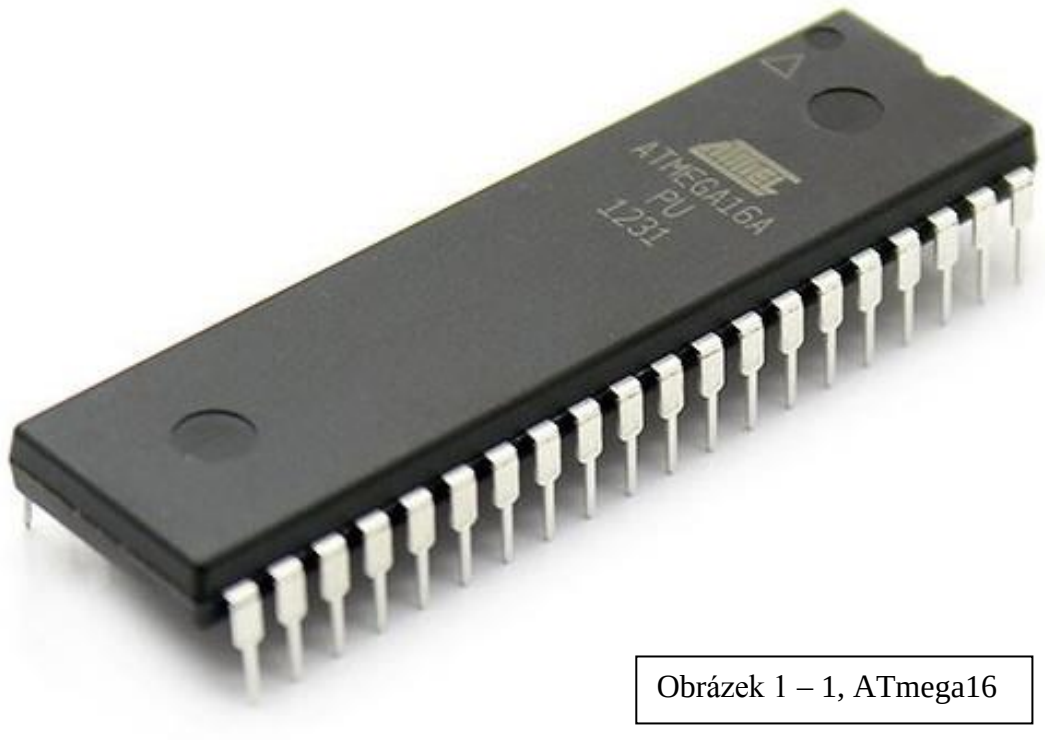
Dalším cílem této práce je zdokonalení se v užívání programovacího jazyka C a Python a zveřejnění postupu výroby zařízení na internetu.

Tato práce s webovou dokumentací by mohla v budoucnu sloužit jako výukový materiál pro zájemce o programování mikrokontroléru, kteří by buď chtěli vytvořit zařízení ovládané počítačem (UART převodník by bylo možné nahradit Bluetooth převodníkem a výsledné zařízení by bylo možné ovládat bezdrátově), nebo zařízení založené na LCD displeji.

1. POUŽITÉ SOUČÁSTKY

1.1. ATMEGA16

ATmega16 je 8 - bitový AVR mikrokontrolér od firmy Atmel. Tento mikrokontrolér je založen na Harvardské architektuře (rozdělena paměť dat a programu) a má omezený instrukční soubor (RISC – *Reduced Instruction Set Computing*).



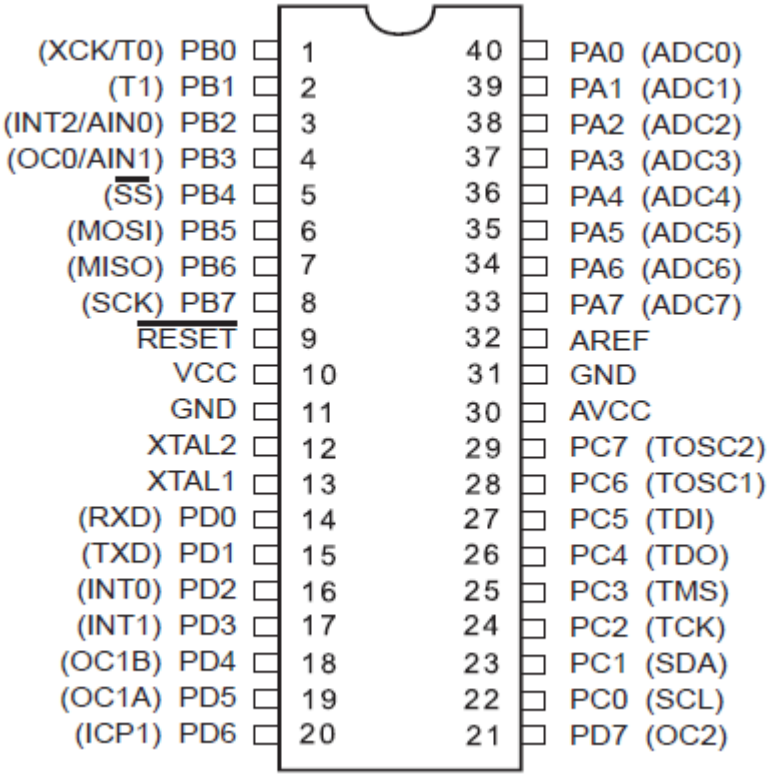
Obrázek 1 – 1, ATmega16

Parametry:

- **Programovatelná flash paměť:** 16 KB
- **EEPROM:** 512 B
- **Vnitřní SRAM:** 1 KB
- **Udržení dat:** 20 let při 85 °C/100 let při 25 °C
- **MIPS (***Milion Instruction Per Second***):** až 16 s 16 MHz oscilátorem
- **Hlavní účelové pracovní registry:** 32 x 8
- **Zápis/Mazání:** 10 000 cyklů Flash Paměť/100 000 cyklů EEPROM
- **Pracovní napětí:** 2.7 – 5.5 V
- **CPU:** 8bit AVR

Periferie:

- 40 pinů, z toho 32 I/O programovatelných
- dva 8 – bitové Čítače/Časovače
- 16 – bitový Čítač/Časovač
- čtyři PWM výstupní kanály
- jednotka USART (rozhraní pro synchronní/asynchronní sériovou komunikaci)
- a další …



Obrázek 1 – 2, Popis výstupních pinů ATmega16

1.2.LCD DISPLAY

Ve své práci jsem využil čtyřřádkový LCD display s řadičem SPLC780, který je dvojčetem řadiče HD44780. Tento řadič obsahuje znakovou sadu, takže není třeba řešit rozsvěcování každého pixelu samostatně. Stačí pouze do LCD displeje zaslat znak a řadič sám rozsvítí pixely v 5 x 8 matici.

Obrázek 1 – 3, LCD display 4 x 20

Parametry:

- **Barva:** zelená + černá
- **Úhlopříčka:** 3.1 “
- **Řádky:** 4
- **Znaků na řádku:** 20
- **Pracovní napětí:** 5 V
- **Podsvícení:** žlutozelené
- **Pracovní teplota:** - 10 °C ~ 60 °C
- **Řadič:** SPLC780/HD44780
- **Skladovací teplota:** - 20 °C ~ 70 °C
- **Rozměry:** 9.8 x 6.0 x 1.2 cm
- **Hmotnost:** 74 g

LCD displej má 16 vstupně/výstupních pinů:

- 1, 16 – GND
- 2, 15 – VCC (+ 5 V)
- 3 – připojen na potenciometr (ovládání jasu displeje)
- 4 – RS pin (rozlišuje, jestli budou znaky uloženy do paměti nebo zobrazeny)
- 5 – RW pin (řídí zápis/čtení displeje)
- 6 – E pin (řídí vzorkování řídících a datových pinů)
- 7, 8, 9, 10 – datové piny potřebné pouze v „*memory mapped mode*“
- 11, 12, 13, 14 – datové piny

Ve své práci jsem využil „*4 – bit IO port mode*“, ve kterém jsou užity pouze 4 datové piny. Byte je tedy rozdělen na dvě poloviny, které jsou odeslány za sebou.

Ovládání LCD displeje

K ovládání LCD displeje slouží knihovna „lcd.h“ vytvořená pro LCD displeje založené na řadiči HD44780. Tato knihovna je „open source“ a jejím autorem je Peter Fleury.

V knihovně „lcd.h“ se nachází definice LCD displeje jako například:

```
#define LCD_LINES           4     //Počet řádků displeje
#define LCD_DISP_LENGTH    20     //Počet znaků na řádek
#define LCD_PORT         PORTD    //Port mikrokontroléru využitý pro LCD
```

Tyto hodnoty si můžeme upravit podle potřeby.

Dále jsou ve knihovně deklarace funkcí definovaných ve zdrojovém souboru „lcd.c“:

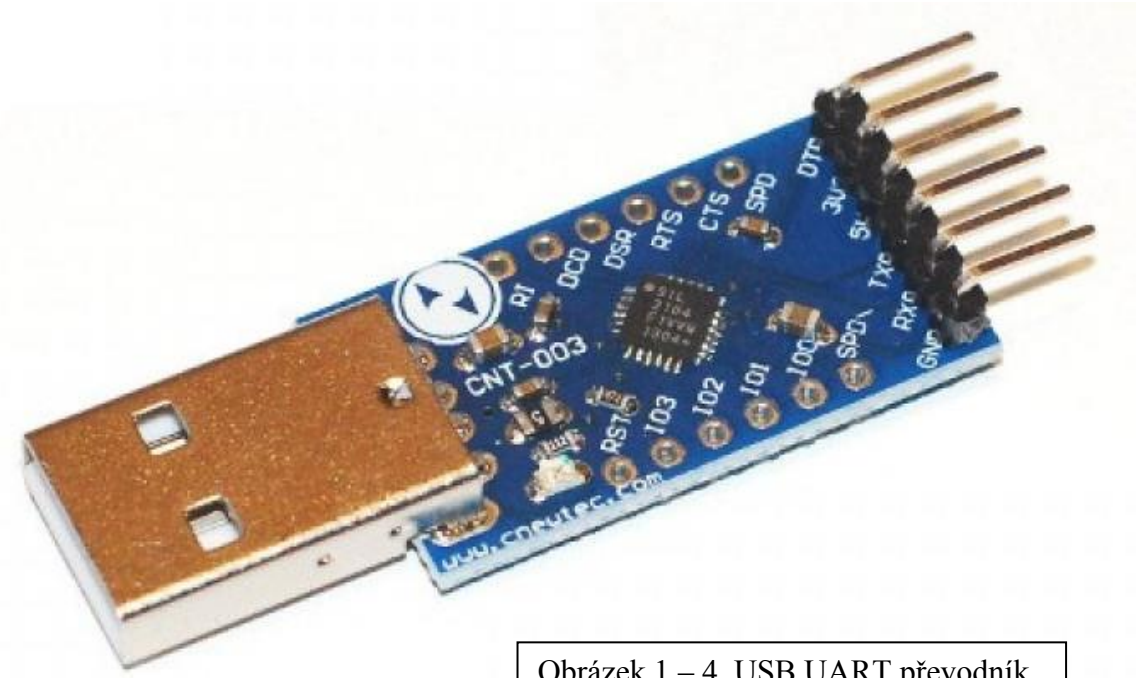
```
extern void lcd_init(uint8_t dispAttr);  // Inicializace displeje
extern void lcd_clrscr(void);     // Vymaže displej
extern void lcd_gotoxy(uint8_t x, uint8_t y);   //Nastavení kurzoru (x, y)
extern void lcd_putc(char c);     //Zapsání znaku
```

Kdybychom například chtěli zapnout displej, nastavit kurzor na určitou pozici a zapsat řetězec, tak to bude vypadat následovně:

```
int main(void)
{
      lcd_init(LCD_DISP_ON);     //aktivace displeje
      lcd_clrscr();              //vyčistí displej
      lcd_gotoxy(x, y);          //nastaví kurzor na souřadnice x, y
      lcd_puts("Hello World");   //zapíše na displej konstantní řetězec

      return 0;
}
```

1.3.UART PŘEVODNÍK

Využil jsem UART TTL převodník CP2104. Tento USB převodník slouží jako sériový port RS232/RS485 v asynchronním režimu. Pin RXD (*receive*) UART převodníku spojíme s pinem TXD (*transmit*) u mikrokontroléru ATmega16 a ekvivalentně k tomu spojíme pin RXD mikrokontroléru ATmega16 s pinem TXD UART převodníku. Dále už stačí jen pin 5V připojit na napájení a pin GND uzemnit.



Obrázek 1 – 4, USB UART převodník

Parametry:

- Typ USB: standardní USB typ A
- Počet pinů: 6 (DTR, 3V3, 5V, TXD, RXD, GND)
- Přenosová rychlost: 300 bps až 2 Mbps (bps = bites per second)
- Teplotní rozsah: - 40 až + 80 °C
- Pracovní napětí: 3.3 V až 5 V
- 1 B přijímací buffer, 576 B odesílací buffer

1.4. Piezo bzučák

Obrázek 1 – 5, Piezo bzučák

Piezo bzučák je elektronická součástka, která při připojení na napětí začne generovat zvuk.

Princip vzniku tónu

Tón o určité frekvenci je generován tak, že po dobu půlperiody tónu bude logická úroveň zařízení 1 a po druhou polovinu půlperiody bude logická úroveň zařízení 0.

Půl periodu vypočítáme takto:

$$\frac{T}{2} = \frac{\frac{1}{f}}{2}$$

Tedy hodnotu půlperiody tónu C1 o frekvenci f = 262 Hz vypočítáme takto:

$$\frac{T}{2} = \frac{\frac{1}{262}}{2} = 0.001908\,s = 1908\,\mu s$$

V programovacím jazyce C bude funkce na generování tónu C1 vypadat následovně:

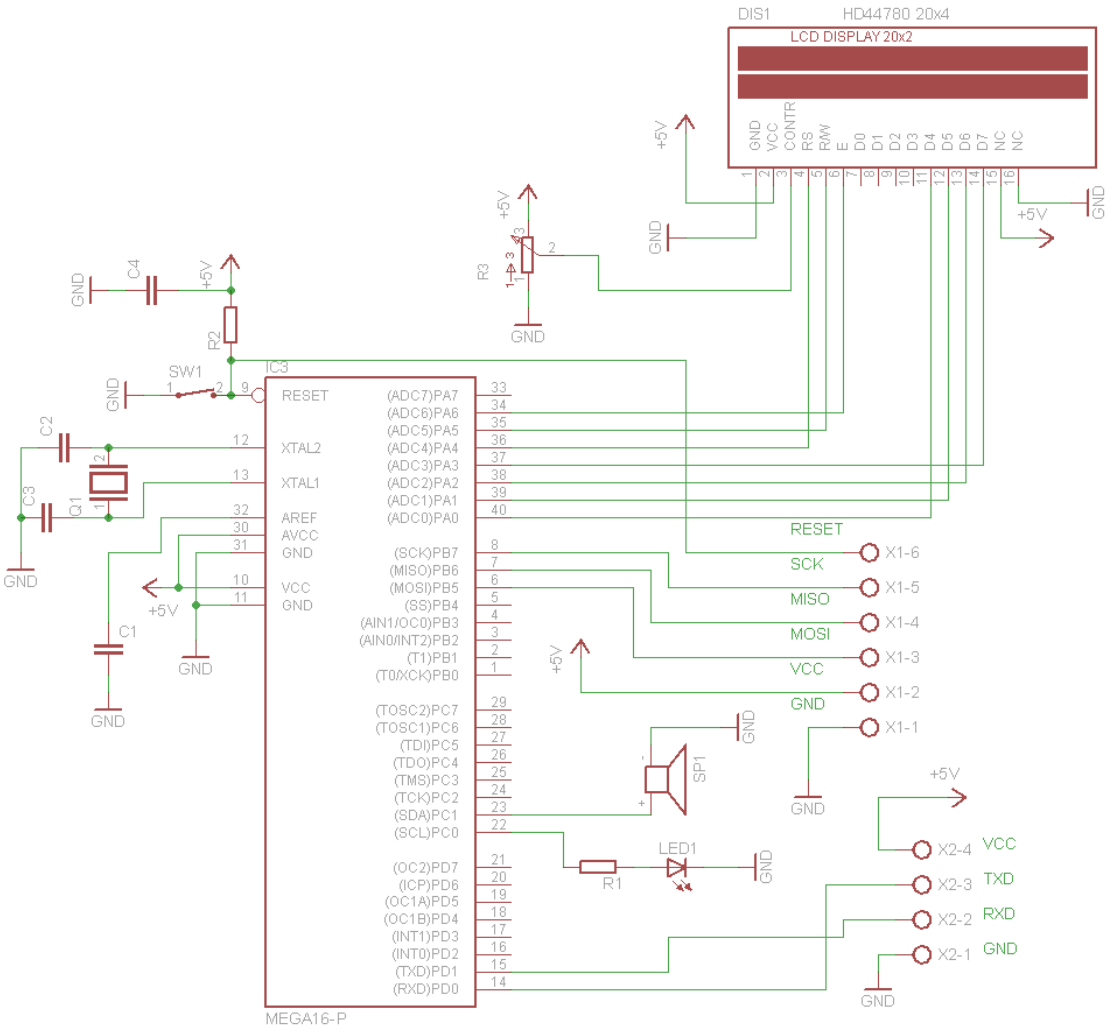
```
#define F_CPU 16000000UL    //FREKVENCE externího oscilátoru (16 Mhz)
#include <util/delay.h>     //knihovna k práci s prodlevou _delay_ms()


void C1(void);       //deklarace funkce C1


int main(void)
{
       DDRC |= 0b00000001;  //nastaví pin 0 portu C jako výstupní


    while (1)
    {
             C1();  //volání funkce C1

    }
       return 0;
}


void C1(void)        //definice funkce C1
{
       for (int i = 0;i < 52;i++)        //délka tónu 0.2 s
       {
             PORTC |= 0b00000010;        //log. 1 na vstup piezo bzučáku
             _delay_us(1908);            //prodleva o délce půlperiody C1
             PORTC &= 0b11111101;        //log. 0 na vstup piezo bzučáku
             _delay_us(1908);            //prodleva o délce půlperiody C1

       }
}
```

Frekvence a půlperiody tónu stupnice C dur

Tón	Frekvence [Hz]	Půlperioda [µs]
C1	262	1908
D1	294	1700
E1	330	1515
F1	349	1432
G1	392	1275
A1	440	1136
H1	494	1012
C2	523	956

Tabulka 1, Stupnice C dur

1.5. SCHÉMA ZAPOJENÍ



Obrázek 1 – 6, Schéma zapojení zařízení

Popis součástek:

- DIS1, HD44780 20 x 4 – LCD displej 20 x 4
- IC3, MEGA16-P – ATmega16
- x1 - (1 – 6) – usbasp programátor
- x2 – (1 – 4) – UART převodník
- SP1 – „piezo bzučák“
- LED1 – LED dioda
- R1 – 100 Ω rezistor
- R2 – 10 KΩ rezistor
- R3 – 10 KΩ potenciometr
- C1, C4 – 100 nF kondenzátor
- C2, C3 – 22 pF kondenzátor
- Q1 – 16 MHz krystal
- SW1 - spínač

2. ČASOVAČ, PŘERUŠENÍ A JEDNOTKA USART

2.1. ČASOVAČ V NORMÁLNÍM REŽIMU

Časovač slouží, jak je z názvu patrné, k práci s časem, časovými intervaly. Jeho využití si můžeme představit třeba na funkci hodin. Kdybychom pracovali s prodlevou „_delay_ms(1000)“ a po každém intervalu bychom provedly přičtení a přepočítání času, tak by mikrokontrolér celou sekundu nedělal nic jiného, než že by počítal prodlevu a nebylo by tedy možné provádět jakékoli další operace. K vyhnutí se těmto problémům nám slouží právě zmíněný časovač. Mikrokontrolér ATmega16 je vybaven dvěma 8 – bitovými a jedním 16 – bitovým časovačem. Funkce časovače v normálním režimu je poměrně jednoduchá. Časovač do svého registru přičte 1 při každém strojovém cyklu mikrokontroléru a po dosažení maximální hodnoty registru tento registr vynuluje. Přetečení je jedna z možností, která umožňuje časovači vyvolat přerušení. Časovače mikrokontroléru ATmega16 jsou vybaveny 10 bitovým děličem hodinového taktu, který nám umožňuje snížit frekvenci přičítání do registru, čímž dosáhneme toho, že naplnění registru bude trvat déle.

Práce s časovačem v normálním režimu

Výpočet délky naplnění registru bude vypadat následovně:

$$\frac{1}{F\_CPU/dělič} \times velikost\ registru$$

- F_CPU = hodinový takt mikrokontroléru (16 MHz externí oscilátor)
- dělič = dělič hodinového taktu mikrokontroléru (může nabývat hodnot 1, 8, 64, 256, 1024)
- velikost registru = 256 pro 8 – bitový registr a 65 536 pro 16 – bitový registr

Pro ukázku využiji prodlevu 1 s potřebnou k funkci hodin:

$$\frac{1}{16\,000\,000/256} \times 65\,536 = 1.048576\ přetečení\ registru/s$$

Abychom vykompenzovali 50 ms nepřesnost při každé sekundě, tak do registru po jeho přetečení nastavíme hodnotu 3036 a tím dostaneme:

$$\frac{1}{16\,000\,000/256} \times (65\,536 - 3036) = 1\ přetečení\ registru/s$$

Časovač může pracovat i v jiných režimech, než v normálním:

- CTE – reagující na shodu
- PWM – rychlý režim
- PWM – fázově korigovaný režim

K nastavení časovače nám slouží tyto registry:

- TIMSK – povoluje přerušení po přetečení datového registru časovače
- TCCR – určuje režim časovače a dělič hodinového taktu mikrokontroléru

2.2. PŘERUŠENÍ

Přerušení nám u mikrokontrolérů umožňuje měnit běh hlavního programu a při námi určených podmínkách hlavní program přerušit a provést požadovaný kód. Funkci přerušení si můžeme představit například na nějakém elektronickém zařízení, jako je například MP3/MP4 přehrávač. Přehrávač má tlačítka zastavit, zeslabit, zesílit a další. Kdybychom využili klasické metody ošetření stisknuté klávesy, při níž se stále dokola v hlavním programu zjišťuje, jestli je tlačítko stisknuté, tak bychom zpožďovali námi přehrávanou hudbu, což je uživatelsky nepřípustné. Právě k zabránění tomuto zpoždění se využívají přerušení. Přerušení mohou být externí, nebo je mohou vyvolat samotné periferie mikrokontroléru.

U mikrokontroléru ATmega16 můžeme vyvolat externí přerušení piny INT0 (PD2), INT1 (PD3), INT2 (PB2). Externí přerušení na těchto pinech může reagovat na:

- logickou 0 na pinu
- změna logické hodnoty na pinu
- sestupná hrana
- náběžná hrana

Mimo externí přerušení můžeme přerušení vyvolat samotným mikrokontrolérem. Přerušení může být způsobeno časovačem, jednotkou USART, A/D převodníkem atd.

Ve své práci jsem využil přerušení způsobené:

- přetečením 8/16 bitového registru časovače
- kompletním příjmem dat jednotkou USART

V programovacím jazyku C slouží k práci s přerušeními knihovna <avr/interrupt.h>. Definice přerušení se provádí následovně:

```
ISR(vektor přerušení)
{
	//námi požadovaný kód, který se provede při vyvolání přerušení
}
```

Vektory přerušení v jazyce C

	Adresa	Parametr v C (GCC)	Popis přerušení
1	0x0000		Externí reset, připojení napájení
2	0x0002	INT0_vect	Externí požadavek na přerušení 0
3	0x0004	INT1_vect	Externí požadavek na přerušení 1
4	0x0006	TIMER2_COMP_vect	Čítač/časovač 2 – shoda komparace
5	0x0008	TIMER2_OVF_vect	Čítač/časovač 2 – přetečení
6	0x000A	TIMER1_CAPT_vect	Čítač/časovač 1 – zachycení
7	0x000C	TIMER1_COMPA_vect	Čítač/časovač 1 – shoda s komparátorem A
8	0x000E	TIMER1_COMPB_vect	Čítač/časovač 1 – shoda s komparátorem B
9	0x0010	TIMER1_OVF_vect	Čítač/časovač 1 – přetečení
10	0x0012	TIMER0_OVF_vect	Čítač/časovač 0 – přetečení
11	0x0014	SPI_STC_vect	Dokončení sériového přenosu SPI
12	0x0016	USART_RXC_vect	USART – kompletní příjem dat
13	0x0018	USART_UDRE_vect	USART – prázdný datový registr
14	0x001A	USART_TXC_vect	USART – kompletní vyslání dat
15	0x001C	ADCvect	ADC – dokončení A/D převodu
16	0x001E	EE_RDY_vect	EEPROM – komunikace připravena
17	0x0020	ANA_COMP_vect	Změna výstupu analogového komparátoru
18	0x0022	TWLvect	Událost na I2C sběrnici
19	0x0024	INT2_vect	Externí požadavek na přerušení 2
20	0x0026	TIMER0_COMP_vect	Čítač/časovač 0 – shoda komparace
21	0x0028	SPM_RDY_vect	Uložení do programové paměti připraveno

Tabulka 2, Vektory přerušení v programovacím jazyce C

2.3. JEDNOTKA USART

(*Universal Synchronous, Asynchronous seriál Receiver and Transmitter*)

Jedná se o zařízení užívané k sériové komunikaci. Toto zařízení můžeme používat ve dvou režimech:

- Asynchronní režim (SCI – například linky RS232/RS485)
- Synchronní režim (SPI)

Jednotkou USART je vybavena většina mikroprocesorů firmy Atmel.

Jednotka USART je tvořena 3 částmi:

- Generátor hodin (potřebný pouze v synchronním režimu)
- Přijímač (posuvný sériový registr, detektor parity, dvouúrovňový buffer)
- Vysílač (posuvný sériový registr, buffer, generátor parity, …)

Asynchronní režim

Jednotka USART vysílá data přes pin označený jako TX (*transmit*), což je u mikrokontroléru ATmega16 pin PD1 (TXD). Data jsou přijímána na pinu RX (*receive*), což je u ATmega16 pin PD0 (RXD). Klidová hodnota signálu, pokud neprobíhá příjem ani odeslání dat je logická 1. Přenos byte je zahájen tzv. „start“ bitem, při kterém dojde ke změně logické úrovně na logickou 0. Poté jsou odeslány datové bity v pořadí od nejnižšího po nejvyšší. Po zaslání nejvyššího datového bitu je zaslán tzv. „stop“ bit, který má logickou úroveň 1. Po odeslání „stop“ bitu může být zahájen přenos dalšího datového byte.

Vysílač

Po načtení datových bitů do registru UDR jsou tyto bity poslány do posuvného sériového registru, kde jim je přiřazen „start“ a „stop“ bit a následně jsou tyto data odeslány přes TX pin. V registru UDR může být načten pouze jeden datový byte, a proto nám speciální posuvný registr umožňuje, abychom načítali do registru UDR další datový byte již během odesílání minulého.

Přijímač

Po zjištění „start“ bitu na pinu RX se další datové bity načtou do posuvného registru. Jakmile je detekován „stop“ bit, tak jsou data odeslána do bufferu, ze kterého jsou poté předány registru UDR za předpokladu, že je prázdný. Buffer i registr UDR jsou elementy FIFO (*First In First Out*). Využitím dvojúrovňového bufferu dosáhneme toho, že i když ještě nejsou data z registru UDR přečtena, tak nebudou přepsána novými daty uchovávanými v bufferu.

Registry jednotky USART

K nastavení jednotky USART slouží u mikrokontroléru ATmega16 4 registry:

- UCSRA – obsahuje komunikační příznakové bity o chybách (chyba rámce, ztráta dat, nekorespondující parita)
- UCSRB – slouží k nastavení jednotlivých přerušení (viz. Tabulka 1)
- UCSRC – nastavuje režim komunikace (synchronní/asynchronní režim, počet datových bitů, parita, počet „stop“ bitů – 1, 2)
- UBRR (UBRRH, UBRRL) – definuje přenosovou rychlost

UDR

Registr UDR je datový buffer (vyrovnávací paměť – paměť pro dočasné uložení dat do té doby, než budou přesunuty na jiné místo), který je společný pro vysílač i přijímač. Čtením registru UDR získáme data z bufferu přijímače. Zápisem do registru UDR načteme data do bufferu vysílače.

```
UDR = data;   //zapsání dat do bufferu registru UDR
data = UDR;   //čtení dat z bufferu registru UDR
```

3. POČÍTAČOVÁ APLIKACE

Počítačovou aplikaci sloužící k ovládání elektronického zařízení přes sériový port jsem vytvořil v programovacím jazyku Python s pomocí grafické knihovny Tkinter a knihovny Pyseril, která slouží k ovládání sériového portu.

3.1. TKINTER

Tkinter je knihovna fungující pod programovacím jazykem Python. Tato knihovna slouží k vykreslování oken. Tkinter funguje na Windows i na Linuxu (za předpokladu, že je nainstalován Python)

Základní práce s knihovnou Tkinter

```
from Tkinter import *   #importuje knihovnu Tkinter do našeho programu
hlavni_okno = Tk()      #vytvoří okno

hlavni_okno.title(u"Ovládání zařízení")              #nastavení nadpisu
hlavni_okno.minsize(width = 400, height = 300)       #velikost okna
hlavni_okno.maxsize(width = 400, height = 300)       #velikost okna

mainloop()      #vykreslí hlavni_okno
```

Dále můžeme vytvářet různé komponenty s odlišnými parametry a umísťovat je do okna:

- Tlačítko (Button)
- Vstup (Entry)
- Pole (Label)
- Menu (Menu)
- a mnoho dalších

Komponenty musí být do pole umístěny jednou ze tří metod:

- Grid – umístí komponentu na určitý řádek a sloupec
- Pack – umísťuje komponenty za sebe
- Place – umísťuje komponenty na pevnou/relativní pozici

3.2. PYSERIAL

Knihovna Pyserial je modul, který umožňuje přístup programu k sériovým portům. Tato knihovna funguje pod programovacím jazykem Python na operačních systémech jako Windows a Linux. Knihovna Pyserial nám umožňuje přijímat a odesílat data přes sériový port s mnoha parametry jako například:

- různá délka datového byte (5 – 8 datových bitů)
- zasílání „start“ a „stop“ bitů
- podpora paritního bitu
- přijímaní dat s prodlevou a další

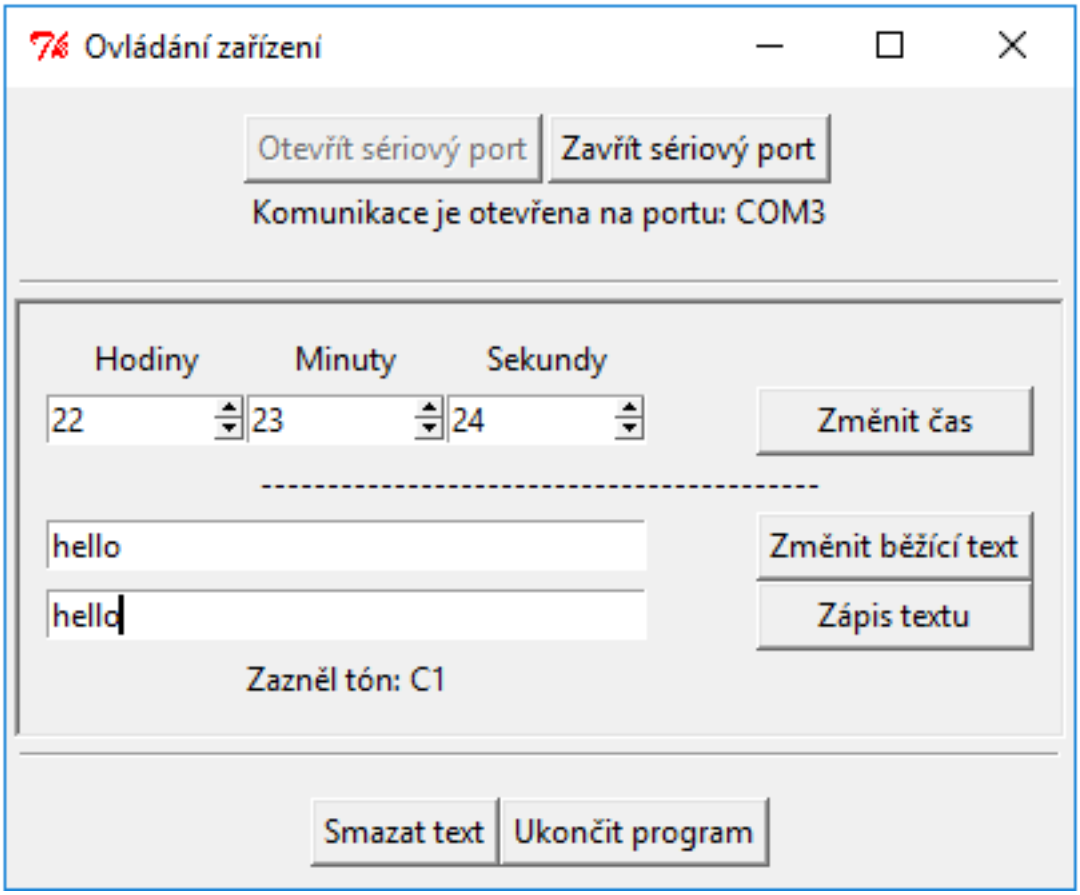
Základní práce s knihovnou Pyserial

```
import serial           #importuje knihovnu Pyserial
ser = serial.Serial(port='COM3', baudrate=9600, timeout=1, bytesize=8,
parity='N', stopbits=1, xonxoff=0, rtscts=0)
#otevře sériový port s danými parametry ('COM ...' pro windows, '/dev/tty ...'
pro Linux)

print(ser.name)         #vypíše jméno portu

ser.write("Hello")      #odešle řetězec "Hello"
ser.read(100)           #přečtě až 100 byte z bufferu přijímače
ser.close()             #zavře sériový port "ser"
```

4. OVLÁDÁNÍ A FUNKCE ZAŘÍZENÍ

4.1. OVLÁDÁNÍ

Elektronické zařízení je možné ovládat pomocí grafického rozhraní přes počítač. Po odpojení od počítače je zařízení stále funkční.



Obrázek 4 – 1, Počítačová aplikace

Pomocí počítačové aplikace je možné měnit digitální čas funkce hodin, nastavovat běžící text, zapisovat/mazat text, otevírat a zavírat sériový port, ke kterému je připojen UART převodník a pomocí kláves F1 – F8 generovat tóny stupnice C dur.

4.2. Funkce

Zařízení plní funkci digitálních hodin ve formátu 00:00:00, dále zařízení zobrazuje běžící text ve frekvenci cca 0.5/s. Na první řádek LCD displeje je možné zapisovat znaky a poté znaky na tomto řádku smazat. Po stisknutí klávesy F1 – F8 je generován tón a při jeho generování je z mikrokontroléru ATmega odeslána informace o který tón se jedná, které se poté na krátký okamžik objeví v počítačové aplikaci.

Obrázek 4 – 2, Funkce na LCD displeji

ZÁVĚR

Mým úkolem bylo vytvořit zařízení s mikrokontrolérem ATmega, které bude schopné komunikovat s počítačem. Výsledku jsem nakonec dosáhl, ale měl jsem s propojením mikrokontroléru několik problémů, které se mi ale nakonec povedlo vyřešit.

Poměrně náročné bylo vyřešit přenos dat mezi mikrokontrolérem a počítačem. Interpretace přijatých dat byla určena stovkami řádků kódu a chvíli mi trvalo vše vychytat tak, aby vše fungovalo, jak mělo.

S vykreslováním znaků na LCD displej jsem měl zpočátku také několik problémů. Kombinace toho, že jsem využil špatný parametr inicializace displeje a toho, že jsem měl mírně snížený jas displeje, vedlo k tomu, že jsem velmi dlouho strávil nad jeho zprovozněním.

Samotné zapojení na nepájivém poli proběhlo úspěšně, až na několik drátů, které nebyli zkratovány, ale to jsem velmi rychle zjistil a dráty nahradil funkčními.

Dokumentace výroby je zveřejněna na Wiki knihách (viz. Seznam použité literatury)

Když to shrnu, tak i přes mnohé překážky je výrobek prakticky plně funkční.

Z mého pohledu pro mě byla tato práce velmi přínosná. Procvičil jsem si knihovnu Tkinter a programovací jazyk Python, což využiji ke stále se přibližujícím maturitním zkouškám. Zdokonalil jsem se v syntaxi programovacího jazyku C, což pro mě bude velmi užitečné, když budu pokračovat ve studiu na vysoké škole, kde se tento programovací jazyk hojně využívá. Také jsem se zdokonalil v práci s mikroprocesorovou technikou, což považují také za velmi přínosné.

SEZNAM POUŽITÉ LITERATURY A STUDIJNÍCH MATERIÁLŮ

1) http://www.tajned.cz/
2) http://homepage.hispeed.ch/peterfleury/avr-software.html#libs
3) http://www.dx.com/p/3-1-2004a-20x4-character-lcd-module-display-168593#.VuAeAvnhDIU
4) http://www.atmel.com/images/doc2466.pdf
5) https://cs.wikipedia.org/wiki/USART
6) http://programujte.com/clanek/2006111611-avr-usart/
7) http://programujte.com/clanek/2006091410-avr-citace/
8) http://www.edunet.souepl.cz/valecka/at_mega_32_popis%20(Opraveno).htm
9) http://maxembedded.com/2011/06/avr-timers-timer1/
10) http://www.asciitable.com/
11) https://www.fabtolab.com/cp2104-USB-TTL
12) http://tkinter.programujte.com/
13) https://www.interval.cz/podklady/1999-2008/bittnerova/866/tony.html

Technická dokumentace

https://cs.wikibooks.org/wiki/Propojen%C3%AD_mikrokontrol%C3%A9ru_ATmega_s_po%C4%8D%C3%ADta%C4%8Dem

www.wikibooks.org – pod tagem: „Propojení mikrokontroléru ATmega s počítačem“

Citace

Obrázek 1 – 1: ATMEGA16A-PU AVR Mega16 DIP-40 Microcontroller. In: fasttech [online]. © 2012-2015 FastTech.com. [vid. 24.3.2016] [cit. 2016-03-24]. Dostupné z: https://www.fasttech.com/product/1453900-atmega16a-pu-avr-mega16-dip-40-microcontroller

Obrázek 1 – 2: ATmega16 Pin configuration. In: robotix - technology robotix society [online]. © 2015 Technology Robotix Society, IIT Kharagpur. [vid. 24.3.2016] [cit. 2016-03-24]. Dostupné z: https://www.robotix.in/tutorial/avr/avrbasics/

Obrázek 1 – 3: 3.1" 2004A 20x4 Character LCD Module Display. In: dealextreme [online]. © 2006~2016 DX.com. All rights reserved. [vid. 24.3.2016] [cit. 2016-03-24]. Dostupné z: http://www.dx.com/p/3-1-2004a-20x4-character-lcd-module-display-168593#.VvPNO_nhDIU

Obrázek 1 – 4: CP2104 USB to TTL Converter. In: fabtolab [online]. Fab.to.Lab. [vid. 24.3.2016] [cit. 2016-03-24]. Dostupné z: https://www.fabtolab.com/cp2104-USB-TTL

Obrázek 1 – 5: Subminiature Piezo Sounder. In: jpr electronics [online]. Copyright © 2010 JPR Electronics. [vid. 2.4.2016] [cit. 2016-04-02]. Dostupné z: https://www.jprelec.co.uk/store.asp/c=940/Subminiature-Piezo-Sounder

Tabulka 2 [převzato z]: [1] FRÝZA, Tomáš., FEDRA, Zbyněk., ŠEBESTA, Jiří. Mikroprocesorová technika. Počítačová cvičení. Elektronické skriptum. Brno: FEKT VUT v Brně, 2009.

SEZNAM OBRÁZKŮ, GRAFŮ A TABULEK