Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra aplikované fyziky a techniky

Diplomová práce

Analyzátor komunikace na sériové lince s mikrokontrolérem Atmel AVR

Vypracoval: Bc. Jaroslav Martínek

Vedoucí práce: Ing. Michal Šerý, Ph.D.

České Budějovice 2015

Prohlášení

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 28. 6. 2015

Bc. Jaroslav Martínek

Anotace

Tato diplomová práce se zabývá návrhem a zpracováním analyzátoru sériové komunikace s použitím jednočipových mikroprocesorů Atmel AVR. Práce popisuje hardwarové a softwarové vybavení potřebné pro výrobu prototypu na platformě Arduino. Zaměřením se orientuje zejména na použití desky Arduino Mega 2560 a její programování v prostředí Arduino IDE.

Abstract

This thesis describes the design and processing of serial communication analyzer using Atmel AVR microcontrollers. The thesis describes the hardware and software required for the production of a prototype on the platform Arduino. The focus is mainly on the use of the board Arduino Mega 2560 and its programming using Arduino IDE environment.

Poděkování

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Michalu Šerému, Ph.D. za odbornou a vstřícnou pomoc, poskytnutí cenných rad a veškerý čas, který mi věnoval.

Dále bych rád poděkoval rodině a blízkým za podporu během zpracování této práce.

Obsah

Úvod

Svým jednoduchým použitím, nízkou spotřebou elektrické energie a vysokým stupněm integrace jsou 8 bitové mikroprocesory Atmel AVR používány nejen profesionálními odborníky pro řízení průmyslových aplikací, ale i pro amatérské použití při řízení jednoduchých aplikací různých projektů. Důkazem je nabídka několika druhů elektronických stavebnic s mikroprocesory Atmel AVR a odborné literatury obsahující velké množství zajímavých projektů pro aplikaci v praxi.

Cílem této diplomové práce je vytvořit zařízení, které bude monitorovat a analyzovat datovou komunikaci na sériové lince pomocí mikrokontroléru Atmel AVR na platformě Arduino. Zaměřuje se zejména na vývoj prototypu při použití desky Arduino Mega 2560 a programování pomocí softwarového prostředí Arduino IDE.

1 Popis rodiny jednočipů Atmel AVR

1.1 Mikroprocesory Atmel obecně

Mikroprocesory Atmel AVR jsou licencovaným produktem americké společnosti Atmel, která je světovou jedničkou ve vývoji a výrobě polovodičů a integrovaných obvodů. Společnost Atmel byla založena v roce 1984 a její sídlo je ve městě San Jose v Kalifornii. Atmel se orientuje na velký okruh aplikačních segmentů v telekomunikacích, počítačových sítích, průmyslu, zdravotnictví, automobilovém, leteckém a vojenském průmyslu. Dále podniká na trhu bezpečnostních systémů, pro které dodává čipové a RFID karty. [1]

V portfóliu mikroprocesorů nalezneme 8 bitové nízko-příkonové, využívající koncepci harvardské architektury s redukovanou instrukční sadou RISC. Tato architektura fyzicky odděluje paměť programu a dat a jejich spojovací obvody. Jednočipové počítače v sobě integrují aritmeticko-logickou jednotku, paměť SRAM, non-volatilní paměti EEPROM a Flash, vstupně-výstupní počítače, časovače, čítače, komunikační rozhraní a další periferie dle konkrétního typu obvodu. Používá se převážně pro malé jednoúčelové mikrokontroléry, používané v mnoha běžných aplikacích. Tyto procesory jsou charakteristické svojí malou kapacitou pamětí, ale především těží z výhod harvardské architektury a RISC sady, které zajišťují, že většina instrukcí může být vykonána během jednoho strojového cyklu. Výhoda rozdělené paměti spočívá v možnosti použití různých typů pamětí, především však různé bitové šířky obou pamětí. [7]

1.2 Arduino

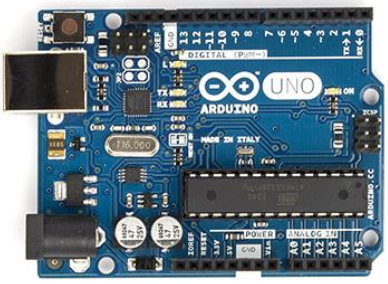
Platforma Arduino je kombinací desek osazených mikroprocesory Atmel a softwarového prostředí Arduino IDE. Vývoj Arduino začal v roce 2005 v italském městě Ivrea. Cílem bylo poskytnout studentům jednoduchou platformu pro vytváření prototypů pro vzdělávací účely. Projekt byl velmi rychle úspěšný a Arduino mělo mezi studenty velkou oblibu. Tvůrci se proto rozhodli, že Arduino nabídnou všem vývojářům po celém světě. Od začátku se jednalo o open-source,

které je volně dostupné všem uživatelům. Z tohoto důvodu je snadno dostupná dokumentace, schémata a návody. [10]

Obr. 1 – Logo platformy Arduino [2]

TYPY DESEK (Boards)

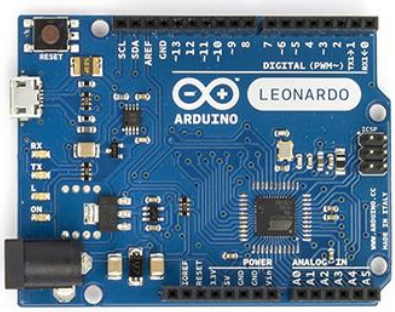
Arduino UNO



Obr. 2 – Arduino Uno [2]

Uno je v současné době zřejmě nejpoužívanějším typem desky. Deska je osazena procesorem ATmega328. Procesor může být na desce osazen ve dvou variantách a to jako integrovaný obvod umístěný v patici nebo ve verzi SMD přímo napájeném na desce. První varianta má výhodu jednoduché výměny v případě, že procesor se zničí. Pro připojení počítače k obvodu je použit USB konektor, který je standardem u větších typů desek. Arduino Uno má 14 digitálních vstupů/výstupů (6 jich může být použito jako PWM) a 6 analogových vstupů. Taktovací rychlost procesoru je 16 MHz a použitelná paměť pro program je 32 kB. [2]

Arduino Leonardo



Obr. 3 – Arduino Leonardo [2]

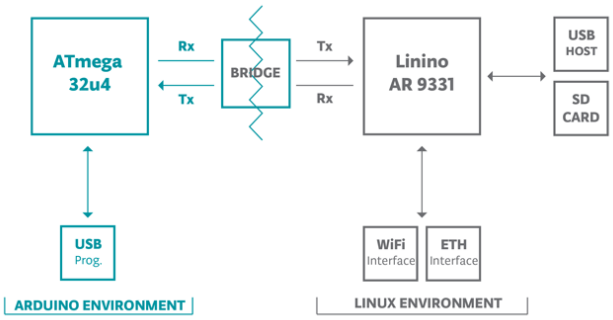
Arduino Leonardo vychází z Arduino Uno. Rozdílem je použití rozdílného procesoru – ATmega32u4. Tento procesor již v sobě integruje převodník na USB komunikaci, na desce tedy není osazený další procesor pro tuto potřebu. Deska má 20 digitálních vstupů/výstupů (7 jich může být použito jako PWM). Taktovací rychlost procesoru je 16 MHz a použitelná paměť pro program je 32 kB. [2]

Arduino Yún

Obr. 4 – Arduino Yún [2]

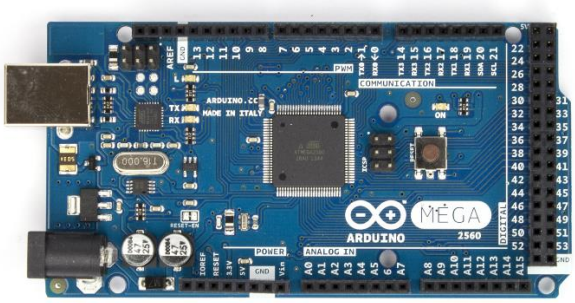
Tento model navazuje na Arduino Leonardo, kromě procesoru ATmega32u4 je osazen mikrokontrolérem Atheros AR9331. Atheros podporuje běh odlehčeného linuxu Linino. Tato deska má vestavěný Ethernet port a podporu WiFi. Ze spodní strany desky je navíc osazen micro-SD slotem. Má stejně jako Leonardo 20 digitálních vstupů/výstupů a taktovací frekvenci procesoru 16 MHz.

Na obrázku č. 5 je blokové schéma Arduino Yún se zobrazením přenosového můstku a použitelných periférií. [2]



Obr. 5 – blokové schéma Arduino Yún [2]

Arduino Mega 2560

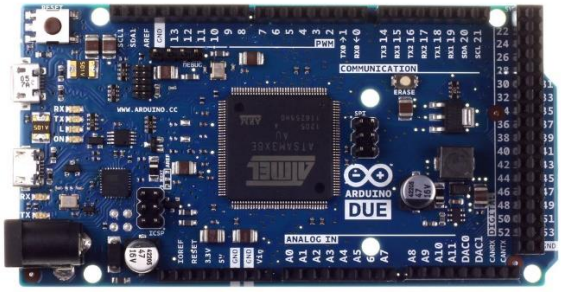


Obr. 6 – Arduino Mega 2560 [2]

Arduino Mega 2560 je jedna z desek, která je oproti základním deskám rozšířená o další vstupy a výstupy. Vzhledem k její velikosti je osazena výkonnějším mikroprocesorem ATmega2560. Má celkem 54 digitálních vstupů a výstupů. Pro sériovou komunikaci lze vyžít celkem 4 UART porty. Paměť pro program je 256 kB. Více o desce Arduino Mega 2560 viz kapitola 4.

V portfoliu Arduino desek je ještě jedna varianta desky – Arduino Mega ADK, která má shodné parametry jako Arduino Mega 2560, ale navíc je osazena USB host portem, pro připojení k telefonům s operačním systémem Android. [2]

Arduino Due



Obr. 7 - Arduino Due [2]

Typově vychází z desky Arduino Mega 2560, ale má výkonnější 32-bitový procesor AT91SAM3X8E na frekvenci 84 MHz. Na desce nalezneme dvojici microUSB konektorů. [2]

Kromě výše zmíněných desek Arduino, které můžeme označit jako nejpoužívanější, je portfolio daleko širší. Výběr vhodné desky vždy závisí na konkrétním projektu. Následující tabulka zobrazuje parametry všech desek pro rychlé porovnání.

Název	Processor	Vstupní napájení	Rychlost CPU	Analogové vstupy	Digitální IO / PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	UART
Uno	ATmega328	5 V/7-12 V	16MHz	6/0	14/6	1	2	32	Standard	1
Due	AT91SAM3X8E	3,3 V/7-12 V	84 MHz	12/2	54/12	-	96	512	2 Micro	4
Leonardo	ATmega32u4	5 V/7-12 V	16MHz	12/0	20/7	1	2,5	32	Micro	1
Mega 2560	ATmega2560	5 V/7-12 V	16MHz	16/0	54/15	4	8	256	Standard	4
Mega ADK	ATmega2560	5 V/7-12 V	16MHz	16/0	54/15	4	8	256	Standard	4
Micro	ATmega32u4	5 V/7-12 V	16MHz	12/0	20/7	1	2,5	32	Micro	1
Mini	ATmega328	5 V/7-9 V	16MHz	8/0	14/6	1	2	32	-	-
Nano	ATmega168 ATmega328	5 V/7-9 V	16MHz	8/0	14/6	0,512 1	1 2	16 32	Mini-B	1
Ethernet	ATmega328	5 V/7-12 V	16MHz	6/0	14/4	1	2	32	Standard	-
Esplora	ATmega32u4	5 V/7-12 V	16MHz	-	-	1	2,5	32	Micro	-
ArduinoBT	ATmega328	5 V/2,5-12 V	16MHz	6/0	14/6	1	2	32	-	1
Fio	ATmega328P	3,3 V/3,7-7 V	8MHz	8/0	14/6	1	2	32	Mini	1
Pro (168)	ATmega168	3,3 V/3,35-12 V	8MHz	6/0	14/6	0,512	1	16	-	1
Pro (328)	ATmega328	5 V/5-12 V	16MHz	6/0	14/6	1	2	32	-	1
Pro Mini	ATmega328	3,3 V/3,35-12 V 5 V/5-12 V	8MHz 16MHz	6/0	14/6	0,512	1	16	-	1
LilyPad	ATmega168V ATmega328V	2,7-5,5 V/2,7-5,5 V	8MHz	6/0	14/6	0,512	1	16	-	-
LilyPad USB	ATmega32u4	3,3 V/3,8-5V	8MHz	4/0	9/4	1	2,5	32	Micro	-
LilyPad Simple	ATmega328	2,7-5,5 V/2,7-5,5 V	8MHz	4/0	9/4	1	2	32	-	-
LilyPad SimpleSnap	ATmega328	2,7-5,5 V/2,7-5,5 V	8MHz	4/0	9/4	1	2	32	-	-
Yun	ATmega32u4	5 V	16MHz	12/0	20/7	1	2,5	32	Micro	1

Tab. 1 – Srovnávací tabulka parametrů desek Arduino [2]

1.3 Arduino rozšiřující moduly

Desky Arduino jsou díky vstupům a výstupům snadno rozšiřitelné a výrobce přímo poskytuje další rozšiřující moduly. Dále se můžeme setkat s neoriginálními

rozšiřujícími moduly, jejichž možnosti jsou v podstatě neomezené a záleží na konkrétním vývojáři, jakou funkcionalitu implementuje a to jak po hardwarové stránce v podobě modulu nebo softwarové v podobě knihovny. Moduly se nasazují přímo na konektory desek Arduino viz obrázek č. 8.

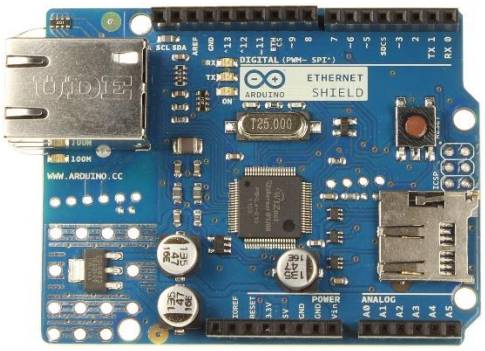
Obr. 8 – Arduino Uno s GSM a Ethernet modulem [11]

Arduino GSM modul

Obr. 9 – Arduino GSM modul [2]

Tento modul slouží pro připojení k internetu pomocí GPRS dat. Obsahuje slot na SIM kartu a konektor pro připojení antény. Modul dále umožňuje zasílání SMS a případně i hlasové služby, pokud se připojí mikrofon a sluchátko. [2]

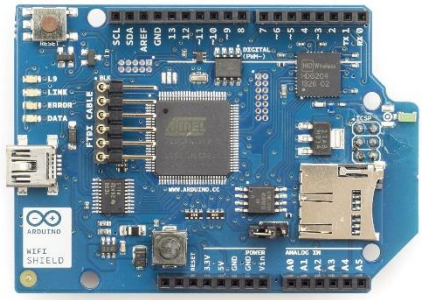
Arduino Ethernet modul



Obr. 10 – Arduino Ethernet modul [2]

Tento modul je určen pro komunikaci po počítačové síti. Je osazen konektorem RJ45. Parametry připojení jsou 10/100 Mbps. K desce Arduino se připojuje pomocí SPI portu. Navíc obsahuje micro-SD slot pro paměťové karty. [2]

Arduino WiFi modul



Obr. 11 – Arduino WiFi modul [2]

Modul pro bezdrátové připojení k počítačové síti. Pro připojení využívá protokol 802.11b/g s šifrováním WEP nebo WPA2. K desce Arduino se připojuje pomocí portu SPI a zároveň obsahuje i micro-SD slot pro paměťové karty. Na modulu je osazen mini USB port pro aktualizaci firmware modulu. [2]

Jak již bylo zmíněno na začátku této kapitoly, tak možnosti rozšiřujících modulů jsou téměř neomezené. Mezi další používané moduly rozhodně patří např. modul pro SD paměťovou kartu, USB host modul, modul pro ovládání servomotorů, moduly s LCD displejem a další.

2 Komunikace mikrokontroléru s okolím

2.1 Vstupně/výstupní porty

Vstupně/výstupní subsystém je důležitou součástí každého počítače, který zprostředkovává vstup dat z okolního prostředí do počítače a naopak výstup dat z počítače směrem ven. Data mohou vstupovat ze standardních zařízení jako je například klávesnice, digitální a analogové senzory, herní ovladače aj. a vystupovat na LCD displej, paměťové karty, tiskárny apod.

Pro spojení s řízeným technologickým procesem je obvykle nutné navrhnout vlastní rozhraní s příslušnou programovou obsluhou. Pro vstup/výstup signálu se používají specializované programovatelné obvody, které podle obsahu konfiguračního registru dokáží měnit svoje vlastnosti (přepínat piny jako vstupní nebo výstupní, parametry sériové komunikace apod.). [6]

Mikrokontroléry Atmel mají dle typu procesoru několik analogových vstupů, digitálních vstupů/výstupů a speciální rozhraní pro komunikaci. Viz následující kapitoly.

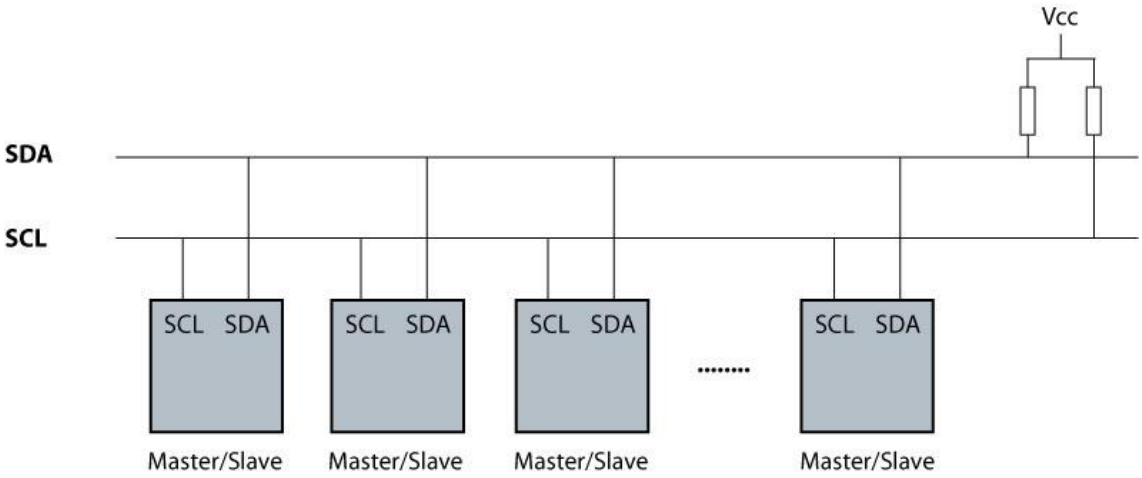
2.2 I^2C

Komunikace probíhá na principu MASTER/SLAVE. Každé připojené zařízení má svoji adresu. V okamžiku zahájení komunikace vyšle MASTER zařízení adresu jiného zařízení, se kterým chce navázat spolupráci. Všechna zařízení, která jsou připojena na sběrnici naslouchají. Zařízení, které identifikuje svoji adresu potvrdí přijetí komunikace vysláním signálu – ACK. [3]

Jedná se o oboustrannou sériovou komunikaci vytvořenou firmou Philips Semiconductors, která bývá někdy také označováno jako TWI. Pro komunikaci mezi zařízeními je použito dvou vodičů. Jeden z nich je použit jako datový vodič a druhý plní funkci hodinového signálu. Datový vodič je označovaný jako SDA, hodinový signál jako CLK.

Na SDA i na CLK musí být připojen zdvihací „pull-up“ rezistor o velikosti několika kilo ohmů. Tímto ochráníme obvody v situaci, kdy by chtěl vysílat zároveň více než jeden obvod. Hodinový signál a obvod, s kterým se bude komunikovat, určuje zařízení označované jako MASTER.

Toto zařízení je na sběrnici pouze jedno a ostatní připojené obvody se označují jako SLAVE. Strukturu obvodů MASTER a SLAVE připojených na sběrnici I²C lze vidět na obr. 12.



Obr. 12 – Blokové zapojení sběrnice I2C [5]

Komunikace mezi jednotlivými zařízeními probíhá v následujících etapách. Jako první MASTER vyšle start-bit označovaný jako S a zahájí komunikaci po sběrnici. Následující osmice bitů obsahuje adresu obvodu a řídící bit, který je na pozici D0 a určuje směr přenosu. Adresa je pevně zabudovaná v obvodu, ovšem tři bity bývají většinou volitelné, pro případ použití jednoho obvodu vícekrát. Pokud spojení proběhlo úspěšně, potvrdí připojený obvod spojení potvrzovacím bitem, který je značený jako ACK. Po potvrzení spojení obvodu pokračuje komunikace dalšími osmi datovými bity. Komunikace se ukončí stop-bitem, který je označovaný jako P. Start-bit je proveden tehdy, pokud se změní stav na datové lince SDA log 1 do log 0 za předpokladu, že je CLK ve stavu log 1. Naopak stop-bit určuje přechod linky SDA z log 0 do stavu log 1, při vysoké úrovni hodinového signálu. Z těchto podmínek pro řídící signál plyne, že při komunikaci musí mít datová linka SDA ustálený stav

během celého hodinového impulsu, jinak by ho komunikující obvody braly jako signál řídící start popřípadě stop. [13]

2.3 RS485

Jedná se o asynchronní komunikační standard, který se používá především v průmyslu. Na rozdíl od RS232 nejsou jeho logické stavy vyjádřeny hodnotou proti společné zemi, ale jsou vyjádřeny pomocí rozdílového napětí na dvou vodičích označovaných A a B. Díky tomu, že jsou tyto diferenciální vodiče vedeny stejnou cestou a rušení se na nich projeví stejným způsobem, je toto rušení téměř nulové pro rozhodovací úrovně rozdílových signálů. Tím lze dosáhnout delších komunikačních vzdáleností a také větších rychlostí. Běžně se dosahuje rychlostí 2,5 MB/s až 10 MB/s na krátké vzdálenosti, maximální udávaná vzdálenost je kolem 1200 m, pokud je správně provedeno impedanční zakončení a je použita kroucená dvojlinka. Další důležitou vlastností standardu RS485 je, že eliminuje rozdíl zemnících potenciálů mezi přijímačem a vysílačem. To je v praxi výhodné, pokud jsou systémy napájené jinými napájecími zdroji.

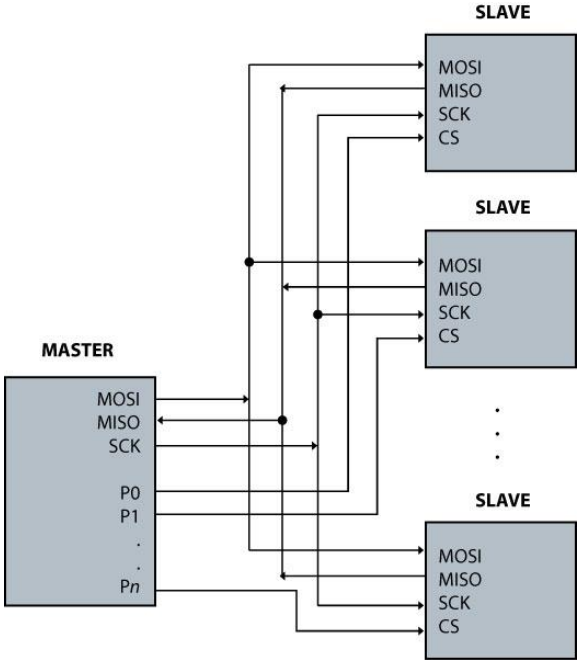
RS485 se používá pro komunikace s více zařízeními, které jsou spojeny jedním signálovým vedením. Většinou je použita architektura MASTER a SLAVE zařízení, přičemž každé komunikující zařízení má vlastní adresu a přijímá pouze pakety, které jsou určeny přímo pro dané zařízení. Pakety generuje MASTER, což může být třeba osobní počítač, který periodicky posílá zprávy všem připojeným SLAVE zařízením.

Komunikace může probíhat po jednom diferenciálním páru, kde je komunikace po tomto páru v obou směrech. Druhá možnost je, že SLAVE zařízení mají vlastní diferenciální pár pro posílání dat do MASTERU a MASTER má opět vlastní diferenciální pár pro zasílání zpráv do SLAVE zařízení.

Pro komunikaci se používá tzv. vyvážený diferenciální pár (Balanced differential pair), kdy vysílač generuje napětí v rozmezí od 2 V až do 7 V mezi výstupy A a B. [13]

2.4 SPI

SPI (Serial Peripheral Interface) je sériová synchronní datová sběrnice, která pracuje v plně duplexním režimu. Z toho plyne, že zařízení mohou zároveň vysílat i přijímat data. Tato komunikace probíhá po čtyřech datových linkách a to MOSI (Master Output, Slave Input), MISO (Master Input, Slave Output), CLK a SS (Slave Select). Jak název napovídá, linka MOSI slouží pro přenos z MASTER obvodu do vybraného SLAVE zařízení. Opačný 25 směr komunikace zajišťuje signál MISO. Následující signál CLK slouží pro hodinové impulsy a SS slouží k výběru SLAVE zařízení. SLAVE zařízení mají tří-stavové výstupy, které jsou ve stavu vysoké impedance tehdy, pokud není zařízení vybráno ke komunikaci. Komunikace může probíhat mezi dvěma nebo i více zařízeními. Vždy je jedno zařízení MASTER, které ovládá hodinový signál a výběr zařízení, S kterým proběhne komunikace. Pokud je na sběrnici pouze jedno SLAVE zařízení, lze jeho pin SS uzemnit, pokud to použitý obvod dovoluje. Zapojení jednoho MASTER a SLAVE zařízení je vidět na obr. č. 13.



Obr. 13 – Zařízení na sběrnici SPI [6]

Komunikace mezi zařízením MASTER a SLAVE je provedena následujícími kroky. Jako první MASTER nastaví frekvenci hodinových taktů, která se může pohybovat

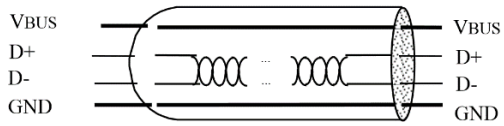
mezi 1–70MHz. Maximální možnou frekvenci určují možnosti zařízení SLAVE. Dále zařízení MASTER vybere, s kterým zařízením SLAVE bude komunikovat a to tím, že jeho SS nastaví na log 0. Někdy může být vyžadováno čekání před samotnou komunikací, které MASTER musí dodržovat. Toto je například uplatněno při A/D převodech. Dále jsou s hodinovými impulsy vysílány datové bity ze zařízení MASTER do SLAVE přes MOSI linku a přijímány data v zařízení MASTER ze SLAVE pomocí linky MISO. Vše probíhá v plném duplexním režimu.

Před počátkem komunikace je také nutné nastavit fázi a polaritu přijímání dat v závislosti na hodinovém kmitočtu. Nastavení těchto parametrů lze provést v registru SPCR a to nastavením bitů CPHA a CPOL do stavu log 1. Pokud je bit CPHA spolu s bitem CPOL roven log 0 jsou data vzorkována s náběžnou hranou hodinového impulsu. Pokud je ovšem bit CPOL roven log 1 při log 0 na bitu CPHA, jsou data vzorkována sestupnou hranou hodinového impulsu. Pro stav bitu CPHA, kdy je ve vysoké logické úrovni, platí inverzně předchozí stavy řízení. Bitem CPOL se také řídí stav CLK při neaktivním přenosu. Při nastavení log 0 na bitu CPOL je CLK také v nízké úrovni, naopak oproti tomu CPOL ve stavu log 1 zajistí i vysokou logickou úroveň na pinu CLK. [13]

2.5 USB

Rozhraní USB bylo navrhnuto společnostmi Compaq, DEC, IBM, Intel, Microsoft, NEC a Nortel v roce 1994. Cílem bylo zjednodušit komunikaci mezi zařízeními a sjednotit několik různých konektorů, které se používali do té doby, do jednoho rozhraní. V roce 1996 byl představen standard USB 1.0. Byly definovány dvě přenosové rychlosti 1,5 Mbit/s označované jako „Low Speed“ a 12 Mbit/s označovaného jako „Full Speed“. [8]

Obr. 14 – Grafická značka USB [7]



Obr. 15 – Zapojení USB kabelu [8]

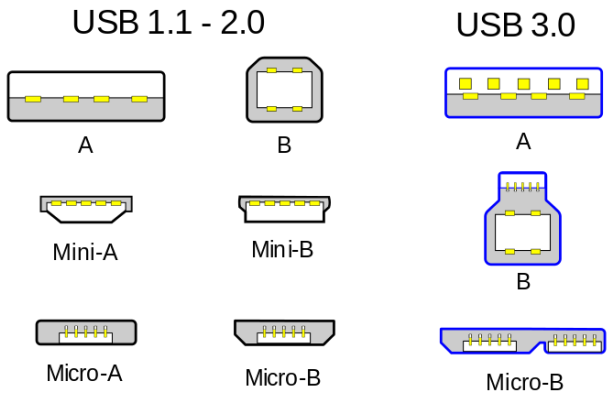
V současné době je sběrnice USB nejpoužívanějším asynchronním sériovým rozhraním.

USB 1.1
Představeno v roce 1998. Jednalo se o první šíře použitelný standard. Přenosové rychlosti jsou shodné s USB 1.0.

USB 2.0
Tento standard byl představen v roce 2000 a přinesl větší přenosovou rychlost 480 Mbit/s označovanou jako „High Speed“.

USB 3.0
Verze 3.0 byla ohlášena již v roce 2008, ale k širšímu nasazení došlo až v roce 2010. Disponuje více než 10x rychlostí standardu 2.0, přenosová rychlost je 5 Gbit/s a označována jako „Super Speed“. Nová technologie má 9 vodičů, což přineslo i změnu konektoru, který je zpětně kompatibilní (Typ A).

USB 3.1
Představen v roce 2013 jako vylepšení verze 3.0. Měla by přinést dvojnásobnou rychlost 10 Gbit/s a je označována jako „Superspeed+“.



Obr. 16 – Typy USB konektorů [7]

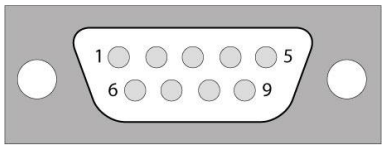
2.6 RS232

RS232 je asynchronní standard, který obsahuje bipolární inverzní reprezentaci logických úrovní. Vysoká logická úroveň je reprezentována hodnotou -3 V až -15 V, zatímco nízká logická úroveň je reprezentována hodnotou napětí 3 V až 15 V. V počítačové technice se používají úrovně ±12 V, kvůli větší odolnosti vůči rušení. [4]

K připojení zařízení, které spolu komunikují pomocí RS232 jsou používány konektory Cannon 25, Cannon 9 a RJ45. Nejčastější použití je pomocí konektoru Cannon 9. Oba konektory jsou popsány následující tabulkou č. 2 a obrázkem č. 17.

Signál	Canon 9	Canon 25	Popis
DCD	1	8	Data Carrier Detect
RxD	2	3	Receive Data
TxD	3	2	Transmit Data
DTR	4	20	Data Terminal Ready
GND	5	7	System Ground
DSR	6	6	Data Set Ready
RTS	7	4	Request to Send
CTS	8	5	Clear to Send
RI	9	22	Ring Indicator

Tab. 2 – Zapojení konektorů Canon 9 a Canon 25 [3]



Obr. 17 – Konektor Canon 9 [3]

2.7 Další typy sériové komunikace

Pro úplnost uvádím další významné typy sériové komunikace:

CAN-BUS

Jedná se o sběrnici, která se nejčastěji používá pro vnitřní komunikaci v automobilovém průmyslu. Z této aplikační oblasti se CAN rozšířil i do sféry průmyslové automatizace. Sériová datová sběrnice, vyvinutá společností Robert Bosch GmbH. Maximální teoretická rychlost přenosu je 1 Mb/s. [7]

MIDI (Musical Instrument Digital Interface)

Mezinárodní technický standard používaný v hudebním průmyslu, který popisuje protokol, digitální rozhraní a konektory, umožňující vzájemné propojení hudebních nástrojů a počítačů. [14]

SATA

Označuje počítačovou sběrnici, využívající datové rozhraní k připojení velkokapacitních paměťových zařízení jako jsou pevné a optické disky. V současné době se jedná o nejrozšířenější rozhraní používané v osobních počítačích pro připojení pevného disku. Aktuální revize SATA verze 3.0 umožnuje komunikovat v přenosové rychlosti 6 Gb/s, které využívají pevné disky s technologií SSD. Pro připojení externích datových zařízení jako např. mechaniky optických disků se používá odvozený standard eSATA. Ten je teoreticky schopen zvládnout stejné přenosové rychlosti jako SATA, čímž je rychlejší než USB 3.0, ale na rozdíl od USB kabel eSATA nemá integrované napájení. [7]

Ethernet

Nejpoužívanější protokol pro lokální i vzdálené počítačové sítě. Síť Ethernet byla vyvinuta v letech 1973–1975 ve výzkumných laboratořích Xerox. Pro komerční

použití byla představena v roce 1980 a v roce 1983 byla přijat jako standard IEEE 802.3. Ethernet je tedy k dispozici již více než 35 let a celou dobu se postupně vyvíjí. V současné době se k němu vztahuje několik dalších norem. Pod označením kategorie 802 se můžeme ještě setkat např. s 802.11 (WiFi) nebo 802.15 (Bluetooth). [15]

Důležité podkategorie pro 802.3 jsou: [7][15]

802.3	původní protokol 10 Mb/s Ethernet. Označení 10BASE5 a 10BASE2 znamenalo použití koaxiálního kabelu. Kroucená dvoulinka měla označení 10BASE-T.
802.3u	100 Mb/s Ethernet. V součastnosti se používá norma 100BASE-TX po dvou párech na UTP kabelu kategorie 5 a vyšší.
802.3z	Gigabitový Ethernet na optice. Označován jako 1000BASE-SX/LX/CX.
802.3ab	Gigabitový Ethernet na UTP. 1000BASE-T specifikuje komunikaci se 4 páry, každý pár rychlostí 125 Mbps.
802.3ae	10 gigabitový Ethernet na optických vláknech. Standard norem 10GBASE-SR/LR/ER odlišnou dosahem signálu.
802.3an	10 gigabitový Ethernet na UTP. 10GBASE-T pro kategorie kabelů 6A a lepší.
802.3ba	40 gigabitový a 100 gigabitový Ethernet.
802.3bm	100 gigabitový Ethernet. Pokud bude vyvinut, tak na přenos bude potřeba poloviční množství kabelů než u současných protokolů. Plánován pro vývoj (2015).
SG	400 gigabitový Ethernet. Plánován pro vývoj (2016).

3 Programování ve vyšších programovacích jazycích

3.1 Bascom-AVR

Programovací jazyk Bascom je vyšší programovací jazyk s kompilátorem, vyvinutý v 90. letech za účelem programování jednočipových mikroprocesorů Atmel. Vývojové prostředí je určeno pouze pro platformu Windows a je kompatibilní s verzemi operačního systému XP/Vista/Windows 7 a Windows 8 [17]. Definice jazyka měla za cíl vytvořit jednoduše použitelný programovací jazyk k programování jednočipových počítačů bez znalosti jazyka Assembler s možností konfigurace a využívání dostupných periferií procesoru, snadné nastavení vnitřních registrů a obsluhou přerušení. Jeho výhodou jsou předem definované operace, cykly, práce s proměnnými různých velikostí, rozhodování, funkce pro práci s textem, předdefinované hardwarové operace s diskrétními součástkami, definice vlastních funkcí a procedur. [18]

Jazyk Bascom je inspirován vyšším programovacím jazykem Basic. Jeho příkazy, syntaxe a další použité struktury v jazyce připomínají Visual Basic. Jazyk Bascom z Basicu přebírá i velmi vysokou míru abstrakce, což jej činí velmi snadno pochopitelným a vhodným i pro úplné začátečníky. [18]

Jazyk Bascom disponuje vlastním vývojovým prostředím s kompilátorem. Přestože je k dispozici zdarma, omezuje velikost výsledného kompilovaného programu na 4 kB. [17]

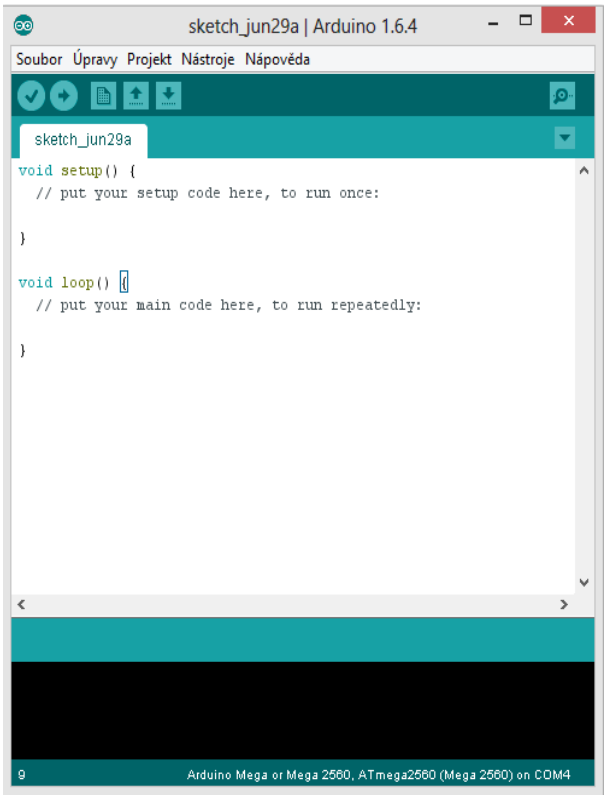
3.2 Wiring

Wiring je jazyk pro programování elektronických prototypů na základě jednočipových mikrokontrolérů. Je postaven na základech jazyka Processing. Patří do open-source programovacích jazyků. [16]

Více o programování v jazyce Wiring v následující kapitole.

3.3 Arduino IDE

Pro programování mikročipů AVR Atmel je vhodné použít softwarové prostředí Arduino IDE, která doplňuje platformu Arduino. Programovací jazyk vychází z jazyka Wiring. Software je vyvíjen pod licencí open-source, jeho použití je zdarma. Zároveň jsou dostupné knihovny pro různé moduly apod. od vývojářů z celého světa. Programování mikrokontroléru probíhá připojením desky Arduino k USB portu PC (podporované jsou i platformy Mac OS a Linux). [2]



Obr. 18 – Programovací prostředí Arduino [zdroj: autor]

Stručný úvod standardních instrukcí pro použití programování Arduino pomocí jazyka Wiring [2][19].

STRUKTURA

Sketch Arduino se dělí na dvě základní části:

```
void setup()
```
Zde se umístí inicializační kód. Obsahuje soubor instrukcí, které nastaví parametry před spuštěním hlavní smyčky.

```
void loop()
```
Obsahuje hlavní kód. Jedná se o soubor instrukcí, které se opakují stále dokola do té doby než je deska odpojena od napájení.

SPECIÁLNÍ SYMBOLY

Arduino obsahuje několik symbolů, které jsou určeny pro oddělení řádek kódu, komentáře nebo bloky kódu.

; (středník)
Každá instrukce je ukončena středníkem.

Příklad:

```
digitalWrite(13,HIGH); // Pin 13 prepni na uroven 1
```

{} (složené závorky)
Označují blok kódu. Pokud například píšeme kód funkce loop(), použijeme složené závorky před a za kódem.

Příklad:

```
void loop() { // Zacatek bloku
      Serial.println("Ahoj svete"); // Zapis na seriovou linku text
} // Konec bloku
```

KOMENTÁŘE

Jedná se o text, který je ignorován Arduino procesorem, ale je velmi užitečný pro upřesnění kódu pro vlastní potřebu. Přestože je v okamžiku programování vše zřejmé, tak po vrácení se ke kódu s odstupem času se při použití komentářů v kódu jednodušeji zorientujeme.

Arduino umožňuje komentáře napsat ve dvou variantách:

```
// Jednoradkovy zapis, text je ignorovan do konce radky
/* Nekolik radku
Vsechny radky mezi symboly jsou ignorovany
*/
```

KONSTANTY

Arduino obsahuje několik předdefinovaných klíčových slov se speciální hodnotou. HIGH a LOW se používají pro nastavení signálu na Arduino pinech. INPUT a OUTPUT slouží pro nastavení specifického pinu jako vstup nebo výstup.

True a False se používají v podmínkách nebo výrazech pro označení logických hodnot PRAVDA/NEPRAVDA.

PROMĚNNÉ A DATOVÉ TYPY

Proměnná je vyhrazené místo v paměti Arduino, ve kterém lze ukládat data, používaná ve skriptu. Hodnota proměnné může být v průběhu skriptu bez omezení měněna.

Při definici proměnné se vždy musí uvést její typ. Typ označuje velikost hodnoty, kterou chceme uložit do paměti.

boolean	Nabývá hodnot true nebo false
char	Uchovává jeden znak, například „A“. Arduino jej ukládá jako číslo v hodnotách od 0 do 255 dle ASCII tabulky.
byte	Uchovává číslo v rozsahu mezi 0 a 255. Podobně jako char, využívá pouze jeden byte paměti.
int	Využívá 2 byty paměti pro uchování čísla v číselném rozsahu mezi -32 768 a 32 767. Jedná se o jeden z nejpoužívanějších typů.

unsigned int Podobně jako int využívá 2 byty paměti ale bez záporných čísel. Rozsah je mezi 0 a 65 535.

long Dvojnásobná velikost než je int, rozsah je mezi -2 147 483 648 do 2 147 483 647

unsigned long Podobně jako unsigned int nepoužívá záporná čísla pro typ long. Rozmezí hodnot od 0 do 4 294 967 295.

float Poměrně veliká proměnná může obsahovat hodnoty s plovoucí řádovou čárkou, neceločíselná čísla s desetinnou čárkou. Využívá 4 byty paměti a může uchovávat číselné hodnoty od $-3{,}4028235 \cdot 10^{-38}$ do $3{,}4028235 \cdot 10^{38}$

double Při použití této proměnné u desek Uno a Mega je shodný jako typ float. U desky Due má dvojnásobnou velikost 8 byte. Rozsah hodnot od $-1{,}7 \cdot 10^{-308}$ do $1{,}7 \cdot 10^{308}$

string Řetězec ASCII znaků pro uchování textové informace. Pro každý znak je použit 1 byte paměti plus 1 byte pro znak ukončení řetězce.

Příklad:

```
char retezec1[] = "Arduino"; // 7 znaku + 1 ukoncovaci znak
char retezec2[8] = "Arduino"; // Jina forma zapisu
```

array Pole proměnných, které jsou dostupné pomocí indexu.

Příklad:

```
int pinArray[] = {2, 3, 4, 5, 6, 7};
// pinArray[0] obsahuje 2 apod.
```

ŘÍDÍCÍ STRUKTURY

if ... else

Tato struktura tvoří rozhodnutí v kódu programu. Za `if` musí následovat podmínka. Pokud je podmínka splněna – true, tak je spuštěn následující výraz za podmínkou. Pokud splněna není – false, tak je spuštěn výraz následující za klauzulí `else`. Klauzule `else` není povinná a může být vynechána.

Příklad:

```
if (hodnota == 1) { // Pokud je splnena podminka
	digitalWrite(LED, HIGH); // Zapni diodu LED
}
else
{
	digitalWrite(LED, LOW); // Vypni diodu LED
}
```

for

Zajistí několikanásobné opakování bloku kódu.

Příklad:

```
for (int i = 0; i < 10; i++) { // Pocet opakovani
	Serial.print ("Ahoj"); // Zapis na seriovou linku
}
```

switch case

Podobně jako podmínka `if`, tak i `switch..case` ovládá běh programu.

Příklad:

```
switch (hodnota) {
	case 1:
	// Proved blok kodu pokud je hodnota rovna 1
	break;
	case 2:
	// Proved blok kodu pokud je hodnota rovna 2
	break;
	default:
	// Pokud se hodnota nerovna ani jedne z moznosti proved vychozi kod
	// Vychozi kod default je nepovinny
}
```

while

Podobně jako `if`, `while` provádí blok určitý blok instrukcí, dokud platí podmínka.

Příklad:

```
// Blikani LED diody pokud ma senzor nizsi hodnotu nez 512
senzorHodnota = analogRead(1);
while (senzorHodnota < 512) {
	digitalWrite(13, HIGH);
	delay(100);
	digitalWrite(13,LOW);
	delay(100);
	sensorValue = analogRead(1);
}
```

do ... while

Stejné jako while, ale podmínka je umístěna až na konci bloku. Tento příkaz se používá, pokud potřebujeme, aby hlavní kód proběhl minimálně jednou než je splněna podmínka.

Příklad:

```
// Blikani LED diody pokud ma senzor nizsi hodnotu nez 512 s pouzitim
do {
	digitalWrite(13, HIGH);
	delay(100);
	digitalWrite(13, LOW);
	delay(100);
	sensorValue = analogRead(1);
} while (senzorHodnota < 512);
```

break

Použijeme k opuštění smyčky a k pokračování spuštění programu, který je za smyčkou. Zároveň tento příkaz použijeme k rozdělení sekcí příkazu `switch`.

Příklad:

```
// Blikani LED diody pokud ma senzor nizsi hodnotu nez 512
do {
	// Opusti smycku pokud je stisknuto tlacitko
	if (digitalRead(7) == HIGH)
		break;
	digitalWrite(13, HIGH);
	delay(100);
	digitalWrite(13,LOW);
	delay(100);
	sensorValue = analogRead(1);
} while (senzorHodnota < 512);
```

continue

Pokud je tento příkaz použit uvnitř smyčky, příkaz `continue` zajistí vynechání programu a pokračuje v testování podmínky.

Příklad:

```
for (intenzita = 0; intenzita < 255; intenzita ++)
{
	// Vynecha intenzitu v rozmezi mezi 150 a 200
	if ((x > 150) && (x < 200))
		continue;
analogWrite(pinPWM, intenzita);
delay(10);
}
```

return

Zastaví běžící funkci a vrací výsledek.

Příklad:

Pokud máme například funkci s názvem `pocitejTeplotu()` a potřebujeme vrátit výsledek do části kódu, který funkci zavolal, použijeme příkaz `return` následovně.

```
int pocitejTeplotu() {
      int teplota = 0;
      teplota = (analogRead(0) + 45) / 100;
      return teplota;
}
```

SROVNÁVACÍ OPERÁTORY

Používají se v řídících strukturách `if`, `while` nebo `for`.

==	rovno
!=	není rovno
<	menší než
>	větší než
<=	menší nebo rovno
>=	větší nebo rovno

LOGICKÉ OPERÁTORY

Tyto operátory se používají, pokud je potřeba kombinovat více podmínek. Například potřebujeme zkontrolovat hodnotu mezi 5 a 10:

```
if ((senzor => 5) && (sensor <= 10))   // Pokud je hodnota mezi 5 a 10
```

Operátory jsou celkem tři. AND reprezentován znaky „&&“, OR reprezentován „||“ a NOT reprezentován „!“

INPUT/OUTPUT FUNKCE

Arduino zahrnuje funkce, které zajišťují vstup a výstup. Jsou to:

pinMode(pin, mode)
Nastavuje digitální piny jako vstupní nebo výstupní.

Příklad:

```
pinMode (7, INPUT);       // Nastavi pin 7 jako vstup
```

digitalWrite(pin, value)
Zapíná nebo vypíná výstupní pin nastavením log 0 a log 1.

Příklad:

```
digitalWrite(8, HIGH);    // Zapne digitalni pin 8
```

int digitalRead(pin)
Čte stav vstupního pinu a vrací HIGH pokud je na vstupu nějaké napětí nebo LOW pokud na vstupu není žádné napětí.

Příklad:

```
val = digitalRead(7);           // Nacita stav pinu 7 do promenne val
```

int analogRead(pin)
Zjišťuje napětí na analogovém vstupu a vrací číslo mezi 0 a 1023, která reprezentuje napětí mezi 0 a 5 V.

Příklad:

```
val = analogRead(0);      // Nacita analogovy vstup na pinu 0 do promenne val
```

analogWrite(pin, value)
Mění modulaci PWM na některém z PWM pinů. Hodnota je mezi 0 a 255 a reprezentuje napětí mezi 0 a 5V.

Příklad:

```
analogWrite(9, 128);      // Ztlumi LED diodu na pinu 9 na 50%
```

ČASOVÉ FUNKCE

Arduino pracuje s funkcemi, které měří čas nebo slouží k pozastavení kódu.

unsigned long millis()
Vrací počet milisekund od spuštění programu.

delay(ms)
Zastavuje program na dobu uvedenou v závorce a měřenou v milisekundách.

delayMicroseconds(µs)
Zastavuje program na dobu uvedenou v parametru v mikrosekundách.

MATEMATICKÉ FUNKCE

Arduino pracuje s mnoha matematickými a trigonometrickými funkcemi. Některé z nich jsou následující:

min(x, y) Funkce vrací minimální hodnotu

max(x, y) Funkce vrací maximální hodnotu

abs(x) Funkce vrací absolutní hodnotu

constrain(x, a, b) Funkce vrací hodnotu *x* v rozmezí od *a* do *b*. Pokud je hodnota nižší než *a*, vrací *a*. Pokud je hodnota vyšší než *b*, vrací *b*.

FUNKCE RANDOM

Pokud je potřeba generovat náhodná čísla, tak lze použít pseudo-generátor Arduino.

randomSeed(seed)

Resetuje Arduino pseudo-generátor. Přestože je funkce `random()` teoreticky náhodná, tak její chování je předvídatelné. Z tohoto důvodu je potřeba resetovat generátor, abychom dostali skutečně náhodné číslo. Pokud máme nepřipojení analogový vstup, tak tak funkce z tohoto pinu načte náhodný ruch, který je způsoben radiofrekvenčními vlnami, kosmickým zářením, rušením z mobilních sítí apod.

Příklad:

```
randomSeed(analogRead(5));      // Randomizuje ze vstupu pinu 5
```

long random(max)
long random(min, max)

Vrací náhodné číslo v rozmezí *min* a *max*, pokud *min* není specifikováno, tak dolní hranice je 0.

SÉRIOVÁ KOMUNIKACE

Pro potřeby komunikace s dalšími zařízeními jsou použity následující funkce:

Serial.begin(speed)
Připravuje Arduino na komunikaci pomocí sériového portu. Hodnoty jsou v rozmezí od 9600 bps do 115200 bps.

Serial.print(data)
Serial.println(data)
Odesílá data na sériový port. Druhá varianta funkce navíc přidává odřádkování, podobně jako když se stiskne klávesa Enter.

Příklad:

```
Serial.print(75);         // Odesle na seriovy port cislo 75
Serial.println(75);       // Odesle na seriovy port cislo 75 a odradkuje
```

int Serial.avaliable ()
Funkce vrací kolik nepřečtených dat je dostupných na sériové lince. Data se čtou funkcí read(). Poté, co jsou data přečtena tak je vrácena hodnota 0.

int Serial.read()
Načítá data na vstupu.

Serial.flush()
Data, která jsou na vstupu se mohou vznikat rychleji než je Arduino schopno zpracovávat. Arduino ponechává všechna příchozí data v bufferu, pokud je potřeba buffer vyprázdnit, tak se použije funkce flush().

4 Použití mikrokontroléru AVR pro sběr dat ze sériové linky

Desky Arduino s mikrokontroléry AVR jsou vhodné pro sběr dat ze sériové linky, většina desek má minimálně jeden sériový UART port pro komunikaci s dalšími zařízeními. Existují i speciální desky, které podporují jiné druhy komunikace např. Bluetooth, Ethernet či Wifi.

4.1 Výběr vhodného mikrokontroléru

Následující skutečnosti byly rozhodující pro výběr vhodného mikrokontroléru s vyžitím pozitiv platformy Arduino:

1. Jedná se o open-source, software i hardware je snadno dostupný a jednoduše se upravuje a rozšiřuje
2. Je flexibilní, nabízí množství analogový a digitálních vstupů a PWM a digitálních výstupů
3. Je jednoduchý na použití, k PC se připojuje pomocí USB konektoru. Rozhraní je možné připojit k PC nebo Mac platformě
4. Je relativně levný dá se pořídit od 133 Kč (Arduino mini ATmega328 kompatibilní klon) případně 399 Kč (Arduino mini ATmega328 originál).
5. Má velmi širokou základnu online komunity, která se rychle rozrůstá. Tím je snadno dostupná bohatá dokumentace a odborná fóra, zabývající se touto tématikou.

Vzhledem k záměru použití – analyzátor sériové linky, byl jedním z hlavních parametrů počet sériových portů. Minimální počet portů potřebný pro základní komunikaci tj. zařízení, které bude vloženo mezi dvě komunikující zařízení jsou dva, jeden pro každé zařízení. Z tohoto důvodu byla vybrána deska Arduino Mega 2560, která má čtyři hardwarové UART porty pro sériovou komunikaci. Jeden z UART portů je již hardwarově řešen jako součást desky ve formě USB konektoru tvořeném TTL převodníkem na sériovou linku, takže je možné jej jednoduše použít pro připojení k PC a to jak pro programování desky, tak analyzování/ovládání zařízení bez nutnosti použití dalšího samostatného převodníku.

4.2 Parametry desky Arduino Mega 2560

V následující tabulce jsou shrnuty základní parametry vybraného typu desky Arduino, která byla použita pro vytvoření prototypu.

Parametry

Mikrokontrolér	ATmega2560
Provozní napětí	5 V
Vstupní napětí (doporučené)	7 – 12 V
Vstupní napětí (hraniční)	6 – 20 V
Digitální vstupy/výstupy	54 (15 pro výstup PWM)
Analogové vstupy	16
SS proud pro I/O piny	40 mA
SS proud pro 3,3V piny	50 mA
Flash paměť	256 kB, 8 kB použito pro bootloader
SRAM	8 kB
EEPROM	4 kB
Frekvence	16 MHz

Tab. 3 – Vybrané parametry Arduino Mega 2560 [2]

Napájení

Arduino Mega 2560 může být napájeno pomocí USB portu nebo pomocí externího zdroje. Zdroj napájení je zvolen automaticky.

Externím zdrojem může být stejnosměrný napájecí adaptér nebo baterie. Pro napájecí zdroj je možné využít konektor s průměrem 2,1 mm, který je přímo na desce. V případě použití baterií lze zapojit přímo do desky na piny GND a V_{in}.

Napájecí napětí může být v rozmezí 6 V – 20 V. Přesto, pokud je použito napájení nižší než 7 V tak 5 V piny mohou mít nestabilní napětí nižší než 5 V a deska se chová nestabilně. Pokud se použije napětí vyšší než 12 V dochází k přehřívání a může dojít k poškození desky. Z tohoto důvodu je doporučené napětí v rozmezí od 7 V do 12 V. [2]

Paměť

Mikročip ATmega2560má 256 kB flash paměti pro uložení programu (z čehož 8 kB je použito pro bootloader), 8 kB SRAM a 4 kB EEPROM paměti. [2]

Vstupy a výstupy

Každý z 54 digitálních pinů může být použit jako vstup nebo výstup pomocí funkcí `pinMode()`, `digitalWrite()` a `digitalRead()`. Každý z nich pracuje s napětím 5 V a proudem maximálně 40 mA. Některé z pinů mají specifické funkce:

Serial: 0 (Rx) a 1 (Tx); Serial 1: 19 (Rx) a 18 (Tx); Serial 2: 17 (Rx) a 16 (Tx); Serial 3: 15 (Rx) a 14 (Tx). Použité pro přijímání (Rx) a odesílání (Tx) TTL dat. Piny 0 a 1 jsou zároveň připojené k převodníku ATmega16U2 USB na úroveň TTL.

- **Externí přerušení: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3) a 21 (interrupt 2).**
- **PWM: piny 2-13 a 44-46.** Umožňují 8–bitový PWM výstup pomocí funkce analogWrite().
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** Slouží pro podporu SPI komunikace použitím knihovny SPI.
- **LED: 13.** Vestavěná LED dioda propojena s digitálním pinem 13.
- **TWI: 20 (SDA) a 21 (SCL).** Podporuje TWI komunikaci použitím knihovny Wire.

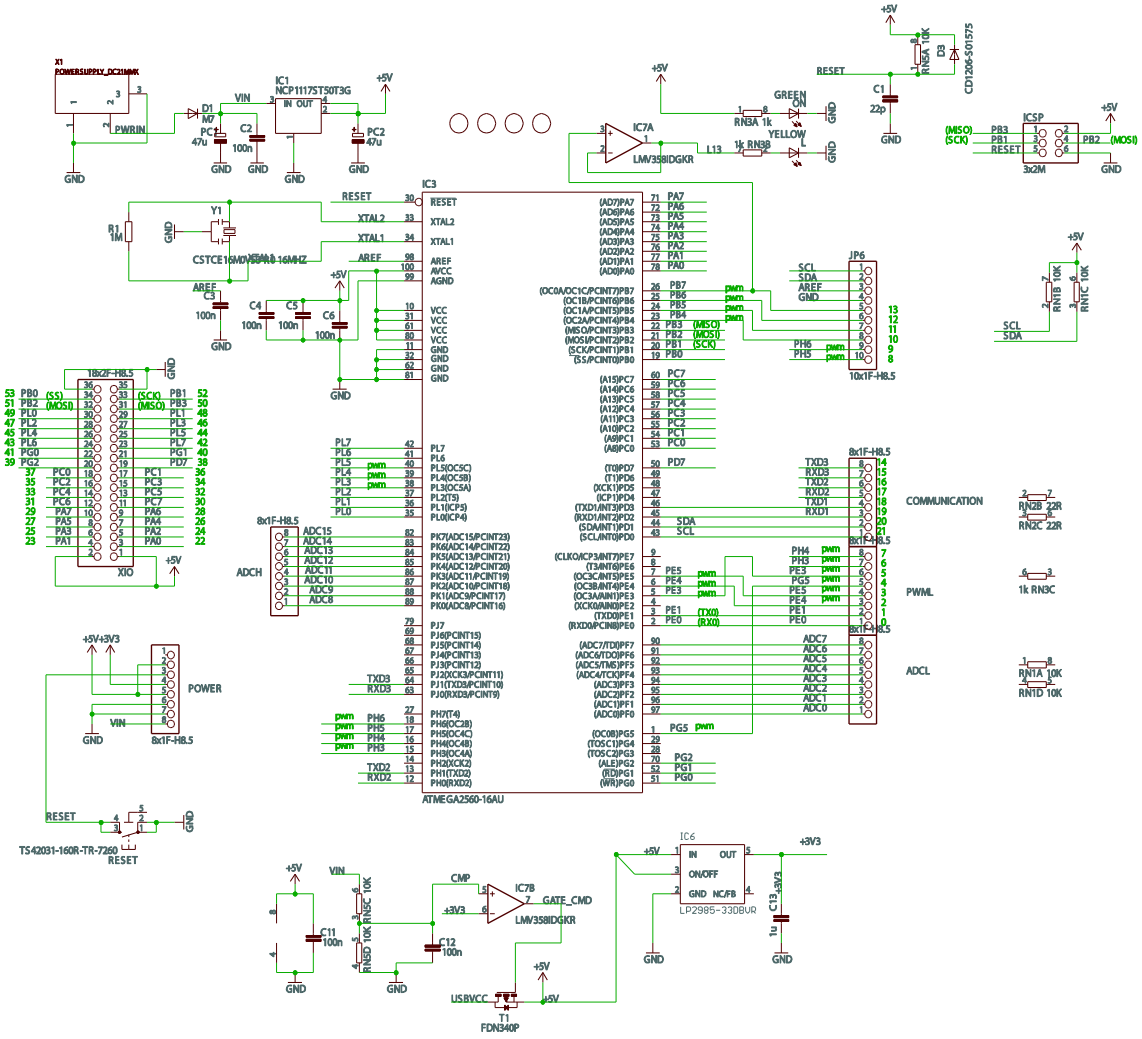
Mikročip ATmega2560 má 16 analogových vstupů, každý z nich nabízí rozlišení 10 bitů (tj. 1024 různých hodnot). Standardně pracují od 0 do 5 V. [2]

Ošetření přetížení a zkratu na portu USB

Přestože většina počítačů má svou vlastní interní ochranu před přetížením a zkratem, tak Arduino Mega 2560 má resetovatelnou pojistku přímo na desce pro zajištění zvláštní ochrany. Pokud je proud v USB vyšší než 500 mA, tak se komunikace automaticky odpojí, do té doby než je přetížení nebo zkrat odstraněn. [2]

Základní zapojení Arduino Mega 2560

Základní zapojení mikročipu ATmega2560 je na obrázku č. 19.

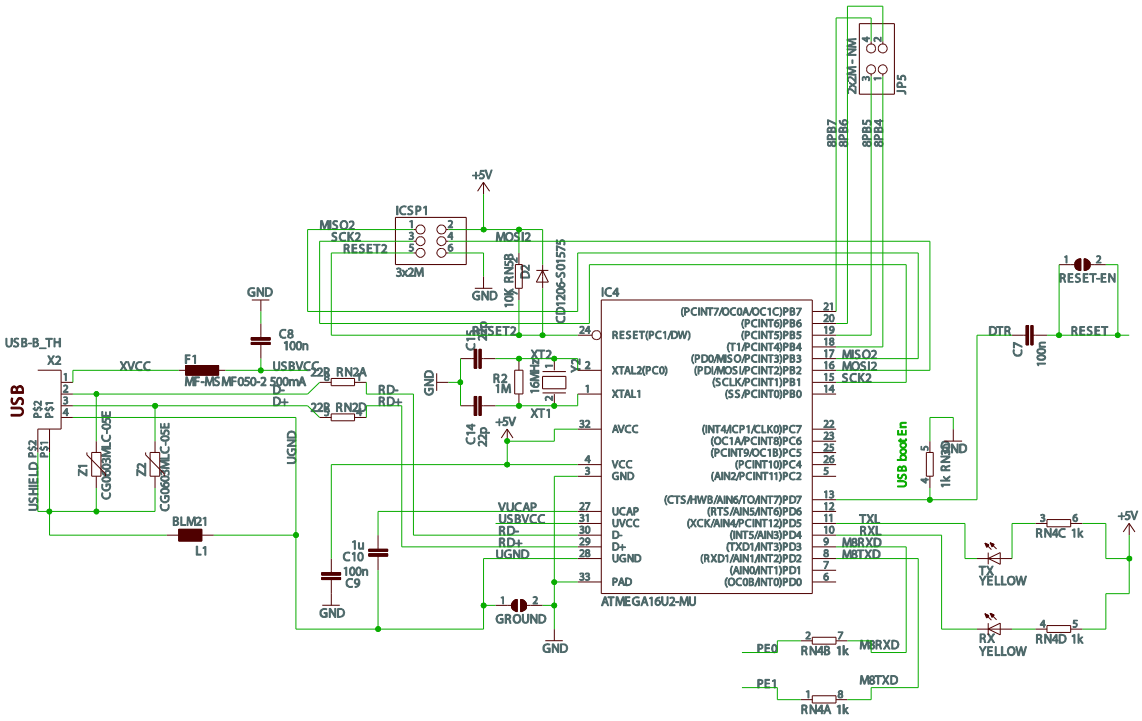


Obr. 19 – Arduino Mega 2560 – propojení mikročipu s ostatními komponenty [2]

Datová komunikace

ATmega2560 obsahuje čtyři hardwarové UART porty pro sériovou komunikaci TTL (5 V). Integrovaný čip ATmega16U2 (ATmega 8U2 v desce verze 1 a verze 2) slouží ke komunikaci pomocí USB portu ve formě virtuálního COM portu. Software Arduino IDE obsahuje monitor sériové linky, který umožňuje zasílat a přijímat jednoduchá textová data. Na desce je osazena dvojice LED (Rx a Tx), které indikují komunikaci pomocí USB sběrnice. [2]

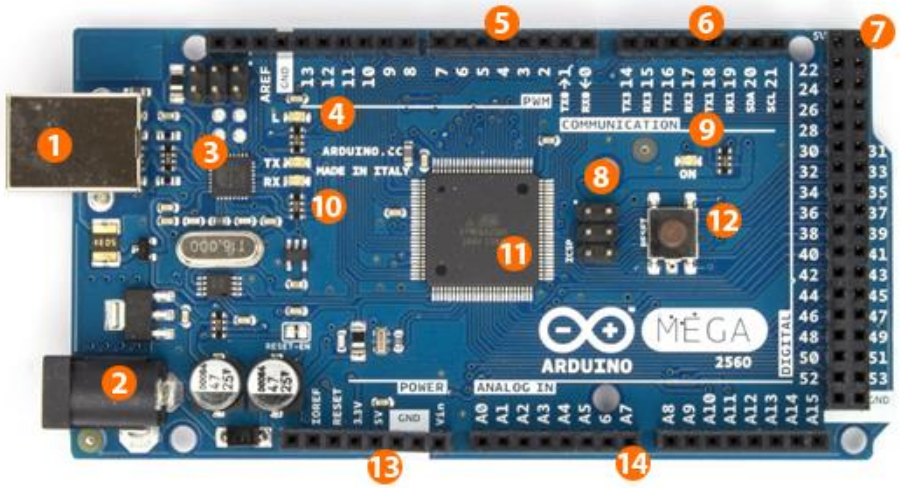
Zapojení čipu ATmega16U2 pro komunikaci přes datovou sběrnici USB je na následujícím obrázku č. 20.



Obr. 20 – Arduino Mega 2560 - zapojení USB [2]

Osazení desky Arduino ATmega2560

Osazení desky Arduino s popisem jednotlivých součástí je na následujícím obrázku.



Obr. 21 – Arduino Mega 2560 - horní pohled [2]

Popis Arduino Mega 2560:

1. USB port
2. Napájecí konektor
3. Převodník ATmega16U2 RS232 – USB
4. LED dioda propojená s pinem 13
5. Digitální vstupy/výstupy s podporou PWM
6. UART porty
7. Digitální vstupy/výstupy
8. ICSP port
9. LED pro signalizaci napájení
10. Dvojice LED pro signalizaci RxTx komunikace USB
11. Mikročip ATmega2560
12. Tlačítko RESET
13. Napájení komponent
14. Analogové vstupy

5 Zhotovení a otestování prototypu

5.1 Základní koncept

Analyzátor sériové komunikace je koncipován jako zařízení, které má dva RS232 porty pro připojení jako mezičlánek mezi dvě další zařízení. Tímto jsou obsazeny dvě sériové linky UART1 a UART2 na desce Arduino Mega 2560 a USB konektor je použit jako volitelné připojení třetí sériové linky k PC sloužící k monitoringu a logování sériové komunikace mezi jednotlivými zařízeními. Datová komunikace se bude ukládat na paměťové médium tvořené slotem SD karet připojeného k desce Arduino. Pro monitoring pomocí PC bude naprogramována VB.NET aplikace „Analyzátor sériové linky“, která umožňuje logovat probíhající komunikaci na sériové lince i pomocí PC, viz kapitola 7.

Analyzátor bude zahrnovat následující funkční prvky:
Tlačítko, které bude měnit rychlost komunikace. Zvolená rychlost bude tvořit 5 žlutých LED diod. Každé stisknutí v cyklu bude měnit rychlost komunikace.

Tlačítko, které bude zapínat logování komunikace. Zapnutí bude signalizováno zelenou LED diodou. Při zapnutí se vytvoří nový soubor na paměťové kartě. Po vypnutí se soubor uzavře.

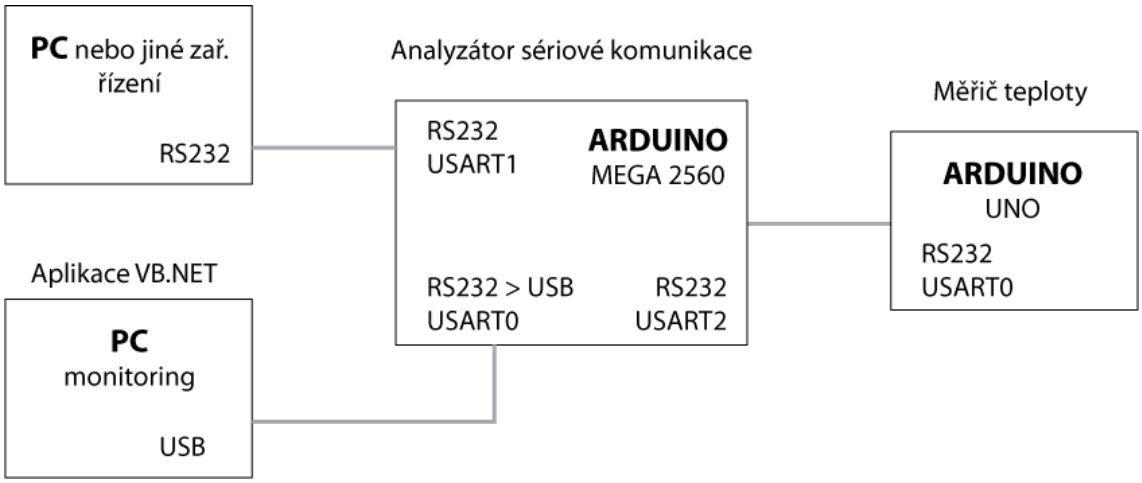
Tlačítko pro výběr portu sériové komunikace. Možností bude vybrat USB port nebo RS232. Pokud bude vybraný USB port, nebude možné monitorovat komunikaci pomocí VB.NET aplikace. Vybraný port bude identifikován dvojicí LED diod. Při použití RS232 portu bude vstup USB na desce sloužit jako monitorovací sériová linka, pomocí které bude možné sledovat logování analyzátoru a na vyžádání (přes VB.NET aplikaci příkazem/tlačítkem – <MONITOR>ON</MONITOR>) se komunikace ze zdrojového i cílového zařízení bude přenášet do terminálu (včetně stavových hlášek). Pokud bude přes příkaz/tlačítko zasláno vypnutí monitoringu <MONITOR>OFF</MONITOR> budou se v aplikaci zobrazovat pouze stavové hlášky (indikace, že se provádí logování, jméno souboru, změnu rychlosti RS232 apod.)

Červená LED dioda pro signalizaci nefunkční paměťové karty. Bude identifikovat problém s inicializací karty a dále překročení maximálního počtu souborů na kartě. Pokud bude LED identifikovat chybový stav, tak nebude možné zapnout logování komunikace.

Jeden RS232 konektor pro koncové zařízení.

Jeden RS232 konektor pro vstupní signál (PC nebo jiné zařízení).

Blokové schéma použití analyzátoru na obrázku č. 22:

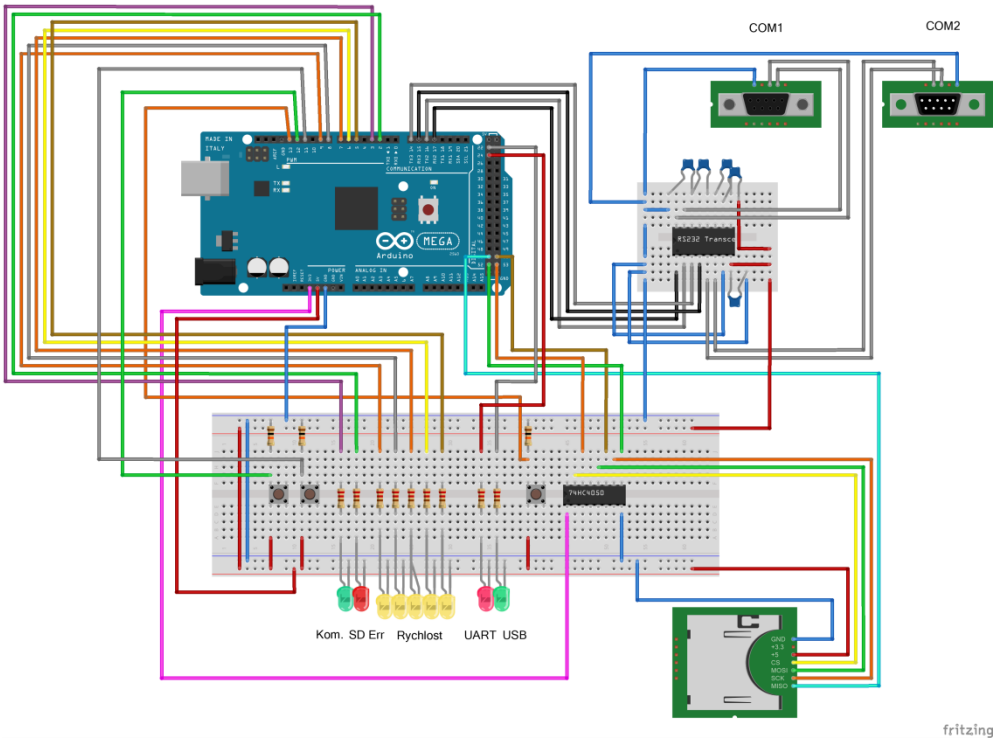


Obr. 22 – Blokové schéma použití analyzátoru dat [zdroj: autor]

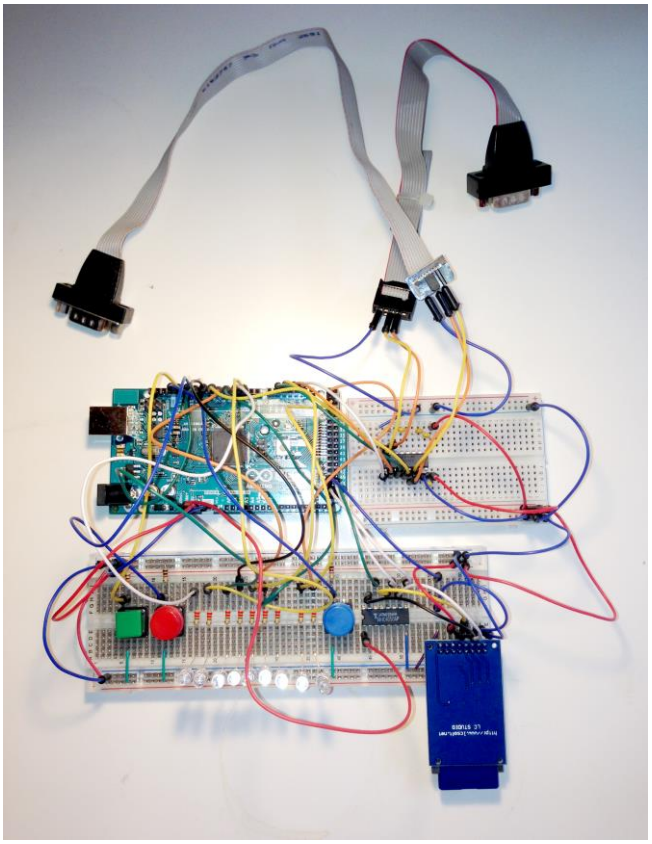
5.2 Sestavení prototypu na nepájivém poli

Pro potřeby experimentování s prototypem analyzátoru je vhodné začít sestavováním jednotlivých modulů a součástek na nepájivém poli. Pro základní návrh byl použit oblíbený open-source software Fritzing. Toto prostředí umožňuje snadno a rychle kombinovat jednotlivé moduly, součástky a jejich propojení na nepájivém poli. Software obsahuje většinu originálních desek i jejich klonů včetně rozšiřujících modulů a dalších elektronických součástek. Na internetu je podpora ve formě silné komunity, která se zabývá vývojem samotné aplikace a dalších rozšiřujících knihoven. Díky komunitě je dostupná podrobná dokumentace. [9]

Po postupném experimentování a ladění programu vzniklo výsledné schéma, které je na obrázku č. 23.



Obr. 23 – Celkový pohled na schéma v programu Fritzing [zdroj: autor]



Obr. 24 – Fotografie prototypu na nepájivém poli [zdroj: autor]

5.3 Programování desky v prostředí Arduino IDE

Pro programování desky Arduino Mega 2560 bylo použito prostředí Arduino IDE, které je dostupné zdarma přímo na webových stránkách výrobce. Po jednoduché instalaci programu, připojení přes USB a nastavení typu desky v aplikaci bylo započato se samotným programováním.

Pro přehlednost jednotlivých částí kódu bylo použito formátování textu, které odpovídá formátování v editoru Arduino IDE.

Vzhledem k použití modulu SD karty bylo potřeba využít direktiv, které obsahují samostatné knihovny SPI a SD.

Pro připojení knihoven slouží v jazyce IDE příkaz `#include`.

```
#include <SPI.h>    // Pouzijeme knihovnu SPI
#include <SD.h>     // Pouzijeme knihovnu SD
```

Většina objektů (entit) musí být deklarována a definována dříve, než jsou použity. K těmto objektům patří i konstanty, proměnné a funkce. Deklarace a definice konstant a proměnných pro označení vstupů a výstupů je následující.

```
const int chipSelect = 53;             // Output modulu SD
int komunikace;           // Promenna pro zapnuti-vypnuti komunikace
int komunikace_tlacitko;  // Deklarace stavu tlacitka pro komunikaci
int komunikace_tlacitko_old;    // Deklarace pro ulozeni predchoziho stavu
                                tlacitka komunikace
int tlacitko_rychlost = 12;     // Definice input pinu tlacitka rychlosti
int tlacitko_prenos = 11;       // Definice input pinu tlacitka prenosu
int led_sd_error = 2;           // Definice output pinu LED stavu SD
int led_komunikace = 3;         // Definice output pinu LED stavu komunikace
int sd_error = 0;               // Definice chyboveho stavu SD
int cteni;                // Deklarace promenne pro vyber rychlosti
int tlacitko_input = 13;  // Definice input pinu tlacitka vyberu vstupu
int led_input_usb = 22;   // Definice output pinu LED stavu pro vstup USB
int led_input_uart1 = 24; // Definice output pinu LED stavu pro vstup COM1
int cteni_input;    // Deklarace stavu pro vyber portu pro vstup
int input;   // Deklarace vstupu USB-COM1
int zmena_input;    // Deklarace zmeny vstupu USB-COM1
int usbmonitor=0; // Definice stavu monitoringu na USB
long rychlosti[5] = {9600,19200,38400,57600,115200};
// Definice pole rychlosti komunikace
int vybranarychlost=0; //  Definice promenne na pocatecni hodnotu
int zmenarychlosti=0;
// Definice promenne zmeny rychlosti na pocatecni hodnotu
```

```
int led1=5; // LED 9600
int led2=6; // LED 19200
int led3=7; // LED 38400
int led4=8; // LED 57600
int led5=9; // LED 115200
```

Pro použití modul SD karty bylo potřeba deklarovat proměnné pro práci se soubory na kartě.

```
File logfile;      // Deklarace souboru pro logovani komunikace na SD kartu
// Nastaveni promennych pro pouziti knihovny SD
Sd2Card card;
SdVolume volume;
SdFile root;
// Promenne pro preposilani komunikacnich dat
String inputString = "";
String inputString1 = "";
String inputString2 = "";
// Promenne pro stav prenosu dat
boolean stringComplete = false;
boolean stringComplete1 = false;
boolean stringComplete2 = false;
```

Dále bylo potřeba definovat uživatelské funkce, které jsou volány v samostatné části kódu.

Funkce `zapniled(zl)`

Definice této funkce slouží k periodické změně rozsvícení jedné z pětice žlutých LED, které signalizují vybranou rychlost.

```
void zapniled(int zl) {          // Funkce pro ovladani LED
for (int i=5;i<10;i++){          // Cyklus od 5 do 9 (piny 5-9)
      digitalWrite(i,0);         // Vypnuti LED
      }
zl=zl+5;                  // Definice promenne pro rozsviceni LED
digitalWrite(zl,1);       // Rozsviceni prislusne LED
}
```

Funkce `zapniled_input(zli)`

Je určena pro ovládání LED signalizace vstupu. Periodicky po stisknutí tlačítka `tlacitko_input` pro výběr vstupu přepíná mezi `led_input_usb` a `led_input_uart1`.

```
void zapniled_input(int zli) {  // Funkce pro ovladani LED pro vyber vstupu
if (zli == 0) {                 // Pokud je promenna 0
      digitalWrite(led_input_usb,1);         // Zapni USB LED
      digitalWrite(led_input_uart1,0);       // Vypni UART1 LED
      } else {
      digitalWrite(led_input_usb,0);         // Vypni USB LED
```

```
        digitalWrite(led_input_uart1,1);        // Zapni UART1 LED
        }
}
```

Funkce `SDcard_init()`
Inicializace SD karty.

```
void SDcard_init(){       // Funkce pro inicializaci SD karty
sd_error = 0;
Serial.print("\nInicializace SD karty...");
        pinMode(chipSelect, OUTPUT);     // Nastaveni output pinu (53, OUTPUT);
        if (!card.init(SPI_HALF_SPEED, chipSelect)) {
                Serial.println("Inicializace neprobehla. Je karta vlozena?");
        // Vypis chyboveho hlaseni v pripade, kdy inicializace neprobehla
                sd_error = 1;
        } else {
                Serial.println("Inicializace probehla OK");
                // Inicializace karty je poradku
```

Test typu karty a výstup informací na sériový port pro logování. Použitý Arduino SD modul společnosti LC STUDIO podporuje SD karty verze 1 a verze 2. Pro verzi 2 se rozlišuje, zda se jedná o kartu SDHC či nikoliv. Modul nepodporuje karty SDXC.

Následující řídící struktura vypíše do logu typ použité karty.

```
        if (sd_error == 0) {
        // Typ karty pres RS232
                Serial.print("\nTyp karty: "); // Vystup na seriovou linku
                switch(card.type()) {

                // Tisk typu karty
                case SD_CARD_TYPE_SD1:
                        Serial.println("SD1");    // SD karta je verze 1
                        break;
                case SD_CARD_TYPE_SD2:
                        Serial.println("SD2");    // SD karta je verze 2
                        break;
                case SD_CARD_TYPE_SDHC:
                        Serial.println("SDHC");   // SD karta je verze 2 SDHC
                        break;
                default:
                        Serial.println("Neznama");
        }
```

Další řídící strukturou je kontrola na typ diskového oddílu. Arduino SD modul podporuje pouze oddíl FAT16 nebo FAT32.

```
if (!volume.init(card)) {       // Funkce pro kontrolu partition
// Test jestli je partition FAT16 nebo FAT32
```

```
        Serial.println("Nelze nalezt FAT16/FAT32 partition.\nPouzili jste
        SDFormatter");      // Vypis chybového hlaseni na port
        sd_error = 1;
}
```

Výpis informací o typu FAT a velikosti použité SD karty v modulu.

```
uint32_t volumesize; // Typ FAT a velikost
Serial.print("\nTyp svazku je FAT");   // Vypis informaci na port
Serial.println(volume.fatType(), DEC);
Serial.println();
volumesize = volume.blocksPerCluster();
// Clustery jsou skupiny bloku
volumesize *= volume.clusterCount();
// Bloky vynasobime poctem clusteru
volumesize *= 512;
// A velikosti bloku, ktera je 512
Serial.print("Kapacita (bytes): ");
Serial.println(volumesize);      // Vysledna velikost v bytech
Serial.print("Kapacita (Kbytes): ");
volumesize /= 1024;
Serial.println(volumesize);      // Vysledna velikost v Kbytech
Serial.print("Kapacita (Mbytes): ");
volumesize /= 1024;
Serial.println(volumesize);      // Vysledna velikost v Mbytech
```

Knihovna pro obsluhu SD karet umožňuje pracovat pouze se soubory, které mají maximálně 8 znaků, a přípona musí mít znaky tři. V následujícím kódu je řešen výpis souborů do logu a kontrola na maximální počet souborů, který je 100. V případě, že je na SD kartě více souborů než je limit, tak se rozsvítí chybová LED a nelze spustit sledování komunikace na analyzátoru.

```
Serial.println("\nSoubory na karte (jmeno, datum a velikost [bytes] ): ");
root.openRoot(volume);
// Seznam vsech souboru
root.ls(LS_R | LS_DATE | LS_SIZE);
}
if (!SD.begin(53)) {      // Kontrola inicializace
        Serial.println("Inicializace neprovedena!");
        sd_error = 1;
        }
else {                    // Kontrola inicializace
        Serial.println("Inicializace provedeno.");
        }
}
// Resim zaplneni karty
if (SD.exists("LOG99.CSV")) {
sd_error = 1;
}
// Pokud je sd_error=1 zapnu error LED, jinak ji vypnu
if (sd_error == 1) {
        digitalWrite(led_sd_error,1);
```

```
} else {
      digitalWrite(led_sd_error,0);
      }
}
```

Následuje nastavení parametrů před spuštěním hlavní smyčky.

```
void setup() {     // Kod nastaveni, ktery se pousti pouze jednou
// Inicializace I/O
pinMode(tlacitko_rychlost, INPUT);     // Inicializace tlacitka rychlosti
pinMode(tlacitko_prenos, INPUT);       // Inicializace tlacitka prenosu
pinMode(tlacitko_input, INPUT);        // Inicializace tlacitka vstupu
pinMode(led_sd_error, OUTPUT);         // Inicializace LED chyby SD
pinMode(led_komunikace, OUTPUT);       // Inicializace LED komunikace
pinMode(led_input_usb, OUTPUT);        // Inicializace LED USB portu
pinMode(led_input_uart1, OUTPUT);      // Inicializace LED UART1 portu
// Inicializace rychlosti
for (int i=5;i<10;i++){
      pinMode(i, OUTPUT);       // Nastaveni LED na pinech 5-9 jako output
      }
zapniled(vybranarychlost);      // Rozsviceni prislusne LED rychlosti
Serial.begin(rychlosti[vybranarychlost]);    // Nastaveni rychlosti USB portu
Serial1.begin(rychlosti[vybranarychlost]);   // Nastaveni rychlosti UART1
Serial2.begin(rychlosti[vybranarychlost]);   // Nastaveni rychlosti UART2
inputString.reserve(200);       // Rezervace 200 bytu pro retezec USB
inputString1.reserve(200);      // Rezervace 200 bytu pro retezec UART1
inputString2.reserve(200);      // Rezervace 200 bytu pro retezec UART2
// Inicializace vstupu
input=0;
digitalWrite(led_input_usb,1);
digitalWrite(led_input_uart1,0);
komunikace=0; // Vypnu stav sberu dat
komunikace_tlacitko_old =0;
SDcard_init();
}
```

Funkce serialEvent()

Tato rutina je volána pokaždé, když se na komunikačním portu objeví data. Následně je využívána pro příkaz `Serial.Read()`. Pro Arduino Mega 2560, které má více UART portů musí být definována tolikrát, kolik portů je použito. V našem případě se jedná o tři porty UART:

Definice pro UART0

```
void serialEvent(){
      while (Serial.available()) {
      // Novy byte
            char inChar = (char)Serial.read();
            inputString += inChar;
            // Koncim znakem pro novou radku nastavuji obsah retezce na true
            if (inChar == '\n') {
```

```
            stringComplete = true;
            }
      }
}
```

Definice pro UART1

```
void serialEvent1(){
      while (Serial1.available()) {
      // Novy byte
            char inChar1 = (char)Serial1.read();
            inputString1 += inChar1;
            // Koncim znakem pro novou radku
            if (inChar1 == '\n') {
            stringComplete1 = true;
            }
      }
}
```

Definice pro UART2

```
void serialEvent2(){
      while (Serial2.available()) {
      // Novy byte
            char inChar2 = (char)Serial2.read();
            inputString2 += inChar2;
            // Koncim znakem pro novou radku
            if (inChar2 == '\n') {
            stringComplete2 = true;
            }
      }
}
```

Následuje hlavní kód programu spouštěný opakovaně ve smyčce.

Na začátku smyčky kontrolujeme stav tlačítka pro spuštění komunikace. Zároveň probíhá kontrola, zda je správně iniciovaný modul karty SD a karta je v pořádku. Pokud je na modulu nebo SD kartě nějaký problém, tak se iniciace SD provede znovu.

```
void loop() {
// Hlavni smycka, vlozeny kod se opakuje
komunikace_tlacitko = digitalRead(tlacitko_prenos);
if (komunikace_tlacitko_old != komunikace_tlacitko) {
// Reaguji na vzestupnou hranu
      if (komunikace_tlacitko==1) {
      // Pokud je na karte error, znovu dojde k inicializaci
            if ((sd_error == 1)) {
            SDcard_init();
            komunikace_tlacitko_old=komunikace_tlacitko;
            } else {
                  if (komunikace==0)  {
                  // Zapni komunikaci
```

```
                komunikace=1;
                digitalWrite(led_komunikace,1);
                komunikace_tlacitko_old=komunikace_tlacitko;
```

V případě, že je vše v pořádku a komunikace je spuštěna na SD kartě se otevře nový soubor a na USB se posílají monitorovací informace.

```
            Serial.println("KOMUNIKACE ZAPNUTA!");
            Serial.print("RYCHLOST :");
            Serial.println(rychlosti[vybranarychlost]);
            // Musim otevrit novy soubor
            char filename[] = "LOG00.CSV";  //Nazev souboru na SD karte
            for (uint8_t i = 0; i < 100; i++) {
            // Zmena indexu nazvu soboru
                  filename[3] = i/10 + '0';
                  filename[4] = i%10 + '0';
                  if (! SD.exists(filename)) {
                  // Otevri novy pouze pokud soubor jiz neexistuje
                        logfile = SD.open(filename, FILE_WRITE);
                  if (logfile) {
                        Serial.print("Otevren novy soubor pro log : ");
                        Serial.println(filename);
                        inputString = "//ZACATEK LOGU";
                        logfile.println(inputString);
                        inputString = "";
                        logfile.print("RYCHLOST : ");
                        logfile.println(rychlosti[vybranarychlost]);
                  }
                  else {
                  // Soubor nelze otevrit
                        Serial.print("Nelze otevrit soubor pro log :");
                        Serial.println(filename);
                  }
                  break;  // Opusteni smycky
                  }
            }
      delay(100);
}
```

Pokud je komunikace tlačítkem vypnuta, do USB se posílají monitorovací informace o ukončení, uzavírá se soubor na SD kartě a probíhá kontrola na maximální počet souborů na kartě, limitem je sto souborů.

```
      else if (komunikace==1) {
      // Vypni komunikaci
            komunikace=0;
            digitalWrite(led_komunikace,0);
            komunikace_tlacitko_old=komunikace_tlacitko;
            Serial.println("KOMUNIKACE VYPNUTA!");
            inputString = "//KONEC LOGU";
            logfile.println(inputString);
            logfile.close();
            inputString = "";
```

```
                delay(100);
                // Kontrola max. poctu souboru
                if (SD.exists("LOG99.CSV")) {
                        sd_error = 1;
                }
        }
}
}
        // Sestupna hrana
        else {
                komunikace_tlacitko_old=komunikace_tlacitko;
        }
}
```

Rychlost komunikace se nastavuje deklarovaným tlačítkem `tlacitko_rychlost`. V závislosti na výběru se rozsvítí příslušná LED dioda, která signalizuje vybranou rychlost. Výběr rychlosti tlačítkem periodicky cykluje v rozmezí definovaných hodnot pole `rychlosti`. Na monitorovací výstup se zapisuje informace o vybrané rychlosti sériové komunikace.

```
// Tlacitko na zmenu rychlosti RS232

if (komunikace==0) {
        cteni = digitalRead(tlacitko_rychlost);
        if (cteni==0) {
                zmenarychlosti=1;
        }
        if ((cteni==1) && (zmenarychlosti==1)) {
                if (vybranarychlost<4) {
                        vybranarychlost++;
                }
        else {
                vybranarychlost=0;
        }
        zmenarychlosti=0;
        zapniled(vybranarychlost);
        Serial.print("ZMENA RYCHLOSTI NA :");
        Serial.println(rychlosti[vybranarychlost]);
        Serial.end();
        Serial1.end();
        Serial2.end();
        Serial.begin(rychlosti[vybranarychlost]);
        Serial1.begin(rychlosti[vybranarychlost]);
        Serial2.begin(rychlosti[vybranarychlost]);
        }
}
```

Výběr vstupu je závislý na volbě tlačítkem `tlacitko_input`. Vstupem může být USB port nebo port COM1 na analyzátoru. Změna je opět odeslána na monitorovací port a dochází k rozsvícení jedné ze dvou LED diod, které signalizují vybraný způsob komunikace.

```
// Tlacitko na zmenu vstupu
if (komunikace==0) {
	cteni_input = digitalRead(tlacitko_input);
	if (cteni_input==0) {
		zmena_input=1;
	}
	if ((cteni_input==1) && (zmena_input==1)) {
		Serial.print("ZMENA VSTUPU NA :");
		if (input==0) {
			input=1;
			digitalWrite(led_input_usb,0);
			digitalWrite(led_input_uart1,1);
			Serial.println("COM1 INPUT");
		}
	else {
		input=0;
		digitalWrite(led_input_usb,1);
		digitalWrite(led_input_uart1,0);
		Serial.println("USB");
	}
	zmena_input=0;
	zapniled_input(input);
	}
}
```

Pokud je USB port vybrán jako vstupní, tj. port určený pro primární komunikaci mezi zařízeními, tak jej nelze použít jako monitorovací. V případě, kdy je vybrán pro komunikaci port COM1, tak lze pomocí USB portu monitorovat stav analyzátoru. V závislosti na vybraném vstupu potom probíhá čtení a zápis komunikace mezi vstupním portem (USB nebo COM1) a výstupním portem (COM2). Data komunikace se ukládají na SD kartu a pokud je zapnuto monitorování tak zároveň na USB port.

```
if (stringComplete) {
// Monitoring komunikace
	if ((inputString.substring(0,9) == "<MONITOR>") &&
	(inputString.substring(10,20) == "</MONITOR>")) {
	// Prijat pozadavek zmenu monitorovani
		if (input==0){
		// USB je vstup nelze monitorovat
			Serial.println("USB je nastaveno jako vstup, nelze
			monitorovat!");
		}
		else {
		// Resim prichozi hodnotu 0,1
			Serial.print("USB neni nastaveno jako vstup, lze
			monitorovat : ");
			inputString.replace("<MONITOR>", "");
			inputString.replace("</MONITOR>\n", "");
			usbmonitor = inputString.toInt();
			Serial.print(usbmonitor);
			if (usbmonitor == 0) {
```

```
                        Serial.print("MONITORING VYPNUT");
                    }
                    else {
                        Serial.print("MONITORING ZAPNUT");
                    }
                }
        } else if (komunikace==1){
        // Neprisel pozadavek na monitoring
        // Pokud je USB nastaveno jako vstup, zapisu na SD a poslu na COM2
        if (input == 0) {
                logfile.print("USB -> COM2;");
                logfile.println(inputString);
                Serial2.print(inputString);
                Serial3.print(inputString);
        }
}
        // Vymazu promennou
        inputString = "";
        stringComplete = false;
}
if (stringComplete1) {
        if (komunikace==1){
        // Zapisu na SD a podle nastaveneho vstupu preposlu
                if (input == 1) {
                        logfile.print("COM1 -> COM2;");
                        logfile.println(inputString2);
                        Serial2.print(inputString1);
                if (usbmonitor==1) {
                // Pokud je zapnuto monitorovani, posilam na USB
                        Serial.print("COM1 -> COM2 :");
                        Serial.println(inputString1);
                }
        }
}
Serial.print(inputString1);
// Vymazu promennou
inputString1 = "";
stringComplete1= false;
}
if (stringComplete2) {
        if (komunikace==1){
        // Zapisu na SD a podle nastaveneho vstupu preposlu
                if (input == 0) {
                        logfile.print("COM2 -> USB;");
                        logfile.println(inputString2);
                        Serial.print("COM2 -> USB : ");
                        Serial.println(inputString2);
                }
                else {
                        logfile.print("COM2 -> COM1;");
                        logfile.println(inputString2);
                        Serial1.print(inputString2);
                if (usbmonitor==1) {
                // Pokud je zapnuto monitorovani, posilam na USB
                        Serial.print("MONITOR > COM2 -> COM1 : ");
                        Serial.print(inputString2);
                }
        }
```

```
}
// Vymazu promennou
inputString2 = "";
stringComplete2= false;
      }
}
```

5.4 Testovací zařízení

Pro potřeby testování analyzátoru bylo zapotřebí vytvořit jednoduché zařízení, které obousměrně komunikuje pomocí sériové linky. Hlavní důvod byl, aby pro testování prototypu analyzátoru byl k dispozici známý komunikační protokol a bylo možné ověřit, že analyzovaná a ukládaná data odpovídají reálné komunikaci mezi dvěma zařízeními.

Pro tyto účely bylo navrhnuto a sestaveno zařízení pro měření teploty a byl naprogramován jednoduchý komunikační protokol pro monitoring a řízení prototypu.

Základní koncept

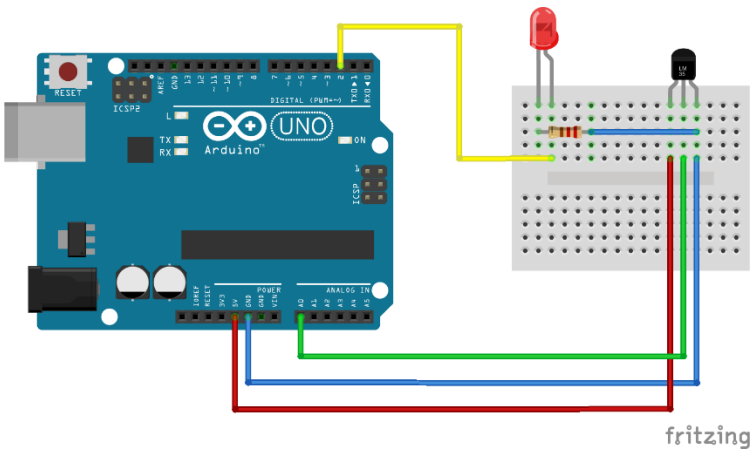
Navrhnout a zrealizovat zařízení pro měření teploty. Snímač teploty bude připojen k dalšímu zařízení (PC nebo analyzátor) pomocí sériové linky (USB/RS232). Zařízení bude odesílat na pokyn informaci o aktuální teplotě snímače. Pro řízení bude naprogramován příkaz, který nastaví prahovou teplotu snímače. Po překročení prahové teploty se na snímači rozsvítí LED a pomocí sériové linky se odešle informace o překročení teploty s naměřenou a prahovou hodnotou. Při poklesu teploty pod nastavený práh, se LED zhasne a opět se odešle informace na sériovou linku.

Nastavení komunikačního protokolu

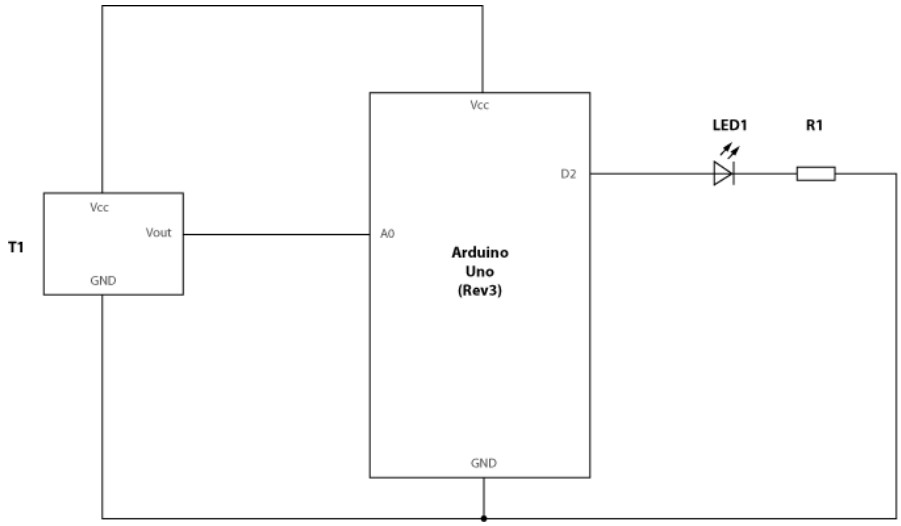
Příkaz	Popis	Vzor komunikace
GT	Zjisti teplotu (GetTemperature)	<T>27.12</T>
GRT	Zjisti prahovou hodnotu (TrasholdTemperature)	<TT>25.00</TT>
<STT>30.00</STT>	Nastav prahovou hodnotu na 30 °C (SetTrasholdTemperature)	<TT>30.00</TT>
	Při překročení prahové hodnoty	Teplota presahla prah, zapinam LED! 30.03>30.00
	Při poklesu prahové hodnoty	Teplota klesla pod prah, vypinam LED! 29.55<30.00

Schéma zařízení pro měření teploty na nepájivém poli

V prostředí aplikace Fritzing bylo vytvořeno schéma zařízení s použitím nepájivého pole.



Obr. č. 25 – Schéma zapojení teploměru [zdroj: autor]



Obr. č. 26 – Elektrické schéma zapojení teploměru [zdroj: autor]

Seznam použitých součástek:

Název	Popis	Počet
Arduino UNO	Deska AVR ATmega328	1
T1	Teplotní senzor Texas Instruments LM35	1
R1	Rezistor 220Ω	1
LED1	LED dioda	1

Programování desky měření teploty

Pro programování desky Arduino Mega UNO bylo použito prostředí IDE. Nejprve bylo potřeba definovat a deklarovat proměnné a konstanty pro teplotní čidlo, LED diodu a datovou komunikaci sériové linky.

```
int tempPin = A0;                // Analogovy vstup cidla LM35
int ledtemppin = 13;             // LED vystup, pouzita LED na desce
int val;                         // Promenna analogoveho vstupu
float temp_treshold = 200;       // Promenna pro nastaveni prahu teploty
String inputString = "";         // Retezec vstupu dat na RRS 232
boolean stringComplete = false; // Promenna pro detekci vstupnich dat
boolean led_on = false;          // Promenna stavu LED diody
float teplota;                   // Promenna pro teplotu ve °C
```

V inicializační části kódu je umístěno nastavení rychlosti sériové linky rezervace paměti pro UART řetězce a definice výstupního zařízení v podobě LED.

```
void setup() {
// Kod nastaveni, ktery se pousti pouze jednou
Serial.begin(9600);       // Nastaveni rychlosti UART portu
inputString.reserve(200); // Rezervace 200 bytu pro retezec UART portu
pinMode(ledtemppin, OUTPUT);    // Nastaveni pinu LED jako vystup
}
```

Funkce `posli_teplotu()`
Pokud je tato funkce zavolána v průběhu programu, tak vrací aktuální teplotu čidla.

```
void posli_teplotu () {
// Posli aktualni teplotu na RS232
val = analogRead(tempPin);                    // Cti analogovy vstup
teplota = ((4.96 * val) / 1024) * 100; // Vypocitej teplotu
Serial.print("<T>");                          // Odesli teplotu na UART
Serial.print(teplota);
Serial.println("</T>");
}
```

Funkce `posli prah()`
Funkce vrací aktuálně nastavenou prahovou hodnotu teploty.

```
void posli_prah () {
// Posli prahovou hodnotu pres RS232
Serial.print("<TT>");     // Odesli aktualni prahovou teplotu na UART
Serial.print(temp_treshold);
Serial.println("</TT>");
}
```

Funkce serialEvent()
Tato rutina je volána pokaždé, když se na komunikačním portu objeví data. Následně je využívána pro příkaz `Serial.Read()`.

```
void serialEvent() {
      while (Serial.available()) {    //    Novy byte
            char inChar = (char)Serial.read();
            inputString += inChar;
            // Koncim znakem pro novou radku nastavuji obsah retezce na true
            if (inChar == '\n') {
                  stringComplete = true;
            }
      }
}
```

Následuje hlavní smyčka programu, která se opakuje. Na začátku smyčky je ze senzoru, který je připojen k analogovému vstupu snímáno napětí, které se přepočítává na velikost teploty zobrazenou v °C. Pokud teplota přesáhne nebo

naopak poklesne pod stanovený práh, tak se na sériovou linku zasílá informace o této skutečnosti zároveň s prahovou a okamžitou teplotou.

```
void loop() {
// Hlavni smycka, vlozeny kod se opakuje
// Resim treshold
delay (100);
      val = analogRead(tempPin);      // Cti analogovy vstup
      float teplota = ((4.96 * val) / 1024) * 100; // Vypocitej teplotu

      if (teplota > temp_treshold) {
      // Pokud teplota presahne prah vypis info na UART
            if (led_on == false) {    // Pouze v pripade ze jiz nesviti LED
            Serial.println("Teplota presahla prah, zapinam LED! ");
            // Vypis na UART
            Serial.print(teplota);
            Serial.print(">");
            Serial.println(temp_treshold);
            digitalWrite(ledtemppin,1);     // Zapni LED
            led_on = true;                  // Nastaveni stavu aktivace
            delay(5000);
            // Prodleva pro stabilitu prechodu teploty pres prah
            }
      }

      if (teplota < temp_treshold)  {
      // Pokud teplota klesne pod prah vypis info na UART
            if (led_on == true) {     // Pouze v pripade ze jiz sviti LED
            Serial.println("Teplota klesla pod prah, vypinam LED! ");
            // Vypis na UART
            Serial.print(teplota);
            Serial.print("<");
            Serial.println(temp_treshold);
            digitalWrite(ledtemppin,0);     // Vypni LED
            led_on = false;                 // Nastaveni stavu deaktivace
            delay(5000); // Prodleva pro stabilitu prechodu teploty pod prah
            }
      }
```

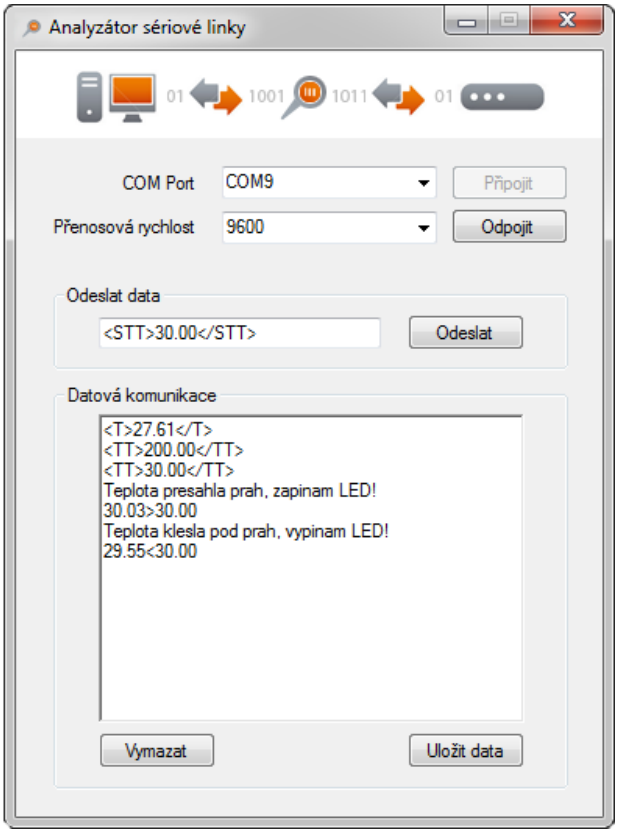
Dále se kontroluje, jestli na vstupu sériové linky nevznikl požadavek na odeslání teploty či nastavení prahové hodnoty.

```
// Sledovani vstupu UART
if (stringComplete) {     // Pokud prisla data na UART
      if (inputString.substring(0,2) == "GT") {
      // Prijat pozadavek na cteni teploty
            posli_teplotu();    // Posli info o aktualni teplote zpet
      }
      if (inputString.substring(0,3) == "GRT") {
      // Prijat pozadavek na aktualni nastaveni prahu teploty
            posli_prah();       // Posli info o aktualnim nastaveni prahu
      }
      if (inputString.substring(0,6) == "<STT>") {
      // Prijat pozadavek na nastaveni prahove teploty
            inputString.replace("<STT1>", "");
```

```
            inputString.replace("</STT1>\n", "");
            temp_treshold = inputString.toFloat();
            Serial.print("<TT1>");
            Serial.print(temp_treshold);
            Serial.println("</TT1>");
        }
        // Smazani vstupniho retezce
        inputString = "";
        stringComplete = false;
        }
}
```

Testování zařízení pro měření teploty

Pro účely testování a ladění zařízení pro měření teploty a pro ověření přenosového protokolu byla použita aplikace „Analyzátor sériové linky“ viz kapitola 6.



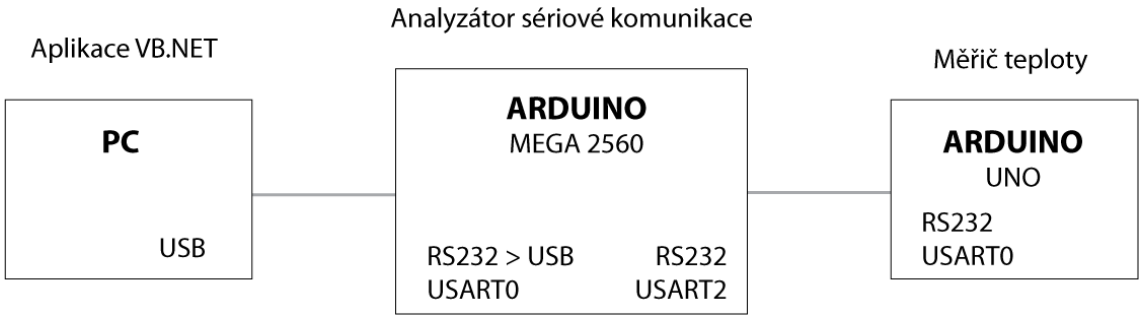
Obr. č. 27 – Testování zařízení pro měření teploty [zdroj: autor]

5.5 Testování prototypu analyzátoru

Prototyp analyzátoru byl testován pomocí aplikace popsané v kapitole 6. Sestava byla testována na platformě operačního systému Windows 7 s instalovanou podporou .NET Framework 4.5 .

Pro testovací účely byl použit analyzátor sériové komunikace a zařízení pro měření teploty s definovaným komunikačním protokolem.

Blokové schéma zapojení testovací soustavy je na obrázku č. 28.



Obr. 28 – Blokové schéma zapojení testovací soustavy [zdroj: autor]

Připojení k zařízení

Po připojení počítače k zařízení pro měření teploty sériovou linkou, v tomto případě USB (virtuální sériový port), a nastavení přenosové rychlosti se aplikace připojí k analyzátoru sériové komunikace. Pokud by byl problém v připojení, např. sériová linka by již byla použita jinou aplikací, zobrazí se chybové hlášení o nedostupnosti vybraného portu. Pokud je připojení k sériovému portu v pořádku, zobrazí se informační okno o úspěšném připojení a lze pokračovat v monitorování a konfiguraci koncového zařízení.

Od okamžiku připojení aplikace monitoruje vybraný sériový port a data přenáší do okna „Datová komunikace“. Pro ovládání zařízení slouží textové pole „Odeslat data“. Do tohoto pole lze zadávat příkazy, které se odesílají sériovou linkou na připojené zařízení.

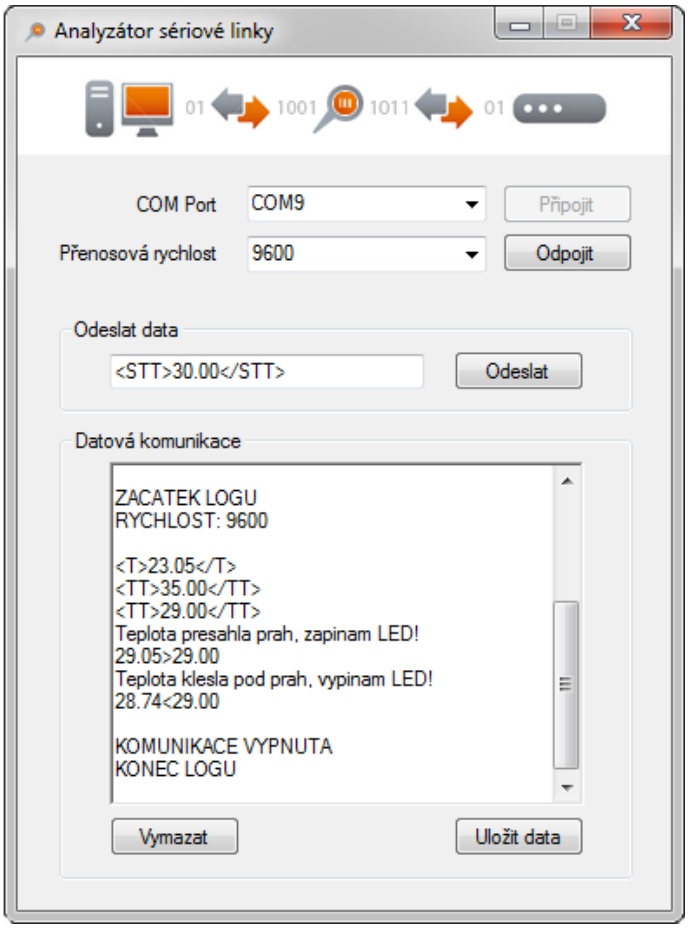
Po zapnutí a inicializaci analyzátoru byl nejprve odeslán požadavek na zjištění aktuální teploty příkazem GT. Testovací zařízení pro měření teploty vrátilo hodnotu <T>22.03</T>. Naměřená hodnota odpovídala naměřené hodnotě samostatným digitálním teploměrem umístěným v blízkosti senzoru teploty testovacího zařízení s minimální odchylkou. Dále byl nastaven práh teploty pomocí příkazu <STT>23.40</STT>. Odpověď zobrazená a uložená na paměťové kartě byla <TT>23.40</TT>. Postupným zvyšováním teploty senzoru zahříváním byla překročena prahová hodnota a zařízení poslalo na sériovou linku informaci „Teplota presahla prah, zapinam LED! 28.59>23.40“. Po snížení teploty na původní hodnotu bylo zalogováno „Teplota klesla pod prah, vypinam LED! 22.91<23.40“.

Výpis protokolu v aplikaci a na SD kartě:

```
Inicializace SD karty…
Inicializace probehla OK
Typ karty: SDHC
Typ svazku je FAT
Kapacita (bytes):
Kapacita (Kbytes):
Kapacita (Mbytes):
Soubory na karte (jmeno, datum a velikost [bytes]):
Inicializace provedeno

KOMUNIKACE ZAPNUTA
RYCHLOST : 9600

Otevren novy soubor pro log : LOG01.CSV

ZACATEK LOGU
RYCHLOST: 9600

GT
<T>22.03</T>
<STT>23.40</STT>
<TT>23.40</TT>
Teplota presahla prah, zapinam LED!
28.59>23.40
```

```
Teplota klesla pod prah, vypinam LED!
22.91<23.40

KOMUNIKACE VYPNUTA
KONEC LOGU
```

Uložené okno aplikace s výsledným protokolem je na obrázku č. 29:



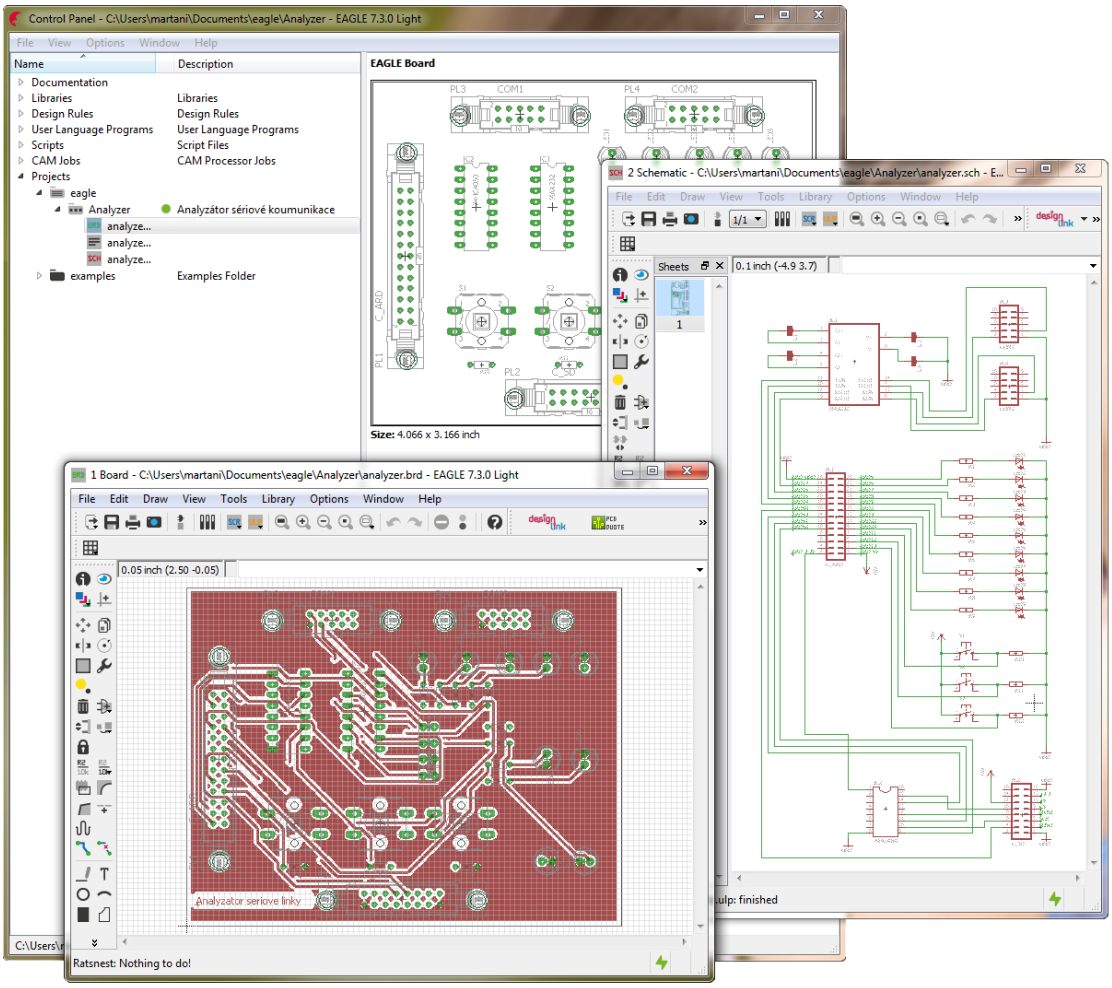
Obr. 29 – Výsledný protokol analýzy dat v aplikaci [zdroj: autor]

Pro testování byl použit definovaný protokol a bylo ověřeno, že data odesílaná a přijímaná na sériové lince odpovídají datům, která byla interpretována ve VB.NET aplikaci a zároveň byla uložena na paměťové kartě analyzátoru.

5.6 Vytvoření desky plošného spoje analyzátoru

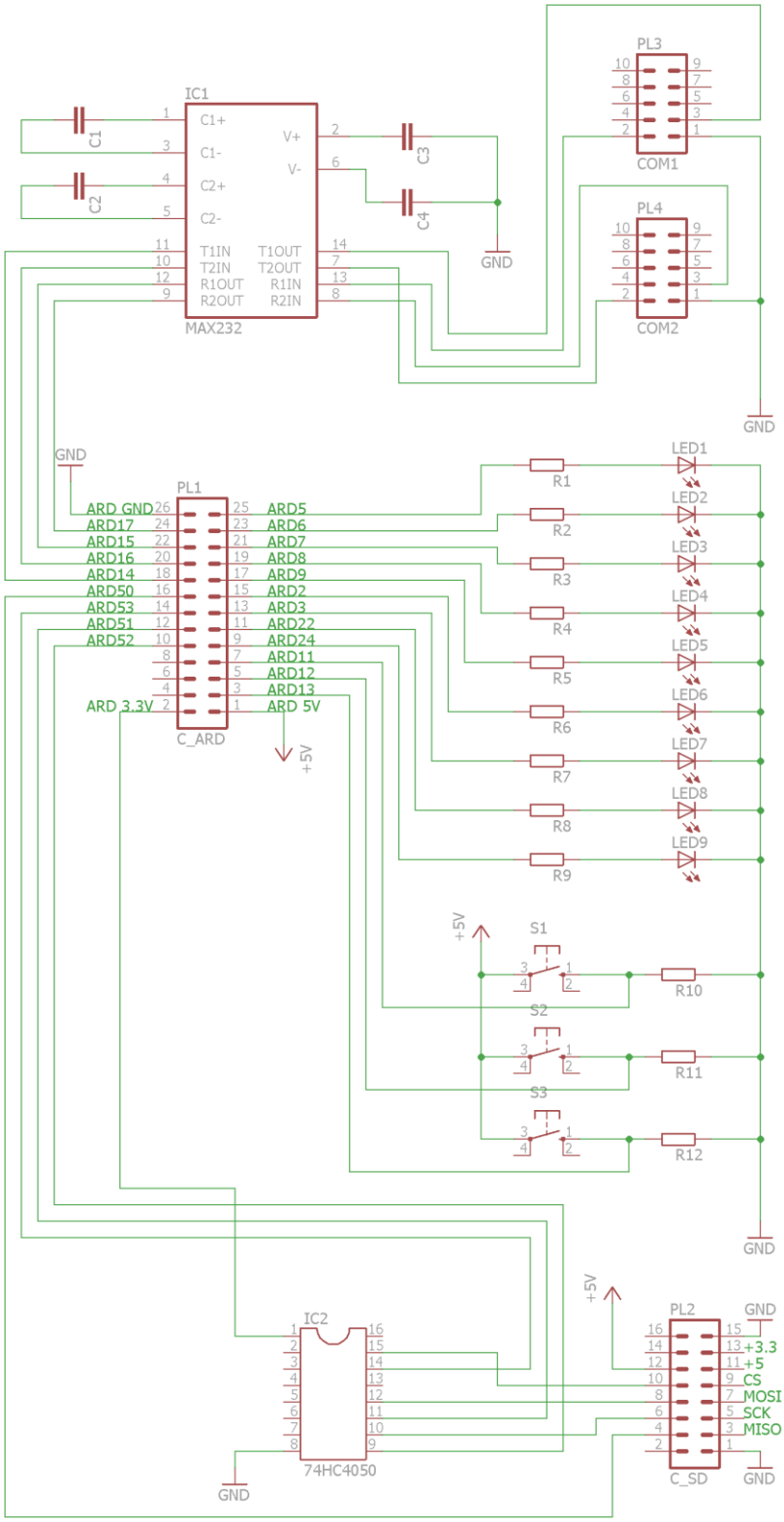
Pro vytvoření desky byla použita aplikace EAGLE ve verzi 7.3.0. Tento software pro návrh elektronických obvodů a plošných spojů je mezi vývojáři hodně oblíbený, jeho distribuce je volně šiřitelná pro nekomerční použití. Omezení je pouze v maximální velikosti plošného spoje na 80x100 mm. Pomocí tohoto nástroje je možné jednotlivé součástky vyhledávat dle parametrů a online nakupovat dokonce v českém jazyce.

Silná komunita uživatelů stojí i za vytvářením dalších knihoven součástek a modulů pro aplikaci Eagle. [12]

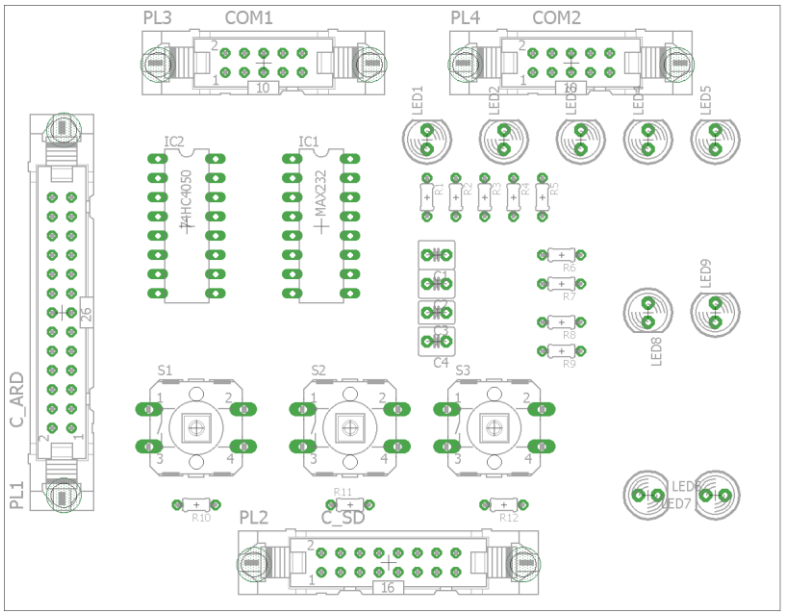


Obr. 30 – Prostředí aplikace Eagle [zdroj: autor]

Výstupní schémata z vývojového software Eagle jsou na následujících obrázcích.



Obr. 31 – Schéma obvodu analyzátoru [zdroj: autor]

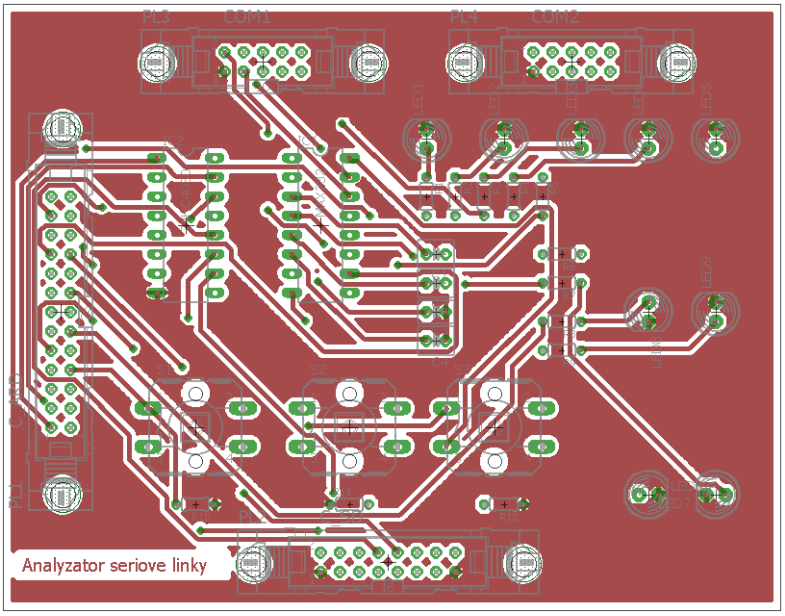


Obr. č. 32 – Rozložení součástek na plošném spoji [zdroj: autor]

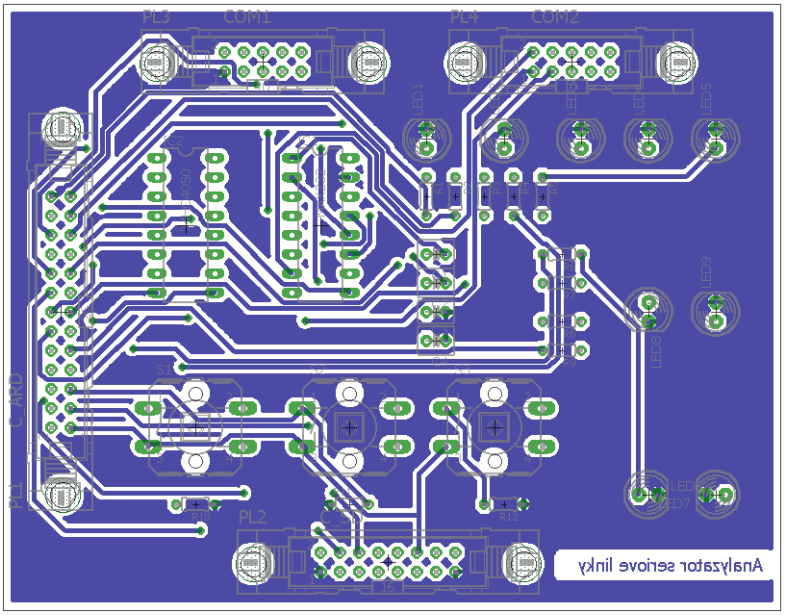
Seznam použitých součástek:

Název	Popis	Počet
IC1	Integrovaný obvod 74HC4050	1 ks
IC2	Integrovaný obvod MAX232	1 ks
C1 – C4	Kondenzátor 10 µF	4 ks
R1 – R9	Rezistor 220 Ω	9 ks
R10 – R12	Rezistor 10 kΩ	3 ks
LED1 – LED5	LED dioda žlutá	5 ks
LED6 – LED8	LED dioda zelená	3 ks
LED9	LED dioda červená	1 ks
S1 – S3	Mikrospínač	3 ks
PL1	26 pinový konektor na desku Arduino	1 ks
PL2	16 pinový konektor na SD modul	1 ks
PL3 – PL4	10 pinový konektor na RS232	2 ks

Návrh oboustranného plošného spoje vygenerovaný a upravený v programu Eagle je na obr. č. 33 a č. 34.



Obr. 33 – Plošný spoj horní pohled [zdroj: autor]



Obr. 34 –Plošný spoj spodní pohled [zdroj: autor]

6 Vytvoření .NET aplikace pro načítání a vyhodnocení dat

6.1 Základní koncept

Aplikace bude logovat a analyzovat komunikaci pomocí samostatného portu USB. Tento způsob sledování je zapotřebí zejména v situacích, kdy je vhodné, aby výsledky byly vidět v reálném čase.

Dále bude umožňovat řízení připojeného analyzovaného zařízení, tak že data bude možné pomocí sériové linky odeslat. Výsledný protokol příchozí i odchozí komunikace bude možné uložit do počítače k archivaci a případné další analýze.

Jako programovací jazyk byl vybrán Visual Basic .NET, který pracuje na rozhraní .NET. Toto rozhraní nekompiluje výsledný program přímo do strojového kódu, ale do určitého mezikódu, které nazýváme CIL (Common Intermediate Language). Mezikód se uloží do EXE souboru a v této podobě se přenáší na koncové počítače uživatelů. V okamžiku, kdy se program spustí, tak se teprve kompiluje do strojového kódu. Při kompilaci nedochází k plnému přeložení programu, ale pouze toho co je z programu potřeba, před prvním spuštěním. Této funkci se říká JIT (Just In Time).

Výhodou je to, že při kompilaci dochází k optimalizaci programu pro konkrétní procesor, který kód spouští. Tímto je zaručena kompatibilita programu s procesorem a program je zároveň optimalizován tak aby běžel co nejrychleji.

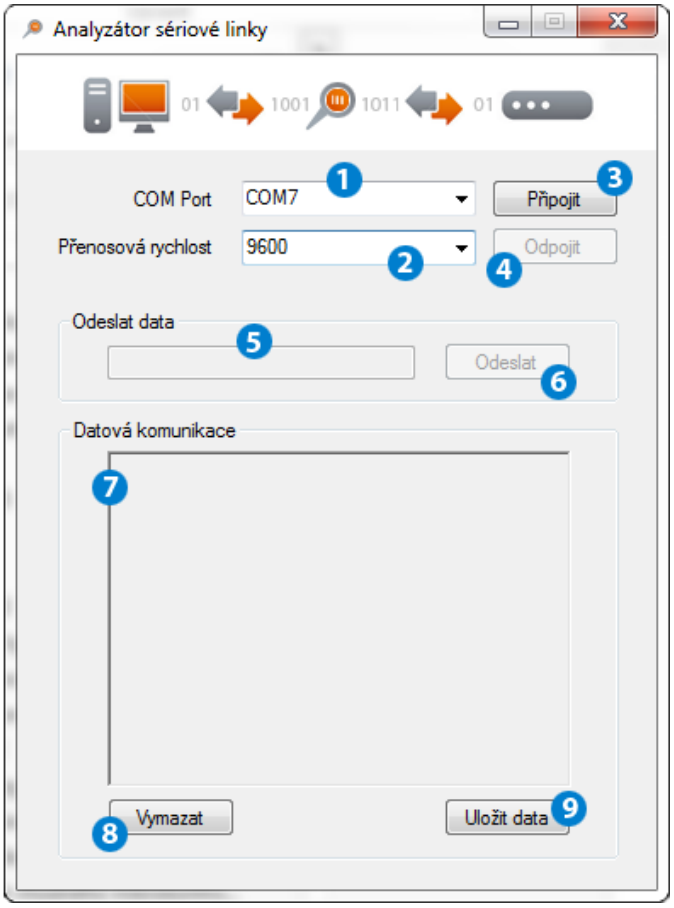
Mezi hlavní důvody výběru jazyka VB.NET byla zejména jednoduchost, srozumitelnost a dostupnost vývojového prostředí. Na rozdíl od většiny skriptovacích jazyků, které nemají podporu grafického prostředí Windows VB.NET disponuje plnohodnotnou podporou a knihovnami pro tvorbu grafických aplikací s formuláři, databázovým rozhraním, systémovými funkcemi a dokonce rozhraním pro multimédia a hry. Díky podpoře .NET Framework je Visual Basic plnohodnotný nástroj pro vývoj hobby i profesionálních aplikací.

Pro programování bylo použito prostředí Visual Studio Community 2013, které je vyvíjeno společností Microsoft a je k dispozici zdarma. Pro získání licence je zapotřebí registrace Microsoft účtu. Bez licence je aplikace v 30ti denní zkušební lhůtě bez omezení funkcionality. [20]

6.2 Grafický design a ovládání aplikace

Projekt ve Visual Studio Community se skládá ze dvou základních částí. Z grafického formuláře, který definuje ovládací prvky a zdrojového kódu.

Před samotným programováním aplikace byl v aplikaci Microsoft Visual Studio nejprve vytvořen grafický návrh formuláře s rozmístěním ovládacích prvků. Na obrázku č. 35 je vidět výsledný vzhled aplikace vytvořený ve vývojovém prostředí Visual Studio Community 2013.



Obr. 35 – Vzhled vytvořené aplikace [zdroj: autor]

Popis ovládání:

1. Výběr komunikačního sériového portu
2. Výběr rychlost datové komunikace
3. Tlačítko pro připojení k vybranému portu
4. Tlačítko pro odpojení od sériové linky. Zároveň maže komunikaci a odeslané příkazy.
5. Pole pro data k odeslání na sériový port
6. Tlačítko pro odeslání dat
7. Pole pro analýzu dat na komunikačním portu
8. Tlačítko pro vymazání analýzy komunikace
9. Tlačítko pro uložení komunikace do textového souboru

6.3 Vývoj aplikace

Po vytvoření formuláře a umístění ovládacích prvků aplikace se k jednotlivým položkám tvoří zápis zdrojového kódu v jazyce VB.NET.

Vzhledem k svému zaměření – Analyzátor sériové linky, bylo zapotřebí nejprve definovat použitý COM port a rychlost připojení. K tomuto účelu slouží seznamy položek `ComboPort` a `ComboRate`. Seznam použitelných portů je v aplikaci generován automaticky. Zobrazí se pouze dostupné COM porty v počítači, ze kterého aplikace byla spuštěna.

```
Dim myPort As Array  'Pole pro uložení dostupných COM portů

Private Sub MainForm_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
       'Automatická detekce dostupných COM portů
       myPort = IO.Ports.SerialPort.GetPortNames()
       'Načti dostupné COM porty

       ComboRate.Items.Add(9600)
       'Nastavení rychlostí COM portu
       ComboRate.Items.Add(19200)
       ComboRate.Items.Add(38400)
       ComboRate.Items.Add(57600)
       ComboRate.Items.Add(115200)

       For i = 0 To UBound(myPort)
       ComboPort.Items.Add(myPort(i))
       Next
```

```
        ComboPort.Text = ComboPort.Items.Item(0)
        'Nastavení prvního záznamu dostupného COM portu do ComboBoxu
        ComboRate.Text = ComboRate.Items.Item(0)
        'Nastavení prvního záznamu rychlosti do ComboBoxu
        ButtonDisconnect.Enabled = False
        'Po spuštění aplikace je tlačítko pro odpojení neaktivní
        TextBoxSend.Enabled = False
        'Po spuštění aplikace je textové pole pro odesílání dat neaktivní
        ButtonSend.Enabled = False
        'Po spuštění aplikace je tlačítko pro odesílání dat neaktivní
        RichTextBoxReceive.Enabled = False

        'Po spuštění aplikace je textové pole pro sledování dat neaktivní
End Sub
```

Ovládání aplikace je zajištěno dvojicí tlačítek **Připojit** a **Odpojit**. Byla definována dvě nová tlačítka `ButtonConnect` a `ButtonDisconect`. Zároveň je zajištěno, že v jeden okamžik je aktivní pouze jedno z tlačítek, což slouží jako vizuální kontrola stavu připojení počítače k analyzátoru Arduino.

```
Private Sub ButtonConnect_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonConnect.Click
        SerialPort1.PortName = ComboPort.Text
        'Přiřazení sériového portu k vybranému COM portu
        SerialPort1.BaudRate = ComboRate.Text
        'Nastavení rychlosti vybraného sériového portu

        SerialPort1.Parity = IO.Ports.Parity.None
        'Nastavení bitů
        SerialPort1.StopBits = IO.Ports.StopBits.One
        SerialPort1.DataBits = 8
        'Nastavení bitů

        Try
        SerialPort1.Open()
        'Otevření portu
        Catch ex As Exception
                MsgBox("Nelze otevřít COM port", vbCritical)    'Ošetření vyjímky
                Return
        End Try


        MsgBox("Připojeno k portu " + SerialPort1.PortName, vbInformation)

        ButtonConnect.Enabled = False
        'Po připojení zneaktivnit tlačítko připojení
        ButtonDisconnect.Enabled = True
        'Aktivovat tlačítko odpojení
        TextBoxSend.Enabled = True
        'Aktivovat textové pole pro odesílání dat
        ButtonSend.Enabled = True
        'Aktivovat tlačítko pro odesílání dat
        RichTextBoxReceive.Enabled = True
        'Aktivovat textové pole pro sledování dat
End Sub
```

```
Private Sub ButtonDisconnect_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonDisconnect.Click
	SerialPort1.Close()
	'Zavření COM portu

	ButtonConnect.Enabled = True
	'Aktivace tlačítka Připojení
	ButtonDisconnect.Enabled = False
	'Deaktivace tlačítka Odpojení
	TextBoxSend.Enabled = False
	'Deaktivace textového pole odeslání dat
	RichTextBoxReceive.Clear()
	'Vymazání textového pole
	TextBoxSend.Clear()
	'Vymazání textového pole
End Sub
```

Aplikace umožňuje řídit připojené zařízení odesláním Tx příkazu na zařízení. Tento příkaz slouží zejména k ladění komunikace, kdy můžeme simulovat požadavky na zařízení bez použití původní originální aplikace.

```
Private Sub ButtonSend_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonSend.Click
	SerialPort1.Write(TextBoxSend.Text & vbLf)
	'Odeslání ASCII textu na COM port a LF příkaz (Enter)
End Sub
```

Aby bylo možné data odeslat po stisknutí tlačítka ENTER, bylo zapotřebí přidat proceduru `TextBoxSend_KeyDown`.

```
Private Sub TextBoxSend_KeyDown(ByVal sender As Object, ByVal e As KeyEventArgs)
Handles TextBoxSend.KeyDown
	If e.KeyCode = Keys.Enter Then    'Stisknutí klávesy ENTER
	SerialPort1.Write(TextBoxSend.Text & vbLf)
	'Odeslání ASCII textu na COM port a LF příkaz (Enter)
		e.SuppressKeyPress = True  'Vypnutí zvukového tónu
	End If
End Sub
```

Pro čtení dat ze sériové linky zařízení jsou deklarovány následující procedury `SerialPort1_Datareceived` a procedura `ReceivedText`.

```
Private Sub SerialPort1_DataReceived(ByVal sender As Object, ByVal e As
System.IO.Ports.SerialDataReceivedEventArgs) Handles SerialPort1.DataReceived
	ReceivedText(SerialPort1.ReadExisting())
	'Automatické čtení dat ze sériové linky
End Sub

Private Sub ReceivedText(ByVal [text] As String)
'porovnává ID vláken
	If Me.RichTextBoxReceive.InvokeRequired Then
		Dim x As New SetTextCallback(AddressOf ReceivedText)
```

```
            Me.Invoke(x, New Object() {(text)})
        Else
            Me.RichTextBoxReceive.Text &= [text]
        End If
End Sub
```

Na konci programu je umístěné ošetření chybových stavů, tak, aby po připojení k zařízení nebylo možné měnit sériový port a rychlost sériové komunikace.

```
Private Sub ComboPort_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboPort.SelectedIndexChanged
        If SerialPort1.IsOpen = False Then
            SerialPort1.PortName = ComboPort.Text
            'Ošetření změny portu na otevřené spojení
        Else
            MsgBox("Před změnou portu odpojte COM port", vbCritical)
        End If
End Sub

Private Sub cmbBaud_SelectedIndexChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles ComboRate.SelectedIndexChanged
        If SerialPort1.IsOpen = False Then
            SerialPort1.BaudRate = ComboRate.Text
            'Ošetření změny rychlosti port na otevřené spojení
        Else
            MsgBox("Před změnou rychlosti odpojte COM port", vbCritical)
        End If
End Sub
```

Pro vymazání dat z textového okna pro sledování komunikace slouží tlačítko `ButtonClear`. Tlačítkem `ButtonSave` je možné uložit výsledný protokol s daty do textového souboru.

```
Private Sub ButtonClear_Click(sender As Object, e As EventArgs) Handles
ButtonClear.Click
            RichTextBoxReceive.Clear() 'Vymaž textové okno komunikace
        End Sub

Private Sub ButtonSave_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonSave.Click
        Dim nsavetxtoutput As New SaveFileDialog()
        'Otevři dialogové okno pro uložení souboru
        nsavetxtoutput.Filter = "txt files (*.txt)|*.txt"      'Typ souboru
        nsavetxtoutput.FilterIndex = 2
        nsavetxtoutput.RestoreDirectory = True
        If nsavetxtoutput.ShowDialog() = DialogResult.OK Then
            IO.File.WriteAllText(nsavetxtoutput.FileName,
            RichTextBoxReceive.Text)    'Zápis souboru
        End If
End Sub
```

7 Zhodnocení dosažených výsledků

Cílem této diplomové práce bylo navrhnout hardwarové a softwarové řešení pro sledování komunikace dvou zařízení na sériové lince. Vznikl prototyp zařízení, který je postaven na relativně mladé platformě Arduino, používající mikroprocesory AVR společnosti Atmel a open-source vývojové prostředí. Pro realizaci prototypu zařízení bylo zapotřebí využít celkem tři porty pro sériovou komunikaci. Převážně na základě těchto parametrů byla vybrána deska Arduino Mega 2560, která tento předpoklad splňuje. Společně s deskou je použit i modul pro ukládání dat na paměťové karty SD a další elektronické komponenty pro propojení a komunikaci s ostatními zařízeními. Základní návrh a experimentování se zapojením prototypu vzniklo pomocí open-source aplikace Fritzing. Po ověření funkcionality byla použita aplikace Eagle pro návrh elektronického schéma a plošného spoje. Aplikace pro monitorování a logování komunikace s připojeným analyzátorem byla naprogramována v jazyce VB.NET a prostředí Visual Studio Community.

V průběhu projektu bylo potřeba překonat mnohé problémy, které bylo zapotřebí vyřešit. Jejich vyřešení bylo dosaženo samostatným experimentováním, hledáním řešení v odborné literatuře nebo na internetových fórech a v neposlední řadě konzultacemi s vedoucím diplomové práce.

Během práce na prototypu si autor ověřil a prohloubil znalosti z oblasti elektroniky a programování.

Závěr

Arduino je elektronická open-source platforma, integrující hardware s mikroprocesory Atmel a software pro jednoduché použití. Právě vzhledem k dostupnosti vývojových nástrojů, desek a modulů Arduino a dalších elektronických součástí má vysoký potenciál nejen v edukačních procesech, ale i hobby využití. Některé z mnoha možností byly využity při vývoji prototypu analyzátoru sériové komunikace. Budoucnost dalšího rozvoje této platformy autor spatřuje například v hobby využití instalací inteligentních domů (ovládání, regulace, zabezpečení, monitoring apod.), ale i v aktuálně hodně skloňované perspektivě „internetu věcí“ – IoT. Pro nadšené vývojáře je po osvojení dovedností na platformě Arduino možnost dalšího rozvoje ve složitějších projektech s přechodem k průmyslovým platformám.

Seznam literatury

[1] ATMEL. *Webová prezentace společnosti* [online]. 2015 [cit. 2015-06-26]. Dostupné z: http://www.atmel.com

[2] ADRUINO. *Webová prezentace projektu* [online]. 2015 [cit. 2015-06-26]. Dostupné z: http://www.arduino.cc

[3] HRBÁČEK, Jiří. *Komunikace mikrokontroléru s okolím*. Praha: BEN, 1999. ISBN 80-86056-45-2.

[4] MATOUŠEK, David. *Práce s mikrokontroléry ATMEL AVR 2. díl.* Praha: BEN - technická literatura, 2002. ISBN 80-7300-066-0.

[5] MATOUŠEK, David. *Práce s mikrokontroléry ATMEL AVR 4. díl.* Praha: BEN - technická literatura, 2006. ISBN 80-7300-174-8.

[6] CHYSKÝ, Jan. *Vestavěné systémy I*. 2., přeprac. vyd. V Praze: České vysoké učení technické, 2010. ISBN 978-80-01-04629-6.

[7] *Wikipedia* [online]. 2015 [cit. 2015-06-26]. Dostupné z: https://cs.wikipedia.org

[8] COMPAQ, HEWLET-PACKARD, INTEL, LUCENT, MICROSOFT, NEC, PHILLIPS. *Universal Serial Bus Specification* [online]. 2000 [cit. 2015-06-26]. Dostupné z: http://www.usb.org

[9] *Fritzing: Open-source aplikace pro návrhy Arduino* [online]. 2015 [cit. 2015-06-26]. Dostupné z: http://www.fritzing.org

[10] VODA, Zbyšek. Průvodce světem ARDUINA. In: *Průvodce světem ARDUINA* [online]. 2015 [cit. 2015-06-26]. 2015. Dostupné z: http://www.arduino.cz

[11] *HW Kitchen* [online]. 2015 [cit. 2015-06-26]. Dostupné z: http://www.hwkitchen.com

[12] CADSOFT. *EAGLE* [online]. 2015 [cit. 2015-06-26]. Dostupné z: http://www.cadsoftusa.com

[13] TAJČ, Martin. *Univerzální logický analyzátor*. Brno, 2013. Diplomová práce. Vysoké učení technické v Brně.

[14] *MIDI Manufacturers Association* [online]. 2015 [cit. 2015-06-26]. Dostupné z: http://www.midi.org

[15] PALOVSKÝ, Radomír. *Informační a komunikační sítě*. Vyd. 1. Praha: Oeconomica, 2010. ISBN 978-80-245-1729-2.

[16] *Wiring: Programovací jazyk* [online]. 2015 [cit. 2015-06-26]. Dostupné z: http://wiring.org.co

[17] MCS ELECTRONICS. *BASCOM AVR* [online]. 2015 [cit. 2015-06-26]. Dostupné z: http://www.mcselec.com

[18] BUŠEK, Vojtěch. *Stavebnice a úlohy s jednočipovými počítači Atmel AVR pro základní školy*. České Budějovice, 2013. Diplomová práce. Jihočeská univerzita v Českých Budějovicích.

[19] BANZI, Massimo. *Getting started with Arduino*. USA: O'Reilly, 2008. ISBN 978-0-0594-15551-3.

[20] MICROSOFT. *Microsoft Visual Studio Community* [online]. 2015 [cit. 2015-06-26]. Dostupné z: https://www.visualstudio.com

Zkratky použité v dokumentu

ACK	Acknowledgement, signál přijmutí požadavku
CIL	Common Intermediate Language
I²C	Inter-Integrated Circuit
ICSP	In-Circuit Serial Programming
IoT	Internet of Things
open-source	Počítačový software s dostupným zdrojem
PWM	Pulse Width Modulation, Pulzní šířková modulace
Sketch	Skript vytvořený v Arduino prostředí
SPI	Serial Peripheral Interface
TTL	Transistor-transistor logic
TWI	Two-wired Interface, dvouvodičový interface
UART	Universal Asynchronous Receiver and Transmitter
USART	Universal Synchronous/Asynchronous Receiver and Transmitter
USB	Universal Serial Bus