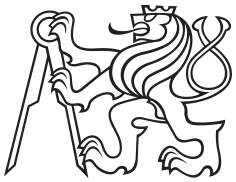
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Měřicí systém se vzdáleným přístupem

Využití terminálových aplikací

Jan Holčák

Vedoucí: doc. Ing. Jan Fischer, CSc.
Obor: Otevřená informatika
Studijní program: Informatika a počítačové vědy
Květen 2018

Poděkování

Tímto bych chtěl poděkovat doc. Ing. Janu Fischerovi, CSc. za čas věnovaný odborným konzultacím a úvod do programování mikrokontrolérů architektury ARM. Dále pak děkuji své rodině, za podporu během celé doby studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 24. května 2018

Abstrakt

Možnost propojení chytrých zařízení vybavených elektronikou, software, senzory, akčními členy a bezdrátovou komunikací vedlo k rozvoji oboru nazývaného Internet Věcí. Nový obor propojuje vývoj software pro vestavěné systémy a osobní počítače. Programátor se znalostí vývoje software pro architekturu x64 může narazit na počáteční problémy s nasazením jednoúčelové aplikace omezené parametry mikrokontrolérů.

S použitím předpřipravených aplikací umožňujících komunikaci mezi mikrokontrolérem a osobním počítačem můžeme minimalizovat problémy s oživením prototypu vyvíjeného zařízení.

Klíčová slova: mbed, Terminál, UART, ESPTerm, Mikrokontrolér, IoT

Vedoucí: doc. Ing. Jan Fischer, CSc.

Abstract

Possibility to connect devices implementing electronic circuits, sensors, actuators, and wireless technology lead to grow of new field called Internet of Things. This new field connects software development for embedded systems and personal computer applications. An experienced software developer for x64 architecture may encounter initial problems with deploying dedicated applications with limited microcontroller resources.

Using preconfigured applications to enable communication between the microcontroller and the personal computer, we can minimize prototype build-up problems.

Keywords: mbed, Terminal, UART, ESPTerm, Microcontroller, IoT

Title translation: Measurement System with Remote Access — Usage of terminal applications

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

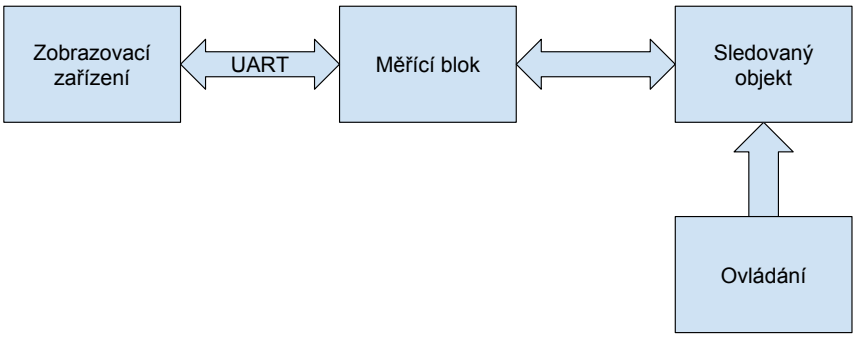
1.1 Motivace

S rozšířením mobilních zařízení zároveň rostou možnosti interakce s okolními digitálními zařízeními. Jednoduchá digitální zařízení využívající bezdrátovou komunikaci se zároveň stávají velmi dostupnými pro použití v množství nových aplikací. Zařízení nám pak umožňují měřit data na sledovaném objektu v podobě elektrických signálů nebo fyzikálních veličin. Jiná zařízení nám umožňují sledovaný objekt ovládat nebo regulovat v závislosti na získaných hodnotách.

Příkladem může být rozšíření chytrých zařízení v budovách a domácnostech. Tato zařízení často provozují specifickou činnost a jejich fungování je řízeno jednoduchým obvodem pro zajištění funkce a komunikace. Nároky na tato zařízení se díky zvyšující dostupnosti chytrých mobilních telefonů a jiných přenosných zařízení (tzv. wearables) rychle mění. Výpočetní zařízení obsažená v běžných objektech s možností propojení a komunikace prostřednictvím sítě internet jsou označována pojmem Internet of Things.

Internet of Things (dále jen IoT) je rychle se rozvíjející obor s velkým prostorem pro inovace. Tato práce se zabývá těmi nejobecnějšími příklady aplikace s možností následného rozšíření. Pozornost je věnována především základním činnostem, které jsou pro použití mikrokontrolérů typické. Většinu složitějších případů implementace IoT zařízení lze totiž rozložit na tyto jednodušší úseky jako je měření a ovládání.

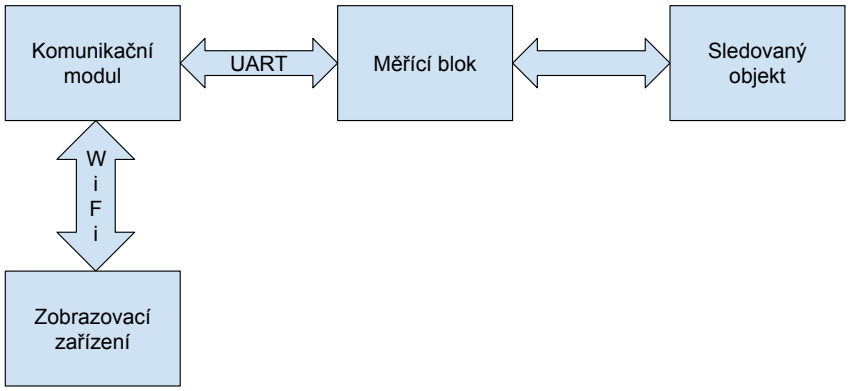
1.2 Měřicí systém s lokálním přístupem



V praxi často využíváme senzory, např. pro měření teploty, vlhkosti nebo intenzity osvětlení v prostředí sledovaného objektu. Pro získávání těchto fyzikálních veličin (tzv. DAQ) využíváme měřicího bloku. Měřicí blok se skládá z řídící jednotky a analogově digitálních převodníků. Obdržená data sledovaného objektu, jako je např. budova, stroj nebo vybavení domácnosti, dále zobrazujeme uživateli prostřednictvím lokálně připojeného zobrazovače.

Připojení zobrazujícího zařízení kabelem ale v některých případech není praktické. Z těchto důvodů se práce zabývá také návrhem a realizací měřicího později ovládacího systému se vzdáleným přístupem. Vzdálený přístup prostřednictvím bezdrátové komunikace nevyžaduje realizaci nového zařízení. Bezdrátová komunikaci umožňující vzdálený přístup lze totiž realizovat pouhým rozšířením původního systému s měřícím blokem.

1.3 Měřicí systém se vzdáleným přístupem



Použití komunikačního modulu přináší výhody nejen v pohodlí ovládání a měření. Použitím vhodného komunikačního modulu lze také značně zkrátit dobu vývoje aplikace pro řídící jednotku měřicího bloku. Vzdálený přístup a komunikační modul řeší také problémy s kompatibilitou množství zařízení. S využitím běžné bezdrátové komunikace jako je například rozhraní WiFi se otvírá cesta pro komunikaci s dříve uzavřenými platformami. Využitím možnosti předat naměřená data dále můžeme využít pokročilejší specializace navrhovaného zařízení.

Kapitola 2

Rozbor zadání

Systém bude měřit data z různých senzorů umístěných na sledovaném objektu prostřednictvím měřícího bloku. Měřicí blok systému bude využívat mikrořadiče řady STM32 a to konkrétně mikrokontroléru STM32F042. K měřícímu bloku bude možné připojit zobrazovací zařízení umožňující lokální měření. Výstup měření bude zobrazen ve formě textu pro zařízení libovolné platformy.

Dále bude systém rozšířen o vzdálený přístup prostřednictvím rozhraní WiFi za použití hotového modulu. Komunikační modul umožní zasílání naměřených dat z různých snímačů sledovaného objektu. Podobně bude možné využít vzdálený přístup pro nastavení výstupů systému. Nastavení výstupů umožní ovládání a nastavování parametrů sledovaného objektu.

Pro komunikaci bude využito modulu ESP8266 s využitím volně dostupného firmware ESPTerm. S použitím modulu bude ověřena možnost zasílání naměřených dat v lokální síti. Dále budou realizovány demonstrační aplikace pro řídící jednotku kompatibilní s firmware ESPTerm. Aplikace vyvíjené v online prostředí mbed implementují ovládání a měření.

Práce zpracuje následující kroky.

1. Zapojení obvodu a nahrávání firmvare
2. Návrh pseudografické terminálové aplikace.
3. Implementace pseudografické aplikace pro měření.
4. Implementace pseudografické aplikace pro ovládání.
5. Rozšíření o možnost bezdrátové komunikace.
6. Prozkoumání možností komunikace v síti.
7. Realizace aplikace pro nadřazený systém.

Kapitola 3

Realizace měřícího přístroje

3.1 Použitý hardware

3.1.1 Mikrokontrolér

Jednočipový počítač je integrovaný obvod, který v sobě zahrnuje zpravidla vše potřebné k tomu, aby mohl obsáhnout celou aplikaci, aniž by potřeboval další podpůrné obvody. Především jde o paměť pro uložení programu a operační paměť pro uložení aplikačních proměnných a zásobníku. Většina jednočipových počítačů také obsahuje rozsáhlou sadu podpůrných obvodů. Typicky jde o bloky pro logické a analogové vstupy/výstupy, pro komunikační linky a další aplikační logiku.

3.1.2 STM32 32-bit ARM Cortex MCUs

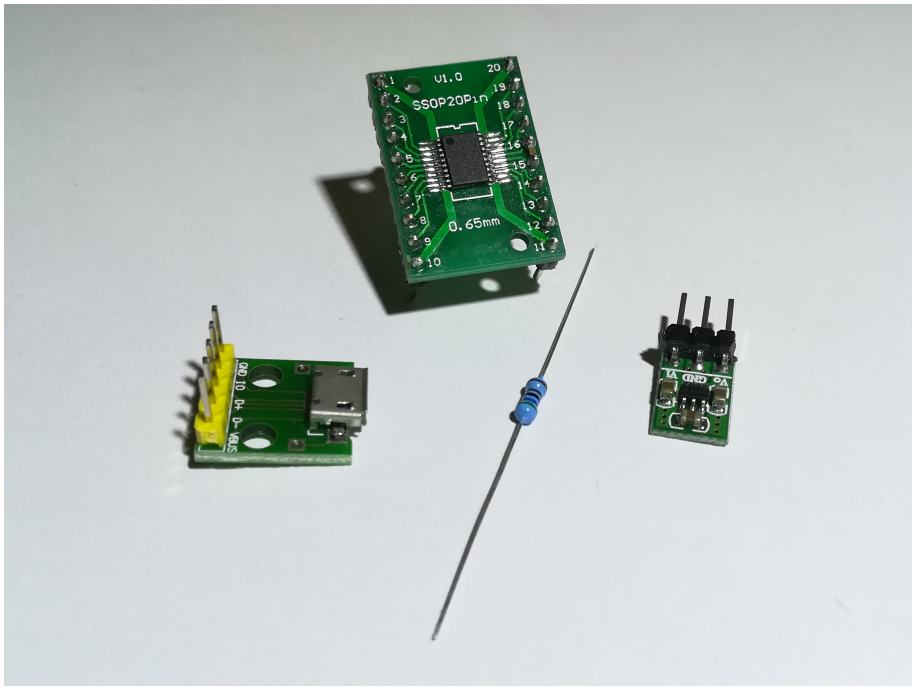
Mikrokontroléry STM32 jsou založeny na 32-bitovém jádru ARM Cortex. Firma STMicroelectronics vyrábí množství mikrokontrolérů s různým zaměřením v minimálně deseti řadách. Mikrokontrolér používaný pro realizaci této práce patří do řady F0. Skupina mikrokontrolérů patřící do této skupiny je samotným výrobcem[01] označovaná jako řada vhodná pro začátečníky nebo také jako hlavní proud.

Mikrokontrolér STM32F042

Mikrokontrolér označený jako STM32F042 použitý v této práci je vhodný pro použití v množství aplikací. Tento mikrokontrolér je zaměřen na výrobu ekonomicky dostupných zařízení s malou spotřebou elektrické energie. Vhodný je především pro použití v zařízeních využívajících USB díky hardwarové podpoře tohoto rozhraní. Díky těmto vlastnostem je použití této řady mikrokontrolérů vhodné pro stavbu prototypů.

Mikrokontrolér poskytuje 6KB paměti SRAM a 32KB paměti flash. Tato paměť umožňuje uložení velkého množství textu potřebného pro nasazení terminálových aplikací. Podpora USB umožňuje využití sériové komunikace prostřednictvím virtuálního COM Portu bez potřeby připojení USB to Serial

převodníku. Připojení rozhraním USB může být také použito pro upgrade firmwaru zařízení. Mikrokontrolér tímto minimalizuje počet součástek potřebných pro svůj běh na řád jednotek.



Obrázek 3.1: Součástky využívané mikrokontrolérem STM32F042

3.1.3 Motivace pro vývoj vestavěných systémů

Mikrokontrolér STM32F042 i modul ESP8266 používané v této práci jsou používány také v praxi. Tento fakt může být užitečný při migraci projektu z prototypu do reálného prostředí. Použitím stejného mikrokontroléru odpadá potřeba migrace kódu a následné testování. Cena jednotky v pouzdře s dvaceti vývody verze STM32F042F6P6 se pohybuje okolo 40 korun. Odběrem velkého množství mikrokontrolérů lze cenu snížit až na polovinu. Nízká cena umožňuje použití mikrokontroléru ve velkém množství experimentů.

Programátor bez předchozí zkušenosti s vývojem embedded zařízení se často snaží využít vývojových kitů obsahujících plnohodnotný počítač s běžným OS. Tato cesta ale často vede na problémy při následné optimalizaci zařízení. Hlavní nevýhodou použití víceúčelového počítače je jeho vysoká pořizovací cena. Tento fakt brání snadnému nasazení do sériové produkce. Dalším problémem může být nižší rychlost komunikace se sběrnicí z důvodu režije architektury OS. Na rozdíl od modulů plnohodnotného počítače, MCU zabírá menší prostor a vykonávaný program může běžet bez přerušení.

Moderní mikrokontroléry nabízejí programování vysokoúrovňovými jazyky. Pro programování lze využít také online vývojové prostředí s možností výběru používaného mikrokontroléru. Online vývojové prostředí zároveň obsahuje množství knihoven a ukázkových programů. Takto předem připravené pro-

středí a dostupné šablony vedou na strmou křivku učení a rychlé oživení navrženého prototypu.

Kapitola 4

Zapojení obvodu prototypu

Obvod s mikrokontrolérem STM32F042 potřebuje pro svou funkci připojení několika součástek. Obvody pro měření a ovládání realizované v této práci jsou zapojeny na kontaktním nepájivém poli. Zapojení součástek na kontaktním nepájivém poli není ideální a může přinést množství problémů. Pro bezporuchový běh by proto zapojení těchto součástek mělo splňovat několik zásad správného zapojení.

Použití digitálních obvodů s mikrokontroléry vyžaduje stabilní napájení při proměnlivém odběru proudu. Při zapojení spojů, které potečou nezanedbatelné proudy, proto musíme dávat pozor na vysoký odpor kontaktního pole. Často používaná kontaktní pole pro stavbu prototypů mohou mít některé kontakty častým požíváním opotřebené. Při zapojení na nekvalitním poli se můžeme setkat také s použitím nevhodných materiálů s vysokým odporem. Vysoký odpor, způsobený opotřebeným kontaktem nebo nekvalitním materiálem, může v případě zvýšeného protékajícího proudu vyvolat těžko odhalitelné chyby. Pro omezení těchto problémů je vhodné použít kvalitní pole a snížit počet propojů u rizikových spojů.

Elektronické součástky jsou vyráběny s různými průměry vývodů. Při výběru součástek je proto vhodné vybírat součástky s průměrem vhodným pro nepájivé kontaktní pole. Zapojení součástky s příliš velkým průměrem vodiče vývodu poškozuje pole. Součástka s příliš malým průměrem vodiče vývodu bude mít tendence z pole vypadávat a ztrácet kontakt.

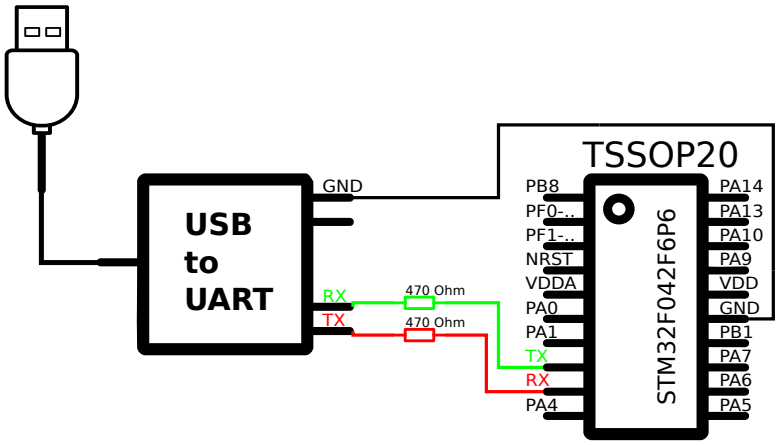
V obvodu obsahujícím mikrokontrolér připojený k USB se běžně vyskytuje několik různých hladin napětí. Na zdroji napájení v podobě USB2.0 můžeme naměřit přibližně 5V[02], kdežto na výstupech mikrokontroléru by se nemělo objevit napětí mimo rozmezí 0V až 3,3V. Kontakt takového pinu mikrokontroléru s napětím vyšším než 3.6V[03] může způsobit jeho nevratné poškození. V případech, kde může nastat tato situace je proto vhodné použít ochranné rezistory.

Ochranné rezistory chrání mikrokontrolér před překročením maximálního povoleného proudu. V datasheetu mikrokontroléru je uveden maximální dovolený proud 25mA[04]. Volba rezistoru by měla počítat s rezervou a pokud je to možné, tak vždy volíme rezistor větší hodnoty. Doporučená hodnota pro většinu případů je 470Ohm rezistor s vývody dostatečné šířky.

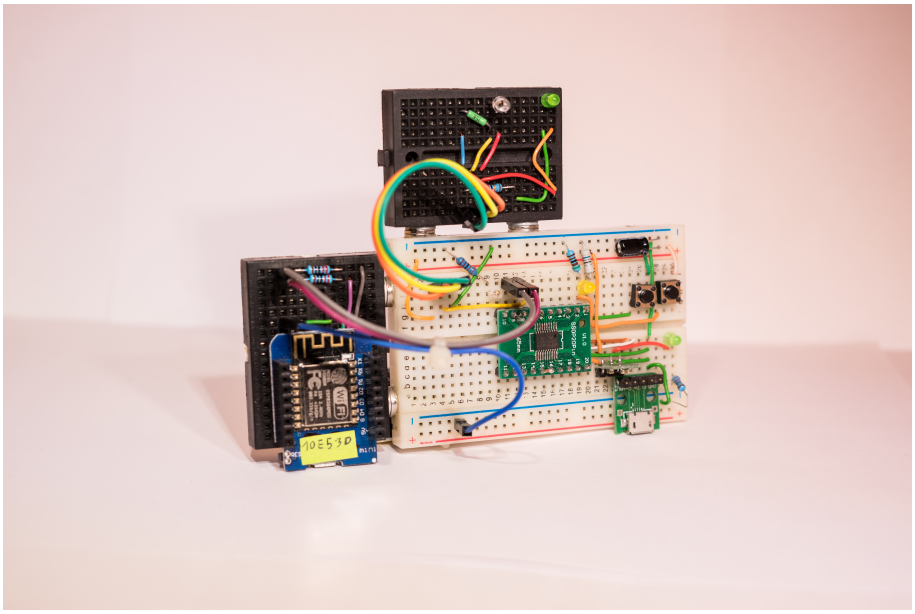
Použití ochraných rezistorů může být problematické u modulů, které mají piny vyhrazené pro sériovou komunikaci pevně spojeny se zabudovaným převodníkem. Některé převodníky totiž obsahují pull-up rezistor, který nelze použít společně s ochranným rezistorem s vysokou hodnotou. Jako příklad lze uvést zapojení modulu ESP Wemos D1 mini modul[05] s pěvně zapojeným převodníkem USB-Serial CH340[06] s mikrokontrolérem STM32F042. Pro umožnění přenosu dat prostřednictvím sériové komunikace nebylo možné použít rezistor s hodnotou vyšší než 320 Ohm.

Některé obvody nebo moduly mohou způsobovat krátkodobý nárust protékajícího proudu. Velké proudové špičky můžeme očekávat převážně u bezdrátových modulů. Krátkodobý nárust odebíraného proudu může mít za následek nepředvídatelné chování obvodu a v ojedinělých případech může dojít až k poškození modulu. Pro zamezení šíření těchto výchylek obvodem proto přidáváme v blízkosti napájení modulu paralelně zapojený keramický a elektrolytický kondenzátor.

Propojení zařízení komunikujících prostřednictvím UART musí mít správně zapojeny piny RX a TX. Pin označovaný jako TX slouží k odesílání dat. Pin RX slouží ke čtení příchozích dat. Z toho vyplývá, že pro správnou funkci je potřeba propojení pinů RX a TX komunikujících zařízení.



Obrázek 4.1: Schéma zapojení sériové komunikace

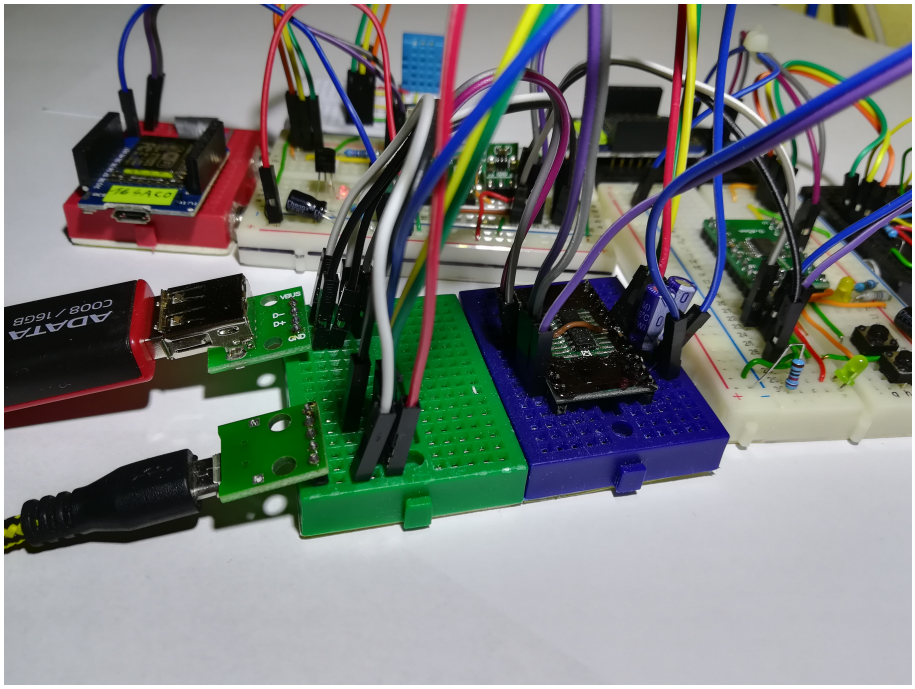


Obrázek 4.2: Kompletní zapojení obvodu pro vzdálené měření

4.1 Použití většího množství mikrokontrolérů

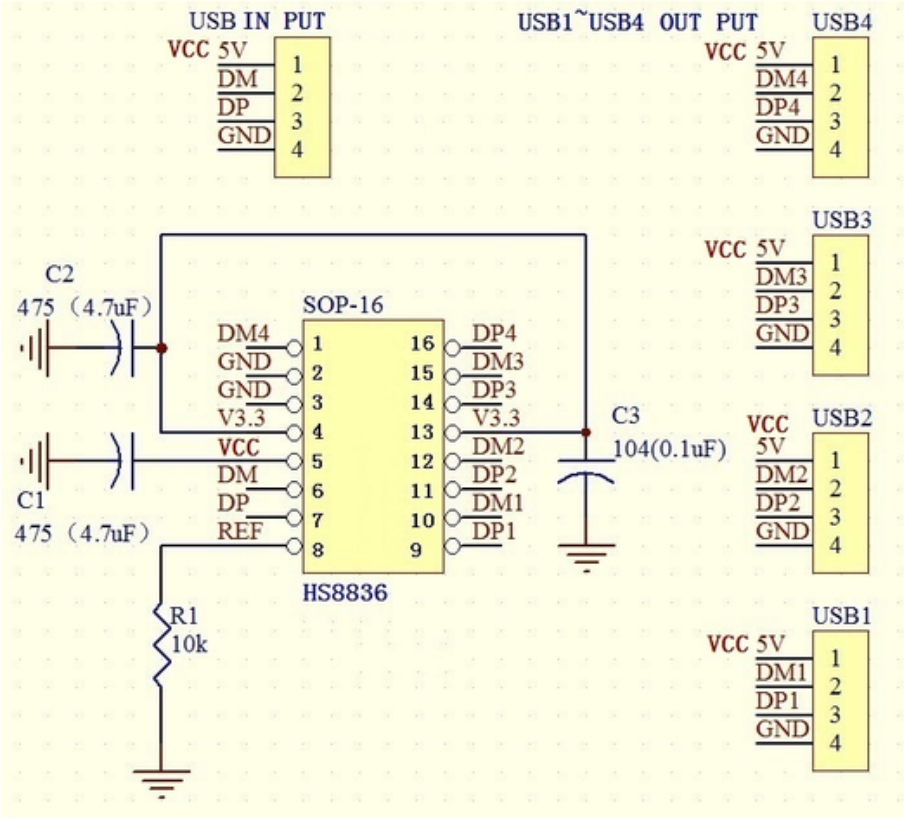
Některé aplikace pro mikrokontroléry jsou dodávány ve formě binárních souborů. Chceme li využít funkci této aplikace, pak je nutné využít samostatného mikrokontrolérů na kterém aplikace poběží. Jako příklad může posloužit programování generátoru signálu pro mikrokontroler a možnost měření aplikací osciloskopu vykonávané na druhém mikrokontroléru. Zapojení v tomto příkladu obsahuje dvě zařízení, které zároveň komunikují prostřednictvím rozhraní USB. Problém s zapojením dvou mikrokontroléru může být potřeba dvojnásobku potřebných kabelů a konektorů na obou stranách propoje.

Pro zapojení kombinace několika prototypů měřících přístrojů bylo nutné pořízení USB HUBu a množství kabelů. Problém s množstvím kabelů byl později vyřešen vyhledáním obvodu, který je využíván jako USB HUB v levných rozbočovačích. Obvody pro USB HUB jsou běžně osazeny do pouzder s velkým počtem vývodů, které jsou nevhodné pro použití v kontaktním poli. Rozebrání nejlevnějších verzí dostupných USB HUBů ukázalo na možnost použití obvodu HS8836. Obvod HS8836 v pouzdře SOP-16 značícího 16 vývodů je výbornou volbou pro použití na kontaktním poli.



Obrázek 4.3: Zapojení obvodu USB HUB HS8836 s několika USB zařízeními

Přestože byl obvod zapojen do kontaktního nepájivého pole, nedošlo k citelné degradaci rychlosti přenosu dat. Stabilita přenosu byla otestována kopírováním souboru velikosti 200MB. Datový přenos zpočátku dosahoval maximální rychlosti 800kB/s. Přenosová rychlost byla později výrazně navýšena přidáním elektrolytického kondenzátoru v blízkosti výstupního portu. Elektrolytický kondenzátor kapacity 47uF byl zapojen paralelně s výstupem napájení a země, čímž byly kompenzovány nedokonalosti kontaktního pole. Rychlost přenosu po tomto vylepšení neklesla pod hranici 7MB/s, což je běžná rychlost pro USB2.0 HUB a použitý typ flash disku.



Obrázek 4.4: Schéma zapojení HS8836

Kapitola 5

Postup a nástroje pro firmware upgrade

Vývoj programů pro vestavěné systémy je oproti osobním počítačům dosti odlišný. Z důvodu nasazení mikrokontroléru do vestavěného systému vykonávajícího často jedinou činnost odpadá potřeba operačního systému. Sériová výroba a cenové úspory limitují parametry použitého vybavení mikrokontroléru. Programování mikrokontrolérů se oproti běžným aplikacím liší v možnosti využívání velkého množství knihoven.

5.1 Programovací styl

Pro programování aplikací běžících na mikrokontroléru STM32F042 bylo využíváno knihovny mbed. Tato knihovna stejnojmené platformy poskytuje několik vrstev oddělujících použitý hardware od svého API. Využívá přitom základních konstrukcí jazyka C++ pro vývoj OOP aplikací.

Jazyk C++ s sebou přináší možnost použití řady knihoven, které nejsou dostupné pro jazyk C. I přes možnost využití jazyka C++ je důležité si uvědomit, že ne všechny konstrukce tohoto jazyka jsou vhodné při vývoji software pro mikrokontroléry.

V počátcích vývoje projektu bylo pro usnadnění práce s množstvím úprav textu využito implementace řetězců std::string jazyka C++. Volba řetězců standardní knihovny se ukázala jako velmi nešťastná. Knihovnu poskytující operace nad těmito řetězci nebylo možné optimalizovat a brzy se ukázalo, že touto volbou byla znemožněna přenositelnost na MCU s menší velikostí flash paměti. Místo použití standardních knihoven je tedy lepší použít funkcí jazyka C.

5.1.1 Programování v IDE mbed Compiler

Tato práce se zabývá nejen návrhem aplikací umožňujících komunikovat s libovolnou platformou, ale také o nezávislost na platformě při implementaci těchto aplikací. Pro programování aplikací platformy mbed lze využít online IDE[07] volně poskytovaného a vyvíjeného společností ARM. Online IDE poskytuje možnost úprav v editoru a následnou kompilaci kódu v cloudu. V

prostředí prohlížeče lze implementovat celou aplikaci a stáhnout výsledný binární soubor. Tento binární soubor poté můžeme nahrát nástroji zmíněnými níže na vybraný mikrokontrolér.

Online prostředí mbed poskytuje editor a množství nástrojů běžných pro moderní vývoj software. Programátor si může vybrat z ukázkových aplikací a rovnou začít s implementací složitějších problémů. Zdrojové kódy aplikace nebo knihovny je poté možné sdílet v online balíčkovacím systému. Celý vývoj a všechny dostupné funkce jsou k dispozici zdarma s nutností registrace na webu.

Online prostředí obsahuje

1. Textový editor a adresáře.
2. Kompilátor ARMCC C/C++.
3. Základní programy pro každý mikrokontrolér.
4. Verzovací nástroj Mercurial.
5. Balíčkovací systém obsahující programy a knihovny.
6. Vyhledávač pro balíčkovací systém.
7. Nástroj pro generovaní dokumentace.

Výhody použití online prostředí

1. Není zapotřebí instalace velkých vývojových prostředí.
2. Umožnění výběru cílového mikrokontroléru různých výrobců[08].
3. Není zapotřebí manuálního nastavení kompiléru ARMCC C/C++.
4. Přívětivé uživatelské prostředí a strmá křivka učení.

V online prostředí si můžeme vybírat mezi množstvím podporovaných zařízení pro které chceme náš kód kompilovat. Nabídka však neobsahuje seznam mikrokontrolérů, ale množství vývojových kitů podle kterých se prostředí rozhoduje. V případě použití mikrokontroléru STM32F042 si proto vybíráme platformu NUCLEO-STM32F042 s čipem STM32F042K6T6. Pouzdro mikrokontroléru, kterým se STM32F042xxxx od STM32F042K6T6 liší, v naši situaci neovlivňuje výsledný kód.

Nevýhody použití online prostředí

1. Závislost programátora na online připojení a funkčnosti prostředí.
2. V některých dnech se můžeme setkat s dlouhou odezvou prostředí.

3. Mimořádně dochází k přetížení backendu kompilátoru nebo pádům online prostředí.
4. Občas dojde k desynchronizaci editoru a jedinným řešením je pouze opětovné načtení stránky.
5. V editoru nelze kopírovat kód tlačítky myši.
6. Nelze kopírovat text z generované dokumentace.
7. Editor nepodporuje našeptávání kódu.

Součástí balíku softwaru dodávaného v přílohách této práce je implementace knihovny pro pohodlný zápis escape sekvencí. Knihovna obsahuje velké množství funkcí pro tvorbu těchto sekvencí. Cílem vytvoření knihovny bylo usnadnění zápisu sekvence výběrem příslušné funkce. Bez podpory našeptávání musí programátor pracně prohledávat dokumentaci knihovny. Dokumentace ale neumožňuje kopírovat názvy funkcí, což dále zpomaluje vývoj.

V prostředí MBED s knihovnou mbed.h je potřeba dbát na správné pojmenování souborů zdrojového kódu. V případe pojmenování souboru koncovkou .c bude prostředí zahlceno chybovými hlášeními. Tato chybová hlášení ale ukazují na knihovnu mbed, což může být značně matoucí.

5.1.2 Použití přerušení

Množství tříd knihovny mbed umožňuje volat callback funkce v případě přerušení. Neuvážené použití velkého množství obsluh přerušení může přinést neočekávané situace. Nevhodné je především použití blokujících funkcí, které nutí setrvání běhu programu ve stavu přerušení. Mezi tyto funkce patří časově nebo výpočetně náročné funkce jako například wait, printf nebo manipulace s velkou částí paměti.

5.1.3 Architektura Model-View-Controller (MVC)

Základní programy vytvořeny v této práci mohou být použity také jako kostra pro pokročilejší aplikace. Programátor využívající ukázkové aplikace musí být schopný upravovat části kódu a musí se rychle zorientovat uvnitř projektu. V některých případech programátor použije pouze část pro měření dat, ale funkce uživatelského prostředí bude upravována. V opačném případě někoho může zaujmout uživatelské prostředí terminálové aplikace, ale část měření dat bude nahradí lepší implementací. Z těchto příkladů vyplývá nutnost oddělení uživatelského prostředí a měřící části. Pro rozdělení aplikace byla proto použita známá architektura pro návrh aplikací MVC.

Architektura MVC[09] odděluje měřící a ovládací část od části starající se o interakci s uživatelem. Aplikaci proto dělíme na tři části:

1. model (model), což je část starající se o měření a reprezentaci informací, s nimiž aplikace pracuje.

2. view (pohled), který převádí data reprezentovaná modelem do podoby vhodné k interaktivní prezentaci uživateli.

3. controller (řadič), který reaguje na události (typicky pocházející od uživatele) a zajišťuje změny v modelu nebo v pohledu.

Při implementaci této architektury v textové aplikaci tato architektura řeší další problémy. Výhodou tohoto rozdělení je i ochrana před voláním funkcí pro změnu uživatelského rozhraní z měřící části. Měřící část často obsahuje obsluhu přerušení, které vykonává měření vzorků závislého na časové prodlevě. Měření tedy nesmí nabýt závislosti na časově náročných změnách uživatelského rozhraní.

5.1.4 Přenositelnost a makra jazyka C

Použití jazyka c a implementace využívající specifické vlastností mikrokontrolérů znemožňuje přenos programu mezi jednotlivými řadami mikrokontrolérů. Tento problém lze řešit použitím knihovny obalující běžně používané funkce mikrokontroléru. Všechny programy realizované pro mikrokontrolér STM32F042 využívají knihovny mbed. Zdrojové kódy lze proto kompilovat pro různé mikrokontroléry od firmy STMicroelectronics. V mnoha případech pak není potřeba zásahu do kódu aplikace.

Problém nastává při použití vývojového kitu nebo mikrokontroléru s jiným počtem pinů. Některé specifické funkce pinů jsou umožněny jen na vybraných pinech mikrokontroléru. Různé značení pinů, různé rozmístění požadované funkce mezi piny a další rozdíly v parametrech mikrokontroléru jsou překážkou v přenositelnosti. Některé rozdíly je možné řešit použitím maker jazyka C a využitím definice zvoleného mikrokontroléru v online IDE mbed. Například rozdílné umistění LED může být vyřešeno podmínkou využívající proměné nastavené vývojovým prostředím.

```
#if defined(TARGET_STM32F0)
    #define PIN_LED PA_4
#elif defined(TARGET_STM32L0)
    #define PIN_LED LED1
#else
    #warning Please set PIN_LED for this platform
    #define PIN_LED LED1
#endif
```

5.2 Nástroje pro firmware upgrade

Online prostředí nemá přístup k hardware připojeného k zařízení a z tohoto důvodu není možné program do tohoto zařízení nahrávat přímo. Prostředí proto generuje binární soubor, který umožní uživateli stáhnout do libovolného adresáře. Tento binární soubor je poté nutné přenést do připojeného zařízení.

Metoda nahrávání se liší podle použitého operačního systému, mikrokontroléru a v některých případech také volbou převodníku.

Následující část kapitoly bude věnována především operačnímu systému Ubuntu a macOS. Zkušenosti s nahráváním firmware pod operačním systémem windows jsou již popsány na veřejně dostupné subdoméně fakulty[13a].

5.2.1 Debugger a programátor ST-LINK

Výhodou použití kitu obsahujícího obvod ST-link[10](např. Nucleo) je multiplatformnost i v případě upgrade firmwaru. Mikrokontrolér umístěný na obvodu ST-LINK využívá USB knihovny pro vytvoření malého virtuálního uložiště. Virtuální uložiště je implementaci protokolu USB Mass Storage, který je podporován všemi moderními zařízeními. Upgrade firmwaru pak může být uskutečněn pouhým kopírováním binárního souboru do tohoto uložiště, které se pro OS jeví jako malý USB flash disk.

Při připojení nového zařízení se mohou vyskytnout problémy v podobě nerozpoznání zařízení. Tento problém bývá způsoben zastaralým firmware v mikrokontroléru obvodu ST-LINK. Starší verze například nepodporují USB3.0 což lze řešit použití USB2.0 Hubu. Další problém spočíval v potřebě odpojení obvodu ST-LINK po dokončení upgrade firmware a nemožnost nahrání nového souboru do virtuálního uložiště. Všechny problémy se podařilo vyřešit aktualizací mikrokontroléru v obvodu pomocí aplikace dostupné na stránkách st.

Program st-flash pro upgrade mikrokontroleru prostřednictvím ST-LINK

Při vývoji může byt upgrade firmwaru prováděný kopírováním do virtuálního uložiště problematický z důvodu chybějící zpětné vazby o stavu kopírování firmware. Pro více možností můžeme nainstalovat nástroj st-flash, který umožňuje komunikaci s modulem ST-LINK. Aplikace st-flash umožňuje rozpoznat průběh firmware upgrade proběhl úspěšně, ale může být také nápomocná v případě problémů s virtualním uložištěm.

5.2.2 Program pro upgrade USB zařízení dfu-util

V případě zapojení mikrokontroléru bez použití převodníku použijeme nástroj DFU (Device Firmware Upgrade) zaměřený na kopírování nového firmware do USB zařízení. Protože náš mikrokontrolér podporuje komunikaci s USB sběrnicí, mužeme binární soubor generovaný vývojovým prostředím nahrát prostřednictvím tohoto standardu. Nástroje umožňující proces DFU[11] nejsou základní výbavou operačních systému a potřebujeme proto nainstalovat příslušný software. Mikrokontrolér musí být tímto systémem rozpoznán a proto je nutné uvést mikrokontrolér do DFU módu. Instalací tohoto software se zabývá dokument v přílohách.

5.2.3 Vytvoření programu pro automatický upgrade

Ladění programu v některých případech vede na opakovaný upgrade firmwaru. Manuální výběr souboru a potvrzení upgrade firmwaru může být v součtu časově nákladný. Pro časovou úsporu byl při této práci implementován a využíván skript pro automatický upgrade binárního souboru. Skript automaticky spouští program st-flash nebo dfu-util a nahraje soubor vygenerovaný prostředím mbed do připojeného zařízení.

Skript v jazyce bash je dostupný v přílohách a to ve dvou verzích. První pro operační systém Ubuntu a druhá pro MacOS od společnosti apple. Skript je závislý na přítomnosti nástrojů pro firmware upgrade a proto nebude fungovat bez instalace požadovaných balíčků. Dále je vhodné nastavit umístění, do kterého bude ukládán generovaný binární soubor. V případě, že skript běží, bude ihned po stažení binárního souboru zahájen upload do připojeného zařízení.

5.3 První firmware upgrade

5.3.1 Program blikající LED pro otestování zapojení

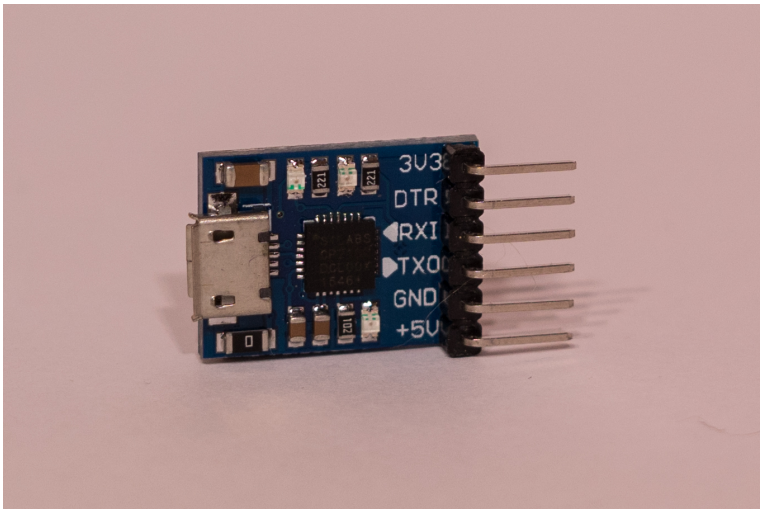
V tomto kroku ověříme správné zapojení mikrokontroléru v zkušebním poli. Před zapojením složitějšího obvodů je důležité otestovat zda má mikrokontrolér správně zapojeno napájení a jestli je možné nahrávat nový firmware. Tento krok je obvykle testován použitím triviálního programu pro rozblikání připojené LED.

5.3.2 Instalace ovladačů pro převodník USB-UART

V případě propojení zařízení s USB prostřednictvím převodníku USB-UART je pro rozpoznání zařízení nutné nainstalovat ovladače. Zařízení je pak rozpoznáno jako virtual COM port (VCP) a dojde k přiřazení adresy tomuto zařízení. Použití převodníku nemusí být zřejmé díky integraci v některých modulech s konektorem USB.

Ovladače jsou specifické pro každý systém, což je překážkou ve snaze této práce o funkci na všech platformách. V systému macOS není k dispozici žádný nativní ovladač podporující virtual COM port. Pro operační systém Windows se dokonce implementace ovladače liší podle nainstalované verze systému.

Na vývojovém kitu ESP8266 obsahujícím USB-Serial převodník bývá osazen čip CH340G. Vývojové kity nucleo virtualizují rozhraní pro které je potřebná instalace ovladače CP210x. Tento ovladač se také nachází na některých levných externích USBtoUART převodnících. Ovladače pro CP210x[13b] jsou implementovány pro každý operační systém zvlášť. Volně ke stažení existuje několik implementací od různých autorů. Instalační soubory byly proto otestovány a přidány do příloh této práce.



Obrázek 5.1: Externí USBtoUART převodník

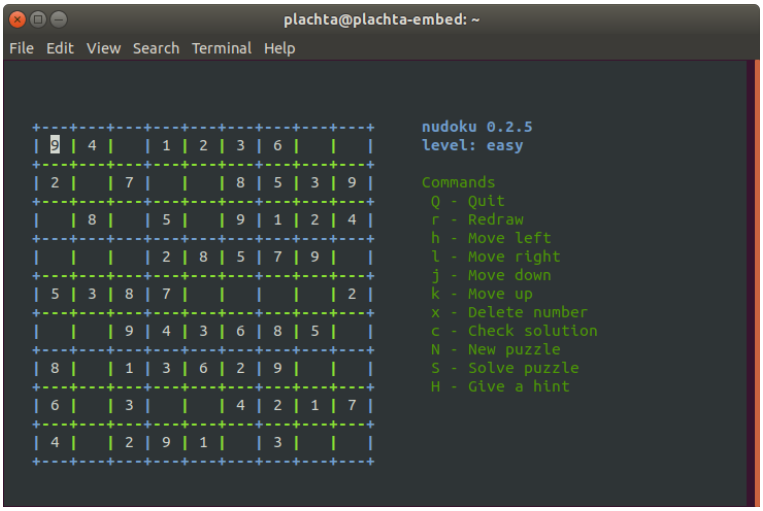
5.3.3 Nastavení oprávnění správce udev

Systémy s linuxovým jádrem obsahují správce zařízení *udev*, který řídí připojení externích zařízení a jejich oprávnění. Nově připojené zařízení má jen základní oprávnění. Pro nahrání firmware nebo komunikaci s připojeným zařízením je proto nutné nastavit dostatečná oprávnění v konfiguračních souborech správce *udev* Možné příznaky při nedostatečném oprávnění se mohou vyskytovat v podobě hlášky „**device is busy**“ nebo více nápomocné „**Permission denied**“. Pro práci se zařízením proto nastavíme oprávnění v konfiguračních souborech Udev. Pro mikrokontroléry řady STM32 lze využít manuál a bash skript obsažený v přílohách této práce.

Kapitola 6

Terminály a textové aplikace

Měření fyzikálních veličin a stav ovládaného objektu je ve většině případů možné zobrazit v textové podobě. Tato vlastnost umožňuje použít terminál jako výchozí zobrazovací aplikaci. Použití terminálu umožňuje běh na velkém množství operačních systémů a zároveň se můžeme vyhnout instalaci úzce zaměřených aplikací.



Obrázek 6.1: Možnost tvorby pseudografických her (Nudoku - Michael Vetter)

6.1 Sériová komunikace

Digitální data můžeme přenášet paralelně či sériově. Mikrokontroléry běžně využívají asynchronní sériovou komunikaci UART pro malé nároky na rychlost přenosu dat a minimalizaci počtu použitých vývodů. Pro komunikaci prostřednictvím sériové komunikace rs-232 potřebujeme pin pro příjem dat RX dále pak pin pro odesílání dat TX a vodič spojující zem komunikujících obvodů. Z důvodu častého použití tohoto druhu komunikace mikrokontroléry obsahují několik dvojic vývodu RX a TX s hardwarovou podporou.

Mikrokontroléry umožňují připojení množství modulů se specializací na určitou činnost jako měření, generování signálu nebo komunikaci. Tyto moduly

mohou poskytovat nebo přijímat již zpracovaná data v digitální podobě. V prostředí mikrokontroléru je využíváno množství různých druhů sériové komunikace jako I2C, SPI, RS-485, RS232. Připojit takto můžeme množství modulů jako ESP8266, GPS, GSM a další.

6.1.1 UART - universal asynchronous receiver-transmitter

Sériová komunikace prostřednictvím UART je oproti možnostem mikrokontrolérů relativně pomalá. Při volbě rychlosti komunikace nejčastěji volíme mezi hodnotou baudrate 9600 nebo 115200. Navzdory použití slova asynchronní v názvu tohoto protokolu nelze v knihovně mbed funkce pro seriový výstup asynchronně volat. Čekání na odeslání textu se z těchto důvodů musíme vyvarovat v částech kódů vykonávaných jako obsluha přerušení. Při dlouhém čekání by mohlo dojít k odkládání jiných důležitých přerušení a neočekávaným stavům do kterých by se naše aplikace mohla dostat.

6.1.2 Sériová komunikace platformy mbed

Zvýšené pozornosti je potřeba dbát při přijímání znaků, kde je musíme upřednostnit příjem informace z bloku UART. Z důvodu omezené velikosti bufferu pro sériovou komunikaci může docházet ke ztrátě přijímaných informací v případě přetečení. Překvapivé může být také chování funkce Serial::getc, která po přečtení jediného znaku zahodí zbytek přijatých informací. Další funkce Serial::read umožňuje data načíst do námi připraveného bufferu pevné velikosti. Tato funkce poté blokuje vykonávání kódu, dokud nedorazí požadovaný počet znaků zadaný při volání funkce.

Pro některé programátory může být používání standardní knihovny pro obsluhu UART příliš nepraktické. Naštěstí můžeme použít volně dostupnou knihovnu BufferedSerial, která za nás čtení do bufferu řeší přerušením při příchodu znaku. Knihovna není náročná na velikost paměti flash a velikost bufferu pro práci s daty lze nastavit při inicializaci. Při použití vývojového prostředí mbed můžeme tuto knihovnu jednoduše importovat nástrojem s vyhledáváním v tomto prostředí.

6.1.3 Escape sekvence

Escape sekvence používané v této práci jsou posloupnosti znaků využívané ke změně významu dat posílaných ve formě textu. Tato sekvence začínající kontrolním znakem ESC umožňuje rozlišit zda jsou data určená pro tisk na obrazovku nebo zda tato data představují nějaký interní příkaz k provedení. Tímto způsobem zajistíme změny v chování zobrazujícího zařízení bez narušení textové komunikace pro výpis na obrazovku. Sekvence jsou nezbytné pro běh některých terminálových aplikací a podrobněji budou rozebrány v nasledujících kapitolách.

Příklad důležitých funkcí podporovaných terminály:

- Změna polohy kurzoru
- Vymazání obrazovky
- Kontrola barev a grafických nastavení

6.2 Terminál

Terminál vytváří vzdálené uživatelské rozhraní, pomocí kterého uživatel zařízení ovládá připojené zařízení. Umožňuje spouštět programy, zadávat vstupy a zobrazovat programové výstupy. Dřívější terminály v podobě hardwarového zařízení podporovaly pouze jednoduché zadání a výstup dat. Později nabyly schopnosti ovládání kurzoru, barevného výstupu a dalších užitečných funkcí realizovaných pomocí escape sekvencí. Později byly hardware terminály nahrazeny jejich emulacemi. Tyto emulace byly označovány jmény souvisejicími s funkcemi podporovanými jejich vzory v hardware provedení. Emulátory terminálů zůstaly součástí většiny moderních operačních systémů.

6.3 Emulátory terminálu

Emulátory terminálu jsou využívány k ovládání množství textových aplikací operačního systému. Dále jsou používány například pro vzdálenou administraci. V této práci budou použity pouze pro připojení k virtual COM portu. Emulovaný terminál nám umožní výpis dat odeslaných připojeným mikrokontrolérem. Uživatel bude moct také odesílat textové zprávy a tímto připojený mikrokontrolér ovládat.

6.3.1 Emulátory používané v moderním operačních systémech

Emulátor terminálu je grafická aplikace, která v grafickém uživatelském prostředí vytváří okno, uvnitř kterého je emulován textový terminál. Nejznámější terminály umožňující zobrazení implementovaných aplikací jsou zmíněny v následujícím seznamu.

Testované terminály vyhovující pro běh pseudografických aplikací:

PuTTY - aplikace pro Windows i unixové systémy

xterm - emulace pro systémy obsahující X Window System

gnome-terminal - emulace xterm terminálu v prostředí GNOME (výchozí terminál systému Ubuntu)

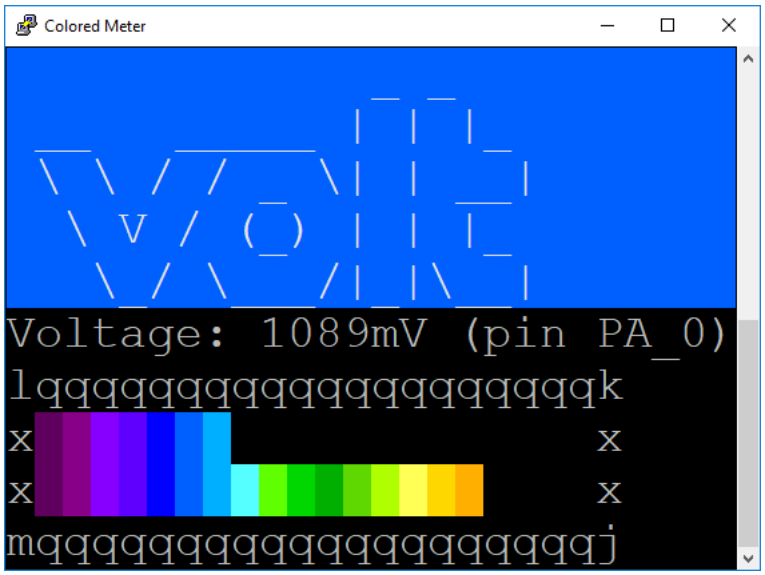
Terminal.app - emulace xterm-256color terminálu v macOS operačním systému od Apple

Emulátor terminálu umožňuje běh terminálových textových aplikací. Výhodou tohoto druhu aplikací je nezávislost na softwaru vytvářejícím grafické uživatelské prostředí (GUI). Použití knihovny pro specifické GUI často zapříčiní závislost na dané platformě a brání tak v snadné tvorbě multiplatformních aplikací.

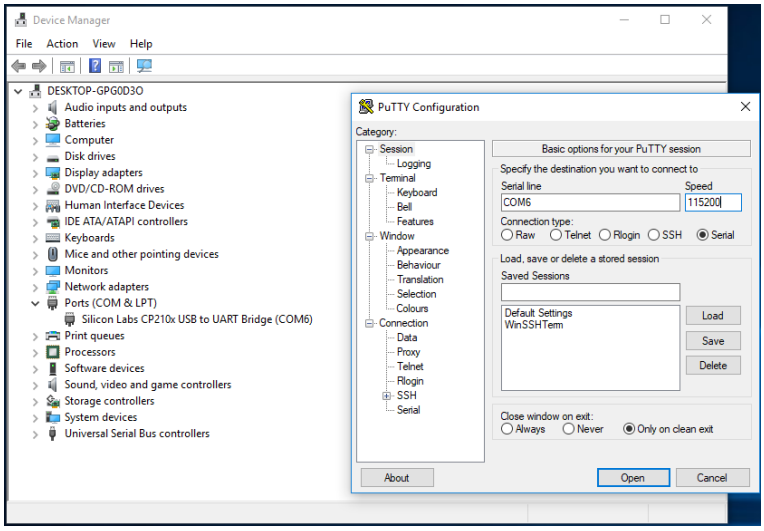
U terminálu můžeme globálně měnit velikost písma a výchozí barvu písma i jeho pozadí. Dále můžeme upravovat velikost terminálu podle potřeb uživatele. Některé emulátory také podporují vytvoření záložek podobných záložkám známým z prohlížeče webových stránek.

6.3.2 PuTTY ve Windows

PuTTY je open-source software původně vyvíjený pro windows. Tento software byl dostupný také pro ostatní platformy díky práci dobrovolníku. Dnes je dostupný pouze pro windows a unix systémy. Podpora pro macOS byla zrušena kvůli ukončení podpory pro zdejší zobrazovací server.



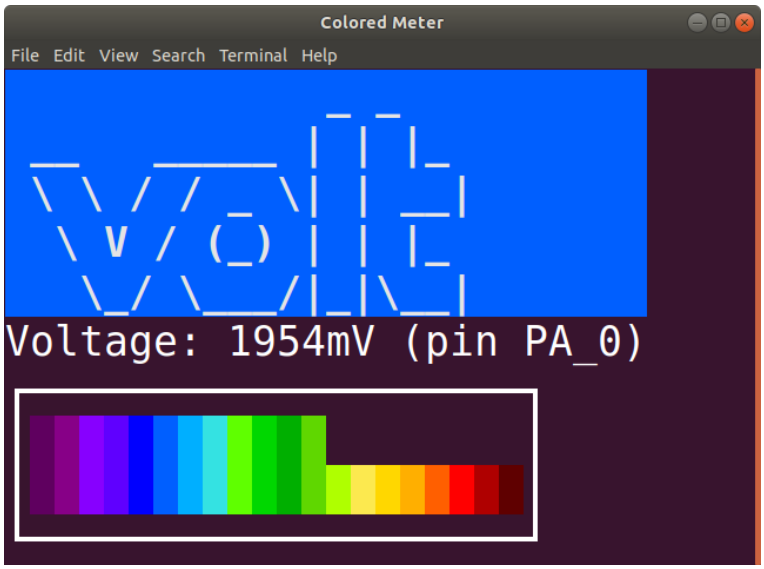
Obrázek 6.2: Aplikace voltmetru v PuTTY



Obrázek 6.3: Konfigurace emulátoru terminálu v PuTTY

6.3.3 Terminál gnome-terminal v Ubuntu

Aplikace Terminal je dodáván jako základní vybavení systému Ubuntu. Tento terminál umožňuje spustit terminálovou aplikaci GNU screen, která obstarává sériovou komunikaci. Tato aplikace sice není součásti systému ale je obsažena ve správci balíčků apt, který zajišťuje její snadnou instalaci. Výchozí velikost terminálu po spuštění je 80 sloupců na 20 řádků.

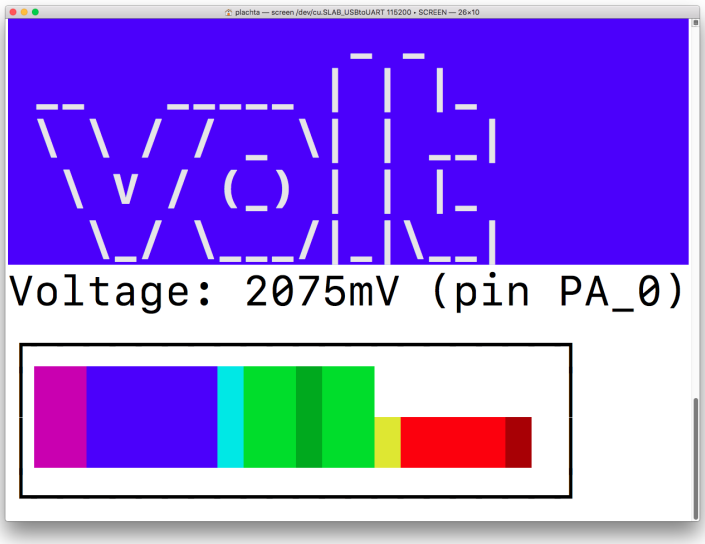


Obrázek 6.4: Aplikace voltmetru v gnome-terminal

6.3.4 Terminál terminal.app v macOS

Tento emulátor terminálu je dodáván jako základní vybavení systému macOS. Emulátor nám umožňuje spustit terminálovou aplikaci GNU screen, která je

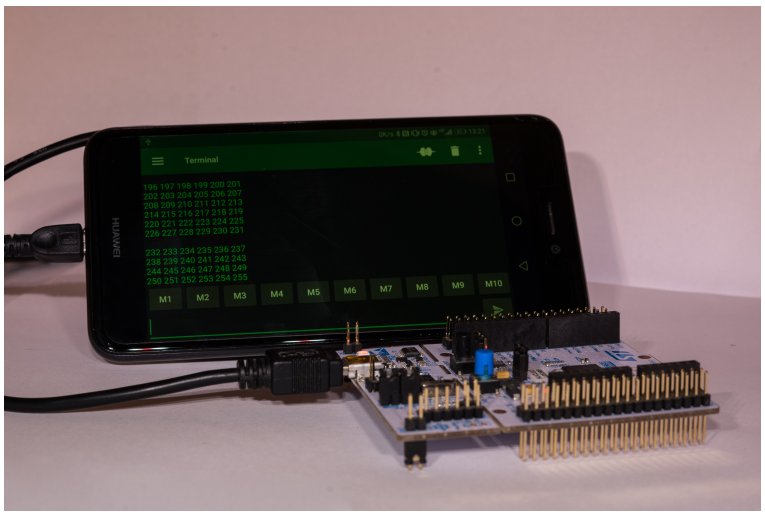
na rozdíl od odlehčené verze Ubuntu v tomto systému základní výbavou.



Obrázek 6.5: Aplikace voltmetru v terminal.app

6.3.5 Serial USB Terminal v Androidu

Emulátor terminálu pro operační systém Android s podporou různých ovladačů umožňuje komunikovat se zařízeními připojenými kabelem USB On-The-Go. Tento terminál neumožňuje využívat escape sekvence, ale netrpí zobrazováním znaků určených pro kontrolní funkce. I přes absenci kontrolních funkcí může být tento terminál přínosný v situacích, kdy jsme omezeni na použití zařízení s operačním systémem Android. Možnost využití terminálu na mobilním telefonu nebo tabletu není příliš známá, demonstruje ale možnost použití textových aplikací na rozmanité množině zařízení.



Obrázek 6.6: Kit nucleo připojen k telefonu s aplikací Serial USB Terminal

Serial USB Terminal podporuje komunikaci s využitím ovladačů USB CDC, FTDI FT231/FT232, CP210x, Quinheng CH340, Prolific PL2303. Rozmanitost podporovaných ovladačů umožňuje připojení všech převodníků použitých v této práci. Například kit Nucleo (CDC), mikrokontrolér STM32F042 (CDC) i komunikační modul ESP8266 (CH340) tak mohou zobrazovat textový výstup na tomto terminálu.

Pro použití terminálové aplikace v systému Android je zapotřebí podpora technologie OTG. S touto technologií lze přepnout mobilní zařízení do hostujícího módu. Některé telefony tuto funkci nepodporují. Zmínění možnosti spojení mikrokontroléru a mobilního telefonu slouží jako ukázka možnosti využití sériové komunikace a mobilního zdroje elektrické energie. Užitečné by mohlo být také využití možnosti komunikace prostřednictvím mobilních sítí v případě implementace android aplikace.

6.4 Aplikace GNU screen

Aplikace GNU screen spravuje běh terminálových aplikací a připojení k VCP v emulátoru terminálu. Někdy je tato aplikace označována jako okenní multiplexor. I když se GNU screen často využívá pro správu aplikací, uživateli stačí znát pouze základní kroky pro připojení k sériovému portu.

Před připojením k zařízení je potřeba zjistit adresu VCP, která byla tomuto zařízeni přiřazena. Tato část se bohužel liší v závislosti na operačním systému.

1. V OS Windows je připojené zařízení indexováno ve tvaru COM#, kde # značí číslo v pořadí připojených zařízení.
2. V OS Ubuntu je možné ovlivnit odkaz na zařízení při úpravě konfiguračních souborů udev. V případě výchozího nastavení se zařízení připojí do adreváře /dev/ s názvem ttyUSB# nebo ttyACM#, kde # značí číslo v pořadí připojených zařízení.
3. V OS macOS je připojené zařízení nazváno podle odhadu typu tohoto zařízení. Podobně jako v OS Ubuntu se zařízení nachází v adresáři /dev/ a název často začíná písmeny **cu.** jako například /dev/cu.slabUSBtoSERIAL

Další úkon závislý na použitém operačním systému je bezpečné uvolnění sériového portu a následné ukončení aplikace. V případě násilného ukončení se může stát sériový port nedostupný pro budoucí připojení. Podrobněji jsou tyto úkony popsány v přiloženém souboru.

6.5 Podpora ANSI sekvencí

Při testování aplikací v terminálových emulátorech se ukázalo, že i přes podporu escape sekvencí je v některých případech implementace neúplná. Například v operačním systému macOS a jeho zakladním terminálu je možné

zobrazit pouze 16 barev. V terminálové aplikaci PuTTY nelze využít sekvenci pro přepínání znakových sad.

Mezi známé testované emulátory, které nepodporovaly funkce pro změny barev nebo pohyb kurzoru patří například Realterm (Windows) a Coolterm (macOS). Podporu escape sekvencí lze rozdělit do několika kategorií.

1. Terminál neošetřuje kontrolní znak escape a vytiskne parametry (znaky) escape sekvence
2. Terminál ošetřuje kontrolní znak escape
   a. Terminál ignoruje escape sekvence a netiskne je na obrazovku.
   b. Terminál vykonává akce náležící escape sekvencím

V případě že emulátor terminálu ignoruje escape sekvence můžeme použít kontrolní znaky, které mohou vykonávat ty nejzákladnější funkce. Pro terminály bez podpory escape sekvencí není zapotřebí manuálně přepínat mezi výpisem tvořeným escape sekvencemi a kontrolními znaky. Součástí standardu je totiž podpora sekvence pro dotaz na verzi terminálu zobrazujícího výstup. V případě úspěšného přijetí odpovědi na vyslanou sekvenci Send Device Attributes máme zaručenou podporu sekvencí daných přijatou verzí.

Kapitola 7

Návrh terminálové aplikace

Měřící programy běžně poskytují výstupní data v textové podobě. Pro zobrazení stručných informací mikrokontrolérem se někdy využívá textového dipleje. V některých případech je ale snazší využít spojení se zařízením sériovou komunikaci s využitím terminálu. Terminál je plnohodnotnou náhradou za textový displej a umožňuje také zpětné zasílání textu a příkazů.

V programovacím jazyce C jsou znaky uloženy pomocí 7 bitů v datové struktuře char. Znaky jsou kódovány na čísla podle tabulky ASCII. Tato tabulka *C0* obsahuje 128 znaků z nichž část není určena k tisku nebo zobrazení. Jedná se o prvních 32 *kontrolních znaků* rezervovaných pro kontrolní funkce. Tyto kontrolní znaky umožňují vykonávat základní funkce v terminálu zobrazujícím textová data. Pro zjednodušení zápisu netisknutelných znaků je v jazyce C použito escapování pomocí zpětného lomítka a zástupného znaku.

Standard ASCII obsahuje také rozšíření základní tabulky *C0* o další úsek C1 obsahující znaky chybějící v základní tabulce. Rozšíření obsahuje dalších 128 znaků, které nejsou příliš podporovány a mohou vést na nejednotné zobrazení. V případě nutnosti použití takového znaku můžeme v C provést zápis tohoto znaku v hexadecimální formě "**xHH**". Pro použití jednociferné hodnoty hexadecimální formy požadovaného znaku je nutné použít před touto cifrou také nulu, pro správné rozpoznání požadované hodnoty.

V případě že potřebujeme speciální znak, který nutně nemusí pocházet z ASCII tabulky, pak je lepší variantou použití alternativy ve znakové sadě unicode. Unicode znak označený jako U+**HHHH** můžeme v C řetězci zapsat jako "\u**HHHH**". Znaky jsou podporovány většinou terminálů a použití by nemělo činit žádné problémy. Výhodou použití formátu UTF-8 je také možnost kreslení množství grafických znaků z této sady.

7.1 Kontrolní znaky

Kontrolní znaky nám umožňují základní formátování textu. Jejich funkce jsou omezené. Hlavní nevýhodou je nemožnost cestování mezi řádky, což zapřičiňuje potřeby odřádkovávat vypsaný text. Analogií může být tisk na papír účtenky, která může nepřetrčitě poskytovat nová data, ale ta stará jsou

stále viditelná. Návrh aplikací využívajících kontrolní znaky je proto velmi nepraktický a omezen.

Zápis	Význam česky	Význam anglicky	DEC [1]	HEX [2]
"\a"	zvonek	bell	07	07
"\b"	backspace	Backspace	08	08
"\t"	tabulátor	tab	09	09
"\n"	nový řádek	line feed	10	0A
"\v"	vertikální tabulátor	vertical tab	11	0B
"\f"	nová stránka	form feed	12	0C
"\r"	návrat vozíku	carriage return	13	0D

[1] Dekadická reprezentace znaku v tabulce ASCII
[2] Hexadecimální reprezentace znaku v tabulce ASCII

Tabulka 7.1: Seznam kontrolních znaků

Mezi kontrolními znaky můžeme najít speciální znak ESC. Tento znak je využíván pro rozšíření funkcí kontrolních znaků zavedením escape sekvencí. S použitím knihoven lze tvořit zajímavé aplikace i s těmito základními znaky. Nevyhodou je ale mnohem větší množství zasílaných znaků.

7.2 Escape sekvence

Escape sekvence jsou důležitý mechanizmus pro interpretaci sériové komunikace a tvorbu interaktivních textových aplikací. Sekvence kontrolující barvy umožňují tvořit pseudografiku aplikace změnou barvy písma nebo pozadí. Emulátory terminálu implementují velké množství sekvencí pro ovládání dalších funkcí terminálu. Sekvence se od sebe liší nejen svou délkou ale také způsobem jakým tyto sekvence syntakticky analyzujeme (parsování).

Speciální sekvence také používáme pro rychlé umístění kurzoru na libovolnou pozici. Tato funkce umožňuje vytvářet mnohem pokročilejší programy bez nutnosti překreslení terminálu v případě návratu kurzoru na předchozí řádek. Sekvence tímto umožňují šetřit počet zaslaných znaků a zrychlit odezvu aplikace. Tato funkce je zvláště užitečná pro textové editory, kde vidíme celou stránku a obsah části textu lze měnit najetím kurzoru na požadovanou pozici. Další ukázkou mohou být textové aplikace s menu a tlačítky (tzv. textové uživatelské rozhraní). Textové aplikace často využívají také pseudografiku.

Velké množství sekvencí a globální podpora zapřičinila vznik standardu ECMA-48 sjednocujícího tyto sekvence. Standard popisuje správné chování desítek sekvencí a jejich formát. Standard používá binární namísto hexadecimální reprezentace znaků tabulky ASCII. Standard může být pro vývoj příliš obsáhlý a proto je někdy vhodné čerpat ze stručnějšího popisu. Problematikou zapisování escape sekvencí se zabyvá množství webů[Moy]. V případě vývoje pro terminál ESPTerm je vhodné z důvodu drobných odlišností navštívit stránky dokumentace tohoto projektu[16].

7.2.1 Generování escape sekvencí

Sekvence	Reprezentace ECMA-48	Zápis v C
Escape (ESC)	01/11	"\x1B"
Save Cursor	ESC 7	"\x1B7"
CSI [1]	ESC 05/11 [2]	"\x1B\x5B"[3]
Cursor Up Pn Times	CSI Pn 04/01	"\x1B\x5B%d\x41"

[1] Zkratka pro escape sekvenci Control sequence introducer
[2] Existuje také ve formě znaku ASCII C1 sady 09/11
[3] Lze zapsat také jako jediný znak "\x9b"(nedoporučeno)

Tabulka 7.2: Ukázka zápisu sekvencí

Všimněme si, že zápis sekvence CSI lze provést zápisem jediného znaku. Jak již bylo zmíněno výše, používání kontrolních znaků sady C1 může mít za následek nekompatibilitu s množstvím terminálů. V sekvencích je dobrým zvykem použití sekvence pro zápis kontrolních znaků ze sady C1. Escape sekvence pak vždy začínají znakem ESC a jsou tůdíž snadněji dekódovatelné pro přijímající zařítení. Pro otestování funkce sekvence **Select Graphic Rendition** můžeme použít nástroj echo.

```
MacOS Terminal.app

bash:~ USER$ echo -e "\x1B\x5B34mHello\x1B\x5B0m"
Hello
```

Escape sekvence jsou zapisovány mezi odesílané řetězce. Z důvodu velkého množství těchto sekvencí byla pro lepší čitelnost implementována knihovna. Hledání podle názvu sekvence pomáhá čitelnosti kódu a omezuje množství překlepů. Knihovnu lze nalézt v přílohách této práce.

7.2.2 Parsování escape sekvencí

Parsování příchozích escape sekvencí bylo realizováno třídou využívající implementaci stavového automatu. Tato třída byla realizována z důvodu úspory času při nakládání s velkým množstvím možných sekvencí. Příchozí znaky můžeme po jednom vkládat voláním funkce parse. V případě přechodu do koncového stavu je vykonána akce na základě rozpoznané příchozí sekvence.

7.3 Využití prostoru pro tisk

Terminálové aplikace mohou poskytovat plnohodnotné uživatelské prostředí. Umožňují zobrazovat informace v textové podobě. Úpravami atributů textu můžeme docílit speciálního významu zobrazovaného textu. Například změnou barvy pozadí znaků můžeme uživatele upozornit na požadovaný úsek textu

a zjednodušit orientaci v aplikaci. Dalším důležitým nástrojem je tvorba pseudografických rámců, které logicky rozděluji místa pro zobrazovaný text.

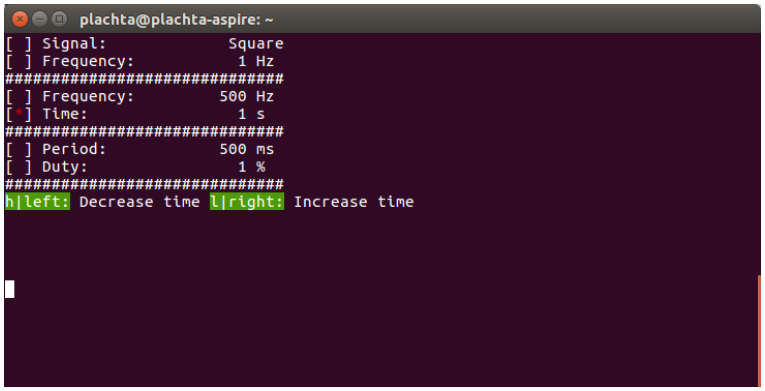
Pro správné zobrazení aplikace je vhodné myslet na velikost terminálu po spuštění

Operační systém/aplikace	Počet sloupců	Počet řádků
Prohlížeč/ESPTerm	26	10
Windows/PuTTY	80	24
Ubuntu/Terminal	80	24
macOS/Terminal	80	24

Tabulka 7.3: Výchozí velikosti terminálových aplikací

Aplikace musí počítat s omezeným prostorem terminálu pro tisk znaků. K dispozici je možnost měnit rozlohu terminálu změnou počtu řádků a sloupců. Standardně se ale možnosti úprav velikosti nevyužívá z důvodu neznámé maximální velikosti pro zobrazovací zařízení. Omezení na počet zobrazovaných znaků lze obejít například využitím menu, kterým uživatel zvolí požadovanou část obsahu aplikace.

Použití menu umožňuje také snížit počet potřebných ovládacích znaků. V případě realizace aplikace umožňující nastavení několika odlišných hodnot bylo možné využít klávesy šipek pro všechna dostupná nastavení.

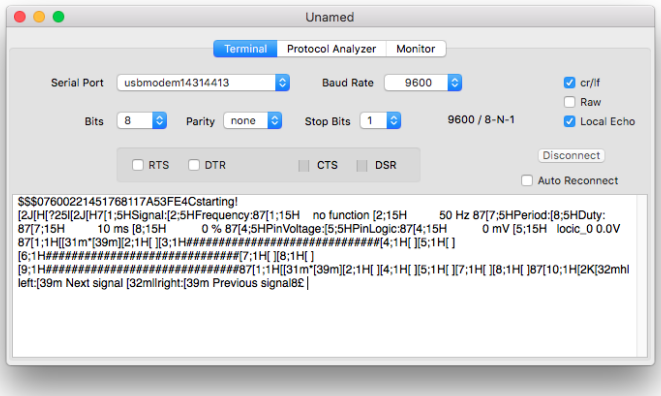


Obrázek 7.1: Ukázka využití menu pro ovládání s omezeným počtem kláves

Při návrhu je vhodné vyhranit volný prostor, který bude určen pro zobrazení důležitých informací. I přes malý prostor je velmi důležité uživatele zřetelně upozornit v případě abnormálních hodnot, či při přechodu do stavu ve kterém hrozí nebezpečí. V takovém případě je vhodné využít změn barev znaků a vykreslení na vyhraněném iditelném místě.

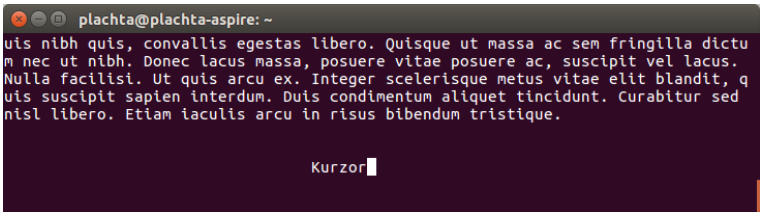
Použití některých terminálu jako např. ESPTerm umožňuje využít přidaných tlačítek, která nejsou běžnou součástí emulátoru terminálu. Při použití terminálu bez možnosti nastevení vlastního makra by měla být zachována možnost obsluhy aplikace. Z těchto důvodů je vhodné imlementovat alternativní možnost ovládání prostřednictvím zasílání textových znaků.

Problém může nastat také v případě, kdy uživatel použije terminalovou aplikaci s neúplnou podporou sekvencí escape. Některé terminály sekvence escape vůbec nepodporují a tyto ziskané sekvence tisknou jako běžný text což má za následek degradaci čitelnosti. Tento problém je možné částečně přirovnat k webovému prohlížeči nepodporujícího kaskádové styly CSS.



Obrázek 7.2: Ukázka terminálové aplikace ignorující ESC charakter

Při návrhu rozvržení umísťujeme text tak, aby bylo zapotřebí co nejméně návratů a skoků kurzoru. V případě časté aktualizace části zobrazovaných dat je vhodné pro zobrazení této informace vyčlenit pevný prostor. Využitím skoku kurzoru poté můžeme aktualizovat informace bez přepisu statických části a ušetřit tím počet přenesených znaků. Escape sekvence se skládá z několika málo znaků a často je výhodnější využít skoku na přesnou pozici oproti zaslání množství znaků mezer a odřádkování.



Obrázek 7.3: Terminálová aplikace demonstrující adresaci kurzoru

7.4 Implementace aplikace v jazyce C++

Programování terminálových aplikací se opírá o zpracování velkého množství textu. V jazyce C++ existuje množství dostupných funkcí zahrnutých v Standard Template Library (STL). Použití těchto funkcí ale nebylo navrženo s ohledem na využití v prostředí mikrokontroléru. Funkce lze použít za cenu zaplnění flash paměti a nutnosti převodu na řetězce jazyka c při volání knihovních funkcí.

Toto tvrzeni je podloženo kompilací několika testovacích programů ve dvou různých verzích. Jedna verze používá běžné konstrukce používané v jazyce C a verze druhá využívá funkcionality standardní knihovny STL. Prostředí mbed umožňuje zobrazit celkové využití paměti flash. Otestováno bylo několik scénářů použití knihoven pro mikrokontrolér STM32F042 a také nukleo kit s mikrokontrolérem L053R8.

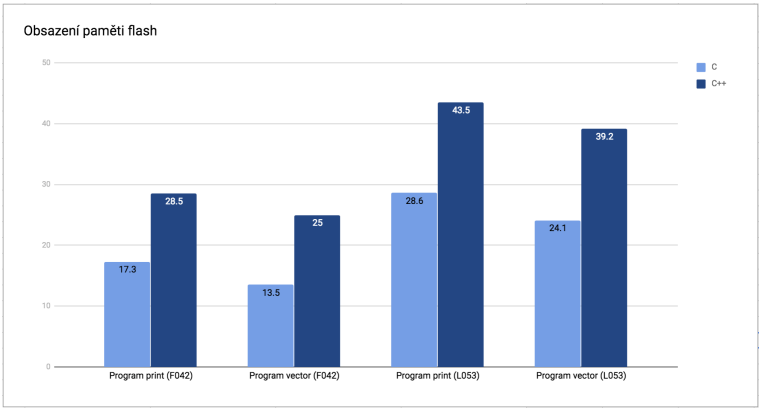
Mikrokontrolér L053R8 je oproti mikrokontroléru F042 vybaven dvojnásobnou velikostí paměti flash. I přes lepší parametry mikrokontroléru L053R8 programy využívající STL spotřebovaly velké množství paměti. Funkce C obvykle nazabírají více místa z důvodu úsporné implementace a zahrnutí v knihoně *mbed.h* Pro srovnání byly implementovány programy stejné funkce a zaznamenány změny v obsazení paměti programu.

První testovací program vykonává inicializaci řetězce znaků, ke kterému je vzápětí připojen další řetězec. Nakonec je celý řetězec vytisknut na sériový port mbed knihovnou. Další test se zaměřil na použití kontejneru *std::vector*. V tomto případě bylo do kontejneru přiřazeno několik čísel. Aby kompilér neoptimalizoval potřebnou část kódu, byly čísla použity pro nastavení délky prodlevy mezi blikáním LED. Následně se ukázalo, že kompilér se možností optimalice kódů nezabýval.

Program	**Print**	**Vector**	**Print**	**Vector**
MCU	**F042** [1]	**F042**	**L053** [2]	**L053**
C	17.3	13.5	28.6	24.1
C++	28.5	25.0	43.5	39.2
Rozdíl	11.2	11.5	14.9	15.1

[1] Maximální velikost programu 32KB
[2] Maximální velikost programu 64KB

Tabulka 7.4: Obsazenost paměti flash



Obrázek 7.4: Porovnání velikosti výsledného programu na základě implementace v c a c++

7.5 Funkce nad polem znaků

V jazyce C je běžné deklarovat pole znaků statické délky. S deklarovanou proměnnou poté pracujeme jako s ukazatelem na počátek pole. Z tohoto chování vyplývá, že není možné vracet lokálně deklarovaný řetězec. Pole by po návratu z funkce přestalo existovat a navrácený ukazatel by tudíž vedl na neexistující pole.

Pro generování řetězců je nutné znát funkci *printf* a několik základních funkcí z knihovny *string.h* jako například *strcpy* a *strcat*. Znalostí funkce *printf* je myšlena především práce s modifikátory formátovacího řetězce. Formátovací řetězec umožňuje formátovat znaky do bloku pevné délky, pro zajištění přepisu celého bloku.

7.6 Časté chyby při návrhu aplikace

Použitím nativní terminálové aplikace na sebe přenášíme riziko v podobě perzistence nastavení po ukončení sériové komunikace. Problém nastává hlavně při nenadálém ukončení aplikace, kdy nedochází k nastavení původních hodnot. Můžeme tak skončit s terminálem který má nastavenou stejnou barvu pro pozadí i text nebo neočekávanou znakovou sadu. Takto postižený terminál je možné vyčistit příkazem reset.

Aplikace terminálu používají různá nastavení pro barvy pozadí i barvu textu. Navíc může tyto nastavení měnit samotný uživatel. Z tohoto důvodu se musíme vyvarovat některým barvám textu a nastavit barvu pozadí na místech s důležitými informacemi.

Terminál	**Systém**	**Barva textu**	**Barva pozadí**
PuTTY	multiplatformní	Bílá	Černá
gnome-terminal	Ubuntu	Bílá	Fialová
Terminal.app	macOS	Černá	Bílá

Tabulka 7.5: Výchozí barva textu a pozadí terminálových aplikací

Množství uživatelů má použití tlačítka escape spojeno s možností opuštění části programu nebo jeho ukončení. Chování je odlišné od původních hardwarových terminálu kde se klávesa ESC využívala jako inicializační znak pro zadání ové sekvence. Použití stejného znaku pro stisk klávesy ESC a počáteční znak escape sekvencí neumožňuje interpretaci požadované akce. Nevíme zda uživatel stiskl klávesu ESC nebo se jedná o počátek příchozí escape sekvence.

Tento problém lze řešit požadavkem na opětovné stisknutí tlačítka ESC. Potřeba opětovného stisknutí ale nemusí být uživateli zřejmá. Další možností je počítat čas od počátku příchodu sekvence a v případě dlouhé prodlevy vyhodnotit znak jako stisk klávesy ESC.

Jak již bylo uvedeno dříve, používání přerušení pro tisk znaků na sériový port není doporučováno. Dalším důvodem, proč nepoužívat přerušenítímto

způsobem, je možnost narušení funkce, která právě zpracovává odeslání escape sekvence pro změny barev. I v případě ošetření návratu kurzoru na správnou polohu není možné zaručit navrácení identického nastavení jeho grafických atributů. Narážíme zde na problém, který vzniká při použití terminálu, který nepodporuje sekvenci pro uložení grafických atributů sekvencí.

Kapitola 8

Základní programy jako stavební kameny prototypů

Vzorem pro tvorbu následujících příkladů bylo využití IoT technologií v domovní automatizaci. Vývoj IoT je pro některé programátory atraktivní díky možnosti viditelného výstupu v podobě hmatatelného předmětu. Pro zvýšení motivace a snadnější testování je vhodné vyrobit zařízení, které napodobuje funkci plánovaného výrobku. Slovem prototyp je myšlen tento základní výrobek, který nesplňuje cílové parametry ale demonstruje funkci výstupního výrobku.

Zařízení často vykonávají jednoduchý úkon, který lze snadno napodobit s použitím periferií mikrokontroléru. Ukázkové programy popsané níže využívají řadu senzorů pro měření a digitální výstup pro ovládání připojeného zařízení. V případě prototypu zapojeného obvodu tak můžeme využít programy s uživatelským prostředím, které zpracovává výstup měření na vstupech mikrokontroléru. Dále mohou tyto programy přijímat příkazy uživatele a ovládat jimi digitální výstup.

Pro sběr dat senzorů a ovládání připojených součástí se využívá několik způsobů komunikace provozovaných periferiemi mikrokontroléru. Při využití AD převodníků lze měřenou hodnotu přenášet v podobě úrovně napětí. Dále lze komunikovat změnou binárního výstupu na pinu mikrokontroléru. Při nutnosti využití rychlé sériové komunikace se využívá komunikace prostřednictvím SPI nebo I2C. Některé senzory využívají pro přenos hodnot svůj protokol. Tato zařízení a senzory mají obvykle úzké zaměření, jejichž použití nelze dostatečně zobecnit a proto nejsou zahrnuty v této práci.

U programu Colored meter je mnohem pravděpodobnější využití i bez úprav programu. Z tohoto důvodu byl vstupní pin uveden přímo v běžící aplikaci. Pro všechny ukázky je využíván stejný pin *PA_0*. V případě dalších aktivních pinů vyběhne upozornění po startu mikrokontroléru.

8.1 Programy pro sběr dat

Pro sběr dat využijeme senzory měnící sledované veličiny na elektrické signály. Výstup senzoru bývá realizován analogovým nebo digitálním výstupem.

Základní senzory mají obvykle výstupní pin na který je přivedena naměřená veličina jako úroveň napětí. Mezi známé senzory založené na tomto principu patří například teplotní senzor TMP36 použitý v ukázce teploměru popsané níže.

Další metoda měření využívá změn odporu součástky. Senzor v podobě elektrické součástky pak zapojíme sériově do obvodu s rezistorem. Tímto vytvoříme napěťový dělič, který má na výstupu podobné vlastnosti jako senzor TMP36. Tímto příkladem se zabývá například zapojení fototranzistoru v příkladu popsaném níže.

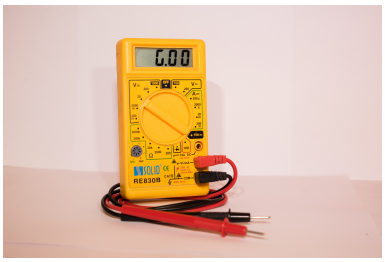
Některé senzory využívají digitální komunikační protokol. Může to být z důvodu přenosu většího množství informací při zachování jedinného výstupního pinu. Výhodou je také odolnost proti rušení v případě delší vzdálenosti senzoru od řídící jednotky. Tímto případem se v této práci nebudeme zabývat z důvodu úzkého zaměření a protokolu specifického pro daný senzor. Jako příklad takového senzoru lze uvést DHT11[18], měřící teplotu a vlhkost vzduchu. Data lze zpracovat hotovou knihovnou, které má množství implementací volně dostupných v online mbed prostředí.

8.2 Měření analogového signálu

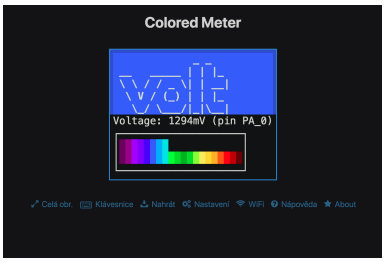
Některé senzory nejsou přizpůsobeny pro měření mikrokontroléry pracujících s napěťovou úrovní 3.3V. Výstupní signál tohoto druhu senzorů proto upravíme na měřitelné rozsahy napěťovým děličem. V některých případech se počítá s použitím pull-up nebo pull-down rezistoru. Pro eliminaci naměřených extrémních hodnot je vhodné přidávat keramické kondenzátory k napájení a výstupů senzorů.

Při sestavování analogových obvodů je pro kontrolu funkce potřeba na některých částech měřit úroveň napětí. Pro tuto úlohu můžeme použít měřící zařízení voltmetr. V praxi se běžně využívá multimetr, přístroj obsahující větší množství funkcí. Multimetr je dostupným běžně používaným zařízením.

Použití běžného multimetru v obvodech obsahujících mikrokontrolér není optimální. Tato digitální měřící zařízení zobrazují naměřená data na sedmisegmentový displej. Tento displej mívá omezený počet segmentů, na kterých jsou informace zobrazovány. Zobrazení omezeného počtu číslic je kompenzováno možností změny rozsahu pro měření napětí. Standardní rozsahy jsou rozmístěny v úrovních měření napětí do 200mV, 2000mV, 20V.



(a) : Běžný multimetr



(b) : Terminálová aplikace

Obrázek 8.1: Porovnání hardwarového a softwarového voltmetru

Bezpečně můžeme měřit pouze v módu měřícím do 20V, který nám pro obvod fungující v rozsahu 0 až 3300mV poskytuje pouze 330 hodnot. Rozlišení v tomto rozsahu je u levnějších přístrojů 10mV[19] s přesností plus mínus jedno procento. V případě použití mikrokontroléru můžeme provést měření s vyšším rozlišením a využít také možnosti zpracování naměřených dat.

DC napětí Rozsah	Rozlišení	Přesnost
200mV	0,1mV	± 1,0
2000mV	1mV	± 1,0
20V	10mV	± 1,0
200V	100mV	± 1,0
500V	1V	± 1,2

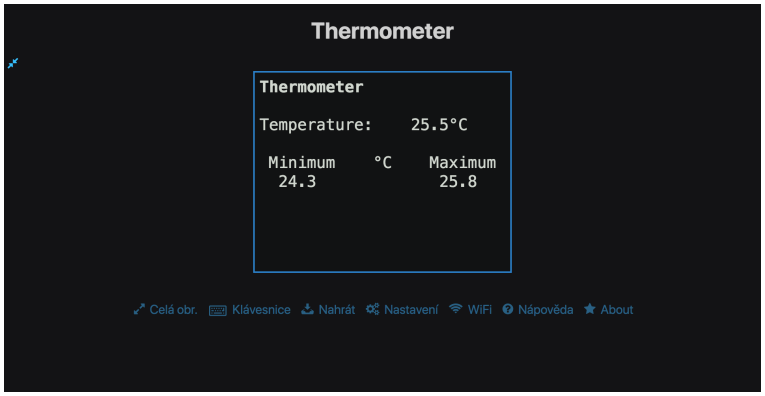
Tabulka 8.1: Ukázka parametrů multimetru nižší cenové třídy

Množství lidí využívá zobrazení měřených hodnot napětí na připojeném osobním počítači. Použití mikrokontroléru s podporou USB přímo nabízí takovou funkci například při natáčení edukačních materiálů. S možností vzdáleného zobrazení se přístroj dostává svými schopnostmi mezi měřící přístroje mnohem vyšší cenové kategorie. Zařízení dostupné na trhu navíc často neposkytují multiplatformní software pro nadřazený systém.

Převod úrovní analogového signálu na signál digitální probíhá analogově digitálním převodníkem. Používaný mikrokontrolér převádí naměřenou hodnotu pomocí 12-bitového převodníku[20]. Převodník je tedy schopen rozlišit 4096 úrovní napětí v rozmezí 0V až do 3.3V napětí napájecího. Rozlišení jednoho kroku při měření je tedy $3300/4096 = 0.8056$mV mezi jednotlivými úrovněmi.

8.2.1 Ukázka programu teploměr

V budovách můžeme měřit množství veličin a dějů senzory produkujícími analogový signál. Tento signál může být poté převeden analogově digitálním převodníkem na digitální hodnotu. Přepočet naměřené hodnoty je pak často usnadněn využitím lineární závislosti sledované veličiny na výstupním napětí. Typickým příkladem je měření teploty senzorem TMP36[21].

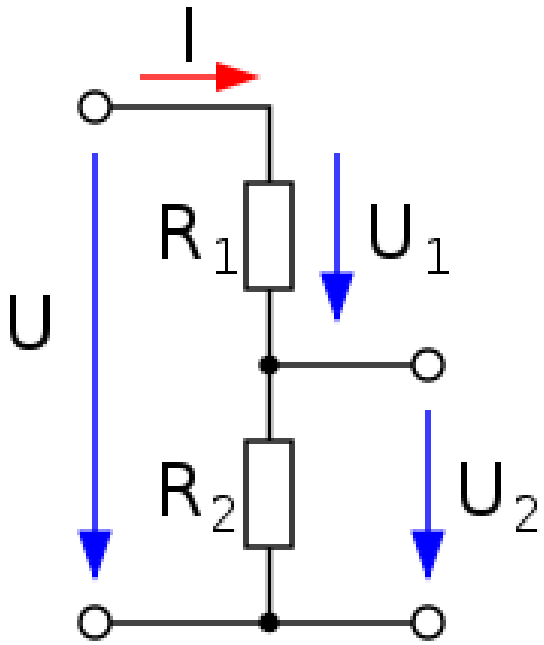


Obrázek 8.2: Ukázka uživatelského prostředí programu Teploměr

Aplikace využívá třídu knihovny mbed AnalogIn, která umožňuje měření na pinu podporující analogový vstup. Výstup v podobě desetinného čísla v rozmezí 0.0f až 1.0f je přepočítáno podle vzorce z dokumentace použitého senzoru. Zmíněný senzor měří teploty v rozmezí -40°C až 125°C s přesností ± 2°C[21].

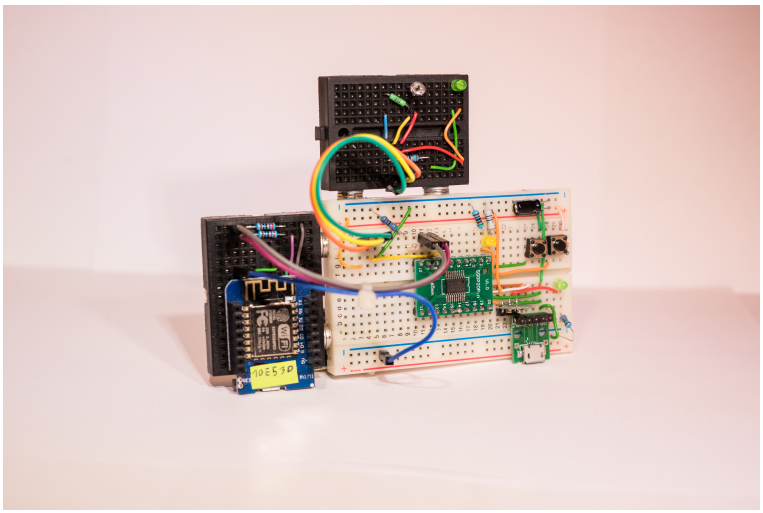
8.2.2 Zapojení součástky fototranzistor

V budovách můžeme měřit například intenzitu osvětlení. Měření funguje na podobném principu jako předchozí ukázka. Součástka fototranzistor propouští proud v závislosti intenzity dopadajícího záření. Zapojením do sériového obvodu s rezistorem lze docílit převod intenzity osvětlení na výstupní napětí.



Obrázek 8.3: Zapojení fototranzistoru jako napěťového děliče[Bie]

Výstupní napětí je poté možné měřit a převést na intenzitu osvětlení. Z důvodu neznámých parametrů součástky a nizkých nároků na měření intenzity lze použít univerzálního měření voltmetrem popsaným níže. Voltmetr implementovaný v této práci obsahuje grafické zobrazení hladiny napětí, které odpovídá intenzitě osvětlení. Zapojení také využívá stejnou třídu pro měření analogového signálu a proto tato ukázka nemá svou implementaci aplikace.

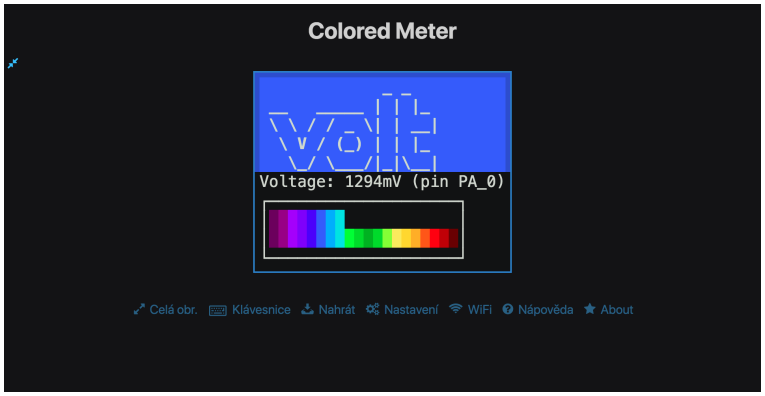


Obrázek 8.4: Zapojení fototranzistoru v kontaktním poli

8.2.3 Pseudografický voltmeter

V některých případech je programování aplikace se specifickým zaměřením nerentabilní. Z důvodu častého využití analogového vstupu mikrokontroléru při měření, byla pro tyto účely vytvořena aplikace Colored Meter měřící vstupní napětí. Naměřená úroveň napětí je zobrazena barevným pruhovým indikátorem spolu s aktuální hodnotou v podobě textu.

Podobná zařízení již existují v podobě obvodu využívajícího barevných LED. Využitím barev přístroj umožňuje nenáročnou signalizaci stavu vyžadujícího zvýšenou pozornost. Výhodou použití mikrokontroléru je možnost následného zpracování dat nebo vzdáleného přístupu při použití komunikačního modulu.



Obrázek 8.5: Ukázka pseudografického voltmetru

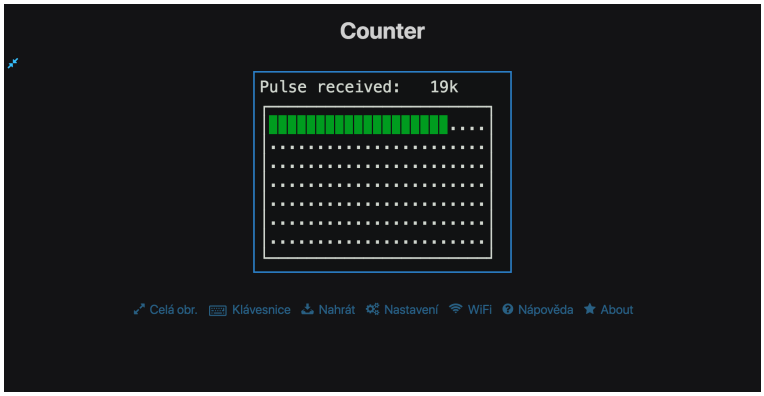
Ukázka může měřit množství zakladních senzorů pro měření teploty, vlhkosti nebo například Hallovy sondy. Barevné zobrazení dělí měřený rozsah na 20 barevných dílků se zobrazením podle aktuální měřené hodnoty. Pro vyšší citlivost byl implementován druhý řádek, který zobrazuje změnu napětí v rozmezí jednoho barevného proužku hlavního řádku. Barvy lze jednoduše měnit v konfiguračním souboru projektu, kde se nachází pole obsahující barvy pro každou pozici. Volba výchozích barev byla provedena přepočtem vzorků z obrázku duhy na paletu 256 barev.

8.3 Měření digitálního signálu

Využití digitálního vstupu můžeme využít například v bezpečnosti. Dále například pro diagnostiku, kde není nutný fyzický přístup k diagnostikovanému zařízení.

8.3.1 Ukázka programu Počítadlo

Ukázka počítadla využívá zachycení událostí, které se projevují změnou napětí, tzv. náběžnou hranou na pinu digitálního vstupu. Využití můžeme najít například při počítání počtu průchodu osob, otáček motoru či impulsů na výstupním pinu zařízení.

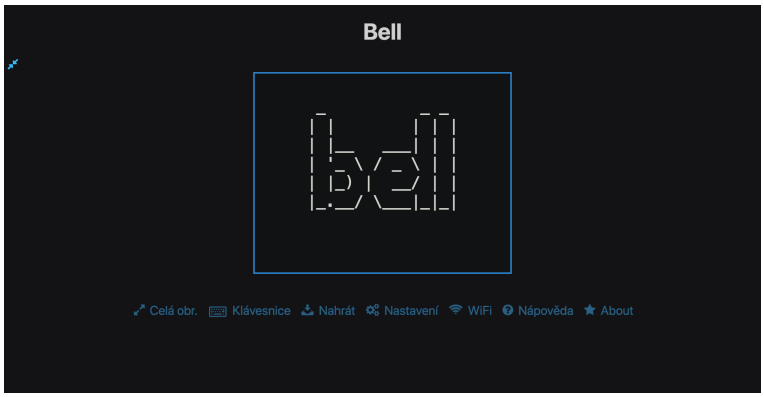


Obrázek 8.6: Ukázka aplikace počítající počet impulsů

Zapojení a aplikace využívají funkci pro obsluhu přerušení poskytovaného třídou InterruptIn. Frekvenci můžeme měřit pouze přístrojem pracujícím s minimálně dvojnásobnou vzorkovací frekvencí[17]. S použitím knihovny mbed jsme navíc omezeni využitím složitější obsluhy přerušení, které přináší režijní náklady v podobě vyššího množství vykonaných instrukcí. Pro naše příklady je ale rychlost v řádech desítek kHz dostatečná.

8.3.2 Ukázka programu Zvonek

Tato aplikace realizuje klasicky domovní zvonek. Stisknutí tlačítka vyšle do terminálu escape sekvenci, která volá kontrolní funkci BEL. Tato funkce umožňuje vyvolat zvuk aplikace terminálu na připojeném výstupním zařízení. Zvuk se liší v závislosti na použitém emulátoru terminálu. Funkce BEL patří mezi široce podporovanou funkci umožňující upozornit uživatele pomocí dalšího smyslu.



Obrázek 8.7: Ukázka aplikace s funkcí domovního zvonku

Tato ukázka také umožňuje v nastavení zadat IP adresu databázové aplikace. Tímto může být zvonek přeměněn na senzor s ukládáním dat. Jednoduchou výměnou tlačítka za fototranzistor bylo zařízení například použito pro zaznamenávání impulzů měřiče spotřebované energie. Zaznamenáním impulsu a

uložením s příslušným časem může poskytovat zajímavé statistiky odběru elektrického proudu.

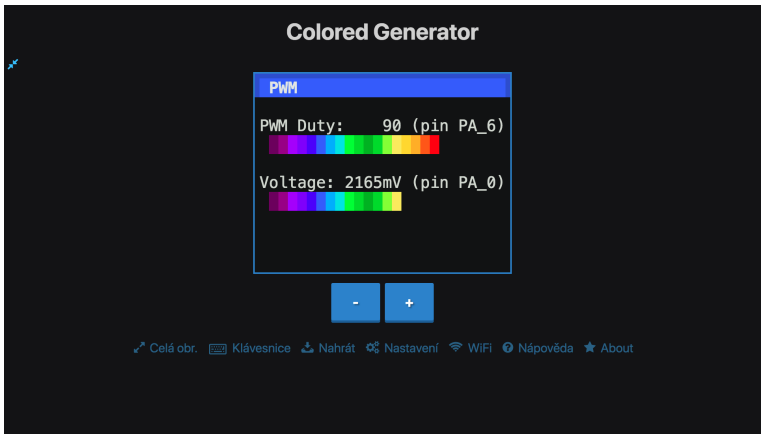
8.4 Programy ovládající připojenou komponentu

Ovládání analogového obvodu může být realizováno generováním napěťové úrovně podle zadané digitální hodnoty. O tuto funkci se stará obvod nazývaný digitálně analogový převodník (DAC). Tento obvod není často využívanou výbavou mikrokontrolérů. Použití analogového signálu má jen omezenou množinu využití. V případě nutnosti použití se často využívá externího modulu, který umožňuje použití záporného napětí. Z tohoto důvodu bývá počet DAC převodníků omezen.

Mikrokontrolér používaný v této práci neobsahuje žádný DAC převodník. Generování napěťové úrovně může být realizováno s použitím elektrických součástek a pulzně šířkovou modulací, neboli PWM.

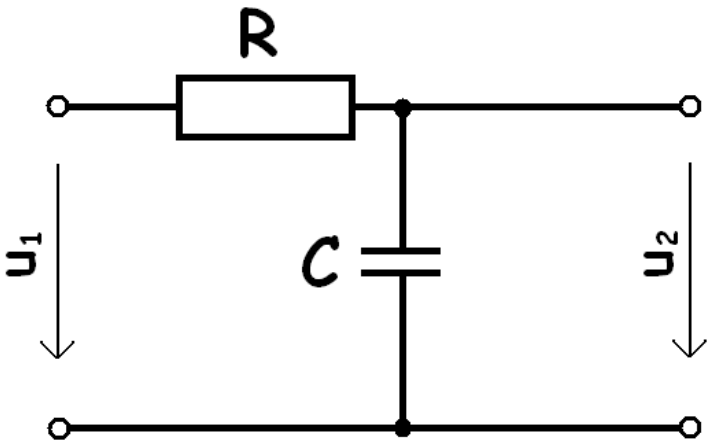
8.4.1 Pseudografický PWM generátor

Sdílením pseudografického prostředí aplikace Colored meter popsané výše vedlo k vytvoření PWM generátoru. Střídu generovaného signálu je možné ovládat zadanými příkazy. Příkazy jsou zasílány znaky '+' a '-' odeslanými do okna terminálu nebo použitím dvou tlačítek v případě terminálu ESPTerm. Generátor zobrazuje aktuální střídu v procentech a zároveň i v textové podobě. Součástí aplikace je voltmetr s průměrováním měřených hodnot.



Obrázek 8.8: Ukázka aplikace pro nastavitelnou střídu PWM

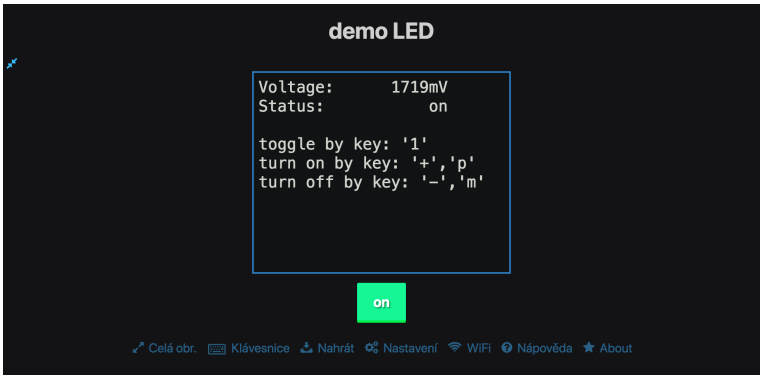
Tohoto generátoru PWM lze také využít jako náhradu za DA převodník. Výstup PWM lze spojit s obvodem dolní propusti a úpravou střídy regulovat výstupní napětí.



Obrázek 8.9: Schéma dolní propusti[Kuk]

8.4.2 Program ovládající logickou úroveň na výstupním pinu

Základním případem vzdáleného ovládání je změna binárního stavu (vypnuto/zapnuto) parametru zařízení. Jednoduché zařízení podporující tuto funkci umožňuje reakci na nejrůznější scénáře. Například běžná lampička, ventilátor nebo okenní rolety mohou zvýšit komfort při používání pouhým spojením s výstupem chytré televize. Chytré televize nabízí tzv. “kino mód” který mohou ostatní zařízení využít jako výzvu ke změně svého stavu.



Obrázek 8.10: Ukázka aplikace umožňující nastavení logické úrovně digitálního pinu mikrokontroléru

Tímto případem se zabývá demo měnící stav připojené LED. Součástka ale může být nahrazena množstvím periferií využívajících binární signalizaci pro kontrolu svého běhu. Aplikace přijímá vstup několika znaků (např +, -, 1), které ovládají výstup řídící jednotky. Funkce jednotlivých znaků je vysvětlena pod výpisem aktuálního stavu a je pravidelně přepisována pro zajištění zobrazení při výpadku komunikačního modulu.

Kapitola 9

WiFi Modul ESP8266 a bezdrátová komunikace

9.1 Systém na čipu ESP8266

Systém na čipu ESP8266 poskytuje komunikaci standardní 2.4 GHz Wi-Fi (802.11b/g/n, s podporou WPA/WPA2) a integraci TCP/IP protokolu. ESP8266 obsahuje integrovaný mikrokontrolér, který v základu umožňuje komunikaci prostřednictvím základních AT příkazů. Hlavní výhodou použití ESP8266 je možnost upgradu firmware vnitřního mikrokontroléru. Moduly obsahují nadstandardní paměť FLASH a RAM. Dále je možné připojit externí QSPI flash paměť, čehož bývá u hotových modulů využíváno pro připojení paměti velikosti 4MiB.

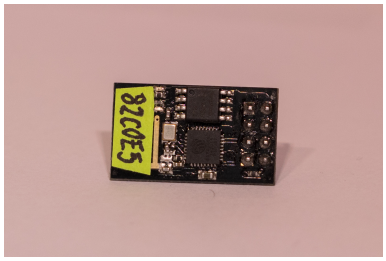
9.2 Vyráběné varianty

S nástupem popularity odvětví IoT narůstala také poptávka po možnostech bezdrátové komunikace. Díky velkému množství výbavy a malé pořizovací ceně proto moduly obsahující ESP8266 rychle získaly na popularitě. Moduly jsou dodávány různými výrobci, kteří poskytují různá vylepšení při zachování kompatibility. Při volbě modulu se zaměříme převážně na počet výstupních pinů a umístitelnost na nepájivá kontaktní pole. Vybrat si můžeme například moduly s možností přídavné antény nebo větší velikostí paměti flash. Možnost připojení externí paměti bývá využíváno pro uložení velkých souborů potřebných pro provoz webového serveru.

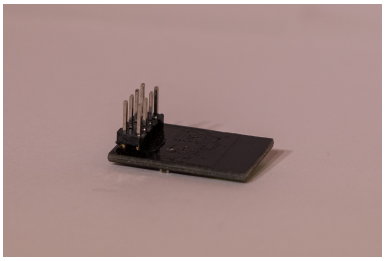
9.2.1 Parametry modulu ESP-01

Verze ESP-01 má jen malý počet výstupních pinů, které navíc nejsou polohou uspořádány pro zapojení do kontaktního pole. Tento modul nachází uplatnění při výrobě většího množství zařízení, která potřebují pouze výstupy pro sériovou komunikaci. Modul ESP-01 je jeden z nejlevnějších modulů využívajícího čipu ESP8266. Tento modul ale není vybaven vlastním stabilizátorem napětí,

jehož pořízení může výslednou cenu navýšit. Použití externího stabilizátoru napětí je také nevýhodné při zapojení na kontaktních polích. Menší napětí při zachování stejného vysílacího výkonu znamená vyšší proudy protékající kontakty pole.



(a) : ESP-01 modul



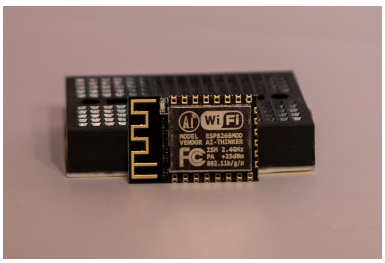
(b) : ESP-01 piny nekompatibilní s kontaktním polem

Obrázek 9.1: Modul ESP-01 foto desky plošného spoje

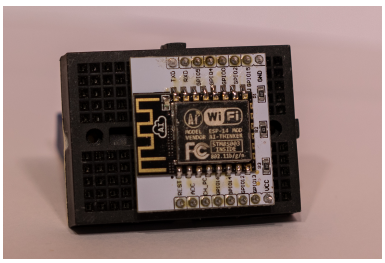
Dalším problémem, který se vyskytl při testování vlastního firmware, byla nemožnost využití pinů jako digitálních výstupu pro ovládání relé. Při startu modulu není možné zajistit logickou hodnotu do určité úrovně. Piny modulu GPIO jsou využívány pro nastavení módu[Webb] při startu modulu a nejsou proto vhodné pro jiné využití. Digitální výstup ovládající například relé způsobí kmitání, které nelze napravit použitím pull rezistoru.

9.2.2 Parametry modulu ESP-14

Mnohem výhodnějším se oproti modulu ESP-01 ukázalo použití modulu ESP-14[Weba]. Modul je nejčastěji vybaven externí pamětí 4MiB. V základním provedení nelze připojit do kontaktního pole, protože využívá jinou rozteč pinů. Existuje ale množství expandujících plošných spojů, které tento problém řeší.



(a) : ESP-14 modul

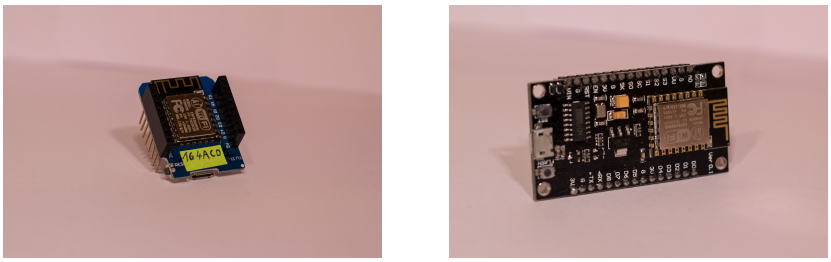


(b) : ESP-14 expandér pro nepájivá kontaktní pole

Obrázek 9.2: Modul se stínícím krytem ESP-14

Pro použití modulu při tvorbě prototypů zařízenív nepájivém kontaktním poli se osvědčilo použití expandujicího modulu WEMOS D1 mini. Modul

poskytuje pin pro napájení 5V se zabudovaným stabilizátorem, převodník USB-Serial a dostatečné množství GPIO. Modul zároveň nezabírá příliš místa, které můžeme využít pro zapojení dalších obvodů. Další možností je využití mnohem většího vývojového kitu NodeMCU, který je pro běžná pole příliš široký. Modul je možné zapojit do pole s omezením v podobě nutnosti vedení propojů pod vývojovým kitem. Při úpravě propojů je nutné vyjmutí modulu, čímž je kontaktní pole opotřebováváno.



(a) : ESP modul WEMOS D1 mini **(b) :** ESP modul NodeMCU

Obrázek 9.3: Moduly ESP8266 s možností zapojení do kontaktního pole

Možnosti použití SoC ESP8266 v této práci jsou omezeny nedostatečným počtem převodníků. Somotný čip obsahuje pouze jedinný 10-bitový analogově digitální převodník. Rozlišení 1024 úrovní a rozmezí vstupního napětí 0-1V není vhodné v prostředí využívajícího napětí 3.3V. Dále je potřeba aktivní obsluhy běhu WIFI, která může přerušit probíhající měření. Z těchto důvodů byl modul použit převážně pro zprostředkování komunikace s měřícím blokem.

Při výrobě je do ESP8266 nahrán firmware používající příkazy AT. Tento firmware je určen pro navázání spojení a komunikaci se vzdáleným serverem. Pro příklady řešené v této práci je dodávaný firmware příliš omezující. Firmware je určen pro komunikaci na nižších vrstvách TCP/IP a pro jeho využití by bylo zapotřebí implementovat aplikaci pro vzdálený server. V této práci se zabýváme přístupu pro libovolnou platformu a proto je využito webového serveru. Pro běh webového serveru proto využijeme alternativní firmware ESPterm.

9.3 Webový emulátor terminálu ESPTerm

Bezdrátový přenos sériové komunikace by vyžadoval instalaci aplikace, která by přenášená data zprostředkovala pro terminálové aplikace. Tento problém vyřešil student ČVUT na katedře měření svým projektem ESPTerm. Tento projekt umožňuje běh terminálu bez nutnosti instalace aplikace a to bezdrátově prostřednictvím WiFi. Využití prohlížeče, který je zakladní výbavou operačního systému, pak umožňuje pohodlný běh textových aplikací.

Emulator terminálu ESPTerm využívá vnitřního mikrokontroléru v modulu ESP8266 pro běh webového serveru. V prostředí webové stránky ESPTerm emuluje plnohodnotnou verzi terminálu. Uživatel má možnost komfortního

ovládání a využít může přednastavená makra pro stisk několika tlačítek. Zdrojem dat pro terminál je UART rozhraní na pinech TXD a RXD mikrokontroléru ESP8266.

Využití WiFi pro komunikaci s ESPTerm lze využít také jako náhradu serial-usb převodníku. Tato možnost je výhodná pro zařízení, která nejsou vybavena USB portem. Dále lze také obejít instalaci terminálové aplikace v systému, který aplikaci neobsahuje. Zařízení lze tímto způsobem elektricky izolovat a předejít tak možnosti jeho poškození. Elektrická izolace je vhodná hlavně při sestavování obvodů pracujících s napětím vyšším než 5.25V povolených pro standard USB.

Navzdory bezdrátové komunikaci lze zařízení ovládat bez ztráty bezpečí. Wifi protokol umožňuje šifrovat přenášená data a přístup lze omezit pouze pro autorizovaného uživatele.

9.4 Nástroj pro firmware upgrade esptool

Pro nahrání programu do modulu ESP je možné použít multiplatformní nástroj esptool[24]. Tento nástroj umožňuje nahrát binární soubory firmwaru. Binární soubory pro firmware ESPTerm lze získat na veřejném repozitáři projeku[25]. Nahrávání projektu je podobné pro všechny platformy a postup byl již zpracován (najit link) a proto není dále rozveden v této práci.

9.5 Realizace firmware domácí automatizace

V případě malých nároků na měření nebo ovládání můžeme vynechat zapojení řídící jednotky a využít samotného komunikačního modulu. Vývoj pro ESP8266 byl testován pouze v systému Ubuntu a nástrojích dostupných pro tento systém. Zpočátku byl pro programování zvolen doporučovaný toolchain[26] se všemi potřebnými nástroji. Později bylo otestováno také vývojové prostředí Arduino, které umožňuje instalaci rozšíření podporujicího modul s ESP8266. Prostředí Arduino není doporučováno, protože instalované rozšíření má problémy s kompatibilitou při použití některých verzí Arduino IDE. Knihovny poskytované tímto zdrojem neposkytují zpětnou kompatibilitu při aktualizacích ani varování o ukončení podpory funkcí.

Testovací aplikace je rozšířením pro chytrou domácnost HomeKit společnosti Apple při využití připojení k emulátoru API Homebridge. Aplikace využívá možnosti dotazu nad HTTP protokolem a metody GET pro získání informací o požadovaném stavu ve formátu JSON. Příchozí odpověď je použita pro změnu stavu digitálního výstupu na pinu D8 modulu WEMOS D1 mini. Modul ESP je připojen k modulu ovládajícímu obvod sledovaného objektu.

9.6 Cenová dostupnost modulu

Jako demonstrace možnosti využití modulu bylo přetvoření obyčejné lampy pro pěstování v dálkově ovládanou lampu s napojením na IoT systém Apple HomeKit. Zařízení řešící tento problém v podobě chytré zásuvky je možné zakoupit v množství obchodů. Hotové řešení ale cenou mnohonásobně převyšuje cenu objektu určeného k úpravě. S použitím modulu ESP8266 bylo možné snížit výdaje na desetinu ceny sériově vyráběného zařízení konkurence.

9.7 Komunikace zařízení v síti internet

Komunikační modul s firmware ESPTerm umožňuje několik variant přístupu. V základním nastavení se lze připojit prostřednictvím vytvořeného přístupového bodu. Při připojení touto cestou modul uživatele automaticky přesměruje na adresu webového serveru běžícího na tomto modulu. Další možností je připojení modulu do již existující WiFi sítě. Druhý způsob je doporučován pro zlepšení odezvy emulace terminálu.

Viditelnost webového serveru je standardně omezená překladem adres IPv4 na lokální síť. Pro zviditelnění zařízení mimo lokální síť je nutné zasáhnout do nastavení routeru a povolit překládání portů na IP komunikačního modulu. Zviditelnit tak lze pouze jediné zařízení na daném portu. V případě omezeného přístupu k nastavení síťové infrastruktury tedy nelze webový server komunikačního modulu zviditelnit. Problémy mohou také nastat při dynamické změně IP adresy. V tomto případě je potřeba zjišťovat aktuální adresu na kterou se budeme připojovat.

Přestože nelze naše zařízení zviditelnit v síti internet, stále máme v této síti možnost data zasílat a přijímat. Využít lze instalace serveru na zařízení disponující veřejnou adresou. Na tento server se lze poté připojit z vnitřní sítě. Zasílat tak můžeme požadavky s naměřenými daty nebo naopak získat informace poskytované prostřednictvím API dotazovaného serveru.

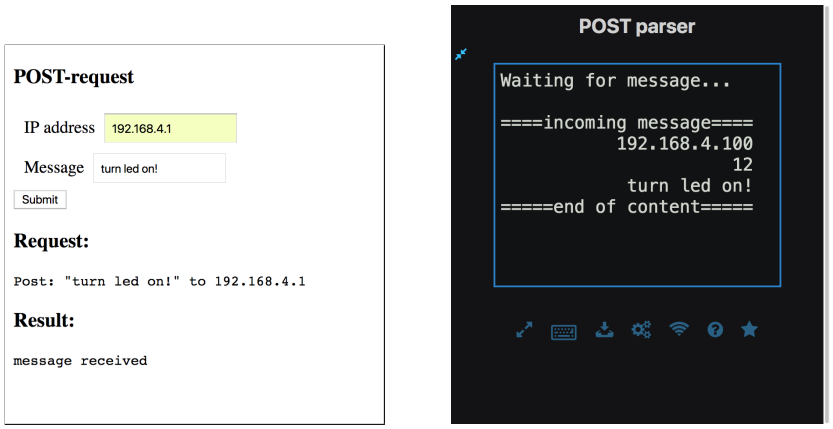
Použít lze také routeru s podporou VPN ale nastavení takového systému vyžaduje instalaci VPN serveru na zařízení s veřejnou adresou. Tento postup byl testován ale realizace se nezdařila kvůli nekompatibilitě routeru a VPN software na serveru.

Kapitola 10

Sběr dat v nadřazeném systému

10.1 Realizace aplikace pro sběr dat

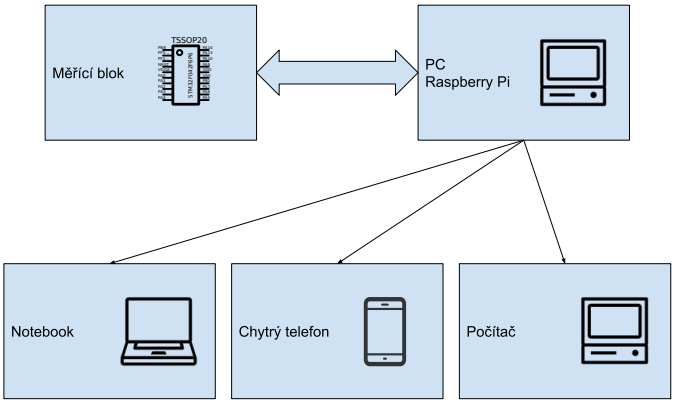
Firmware ESPTerm umožňuje vysílat zprávy do kompatibilních modulů na síti WiFi. Této funkcionality bylo využito při návrhu aplikace běžící na PC. Navržená aplikace umožňuje obousměrnou komunikaci s těmito moduly. Komunikační modul předává zprávy prostřednictvím protokolu HTTP. Tento protokol obsahuje množství metod umožňujících zasílání zpráv mezi webovými aplikacemi. ESPTerm pro komunikaci s dalším modulem využívá metody POST.



(a) : Webová aplikace **(b) :** Příjem zprávy na MCU

Obrázek 10.1: Komunikace webové aplikace s měřící jednotkou

Aplikace byla implementována v jazyce JavaScript. Tento jazyk patří mezi nejpopulárnější jazyky[27] používané na vývojářské platformě GitHub. Nevýhodou použití tohoto jazyka je závislost aplikace na běhovém prostředí a instalaci balíčkovacího systému npm. Aplikace umožňuje vytvoření webserveru a následnou možnost sdílení přístupu pro další zařízení v síti. Aplikace tedy umožňuje běh na jediném zařízení a připojení dalších s využitím webového prohlížeče podobně jako u ESP s aplikací ESPTerm.

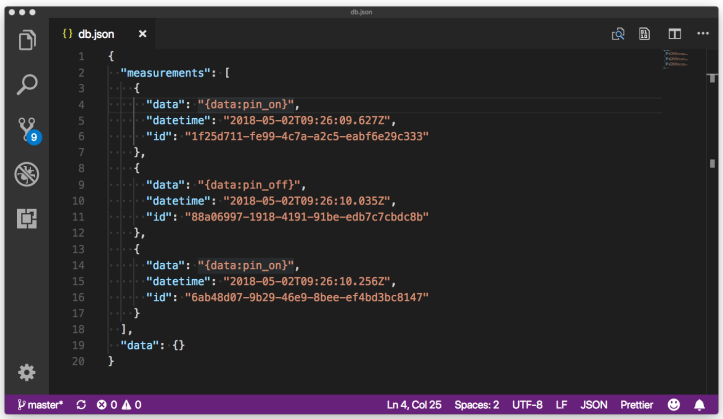


Obrázek 10.2: Sdílení dat v síti prostřednictvím webové aplikace[Des16][Fre]

10.2 Použití databáze

Mikrokontroléry nemají vlastní souborový systém a uložiště dat určené pro ukládání naměřených informací. S bezdrátovou komunikací můžeme zasílat data do aplikace běžící na serveru a ukládat tato data do databáze. Server uchovávající data není ovlivněn při výpadku napájení. Dále umožňuje zpracování velkého množství dat přesahující volnou paměť RAM mikrokontroléru.

Pro přenositelnost a jednoduchost implementace bylo využito databáze spravované knihovnou Lowdb. Tato knihovna umožňuje uložení dat do souboru. Ukládání dat probíhá ve formátu JSON, který je běžně podporován napříč platformami. Soubor je tedy možné zpracovat aplikací implementovanou v libovolném jazyce.



Obrázek 10.3: Zprávy z měřící jednotky v databázi formátu JSON

Ukázka dále upozorňuje na možnost řešení výpočetně složitých úloh na připojeném zařízení. Takové zařízení potom může sloužit jako centrální jednotka pro zpracování datově náročných úloh. To se vyplatí především při použití bezdrátové komunikace s modulem poháněným baterií. Místo dotazů mířených na takový modul můžeme využít centrálního bodu sloužícího jako uložiště naměřených hodnot a zvýšit tak dobu, po kterou může zařízení využívat baterii.

V případě potřeby zaslání jednoduché zprávy není zapotřebí vytváření webserveru. Zasílání požadavků s použitím HTTP protokolu je pro množství programátorů známou aktivitou. Pro manuální zaslání zprávy obsahující text určený pro ESPTerm firmware lze proto využít nástroje curl. Nástroj curl slouží k zasílání zpráv nad různými protokoly, mezi které patří právě zmiňovaný protokol HTTP. Nástroj je standardní výbavou systému Ubuntu i macOS. Operační systém Windows tento nástroj v základu neobsahuje. uživatel tohoto systému má ale možnost stažení volně dostupnou přenosnou (portable) verzi nástroje pro tento systém.

Kapitola 11

Závěr

Cíl realizovat systém umožňující měřit a ovládat sledovaný objekt byl implementován aplikacemi bez úzkého zaměření. Základní programy slouží jako kostra pro implementaci pokročilejších zařízení se specifickým zaměřením. Při realizaci bylo dbáno na možnost zobrazit výstup zařízení na libovolném operačním systému.

Při návrhu aplikací bylo využito možnosti sériové komunikace s využitím terminálu. Použití terminálu umožňuje zobrazení pseudografického výstupu na libovolném zařízení. V případě použití modulu ESP8266 lze využít připojení prostřednictvím WiFi a emulace terminálu v okně webového prohlížeče. Tímto je možné využít vzdáleného přístupu na většině zařízení obsahujících prohlížeč, který bývá základní výbavou operačního systému.

Práce se také zabývá možností zasílání zpráv mezi mikrokontrolérem a aplikací implementovanou pro architektury osobních počítačů. Zprávy je možné zasílat nad protokolem HTTP s využidím metody POST. Pro zaslání zprávy prostřednictvím tohoto protokolu lze využít nástroje curl nebo aplikace implementované v této práci. Implementovaná aplikace využívá webserver pro možnost zaslání uživatelem definované zprávy. Dále tato aplikace umožňuje přijímat a ukládat zprávy generované aplikací běžící na mikrokontroléru. Na straně mikrokontroléru byla implementována knihovna pro zpracování příchozích zpráv.

Při zpracování bodů zadání se ukázalo, že pro pohodlnou práci s výstupem na terminál by bylo vhodné implementovat knihovnu pro interpretaci příchozích sekvencí. Dále se ukázaly problémy při použití prostředí mbed pro programování terminálových aplikací. Implementace knihovny pro generování escape sekvencí řídících terminál obsahuje velké množství funkcí. Velké množství funkcí při chybějícím automatickém doplňování (našeptávání) snižuje komfort užívání knihovny.

Jako pokračování práce by bylo vhodné implementovat knihovnu pro základní terminály bez podpory escape sekvencí. Dále by bylo vhodné využít možnosti ukládáni dat a jejich zpracovaní tvorbou grafů a statistik.

Příloha A

Pojmy a zkratky

Zkratka	Význam anglicky	Význam česky
HW	Hardware	Hardware
OOP	Object-oriented programming	Objektově orientované programování
VCP	Virtual COM Port	Virtuální COM Port
VPN	Virtual private network	Virtuální privátní síť
MCU	Microcontroller Unit	Jednočipový počítač
OS	Operation System	Operační systém
IoT	Internet of Things	Internet věcí
ADC	Analog-to-digital converter	Analogově digitální převodník
DAC	digital-to-analog converter	Digitálně analogový převodník
DAQ	Data acquisition	Získávání dat

Příloha B

Literatura

[01] *STM32 32-bit ARM Cortex MCUs*, http://www.st.com/en/microcontrollers/stm32-32-bit-arm-cortex-mcus.html, Navštíveno 21.5.2018.

[02] *Wikipedia - USB*, http://en.wikipedia.org/wiki/USB, Navštíveno 21.5.2018.

[03] *Datasheet STM32F042x4 STM32F042x6*, http://www.st.com/content/ccc/resource/technical/document/datasheet/52/ad/d0/80/e6/be/40/ad/DM00105814.pdf/files/DM00105814.pdf/jcr:content/translations/en.DM00105814.pdf, Navštíveno 21.5.2018, pp. 31/117–34/117.

[04] *Datasheet STM32F042x4 STM32F042x6*, http://www.st.com/content/ccc/resource/technical/document/datasheet/52/ad/d0/80/e6/be/40/ad/DM00105814.pdf/files/DM00105814.pdf/jcr:content/translations/en.DM00105814.pdf, Navštíveno 21.5.2018, pp. 45–46.

[05] *D1 mini - A mini wifi board with 4MB flash based on ESP-8266EX*, https://wiki.wemos.cc/products:d1:d1_mini, Navštíveno 21.5.2018.

[06] *DataSheet USB to serial chip CH340*, https://cdn.sparkfun.com/datasheets/Dev/Arduino/Other/CH340DS1.PDF, Navštíveno 21.5.2018.

[07] *Online IDE a kompilátor mbed*, https://os.mbed.com/compiler/, Navštíveno 21.5.2018.

[08] *Vývojové kity platformy mbed*, https://os.mbed.com/platforms/, Navštíveno 21.5.2018.

[09] *Softwarová architektura Model-View-Controller (MVC)*, https://cs.wikipedia.org/wiki/Model-view-controller, Navštíveno 21.5.2018.

[10] *ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32*, http://www.st.com/content/ccc/resource/technical/document/data_brief/80/76/2b/dc/45/c5/46/90/DM00027105.pdf/files/DM00027105.pdf/jcr:content/translations/en.DM00027105.pdf, Navštíveno 22.5.2018.

[11] *USB Device Firmware Upgrade tool dfu-util*, https://admin.fedoraproject.org/pkgdb/package/rpms/dfu-util/, Navštíveno 22.5.2018.

[13a] *LPE - Laboratoře z průmyslové elektroniky a senzorů*, https://embedded.fel.cvut.cz/kurzy/lpe_sw, Navštíveno 21.5.2018.

[13b] *Ovladače pro Virtual COM Port CP210x USB-UART převodník*, https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers, Navštíveno 21.5.2018.

[16] *List of all control sequences of xterm, which ESPTerm uses*, https://espterm.github.io/docs/espterm-xterm.html, Navštíveno 22.5.2018.

[17] *Shannonův teorém*, https://cs.wikipedia.org/wiki/Shannon%C5%AFv_teor%C3%A9m, Navštíveno 21.5.2018.

[18] *AOSONG Temperature and humidity module DHT11 Product Manual*, https://akizukidenshi.com/download/ds/aosong/DHT11.pdf, Navštíveno 21.5.2018.

[19] *Digitální multimetr SOLIGHT V14*, www.shop.solight.cz/documents/v14_cz.pdf, Navštíveno 21.5.2018.

[20] *RM0091 Reference manual STM32F0x2 advanced ARM®-based 32-bit MCUs*, http://www.ece.uvic.ca/~ceng355/lab/supplement/stm32f0RefManual.pdf, Navštíveno 21.5.2018.

[21] *Low Voltage Temperature Sensors MP35/TMP36/TMP37*, https://cdn.sparkfun.com/datasheets/Sensors/Temp/TMP35_36_37.pdf, Navštíveno 23.5.2018.

[24] *Nástroj pro firmware upgrade esptool*, https://github.com/espressif/esptool, Navštíveno 21.5.2018.

[25] *ESPTerm - ESP8266 terminal emulator firmware*, https://github.com/espterm/espterm-firmware, Navštíveno 22.5.2018.

[26] *Toolchain pro vývoj programů esp8266*, https://github.com/esp8266/esp8266-wiki/wiki/Toolchain, Navštíveno 22.5.2018.

[27] *The fifteen most popular languages on GitHub 2017*, https://octoverse.github.com/, Navštíveno 21.5.2018.

[Bie] Biezl, *Schéma děliče napětí*, https://cs.wikipedia.org/wiki/D%C4%9Bli%C4%8D_nap%C4%9Bt%C3%AD#/media/File:Einfacher-unbelasteter-Spannungsteiler.svg, Navštíveno 23.5.2018.

[Des16] Designmodo, *Linecons smartphone-outline.svg*, 2016, Navštíveno 24.5.2018.

[Fre] Freepik, *Computer*, Navštíveno 16.5.2018.

[Kuk] Kukusak, *Schéma zapojení dolnopropustního filtru*, https://cs.wikipedia.org/wiki/Doln%C3%AD_propust#/media/File:Doln%C3%AD_propust_sch%C3%A9ma.PNG, Navštíveno 23.5.2018.

[Moy] Edward Moy, *XTerm Control Sequences*, http://invisible-island.net/xterm/ctlseqs/ctlseqs.html, Navštíveno 21.5.2018.

[Weba] Richard Webb, *ESP-12F WiFi Module Datasheet*, https://www.elecrow.com/download/ESP-12F.pdf, Navštíveno 21.5.2018.

[Webb] ———, *ESP8266 Bootloader Modes and GPIO state on Startup*, https://zoetrope.io/tech-blog/esp8266-bootloader-modes-and-gpio-state-startup/, Navštíveno 21.5.2018.

Příloha C

Soubory na CD

- autoupload
  - ubuntu
  - macOS
- contro_sequence_library
- demo
  - bell
  - blink
  - colored_meter
  - counter
  - led
  - post_parser
  - pwm_colored_generator
  - terminal_colors
  - thermometer
- diff_program_C_CPP
- documents
- drivers
  - macos
  - windows
- esp_led_control
- espterm_library
  - libraries
    - ArduinoJson (Knihovna od Benoit Blanchon)
  - zahradka
- nodejs_app
- setup_udev
- update st_link (Nástroje z webu st)
- thesis.pdf

Příloha D

Manuálové stránky

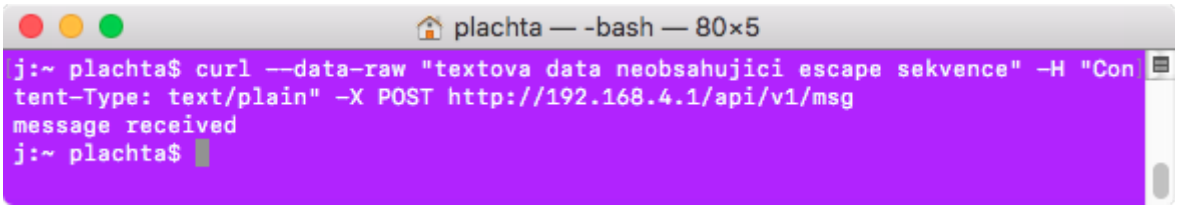
Několik dalších stránek obsahuje podrobnější popis některých úkonů potřebných pro sestavení prototypů. Dokumenty byly sepsány při realizaci programů v několika operačních systémech.

Curl - Zaslání zprávy do ESPTerm

Autor: Jan Holčák (holcajan@fel.cvut.cz)

Otevřeme terminál a zadáme příkaz obsahující naší zprávu v jednom řádku:

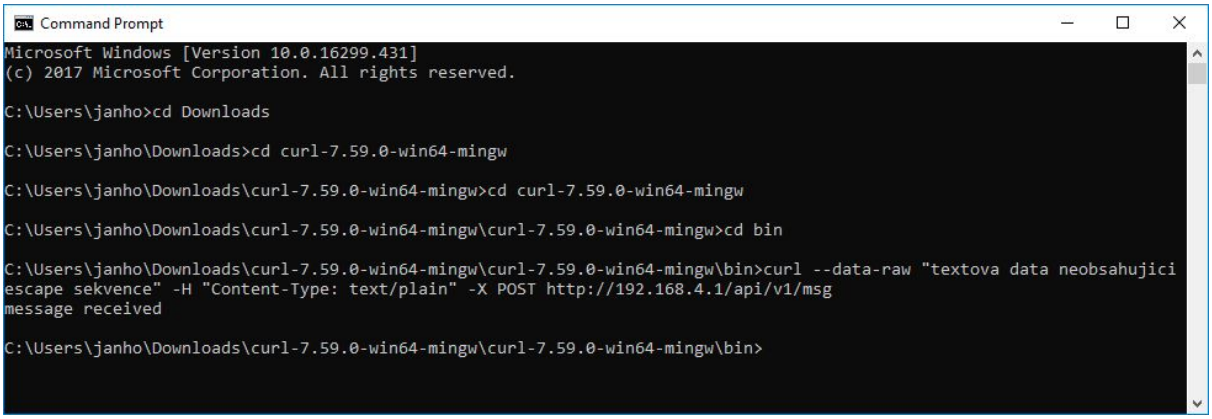
```
curl --data-raw "textova data neobsahujici escape sekvence" -H
"Content-Type: text/plain" -X POST http://192.168.4.1/api/v1/msg
```



Pro více informací lze navštívit manuál aplikace curl příkazem:

```
man curl
```

Chování bylo otestováno pod operačním systémem Windows10, macOS a Ubuntu18.04. Systém windows neposkytuje aplikaci **curl** v základním vybavení systému. Uživatel tohoto systému může využít možnost stažení portable verze tohoto programu.



Testovaná verze Win64 x86_64 zip **7.59.0** binary Viktor Szakats dostupná na webu https://curl.haxx.se/download.html úspěšně kontaktovala ESPterm modul. Před zadáním příkazu musíme nejdříve otevřít složku obsahující binární soubor. Užitečné příkazy pro procházení adresářů pod windows jsou cd (change directory) a dir (directory)

GNU screen

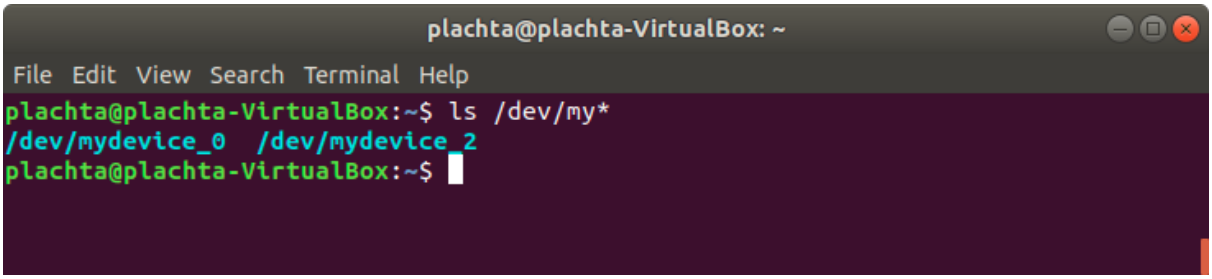
Autor: Jan Holčák (holcajan@fel.cvut.cz)

Screen textová aplikace běžící v emulátoru terminálu umožňuje spravovat běh většího počtu textových aplikací. Aplikace umožňuje mezi těmito aplikacemi přepínat a proto je někdy označována jako “správce oken” nebo “terminálový multiplexor”. V případě potřeby čtení a zápisu sériového portu můžeme tohoto programu využít k zprostředkování komunikace.

Spuštění aplikace

Před otevřením nového sezení musíme znát zdroj se kterým budeme komunikovat.
Pro vyhledání zařízení v operačním systému ubuntu použijeme příkaz `ls`. Tento příkaz vypíše seznam všech položek v zadaném umístění. Protože se

```
ls -al /dev/tty*
```



Pro nahlédnutí do oprávnění k přístupu na hledaný port můžeme využít příkazu:

```
ls - al
```

V operačním systému MacOS lze zařízení hledat podobně jako v Ubuntu

```
ls /dev/cu.*

screen /dev/cu.DEVICENAME BAUDRATE
screen /dev/cu.usbmodem1433 115200
```

V případě výpisu hlášky **[screen is terminating]** je nutné nastavit oprávnění pro připojené zařízení.

Uvolnění portu

Pokud se nám podařilo odpojit od aplikace screen, která ale stále běží na pozadí, můžeme se připojit příkazem:
screen -x

Ukončení aplikace

Postup pro ukončení aplikace se liší napříč používanými OS, což může být značně nepříjemné. Aplikaci ukončíme zápisem příkazů popsaných níže.

Linux (ubuntu)

```
ctrl-a
K (shift k)
y
```

macOS - cesta 1

```
ctrl-a
k
y
```

macOS cesta 2

```
ctrl-a
:quit
enter
```

Zdroje

http://aperiodic.net/screen/
https://pbxbook.com/other/mac-tty.html#screen

Jak na STM32 skrze st-link

Autor: Jan Holčák (holcajan@fel.cvut.cz)

ST-LINK

Pro snadné programování a zároveň ladění (debugging) dodává ST svůj obvod s názvem ST-LINK. U kitů nucleo je tento obvod součásti kitu jako nevratně oddělitelná součást. Tento kit umožňuje připojit zařízení jednoduše pomocí USB a poté simulovat uložiště dat. Nahrání programu lze provést jednoduchým přesunutím binárního souboru do uložiště.

Pokud je potřeba proces nahrávání automatizovat nebo potřebujeme pokročilejší funkce, můžeme zvolit instalaci aplikace st-link. Návod byl testován v operačním systému Ubuntu.

Stažení a kompilace st-link

Instalace potřebných balíčků. Některé možná nejsou potřeba ale všechny se mohou hodit.
```
sudo apt-get install libusb-1.0-0-dev git make cmake build-essential
```
Klonování repozitáře st-link do lokálního uložiště.
```
git clone https://github.com/texane/stlink stlink.git
```
Otevřeme složku staženého nástroje
```
cd stlink.git
```
Tímto zkompilujeme a nainstalujeme program st-flash
```
sudo make install
```
Tento příkaz opraví chybu s knihovnami.
```
sudo ldconfig
```

Pouziti nastroje st-flash

Nástroj automaticky detekuje připojený nucleo kit. Nahrání programu provedeme následujícím příkazem.
```
st-flash --reset write /path/to/image.bin 0x8000000
```

Zdroje

Tutorial instalace anglicky
https://github.com/texane/stlink/blob/master/doc/compiling.md
Zdrojové kódy programu st-link
https://github.com/texane/stlink
Shrnutí různých nástrojů pro dev linux+windows
http://www.emcu.it/STM32/What_should_I_use_to_develop_on_STM32/What_should_I_use_to_develop_on_STM32.html
http://www.instructables.com/id/STM32F103-Getting-started/

Dynamicky správce zařízení pro linux - udev

Autor: Jan Holčák (holcajan@fel.cvut.cz)

Proč oprávnění

Každý proces přistupující k USB zařízení musí mít potřebná oprávnění. Existuje množství způsobů jak přiřadit oprávnění z nichž některé slouží pouze dočasně. Někteří uživatelé pak volí jednoduchou ale ne zcela bezpečnou cestu s použitím uživatele root. Spuštění programu s plnými oprávněními může v některých případech přinést nečekané problémy.

Nastavení přístupu k USB

Dočasné

```
sudo chmod 666 /dev/<link na naše zařízení>
```

Trvalé

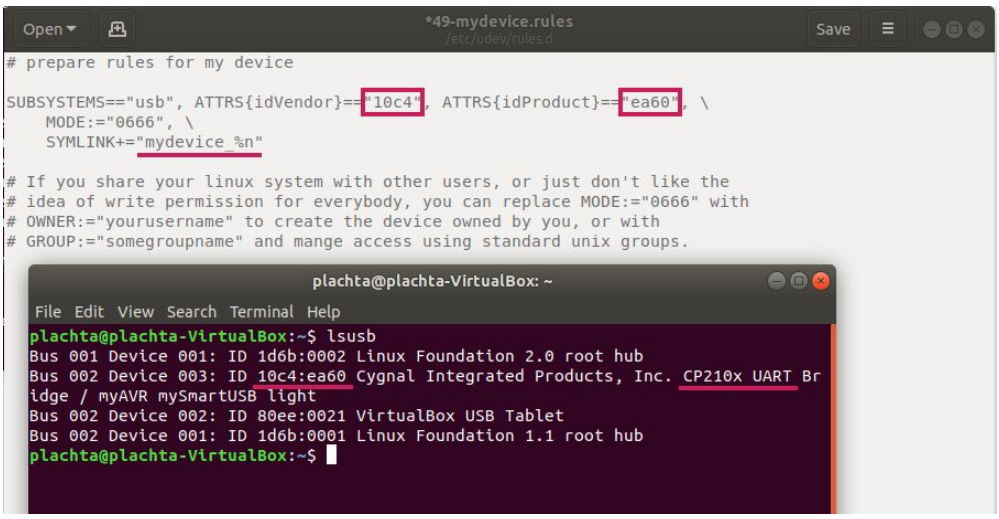
```
sudo usermod -a -G dialout $USER
```

Pro nastavení oprávnění využijeme možnosti změny konfiguračních souborů v adresáři /etc/udev/rules.d/ příkazem:

```
sudo nano /etc/udev/rules.d/49-mydevice.rules
```

Vzorový skript a základní soubory poskytované komunitou okolo výrobce ST obsažené v archivu ovladačů poskytovaných na oficiálních stránkách.

V případě nového zařízení neobsaženého v základních skriptech můžeme toto zařízení přidat zavedením jeho identifikačních atributů do konfiguračních souborů. Potřebné údaje zjistíme příkazem lsusb. Dále můžeme změnit název pod kterým bude zařízení v našem operačním systému vystupovat.

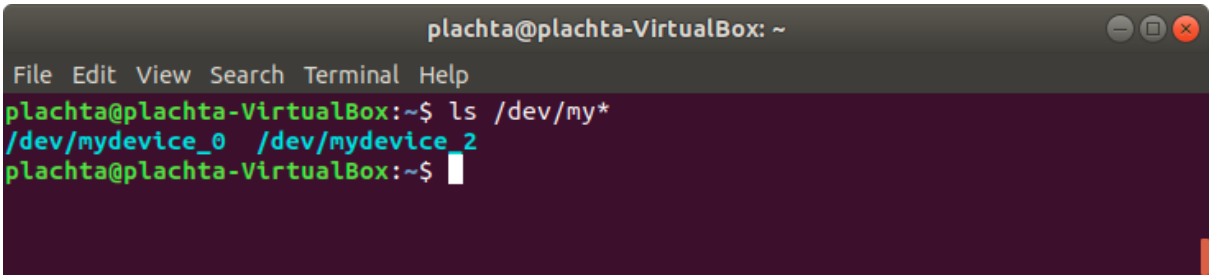


Aktivace pravidel

Změny některých oprávnění vyžadují opětovné přihlášení pro aktivaci změn těchto oprávnění. Konfigurační soubory musíme aktivovat restartem zařízení nebo znovunačtením změněných souborů příkazem:

```
sudo udevadm control --reload-rules && udevadm trigger
```

Pro úspěšné použití nových pravidel zařízení odpojíme a poté znovu připojíme.



Užitečné odkazy

Ovladače pro st-link obsahující udev konfigurační soubory
http://www.st.com/en/development-tools/st-link-v2.html#sw-tools-scroll
Soubory udev pro platformu STM32F4
https://github.com/dhylands/usb-ser-mon/blob/master/mk-udev-rules-stm32.sh

Upload programu do ESP8266

Autor: Jan Holčák (holcajan@fel.cvut.cz)

Na trhu existují různé verze ESP8266 modulů, proto neexistuje jeden univerzální postup. Naštěstí nejsou moduly natolik odlišné aby se musely provádět složité změny v budovaném software. **Pro upload programu bez jeho kompilace není potřeba stahovat celé SDK.** Přeji 0 odrazujících chyb a při nápadu na zlepšení lze připsat k dokumentu komentář.

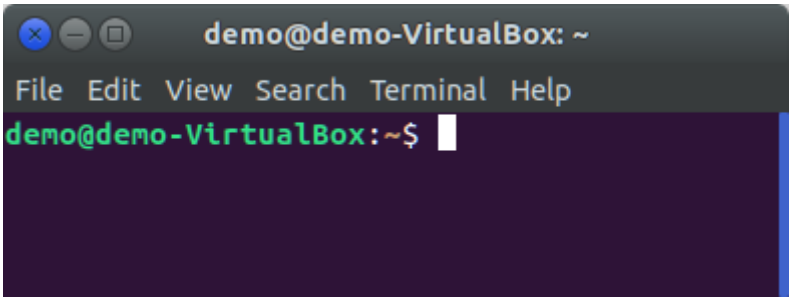
Ubuntu (Testovano na verzi 17.10)

K programování jsem tuto distribuci zvolil, protože patří mezi nejpopulárnější linuxové distribuce. Tento operační systém lze také v případě potřeby nainstalovat na USB flash disk. Linuxové prostředí je ideální pro programování různých programů. Právě v embedded se ukazuje mnoho výhod tohoto prostředí díky možnosti jednoduché automatizace.

Terminal

Nejdůležitější a nejmocnější výhoda prostředí linuxu je možnost ovládání v textovém režimu. Pro zadávání příkazů je potřeba otevřít příkazovou řádku neboli terminal.

Stisknutím kláves ctrl+alt+t vyvoláme požadované okno



Zde budeme zapisovat většinu potřebných příkazů před které budu zapisovat odlišnou barvou. Je potřeba myslet na to že příkaz ctrl+v má v terminálu jiný význam a pro vložení se používá ctrl+shift+v.

Aktualizace (můžeme přeskočit)

Pokud se setkame s operačním systémem, který je dlouho nepoužívaný, provedeme aktualizaci zadáním příkazů.

```
sudo apt update
sudo apt upgrade
```

Použité příkazy

`apt update` (aktualizace informací o balíčcích)
`sudo apt upgrade` (aktualizace zastaralých balíčků)
`cd` (cestování napříč adresáři)
`lsusb` (seznam připojených usb zařízení)
`rm` (mazání souborů a rm -rf pro rekurzivní mazání složek)

Toolchain+SDK

Instalace potřebných programů

```
sudo apt install make unrar-free autoconf automake libtool gcc g++ gperf \
flex bison texinfo gawk ncurses-dev libexpat-dev python-dev python \
python-serial sed git unzip bash help2man wget bzip2 libtool-bin git
```

Stažení SDK

git clone --recursive https://github.com/pfalcon/esp-open-sdk.git

Instalace SDK

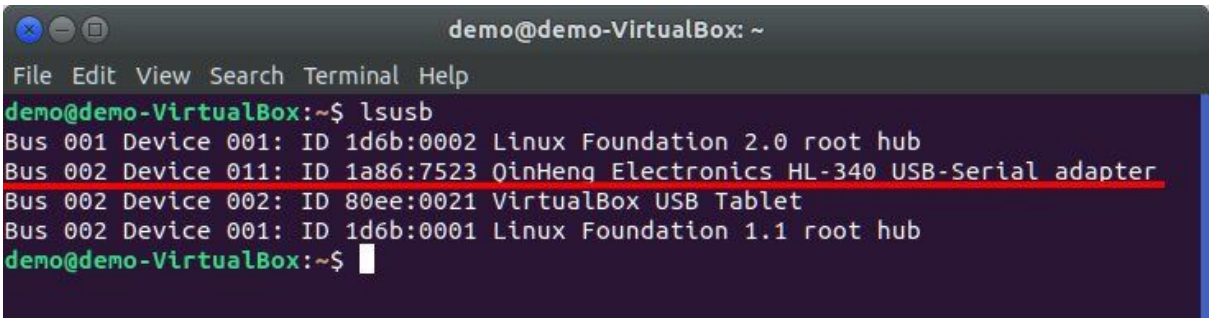
Instalace zabere i několik desítek minut a množství zdrojů. Kompilace nemusí být úspěšná při nedostatku místa na disku nebo v operační paměti. Doporučuji uvolnit alespoň **3GB volného místa** a neotvírat aplikace náročné na operační paměť. Ideální čas dát si kávu :)
`cd esp-open-sdk` (přesuneme se do složky esp-open-sdk)
`make` (sestaví sdk a vygeneruje binárky)
`cd` (návrat do domovského adresáře)

Nastavení cesty k SDK

uložit pro tohle sezeni
`export PATH=$PATH:/home/$(whoami)/esp-open-sdk/xtensa-lx106-elf/bin/`
uložit pro příští sezení
`printf "\nexport PATH=\$PATH:/home/$(whoami)/esp-open-sdk/xtensa-lx106-elf/bin/\n" >> ~/.profile`

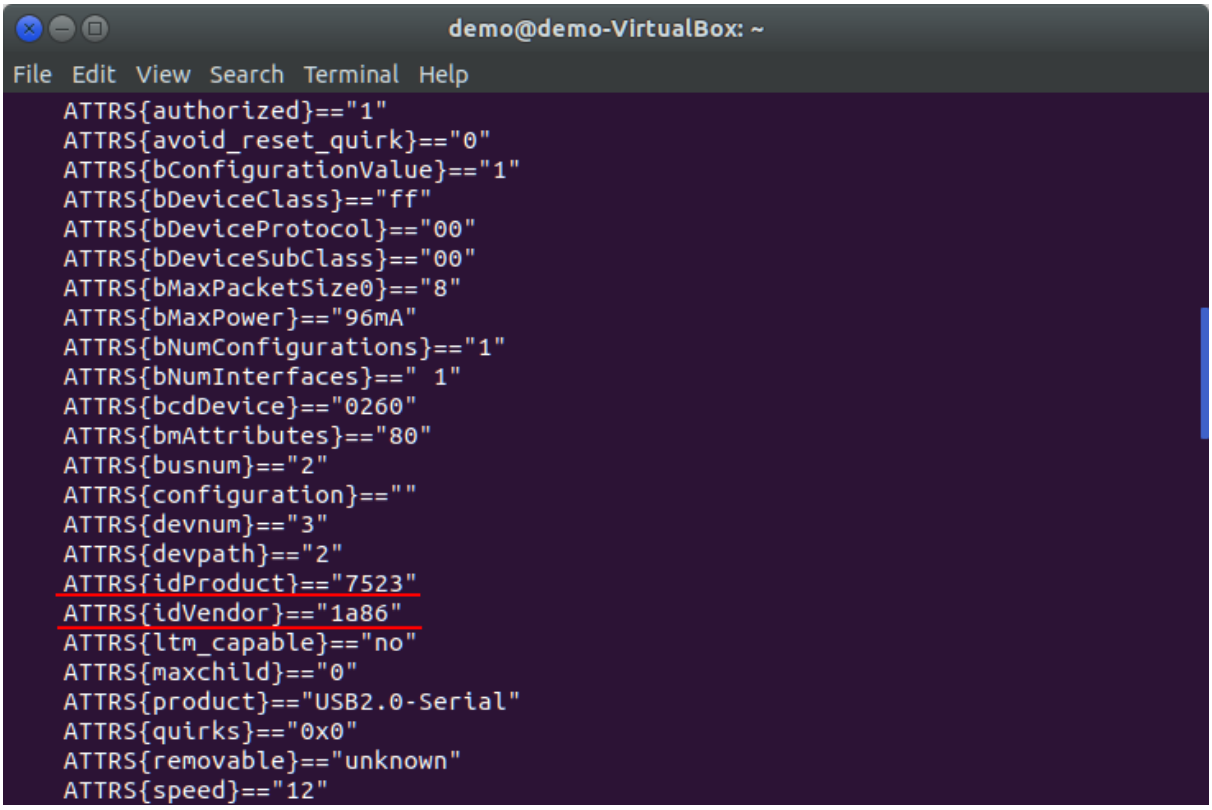
Nastavení práv k přístupu na USB

První se přesvědčíme že je ESP skutečně připojeno a pro systém viditelné.
`lsusb`



Zobrazení parametrů připojeného zažízení.

```
udevadm info -a -p $(udevadm info -q path -n /dev/ttyUSB0)
```



Nastavíme oprávnění pro připojené zařízení. Parametry ATTRS{idVendor} a ATTRS{idProduct} se mohou směnit v závislosti na výrobci modulu a proto je v následujícím příkazu upravíme podle potřeby.

```
sudo sh -c 'echo SUBSYSTEMS=="usb", ATTRS{idVendor}=="1a86",
ATTRS{idProduct}=="7523", MODE="0666", GROUP="plugdev" >>
/etc/udev/rules.d/81-microcontrollers.rules'
```

Nakonec musíme našeho uživatele přidat do skupiny plugdev a změny uložit.

```
sudo usermod -a -G plugdev demo
```

Těmito příkazy změny uvedeme v platnost

```
sudo /etc/init.d/udev restart
sudo udevadm control --reload-rules && udevadm trigger
```

Kompilace programu blinky.c

Přesuneme se do složky blinky
```
cd esp-open-sdk/examples/blinky
```
Zkompilujeme program (Makefile obsahuje možné příkazy)
```
make blinky-0x00000.bin
```
Nahrajeme program do ESP
```
make flash
```
Složku blinky doporučuji zkopírovat a použít jako skeleton pro další vývoj. (TODO: přidat další examples)

esptool.py (mimo SDK)

Nainstalované SDK už tento nástroj pro upload obsahuje. Někdy se ale může stát, že není potřeba stahovat celé SDK a chceme jen tento nástroj pro upload programu. Instalace samotného esptool.py se skládá ze dvou kroků. Nejprve nainstalujeme python baličkovací systém pip
```
sudo apt install python-pip
```
a poté samotný esptool.py
```
sudo pip install esptool
```
Při komplikacích s nahráváním nového firmware může pomoci příkaz
```
esptool.py --port /dev/ttyUSB0 --baud 115200 flash_id
```
který zjistí zda lze navázat komunikaci s připojeným ESP tím že odešle požadavek na typ paměti.

Pokud nechceme instalovat zbytečně python baličkovací systém můžeme použít také programu esptool. Od esptool.py se liší v podobě zadávaného příkazu.
Tento nástroj lze nainstalovat jediným příkazem
```
sudo apt install esptool
```
Nainstalovaný program pak spustíme příkazem
```
esptool
```
Pro informace o ovládání programu využijeme nápovědy
```
esptool -h
```
nebo podrobný manuál
```
man esptool
```

Upload programu ESPTERM

Na github stránce projektu https://github.com/espterm/espterm-firmware/releases si nejprve stáhneme požadovanou verzi. Po rozbalení se přesuneme v terminálu do složky, kterou archiv obsahuje a spustíme upload příkazem `./flash.sh`. Pokud se tímto způsobem upload nezdaří, pak můžeme zkusit zadat příkaz pro upload také ručně a to ve tvaru:
```
esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash 0x00000 ‘nazev_verze_0x00000.bin’ 0x40000 ‘nazev_verze_0x40000.bin’
```
Důležité je změnit název binárních souborů ve vzorovém příkazu.

Upload programu ESPTERM (windows - bash)

```
sudo apt install python-pip
sudo pip install esptool
```

Za pismeno X dosadime cislo COM portu zarizeni, tedy COM0 -> /dev/ttyS0

```
esptool.py --port /dev/ttySX write_flash 0x00000 2.3.0-pre1-0x00000-en.bin
0x40000 2.3.0-pre1-0x40000-en.bin
```

Slovníček

Toolkit

*A **toolkit (toolchain => output of one tool comprises the input of the next)** is a collection of tools that are useful in a given programming environment. Examples include: code compilers, resource compilers, code generators, code formatters, build systems. Exactly which tools are provided by a toolkit will depend on the nature of the platform involved.*
Zkráceně česky: Kolekce nástrojů potřebných k programování na dané platformě.

SDK

*An **SDK** is a collection of APIs, frameworks and toolkits provided by a platform provider to facilitate development of applications and components that will work with the platform.*
Zkráceně česky: Kolekce knihoven, API, frameworků používaných na dané platformě.

Zdroje

Zdrojový kód esptool
https://github.com/espressif/esptool
Zdrojový kód pro SDK
https://github.com/pfalcon/esp-open-sdk
Základní informace o kompilaci c++ pro ESP
https://hackaday.com/2014/12/08/compiling-your-own-programs-for-the-esp8266/
Rozložení paměti v ESP a demo projekt
https://github.com/cnlohr/esp8266ws2812i2s
Nastavení přístupových práv k USB
https://hackaday.com/2009/09/18/how-to-write-udev-rules/
https://www.esp32.com/viewtopic.php?f=2&t=2480
Zdrojový kód projektu ESPTERM
https://github.com/espterm/espterm-firmware
Žebříček linuxových distribucí
https://www.zive.cz/clanky/10-nejpopularnejsich-linuxovych-distribuci-roku-seradte-je-i-vy/sc-3-a-180943/default.aspx
Rozlišení základních pojmů v programování API,framework,toolkit,SDK
https://www.quora.com/What-is-the-difference-between-API-framework-a-toolkit-and-SDK

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jan H o l č á k

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Měřicí systém se vzdáleným přístupem

Pokyny pro vypracování:

Navrhněte a realizujte jednoduchý měřicí a ovládací systém se vzdáleným přístupem s rozhraním WiFi. Pro realizaci systému využijte mikrořadiče řady STM32, např. STM32F042 nebo STM32F072. Pro komunikaci prostřednictvím rozhraní WiFi se orientujte na použití některého hotového modulu, např. s obvodem ESP8266. Systém bude snímat data z různých snímačů umístěných na sledovaném objektu a předávat je dále. Podobně bude možno vzdáleně nastavovat výstupy systému a ovládat jimi sledovaný objekt. Vytvořte potřebné programové vybavení pro mikrořadič i pro nadřazený systém.

Seznam odborné literatury:

[1] The Definitive Guide to ARM® Cortex®-M0 and Cortex-M0+ Processors
[2] STMicroelectronics:RM0091: STM32F0x2 Reference manual
[3] STMicroelectronics: DS10147 STM32F042 Data

Vedoucí bakalářské práce: doc. Ing. Jan Fischer, CSc.

Platnost zadání: do konce letního semestru 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 1. 2017