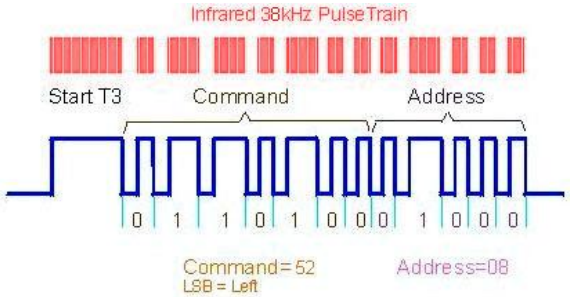
Analýza protokolu dálkového ovladače

Specifikace

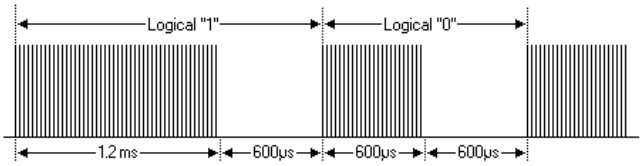
Dálkové ovladače spotřební elektroniky kódují přenášená data rozličnými protokoly. Chceme-li použít existující dálkový ovladač k ovládání vlastního zařízení, musíme přesně vědět, jak ovladač komunikuje. Jediným spolehlivým způsobem jak toto zjistit, je odposlechnout signál „ze vzduchu“ a pak ho podrobit analýze – zjistit délku jednotlivých pulzů a najít znaky typické pro daný protokol. Program graficky zobrazí strukturu paketu přijatého z dálkového ovladače. Jednotlivé pulsy vykreslí do grafu podobně jako osciloskop. Poté paket porovná s databází známých protokolů a pokusí se ho rozpoznat a dekódovat. Na závěr umožní uživateli uložit naměřená data do souboru pro pozdější použití.

Stručný nástin principu přenosu informace infračerveným paprskem

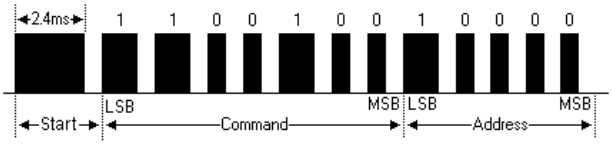
Snad všechny ovladače spotřební elektroniky používají k přenosu informace modulovaný infračervený paprsek. Modulace signálu zajišťuje zvýšenou odolnost proti rušivým vlivům – například jiným zdrojům IR světla (slunce, žárovka). Přenášená informace je kódována jako sled modulovaných pulsů a mezer různé délky. Na ilustračním obrázku je znázorněn paket protokolu Sony SIRC[1]:



Protokol definuje, kde začíná zpráva, jakým způsobem jsou kódovány bity a jak mají být tyto bity interpretovány příjemcem. Například výše uvedený protokol kóduje bit „1“ jako puls délky 1200µs za kterým následuje mezera 600µs. Bit „0“ je kódován jako puls a mezera stejné délky 600µs. Viz ilustrace:



Vysílání je zahájeno startbitem poté následuje 7 bitů nesoucích informaci o stisknutém tlačítku na ovladači a po nich 5 bitů jednoznačně identifikujících typ zařízení, pro které je paket určen.



Více o teorii IR přenosu a používaných protokolech na stránkách na http://www.sbprojects.com/knowledge/ir/ir.htm (anglicky)

[1] http://www.sbprojects.com/knowledge/ir/sirc.htm

Obecné vlastnosti IR protokolů a jejich reprezentace v programu

Chceme-li algoritmizovat rozpoznávání protokolů, musíme zvolit vhodnou reprezentaci všech vlastností kterými jsou protokoly určeny. Zaměřme se nejprve na znaky které jsou pro všechny běžně používané protokoly společné. [2]

1. Protokoly jsou kódovány jako sled různě dlouhých pulzů a mezer.
2. Vysílání je zahájeno startbitem. To je první puls (nebo pusly) které indikují začátek paketu.
3. Bity „0“ a „1“ jsou jednoznačně určeny a dají se popsat jako uspořádaná k-tice pulsů a mezer definované délky.
4. Význam jednotlivých bitů v paketu je jednoznačně určen protokolem.
5. Paket daného protokolu má vždy stejnou strukturu (v důsledku je tedy v jednom paketu přenášen vždy stejný počet bitů)

Nyní naopak uvedu některé znaky kterými se protokoly vzájemně odlišují:

1. Délka, případně struktura startbitu
2. Způsob kódování bitů (délky mezer a pulsů)
3. Počet přenášených bitů a jejich význam
4. Endianita (MSB, LSB)
5. Specifické oddělovače mezi jednotlivými úseky zprávy

Na základě výše uvedených vlastností jsem navrhnul způsob univerzálního popisu protokolů v jazyce XML. Následující příklad ukazuje úplný popis vlastností již zmiňovaného protokolu Sony SIRC:

```
<protocol name="Sony SIRC">
  <description url="http://www.sbprojects.com/knowledge/ir/sirc.htm">
  The SIRC protocol uses a pulse width encoding of the bits. The pulse representing a logical
"1" is a 1.2ms long burst of the 40kHz carrier, while the burst width for a logical "0" is
0.6ms long. All bursts are separated by a 0.6ms long space interval. The recommended carrier
duty-cycle is 1/4 or 1/3.
  </description>

  <bitset> <!– describes bit coding of the protocol -->
    <bit value="1"> #1200 _600 </bit> <!– prefix # for pulse, _ for space -->
    <bit value="0"> #600 _600  </bit> <!– timing in us -->
  </bitset>

  <packet>
    #2400 _600 <!-- startbit -->
    <data name="command" length="7" bitorder="LSB" />
    <data name="address" length="5" bitorder="LSB" />
  </packet>

</protocol>
```

V sekci `<bitset>` je popsán způsob, jakým protokol kóduje bity. Prefix „#“ resp. „_“ před číslem určuje, zda se jedná o puls nebo mezeru. Bezprostředně následující číslo udává délku takového pulsu nebo mezery v jednotkách µs.

Sekce `<packet>` popisuje strukturu celého protokolového paketu. Může obsahovat informace o specifických oddělovačích (jako je např. startbit, stopbit a podobně…). Oddělovače charakterizují protokol a slouží k rozpoznání začátku (a v ojedinělých případech i konce) paketu. Jsou zadány stejným způsobem jako bity.

Naproti tomu tag `<data name="command" length="7" bitorder="LSB" />` označuje skupinu bitů délky `length` nesoucí užitečnou informaci. Tou může být například kód stisknuté klávesy, adresa

[2] Předpokládáme, že zpracovávaný signál je již demodulovaný, a to buď přímo hardwarem přijímače, nebo vhodným algoritmem. Více o problematice získávání signálu v příloze.

přijímacího zařízení nebo jiná data specifická pro daný protokol (např. tzv. *toggle bit*[3]). Parametrem `name` je textový řetězec vystihující charakter dat. Parametr `bitorder` určuje endianitu bitů (v úvahu připadají dvě hodnoty: `MSB` a `LSB`).

Vstup programu

Program bude zpracovávat datový tok (stream). Zdrojem dat bude primárně *přijímač* (více v příloze), ale také bude možné zpracovávat data zachycená v souboru. Program tedy bude pracovat ve dvou režimech:

1. Analýza signálu z přijímače
2. Analýza dat zachycených v souboru

Algoritmus rozpoznávání protokolu

1. Najdi nejbližší další vzestupnou hranu v datovém toku
2. Vyber specifikaci protokolu
   a. procházej postupně všechny položky sekce <packet> a porovnávej je s datovým tokem. Při nalezení prvního konfliktu GOTO 3
   b. Úplná shoda protokolu s datovým tokem (protokol rozpoznán)
   c. Dekóduj protokol. BREAK
3. Dokud nebyly prověřeny všechny protokoly, GOTO 2
4. Dokud nebyly prověřeny všechny vzestupné hrany, GOTO 1

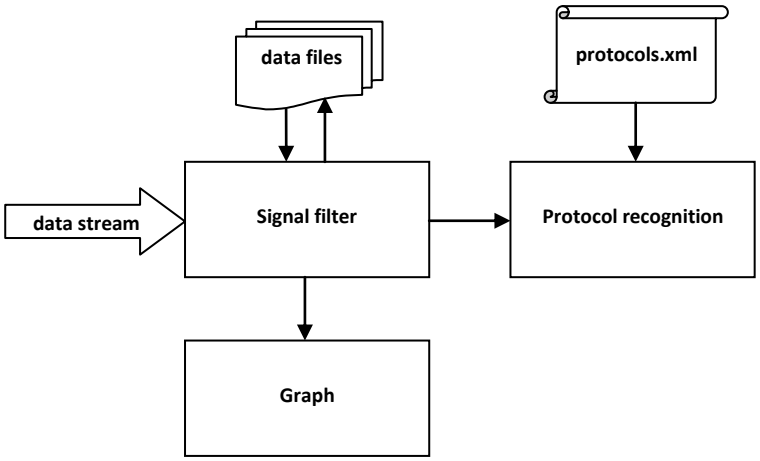
Protože v definici každého protokolu je hned první kritérium – délka startbitu – velmi ostré, je zajištěno účinné ořezávání. V důsledku pak algoritmus postupně prochází datový tok a hledá v něm výskyty pulsů definovaných délek (startbitů). Teprve po nalezení startbitu se pokusí aplikovat na následující pulsy složitější pravidla ze sekce <packet>. Časová složitost algoritmu je lineární a závisí na počtu protokolů v databázi. Algoritmus je možné ještě zefektivnit použitím heuristického pravidla: Z praxe víme, že mezi jednotlivé pakety vysílač vkládá mezery, a ty jsou obvykle o mnoho delší než paket sám. Můžeme tedy první krok optimalizovat následovně:

1. Najdi nejbližší vzestupnou hranu, před kterou je mezera délky alespoň *n*. (kde *n* bude empiricky zjištěná konstanta)

[3] Toggle bit je typický pro populární protokol RC-5. Tento bit je invertovaný při každém stisknutí klávesy, čímž umožňuje příjemci rozpoznat, zda je klávesa držena stisknuta, anebo zda ji uživatel mačká opakovaně.

Implementace

Aplikace bude naprogramována ve vývojovém prostředí Delphi. Z důvodu lepší udržovatelnosti a přehlednosti bude rozdělena do několika víceméně nezávislých zapouzdřených částí (objektů), jak je znázorněno níže:



Signal filter

Zajišťuje výběr zdroje signálu (z přijímače, ze souboru) a umožňuje nahrávání signálu do souboru. Provádí základní filtraci datového toku – ořezává dlouhé mezery a potlačuje šum. Filtrovaná data předává dalšímu modulu k dekódování. Poskytuje základní informace o charakteristice IR přenosu, jako je například prodleva mezi jednotlivými pakety. Rozhoduje, jaká část streamu se vykreslí do grafu.

Protocol recognition

Modul zajišťující rozpoznání a dekódování protokolů. Načítá XML soubor s definicí protokolů. (S největší pravděpodobností použiju knihovnu MSXML 6.0). Po úspěšném rozpoznání protokolu zobrazí informace uložené v sekci `<description>` a přehledně vypíše dekódovanou informaci.

Graph

Vykresluje průběh signálu do grafu (podobně jako osciloskop). Ke každému pulsu a mezeře doplní jako popisek délku v ms. Graf může uživateli sloužit například k analyzování dosud nepopsaných protokolů.

Závěr

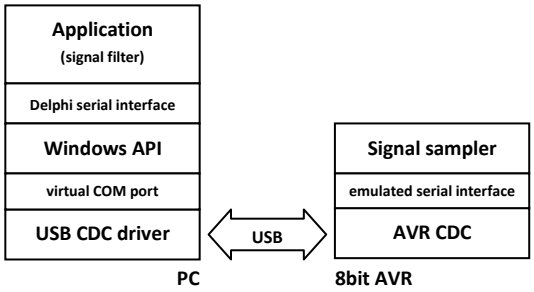
Při návrhu programu je kladen důraz zejména na snadnou rozšiřitelnost databáze protokolů. Proto jsem pro specifikaci vlastností protokolů zvolil formát XML. Struktura XML souboru byla navržena co nejintuitivněji, aby mohl nové protokoly přidávat i sám uživatel. Program je určen především zájemcům o amatérskou robotiku. Nejsnazší způsob, jak řídit zařízení vlastní výroby (robota) bezdrátově, je použít nějaký z dálkových ovladačů co se válí doma… Tento program automatizuje proceduru zjišťování protokolu a také pomůže odhalit, které tlačítko na ovladači má jaký kód. Nabízí se ale i další použití; v principu může být program rozšířen o modul, který umožní ovládat počítač libovolným dálkovým ovladačem.

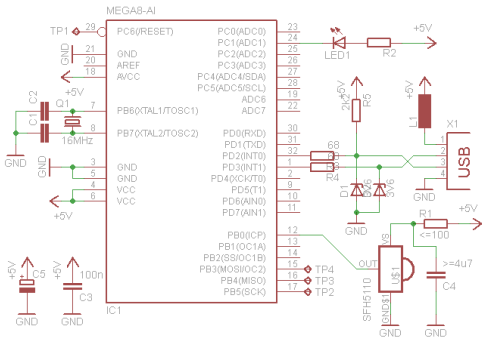
Příloha – možnosti realizace přijímače

Tato příloha se zabývá možnostmi získání signálu z dálkového ovladače – tedy realizací vlastního přijímače. Jelikož se jedná o problematiku, která svou složitostí přesahuje rámec zápočtového projektu z Programování I, rozhodnul jsem se oddělit ji od analýzy samotného zápočtového programu. Byla by však škoda nezmínit se, jakými způsoby může výše popsaný program získávat data z reálného světa. Ostatně právě to nakonec vymezí praktickou použitelnost programu. Proto ve stručnosti nastíním způsoby získávání signálu alespoň v této příloze.

1. Vlastní HW řešení

První uvažovanou možností bylo sestavit vlastní HW přijímač, který se k počítači připojí pomocí sběrnice USB. Přijímač by se skládal z mikrokontroléru a specializovaného obvodu pro příjem IR signálu. Výhodou je, že tento obvod přímo zajišťuje demodulaci IR signálu. Mikrokontrolér by tedy pouze určitou frekvencí samploval demodulovaný signál a data by posílal přes rozhraní USB do počítače.

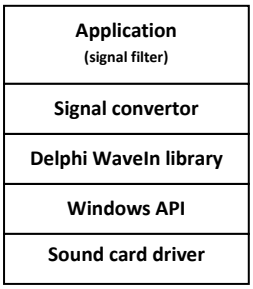


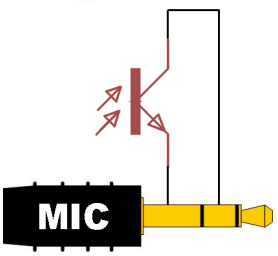


Jak je vidět ze schématu, takový přijímač je robustní a bohužel také velmi složitý jak po hardwarové tak implementační stránce.

2. Využití vstupu zvukové karty

Je mi jasné, že má-li být program použitelný pro co nejširší okruh uživatelů, je potřeba nabídnout jim jednoduchý přijímač snadno vyrobitelný v domácích podmínkách. Následující schéma ukazuje minimalistický přijímač IR signálu pro PC. Přijímač tvoří pouze jedna součástka – a tou je obyčejný fototranzistor. Fototranzistor je zapojen přímo do zvukové karty do vstupu pro mikrofon. Místo audiosignálu z mikrofonu tak tedy vlastně zvuková karta sampluje optický signál z dálkového ovladače. Tento signál musí být samozřejmě ještě demodulován a převeden do formátu datového toku, který vyžaduje aplikace. K tomu bude sloužit vrstva Signal convertor.





Vzhledem k univerzálnosti a menším nárokům na uživatele budu preferovat druhé řešení přijímače.