Středoškolská technika 2017

Setkání a prezentace prací středoškolských studentů na ČVUT

Měření času pomocí mikrokontroléru ATMega328

Vojtěch Dočkal

Střední škola průmyslová, technická a automobilní Jihlava
tř. Legionářů 3, 58601 Jihlava

STŘEDNÍ ŠKOLA PRŮMYSLOVÁ, TECHNICKÁ A AUTOMOBILNÍ JIHLAVA

Měření času pomocí mikrokontroléru

Jihlava 2016/17

Autor Vojtěch Dočkal

Obsah

Seznam obrázků

Seznam tabulek

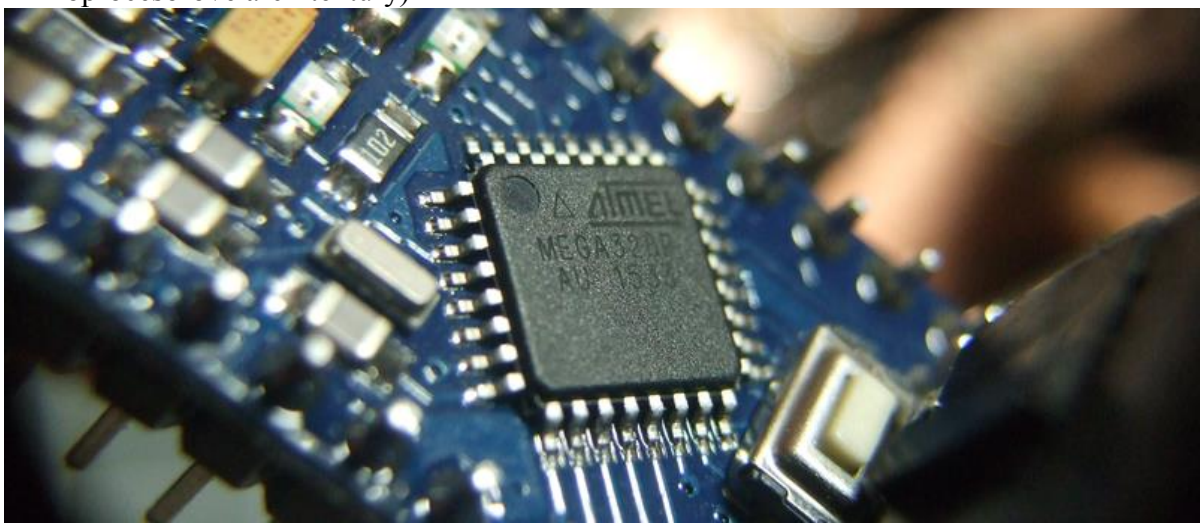
Úvod

Tématem mojí práce je měření času pomocí mikrokontroléru. Hlavní motivací mi byla již nějakou dobu touha po stavbě digitálních hodin s nějakým jiným typen zobrazovače než LED nebo LCD, ale také problematika spojená s konstrukcí a návrhem mi přišla zajímavá.

Pro zvládnutí výše uvedené problematiky jsem byl nucen nastudovat poměrně velký objem dokumentace, jejíž shrnutí jsem umístil do úvodní části práce. Tento text si klade za cíl umožnit dalším studentům, kteří se budou snažit proniknout do této problematiky lepe pochopit podstatu mé práce. Pro čtenáře, který je obeznámen s touto problematikou je studium těchto pasáží zbytečné a doporučuji začít práci číst až od strany 36 kapitoly „Vzorový výrobek“

O procesoru ATmega328P

Procesor ATmega328P je vyráběn firmou Atmel, předním výrobcem mikrokontrolérů a polovodičů pro dotykovou technologii. Tento 8bitový procesor spadá do rodiny 8 a 32bitových mikročipů typu RISC s harvardskou architekturou od firmy Atmel, označenou jako AVR. Díky technologii CMOS si při nízké spotřebě zachová vysoký výkon.
(RISC = Reduced Instruction Set Computing/redukovaná instrukční sada; typ mikroprocesorové architektury)



Obrázek 1 Procesor ATmega328P v SMD provedení na desce Arduino Mini

Platforma Arduino

Arduino je open-source elektronická platforma jednodeskového počítače, vyvinutá za účelem jednoduchého vývoje a realizace prototypů. Tato jednoduchost spočívá ve snadném připojení součástek a periferií k deskám, absenci potřeby podpůrných obvodů, jako napájení, programátoru nebo taktovacích krystalů. Takže stačí Arduino připojit k zapojení, pomocí USB kabelu vše připojit k počítači (tím je obstarána komunikace s počítačem i napájení), nahrát program a může se začít. Ale také v programovacím vývojovém programu Arduino IDE, který je jednoduché a přehledné, díky tomu, že vychází z prostředí Processing, ve kterém píšeme program v jazyce „nečekaně“ pojmenovaném Arduino, založeném na jazyce Wiring.
Právě procesor Atmel ATmega328P je základem této platformy a osazován do většiny desek.

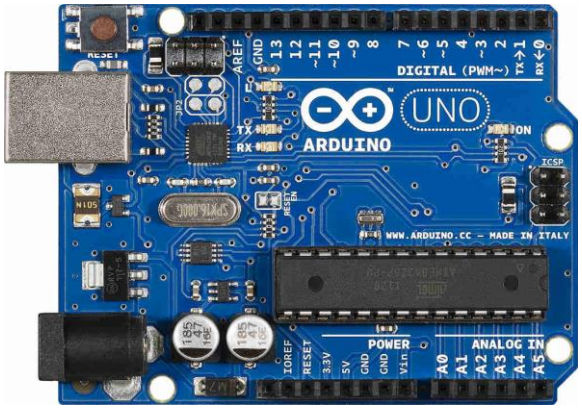
Hardware – desky, shieldy, periferie,…

Desek je několik druhů a liší se podle počtu vstupů a výstupů, ať už digitálních či analogových, podle velikosti pamětí, taktovací frekvenci procesoru, rozměrů a dalších funkcí, jako například Bluetooth, akcelerometr nebo Ethernet připojení. Asi nejvýraznější rozdělení je podle velikosti desky, mými slovy na vývojové a aplikační.

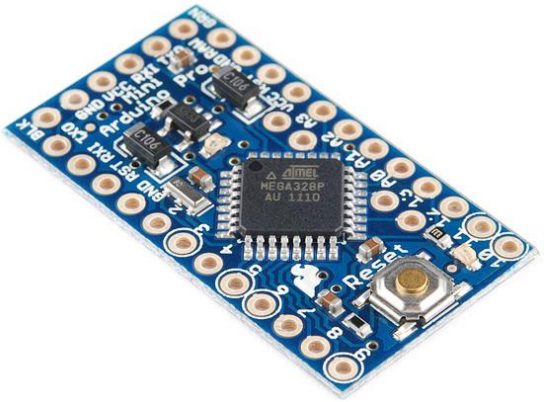
- Vývojové desky jsou pro pohodlnější manipulaci větší, ty nejmenší z nich jako například UNO mají 68x53mm, mají dutinkové lišty, kvůli jednoduchému

připojení periferií (ty mají protikusy – pinové pásy). Tím pádem jsou vhodnější pro vývoj a pokusy, protože v kombinaci s nepájivým polem jde vše snadno upravit nebo úplně předělat.

- Aplikační desky jsou v porovnání s těmi vývojovými dosti kompaktnější a jako konektory mají připravené pinové pásy pro zapájení nebo zasazení do jiné desky jako řídící člen. Také v sobě většinou nemají zabudovaný programátor, až na výjimky jako Nano a některé modely vyráběné ve variantě s USB připojením. Je to z toho důvodu, že vynecháním programátoru se o něco sníží cena i velikost, což se hodí, pokud například nějaké zapojení realizujeme víckrát. Stačí nám jeden programátor, kterým každé Arduino naprogramujeme zvlášť a je zbytečné, aby každé mělo vlastní. Samozřejmě, tyto miniaturní modely nemají takové funkce jako třeba Arduino 101 nebo výkon a množství pinů jako Arduino Mega nebo Due, ale pokud jde o ne tolik náročné zapojení, kde hlavním požadavkem je kompaktnost, tak není co řešit.

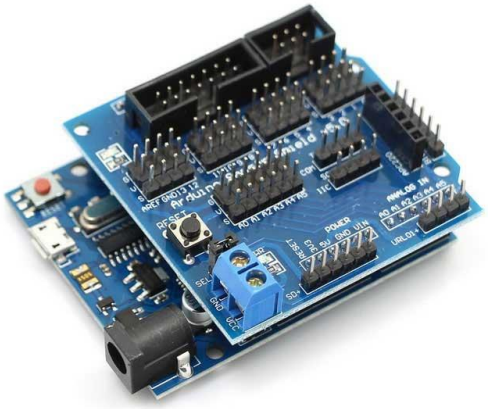


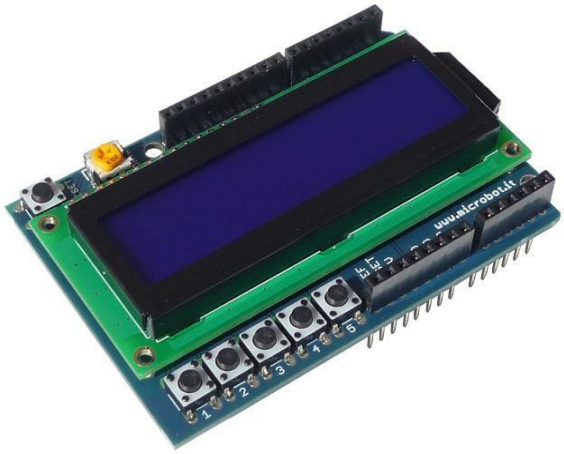
Obrázek 2 Arduino UNO



Obrázek 3 Arduino Mini

Další částí světa Arduina jsou takzvané „shieldy“. Jak už anglický název napovídá, jsou to takové „štíty“, které se na základní Arduino desku nasadí a rozšíří ji o nové funkce. Desky se snadno spojují díky použití oněch dutinkových lišt a pinových pásů, které poskytnou dostatečně pevné spojení samy o sobě, ale v případě konečného použití a usazení, ať už do krabičky nebo kamkoli jinam, je lepší využít připravených děr a sešroubovat vše dohromady.
Těchto shieldů je velké množství a škála, od prototypových, které na sobě mají nepájivé pole, lépe uspořádané piny, přes různé vstupně-výstupní, jako LED diodové matice, displeje s tlačítky pro ovládání, až po specializované, například pro ovládání 3D tiskárny, řízení servomotorů nebo připojení k internetu.





Obrázek 4 Sensor shield

Obrázek 5 LCD shield

Poslední velkou částí Arduino světa jsou periferie. Periferie jsou zařízení, které rozšiřují možnosti zařízení, ke kterému jsou připojené, v našem případě k Arduinu. A jsou, buď:

- vstupní – zprostředkovávají sběr dat, převádějí fyzikální veličiny (světlo, teplo, pohyb, údaje z klávesnice,…) na data zpracovatelná procesorem. Patří sem například: senzory atmosférických podmínek (teplota vzduchu, barometrický tlak a vzdušná vlhkost), měřič intenzity světla, klávesnice, mikrofon, koncové dorazy, průtokoměr, akcelerometr,…

Obrázek 6 Senzor teploty a vlhkosti

Obrázek 7 Membránová klávesnice

- výstupní – zprostředkovávají zobrazení či „zhmotnění“ dat, data zpracovaná procesorem převádějí na fyzikální veličiny (světlo, zvuk, údaje na displeji, otáčení motoru,…). Patří sem například: zobrazovače – LED diody/displeje, LCD/OLED displeje, reproduktor, motory – stejnosměrné/servomotory, relé moduly – pro spínání větší zátěže,…

Obrázek 8 LCD displej

Obrázek 9 8-kanálový relé modul

Software – Arduino Software (IDE)

Programovací prostředí Arduino IDE slouží k napsání programu pro procesor na desce, kontrole/kompilaci a jeho nahrání do procesoru.

Programy napsané pomocí Arduino Software jsou označovány, jako skici. Editor má funkce pro vyjímání a vkládání textu, hledání a nahrazování. Také barevně zvýrazňuje klíčová slova a příkazy v kódu, což dělá program dosti přehlednějším.

Prostředí programu je založeno na programu Processing, takže je jednoduché a přehledné i pro začátečníka. V horním okraji se nachází nástrojová lišta s pěti záložkami pro běžné funkce a několik nabídek:

- *Soubor* – zde jsou tlačítka pro práci se skicou (Nový, Otevřít, Uložit, Tisk, Příklady – jednoduché, komentované programy, které mají názorně předvést některé funkce,…)
- *Úpravy* – tlačítka a funkce pro úpravy a editaci samotného textu (krok zpět/vpřed, několik možností kopírování, Zakomentovat/Odkomentovat, úpravy odsazení, hledání v textu,…)
- *Projekt* – tlačítka pro práci s projektem/skicou (Kontrola/Kompilace, Nahrát, Přidat knihovnu,…)
- *Nástroje* – v této záložce se nachází nástroje pro programování (zobrazení sériového monitoru/plotru, výběr programované desky, programátoru nebo portu,…)
- *Nápověda* – zde se nacházejí tlačítka pro zobrazení nápovědy, informací o prostředí, pomoci pro řešení problémů, referencí (zde jsou jednotlivé příklady a funkce programovacího jazyka stručně a názorně vysvětleny), také úvodní seznámení a řešení problémů s Arduino kompatibilními deskami Galileo a Edison. Plus vše je dostupné off-line.

Pod touto lištou je šest ikon (pět vlevo a jedna vpravo), které slouží pro rychlý přístup k nejpoužívanějším funkcím:

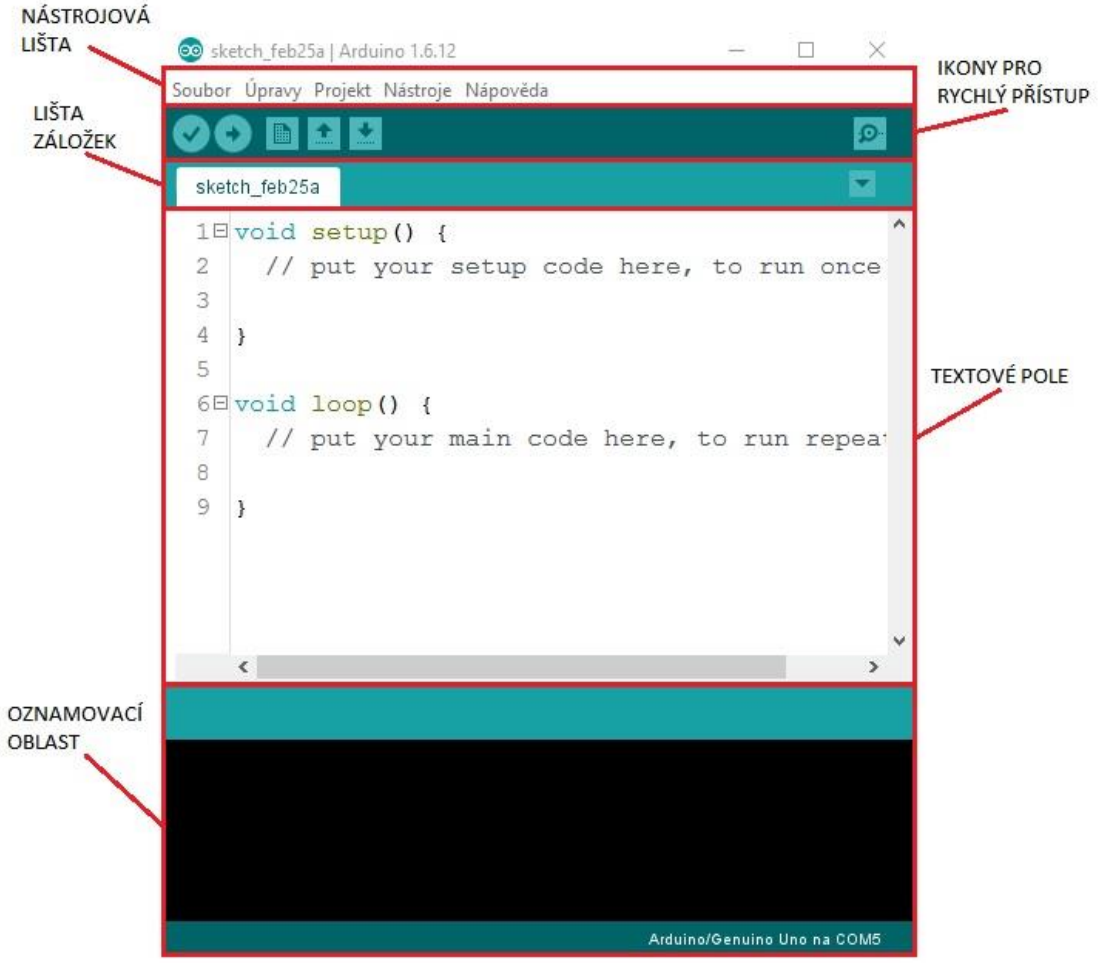
- *Ověřit* – program zkontroluje kód pro chyby syntaxi, které by znemožnily kompilaci
- *Nahrát* – dojde ke kompilaci kódu (včetně kontroly) a nahrání kódu do procesoru
- *Nový* – otevře se nové okno Arduino IDE se skicou s předpřipravenými funkcemi *setup* a *loop*
- *Otevřít* – rozbalí se nabídka s možností vybrat otevíranou skicu, buď ze souboru, z posledních otevřených nebo otevřít nějaký příklad
- *Uložit* – skica se uloží

- *Sériový monitor* – otevře se okno sériového monitoru (zobrazuje data odesílaná Arduinem po sériové lince)

Posledním prvkem v horní části je lišta záložek. Tyto záložky vypadají jako samostatné skici, ale protože se nacházejí v jednom souboru lze vzájemně odkazovat např. na procedury a proměnné mezi jednotlivými záložkami. Rozdělením programu do záložek se dosáhne u rozsáhlejších programů značného zpřehlednění.

Uprostřed okna se nachází textové pole pro psaní kódu, s možností číslování řádků a „sbalením“ úseků ohraničených složenými závorkami.

Ve spodní části okna je oznamovací oblast, ve kterém je zobrazována veškerá zpětná vazba (hlášení o prováděných operacích – kontrola/kompilace, nahrávání programu včetně informací o velkosti programu a kolik místa v paměti procesoru zabírá) a chybová hlášení včetně identifikace chyb a jejich zvýraznění v kódu.



Obrázek 10 Arduino software(IDE)

Způsoby měření času, jeho zobrazení a uchování

V této kapitole se budu zabývat měřením času pomocí samotného procesoru na desce Arduino, ale i pomocí externího modulu DS3231. Dále jeho zobrazením ve formátu HH:MM:SS a jeho uchováním (= nepřerušením měření času) i po odpojení napájecího napětí.

Krátké časové intervaly

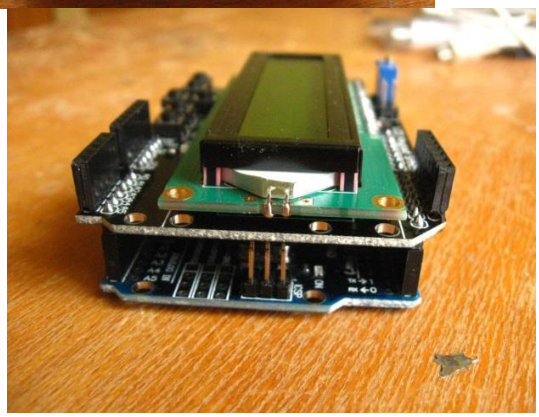
Jelikož krátký časový interval je poměrně široký pojem, tak jsem si ho definoval podle funkce Arduina *millis*. Tato funkce počítá počet milisekund uběhnutých od spuštění programu nahraného v procesoru, které ukládá do vlastní proměnné datového typu *unsigned long*, a voláním této funkce můžeme její hodnotu uložit do vlastní proměnné. Pokud bychom chtěli měřit opravdu krátké okamžiky, Arduino disponuje i funkcí *micros*, která je ještě o řád přesnější a tedy měří uběhnuté mikrosekundy. Unsigned long udává kolik bitů si proměnná v paměti zabere, v tomto případě se jedná o 32 bitů (4 byty). To znamená, že do této proměnné lze uložit číslo v rozsahu od 0 do 4 294 967 295 ($2^{32} – 1$). Přepočítáváním na vteřiny, minuty, hodiny a tak dále, dostaneme se až na 49 dní a téměř 17 hodin, což je obrovské číslo. Po dosažení maximální hodnoty proměnná „přeteče“, tj. vrátí se do nuly. Takže nepředpokládám, že by někdo opravdu využil nebo se alespoň přiblížil maximální možné hodnotě tohoto datového typu. Ovšem nereálné to není, jen mi to připadá zbytečné hlavně z důvodu neustálé potřeby napájení, což u tak dlouhé doby je riskantní, že dojde k přerušení a vše se vynuluje.

Nejlepším příkladem měření krátkých časových intervalů jsou stopky, jejichž realizaci pomocí této funkce a LCD displeje se pokusím představit.

Hardware

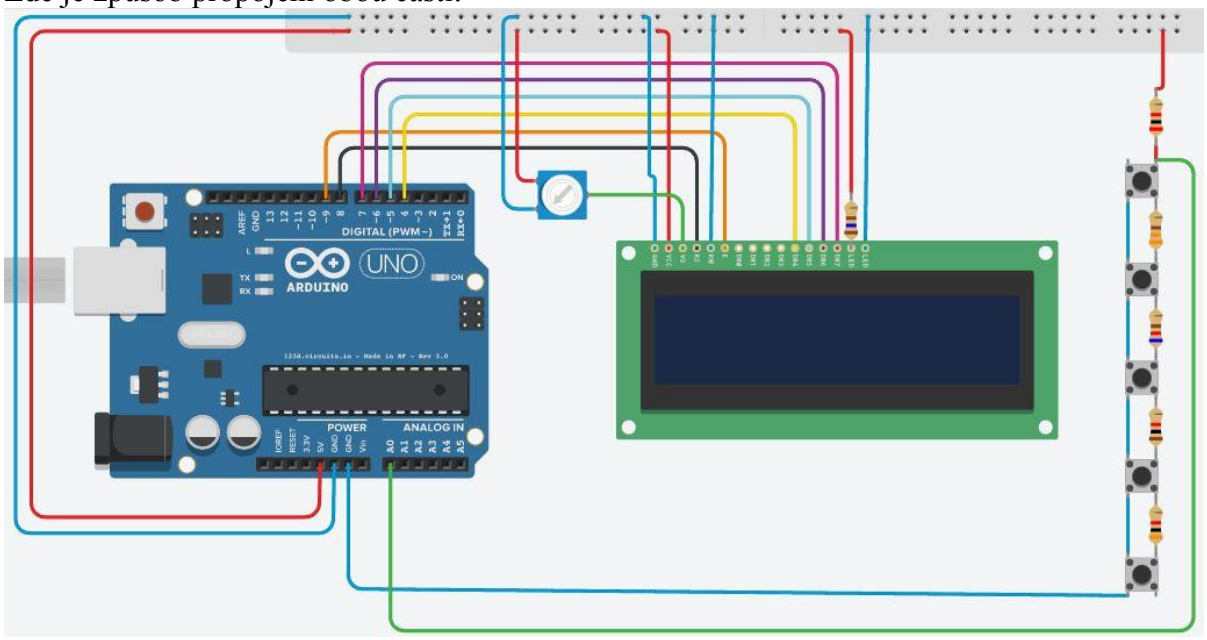
Jelikož při konstrukci stopek, coby prostředku pro demonstraci výše popsané funkce, zvolil jsem cestu nejmenšího odporu a sáhl po vývojové platformě Arduino UNO a po LCD shieldu s 5 tlačítky. Protože se jedná o shield a jak jsem psal v první kapitole, spojení shieldu s Arduinem je velmi jednoduché a tak je hardware připravený během pár vteřin.

Obrázek 11 Arduino + LCD shield (1)



Obrázek 12 Arduino + LCD shield (2)

Zde je způsob propojení obou částí:



Obrázek 13 Propojení Arduina s LCD shieldem

Software

Princip fungování stopek realizovaných pomocí této funkce je velmi jednoduchý, i když Arduino nemá kontrolovatelný časovač. Tento nedostatek jde velmi snadno vyřešit právě pomocí funkce *millis* nebo *micros*, ta udává počet milisekund uběhnutých od spuštění programu. K realizaci stopek jsou potřeba pouze dvě hodnoty (pokud nepočítám výsledek):

1. Počet milisekund na začátku měřeného časového úseku
2. Počet milisekund na konci měřeného časového úseku

Pak jednoduchým odečtením začátečního údaje od koncového získáme délku časového úseku v milisekundách, který poté můžeme přepočítat na vteřiny, minuty a tak dále. Zde na zjednodušeném kódu vysvětlím princip, kompletní kód je k dispozici v příloze.

```
void loop() {
  if (x < 800 && x > 600 )                    //  testování analogového vstpupu
    {                                         //  na stisknutí tlačítka
      if (r == false)                         //  testování činnosti stopek
       {
        start = millis();                     //  uložení startovacího času
       }
    else if (r == true)                       //  testování činnosti stopek
      {
      finished = millis();                    //  uložení konečného času
      DisplayResult();                        //  volání procedury zobrazení výsledku
      }
    r = !r;                                   //  změna logické hodnoty
    }
}
```

Obrázek 14 Zjednodušený program stopek (1)

Loop je hlavní programová větev která se opakuje stále dokola, po spuštění programu a provedení větve *setup*.

Prvním *if* zkouší stisknutí tlačítka a v případě LCD shieldu je potřeba zjišťovat analogovou úroveň, ne digitální, protože všechna tlačítka jsou přivedena na jeden vstup *A0*. Aby bylo možno tlačítka od sebe odlišit, jsou odstupňována odpory (viz. obr.13) a tak se při stisku každého tlačítka na vstupu objeví různá, pevně daná napětí, která můžeme v programu rozlišovat. Pokud ke stisku dojde, program se posune dále na druhý *if*.

Ten zjišťuje jako logickou hodnotu má proměnná *r* (hodnota *false* znamená, že stopky neběží, *true* znamená, že běží). Jestli stopky neběží, program to vyhodnotí jako pokyn pro spuštění stopek a startovací čas uloží do proměnné. A pokud běží, program určí, že jde o pokyn zastavit stopky, uloží konečný čas a „zavolá“ proceduru *DisplayTime*, která hodnoty zpracuje a zobrazí výsledek.

Na konec dojde ke změně logické hodnoty proměnné *r*, což bude mít za následek změnu stavu stopek, který program ve druhém a třetím *if*u vyhodnocuje – takže pokud do teď stopky neběžely (r = false), tak změní stav na „běží“ (r = true) a

program to při dalším stisku tlačítka vyhodnotí, tak že stopky chceme zastavit a uloží koncový čas a tak dále.

```
DisplayResult(){
  float h, m, s, ms;                          //  deklarace proměnných pro
  unsigned long over;                         //  hodiny, minuty, sekundy a zbytek

  elapsed = finished - start;                 //  výpočet délky intervalu v milisekundách


                                              //  přepočet milisekundy na:
  h = int(elapsed / 3600000);                 //    hodiny
  over = elapsed % 3600000;                   //
  m = int(over / 60000);                      //    minuty
  over = over % 60000;                        //
  s = int(over / 1000);                       //    vteřiny
  ms = over % 1000;                           //    a zbylé milisekundy
}
```

Obrázek 15 Zjednodušený program stopek (2)

Tato uživatelsky definovaná procedura slouží ke zpracování a zobrazení naměřených údajů.

Jako první jsou deklarovány proměnné, do kterých se budou ukládat výsledky, potom je vypočítána délka měřeného časového intervalu v milisekundách odečtením startovacího času od konečného. Nakonec je na řadě přepočítání na hodiny, minuty, vteřiny a zbylé milisekundy a zobrazení výsledku na LCD displeji, což jsem do této zjednodušené verze nedal.

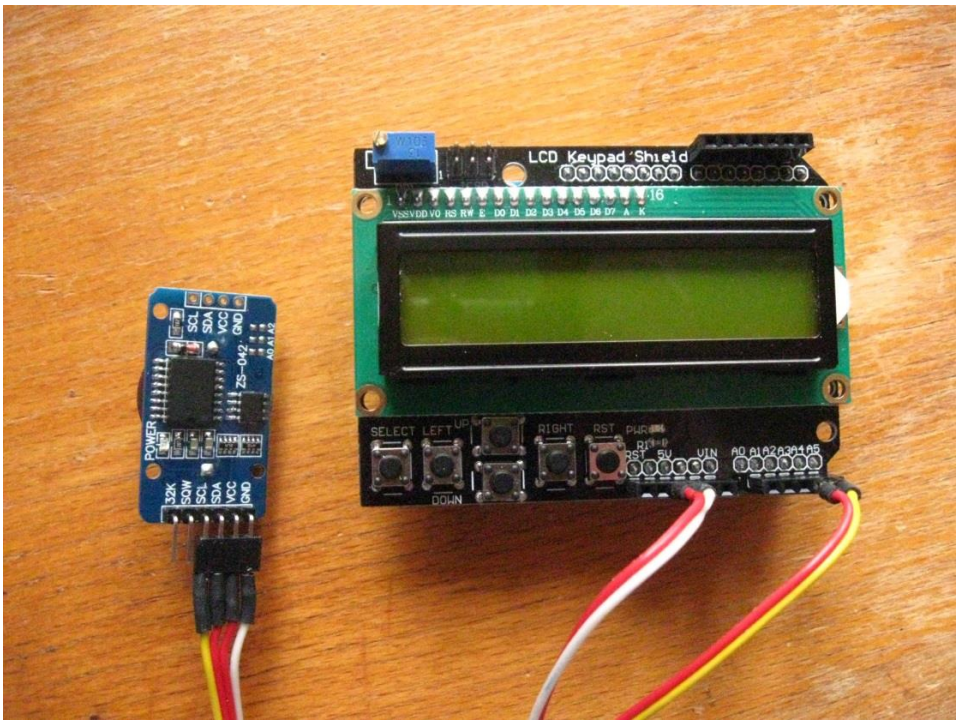
Protože 1h = 60min → 1min = 60s → 1s = 1000ms a znaky „ / “ a „%“ jsou operátory pro dělení se beze zbytku – uchová se pouze hodnota před desetinnou čárkou a zbytkové dělení (modulo) – uchová se pouze hodnota za desetinnou čárkou. Počet hodin zjistíme bezezbytkovým dělením finálního času 3600000 (1h = 3600s, 1s = 1000ms) a vydělením modulo získáme desetiny hodin, které vydělením 60000 (1m = 60s, 1s = 1000ms), opět modulo dělením dostaneme desetiny minut, které vydělením 1000 převedeme na vteřiny a modulo dělením vypočítáme zbylé milisekundy.

Dlouhé časové intervaly

Dlouhý časový interval jsem si stanovil pomocí funkce millis, tedy do dlouhých časových intervalů jsem zařadil všechny, které jsou moc dlouhé, než aby šly změřit pomocí této funkce. Takže podle maximální možné hodnoty této funkce to je 49 dní, což je hrozně dlouhá doba a proto sem zahrnu i intervaly delší než několik hodiny. Takže opět využiji koncepci stopek, tentokrát však na dlouhé časy.

Hardware

Technika použitá v tomto případě se nebude nijak lišit od té v předchozím příkladu. Opět jsem použil Arduino UNO a LCD shield, ale přibylo ještě jedno zařízení a to modul reálného času DS3231 (dále RTC).



Tento obvod blíže popíšu blíže v kapitole 2.4. Tento obvod funguje jako nezávislý časový strojek – měření času je udržováno vlastní baterií typu 2032. Takže velkou výhodou je nepotřebnost napájení po většinu doby, kdy měření probíhá, protože je potřeba jen při startu – vynulování RTC obvodu a při ukončení měření – uložení času, na kterém měření skončilo.

Komunikace s ním probíhá přes I^2C sběrnici a zjednodušeně tak, že se odešlou data s časem, na který se modul má nastavit a od této chvíle měří přesný čas nezávisle na tom, zda je připojen či ne.

Software

Tento způsob měření času je ve své podstatě minimálně stejně jednoduchý, jako funkce *millis*. Do modulu odešleme data s nulovým časem, na který se má nastavit – všechny měřené hodnoty se nastaví do nuly – vynulování. Až nastane konec časového intervalu, který je měřen, pouze si „zavoláme“ o odeslání dat z modulu v okamžik, kdy dojde k ukončení měření (nejčastěji nějaký vnější impuls nebo signál pojící se koncem intervalu, např. stisknutí tlačítka).

```
void loop() {
  int x = analogRead (0);                                          //  čtení tlačítka START/STOP
  if(x < 800 && x > 600){                                          //
   if ((millis() - lastButtonPressTime) > debounceDelay){          //  debounce
      if(r == false){                                              //  rozhodování podle hodnoty "r"
                                                                   //  false -> stopky neběží
                                                                   //  true -> stopky běží
                                                                   //  STOPKY NEBĚŽÍ:
        setDS3231time(0,0,0,0,0,0,0);                              //  vynulování RTC - start
        LCD.clear();                                               //  vyčištení displeje
        LCD.setCursor(0,0);                                        //  nastavení kurzoru
                                                                   //  (začátek prvního řádku)
        LCD.print("Probiha mereni..");                             //  sdělení o probíhajícím měření
      }                                                            //
```

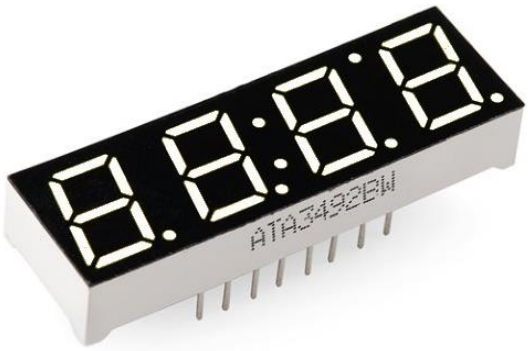
Obrázek 16 Program pro stoky na dlouhé časy (1)

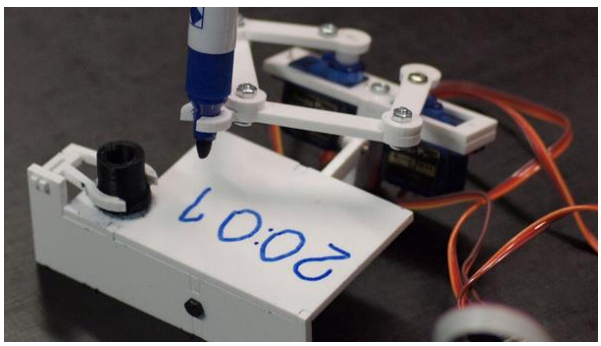
```
      else if(r == true) {                                    //  rozhodování podle "r"
                                                              //  STOPKY BĚŽÍ:
        displayTime();                                        //  přečtení,zpracování a zobrazení
       }                                                      //  údajů z RTC v okamžiku ukončení
                                                              //  měření - konec
      r = !r;                                                 //  překlopení hodnoty "r"
    }                                                         //
    lastButtonPressTime = millis();                           //  uložení času pro debounce
  }
}
```

Obrázek 17 Program pro stoky na dlouhé časy (2)

Zobrazení časové informace

Počet možností jak zobrazit či sdělit čas je tak velký, jak velká je lidská fantazie. Tím chci říct, že použít se dá opravdu leccos, od toho nejjednoduššího jako jsou displeje, přes různé vizuální adaptace, kdy kupříkladu počty rozsvícených LED diod určují počet jednotek a desítek minut nebo hodin až po třeba mechanické konstrukce, které jsou většinou nejzajímavější nebo nejoriginálnější. Dosti nápaditý design mechanického zobrazování času je „psací stroj“, který používá lesklou plochu a smazatelný fix (jako whiteboard). Při každé změně času údaj smaže a znovu napíše.



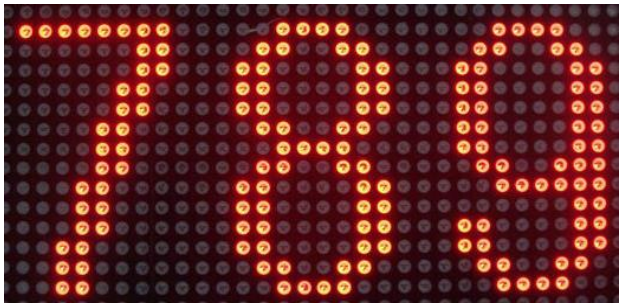


Obrázek 18 LED displej *Obrázek 19 Mechanická varianta*

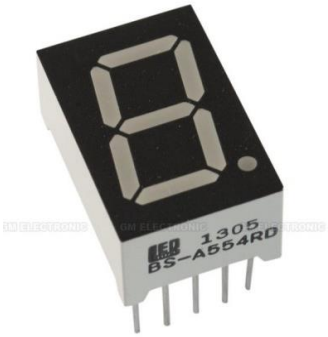
Já jsem se rozhodl zvolit cestu nejmenšího odporu a tedy použití displejů pro ukázku zobrazování časové informace. Rozhodl jsem se pro realizaci pomocí dvou nejběžnějších typů displejů – LED a LCD.

LED displeje

LED displeje patří k těm nejpoužívanějším typům zobrazovačů. Jsou široce využívány od amatérů a nadšenců do elektroniky až po průmyslová použití. Především díky své jednoduchosti, spolehlivosti a nízké spotřebě.



Obrázek 20 LED displej 1



Obrázek 21 LED displej 2

Technologie

Základem je LED (Light Emitting Diode) technologie. LED dioda je polovodičová součástka vyzařující světlo o různé vlnové délce – to záleží na výrobní technologii. Dále má jen jeden P-N přechod a to znamená, že záleží na polaritě připojeného napětí – v jednom směru vede, v druhém nikoliv, proto jsou směry pojmenovány jako „propustný“ a „závěrný“. To také znamená, že v každém směru má jiné charakteristické vlastnosti. Používá se zpravidla ve směru propustném, kdy začne vyzařovat světlo až při dosažení určitého napětí, které je také dáno technologií.
Základním kamenem polovodičové technologie je tzv. P-N přechod. Ten je složen ze dvou částí *P* a *N*, kde jsou základní polovodičové materiály dotovány příměsemi, které způsobí dva typy vodivosti:

- Vodivost typu *P* – positive (kladná). Polovodičový materiál je dotován příměsí označovanou jako *akceptor* – příjemce, ta způsobí, že ve výsledném materiálu je nedostatek elektronů a tím převládají *díry* (= prázdná místa po vázaných elektronech, která se díky absenci záporného náboje elektronu jeví jako kladná). Tím pádem je tento materiál schopen přijímat volné elektrony, které zaplní volná místa (díry), ale protože elektrony a díry nemohou zmizet ani se objevit,

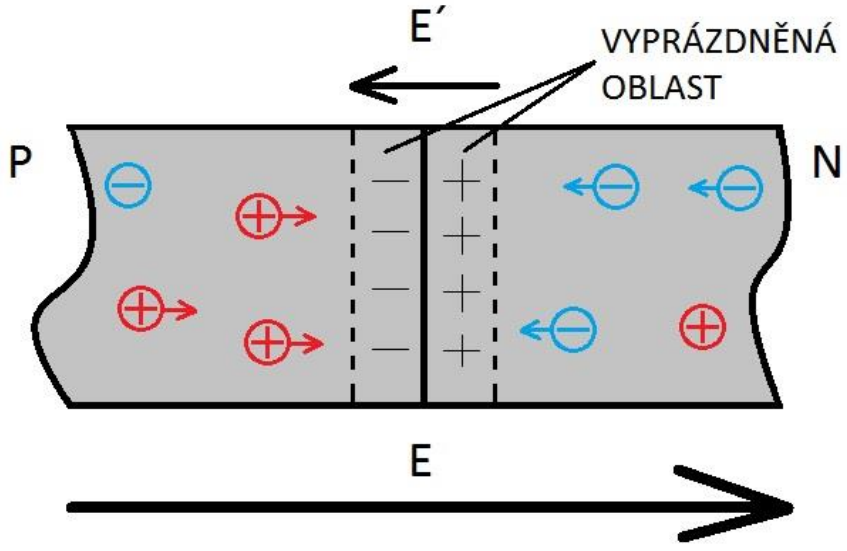
pouze se přesune do místa, odkud přišel volný elektron, takže je stále navíc a materiál si zachová vodivost typu *P*.

- Vodivost typu *N* – negative (záporná). Polovodičový materiál je dotován příměsí označovanou jako *donor* – dárce, ta způsobí, že v materiálu je více elektronů než možných vazeb – dojde k výskytu volných elektronů, které při získání dostatečné energie přeskočí na jiné místo. Ale jak bylo zmíněno v předchozím odstavci, elektrony se nemohou ztratit ani objevit, pouze si prohodí místa a zůstane jich stále nadbytek, takže si materiál zachová vodivost typu *N*.

Vytvořením těsného sousedství materiálů s těmito vodivostmi vznikne na jejich rozhraní přechod *P*-*N* a dojde k tomu, že:

- Elektrony z části *N* směřují k části *P* (jak říká Coulombův zákon – opačné náboje se přitahují) a některé do ní difuzí pronikají.
- Díry z části *P* směřují k části *N* a některé difuzí proniknou.

Tím nastane stav, kdy v blízkosti přechodu je na straně *N* nedostatek elektronů – jeví se kladně oproti zbylé části a na straně *P* nedostatek děr – jeví se záporně oproti zbylé části. U rozhraní tak vzniká elektrické pole *E´* s opačnou orientací než to, které vytvářejí hlavní nosiče náboje *E* – elektrony na straně *N* a díry na straně *P*.
Výsledkem je že v blízkosti přechodu chybí volné nosiče náboje – vznik vyprázdněné oblasti s určitým difuzním napětím U_D, které je v propustném směru potřeba překonat, aby se přechod stal vodivým. Například pro křemík (nejpoužívanější) je $U_D \approx 0{,}7V$.



Obrázek 22 P-N přechod

Připojením stejnosměrného napětí na přechod v:

- Propustném směru – pokud toto napětí bude větší než U_D, dojde k překonání opačného pole vyprázdněné oblasti, takže toto opačně orientované pole již nebrání průchodu volných nosičů náboje a přechod se stane vodivým
- Závěrném směru – dojde k posílení onoho opačného pole a tím se přechod stane více nevodivým

Vše spočívá v jevu zvaném „elektroluminiscence“. K elektroluminiscenci dochází, když je P-N přechod připojen na napětí v propustném směru a je otevřen. V tomto stavu dochází k rekombinaci elektronů s dírami - aby to bylo možné, musí se elektrony, nacházející se ve vyšším energetickém pásmu než díry, svojí energie zabavit. Právě tato energie se uvolňuje v podobě tepla (v případě *Si* a *Ge*) a fotonů (částic světla v případě arsenidů, fosfidů a dalších). Barva emitovaného světla (vlnová délka) závisí na materiálu, ze kterého je *P-N* přechod vyroben. V následující tabulce uvedu hlavní vlastnosti a používané materiály LED diod rozdělených podle barvy.

BARVA	VLNOVÁ DÉLKA [nm]	ÚBYTEK NAPĚTÍ [V]	MATERIÁL
Infračervená	$\lambda > 760$	$\Delta V < 1,63$	GaAs, AlGaAs
Červená	$610 < \lambda < 760$	$1,63 < \Delta V < 2,03$	AlGaAs, GaAsP
Žlutá	$570 < \lambda < 590$	$2,10 < \Delta V < 2,18$	GaAsP, AlGaInP
Zelená	$500 < \lambda < 570$	$1,9 < \Delta V < 4,0$	**Tradiční:** GAP, AlGaInP **Čistá:** InGaN, GaN

Modrá	450 < λ < 500	2,48 < ΔV < 3,7	ZnSe, InGaN, SiC – jako substrát
Ultrafialová	λ < 400	3,0 < ΔV < 4,1	InGaN (385-400nm), AlN (210nm)
Bílá	Široké spektrum	2,8 < ΔV < 4,2	**Studená:** Modrá/UV + žlutý fosfor **Teplá:** Modrá + oranžový fosfor

Tabulka 1 Hlavní vlastnosti základních barev LED

Konstrukce displeje

Hlavní konstrukční rozdělení těchto displejů je podle uspořádání zobrazovacích bodů:

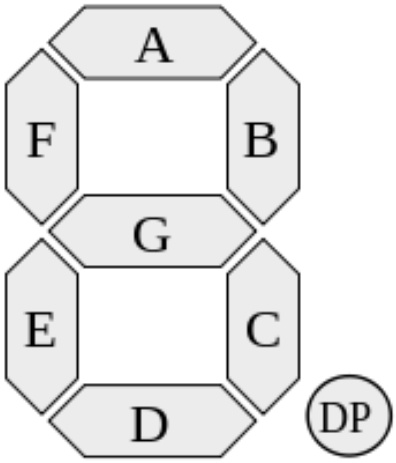
- Segmentové – mají zobrazovací body (segmenty) uspořádány tak, aby se jejich kombinací zobrazovat znaky – zejména arabské číslice. Dále se dají rozdělit podle počtu segmentů na: sedmi, devíti, čtrnácti a šestnácti segmentové. Ty 14 a 16 segmentové jsou už dobře použitelné i pro alfanumerické zobrazování, tj. znaků i čísel.

  Jednotlivé segmenty jsou uspořádány ve tvaru svisle orientovaného obdélníka, kde vodorovné strany jsou tvořeny jedním segmentem, svislé jsou tvořeny dvěma segmenty a sedmý segment se nachází vodorovně uprostřed. Plus jako osmý znak se zde nachází desetinná tečka umístěná vpravo dole, za samotným znakem. Pro rozsvícení jednotlivých segmentů jsou použity LED diody – 1 segment = 1 LED dioda. Každá má anodu a katodu, buď se spojí do společného vývodu:

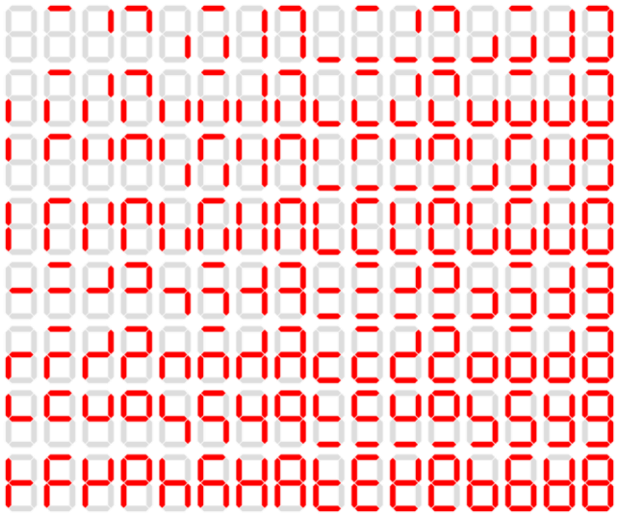
  - Anody – na společnou anodu je přivedeno kladné napětí a segmenty jsou ovládány spínáním katody „do nuly“ – připojením na nulový potenciál.

- Katody – společná katoda je připojena na nulový potenciál a segmenty jsou ovládány spínáním „do jedničky“ – připojením na kladné napětí.

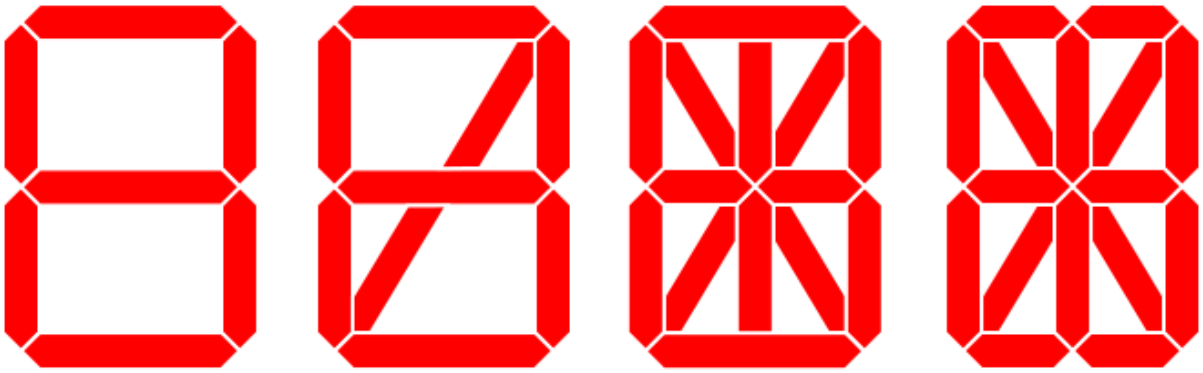
Velmi často se také používá, že se celý znak nakloní mírně doprava, což zlepšuje čitelnost.



Obrázek 23 7-segment: uspořádání



Obrázek 24 Možné kombinace 7 segmentového displeje

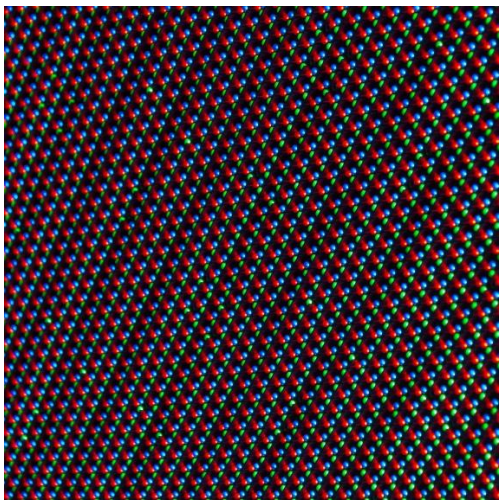


Obrázek 25 Varianty segmentových displejů, zleva: 7, 9, 14, 16 segmentový

- Maticové – zobrazovací body jsou uspořádány v matici (mřížce), kde se kombinací rozsvícených bodů zobrazují libovolné alfanumerické znaky, ale i různé obrazce. Tyto displeje pracují na tzv. multiplexním principu, kdy se body rozmístěné v řádcích a sloupcích ovládají rozsvěcováním jednotlivých bodů v řádku a vybíráním řádku ve kterém se zvolené body rozsvítí. Technologicky se toho dosáhne spojením anod do řádku a spojením katod do sloupců – jeden bod je tvořen jednou LED diodou. A přepínáním vysokou rychlostí se dosáhne toho, že lidské oko nebude stíhat sledovat jednotlivé změny bodů a vše splyne do jednoho obrazce.

  Obecně mají nízké rozlišení, ale díky zlepšující se technologii miniaturizace se již dosáhlo vysokého rozlišení (vměstnání více bodů na stejnou plochu). Ale to už jsou barevné LED obrazovky, což je jiná kapitola.

Obrázek 26 LED maticový displej

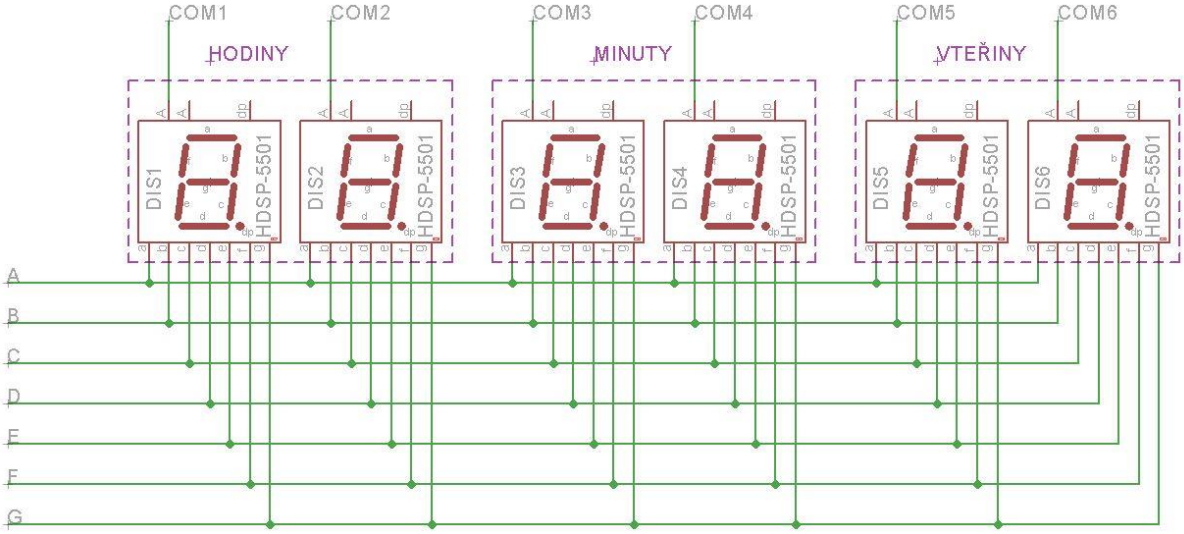


Obrázek 27 Detail barevné LED obrazovky

Použití pro zobrazení času

Segmentové displeje jsou zřejmě nejčastěji využívány pro digitální zobrazení časové informace například v digitálních budících, stopkách, měřících přístrojích,…Prostě téměř všude, kde je potřeba jednoduché zobrazení čísel, případně jiných znaků. Tento druh displejů má dobrou čitelnost a je jednoduše použitelný.
Čas je možné na segmentovém displeji zobrazit například použitím šesti jednoznakových displejů zařazených za sebou pro zobrazení času ve formátu HH:MM:SS. V tomto případě je nejjednodušší řídit displeje multiplexně – vývody od stejných segmentů jsou spojeny dohromady a společné anody/katody (vývod *COM*) jsou ponechány samostatně. Zobrazení čísla se provede tak, že sepnutím segmentů se se provede zobrazení požadovaného znaku na všech displejích a odpojením všech ostatních pinů COM se znak objeví pouze na požadovaném místě. Přepnutím na jiný pin COM se „zvolí“ jiný displej a současně změnou sepnutých segmentů se zobrazí na jiném místě. Při pomalé přepínací frekvenci bude vidět nějaké číslo na jednom displeji, po chvíli se na dalším displeji objeví jiné číslo a tak dále. Ale několika násobným zvýšením přepínací frekvence a díky tomu, že lidské oko nebude stíhat tak rychlou

změnu, bude to vypadat, že se čísla (potažmo znaky) zobrazují na všech displejích současně.

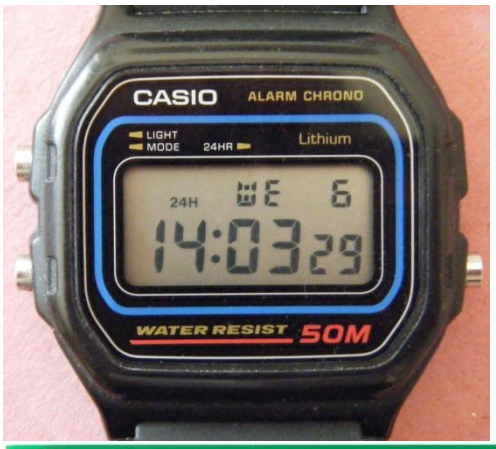


Obrázek 28 Schéma multiplexního zapojení šesti 7 segmentových displejů

LCD displeje

LCD displeje, neboli *displej z tekutých krystalů* (Liquid Crystal Display) je tenký a plochý zobrazovač. Jejich velkou výhodou je minimální spotřeba energie a možnost vytvoření speciálních znaků přesně podle potřeby, což je předurčuje k širokému využití. Samotné původní LCD displeje a jejich vylepšené varianty jsou dnes používány ve všech zařízeních, kde je žádoucí velmi nízká spotřeba, zejména v zařízeních napájených z baterie, jako digitální hodinky, kalkulačky, přenosné měřicí přístroje,…

V této části popíšu a pokusím se přiblížit technologii a konstrukci 2 základních typů LCD displejů – segmentového a maticového, oba monochromatické, pasivní.



Obrázek 29 LCD displej v digitálních hodinkách *Obrázek 30 Alfanumerický LCD maticový displej*

Technologie

LCD displeje používají pro zobrazování údajů body označené jako *pixely*. Ty mohou být buď monochromatické – jednobarevné nebo barevné.

Celý displej je složen z několika vrstev:

1. *Horizontální polarizační filtr* – polarizuje světlo procházející displejem
2. *Skleněný substrát* – má na sobě *ITO* elektrody (ITO = oxid india a cínu) jejichž tvar odpovídá tvaru pixelu zobrazeného na displeji, také jsou na substrátu vyleptány hladké horizontálně orientované drážky
3. *Vrstva nematických tekutých krystalů* – ovlivňují polarizační rovinu světla
4. *Skelněný substrát se společnou elektrodovou fólií* – je potažen *ITO* fólií fungující jako společná elektroda a na substrátu jsou vyleptány hladké vertikální drážky
5. *Vertikální polarizační filtr*

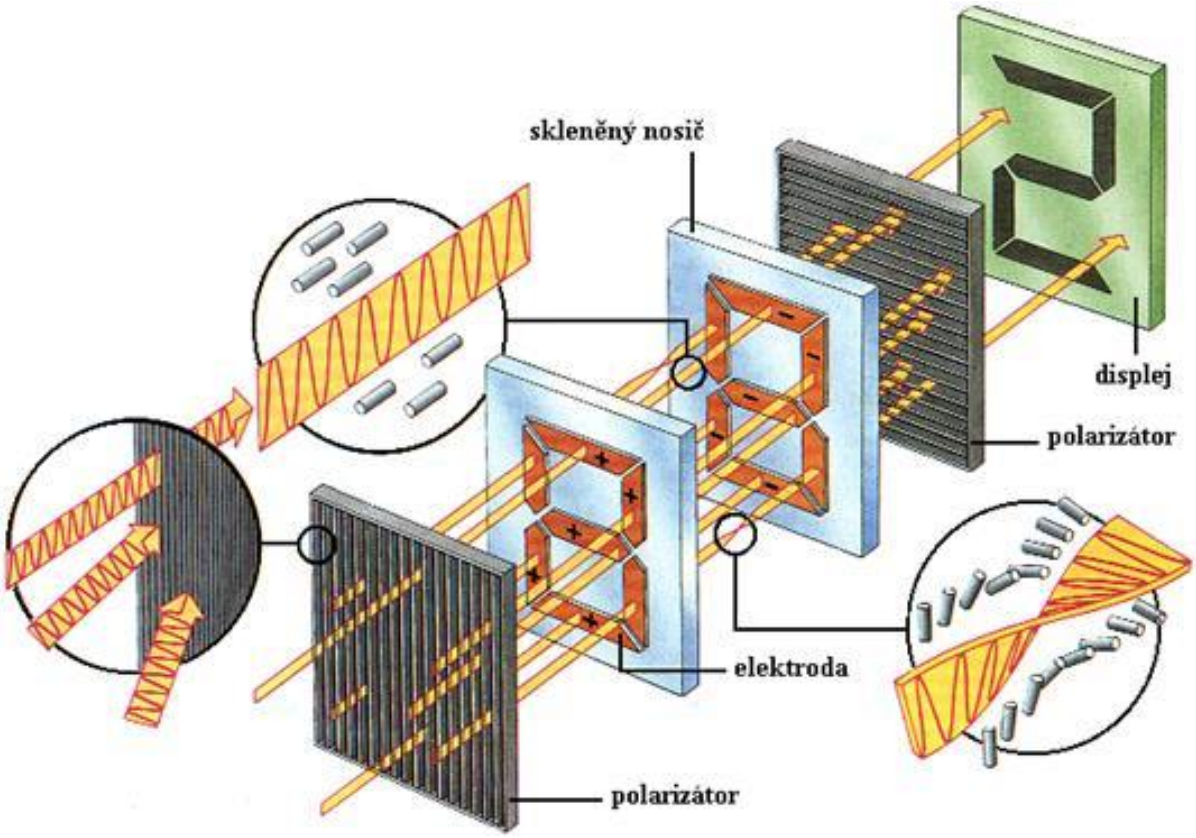
6. *Odrazná plocha / zdroj světla* – zde se světlo, které vstoupilo do displeje, odrazí ven, v případě podsvícených displejů je tato plocha nahrazena zdrojem světla

Vše funguje tak, že světlo, které se odrazilo od spodní vrstvy displeje nebo bylo vyzářeno zdrojem světla – podsvícením a odspodu prochází vrstvami displeje.

Vertikální polarizační filtr propustí pouze svisle kmitající světlo, které pokračuje přes skleněnou destičku s průhlednými elektrodami do vrstvy tekutých krystalů, kde se stanou dvě věci:

- Mezi elektrodami na nichž je elektrické napětí tekuté krystaly změní orientaci podle elektrického pole – otočí se o 90° na stejnou orientaci jako má první filtr – nedojde ke změně polarizace světla
- Mezi elektrodami na kterých elektrické napětí není, se orientace krystalů nezmění (zůstane stejná jako na druhém filtru – horizontální) a protože krystaly mění polarizaci světla podle své orientace, tak změní orientaci procházejícího světla z vertikální na horizontální

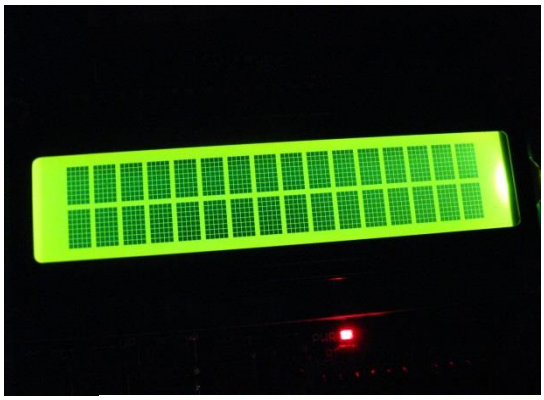
Dále světlo pokračuje přes druhou skleněnou destičku s elektrodami až k horizontálnímu polarizačnímu filtru, ten propustí pouze horizontálně polarizované světlo dál na displej. Jestliže je na určitý pixel přivedeno elektrické napětí krystaly se seřadí podle elektrického pole a nezmění vertikální polarizaci světla, které neprojde horizontálním filtrem – na displeji se objeví tmavý bod v důsledku absence světla. A světlo procházející přes krystaly bez napětí změní polarizaci na horizontální, tím pádem projde filtrem, nedojde k absenci světla a displej zůstane světlý.

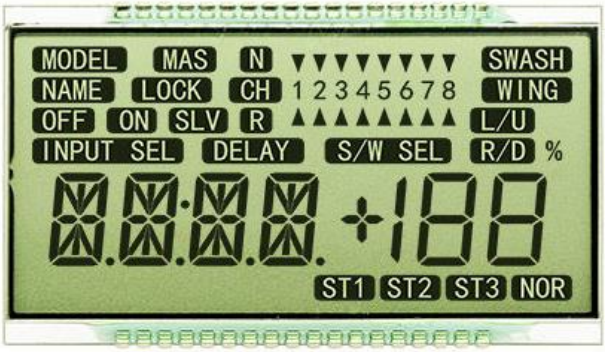


Obrázek 31 Princip pasivního LCD displeje

Konstrukce

Největší rozdíl mezi těmito dvěma displeji je v uspořádání pixelů. Buď mají přímo tvar znaků, které budou zobrazovat (například 7-segmentovka nebo speciální znaky) – zpravidla má každý znak svůj vývod a 7-segmentovky bývají řešeny multiplexně. Nebo jsou uspořádané do matice 5x8 bodů (pixelů), které tvoří prostor pro jeden znak a tyto znaky jsou na displeji uspořádány do řádků a sloupců, jednou z nejčastějších variant je 16x2. Protože například varianta 16x2 znaků má 1280 pixelů (5*8*16*2), což by vyžadovalo obrovské množství vývodů i v případě použití klasického multiplexu. Proto se používá řadič, který adresuje jednotlivé znaky a pixely v nich ovládá multiplexní kombinací adres – nejrozšířenějším řadičem je HD44780 od firmy Hitachi. Komunikuje se s ním paralelní sběrnicí za pomoci šesti nebo deseti pinů, z nichž dva slouží k řízení a výběru registrů a zbylé 4 až 8 k přenosu dat o zobrazovaných znacích. Také existují moduly, které převedou paralelní sběrnici na I^2C sběrnici, které stačí pouze 4 vodiče – 2 na napájení a 2 na komunikaci.





Obrázek 32 Viditelné rozložení pixelů 16x2 znaků, *Obrázek 33 Znakový LCD displej při maximálním kontrastu*

Použití pro zobrazení času

Já jsem zobrazování času zvolil alfanumerický maticový LCD displej s řadičem opět v podobě shieldu pro platformu Arduino a znovu kvůli jeho velmi jednoduchému použití a ovládání.

A protože hardwarovou stránku jsem popsal v části „Měření krátkých časových intervalů“ přesunu se tedy rovnou k programové části, která vše řídí.

Displeje v těchto shieldech jsou řízeny řadičem Hitachi HD44780 (nebo kompatibilními), který je používán nejen v těchto shieldech, ale v drtivé většině samostatných displejů. Z tohoto důvodu obsahuje Arduino knihovnu pro komunikaci s tímto řadičem – *LiquidCrystal*. Nebudu zde vypisovat všechny funkce, kterými knihovna disponuje, ale představím pár nejdůležitějších:

- *LiquidCrystal()* – touto funkcí jsou definovány piny, na které je display připojen
- *begin()* – tento příkaz sdělí Arduinu jak velký displej bude ovládat – do závorky zadávají 2 hodnoty: počet sloupců a počet řádků (např. 16x2, 8x2, 20x4)
- *setCursor()* – příkaz nastavující kurzor displeje do požadované pozice – opět se zadávají 2 hodnoty: kolikátý sloupec a řádek

- *print()* – zobrazení textu na displeji – přednastavený text musí být uvozen, k vypsání hodnoty proměnné uvozovky nejsou potřeba
- *clear()* – vyčistění displeje a nastavení kurzoru do levého horního rohu

Na následujících obrázcích ukáži použití těchto funkcí v programu pro stopky.

```
#include <LiquidCrystal.h>                    //  přidání knihovny

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);          //  deklarace operačního
                                              //názvu displeje a pinů
```

Obrázek 34 Použití funkcí pro LCD (1)

```
LCD.setCursor(0,1);       //  nastavení kurzoru na začátek 2. řádku
LCD.print("       ");     //  prázdné místo(mezery) pro přemazání starého nápisu
                          //  pro případ, že by nový byl kratší
                          //  -> zbyla by tam stará písmena
LCD.setCursor(0,1);       //  znovunastavení kurzoru na začátek 2. řádku
LCD.print("Patek");}     //  vypsání nápisu s dnem v týdnu
```

Obrázek 35 Použití funkcí pro LCD (2)

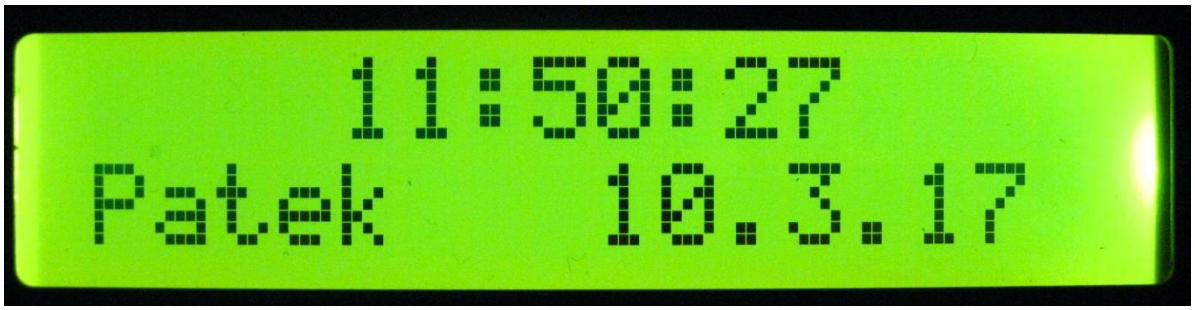
```
LCD.setCursor(4,0);          //  posunutí kurzoru
LCD.print("        ");       //  přemazání starého textu
LCD.setCursor(4,0);          //  posunutí kurzoru
if(hour < 10)                //  zjišťování zda jsou hodiny
                             //  jednociferné
  LCD.print("0");            //  pokud ano, zobrazí se před ním 0
LCD.print(hour, DEC);        //  zobrazení hodin
LCD.print(":");
if(minute < 10)              //  zobrazení minut...
  LCD.print("0");            //  zobrazení nuly před samotným číslem,
LCD.print(minute, DEC);      //  pokud je jednociferné
LCD.print(":");
if(second < 10)              //  zobrazení vteřin...
  LCD.print("0");
LCD.print(second, DEC);
```

Obrázek 36 Použití funkcí pro LCD (3)

```
LCD.setCursor(8,1);          //  přesunutí kurzoru
LCD.print(dayOfMonth, DEC); //  zobrazení dne v měsíci
LCD.print(".");
LCD.print(month, DEC);       //  zobrazení měsíce (čísla)
LCD.print(".");
LCD.print(year, DEC);        // zobrazení roku (poslední dvojčíslí)
```

Obrázek 37 Použití funkcí pro LCD (4)

A výsledek:



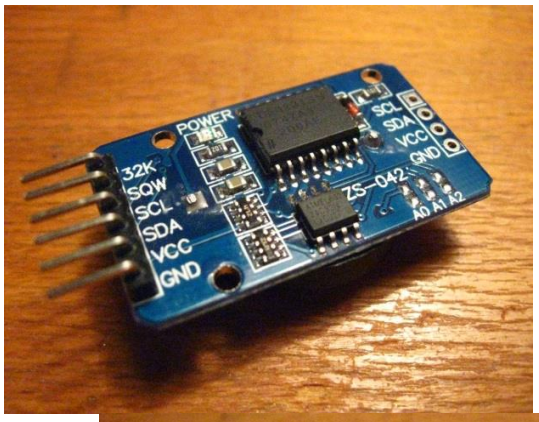
Obrázek 38 Hodiny na LCD displeji

Uchování časové informace

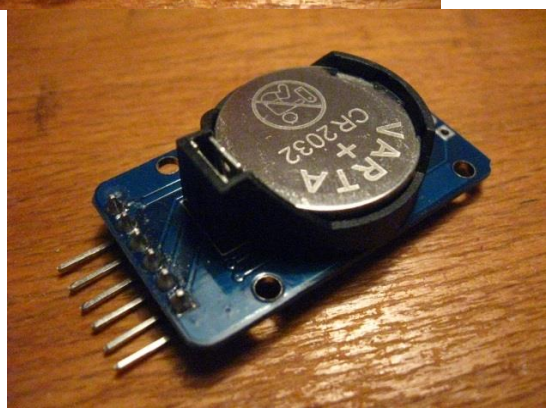
Další problematikou, zejména při konstrukci hodin, je udržet v chodu měření času i během výpadku napájení pro řídicí systém. Nejběžnější situací je výpadek elektrické sítě a po jejím obnovení radiobudík na nočním stolku ukazuje čas krátce po půlnoci (některé mohou blikat displejem, aby na sebe upozornili). Výpadek způsobil vymazání paměti EEPROM, ve které byly uloženy data o čase a například o budíku. Takže se nabízejí dvě možnosti: použít paměť a oscilátor, které se obejdou bez elektrického napájení nebo blok s pamětí a oscilátorem vybavit záložním napájením (i oscilátor proto, že pokud bychom udrželi pouze paměť bez oscilátoru, tak by po obnovení činnosti čas začal tam, kde skončil – zbytečné). Takže v dnešní době je logicky přijatelnější použít záložní napájení – dojde k přerušení napájení pro řídicí systém, ten přestane číst data z paměti a zobrazovat je, ale oscilátor s pamětí poběží dál, takže po obnovení hlavního napájení se obnoví i činnost řídicího systému a ten si přečte stále aktualizovaná data a zobrazí je. Tudíž není potřeba čas a budík znovu nastavovat, měření času běželo dál, jen nebylo zpracováváno.

Hardware

Pro realizaci jsem sáhl po něčem, co již někdo vymyslel a to po modulu ZS-042 s čipem DS3231 od Maxim Integrated, který je extrémně přesný, nízko nákladový obvod reálného času (RTC – Real Time Clock) s integrovaným teplotně stabilizovaným oscilátorem (TCXO – Temperature-Compensated Xtal Oscillator).



Obrázek 39 RTC modul ZS-042 (1)



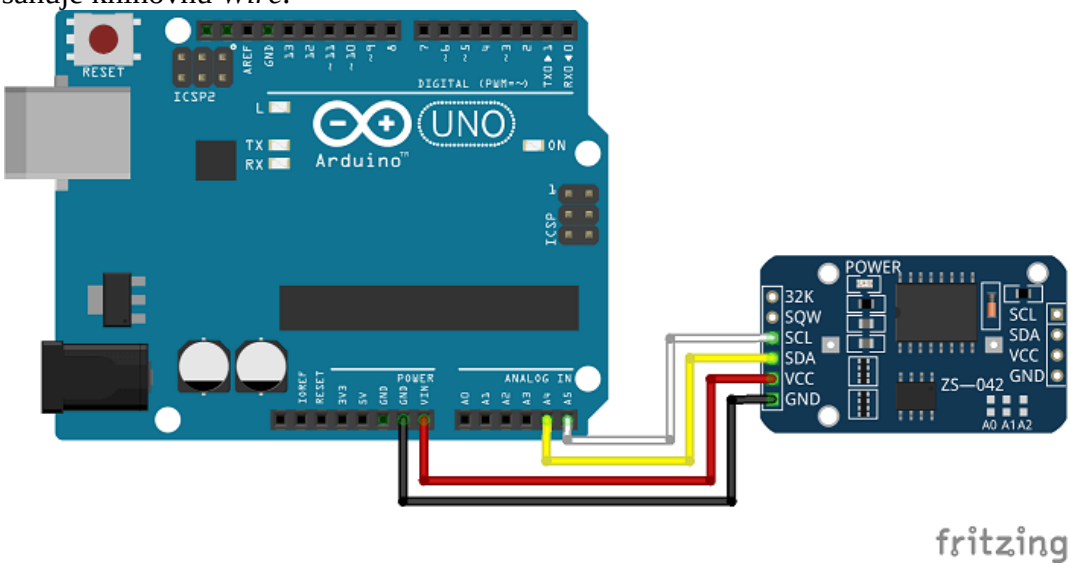
Obrázek 40 RTC modul ZS-042 (2)

Samotný RTC obvod je velmi přesný díky integraci krystalového oscilátoru do pouzdra, tím je dosažena částečná teplotní kompenzace a zlepšení dlouhodobé přesnosti, výrobce uvádí ±2ppm, což je chyba ±2 sekundy z 1 000 000s (= necelých 12 dní). Kromě této obdivuhodné přesnosti disponuje obvod spoustou funkcí, např.: přesné měření od vteřin po roky (včetně dne v týdnu, kompenzací přestupného roku do r. 2100 a automatickou úpravou data pro měsíce kratší 31 dnů), 12/24 hodinový formát a dva programovatelné budíky. Všechny funkce viz datasheet v příloze.

Modul obsahuje nabíjecí diodu a odpor pro dobíjení baterie, když je napájen z vnějšího zdroje. Pokud není použita nabíjecí baterie typu 2032, doporučuje se dobíjení vyřadit odpájením jednoho z prvků nebo přerušením cesty.

Obrázek 41 Dobíjecí obvod baterie

Komunikace s modulem probíhá přes I²C sběrnici, takže k výměně dat mezi Arduinem (či jiným mikrokontrolérem nebo platformou) stačí 2 vodiče – *SDA* (datový kanál) a *SCL* (hodinový signál), plus napájení. Pro komunikaci po této sběrnici Arduino obsahuje knihovnu *Wire*.



Obrázek 42 Připojení ZS-042 k Arudino Uno

Software

Pro ukázku jsem vybral program pro stopky dlouhých časů, které také komunikují s modulem ZS-042.

```
void loop() {
  int x = analogRead (0);                                    //  čtení tlačítka START/STOP
  if(x < 800 && x > 600){                                    //
   if ((millis() - lastButtonPressTime) > debounceDelay){    //  debounce
      if(r == false){                                        //  rozhodování podle hodnoty "r"
                                                             //  false -> stopky neběží
                                                             //  true -> stopky běží
                                                             //  STOPKY NEBĚŽÍ:
        setDS3231time(0,0,0,0,0,0,0);                        //  vynulování RTC - start
        LCD.clear();                                         //  vyčištení displeje
        LCD.setCursor(0,0);                                  //  nastavení kurzoru
                                                             //  (začátek prvního řádku)
        LCD.print("Probiha mereni..");                       //  sdělení o probíhajícím měření
      }                                                      //
```

Obrázek 43 Komunikace s DS3231 (1)

```
      else if(r == true) {                                   //  rozhodování podle "r"
                                                             //  STOPKY BĚŽÍ:
        displayTime();                                       //  přečtení,zpracování a zobrazení
      }                                                      //  údajů z RTC v okamžiku ukončení
                                                             //  měření - konec
     r = !r;                                                 //  překlopení hodnoty "r"
   }                                                         //
   lastButtonPressTime = millis();                           //  uložení času pro debounce
  }
}
```

Obrázek 44 Komunikace s DS3231 (2)

```
void setDS3231time(byte second, byte minute, byte hour, byte dayOfWeek, byte dayOfMonth, byte month, byte year){
                                                  // procedura nastavení času do DS3231
  Wire.beginTransmission(DS3231_I2C_ADDRESS);     // začátek přenosu na adrese čipu
  Wire.write(0);                                  // nastavení následujícího vstupu na začátek 2. registru
  Wire.write(decToBcd(second));                   // nastavení vteřin
  Wire.write(decToBcd(minute));                   // ...minut
  Wire.write(decToBcd(hour));                     // ...hodin
  Wire.write(decToBcd(dayOfWeek));                // ...dne v týdnu (1=neděle, 7=sobota)
  Wire.write(decToBcd(dayOfMonth));               // ...datum-den v měsíci (1 až 31)
  Wire.write(decToBcd(month));                    // ...měsíce
  Wire.write(decToBcd(year));                     // ...roku (0 až 99)
  Wire.endTransmission();                         // konec přenosu
}
```

Obrázek 45 Komunikace s DS3231 (3)

```
void readDS3231time(byte *second, byte *minute, byte *hour, byte *dayOfWeek, byte *dayOfMonth, byte *month, byte *year){
                                                  // procedura čtení času z DS3231
  Wire.beginTransmission(DS3231_I2C_ADDRESS);     // začátek přenosu na adrese čipu
  Wire.write(0);                                  // nastavení "kurzoru" registru na 00h
  Wire.endTransmission();                         // konec přenosu
  Wire.requestFrom(DS3231_I2C_ADDRESS, 7);        // požadavek na 7 bitů z adresy 00h v registru
  *second = bcdToDec(Wire.read() & 0x7f);         // přijmutí vteřin
  *minute = bcdToDec(Wire.read());                // ...minut
  *hour = bcdToDec(Wire.read() & 0x3f);           // ...hodin
  *dayOfWeek = bcdToDec(Wire.read());             // ...dne v týdnu
  *dayOfMonth = bcdToDec(Wire.read());            // ...data - dne v měsíci
  *month = bcdToDec(Wire.read());                 // ...měsíce
  *year = bcdToDec(Wire.read());                  // ...roku
}
```

Obrázek 46 Komunikace s DS3231 (4)

Procedury *bcdToDec()* a *decToBcd()* slouží k převodu údajů mezi číselnými soustavami – desítková a dvojkově kódovaná desítková, protože čip DS3231 funguje v systému BCD a ne v desítkovém.

Vzorový výrobek

Většinu zkušeností a poznatků z minulých úkolů a aplikací jsem ověřil při konstrukci vzorového výrobku „Digitální hodiny s budíkem“.

Návrh

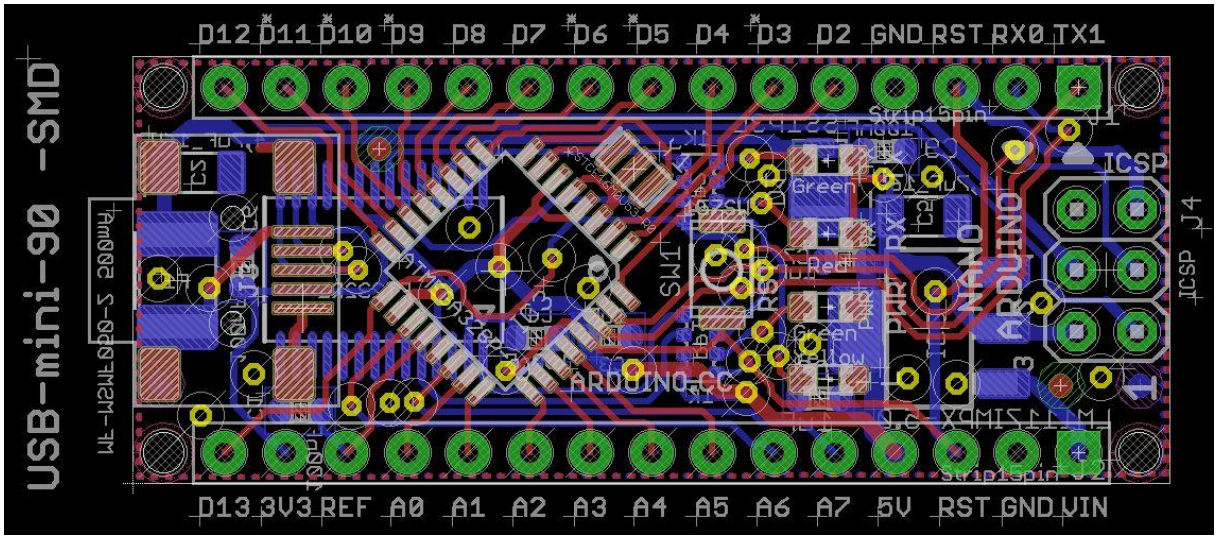
Řízení

Jako řídicí prvek jsem zvolil platformu Arduino, konkrétně desku Arduino Nano osazenou procesorem Atmel ATmega328P. Tuto desku jsem vybral především kvůli malým rozměrům (18x45mm), vestavěným podpůrným obvodům a programátoru.

Jak jsem zmínil v předchozím odstavci, deska je založena na procesoru ATmega328P s taktovacím krystalem o frekvencí 16MHz a 32KB vnitřní paměti. Spojení desky s okolím zajišťuje celkem 22 pinů:

- 14 digitálních vstupně-výstupních, z nichž 6 může být použito jako PWM (Pulse Width Modulation)
- 8 vstupních analogových, prvních 6 jde použít i jako digitální vstupy a na 5. a 6. (A4, A5) se připojují signály SDA a SCL pro I^2C sběrnici

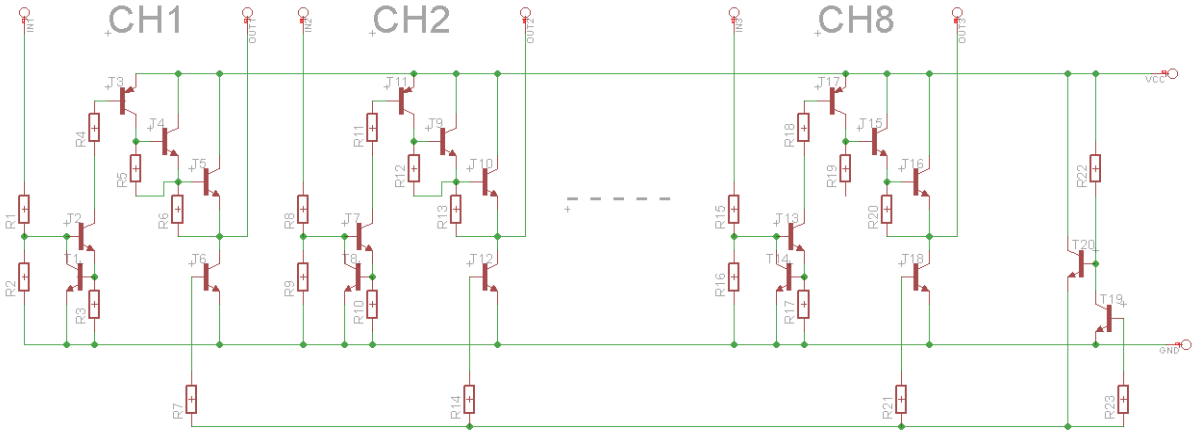
Vše je napájeno buď z USB portu, nebo z externího napájení, u originální Arduino Nano desky je v rozsahu 7-12V. V případě klonů se může lehce lišit.



Obrázek 47 Arduino Nano DPS

Protože segmenty a mřížky, které je potřeba spínat, pracují na napětí 24V a Arduino pracuje s 5V, není možné je ovládat přímo. Je tedy potřeba přidat spínací prvek – tranzistor. Nebo ještě lépe tranzistorové pole – vybral jsem Sanyo LB1290 v pouzdře DIP18 navržené pro propojení nízkonapěťových digitálních zařízení s fluorescenčními zobrazovači, takže naprosto ideální.

Toto pole je složeno z bipolárních tranzistorů a má 8 na sobě nezávislých kanálů (8 vstupů a výstupů). Je navrženo tak, že pokud napětí na vstupu překročí 2,6V, sepne se příslušný výstup, na kterém se objeví napětí, kterým je pole napájeno.

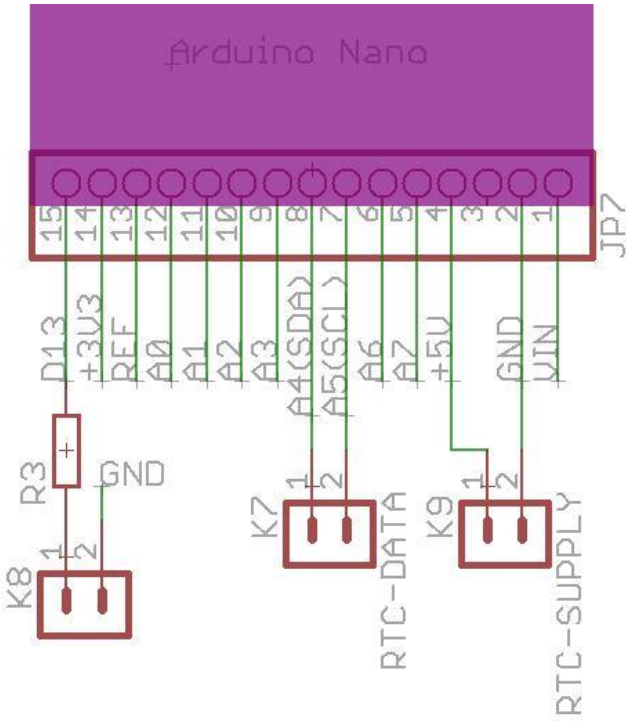


Obrázek 48 LB1290 - vnitřní schéma

Detailněji viz datasheet v příloze.

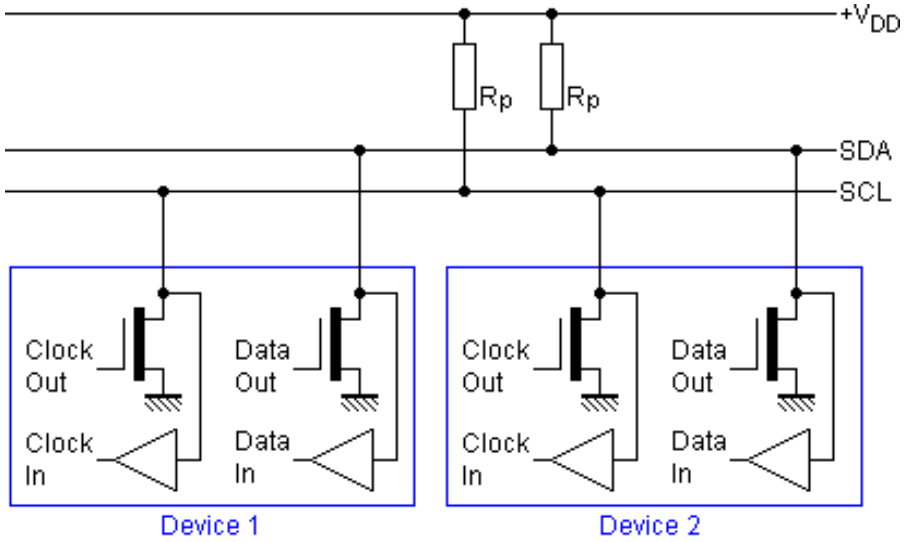
Měření času

Přesné měření času opět zajišťuje modul ZS-042. Ten, jak již víme, komunikuje přes I^2C sběrnici a proto je připojen na piny, které tuto komunikaci umožňují:
SDA → A4, SCL → A5. Ve schématu znázorněno konektorem *K7*.



Obrázek 49 Připojení RTC k Arduinu – schéma

Už zbývá jen napájení, protože I²C sběrnice bez něj z konstrukčních důvodů nefunguje. Zde je realizováno konektorem *K9*.



Obrázek 50 Schéma zapojení I2C sběrnice

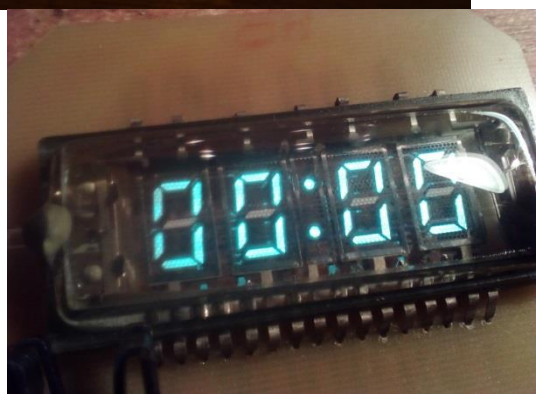
Zobrazování

Dalším krokem je časovou informaci z RTC modulu zobrazit. Jako nejvhodnější se jeví 7 segmentové displeje, ale těch je obrovské množství velikostí, uspořádání, barev, ale i technologických provedení je několik druhů – dnes jsou nejběžnější LED a LCD, v minulosti také tzv. digitrony a VFD.

Obrázek 51 LED displej

Obrázek 52 LCD displej

Obrázek 53 Hodiny s digitrony



Obrázek 54 VFD displej

Mojí jasnou volbou byly právě VFD displeje, protože když jsem na tento typ displejů narazil, tak se mi velice zalíbily.

První zařízení využívající VFD technologii byl jednoduchý indikátor DM160 vyrobený firmou Phillips v roce 1959. O tři roky později, v roce 1962, byl v Japonsku vyroben první více segmentový VFD displej, jednalo se o jednociferný sedmi segmentový displej. Od poloviny 70. let již byly běžné v kalkulačkách a spotřební elektronice, ale i v autech majících digitální ukazatele na palubní desce. A o desetiletí později jich byly vyráběny stamiliony ročně.

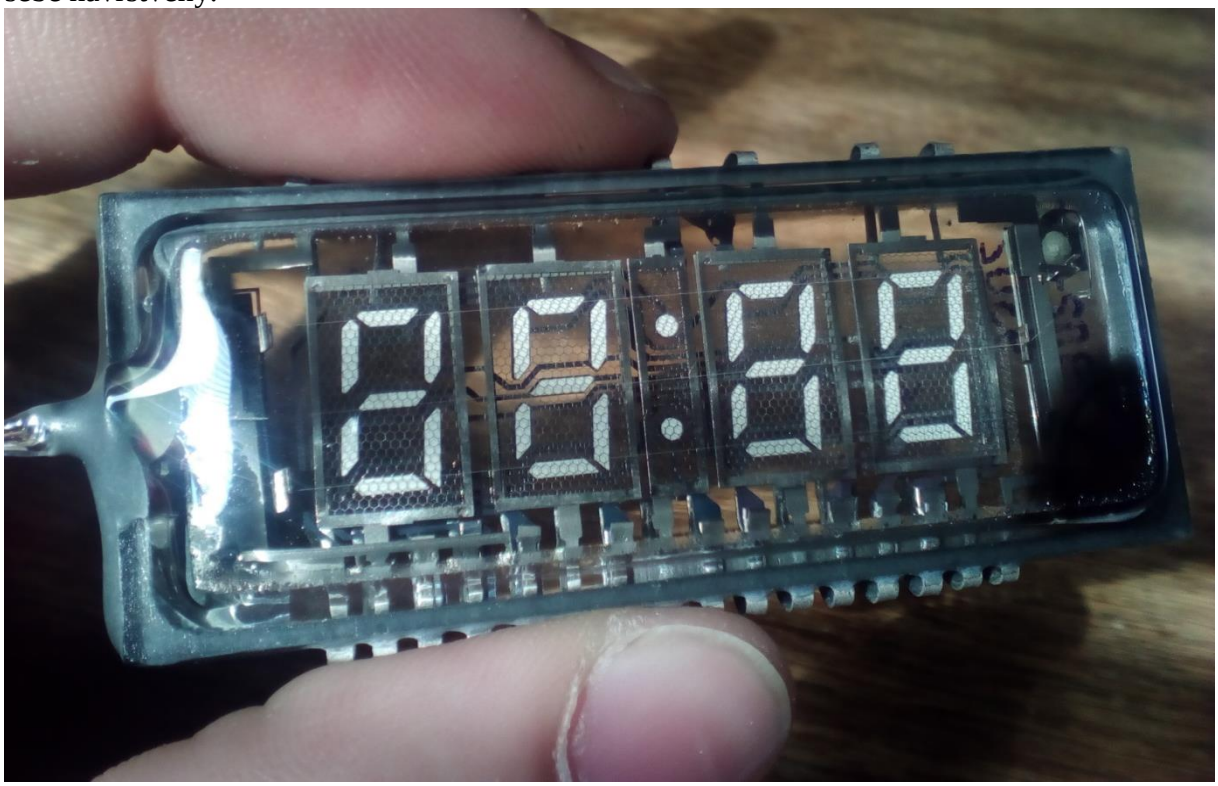
Jako nejvhodnější „kandidát“ se jevil model IVL2-7/5 sovětské výroby, navržený právě pro tyto účely – čtyřciferné zobrazování digitálního času.

Technologie

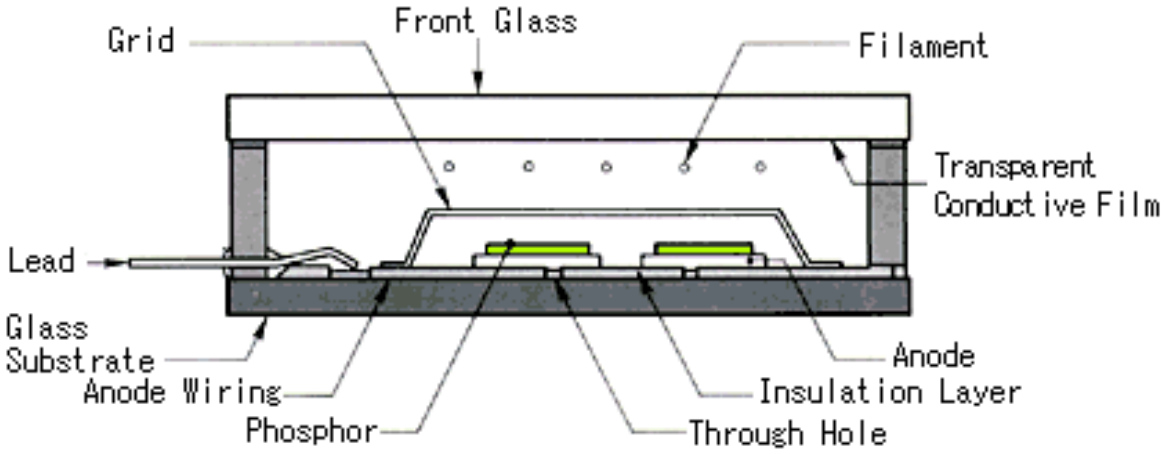
Podstatou fungování vakuových fluorescenčních displejů je jev zvaný *katodoluminiscence*, která je zhruba podobná principu CRT obrazovek. V principu elektrony emitované katodou jsou přitahovány a dopadají na anodu, jejíž povrch je potažen luminiscenční vrstvou (nejčastěji fosforem), která při dopadu elektronů vydává světlo.

V případě VFD displejů je jako zdroj elektronů (katoda) použita žhavená wolframová vlákna potažená oxidy Barya, Stroncia a Kadmia. Anoda je zde ve formě segmentu/znaku zobrazovaného na displeji a mezi vlákny a segmenty je ještě „mřížka“, jejímž potenciálem lze kontrolovat tok elektronů – nejčastěji je využívána ve „spínacím režimu“, kdy jsou pomocí ní zapínány nebo vypínány jednotlivé číslice (v mém případě) nebo skupiny segmentů. Důvodem pro toto konstrukční řešení je především multiplexní řízení displeje.

Z konstrukčního hlediska je displej takový sendvič, kde jsou všechny části na sebe navrstveny.



Obrázek 55 IVL2-7/5



Obrázek 56 Konstrukce VFD displeje

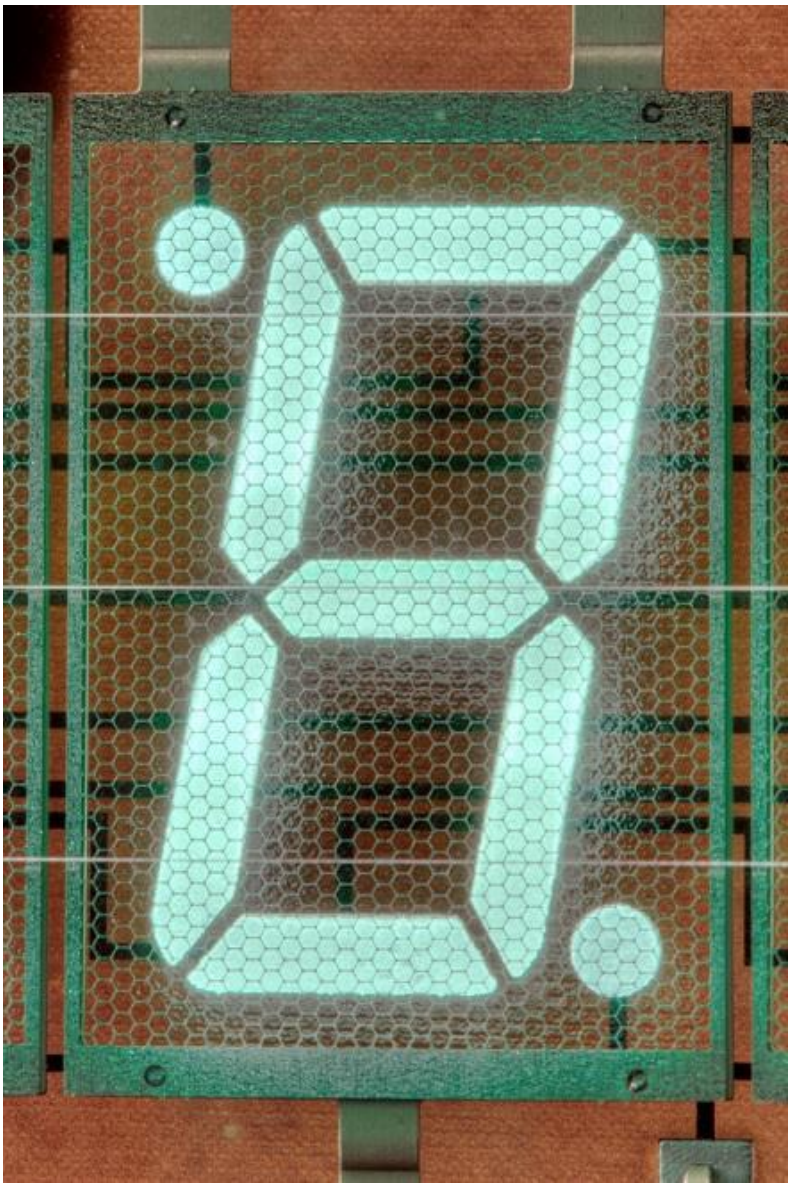
Začnu odspodu:

- *Skleněný substrát*
- *Anody* – vodivá vrstva ve tvaru segmentů a cest k vývodům nanesená na skleněném substrátu
- *Luminiscenční vrstva* – vrstva nejčastěji fosforu nanesená na zobrazované segmenty
- *Vývody* – vyvedení segmentů, mřížek a žhavené katody ven ze skleněného pouzdra
- *Mřížky* – drobné „pletivo“ překrývající danou číslici nebo skupinu segmentů
- *Katoda* – žhavená wolframová vlákna, sloužící jako zdroj elektronů, natažená nad celou zobrazovací plochou displeje
- Skleněné pouzdro s průhledným vodivým filmem na vnitřní straně (v podstatě totožným jakým jsou u LCD provedeny elektrody)

Porovnání s LED a LCD displeji:

- ⊕ Ve své době všestrannější a atraktivnější alternativa
- ⊕ Také měly menší spotřebu než tehdejší LED displeje
- ⊕ Největší výhoda, která je stále aktuální, je velký jas a dobrý kontrast, což zaručuje velmi dobrou čitelnost za všech světelných a klimatických podmínek
- ⊕ Velký rozsah pracovní teploty, od teplot hluboko po nulou až po ± 80°C
- ⊕ Velký pozorovací úhel
- – Dnes vytlačeny LCD co do universálnosti, levné výroby a nízké spotřeby
- – „Blednutí“, postupem času svítivost fosforu ztrácí na intenzitě
- – Zobrazování pouze předdefinovaných znaků

Ale tuto poslední nevýhodu smazala firma *Noritaki itron*, která VFD displeje dovedla k dokonalosti – vyvinuli maticové displeje alfanumerické i grafické, ale dokonce i dotykové.



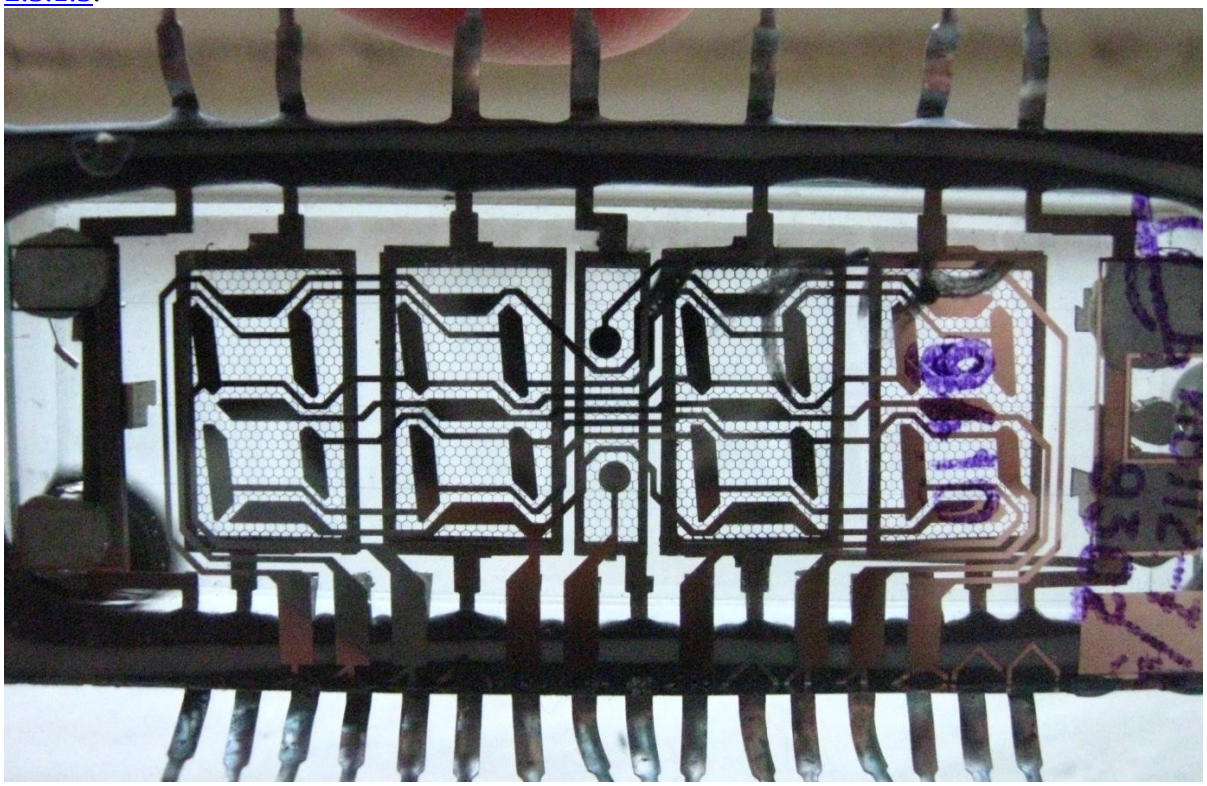
Obrázek 57 Detail VFD displeje

Použití

Tyto displeje potřebují k provozu dvě napájecí napětí – jedno pro žhavení katody a jedno pro ovládání segmentů a mřížek. Také je rozdíl jestli používáme malý nebo velký displej, protože malé displeje se více hodí napájet stejnosměrným napětím a pro velké je zase vhodnější střídavé, což je dáno rozložením napětí po délce katodových vláken, jejich strukturou a také strukturou mřížek.

V případě mého displeje se používá stejnosměrné napětí a to 2,4V pro žhavení katody a 24V pro segmenty a mřížky. Displej je konstruován pro použití multiplexního

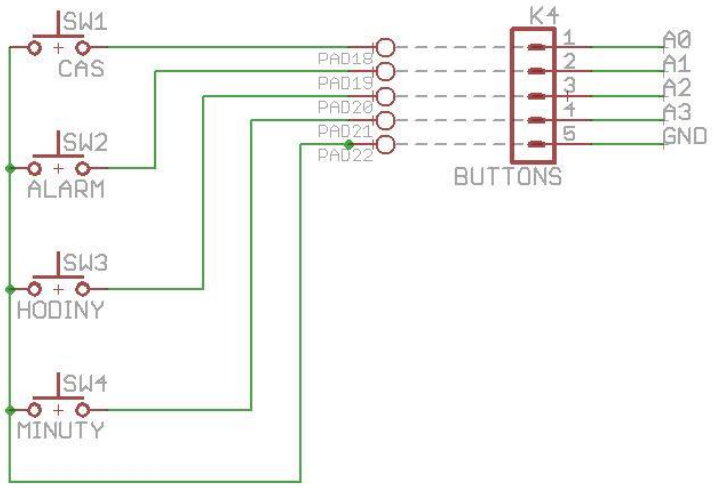
řízení – všechny stejné segmenty jsou propojeny, tedy se rozsvěcejí všechny najednou. Pro rozdělení na jednotlivé číslice (aby se číslo nezobrazovalo na všech pozicích současně) má každé číslo svojí vlastní mřížku, pomocí které se ostatní pozice vypnou – na mřížku na zvolené pozici je přivedeno napětí, na ostatní ne → číslo svítí pouze na jedné na zvolené pozici. Pro zobrazení jiného čísla na jiné pozici přepneme na jinou mřížku a zobrazíme číslo. Vše se ale musí dít dostatečně rychle, aby lidské oko nebylo schopno postřehnout přepínání pozic – uvidíme svítit vše najednou. Popsáno také v části 2.3.1.3.



Obrázek 58 Propojení segmentů IVL2-7/5

Zadávání

Protože jde o vzorový výrobek „digitální hodiny s budíkem“ je potřeba nějak nastavit čas hodin a budíku, ale také zapnout a vypnout budík. Toho, podle mě, jde nejlépe dosáhnout čtyřmi tlačítky a jedním posuvným spínačem.



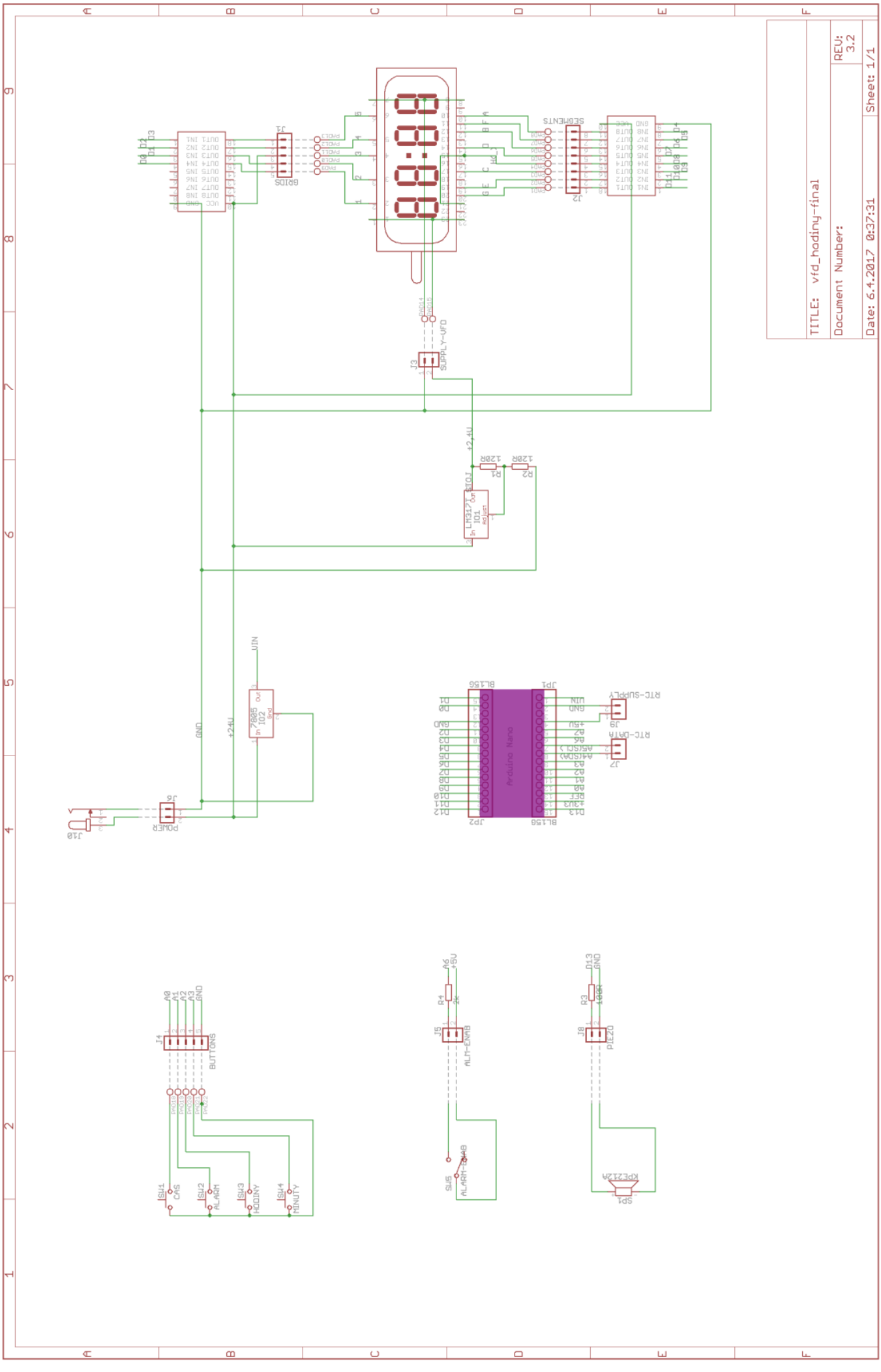
Obrázek 59 Tlačítka

Obrázek 60 Posuvný spínač

Funkce jednotlivých tlačítek jsou následující:

- *Červené* – nastavování času
- *Modré* – nastavování budíku
- *1. Šedé (levé)* – zadávání hodin (s každým stiskem se zvýší o 1)
- *2. Šedé (pravé)* – zadávání minut (s každým stiskem se zvýší o 1)
- *Posuvný spínač* – zapínání a vypínání budíku

Dohromady vše funguje tak, že čas se nastavuje držením červeného tlačítka a mačkáním šedých – hodiny a minuty a budík držením modrého a opět mačkáním šedých. Ten se aktivuje posuvným spínačem a zvuk je vydáván piezoměničem.



Obrázek 61 Kompletní schéma

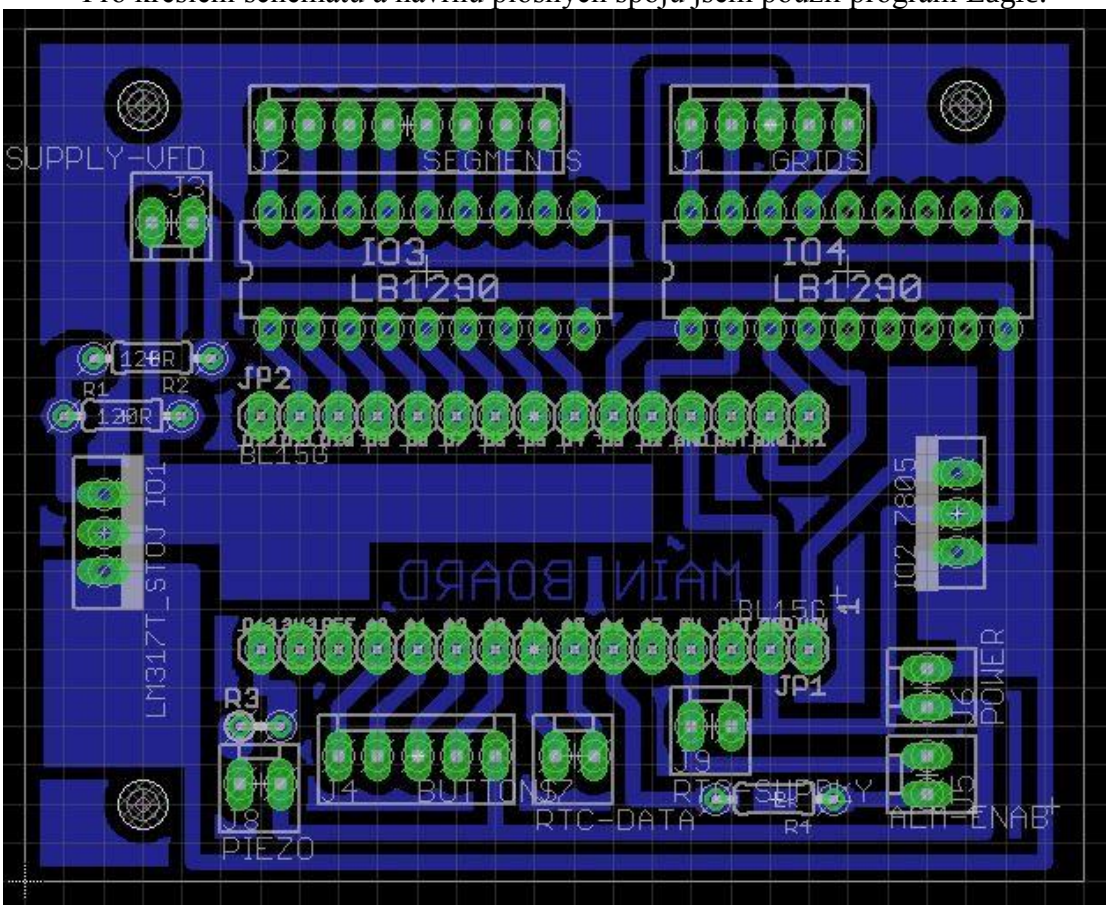
Konstrukce

Plošné spoje

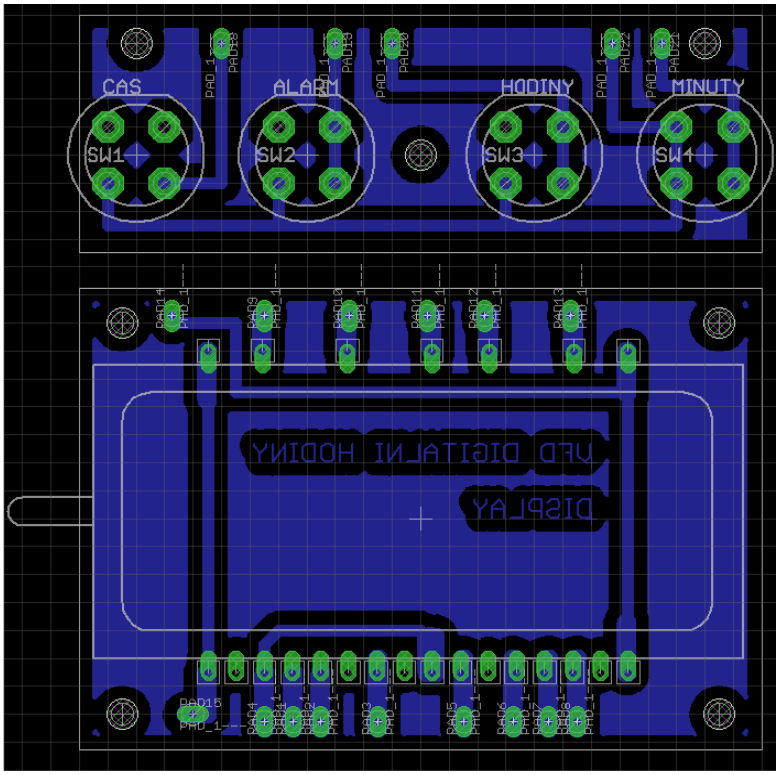
Pro lepší vměstnání všeho do krabičky jsem plošný spoj rozdělil na 3 části: hlavní desku, displej a tlačítka. Hlavní deska je umístěna ve dně krabičky a nachází se na ní řídící elektronika, respektive patice pro Arduino Nano a tranzistorová pole, plus napájení. Displej i tlačítka jsou umístěny na vlastním plošném spoji, které slouží pouze pro snadnější uchycení ke stěně krabičky a jsou z nich pouze vyvedeny vodiče s konektory.

Součástky IO1 a IO2 slouží k úpravě hlavního napájecího napětí (24V) na napájencí napětí pro displej a Arduino. IO1 je nastavitelný stabilizátor napětí LM317T a odpory R1 a R2 je nastaven na 2,4V pro žhavení katody displeje a IO2 je pevný 5V stabilizátor 7805, který dodává napětí pro řídící logiku – Arduino. Samostatné odpory R3 a R4 jsou předřadné odpory k piezoměniči a tlačítku SW5.

Pro kreslení schématu a návrhu plošných spojů jsem použil program Eagle.

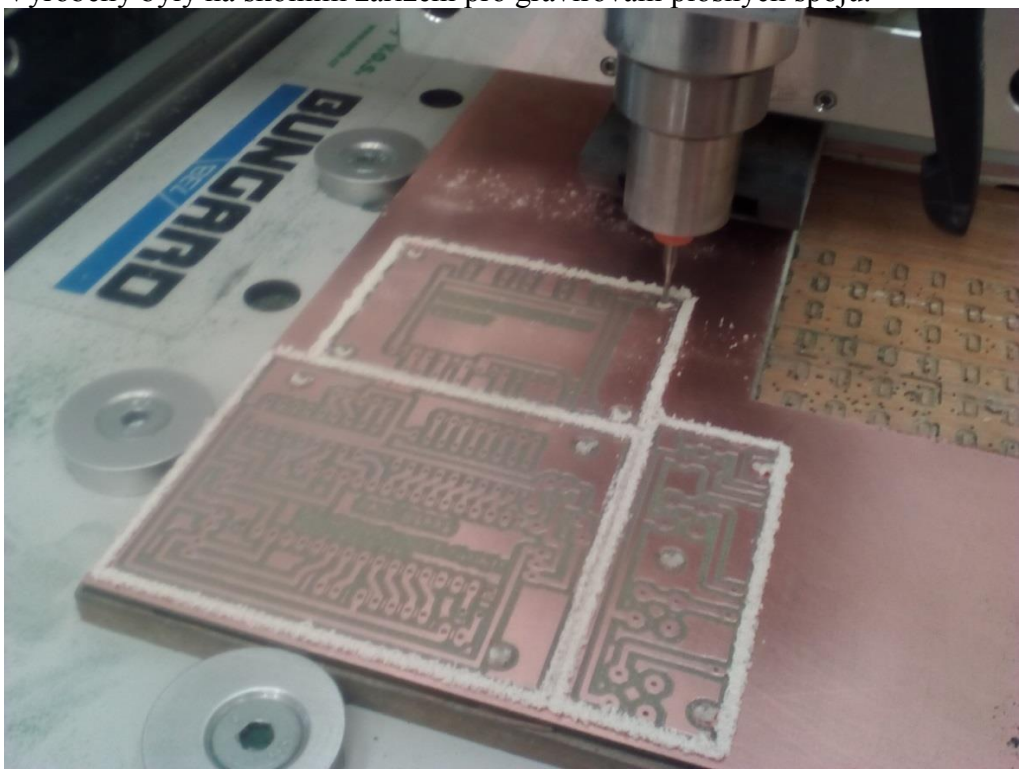


Obrázek 62 Plošný spoj - hlavní deska



Obrázek 63 Plošný spoj pro displej a tlačítka

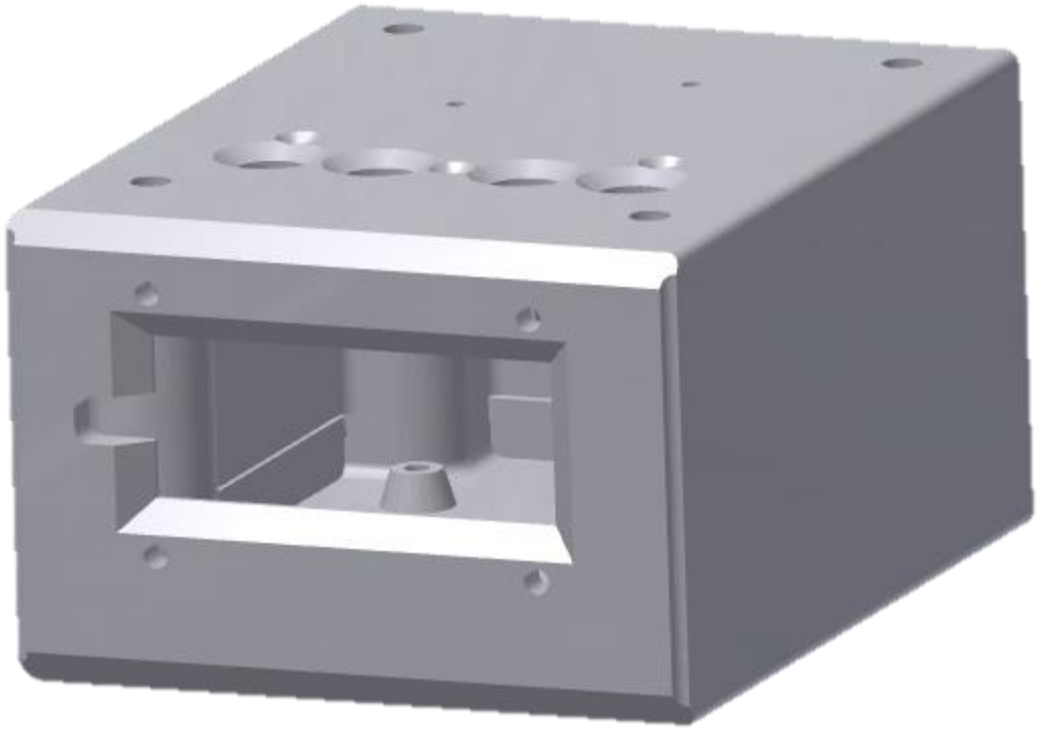
Vyrobeny byly na školním zařízení pro gravírování plošných spojů.



Obrázek 64 Výroba DPS

Krabička

Hotovou elektroniku (nebo jak se říká „střeva“) je vhodné usadit do nějaké krabičky. Buď se dá koupit, nebo vyrobit. Těch, co se se dají koupit je hodně druhů, tvarů a velikostí, ale málokdy budou vyhovovat úplně všem požadavkům, téměř vždy budou potřeba nějaké úpravy. Proto jsem se rozhodl využít toho, že naše škola je vybavena dvěma 3D tiskárnami a vyrobit si krabičku, kterou si sám navrhnu přesně, jak potřebuji.



Obrázek 65 3D model krabičky

Použité součástky

R1, R2	120Ω
R3	100Ω
R4	2kΩ
IO1	LM317T
IO2	7805
IO3, IO4	LB1290 + 2xDIL18PZ
IO5	Arduino Nano
J1, J4	PSH02-05PG

J2	PSH02-08PG
J3, J5,…J9	PSH02-08PG
J10	PC-GM2.1
J11, J14	PFH02-05P + 10xPFF02-01F
J12	PFH02-08P + 8xPFF02-01F
J13, J15,…J19	PFH02-02P + 12xPFF02-01F
SW1,…SW4	P-DT6
SW5	P-B144
JP1, JP2	BL15G
SP1	OZNAČENÍ
Displej	IVL2-7/5
RTC modul	ZS-042
Chladiče	

Tabulka 2 Seznam součástek

Software

V této části chci stručně popsat, jak funguje struktura programu.

Kód je pro přehlednost rozdělen do procedur, které jsou postupně volány ve smyčce *loop*.

```
void loop(){          // OPAKUJÍCÍ SE PROGRAMOVÁ SMYČKA
 // procedura pro zobrazení času z RTC modulu
 displayTime();

 // testování zmáčknutí tlačítek pro nastavení času a budíku
 if((digitalRead(A0) == LOW) || (digitalRead(A1) == LOW)){
     // pokud ano -> procedura nastavování času tlačítky
     checkButtons();
    }
 // zjišťování zda je spínač budíku sepnut
 int x = analogRead(A6);
 if(x < 800 && x > 600){
  // pokud ano -> zkouška zda se aktuální čas rovná času budíku
  checkAlarm();
 }
}
```

Obrázek 66 Hlavní smyčka

- *displayTime()* – tato procedura čte data z RTC obvodu DS3231 a obsahuje vlastní pod-procedury pro samotnou komunikaci a práci s přijatými daty, převody číselných soustav a práci s displejem, ke kterému mi pro zjednodušení pan Kotrba poskytl jeho vlastní knihovnu pro práci se sedmi segmentovým displejem
- *checkButtons()* – v této části je zjišťováno stisknutí tlačítek na vstupech A0, A1 (čas, budík), pokud je jedno z nich stisknuto a drženo, program přejde do smyčky, ve které jsou po jedné zvyšovány proměnné hodin a minut času a budíku. K přerušení smyčky dojde uvolněním drženého tlačítka, při tom se nově zadaná data o čase odešlou do RTC – nastaví se čas.
- *checkAlarm()* – než program přejde do této procedury je zkoušeno sepnutí posuvného spínače, když je sepnut program v této proceduře porovná čas, na který je budík nastaven s aktuálním časem, v případě shody skočí do smyčky se zvoněním. Tímto tlačítkem je tedy budík zapínán a vypínán.

Co se týče budíku, tak se mi bohužel nepodařilo zprovoznit fungování s vnitřními budíky v obvodu DS3231, takže budík je realizován výše popsanou procedurou v Arduinu.

Závěr

V práci jsem představil některé způsoby, jak využít mikrokontrolér (konkrétně Atmel ATmega328P v platformě Arduino) k měření času a úspěšně je realizoval. Dále jsem většinu zkušeností z těchto úloh (například práci s funkcí millis, používání RTC modulu, multiplexní řízení displeje,…) využil při konstrukci vzorového výrobku – digitální hodiny s budíkem. Tento úkol jsem splnil, jen se mi bohužel nepodařilo použít vnitřní budík, který obvod DS3231 nabízí, takže při odpojení napájení se jejich nastavení vymaže, nicméně čas zůstává zachován.

Zdroje informací

http://www.itnetwork.cz/hardware-pc/arduino/arduino-lcd-display/#komentare
https://www.arduino.cc/
https://arduino.cz/
https://en.wikipedia.org/wiki/Vacuum_fluorescent_display
https://en.wikipedia.org/wiki/Seven-segment_display
https://en.wikipedia.org/wiki/Light-emitting_diode
http://www.ikeytop.com/product.asp?ID=4
https://en.wikipedia.org/wiki/Dot-matrix_display
https://en.wikipedia.org/wiki/Liquid-crystal_display
http://www.winstar.com.tw/products/character-lcd-display-module/dot-matrix-lcd.html
http://linksprite.com/wiki/index.php5?title=16_X_2_LCD_Keypad_Shield_for_Arduino
http://fyzika.jreichl.com/main.article/view/523-displej-z-kapalnych-krystalu-lcd
https://www.gme.cz/data/attachments/dsh.513-139.1.pdf
http://randomnerdtutorials.com/guide-for-real-time-clock-rtc-module-with-arduino-ds1307-and-ds3231/
https://www.lammertbies.nl/comm/info/I2C-bus.html
https://www.aliexpress.com/popular/lcd-clock-for-car-motorcycle.html
https://www.etsy.com/market/nixie_tube_clock
http://www.explainthatstuff.com/how-vacuum-fluorescent-displays-work.html
https://web.archive.org/web/20070208014518/http://hem.passagen.se/communication/vfd.html
http://www.datasheetcatalog.com/datasheets_pdf/L/B/1/2/LB1290.shtml