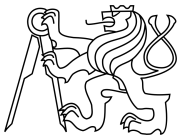
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Návrh a implementace modemového kabelu Jack/RS-232
Student:	David Ovčačík
Vedoucí:	Ing. Kateřina Hašlarová
Studijní program:	Informatika
Studijní obor:	Počítačové inženýrství
Katedra:	Katedra číslicového návrhu
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Navrhněte a sestrojte prototyp modemového kabelu schopného přenášet sériová data. Kabel bude zakončen konektory RS-232 a 3,5mm Jack. Dodržujte standardy a obecné praktiky typické pro tyto konektory – rozumíme analogový/digitální signál, napětí, způsob použití v současných zařízeních. Kabel bude primárně používán se zařízeními splňujícími technické požadavky OS Android. Naprogramujte jednoduchou testovací aplikaci pro OS Android.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

doc. Ing. Hana Kubátová, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 11. října 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra číslicového návrhu

Bakalářská práce

Návrh a implementace modemového kabelu Jack/RS-232

David Ovčačík

Vedoucí práce: Ing. Kateřina Hašlarová

31. prosince 2016

Poděkování

Tímto chci poděkovat vedoucí mé práce, Ing. Kateřině Hašlarové, za vstřícnost a trpělivost ve všech fázích tohoto projektu. Dále děkuji Ing. Miroslavu Skrbkovi, Ph.D. za odborné konzultace a rady při řešení problémů; kamarádům Martinovi a Lukášovi za pomoc při pájení a implementaci aplikace pro Android; firmám *Transfer Multisort Elektronik Sp. z o.o.* a *GM electronic, spol. s r. o.* za bezedný zdroj elektronických součástek a své rodině za neustálou psychickou podporu a motivaci v průběhu celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 31. prosince 2016 .....

Odkaz na tuto práci

Ovčačík, David. *Návrh a implementace modemového kabelu Jack/RS-232.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Předmětem této práce je návrh a implementace modemového kabelu s rozhraními *RS-232* a *Audio Jack*. Účelem tohoto kabelu je (prozatím) jednosměrný přenos sériových dat mezi digitální periferií a mobilním zařízením. Specifickým znakem řešení je A/D převod signálu a tudíž zachování obvyklého způsobu použití obou konektorů. Součástí implementace je zhotovení funkčního prototypu a vývoj softwarových knihoven pro *OS Android*.

Klíčová slova RS-232, PWM, UART, modulace, mikrokontrolér, signál

Abstract

The topic of this thesis is the design and implementation of a modem cable with *RS-232* and *Audio Jack* connectors. The purpose of this cable is (for now) single-directional serial data transmission between a digital periphery and a mobile device. The proposed solution is specific for its A/D signal conversion, thus preserving the ordinary way of using both ends. Part of the implementation includes constructing a functional prototype and development of a software library for the *Android OS*.

Keywords RS-232, PWM, UART, modulation, microcontroller, signal

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Mobilní zařízení používáme v dnešní době stále častěji, na komplexnější úkony ve spojení s větším spektrem periferií než kdy předtím. Tablety a chytré telefony se stávají nejen zábavním centrem, ale rovněž i ovládacím terminálem pro řízení různých elektronických systémů. Každodenním používáním se postupně stávají náhradou stolních počítačů. Tím zároveň rostou očekávání a požadavky uživatelů na podporované softwarové funkce, výdrž baterie a možnosti konektivity.

S pokročilejšími funkcemi a vyššími nároky na výpočetní výkon roste spotřeba energie. Aby bylo zařízení schopné vydržet celý den neustále v provozu, musíme jej po celou dobu napájet ze sítě nebo jiného zdroje elektrického napětí. Tím bohužel obsadíme často jediný konektor umožňující přenos dat, protože je hardwarově spojený s napájecím konektorem. Kvůli tomuto faktu je mnohdy znemožněna lokální datová komunikace s tímto zařízením.

V této práci jsou dokumentovány kroky, které vedly k úspěšnému nalezení nového způsobu komunikace s mobilními zařízeními. Navržené řešení poskytuje alternativní využití audio konektoru jako přijímače sériových dat. Realizace přenosu dat přes toto rozhraní je specifická svým kódováním a charakteristikou nosného signálu. Jelikož je Audio Jack používán primárně pro přehrávání a záznam zvuku, musí být i datový signál podobný zvukovému signálu. Tyto požadavky byly splněny modemovým článkem uprostřed kabelu.

Modulace signálu je v tomto případě nezbytně nutná, jak prokázala analýza. Sériový protokol RS-232, který je pro komunikaci používán, není pro zvolený konektor vhodný. Důvody jsou blíže popsány v sekci 2.2.1. Řešením tohoto problému se navržený kabel odlišuje od existujících řešení, jejichž funkcionalita nevyhovuje kladeným požadavkům.

Mikrokontrolér *ATtiny4313* uvnitř kabelu aktivně zasahuje do přenášeného signálu a generuje výstup, který je možno přijímat mobilním zařízením skrz audio konektor. Tato komponenta je jádrem celého systému a tvoří jedno z hlavních témat této bakalářské práce.

Struktura práce

První kapitola této práce uvádí vytyčené cíle a rekapituluje požadavky na řešení. V druhé kapitole jsou popisována možná řešení zadání práce a konečné specifikování technických parametrů. Zároveň je zde obsažena diskuze k existujícímu řešení a jeho porovnání s navrhovaným řešením. Třetí kapitola je analýzou hardwarových požadavků a jejich souvislostí se softwarovou částí projektu. Detailní popis implementace hardwarové i softwarové části je uveden v kapitole čtvrté. Pátá kapitola komentuje průběh testování a shrnuje výsledky. Poslední kapitola uvádí možnosti rozšíření a budoucích prací na projektu.

Kapitola 1

Cíl práce

Cílem této bakalářské práce je najít a navrhnout způsob, jakým propojit dvě zařízení bezpečným komunikačním kanálem. Požadavků není mnoho, zato jsou velmi specifické. Komunikace musí být vedena protokolem RS-232, který je implementován na straně periferního zařízení vystupujícího jako vysílač dat. Roli přijímače v této situaci hraje mobilní zařízení, u kterého není povoleno využít digitální microUSB konektor.

Zdrojem tématu této práce je hledání řešení propojení mobilního zařízení se stolní digitální váhou. Práce si však klade za cíl najít a poskytnout univerzální řešení komunikace s mobilními zařízeními protokolem RS-232.

1.1 Požadavky na řešení

Výsledné řešení problému je akceptovatelné pouze při splnění následujících požadavků. Nejdůležitělší podmínkou je použití sériového konektoru jakožto zdroje dat. Přes toto rozhraní bude probíhat komunikace protokolem RS-232, avšak následná transformace je přípustná (a jak ukázala analýza i nezbytná). Toto je jedním ze dvou omezení, které je bezpodmínečně neměnné.

Druhým požadavkem na řešení propojení je použití jiného konektoru, než je digitální microUSB na straně mobilního zařízení. Alternativám nejsou přiřazeny žádné priority, vhodnost každé z nich bude diskutována a určena v následující kapitole.

Kapitola 2

Rešerše

Tato kapitola uvádí rozhodnutí, která byla učiněna s cílem blíže specifikovat zadání práce a nalézt nejlepší podmínky pro řešení problému. Analýza technických požadavků a možností probíhala ve spolupráci se zadavatelem bakalářské práce. Prvotním požadavkem a zdrojem této práce bylo propojení digitální váhy s mobilním zařízením. Omezujícími podmínkami bylo nutné využití protokolu RS-232 na straně digitální váhy a propojení skrze jiný konektor než microUSB na straně mobilního zařízení. Předpokládáme totiž, že microUSB konektor bude permanentně obsazen připojením ke zdroji napětí.

2.1 Možnosti řešení

2.1.1 Bluetooth

Bluetooth je bezdrátový standard přenosu dat, který byl vyvinut jako nástupce protokolu RS-232. Jedná se o metodu bezdrátové komunikace na rádiových frekvencích $2{,}4 - 2{,}485\,\mathrm{GHz}$. Specifikace také definuje shlukování zařízení do sítí pro podporu vícebodového přenosu dat. Použití Bluetooth se tedy zdálo být nejlepší volbou, protože je rovněž podporován v široké řadě mobilních zaířzení.

Komplikací ale může být zákaz používání rádiových vln v některých oblastech či prostorách kvůli rušení okolních přijímačů. Při práci s elektronikou citlivou na změny elektromagnetického pole by bylo použití Bluetooth nepřijatelné.

Největší překážkou při používání technologie Bluetooth je však nutnost certifikace výrobků pro komerční použití. Každé zařízení, které implementuje tento standard bezdrátové komunikace, musí projít sérií zdlouhavých a náročných testů před tím, než je schváleno pro použití v komerční sféře. Tyto požadavky jsou příliš vysoké pro tento projekt a proto bylo od možnosti použití Bluetooth upuštěno.

2.1.2 USB rozdělovač

Dalším potenciálním řešením se naskytlo použití USB rozdělovače, kterým by se zvýšil počet možných připojení skrze microUSB konektor. Toto řešení by bylo technicky i finančně úsporné, protože vyžaduje velmi málo komponent. Konkrétně pouze USB rozdělovač a adaptér RS-232 na USB.

I s takto triviálním propojením ale narážíme v oblasti kompatibility mobilních zařízení. Softwarové knihovny implementující přenos sériových dat jsou totiž závislé na módu *USB On The Go (OTG)* [1]. Ten není v některých mobilních zařízeních podporován a proto není toto řešení přijatelné.

2.1.3 Audio konektor

Po vyřazení bezdrátového připojení a microUSB konektoru už zbývá jen možnost využít audio konektor, který je dostupný na většině současných telefonů a tabletů. Od ostatních připojení se ale zásadně liší tím, že jej tyto zařízení používají výhradně k přehrávání a záznamu zvuků, tedy práci s analogovým signálem. Tato skutečnost bude mít silný dopad na náročnost implementace a technologii přenosu dat.

Na druhou stranu s sebou toto řešení nenese předchozí problémy, protože nevyužívá bezdrátovou komunikaci a dostupnost konektoru je velmi dobrá. Obsluha pravděpodobně nebude smět připojovat sluchátka pro poslech hudby. Navrhovaným kabelem sice eliminujeme jediný způsob, jak drátově připojit externí reproduktor, ale původní funkcionalita může být do jisté míry nahrazena vestavěnými komponentami. Naprostá většina dnešních mobilních zařízení totiž obsahuje interní reproduktor a mikrofon.

Se zadavatelem práce jsme se shodli, že využití audio konektoru je vzhledem k omezení ostatních řešení nejvhodnější.

2.2 Existující řešení

Po zhodnocení možností a výběru jedné z nich bylo úkolem vyhledat, zda již takové řešení neexistuje, a případně konfrontovat jeho funkcionalitu s našimi požadavky. Sériový přenos dat protokolem RS-232 je i v dnešní době díky snadné implementaci často používán pro komunikaci s periferiemi. Pokud budeme hledat kabel, který je osazen konektory pro přenos RS-232 → Audio Jack, nalezneme na trhu celkem snadno několik modelů.

Obrázek 2.1: Kabely realizující propojení konektorů DE-9 a Audio Jack

Kabely uvedené v obrázku 2.1 používají pro sériovou komunikaci např. digitální kamery, programovatelné kalkulačky a některé televize pro servisní přístup. V době psaní této práce se jejich průměrná cena pohybuje okolo 250 Kč za kus. Po tomto průzkumu se může zdát, že není potřeba znovu navrhovat tento typ propojení. Ovšem žádné z těchto existujících řešení nesplňuje důležitou podmínku vyplývající ze zvoleného způsobu propojení.

Cílem je navrhnout kabel, který převádí digitání signál na analogový. Výše zobrazené vzorky realizují pouhé propojení vodičem mezi jednotlivými kontakty. V žádném případě aktivně nezasahují do přenášených signálů a tudíž nejsou pro požadované použití vhodné.

2.2.1 Porovnání práce s existujícím řešením

Klíčovým rozdílem mezi existujícím a navrhovaným řešením je schopnost modulovat přicházející digitální signál na analogový, čitelný na mobilním zařízení skrze audio konektor. Použití některého z dříve vyobrazených kabelů není v mobilním zařízení možné, protože audio konektor slouží výhradně k přehrávání a záznamu zvuků. Vnitřní interpretace signálu je tedy zcela rozdílná od sériového protokolu RS-232. Použití kabelu, který je pouhým propojením kontaktů na obou koncích, by nejspíše selhalo kvůli minimálně jedné z těchto příčin:

1. Napětí používané pro komunikaci protokolem RS-232 je příliš vysoké a velmi nebezpečné pro jakékoliv mobilní zařízení. Audio konektor je sice pravděpodobně chráněn proti přepětí nebo zkratu, ale dlouhodobé vystavení napětí až 12 V by zajisté mohlo konektor a přidružené obvody trvale a nenávratně poškodit.

2. Primární použití audio konektoru není přizpůsobeno přivádění stejnosměrného napětí na vstup. Signál typicky generovaný zapojeným mikrofonem má podobu zvukových vln, které zachycuje, a tudíž má střídavé napětí. Uvnitř konektoru jsou obvody filtrující vstupní signál, kvůli kterým je stejnosměrná složka napětí uzeměna a odstraněna. Tímto předzpracováním by se přijímaná sériová data zřejmě stala nečitelnými.

3. Příliš velká přenosová rychlost sériových dat by mohla způsobit zkreslení nebo nečitelnost přijímaných dat. Na většině mobilních zařízení s operačním systémem *Android* je maximální vzorkovací frekvence audio signálu 44,1 kHz. Tím je položeno omezení maximální frekvence signálu a tím pádem i přenosové rychlosti. Z těchto důvodů je pro splnění požadavků potřeba navrhnout a sestavit zcela nový modemový kabel, který aktivně transformuje signál do podoby bezpečné a čitelné pro mobilní zařízení.

Kapitola 3

Analýza

3.1 Analýza fyzického rozhraní RS-232

RS-232 je standard pro sériový přenos dat poprvé publikován v roce 1962. Definuje signály, časování a napětí pro spojení koncové stanice a modemu. Původně určen k telefonní komunikaci je dnes tento standard přenosu používán jako univerzální propojení v podstatě jakýchkoliv dvou zařízení nezávisle na jejich účelu použití.

Pojmem RS-232 je často nesprávně označován konektor typu DE-9 vyobrazený na obrázku 3.1, který je nejpoužívanějším v sériové komunikaci. Standard RS-232 sice žádný konektor specificky nepředepisuje, ale doporučuje použití DB-25, což je starší varianta DE-9 s 25-ti piny [2].

Obrázek 3.1: Konektor DE-9 typický pro sériové protokoly přenosu dat

Standard definuje mnoho signálů, které už dnes kvůli novějším technologiím ztrácejí význam, a proto jsou na zařízeních s konektorem DE-9 většinou implementovány už jen signály TxD, RxD a GND. Tyto tři tvoří dle RS-232 nejmenší množinu signálů realizujících obousměrnou sériovou komunikaci. V mnohých integrovaných obvodech a mikročipech jsou přítomny řadiče pro sériovou komunikaci s názvem UART, většinou pouze s těmito třemi signály. Data jsou posílána sériově po jednom vodiči od vysílače k přijímači a tudíž nehrozí žádné datové kolize.

Napětí datových signálů není dle standardu přesně určeno, ale je doporučen rozsah −5 V až −12 V pro logickou jedničku a +5 V až +12 V pro logickou nulu.

3.2 Analýza protokolu RS-232

Protokol RS-232 definuje průběh komunikace po lince sdílené mezi dvěma zařízeními. Data jsou vysílána po skupinách bitů a tvoří tak logické celky nazývané „*slova*“. Délka těchto slov může být různá a standardem je definována na rozsah 5 až 8 bitů. Délka se však nesmí během vysílání měnit, protože by s velkou pravděpodobností došlo ke špatné interpretaci dat na straně přijímače.

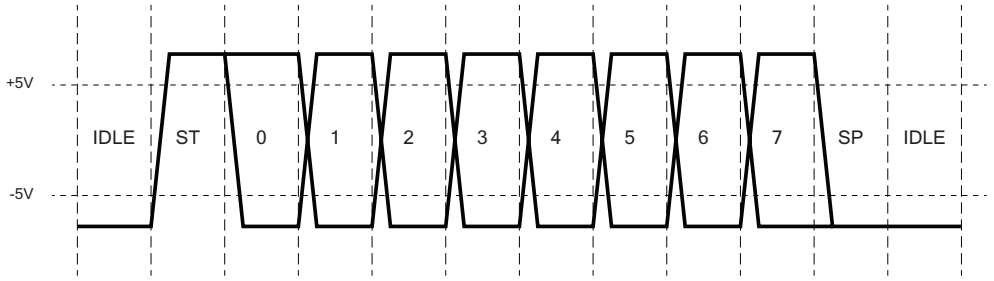
Na obou zařízeních je pro správný průběh komunikace nutné nastavit shodnou přenosovou rychlost. Tato rychlost je označována pojmem *„baudrate“* a de facto znamená počet přenesených bitů za sekundu, protože data mají pouze stavy „0“ a „1“.

RS-232 je protokol asynchronní komunikace, což znamená, že vysílač může začít vysílat data kdykoliv. Aby byl přijímač schopen zjistit, kdy jsou posílána data, zavádí protokol RS-232 koncept tzv. *„start bitu“*. V klidovém stavu je na lince konstantní napětí v rozmezí −5 V až −12 V signalizující logickou jedničku. Start bit má pevně definovanou logickou hodnotu „0“ a proto s každým vysláním start bitu dochází ke změně napětí na lince, kterou přijímač dokáže zaznamenat.

Po přijmutí start bitu synchronizuje přijímač svůj interní časovač dle přenosové rychlosti. Za start bitem následuje několik datových bitů, jejichž logické hodnoty jsou v pravidelných časových intervalech zaznamenávány.

Za datovými bity může následovat tzv. *„paritní bit“*, pokud je tak přenos nastaven. Paritní bit slouží jako kontrola správného obsahu datových bitů a je na straně vysílače počítán pro každé vysílané slovo zvlášť. Paritní bit může být nastaven jako sudý, nebo lichý a pro dané slovo značí sudý, resp. lichý počet logických jedniček v něm obsažených. V žádném případě však neslouží jako mechanismus pro zabezpečení dat před chybami nebo opravu chyb. Dalším nedostatkem paritního bitu je jeho schopnost detekovat pouze lichý počet chyb. Sudý počet nesprávně přenesených bitů nelze paritním bitem odhalit a tento případ musíme ošetřit jinak.

Posloupnost datových bitů a případně jednoho paritního je ukončena jedním až dvěma *„stop bity“*. Jediný význam stop bitu je změna stavu linky do klidového, tedy logické hodnoty „1“. S tímto zakončením dat můžeme ihned začít vysílat další slovo, protože je tímto zajištěna změna stavu s dalším start bitem. Pokud bychom stop bit nezavedli a poslední datový bit by měl hodnotu „0“, přijímač by nebyl schopen zjistit přítomnost ihned následujícího start bitu dalšího slova a přenos by selhal. V diagramu 3.2 je znázorněn typický časový průběh komunikace s osmi datovými bity a jedním stop bitem.



Obrázek 3.2: Průběh signálu TX/RX protokolu RS-232

3.3 Analýza Jack konektoru

Telefonní konektor neboli Jack je druh elektrického konektoru zpravidla používaný pro přenos analogových signálů. Zejména rozšířený je ve zvukovém průmyslu, kde je velmi často používán k připojení sluchátek a malých reproduktorů. Od svého navržení v 19. stol. se jeho podoba příliš nezměnila. K původní velikosti 6,35 mm se přidaly konektory o průměru 3,5 a 2 mm a počet kontaktů se zvýšil. Podstata ale zůstává stejná – konektor je rozdělen jednou nebo více přepážkami na jednotlivé kontakty. Detail konektoru se čtyřmi kontakty je uveden na obrázku 3.3.

Pro sestavení funkčního modemového kabelu je nezbytné použít 4-vodičový TRRS Jack, protože jako jediný umožňuje na mobilních zařízeních vstup i výstup najednou. Tato varianta konektoru má však dva různé standardy přiřazení signálů k vodičům, které mezi sebou nejsou kompatibilní. Jedná se o *Open Mobile Terminal Platform (OMTP)* a *Cellular Telephone Industries Association (CTIA)*, rozšířený na Americkém trhu. Vzájemně se liší v přiřazení signálů MIC a GND. Problém této nekompatibility se dá v koncovém zařízení vyřešit zapojením adaptéru OMTP na CTIA, nicméně kabel bude prozatím sestaven pouze v jedné konfiguraci. Pro sestavení prototypu je vybrán standard CTIA, protože jediné mobilní zařízení k dispozici pro vývoj je osazeno právě tímto konektorem.

CTIA je standard pro stereofonní audio konektor, na kterém jsou jednotlivé kontakty po řadě přiřazeny signálům pro levý stereo kanál (LA), pravý stereo kanál (RA), uzemění (GND) a poslední pro napájecí napětí mikrofonu (MIC).

Data vstupující do mobilního zařízení skrz audio konektor jsou primárně interpretována jako zvukový záznam. Kvůli tomuto je signál ještě před digitalizací filtrován, což v jistém smyslu usnadňuje implementaci. V opačném případě by bylo nutné zapojit RC článek, který by napětí vstupujícího signálu posunul do záporných hodnot, aby se skutečně podobal zvukovému záznamu.

Obrázek 3.3: 3,5mm TRRS audio konektor

3.4 Analýza zvukových dat

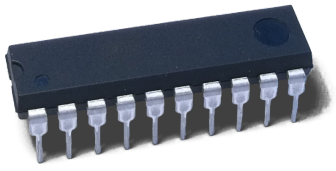
Zvukový záznam může mít doslova libovolnou podobu. Jedinou podmínkou je, že jakýkoliv signál vstupující skrz audio konektor musí mít charakteristiku výstupu mikrofonu – střídavé napětí. Protože je signál při záznamu filtrován, stejnosměrná složka jeho napětí je odstraněna a v zaznamenaných datech se neprojeví. Pokud budeme chtít komunikovat s mobilním zařízením přes jeho audio konektor, musíme generovat signál se střídavým napětím.

Dále je třeba vzít v úvahu způsob, jakým se signál digitalizuje pro použití v softwaru. V naprosté většině mobilních zařízení se používá PCM kódování s maximální vzorkovací frekvencí 44,1 kHz [3]. Toto představuje horní limit přenosové rychlosti. Nyquistův-Shannonův teorém tento limit nadále posouvá na maximálních 22,05 kHz a v praktickém použití to bude pravděpodobně ještě méně.

Mezi jinými omezeními audio konektoru také stojí elektrické napětí a proud generovaného signálu. Příliš vysokým příkonem by se obvody pro záznam zvuků mohly nenávratně zničit.

3.5 Analýza hardwaru

Z předešlých poznatků je zřejmé, že pouhé propojení dvou zařízení skrze sériový port a audio konektor není možné. Mezi tyto rozhraní je nutné vložit prostřední člen vykonávající převod signálu z jedné formy do druhé. Tímto aktivním členem může být např. některý z mikrokontrolérů řady *AVR* výrobce *Atmel*.

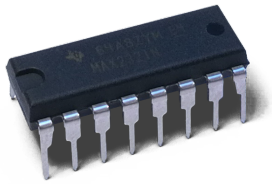


Obrázek 3.4: Mikrokontrolér *AVR ATtiny4313-PU*

Po důkladném zhodnocení byl vybrán model *ATtiny4313* vyobrazený na obrázku 3.4, protože je levný a zároveň poskytuje všechny funkce potřebné pro splnění požadavků práce. Konkrétně jde o nízké provozní napětí (nejméně až 1,8 V), obvody UART pro sériovou komunikaci a časovače s PWM výstupem [4]. Maximální taktovací frekvence není tak důležitým kritériem při rozhodování, protože již zjištěná nejvyšší přenosová rychlost dat 22,05 kHz je řádově mnohem nižší než standardní taktovací frekvence těchto mikrokontrolérů.

Prototyp modemového kabelu bude sestaven s použitím čipu v klasickém PDIP pouzdru, ale pro konečné sestavení bude díky menším rozměrům vhodnější použít VQFN pouzdro. Funkčně se tyto dva typy vzájemně nijak neliší, rozíl je pouze ve velikosti a uspořádání pinů.

Tento čip ale sám o sobě není schopen plnohodnotné komunikace přes protokol RS-232. Hlavním problémem je příliš vysoké standardní napětí signálů, které by mohlo čip poškodit. Sériová komunikace na úrovni TTL používá značně redukované napětí, navíc s opačnou polaritou. Logická jednička je obvykle vyjádřena napětím +5 V nebo +3,3 V; logická nula napětím 0 V. Obousměrnou úpravu těchto napětí realizuje např. čip *MAX232IN*, který bude rovněž velmi důležitým článkem v modulačním obvodu. Detailní podoba je uvedena na obrázku 3.5.



Obrázek 3.5: Mikročip *MAX232IN* regulující napětí linky RS-232

Použití tohoto čipu s sebou ale nese uživatelsky nepříjemnou záležitost – nutnost dodatečného napájení. Čip *MAX232IN* totiž ke své funkci potřebuje zdroj napětí +5 V a to nemusí být dostupné z žádného ze dvou propojených zařízení. Na druhou stranu se tím ale zjednoduší implementace obvodu, protože tento zdroj může být zároveň použit i pro napájení mikrokontroléru. Bez *MAX232IN* by dodatečné napájení nebylo nutné, protože mobilní zařízení kompatibilní se systémem *Android* jsou povinny poskytovat napětí na audio konektoru pro provoz externího mikrofonu.

Toto napětí nazývané *„bias voltage“* je vyžadováno kondenzátorovými mikrofony, které svou funkcí vyvolávají změny tohoto napětí a ty se zpětně zaznamenávají uvnitř přístroje. Specifikace *OS Android* uvádí, že toto napětí musí být v rozmezí 1,8 V–2,9 V [5]. To je zcela dostačující pro napájení mikrokontroléru *ATtiny4313*. V tomto zapojení by ale byl sériový přenos dat možný pouze na úrovni TTL s dříve popsanými hodnotami napětí. Tento typ komunikace nesplňuje požadavky standardu RS-232 a tím pádem nevyhovuje

ani požadavkům uvedeným v zadání práce. Zapojení čipu *MAX232IN* je tedy nevyhnutelné.

Alternativou k externímu zdroji napětí může být konektor DE-9, jehož volné piny bývají mnohdy používány pro různé účely. Některá zařízení přivádí na nevyužité piny napětí, díky kterému je možné napájet případnou podpůrnou elektroniku. Jelikož tato vlastnost konektoru není standardní, nelze předpokládat její přítomnost v každém zařízení a proto je externí napájení jistější volbou.

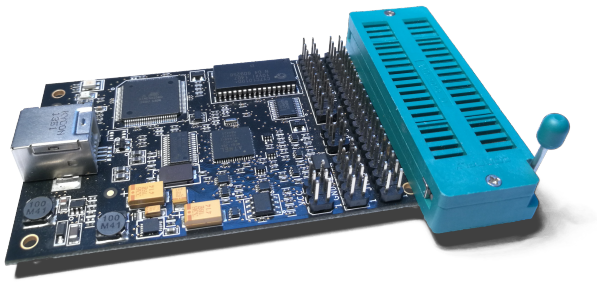
3.6 Analýza softwaru

Úkolem práce se v tuto chvíli stává implementace nejen části mobilní aplikace, ale i programu pro mikrokontrolér.

Knihovna pro mobilní aplikace pro *OS Android* bude implementována v nativním jazyce Java s použitím *Android API* pro přístup k systémovým zdrojům jako mikrofon a oprávnění aplikací. Vývojové prostředí poskytne desktopová aplikace *Android Studio*, která je – stejně jako platforma *Android* – volně ke stažení pod licencí *Apache, verze 2.0*. Dostupné jsou distribuce pro 32- a 64-bitové verze operačních systémů *Windows*, *Mac* a *Linux*.

Program pro mikrokontrolér může být implementován buď v jazyce C nebo assembleru pro mikročipy *AVR*. Z důvodu vyšší efektivity práce zvolím jazyk C. Vývojové prostředí poskytne desktopová aplikace *Atmel Studio*, která je rovněž volná ke stažení pod licencí organizace *Atmel*. K dispozici je pouze distribuce pro *Windows 7* a novější.

Tato aplikace zároveň slouží jako uživatelsky snadný nástroj pro programování čipů skrze programovačku. Programovačka je integrovaný obvod, který má sadu nástrojů pro programování, testování a mazání mikročipů. V tomto projektu bude použit model *AVR Dragon*, který podporuje téměř všechny mikrokontroléry řady *AVR* [6]. Přípravek uvedený na obrázku 3.6 je navíc oproti základní výbavě osazen ZIF paticí pro snadné připojování PDIP mikročipů.



Obrázek 3.6: Programovačka čipů *AVR Dragon*

Kapitola 4

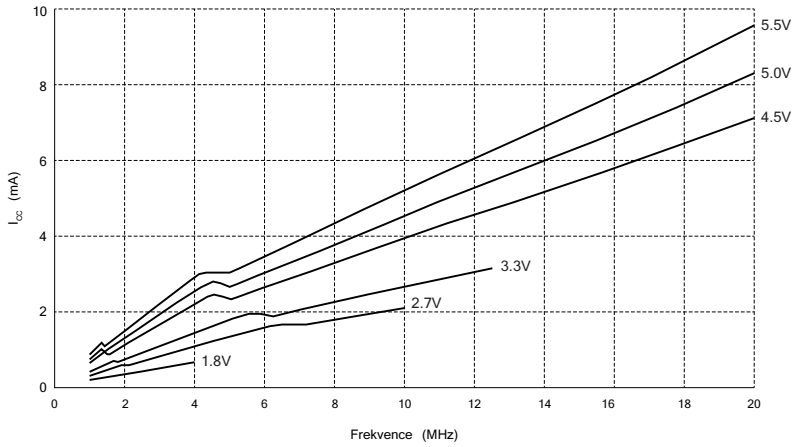
Implementace

4.1 Implementace hardwaru

Proces sestavování obvodu jakožto prostředního článku kabelu byl ve výsledku materiálně velmi náročný. V následujícím seznamu jsou uvedeny všechny hlavní komponenty nezbytné pro vývoj prototypu.

- mikrokontrolér *ATtiny4313*
- RS-232 driver *MAX232IN*
- programovačka *AVR Dragon* včetně USB kabelu
- prototypová deska
- kondenzátory 1 µF
- rezistory 1 000 Ω, 2 700 Ω, 270 kΩ
- propojovací kabely
- TRRS Jack 3,5mm konektor
- DE-9 konektor
- transformátorová páječka včetně cínové pájky

Prvním krokem v implementaci hardwarové části projektu bylo správné nastavení mikrokontroléru. Podle dokumentace podporuje *ATtiny4313* taktovací frekvenci až 20 MHz. Této frekvence lze však dosáhnout pouze s externím oscilátorem. Kvůli nízké přenosové ryhlosti dat ale takto vysoké hodnoty nebyly potřebné a zcela dostačující byl interní krystalový oscilátor s frekvencí 8 MHz. Přesto i tato frekvence byla vyšší než nezbytně nutná a navíc dle grafu na obrázku 4.1 s rostoucí hodinovou frekvencí roste spotřeba proudu, tudíž byla zvolena nižší.



Obrázek 4.1: Graf závislosti proudu na hodinové frekvenci

Tabulka 4.1: Porovnání vzorkovací frekvence f_s a frekvence signálu f_c

$f_s = 44{,}1\,\mathrm{kHz}$

f_c	f_s/f_c	f_{CLK}/f_c	Přesnost
22 050 Hz	2	90,703	40,590 %
14 700 Hz	3	136,054	89,114 %
11 025 Hz	4	181,406	18,820 %
8 820 Hz	5	226,757	51,474 %
7 350 Hz	6	272,109	78,228 %
6 300 Hz	7	317,460	7,934 %

Pro stanovení minimální možné taktovací frekvence mikrokontroléru bylo potřeba stanovit frekvenci nosného signálu pro výstup do mobilního zařízení a přenosovou rychlost UART.

Sériová data se dají protokolem RS-232 posílat různou rychlostí. Standard definuje několik validních hodnot, mezi kterými je možno vybírat. Pro tento projekt bylo za nejvyšší podporovanou přenosovou rychlost zvoleno velmi častých 9 600 Bd. Nejnižší taktovací frekvence, která tuto rychlost přenosu dokáže bezchybně zpracovat, je 2 MHz. Pro tuto hodinovou frekvenci čipu tedy byla hledána nejvhodnější frekvenci výstupního signálu.

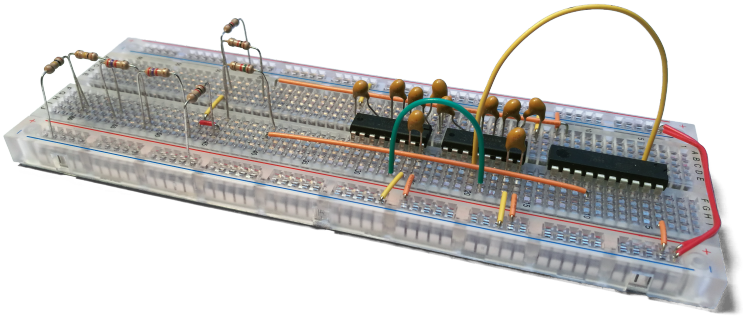
Signál na výstupní straně obvodu je shora omezen maximální frekvencí 22,05 kHz. Aby byl signál přijímán a digitalizován v mobilním zařízení co nejkvalitněji, měla by být vzorkovací frekvence celočíselným násobkem nosné frekvence. Zároveň ale musí být taktovací frekvence mikrokontroléru rovněž celočíselným násobkem, aby bylo vůbec možné takový signál generovat. V tabulce 4.1 jsou uvedeny některé možné frekvence a jejich relativní přesnost.

Je vidět, že nejvyšší přesnosti lze dosáhnout s frevkvencí 14,7 kHz. V praxi se ale ukázalo, že byla tato hodnota příliš velká. Při čtení signálu docházelo

ke zkreslení kvůli malému poměru vzorkovací a přijímané frekvence a začal se projevovat vliv vzájemného fázového posunu. Proto byla zvolena v pořadí další nejlepší nosná frekvence, 7 350 Hz.

4.1.1 Sestavení prototypu

Prototyp modemového kabelu jsem skládal na prototypové desce s nepájivým kontaktním polem. Práci s ní jsem si v tomto projektu vyzkoušel poprvé a velmi kladně hodnotím jednoduchost a efektivitu práce při skládání elektrických obvodů s použitím této pomůcky. Na obrázku 4.2 lze vidět desku se zapojenou částí prototypu rozšířeným o další, redundantní obvody.



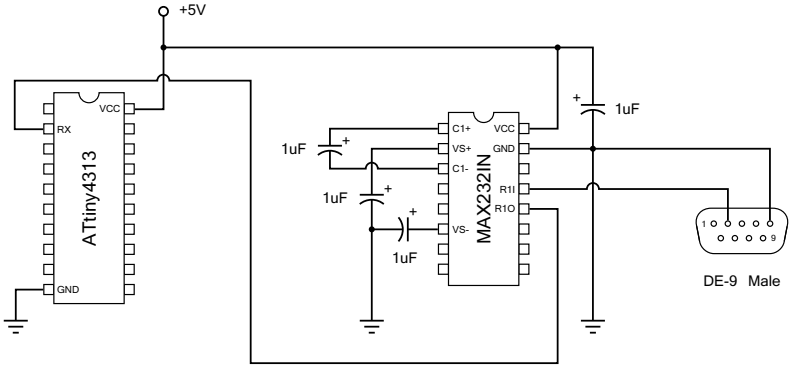
Obrázek 4.2: Prototypová deska s nepájivým kontaktním polem

Programování mikrokontroléru probíhalo přes programovačku *AVR Dragon* skrze rozhraní ISP protokolem SPI. Proces programování byl plně automatický díky skvělé integraci programovačky do vývojového prostředí *Atmel Studio*. Funkčnost všech komponent byla na začátku sestavování otestována napsáním jednoduchého programu a rozsvícením LED diody. Po úspěšném vstupním testu byla prototypová deska připravena pro sestavení všech částí tohoto projektu.

4.1.2 Realizace přijímání sériových dat

Na prvním rozhraní mikrokontroléru z pohledu směru přenosu dat bylo úkolem realizovat přijímání sériových dat ve formátu RS-232. Jak bylo již dříve popsáno, existují dva typy signálů tohoto protokolu. Jeden „plnohodnotný“ a druhý TTL, který má jiné hodnoty napětí a opačnou polaritu. Řešením pro jejich vzájemnou konverzi je zapojení čipu *MAX232IN*.

Mikročip *MAX232IN* funguje na principu nábojové pumpy, díky které dokáže invertovat a zdvojovat napětí [7]. Existují klasické varianty pouzder PDIP, vhodné pro sestavení prototypu, a o něco menší SOP pro použití v konečném produktu. Jeden čip sice dokáže konvertovat celkem dva kanály RS-



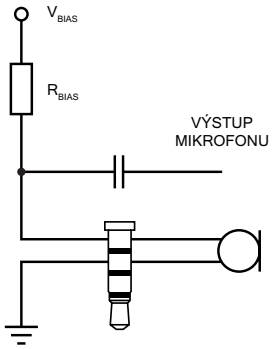
Obrázek 4.3: Schéma zapojení obvodů pro přijímání dat

232 nezávisle na sobě, ale v tomto projektu byl potřeba jen jeden a to dokonce pouze v jednom směru.

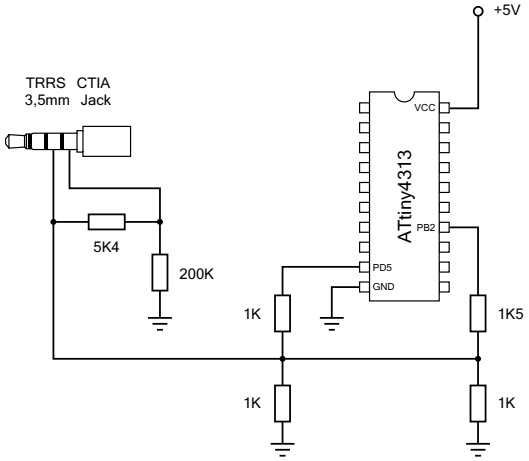
Schéma zapojení a způsob používání jsou detailně popsány v oficiální dokumentaci mikročipu. [8]. Nábojové pumpy obsažené uvnitř pouzdra vyžadují kondenzátory pro zdvojení napětí a ty je nutné připojit dodatečně vedle čipu. Dále je pár kondenzátorů vyžadováno k odfiltrování šumu ze vstupních signálů a napájecího napětí. Schéma zapojení obvodů pro přijímání dat je vyobrazeno na obrázku 4.3.

4.1.3 Realizace vysílání signálu

Dalším krokem implementace byla realizace propojení mikrokontroléru s mobilním zařízením skrz audio konektor. Pro tyto účely jsem musel páječkou spojit dva vodiče s TRRS Jack konektorem, konkrétně s kontakty MIC a GND. Tyto dva vodiče jsou již podle tradičního způsobu použití připraveny pro samotné propojení mikrofonem. Interní filtr signálu je znázorněn na obrázku 4.4. Zbylé dva kontakty na konektoru zůstávají nevyužity, protože obousměrná komunikace zatím nebude implementována.



Obrázek 4.4: Schéma vnitřních obvodů audio konektoru (zjednodušeno)

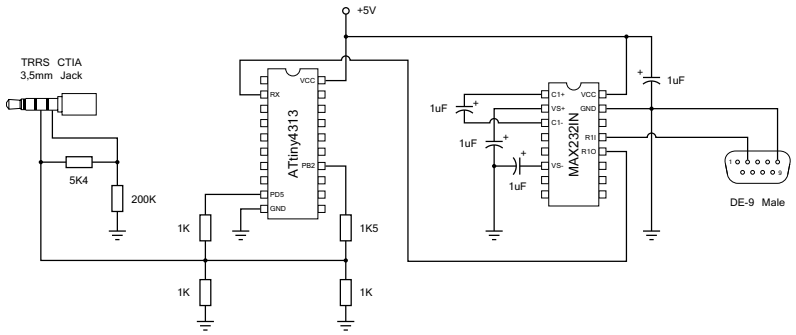


Obrázek 4.5: Schéma zapojení obvodů pro odesílání dat

Cílem tedy bylo napodobit chování mikrofonu včetně všech jeho elektrofyzikálních vlastností. Zařízení se systémem *Android* totiž na pin MIC přivádí napětí pouze pokud je rozpoznána patřičná impedance. Průměrná impedance kondenzátorových mikrofonů se pohybuje mezi 5–10 kΩ dle kvality výroby. Tuto vlastnost lze nasimulovat vložením rezistoru mezi MIC a GND, v tomto případě konkrétně 5,4 kΩ.

Dále bylo potřeba přivést generovaný signál o střídavém napětí na vodič GND. Způsob generování bude dále popsán v sekci 4.2. Výstupem mikrokontroléru je obdélníkový signál se střídou 50 % a proměnlivým napětím. Data na výstupu jsou modulována amplitudovou modulací se čtyřmi možnými úrovněmi. Toho je dosaženo vzájemnou superpozicí dvou signálů se stejnou fází a různou amplitudou. Schéma zapojení obvodů pro odesílání dat je zobrazeno na obrázku 4.5.

Realizací rozhraní pro mobilní zařízení byla implementace hardwaru úspěšně dokončena. Kompletní zapojení obvodů popisuje schéma 4.6.



Obrázek 4.6: Kompletní schéma zapojení všech obvodů

4.2 Implementace softwaru – mikrokontrolér

Program pro mikrokontrolér *ATtiny4313* byl navržen a implementován v jazyce C. Jeho klíčové funkce jsou přijímání sériových dat přes rozhraní UART a generování signálu odesílaného do mobilního zařízení.

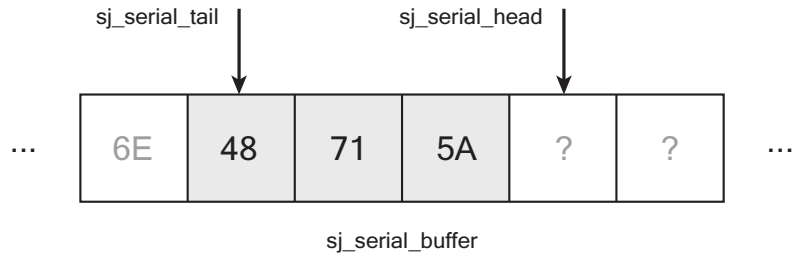
Po spuštění programu se musí nejdříve nastavit správná hodinová frekvence pomocí předděličky a registru `CLKPR` (Clock Prescaler Register). Základních 8 MHz, které produkuje interní oscilátor, je děleno čtyřmi na požadované 2 MHz. Obě části programu – přijímání dat a generování signálu – používají hardwarové přerušení a proto je potřeba jej globálně povolit v regitru `SREG` (Status Register). Hlavní cyklus programu je pak velmi přímočarý. Co se přijme na rozhraní UART, by mělo čip opustit „druhou stranou“ v podobě PWM signálu.

4.2.1 Přijímání sériových dat

Před zahájením jakéhokoliv čtení dat se musí nejprve nastavit správná přenosová rychlost, kterou jsem stanovil na 9 600 Bd. Nastavení rychlosti přenosu se na čipech *AVR* realizuje zápisem do 12-bitového registru `UBRR` (UART Baud Rate Register). Do tohoto registru se zapisuje hodnota vypočítaná vztahem

$$\texttt{UBRR} = \left\lfloor \frac{f_{CPU}}{16{\cdot}baudrate} \right\rfloor - 1 = \left\lfloor \frac{2\,\mathrm{MHz}}{16{\cdot}9\,600\,\mathrm{Bd}} \right\rfloor - 1 = 12\,.$$

Dále se povolí přerušení při přečtení dat a spustí přijímání. Protože je přijímání dát řádově rychlejší než následné odesílání, je nutné přečtená data ukládat do fronty pro pozdější zpracování. Tato fronta byla implementována jako kruhový buffer se dvěma ukazateli nad polem bajtů (obr. 4.7).



Obrázek 4.7: Znázornění kruhového bufferu se dvěma ukazateli

Kapacita fronty je z obou směrů omezena. Nejmenší možná kapacita, se kterou je běh programu možný bez chyb, je určena poměrem rychlosti přijímání a odesílání dat. Nemělo by se stát, že se přijímač zahltí neustálým tokem dat, která se nestihnou z fronty vybírat a odesílat dál, což by mělo za následek ztrátu informací. Tato situace by se měla vyskytnout jen velmi zřídka, pokud možno vůbec.

Shora je kapacita omezena dostupnou pamětí pro alokaci pole. *ATtiny4313* má pouze 256 bajtů pro datovou paměť a v aktuální implementaci programu je tudíž maximální možná kapacita zhruba 130 bajtů. V čipu je také obsažena Flash paměť pro instrukce s celkovou kapacitou 4 KiB, kterou je možné používat i pro ukládání dat. Časy přístupů do této paměti jsou ale vyšší a celková implementace je náročnější. Protože je pro tento projekt rychlost operací velmi důležitá, buffer pro UART přijímač byl alokován v SRAM paměti s maximální možnou kapacitou. Zvýšení této kapacity může být jedním z témat dalších prací na projektu.

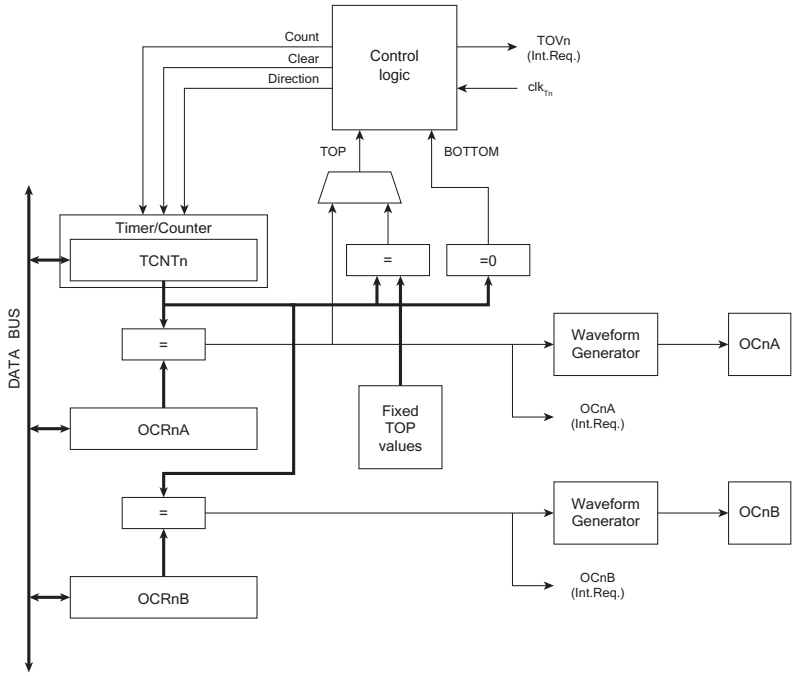
Samotné čtení přijímaných dat má na starosti obvod, který pracuje automaticky. Veškerá interakce s tímto obvodem je zredukována na přečtení hodnoty z bufferu `UDR` (UART Data Register) během přerušení od přijímače. `UDR` je speciální registr, který se používá pro přijímání i odesílání dat přes UART. Zápisem do `UDR` se data uloží do fronty a automaticky proběhne nastavení flagu `UDRE` (UART Data Register Empty) na nulu pro signalizaci, že jsou data připravena k odesílání. Čtením z `UDR` se získávají přijatá data, pokud je nastaven flag `RXC` (Receive Complete). Ve směru přijímače je `UDR` implementován jako dvoustupňový hardwarový buffer a přidává tak další úroveň zabezpečení proti příliš dlouhým intervalům čtení. Data získaná z bufferu `UDR` se v obsluze přerušení ukládají ještě do předem zmiňovaného softwarového bufferu pro pozdější použití.

Funkce `sj_serial_receive()` implementuje tzv. „*busy waiting*“, který čeká do té doby, než budou do softwarového bufferu `sj_serial_buffer` zapsána data. V této busy waiting smyčce bude pravděpodobně hlavní program trávit většinu času svého běhu. Jakmile jsou data dostupná (ukazatele na začátek a konec bufferu se liší), funkce je vrátí volajícímu.

4.2.2 Generování PWM signálu

Podobně jako před přijímáním sériových dat se musí i mechanismus odesílání správně nastavit před zahájením komunikace. Výsledný signál přejímá strukturu dat od protokolu RS-232 a zavádí jen odlišný způsob vyjádření logických hodnot. PWM signál byl předem definován frekvencí 7 350 Hz, která zůstává po celou dobu konstantní. Frekvence jednotlivých bitů byla experimentálně zvolena 10× nižší, tedy 735 Hz. V této formě signálu se však neposílají jednotlivé bity zvlášť, ale po dvojicích díky 4-úrovňové amplitudové modulaci. Start a stop bity nicméně zůstávají nezměněny a proto není výsledné zrychlení 100 %, ale pouze 67 %.

PWM signál se v mikrokontroléru *ATtiny4313* generuje pomocí časovače s funkcí Output Compare, viz obrázek 4.8. V zásadě jde o inkrementální čítač, který má softwarově nastavitelnou periodu a v určitých momentech dokáže vyvolat přerušení. Princip generování signálu PWM spočívá v nastavování logické hodnoty výstupu v závislosti na relativní pozici hodnoty čítače (`TCNT`) a hodnoty porovnávacího registru (`OCR`). `TCNT` je s `OCR` neustále porovnávána



Obrázek 4.8: Blokové schéma čítače na čipu *ATtiny4313*

a v případě shody je vyvoláno přerušení, kde se výstupní hodnota nastaví na logickou jedničku. Další přerušení je vyvoláno při přetečení čítače a v tento moment je výstupní hodnota nastavena zpět na logickou nulu.

Na čipu existují dva časovače/čítače – *Timer0* a *Timer1*. Timer0 je 8-bitový časovač se dvěma kanály, což znamená, že pro jednu hodnotu `TCNT0` existují dva porovnávací registry `OCR0A` a `OCR0B`. Současným použitím se dají genererovat dva signály se stejnou periodou a fází a různou střídou. Klasické použití časovače pro generování signálu v operačním módu *Fast PWM* ale není možné, protože perioda signálu má být 272 hodinových taktů. Timer0 je 8-bitový, tudíž jeho výstupem může být signál s maximální periodou 256 taktů. Oba časovače v *ATtiny4313* sice mají možnost použít předděličku na zdroji hodinového signálu k prodloužení periody, ale kvůli technickým omezením není na časovači Timer0 možné libovolně nastavit periodu a porovnávací hodnotu zároveň. S jedním stupněm volnosti lze libovolně naprogramovat jen jednu z těchto hodnot, přičemž druhá je pak pevně daná a neměnná.

Řešením tohoto problému je použití operačního módu *CTC (Clear Timer on Compare)*, který spojením s vhodnou funkcí pro generování výstupu dokáže požadavky splnit. Podstata módu CTC spočívá v automatickém vynulování časovače při shodě `TCNT` a `OCR`. Modul pro generování signálu na základě hodnot časovače lze následně nastavit tak, aby se logická hodnota jeho výstupu s každým vynulováním časovače převrátila. Tím je na výstupu generován signál se střídou 50 % a dvakrát nižší frekvencí, než je frekvence

Tabulka 4.2: Přiřazení logických hodnot amplitudám signálu

Odesílaná data	Logická hodnota	Amplituda signálu
`00`	0	100 %
`01`	1	75 %
`10`	2	50 %
`11`	3	0 %

vynulování časovače. Důsledkem toho je potřeba zrychlit časovač zkrácením jeho periody na polovinu. Perioda časovače Timer0 se v módu CTC ovládá registrem `OCR0A`, který obsahuje hodnotu, po jejíž shodě s `TCNT0` se časovač v následujícím taktu vynuluje. Pro dosažení výstupu s frekvencí 7 350 Hz je výpočet `OCR0A` následující:

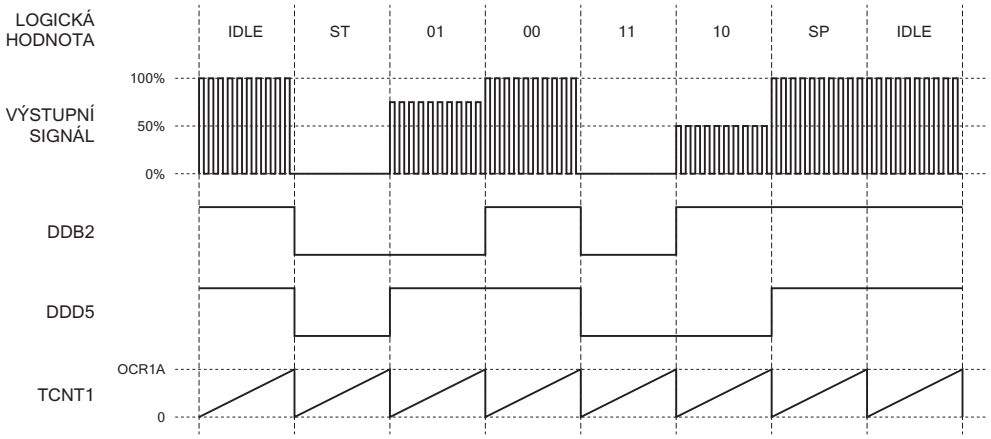
$$\texttt{OCR0A} = \left\lfloor \frac{f_{osc}}{2 \cdot f_{PWM}} \right\rfloor - 1 = \left\lfloor \frac{2\,\text{MHz}}{2 \cdot 7\,350\,\text{Hz}} \right\rfloor - 1 = 135$$

Na stejnou hodnotu je nastaven i registr `OCR0B`, který sice nemá žádný vliv na běh časovače, ale je důležitý pro generování kanálu B, který je mimo čip skládán s kanálem A pro 4 úrovně amplitudové modulace. S tímto nastavením je mikrokontrolér schopen generovat nosný signál pro odesílaná data.

Dalším krokem je použití časovače Timer1 pro pravidelnou modulaci nosného signálu odesílanými daty. Frekvence změn v tomto signálu byla experimentálně zvolena 10× nižší, tedy 735 Hz. Podobně jako u předchozího časovače, je i zde používán operační mód CTC, ale bez následného generování signálu PWM. Timer1 je zde používán vysloveně k pravidelnému časování událostí, nicméně automatické vynulování je výhodou. Perioda vyjádřená registrem `OCR1A` je tedy nastavena na 2 719.

Každých 2 720 taktů je vyvoláno přerušení `TIMER1_COMPA`, ve kterém jsou ovládány výstupy časovače Timer0. Při odesílání jednoho bajtu existuje 6 fází modulace: 1 start bit, 4 dvojice bitů a 1 stop bit. Start bit je odesílán jako signál s nulovou amplitudou, stop bit s maximální. Zbylé čtyři fáze nesoucí data přijaté z UARTu se skládají ze signálů s amplitudou definovanou podle tabulky 4.2. Dvojice bitů jsou předpočítány a uloženy zvlášť ještě před zahájením vysílání, kvůli úspoře času. Data jsou odesílána ve formátu Big-endian.

Čtyř úrovní amplitudové modulace je dosaženo selektivním otevíráním a zavíráním kanálů A a B výstupu PWM z časovače Timer0. Odpojování a připojování výstupních portů mikrokontroléru je uskutečněno zapisováním do `DDR` (Data Direction Register) příslušného portu. Kanál A je vyveden na port B2, kanál B na port D5. Nastavením příslušného bitu v `DDR` na nulu přepneme třístavový buffer do stavu vysoké impedance a výstup je tak odpojen. Nastavením na jedničku se opět připojí a logická hodnota výstupu je definována příslušným bitem v registru `PORT`. Ten je ovládán automaticky v rámci každého vynulování časovače Timer0. Pravidelným střídáním hodnot 0 a 1



Obrázek 4.9: Časový diagram odesílání znaku „N“

je dosaženo signálu se střídou 50 %. V diagramu 4.9 jsou zobrazeny průběhy hodnot, které se podílejí na formování výsledného signálu. Písmeno „N“ bylo vybráno jako ukázková hodnota, protože jeho binární reprezentace 01001110_2 obsahuje všechny kombinace dvojic bitů.

Uživatelům nebude software na mikrokontroléru dostupný a proto není důležité implementovat jednotná rozhraní funkcí nebo se zabývat příliš podrobnou dokumentací kódu. Pro účely bakalářské práce je ovšem kód řádně okomentován.

4.3 Implementace softwaru – mobilní zařízení

Cílem tohoto projektu je rovněž poskytnout řešení pro přijímání dat na straně mobilního zařízení. Implementace je požadována pouze pro operační systém *Android* a proto byl zvolen programovací jazyk Java.

Protože je na mobilních zařízeních audio konektor používán primárně pro reprodukci a záznam zvuku, dekódování signálu je prováděno stejným způsobem, jako by šlo o záznam zvuku. Hlavní třída `SerialJack` poskytuje několik metod pro ovládání vstupního rozhraní:

- `boolean start()` zahájí záznam dat z audio konektoru a přečtená data začne ukládat do interního bufferu.
- `byte receive()` vrátí jeden bajt přečtených dat ve FIFO pořadí.
- `void stop()` ukončí záznam, přičemž data zůstávají stále přístupná.
- `void flush()` vymaže všechna data z bufferu.

V konstruktoru je možno nepovinně určit specifický zdroj dat pomocí třídy `MediaRecorder.AudioSource`, pokud jich fyzické zařízení poskytuje více. Protože se ve skutečnosti jedná o záznam zvuku, je možné přijímání dat spustit

i s odpojeným kabelem a jako zdroj dat se v tu chvíli volí interní mikrofon. Tento způsob „bezdrátového“ přenosu je ale pouze teoretický a pro alespoň minimální šanci správného chodu by musel probíhat bez rušení okolními zvuky a ozvěnou. Navíc je nosný signál ve spektru slyšitelných frekvencí a jeho vysoký tón by byl po dlouhé době vysílání nepříjemný.

Pro interní použití a samotný záznam dat je implementována třída `SerialJackListener`, která obsahuje veškerou logiku spojenou s navázáním komunikace, přijímáním dat a jejich dekódováním.

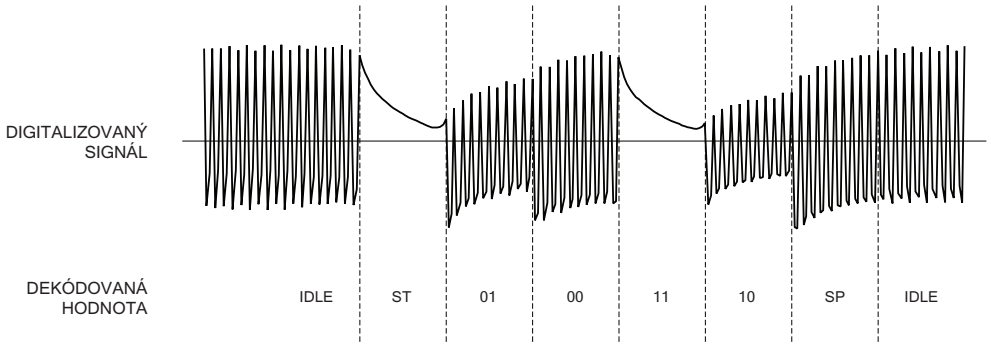
Ještě než je zahájen záznam dat, musí uživatel udělit aplikaci práva pro záznam zvuku, jinak nelze komunikaci s připojeným zařízením navázat. Tuto záležitost má na starost výchozí dialogové okno *OS Android*. Čtení dat probíhá v samostatném vlákně, aby nebyl narušen plynulý chod aplikace. Data jsou čtena se vzorkovací frekvencí 44,1 kHz ve formátu 16-bit PCM [9]. Surová data jsou po blocích dočasně ukládána do bufferu, odkud jsou následně vybírána pro analýzu.

Prvním krokem k úspěšné komunikaci je určení maximální amplitudy přijímaného signálu. Tímto je na spojení kladen požadavek, aby po dobu 250 ms od zahájení záznamu neprobíhal *žádný* přenos dat a linka se nacházela v klidovém stavu. Tento časový úsek by měl být dostatečný pro kvalitní kalibraci dekódovacího algoritmu. Amplituda signálu je při kalibraci i následném přijímání dat počítána pomocí lokálního maxima a minima přečtených hodnot v objemu 10-ti vzorků.

Po úvodní kalibraci je systém připraven přijímat data v dříve popsaném formátu. Z audio konektoru jsou nepřetržitě zaznamenávána data, ze kterých je periodicky počítána aktuální amplituda signálu. V případě zjištění změny (detekce start bitu) se mechanismus přepne do stavu průběžného vzorkování amplitudy a pravidelného odečítání hodnoty. Každý úsek přeneseného „dvojbitu“ je pokryt přesně 60-ti vzorky PCM a po tuto dobu se počítá průměrná amplituda signálu. Po zpracování tohoto objemu dat se vypočítaná průměrná amplituda dekóduje na logickou hodnotu z tabulky 4.2. Logika dekódující úrovně amplitudy na logické hodnoty bohužel není moc pokročilá a její zlepšení by určitě mělo být jedním z cílů dalších prací na tomto projektu.

Mezivýsledky jsou akumulovány a zrekonstruovaný bajt je po přijetí všech čtyř datových částí uložen do fronty pro uživatelské zpracování. Fronta je implementována jako blokující pomocí třídy `ArrayBlockingQueue`, což znamená, že pokus o vybrání bajtu z prázdné fronty volající vlákno pozastaví, dokud není vložen další bajt.

Na obrázku 4.10 je znázorněn časový průběh digitalizovaného signálu při přijímání znaku „N“. Graf byl překreslen ze snímku obrazovky mobilního zařízení. Z prostorových důvodů byla snížena vzorkovací frekvence na 22,05 kHz, aby bylo možné zachytit všechny fáze signálu jedním snímkem.



Obrázek 4.10: Časový diagram přijímání znaku „N“

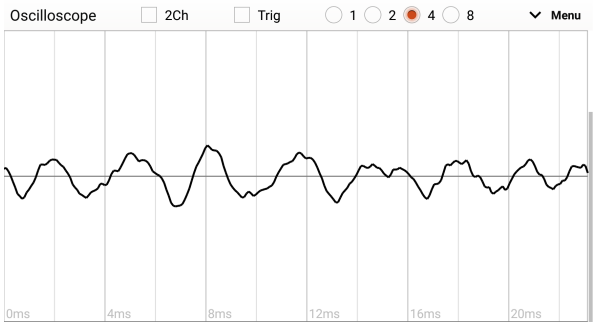
Kapitola 5

Testování

Testování celého systému se dělí na hardwarovou a softwarovou část. V závěru práce bylo potřeba otestovat nejen bezchybnou implementaci programu v mikrokontroléru a algoritmy dekódování signálu v mobilním zařízení, ale i správné zapojení obvodů, hodnoty napětí, odporů atd.

Kontrolu signálu přijímaného v mobilním zařízení Huawei Honor 8 jsem prováděl pomocí aplikace *Oscilloscope*[1], která je velmi kvalitním a uživatelsky jednoduchým nástrojem pro analýzu dat přečtených z audio konektoru. Na obrázku 5.1 je uveden snímek obrazovky při typickém použití. Vizualizace touto aplikací byl jediný možný způsob, jak analyzovat data, která jsou skutečně zaznamenávána softwarem uvnitř. Příčinou toho je filtrování signálu těsně před digitalizací a tím pádem k signálu vystupujícímu z filtru zabudovaného uvnitř mobilního zařízení není žádný jiný přístup. Klasickým osciloskopem je možné monitorovat pouze signál vystupující z mikrokontroléru, který však nemusí být a zpravidla není totožný s jeho digitalizovaným obrazem.

[1] Dostupné z `https://play.google.com/store/apps/details?id=com.xyz.scope`



Obrázek 5.1: Snímek obrazovky v aplikaci *Oscilloscope* (barevně upraveno)

Tabulka 5.1: Měření napětí mezi různými vodiči a zemí

Měřený signál/vodič	Naměřená hodnota napětí (V)
VCC	4,90 ± 0,01
PB2	4,83 ± 0,01
PD5	4,81 ± 0,01
Amplituda PWM signálu (`00`)	2,19 ± 0,01
Amplituda PWM signálu (`01`)	1,605 ± 0,001
Amplituda PWM signálu (`10`)	1,207 ± 0,001
Amplituda PWM signálu (`11`)	0,000 1 ± 0,000 1

5.1 Testování hardwaru

Důležitým nástrojem pro testování hardwarové části systému byl digitální multimetr, který posloužil k měření výstupních napětí mikrokontroléru. V tabulce 5.1 jsou uvedeny některé hodnoty napětí měřené proti zemi. Naměřené hodnoty jsou z velké části shodné s očekávanými. Různé absolutní chyby měření jsou dány různou přesností měření v určitých rozmezích napětí [10].

5.1.1 Přijímání dat

Obvody pro přijímání dat se skládají pouze z elektrických vodičů a mikročipu *MAX232IN*. Při dodržování doporučení stanovených v dokumentaci [8] zde prakticky není žádný prostor pro variabilitu a celý postup zapojení je tím pádem velmi přímočarý. Jediné, co pak musíme otestovat je pouze správné propojení pinů a velikost napájecího napětí.

Nominální hodnota napájecího napětí (VCC) +5 V se od naměřené nepatrně liší, což může být způsobeno několika příčinami. Je možné, že USB konektor v počítači, který je zdrojem tohoto napětí, není přesně nastaven nebo kalibrován. Také je možné, že programovačka *AVR Dragon*, kterou elektrický proud při testování prochází, způsobuje pokles napětí kvůli odběru proudu pro vlastní obvody. Určení konkrétní příčiny ale není podstatné, protože na přesné hodnotě nezáleží. Jedinou podmínkou je, že čip *MAX232IN* má k dispozici napětí 4,5–5,5 V, což bylo splněno. Tímto bylo úspěšně dokončeno testování hardwarové části přijímání dat.

5.1.2 Odesílání dat

Testování druhé hardwarové části bylo kvůli velkému počtu součástek nepatrně náročnější, nicméně o to potřebnější a důkladnější. Nejdříve byla zkontrolována správná funkce výstupních portů mikrokontroléru. Na portech B2 a D5 lze v tabulce 5.1 vidět pokles napětí proti VCC, který je způsoben vnitřní elektronikou čipu. Této skutečnosti nelze zabránit a je považována za přirozenou bez nežádoucích účinků.

Tabulka 5.2: Měření napětí mezi kontakty MIC a GND

Výstupní logická hodnota	Naměřená hodnota napětí (V)
0	$0{,}543 \pm 0{,}001$
1	$0{,}522 \pm 0{,}001$
2	$0{,}516 \pm 0{,}001$
3	$0{,}505 \pm 0{,}001$

Naměřené hodnoty napětí PWM signálu odpovídají očekávaným hodnotám vzhledem k jasně definovaným děličkám napětí, viz obrázek 4.5. Naměřená napětí také svou relativní velikostí k VCC odpovídají hodnotám definovaným v tabulce 4.2.

Následně bylo potřeba otestovat, zda je bezpečné odesílat signál s těmito hodnotami do mobilního zařízení. V tabulce 5.2 jsou uvedeny výsledky měření napětí mezi kontakty MIC a GND audio konektoru při konstantním výstupu jednotlivých logických hodnot PWM signálu.

Při odpojení obou kanálů PWM (logická hodnota 3) je mezi MIC a GND stálé napětí 505 mV, které je použito jako referenční pro detekci změn uvnitř konektoru. Maximální změna oproti referenčnímu byla naměřena o velikosti 38 mV s logickou hodnotou 0. Tento rozdíl napětí je bezpečný, protože se při digitalizaci projeví jen jako necelá polovina celkové dostupné škály. Tímto byla úspěšně otestována hardwarová část odesílání dat.

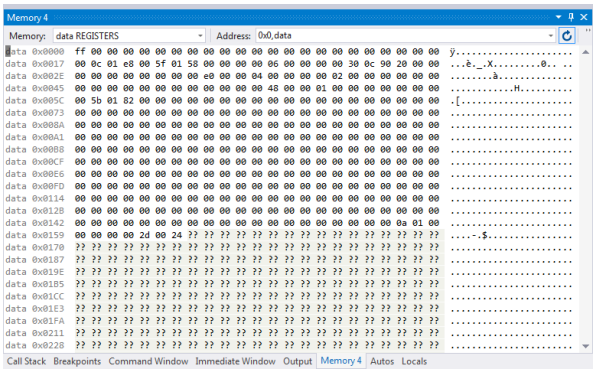
5.2 Testování softwaru

Při testování sofwaru bylo ve velkém rozsahu použito mnoho nástrojů poskytnutých vývojářským prostředím. Zejména důležitou funkcí bylo pozastavení běhu programu a sledování hodnot proměnných.

5.2.1 Mikrokontrolér

Správný chod a ladění programu mikrokontroléru lze dělat přímo za běhu přes rozhraní *debugWIRE* programovačky *AVR Dragon*. Tento způsob ladění umožňuje převzít plnou kontrolu nad chodem programu použitím breakpointů a krokování. Zároveň je možné v jakémkoli okamžiku zobrazit aktuální obsah paměti na čipu včetně změn způsobených vnějšími událostmi, např. přijetím dat přes UART. Náhled paměti je zobrazen na obrázku 5.2. Samotné ovládání běhu programu je softwarově řešeno ve vývojovém prostředí *Atmel Studio*.

Přijímač byl automaticky otestován vzorkem 5 000 bajtů a všechny byly přijaty mikrokontrolérem bez chyby. Tímto lze úspěšně ukončit testování přijímání dat přes linku RS-232. Výsledkem tohoto testu je tvrzení, že přenos dat má spolehlivost minimálně 99,999 8 %. Zároveň byla otestována a prokázána správná funkce softwarového bufferu.



Obrázek 5.2: Zobrazení obsahu paměti čipu v aplikaci *Atmel Studio*

5.2.2 Mobilní zařízení

Podobně jako mikrokontrolér lze i zařízení se systémem *Android* připojit k PC v módu pro ladění a sledovat změny veškerých proměnných. Pro povolení ladění aplikací je potřeba v systému povolit „Vývojářský mód“ a instalaci aplikací z neznámých zdrojů. Desktopová aplikace *Android Studio* následně poskytuje obvyklou sadu nástrojů pro vývoj, ladění a testování aplikace spouštěné přímo na mobilním zařízení.

Hlavním zaměřením této části testování bylo dekódování modulovaného signálu a správné časování přenosu. Důkladné testování dekódování bylo velkým přínosem pro celkovou kvalitu systému, neboť jsem musel implementaci několikrát pozměnit, abych dosáhl větší spolehlivosti. Výsledná podoba nicméně stále není perfektní, protože se průměrně 1 z 1 500 přijatých bajtů lišil v jedné dvojici bitů od očekávané hodnoty. I přes tyto závěry testů lze prohlásit, že testování skončilo úspěšně a spolehlivost dekódování jednotlivých úseků signálu je přibližně 99,999 83 %.

S ohlédnutím k prvotnímu požadavku projektu – čtení sériových dat z digitální váhy – je tento výsledek naprosto dostačující.

5.3 Shrnutí

Prototyp modemového kabelu byl důkladně otestován s velmi dobrými výsledky. Přestože testy odhalily nepřesnosti v dekódování signálu na straně mobilního zařízení, je kabel považován za funkční. Vzhledem k původnímu účelu použití je četnost výskytu těchto chyb zanedbatelná.

Kapitola 6

Možnosti rozšíření

Všechny požadavky zadání bakalářské práce byly tímto návrhem splněny a finální produkt má i přesto potenciál nabídnout více. V této kapitole jsou diskutována možná rozšíření, kterými by se dala funkcionalita kabelu obohatit.

6.1 Obousměrná komunikace

U každého elektrického kabelu je obousměrná komunikace téměř samozřejmostí, přesto u tohoto chybí. Absence druhého směru toku dat však není v rozporu se zadáním, protože hlavním záměrem použití tohoto kabelu je připojení digitální váhy k mobilnímu zařízení. Váha jakožto měřící nástroj je zpravidla pouze zdrojem dat, nikoliv spotřebičem.

Kabel má nicméně velký potenciál pro univerzální použití a proto by bylo vhodné implementovat i opačný směr toku dat. Pro mobilní zařízení by to znamenalo návrh a implementaci generování analogového signálu, který by byl odesílán skrz audio konektor. Pro mikrokontrolér pak implementaci záznamu, dekódování a odeslání dat přes sériovou linku rozhraním UART.

6.2 Nastavitelná rychlost přenosu

V této implementaci je kabel schopen přijímat sériová data pouze jednou, předprogramovanou rychlostí. Tím je silně omezeno množství kompatibilních zařízení, která se mohou této rychlosti přizpůsobit. Pro širší podporu zařízení by bylo vhodné implementovat možnost volit přenosovou rychlost sériové komunikace z několika předdefinovaných hodnot.

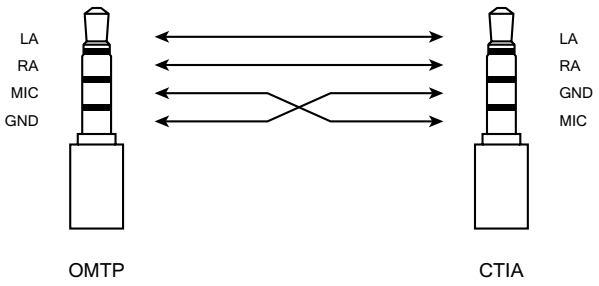
Velmi reálné je použití mechanického přepínače nebo tlačítka přístupného z vnějšku pouzdra obsahujícího mikrokontrolér. Zapojení tohoto prvku by nebylo příliš obtížné a velkou měrou by přispělo k použitelnosti kabelu se zařízeními, jejichž rychlost nelze nastavit.

Jiným způsobem, jak umožnit změnu přenosové rychlosti, je rozšíření softwaru v mobilním zařízení o funkci, která by komunikovala s mikrokontrolérem přes audio konektor a poslala příkaz ke změně rychlosti. To je ale do určité míry podmíněno implementací předchozího rozšíření obousměrné komunikace a bez ní je hardwarové řešení lepší volbou.

6.3 Volitelnost OMTP/CTIA

Přestože byl OMTP dlouhou dobu standardním typem audio konektoru pro výrobce mobilních zařízení se systémem *Android*, novější modely posledních let začínají adoptovat typ CTIA. Rozdíl mezi těmito dvěma je vzájemně obrácená pozice kontaktů MIC a GND (obr. 6.1). Zapojení konektoru jednoho typu do zdířky druhého typu má za následek zkrat obvodu pro mikrofon, který tím pádem přestává fungovat. Přestože je obvod zkratován, konektoru nehrozí žádné riziko poškození, ale záznam zvuků již není možný.

Jedním možným řešením je použití adaptéru, který díky překříženým vodičům dokáže vést signály a napětí na správné kontakty opačného standardu. Z pohledu uživatele to ale znamená nutnost investice do další propojovací techniky. Druhým řešením, přijatelnějším pro uživatele, by mohla být funkcionalita adaptéru zabudovaná přímo do kabelu již od výroby. Vhodným zapojením přepínačů nebo tranzistorů by bylo možné zaměnit význam těchto dvou odlišných kontaktů, čímž by se kabel stal kompatibilní prakticky s každým standardním mobilním zařízením.



Obrázek 6.1: Porovnání audio konektorů typu OMTP a CTIA

Závěr

Na základě této bakalářské práce byl navržen a sestaven modemový kabel pro přenos sériových dat do mobilních zařízení skrz Audio Jack. Touto sestavou byl vytvořen nový způsob, jak protokolem RS-232 komunikovat s mobilním zařízením, na kterém není dostupný digitální USB port.

Z možných řešení byl audio konektor podrobným výběrem zvolen jako nejvhodnější komunikační kanál díky malému počtu technických omezení. Díky jednoduchému fyzickému rozhraní a široké podpoře téměř všemi výrobci je skvělou alternativou digitálního USB portu.

Navržený způsob přenosu dat je unikátní svým transformováním digitálního signálu na analogový. Víceúrovňová amplitudová modulace signálu je prováděna zabudovaným mikrokontrolérem *ATtiny4313*, jehož výstup je možné přijímat v mobilním zařízení jako zvukový záznam. Touto funkcionalitou se kabel výrazně odlišuje od existujících řešení jemu podobných.

Součástí odevzdávaného projektu je i softwarová knihovna pro *OS Android* psaná v jazyce Java, implementující přijímání modulovaných dat. Jejím cílem je poskytnout jednoduché rozhraní k používání audio konektoru jako přijímače sériových dat.

Sestavený prototyp byl automaticky i ručně testován na chyby v přijímání, odesílání a dekódování dat. Testování skončilo úspěšně. Výsledky prokazují, že spolehlivost přenosu dat tímto kabelem při použití přiložených softwarových knihoven je vyšší než 99,999 8 %. Konzultací s vedoucím práce bylo potvrzeno úspěšné splnění cílů bakalářské práce.

Touto prací jsem se naučil používat prototypovou desku k vývoji a testování elektrických obvodů s mikročipem a pronikl jsem do oblasti vývoje softwaru pro mobilní zařízení úspěšným naprogramováním mé první aplikace pro *OS Android*. Nabyté dovednosti plánuji nadále využívat a zdokonalovat.

Literatura

[1] USB Implementers Forum Inc.: *On-The-Go Supplement to the USB 2.0 Specification*. 2001, [cit. 2016-10-12]. Dostupné z: http://www.usb.org/developers/onthego/otg1_0.pdf

[2] ITT Cannon: *D-Subminiature Connectors*. [cit. 2016-10-26]. Dostupné z: http://www.ittcannon.com/Core/medialibrary/ITTCannon/website/Literature/Catalogs-Brochures/D-Sub_Full_Line_Catalog.pdf

[3] Google Inc.: *Android 7.0 Compatibility Definition*. 2016, [cit. 2016-2016-10-27]. Dostupné z: http://static.googleusercontent.com/media/source.android.com/en//compatibility/android-cdd.pdf

[4] Atmel Corporation: *8-bit AVR® Microcontroller with 2/4K Bytes In-System Programmable Flash*. [cit. 2016-10-28]. Dostupné z: http://www.atmel.com/images/doc8246.pdf

[5] Google Inc.: *3.5 mm Headset Jack: Device Specification*. [cit. 2016-10-26]. Dostupné z: https://source.android.com/devices/accessories/headset/jack-headset-spec.html

[6] Atmel Corporation: *The Atmel AVR Dragon Debugger*. [cit. 2016-11-05]. Dostupné z: http://www.atmel.com/Images/Atmel-42723-AVR-Dragon_UserGuide.pdf

[7] Schell, C.: Voltage Doubler Design and Analysis. Technická zpráva, National Semiconductor Corporation, 2001. Dostupné z: http://www.ti.com/lit/an/snaa095/snaa095.pdf

[8] Texas Instruments Inc.: *MAX232x Dual EIA-232 Drivers/Receivers*. 1989, revidováno 2014, [cit. 2016-12-10]. Dostupné z: http://www.ti.com/lit/ds/symlink/max232.pdf

[9] Google Inc.: *Android API Reference*. [cit. 2016-12-12]. Dostupné z: `https://developer.android.com/reference/android/media/AudioRecord.html`

[10] AXIOMET: *Digital Multimeter AX-101B Operation Manual*. [cit. 2016-12-20]. Dostupné z: `http://en.axiomet.eu/product/ax-101b/document/389`

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

CTIA Cellular Telephone Industries Association

FIFO First In – First Out

ISP In-System Programming

LED Light Emitting Diode

OMTP Open Mobile Terminal Platform

OS Operation System

PCM Pulse Code Modulation

PDIP Plastic Dual Inline Package

PWM Pulse Width Modulation

SOP Small Outline Package

SPI Serial Peripheral Interface

TRRS Tip/Ring/Ring/Sleeve

TTL Transistor-Transistor Logic

UART Universal Asynchronous Receiver and Transmitter

USB Universal Serial Bus

VQFN Very thin Quad Flat Non-leaded package

ZIF Zero Insertion Force

Příloha B

Seznam použitých pojmů

Assembler je nízkoúrovňový programovací jazyk symbolicky reprezentující jednotlivé strojové instrukce. Assembler je obecný název a vždy je nutné určit, ke kterému systému se vzthauje.

Big-endian je formát zápisu bajtů, ve kterém je bit nejvyššího řádu uveden jako první.

Breakpoint je místo v kódu programu, po jehož dosažení programovým čítačem při běhu je možné chod pozastavit, či ukončit.

Buffer je vyrovnávací paměť, která slouží jako dočasné meziúložiště dat při asynchronním přenosu. Většinou bývá implementován jako fronta, která zachovává pořadí zapisovaných a čtených dat.

Modulace je proces fyzické nebo logické úpravy signálu pro přizpůsobení komunikačnímu kanálu.

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `app`
  - `serialjack-debug.apk` ..... spustitelná aplikace pro Android
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `avr` ..... program mikrokontroléru
    - `android` ..... softwarová knihovna pro Android
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text`
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS