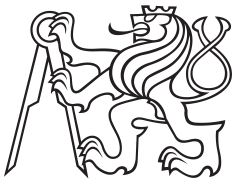
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická

Přenos dat ze senzorů při tvorbě experimentálních hraček a rehabilitačních pomůcek

Marek Jína

Školitel: Ing. Petr Novák, Ph.D.
Leden 2022

Poděkování

Rád bych poděkoval Ing. Petru Novákovi, Ph.D. za odborné vedení, ochotu a hlavně trpělivost, kterou projevil v průběhu zpracování bakalářské práce. Další poděkování patří mému bratrovi Martinu Jínovi za pomoc se syntaxí jazyka C# a své přítelkyni MUDr. Barboře Vyhnánkové za kontrolu pravopisu a korekci.

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, 3. ledna 2022

Abstrakt

Tato bakalářská práce hodnotí existující možnosti pro vývoj senzorových zařízení se zaměřením na praktické využití. Dalším bodem této práce je navržení a sestavení programového vybavení pro procesory typu ARM vhodného pro použití v experimentálních senzorových modulech, včetně návrhu jednotné struktury dat vhodné pro komunikaci. Součástí programového vybavení je jednoduchá desktopová aplikace pro zobrazení a uložení dat. Práce obsahuje návod na konfiguraci knihoven a jejich rozšíření.

Klíčová slova: STM32, Nucleo, Senzor, Sériová komunikace

Školitel: Ing. Petr Novák, Ph.D.

Abstract

This bachelor thesis evaluates the existing possibilities for the development of sensor devices with a focus on practical use. The next point of this thesis is the concept and assembly of software for ARM type processors suitable for use in experimental sensor modules, including the design of a unified data structure suitable for communication. The software includes a simple desktop application for displaying and storing data for later use. The thesis contains instructions for configuring libraries and their extensions.

Keywords: STM32, Nucleo, Sensor, Serial communication

Title translation: Data Transfer from Sensors in the Creation of Experimental Toys and Rehabilitation Aids

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

V dnešní době, kdy svět prochází digitalizací systémů, je pojem „Internet věcí“ (IoT - Internet of Things) velké téma nejen světových organizací vyvíjející domácí spotřebiče, ale i menších firem. Díky tomu je stále větší poptávka po samostatných systémech, jež na základě změřených dat jsou schopny vyhodnotit nejen krizové situace a dokážou bez lidského zásahu předejít pohromám - od zvadlé a nezalité květiny až po vznícení spotřebiče v domácnosti. Jedná se o zařízení obsahující senzor či detektor, který v pravidelných intervalech sbírá data, jež jsou přenášena za účelem jejich dalšího zpracování do hlavní řídící jednotky, kde případně dojde k automatické odezvě.

Mezi senzorová zařízení tohoto typu lze dnes zahrnout i rehabilitační pomůcky, které získaly podobu „krabiček“ s interní baterií. Jejich obsahem je množství potřebných senzorů a komunikační kanál pro přenos dat za účelem vyhodnocení všech obdržených informací. Právě zmíněná komunikace je jedním ze stěžejních problémů každého zařízení. Může se zdát, že se jedná o okrajovou záležitost – „pouhý“ přenos dat, opak je pravdou. Často se jedná o jednu z nejtěžších a časově nejnáročnějších fází raného stádia vývoje elektronického zařízení. Tato fáze zahrnuje výběr vhodného přenosového média (drátově či bezdrátově), správného protokolu pro přenos dat a současně způsob, jakým data budou přenášena, aby nedocházelo ke zbytečným energetickým ztrátám nebo ke ztrátě požadované přesnosti dat.

To vše klade nemalé nároky na znalosti tvůrců zařízení. V případě velkých firem, kde se zařízení vyrábí v tisíci kusových sériích a kde jsou přítomni odborníci na komunikační technologie, můžeme předpokládat vynaložení minimálních časových zdrojů na toto odvětví vývoje. Ovšem v opačném případě, při tvorbě jednokusových experimentálních zařízení ve specializovaných laboratořích či v domácím kutilství, může zprovoznění komunikace pro přenos dat zkonzumovat znatelnou část času vyhrazeného na projekt.

Při vývoji malosériových nebo originálních zařízení je bez pochyb vhodné mít po ruce odladěné programové knihovny umožňující přenos dat na základě často se vyskytujících komunikačních kanálů (například USB, WiFi, Bluetooth), a ty si vždy jednoduše nakonfigurovat podle právě vytvářeného zařízení a jeho možností či potřeb a poté se dále věnovat jen provozu senzorů a testování. Právě problematice ladění programových knihoven se věnuji v mé práci.

Kapitola 2

Existující modulární senzory

U dosavadních vytvořených produktů, které se dají použít ke snímání fyzikálních veličin nebo k detekci požadovaných stavů, je nutné hodnotit na základě jejich snadnosti použití, možnosti konfigurace, dostupnosti pro různé platformy a dle šířky škály použitelných sběrnic.

2.1 Lego Mindstorms

LEGO je stavebnice populární nejen mezi dětmi. Řada Mindstorms svůj princip postavila na řídící jednotce, která přijímá data ze senzorů, a reakci zajišťují elektromotorem poháněné periferie.

Výhodou je jednoduchá programovatelnost jednotky na počítači nebo v mobilní aplikaci skrze GUI (grafické uživatelské rozhraní) i bez znalosti jakéhokoli programovacího jazyka.

Hlavními nevýhodami těchto stavebnic je omezená programovatelnost, která ruku v ruce vzniká při snaze o jednoduché programování a nízký počet použitelných senzorů (na oficiálních stránkách společnosti LEGO jsou pouze 3 typy senzorů). Nepřesné motory a možnost připojení maximálně čtyř senzorů k jedné řídící jednotce jsou dalším záporným bodem v hodnocení. Navíc snímaná data nelze dál odeslat do počítačové sítě.

Z těchto důvodů produkt LEGO Mindstorms není vhodný pro vývoj pokročilých hraček, ale stojí za zmínění pro méně náročné vývojáře.

Obrázek 2.1: LEGO - Mindstorms

2.2 LittleBits

LittleBits je další z rodiny modulárních stavebnic pro děti i dospělé. Všechny komponenty jsou umístěny na destičky, které se navzájem spojují pomocí magnetů nebo kabelů s krokodýlky. Stavebnice obsahuje celkem velké množství součástek a senzorů od akcelerometrů, tlakových a světelných senzorů až po logická hradla. S rozšířením CloudBit je možná i komunikace přes internetové rozhraní (cloud) poskytované a spravované přímo společností Spheroinc (momentální vlastník licencí společnosti LittleBits).

Výhodné je jednoduché programování pomocí mobilní aplikace či webového rozhraní na principu IFTTT (If This Than That). Vzniklý program je nejen schopen rozhodovat se na základě vnějších vstupů nebo informovat svého majitele pomocí SMS či jednoduchého emailu, ale dokáže reagovat i na předem definované zprávy směrem od uživatele.

Nevýhodou je omezené programovací prostředí a možnost použití jen senzorů LittleBits. Škála měřených veličin a použitých senzorů je tedy značné omezená a z tohoto důvodu se opět hodí jen pro nenáročné vývojáře nebo pro počátky výuky programování.

Obrázek 2.2: LittleBits

2.3 Vlastní zařízení s použitím mikrokontroléru

Další možností vytvoření senzoru nebo senzorové sítě je vytvoření vlastního elektrického obvodu s řídící jednotkou a programem.

Mikrokontrolér je integrovaný obvod s CPU (central processing unit) schopný na základně vnitřního algoritmu zpracovávat instrukce a upravovat výstupní data na základě dat vstupních. V dnešní době mikroprocesory ovládají valnou většinu elektronických spotřebičů, dopravních prostředků či zabezpečovacích systémů, včetně již zmíněné stavebnice LEGO Mindstorms.

Velkou výhodou mikrokontrolérů je neomezená programovatelnost v rámci podporovaného jazyka (například C, C++, Wiring, apod.) či přímo pomocí zdrojového kódu neboli Assembleru. S touto výhodou se pojí prakticky nevyčerpatelná škála použitelných senzorů, komunikačních protokolů i možností zpracování dat.

Za použití bateriového napájení pak lze sestavit samostatný bezdrátový senzorový modul, který v nastavených intervalech odesílá data hlavnímu počítači k uložení nebo dalšímu zpracování.

Nevýhodné u tvorby senzorů s mikroprocesorem jsou požadované znalosti stavby samotného mikroprocesoru a programovacího jazyka. Tato nevýhoda se částečně eliminuje použitím vývojových desek (viz následující kapitola).

Obrázek 2.3: Procesor ARM-Cortex

Kapitola 3

Vývojové desky

Vývojová deska je PCB (Printed Circuit Board čili deska plošných spojů) obsahující mikrokontrolér s různými velikostmi paměti, druhy periferií a GPIO piny (general-purpose input/output), pro zachování programovatelných vstupů a výstupů. V závislosti na použitém procesoru má deska vestavěné periferie i počet GPIO pinů pro komunikaci modulu s dalšími zařízeními.

Účelem vývojových desek je zjednodušit práci s mikročipem a usnadnit tak vývoj jejich aplikací za pomoci jednotných prostředí se širokým rozsahem možností.

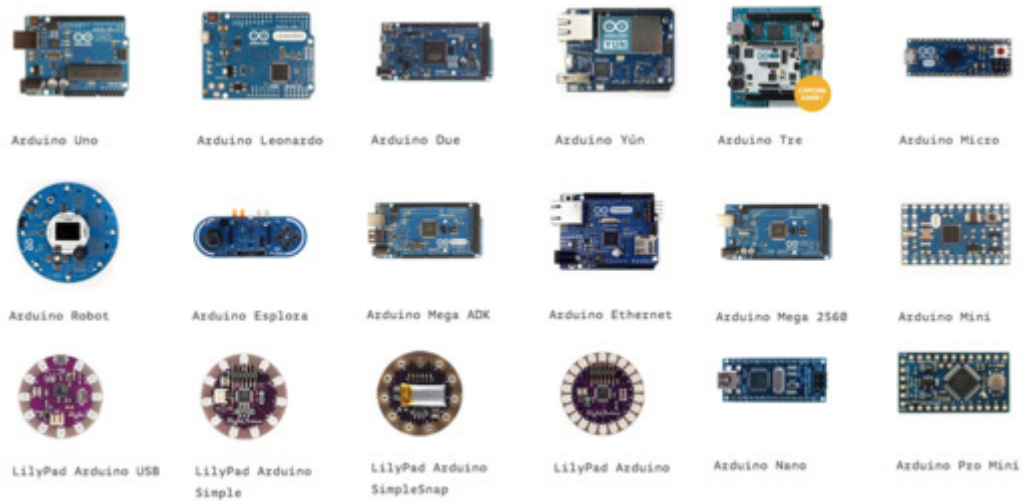
Jednotlivé desky se od sebe liší i v závislosti na výrobci. Za zmínku stojí například firmy: Atmel (později zakoupena firmou Microchip Technology), Cypress Semicondutor, STMicroelectronics a mnoho dalších. Průkopníkem ve světě vývojových desek je Arduino.

3.0.1 Arduino

Arduino je open-source multi-platform elektronická platforma vytvořená pro cenově dostupné použití mikrokontrolérů ve zjednodušené formě. Pro programování byl vytvořen oficiální software Arduino IDE. Jeho programovací jazyk je postavený na jazyku Wiring, který byl vytvořen přímo pro programování mikrokontrolérů a jeho fundamenty stojí na jazyce C.

Historická většina Arduino boardů je postavena za použití 8bitových procesorů od firmy Atmel (ATmega). V dnešní době je však k dostání i 32bitová verze s procesory například typu ARM nebo Intel Quark.

Za velkou nevýhodu Arduino boardů může být považován chybějící debugger, čímž se zprovoznění programového vybavení stává složitějším, než by muselo. Obtížnost ladění programu pak přímou úměrou stoupá s jeho složitostí.



Obrázek 3.1: Arduino family

3.1 Procesory ARM

Advanced Risk Machine (ARM) je rodina 32bitových procesorů vyvinutá britskou společností ARM Holdings, která je poskytuje výrobcům zařízení ve formě licencí. Princip licencí funguje na základě standardního jádra procesoru rozšířeného o periferie koncového výrobce. Periferie mohou být vyvinuty za účelem spotřeby, výkonu, přesnosti měření či k dosažení jiných vlastností.

Samotná architektura ARM procesorů stojí z většiny na (standardním) jádře vykonávajícím instrukce a řídícím množství samostatně pracujících periferií.

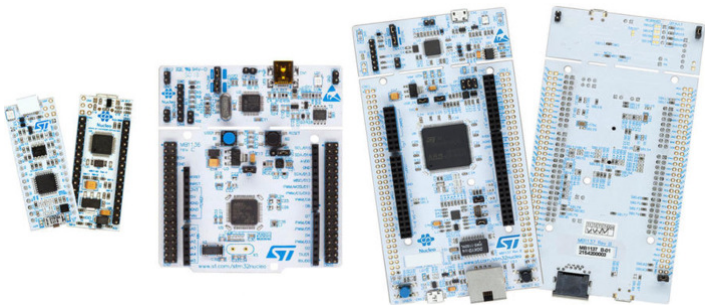
3.1.1 STM32 Nucleo

Pravděpodobně dnešním nejznámějším zástupcem vývojových desek s architekturou procesorů ARM jsou desky STM32 Nucleo od společnosti STMicroelectronics. Zmíněná firma nabízí na trhu širokou škálu desek 8bitových (STM8) a 32bitových (STM32) procesorů v různých variantách velikostí paměti, počtem periferií a programovatelných GPIO pinů.

Velmi známé prostředí pro programování Nucleo boardů je Mbed IDE. Prostředí je volně dostupné ke stažení i k použití ve webovém rozhraní. Tedy jediné, co potřebujete k programování, je připojení k internetu a funkční prohlížeč.

Pro účely této bakalářské práce jsem zvolil oficiální vývojové prostředí STM32CubeIDE, protože umožňuje snazší nastavení periferií a další funkce, jež velmi usnadňují jeho využití. Toto prostředí je také volně k dispozici ke stažení na internetu.

Na rozdíl od prostředí Arduino má Nucleo ve většině verzí k dispozici real-time debugger, díky kterému jeho uživatel může algoritmus pozastavit za běhu a zkontrolovat, jestli nastavení hodnot je tak, jak bylo zamýšleno.



Obrázek 3.2: Nucleo family

Kapitola 4

Cíle práce

Bakalářská práce je zaměřena na návrh programu a jeho sestavení vhodné k použití ve vývojových deskách typu Nucleo a Arduino. Důležitým parametrem programu je možnost použití jakéhokoliv senzoru, v rámci nakonfigurovaných sběrnic, a výběr z několika komunikačních kanálů pro odesílání dat ze senzorového modulu do koncového zařízení, tedy počítače či mobilního telefonu.

Senzorový modul je napsán v jazyce C a je sestaven tak, aby byl snadno rozšiřitelný jak o další sběrnice k periferiím, tak o nové komunikační kanály.

Součástí práce je i jednoduchý program pro koncové zařízení umožňující snadné připojení modulu. Program je vhodný k zobrazení snímaných hodnot. Pro zajištění funkčnosti, bez závislosti na zvolené platformě, byl pro programování zvolen jazyk C#.

Uvědomit si spjatost hotové kostry programového vybavení a dokumentace k ní připojené je velmi podstatné. Tato práce slouží jako záznam o vývoji pomocných knihoven pro senzorová zařízenía též jako průvodce úpravou jejich konfiguračních souborů pro použití uživatelem.

Žádný elektronický systém nemůže fungovat bez napájení. Senzorové moduly postavené na mikrokontroléru je možné napájet z baterie a ze sítě. Vhodné napájení je v praxi vybíráno dle parametrů zamýšleného použití, například umístění modulu (dostupnost elektrické přípojky), jeho spotřeba na základě specifikací senzorů i frekvence měření, apod. Problematice napájení modulu se tato práce nevěnuje.

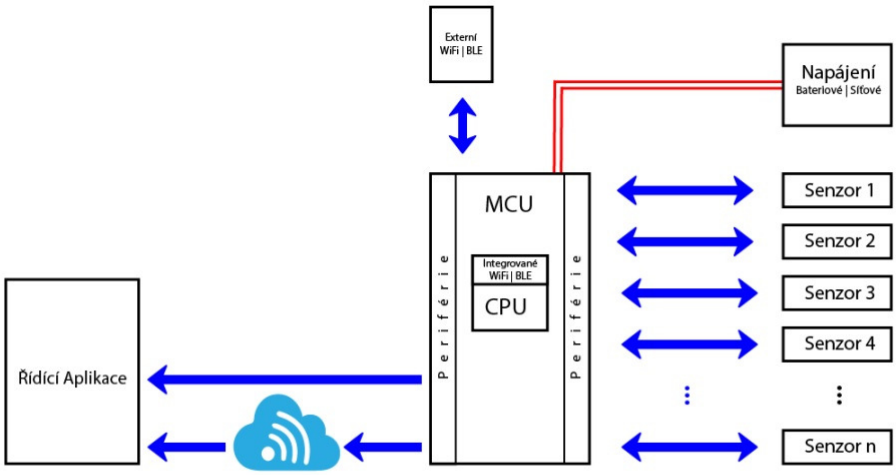
Hlavní cíle bakalářské práce jsou stanoveny následovně:

- Zhodnoťte existující programovou podporu pro snadnou tvorbu experimentálních senzorových modulů s přenosem dat za účelem jejich uložení a dalšího zpracování. Hodnoťte zejména: snadnost použití, možnosti konfigurace, dostupnost pro často využívané platformy a soubor použitelných typů sběrnic / senzorů.
  - V kapitole „Existující modulární senzory“ jsem sestavil ukázku komerčně dostupných možností pro senzorové moduly a zhodnotil jsem jejich vhodnost dle zadaných kritérií.
- Navrhněte protokol a strukturu dat ze senzorů umožňující komunikaci (pokud možno) nezávisle na skutečně použitém komunikačním kanále jako jsou: USB, U(S)ART, BlueTooth (Classic/LE), WiFi, LAN a podobné. Vše tak, aby bylo možno pouhou změnou konfigurace přenášet stejná data jiným komunikačním kanálem podle aktuální potřeby.
  - V kapitole „Struktura dat pro komunikaci“ představuji navrženou strukturu dat určených pro komunikaci mezi modulem a koncovým zařízením. Popisuji části kompletující celé textové řetězce tak, aby se daly snadno rozklíčovat při zachycení mimo zařízení, za účelem snadného ladění případných problémů či vývoje sofistikovanějšího programu kompatibilního s jednou či druhou stranou komunikačního kanálu.
- Vytvořte programové knihovny pro procesorové platformy STM32 (ARM) a Arduino (ESP) na jedné straně a běžné operační systémy Windows, Android, iOS na straně druhé, umožňující komunikaci a tedy přenos dat ze senzorů pomocí aspoň tří různých typů rozhraní, např. USB, U(S)ART, BlueTooth (Classic/LE), WiFi, LAN, atd., ty vyberte podle návrhu.
  - Programové knihovny byly vytvořeny dle požadavků v zadání a jsou přiloženy v podobě programovatelných projektů k práci. Kapitola „Realizace návrhu“ pojednává o stavbě programu nejen senzorového modulu. Kapitola „Počítačová aplikace“ pak slouží jako návod k jeho konfiguraci pro použití uživatelem.
- Demonstrujte výsledky práce vytvořením modulů využívající některé senzory, např. pohybový, tlakový, detekce plynů, čtení NFC tagů atd. a přenášející data do aplikace na PC / tabletu / mobilu s využitím vybraných komunikačních kanálů podle aktuálního nastavení uživatelem.
  - Pro demonstrační účely vznikl senzorový modul s procesorem ARM využívající všechny implementované sběrnice, komunikující po sériové lince USART. K zobrazení výstupních hodnot slouží desktopová aplikace spuštěná na operačním systému Windows.

Kapitola 5

Obecný návrh

Tato kapitola představuje čistě obecný návrh systému pro demonstraci cesty dat od samotného měření senzory, přes úpravu v modulu a zapsání do textové struktury, až po zobrazení na monitoru počítače.



Obrázek 5.1: Blokové schéma měřícího systému

Z blokového schématu je zřejmé, že základem senzorového modulu je mikrokontrolér (vývojová deska) využívající nakonfigurované periferie pro provedení nezbytných operací. Periferie lze pro demonstrační účely rozdělit na pomyslné dvě strany. Na pravou stranu modulu jsem umístil periferie zajišťující komunikaci s jednotlivými senzory a sbírání změřených dat. Komunikace se senzory probíhá skrze sběrnice.

Posbíraná data jsou dočasně uložena v modulu, dokud nedojde k jejich zpracování. Pod pojmem zpracování si lze představit nejen výpočet konkrétní

hodnoty ze změřených dat a kalibračních matematických funkcí daného senzoru, ale i uživatelské funkce měnící například klasickou logickou hodnotu (1/0) na textový výstup (zapnuto/vypnuto).

Dalším krokem je zapsání hodnot do jednoho balíčku dat a jeho následné odeslání komunikačním kanálem nacházejícím se na levé straně modulu na blokovém schématu. Na druhé straně komunikačního kanálu se balíček dat dočasně uloží a jeho obsah se analyzuje. Vyzvednuté hodnoty se zapíší do odpovídajících neznámých a dojde k jejich zobrazení v tabulce senzorů.

Aby celá tato komunikace mohla správně fungovat, je třeba zajistit několik důležitých nastavení, než se vůbec mohou nějaká data změřit. Z tohoto důvodu je potřeba před během hlavní smyčky algoritmu vyvolat inicializační proces, který má na starosti tato nastavení senzorů, komunikací a samotného modulu.

5.1 Kostra programu senzorového modulu

- Inicializace
  - Komunikace
    - Nastavení pinů pro komunikaci s koncovým zařízením
    - Deklarace bufferů pro příchozí i odchozí pakety
  - Senzory
    - Nastavení pinů pro komunikaci se senzory
    - Konfigurace a spuštění periférií sběrnic
    - Prvotní funkce zvolení módu senzorů
- Hlavní smyčka
  - Příkaz z koncového zařízení
    - Přijetí paketu v přerušení
    - Analýza paketu, vyzvednutí příkazu
    - Provedení příkazu
    - Zahození paketu
  - Datový paket
    - Deklarace proměnné podle typu senzoru
    - Získání dat ze senzoru
    - Vložení dat do senzorového bufferu
    - Konverze dat na výstupní hodnotu
    - Zapsání hodnoty do paketu
    - Odeslání paketu
  - Informační paket
    - Získání podrobností o senzorech
    - Vložení do paketu
    - Odeslání paketu

5.2 Hardware požadavky

Stručné shrnutí základních požadavků na fyzickou formu projektu:

- Základ projektu - vybrat minimálně jeden mikrokontrolér typu Nucleo (procesor ARM) a jeden mikrokontrolér typu Arduino (procesor ESP), oba 32bitové.
- Použít dostatečný počet senzorů pro demonstraci základních sběrnic a datových typů změřených hodnot.
- Zprovoznit alespoň dva komunikační kanály (drátový a bezdrátový).

5.3 Software požadavky

Výčet požadavků na programové vybavení:

- Pro vývoj využít rozšířené multi-platform programovací jazyky. C pro mikrokontrolér dovolující snadné přenesení na jiný typ procesoru. C# pro koncovou aplikaci umožňující nezávislost použitého operačního systému.
- Nakonfigurovat základní sběrnice pro připojení senzorů a komunikační kanály pro přenos dat.
- Využít nezbytný počet datových typů s dostatečnou velikostí k zapisování vstupních hodnot.
- Předpřipravit adresář konfiguračních souborů pro snadné nastavení modulu a jeho komunikace či pro jednoduché přidávání dalších senzorů.

5.4 Požadavky na dokumentaci

Jak již bylo zmíněno v cílech práce, tento dokument současně zastupuje funkci manuálu pro uživatele vyvinutých knihoven. Zde je shrnutí, čeho se tato zastupující rovina týká.

- Seznámení se s prostředím STM32CubeIDE.
- Průvodce základní konfigurací komunikačního modulu.
- Vysvětlení principu funkce inicializačních fází použitých sběrnic včetně demonstračně nakonfigurovaných souborů reálných senzorů.
- Rozbor vnitřních funkcí nezbytných pro pokročilou úpravu změřených hodnot před odesláním.
- Popis použití komunikačních kanálů.
- Představení PC aplikace pro zobrazení hodnot a tipy na ladění komunikace.

5.5 Vysvětlení pojmů

Následuje soubor pojmů použitých v práci, které nemusí zcela odpovídat skutečné definici slova.

- Senzor
  - Elektronická součástka převádějící měřenou fyzikální veličinu na analogový či digitální signál komunikující po určitě sběrnici.
- Senzorová sběrnice
  - Soustava vodičů určená na přenos dat mezi senzorem a modulem podle předdefinovaných pravidel.
- Modul (senzorový)
  - Fyzická forma práce. Obsahuje mikrokontrolér, senzory i komunikační modul.
- Komunikační modul
  - Oddělitelná část modulu zajišťující bezdrátovou komunikaci s počítačem.
- Paket
  - String (pole charakterů) obsahující data odesílaná po komunikačním kanále.

Kapitola 6

Použité komponenty

Hlavní komponenty použité v projektu a popis jejich parametrů je náplní této kapitoly.

6.1 Mikrokontrolér

6.1.1 Nucleo SMT32F401RE

Tato deska patří do série Nucleo-64 mezi modely s označením High performance. Označení modelu napovídá, že základem je procesor ARM cortex-M4, k dispozici je 64 pinů a flash paměť o velikosti 512 kB. Obsahuje vestavěný programátor a debugger připojitelný konektorem mini-B USB k počítači s programovacím prostředím.

Další parametry lze podle označení dohledat na oficiálních webových stránkách www.st.com.

6.1.2 Arduino Wemos D1 mini

Základem vývojové desky je procesor ESP8266 s flash pamětí 4 MB. Deska má k dispozici 16 programovatelných pinů a vestavěný WiFi modul. Tento model se jeví jako ideální varianta pro rychlé zprovoznění senzorového modulu komunikujícího bezdrátovou technologií.

Další parametry lze opět podle označení dohledat na webových stránkách výrobce - www.arduino.com

6.2 Senzory/Detektory

Tato kapitola se zabývá senzory použité v projektu. Podstatné je, že nezáleží plně na modelu senzoru či měřené veličině, nýbrž na zastoupení datových typů výstupů a komunikačních sběrnic. Senzory tedy byly vybrány pro demonstrační účely.

6.2.1 Detektor pohybu

Model 287-18001 od firmy MCM detekuje pohyb do vzdálenosti až 7 metrů. Pracuje na napěťové škále napájení 5-20V a průměrný odběr činí 65mA.

Výstup senzoru je formou digitálního signálu zajištěn pomocí TTL 3.3V/0V.

6.2.2 Senzor IR světla

Model D131038 od firmy LaskaKit snímá světlo o vlnové délce 760nm, má k dispozici nastavitelný komparátor pro výstup v digitální formě. Pracuje na napěťové škále napájení 3.3V - 5V a průměrný odběr činí 15mA.

Výstup senzoru je možný digitální i analogový. Pro naše účely využijeme analogový výstup.

6.2.3 Senzor teploty

Model BMP280 od firmy Bosch snímá teplotu a tlak okolního prostředí. Pracuje na napěťové škále napájení 1.8V - 3.6V a průměrný odběr činí 2.7uA při frekvenci měření 1 Hz.

Jako komunikační sběrnici je možné použít I2C či SPI, v tomto případě I2C. Naměřená hodnota se čte z registrů senzoru a upravuje se pomocí kalibračních hodnot senzoru.

6.2.4 Akcelerometr

Model LA131074 od firmy LaskaKit je multifunkční čidlo obsahující akcelerometr, gyroskop a magnetometr. Pracuje na napěťové škále napájení 3V - 5V a odběr proudu se hodně liší v závislosti na zvoleném módu a frekvenci měření od stovek uA do jednotek mA za senzor.

U čidla je též na výběr mezi komunikační sběrnicí I2C a SPI. Pro účely bakalářské práce byl použit pouze akcelerometr a sběrnici SPI. Zajímavým zpestřením úpravy výstupní hodnoty je fakt, že akcelerometr sbírá data ze tří na sebe kolmých směrů.

6.3 Komunikační moduly

Druhy komunikací senzorového modulu s počítačem budou představeny v následující kapitole.

6.3.1 USART

Universal Synchronous / Asynchronous Receive and Transmit je drátový komunikační protokol s možností posílat data skrze dvě komunikační média. Jedná se tedy o komunikaci Tx to Rx na každé straně v asynchronním módu, tedy Full dulpex. V synchronním módu je do komunikace implementován časovač (clock) umožňující rychlejší datové přenosy za cenu dalšího propojení.

Vzhledem k tomu, že se jedná o drátovou komunikaci, není třeba žádný speciální externí hardware. Oba mikrokontroléry stačí správně nakonfigurovat.

6.3.2 Bluetooth

Technologie Bluetooth je bezdrátovou komunikací pracující v ISM pásmu 2,4 GHz pomocí FHSS, což je metoda jejíž principem je přeskakování mezi mnoha frekvencemi ve frekvenčním pásmu a tím maximalizovat přenosovou rychlost.

Ani jedna ze mnou použitých vývojových desek nemá zabudovaný Bluetooth, z toho důvodu použiji externí modul připojitelný pomocí USART technologie popsané výše.

6.3.3 WiFi

WiFi je další bezdrátová komunikační technologie využívající nezpoplatněného ISM pásma 2,4 GHz. Tato technologie je poněkud složitější a to je důvodem, proč se WiFi moduly často integrují přímo do procesorů.

V projektu práce je WiFi modul součástí mikrokontroléru Arduino Wemos.

6.4 Software

6.4.1 STM32CubeIDE

Vývojové prostředí Cube IDE vyvinuté firmou STM je připraveno poskytnout uživateli plný komfort při programování mikrokontrolérů STM32 Nucleo. Díky podpoře programovacího jazyka C/C++ a implementaci knihoven HAL je obsluha komunikací s periferiemi značně zjednodušená a hodí se tak i pro naprosté nováčky v oboru.

Pro projekt bylo použito Cube IDE s ohledem na to, že je multi-platformní a jeho prostředí usnadní ladění programu za běhu.

6.4.2 Arduino IDE

Arduino Integrated Development Environment je multi-platformní vývojové prostředí určeno primárně pro programování desek Arduino. Stejně jako výše zmíněné prostředí od STM i Arduino IDE poskytuje plnou podporu jazyka C a C++ obohacenou o knihovny projektu Arduino.

6.4.3 Microsoft Visual Studio

Pro vývoj desktopové aplikace jsem zvolil Visual Studio od firmy Microsoft. Hlavním důvodem jsou mé předchozí zkušenosti s prostředím a možnost použití profesionálního software díky licenci poskytnutou univerzitou.

6.4.4 CoolTerm

Mezi použité programové vybavení patří freeware program CoolTerm (Roger Meier). S jeho pomocí lze snadno sledovat datový tok na portech počítače a posílat zpět textové příkazy do modulu. V případě zdánlivě bezproblémové práce modulu ale nezachycení dat v koncovém zařízení je tato aplikace jednoduchým způsobem pro detekci problému.

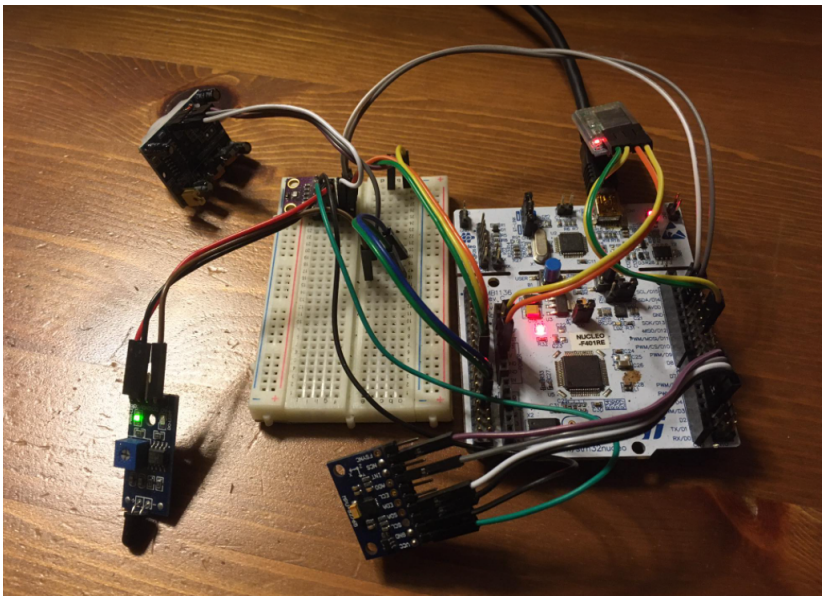
Kapitola 7

Realizace návrhu

Tato kapitola popisuje sestavení programového vybavení.

7.1 Senzorový modul

Každý program má stabilní jádro/tělo sloužící jako kostra celého programu. Další funkce se pak nabalují jako virtuální vrstvy a při dobře zpracovaném základu nevzniká problém s kompatibilitou.



Obrázek 7.1: Senzorový modul

7.1.1 Inicializace

První funkcí modulu před spuštěním hlavní smyčky je inicializace všech periferií a nastavení GPIO pinů. To zajišťují funkce CommunInit() a SensorInit(). Zmíněné funkce mají za úkol postupně projít všechny nakonfigurované komunikace, respektive senzory, a spustit jejich inicializační funkce.

7.1.2 Sběrnice

Funkce hlavní smyčky modulu začíná u senzorů měřících konkrétní veličinu. Aby však modul vůbec mohl se senzorem komunikovat, musí se nejprve zajistit přenos dat mezi nimi. K tomu slouží tzv. sběrnice.

Zadání práce přímo nespecifikuje počet použitých sběrnic, ani konkrétně které mají být použité. Pro demonstrační účely byly zvoleny následující.

Jednovodičové komunikace

Již dle názvu lze poznat, že se jedná o sběrnice komunikující po jednom datovém vodiči. Komunikace probíhá na úrovni čtení dat přímo na pinu kontroleru. Senzor či detektor pak v tomto případě funguje spíš jako převodník měřené veličiny na určitou úroveň napětí na datovém vodiči.

- Digitální vstup
  - V tomto případě modul počítá jen se dvěma logickými hodnotami.
- Analogový vstup
  - Princip spočívá v připojení interního AD převodníku na čtecí pin modulu. Modul pak skrze převodník získává informaci, jak intenzivně senzor danou veličinu detekuje.

Vícevodičové komunikace

Následující sběrnice fungují jako samostatné komunikační kanály s pokročilými senzory, které nejen provádí samostatné měření, ale i ukládají naměřené hodnoty do vlastních registrů přístupných skrze komunikační sběrnice. Typickou charakteristickou vlastností těchto sběrnic je použití časovače pro synchronizaci komunikace a tím i rozdělení zařízení na typy Master a Slave, kdy Master ovládá a řídí komunikaci a Slave jen odpovídá na pokyny. Velkou výhodou těchto typů sběrnic je možnost připojení více senzorů na nízký počet pinů a to díky adresaci jednotlivých povelů.

- I2C
  - Komunikace probíhá na dvou vodičích. Jeden zajišťuje časování pro synchronizaci komunikace a druhý pro přenos dat. Z toho je zřejmé, že komunikace probíhá ve formě Half duplex. Tato sběrnice navíc umožňuje použití více Masterů v jednom obvodu.
- SPI
  - Pro přenos dat jsou použity minimálně 4 vodiče. Kromě časovače jsou vyhrazeny 2 vodiče na komunikaci zajišťující Full duplex a minimálně jeden vodič jako Chip select pro aktivaci komunikace daného senzoru.

7.1.3 Datové typy

Výstupní hodnota může být uložena v několika datových typech. Pro dodržení jednoduchosti ovládání programu jsem se rozhodl, že omezím počet použitelných datových typů na minimum, aniž bych uživatele omezil ve výběru senzorů. Díky použití textových struktur pro komunikaci odpadá nutnost udržení menších datových typů (z hlediska velikosti paměti).

Senzorová část má tedy za úkol posbírat data ze senzorů a vložit je do tzv. senzorového bufferu. Pod tím si lze představit dostatečně velkou řadu bajtů definovanou jako jednodimenzionální pole charakterů, ačkoliv data uvnitř bufferu mohou odpovídat jinému datovému typu.

Pro senzorové hodnoty byly použity pouze tyto datové typy:

- int32
  - Celočíselná hodnota uložená v prvních 4 bajtech paměti bufferu s rozsahem - 2 147 483 648 až 2 147 483 647.
- float
  - Desetinné číslo uložené v prvních 4 bajtech paměti bufferu s rozsahem 1E-37 až 1E+37 (přesnost na 6 desetinných míst).
- string
  - Řada omezeného počtu charakterů (defaultní velikost bufferu je 50 bajtů, lze ji modifikovat). První bajt bufferu obsahuje 1bajtové číslo (uint8) určující počet použitých bajtů následující paměti pro uložení výsledku.
- bool
  - Jednobajtová hodnota formátu uint8 reprezentující hodnoty TRUE/FALSE. Jakákoliv nenulová hodnota pak odpovídá logické hodnotě TRUE.
- int3D
  - 3 int32 hodnoty uložené v prvních 12 bajtech bufferu.
- float3D
  - 3 float hodnoty uložené v prvních 12 bajtech bufferu.

7.1.4 Tvorba paketu

Ve chvíli, kdy je vyžádána tvorba paketu, se proměnná popisující stav odchozího paketu přepne na režim „busy“. To zajišťuje, že případná jiná funkce nezasáhne do paketu, když je v režimu tvorby. V Tx bufferu se automaticky vytvoří hlavička s adresou modulu a typem paketu. Podle zadaného typu paketu se pak spustí algoritmus procházející všechny senzory a plnící paket potřebnými daty.

V případě zvolení datového paketu se postupně projdou všechny senzory a, pokud jejich status není v režimu „vypnuto”, spustí se jejich funkce na přečtení dat ze senzoru a jejich zapsání do senzorového bufferu.

V případě informačního paketu se odešlou příslušná data bez ohledu na status senzoru.

7.1.5 Úprava výstupních hodnot

K přepsání dat ze senzorového bufferu do paketu si uživatel může sám naprogramovat funkce pro zápis do komunikačního Tx bufferu. Pro správnou konfiguraci těchto funkcí mohou uživateli pomoc funkce PaketOutAddName... , kde „...“ symbolizují datový typ hodnoty, se kterou se pracuje. Funkce jsou k nahlédnutí v souboru CmnProtTextV2.c ve složce XCmn.

```
PaketOutAddNameInt(char* name, int32_t value)
PaketOutAddNameFloat(char* name, float value, uint8_t decPlaces)
PaketOutAddNameBytes(char* name, uint8_t *value, uint8_t pos, uint8_t count)
PaketOutAddNameString(char* name, char *value, uint8_t pos, uint8_t count)
```

Na uživatelem nakonfigurované funkce pro zápis hodnot do Tx bufferu musí být zároveň vytvořen odkaz v nastavení senzorů. Více v kapitole Kompletní konfigurace. Pokud odkaz není vytvořen, je programem na základě konfigurace senzoru v nastavení senzorů vyhodnoceno použití předdefinovaných funkcí pro zápis do paketu.

7.1.6 Komunikační moduly

Jakmile je Tx buffer naplněn kompletním paketem volá se funkce pro odeslání dat. Ta podle vybraného komunikačního modulu zahájí přenos.

USART

Pomocí komunikačního protokolu USART docílíme přenosu dat mezi modulem a koncovým zařízením. Konfigurací modulu nastavíme komunikaci prostřednictvím kabelu s USB konektory sloužícím pro napájení i programování mikrokontroléru.

Bluetooth

Díky komunikačnímu modulu HC-06 připojenému přes dva signálové vodiče k senzorovému modulu získáme možnost bezdrátové komunikace. Procesor komunikuje s Bluetooth zařízením prostřednictvím protokolu UART a ke koncovému zařízení se data vysílají vzduchem skrze protokol Bluetooth.

Výstup dat

Na straně počítače se automaticky vytvoří viruální COM port sloužící jako platforma pro komunikaci mezi zařízeními. Aplikace koncového zařízení připojená na vytvořený port data analyzuje, zobrazí a uloží pro další zpracování.

7.2 Arduino

Výhodou práce s vývojovými deskami Nucleo a Arduino zároveň je společná podpora jazyka C. Tím vzniká možnost přenesení většiny programu z jednoho procesoru na druhý. Rozdíl kódů spočívá v použitých knihovnách specifických pro danou platformu. Práce knihoven HAL určených pro mikrokontroléry Nucleo se tedy musí vhodně substituovat knihovnami projektu Arduino.

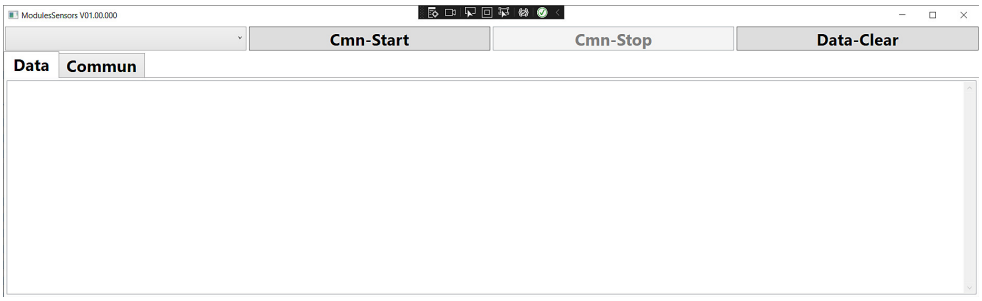
Zatímco v případě Nuclea je třeba zajistit veškerou konfiguraci pinů, sběrnic i ovládacích funkcí senzoru, v případě platformy Arduino stačí stáhnout knihovnu příslušného senzoru a vložit ji do svého kódu. Hlavní část konverze programu mezi platformami jsou funkce obsluhující komunikační moduly.

7.3 Aplikace koncového zařízení

Dle zadání má počítačová aplikace zobrazovat data a ukládat je pro další použití.

7.3.1 Inicializace

Při spuštění programu dojde k analýze všech aktivních COM portů počítače. V zápětí se objeví na popředí obrazovky aktivní okno aplikace

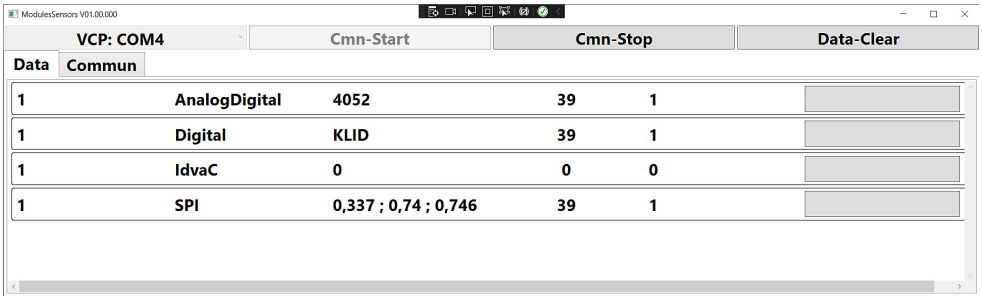


Obrázek 7.2: Aplikace koncového zařízení

V horní části okna se nachází programová lišta sloužící jako hlavní ovládací panel aplikace. Je složena ze čtyř částí. Na levém okraji ovládacího panelu je vygenerován rozbalovací seznam pro výběr virtuálního portu. Následují ovládací tlačítka.

7.3.2 Komunikace

Po výběru komunikačního portu a aktivaci přenosu dat tlačítkem Cmn-Start si aplikace automaticky vyžádá informační pakety od připojených modulů prostřednictvím zpětnovazebního CMD paketu. Při úspěšném příjmu odpovědi modulu ve formě informačního paketu se na základě získaných dat automaticky vytvoří objekty pro uchování senzorových dat a vygeneruje se tabulka pro zobrazení naměřených hodnot.



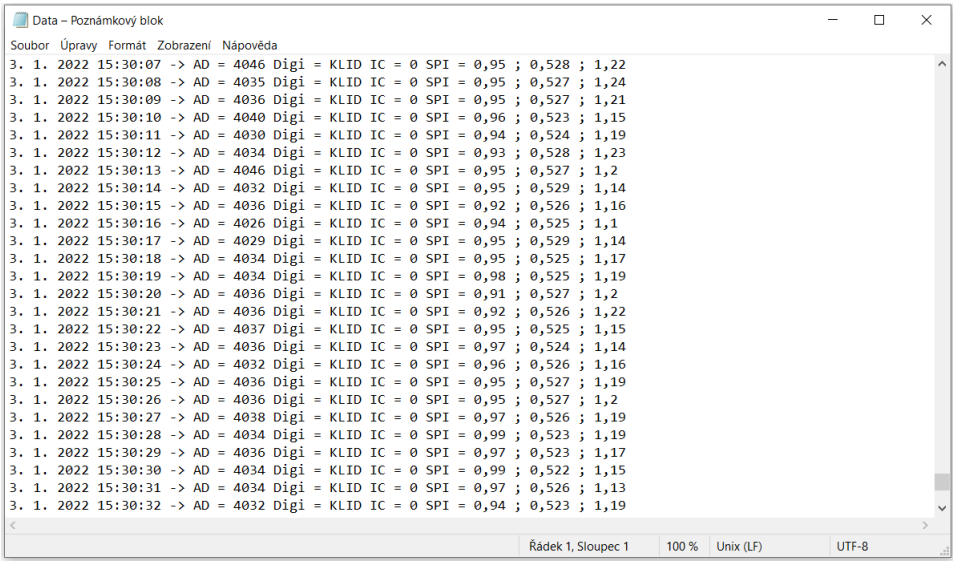
Obrázek 7.3: Aplikace se senzory

Řádek tabulky se skládá z šesti částí. Za ID modulu se nachází jméno senzoru a naměřená hodnota. Následuje počet získaných hodnot v konkrétní relaci a stav (status) senzoru. Na pravé straně se nachází tlačítko, jehož úkolem je přepnutí stavu senzoru on/off.

Modul na základě přijaté výzvy, kromě poskytnutí informací o připojených senzorech, zapíná automatické odesílání datových paketů v závislosti na vnitřní konfiguraci.

7.3.3 Ukládání dat

Získaná data aplikace automaticky ukládá do textového souboru. Umístění souboru je v adresáři projektu ModuleSensors/bin/Debug/netcoreapp3.1/Data.txt



Obrázek 7.4: Aplikace koncového zařízení

Kapitola 8

Struktura dat pro komunikaci

Návrh struktury dat pro komunikaci je příhodné popsat jednoduše, proto struktura nese název Paket. (Pozn. autora: Autor si je vědom faktických rozdílů mezi vytvořenou strukturou dat a definicí slova paket.)

8.1 Obecná struktura

Pro zachování jednoduchosti přenosu dat a pro čitelnost zachycených paketů mezi modulem a počítačovou aplikací je vhodnější strukturu dat udržet ve formátu řady charakterů (String na straně počítačové aplikace).

Všechny pakety mají jednotný formát. Ohraničení paketů je zajištěno hranatými závorkami „ [“, „] “. Oddělení jednotlivých částí zajišťují mezery „ “. Další případné dělení částí je tvořeno znaky „ = “ a „ ; “ podle situace.

První část paketu tvoří identifikační číslo modulu. Formátem je dvojciferné identifikační číslo nakonfigurované před zapnutím modulu.

Druhá část nese označení typu paketu. To je zajištěno předdefinovanou trojicí charakterů zkracující název druhu.

Další části nesou samostatná data vybraná podle typu paketu.

```
[ID TYP DATA1=METADATA1;METADATA2 DATA2=METADATA1;METADATA2 ]
```

- ID = Dvouciferné identifikační číslo modulu
- TYP = 3 znaky charakteritující typ paketu
  - INF - Informační paket nesoucí data o senzorech
  - ANS - Datový paket pro přenášení změřených hodnot
  - CMD - Příkazový paket umožňující zpětnou vazbu
- DATA = Vnitřní informace paketu
- METADATA = Doplňující informace

8.2 Dělení paketů

Rozdělení paketů dle typu dat, které přenášejí.

8.2.1 INF – Informační paket

Informační paket, jak již název napovídá, slouží k informování aplikace v koncovém zařízení o konkrétních parametrech jednotlivých senzorů modulu.

Je volán před zapnutím pravidelného zasílání dat pro vytvoření tabulky podle počtu a datového typu senzorů. Zároveň k jeho použití dochází v případě změny statusu senzoru. K tomu může dojít v případě, že se z aplikace vzdáleně vypne odesílání dat jednoho ze senzorů.

Data jednoho senzoru pak nesou následující formát:

```
[01 INF ID=Status;Name;DatType]
```

- ID = 1 - 8 charakterů pro identifikaci senzoru
- Status = celé číslo označující momentální status senzoru
- Name = 1 - 20 charakterů pro pojmenování senzoru v tabulce
- DatType = celé číslo označující datový typ hodnoty senzoru

Konkrétní INF paket pak může vypadat následovně:

```
[01 INF AD=1;AnalogDigital;2 Digi=1;Digital;4]
```

8.2.2 ANS – Datový paket

Datový paket přenáší data změřená senzory po případné úpravě nakonfigurované uživatelem.

Data jednoho senzoru pak nesou následující formát:

```
[01 ANS ID=Status;Data]
```

- ID = 1 – 8 charakterů pro identifikaci senzoru
- Status = celé číslo označující momentální status senzoru
- Data = Hodnoty ze senzoru v předdefinovaném datovém typu

Konkrétní ANS paket pak může vypadat následovně:

```
[01 ANS AD=1;3288 Digi=1;POHYB]
```

8.2.3 CMD - Příkazový paket

Pro umožnění vzdáleného ovládání modulu z aplikace je potřeba zajistit komunikaci i v tomto směru. Prostřednictvím příkazového paketu bude dosaženo požadované zpětné vazby.

Data jednoho senzoru pak nesou následující formát:

```
[01 CMD C=COM S=SENS]
```

COM = 1 - 8 charakterů pro identifikaci senzoru SENS = celé číslo označující momentální status senzoru

Z legendy je zřejmé číselné zastoupení jednotlivých příkazů. V případě, že použitý příkaz potřebuje dodatečné informace (například číslo senzoru, pro který se má příkaz vykonat) obsahuje paket i část SENS, která se jinak v paketu neobjeví.

Konkrétní CMD paket pak může vypadat následovně:

```
[01 CMD C=5 S=2]
```

nebo

```
[01 CMD C=2]
```

Kapitola 9

Kompletní konfigurace

9.1 Nucleo

Jak již bylo zmiňováno v předchozím textu, součástí práce je návod, jak modul nakonfigurovat. Tomu se bude věnovat tato kapitola.

V projektu jsou vložené vzorové soubory již nakonfigurovaných senzorů, podle kterých se dá jednoduše sestavit konfigurace senzorů vlastních.

Důležité je na začátku všech souborů .c přidat odkaz na hlavní knihovnu všech .h souborů a všechny .h soubory přidat do téže knihovny.

```
#include "Def0Global.h"
```

9.1.1 Nastavení sběrnic (inicializace)

Konfigurace senzoru v programu je vložena prostřednictvím samostatných souborů. Tedy každý senzor má vlastní .c a .h

Předpokládá se vložení těchto souboru do adresáře „Config". Tento adresář je určen pro soubory upravované uživatelem. Až na malé výjimky by se měl uživatel obejít bez úpravy jiných souborů.

První funkce nutná pro správný běh senzoru je funkce inicializační. Na začátku inicializace je ideální prostor pro konfiguraci GPIO pinů pro použitý senzor. Pro zjištění vhodných pinů použitelných pro požadovanou sběrnici je třeba si nejdříve zjistit, které piny podporují potřebnou periferii. Mé osobní doporučení je najít si „pinout" na stránkách mbed.com. Dle mých zkušeností jsou jejich pinové výkresy nejpřehlednější.

Jako příklad je v následujících řádcích popsán vzorový soubor pro senzor s názvem SensorAD.c /.h.

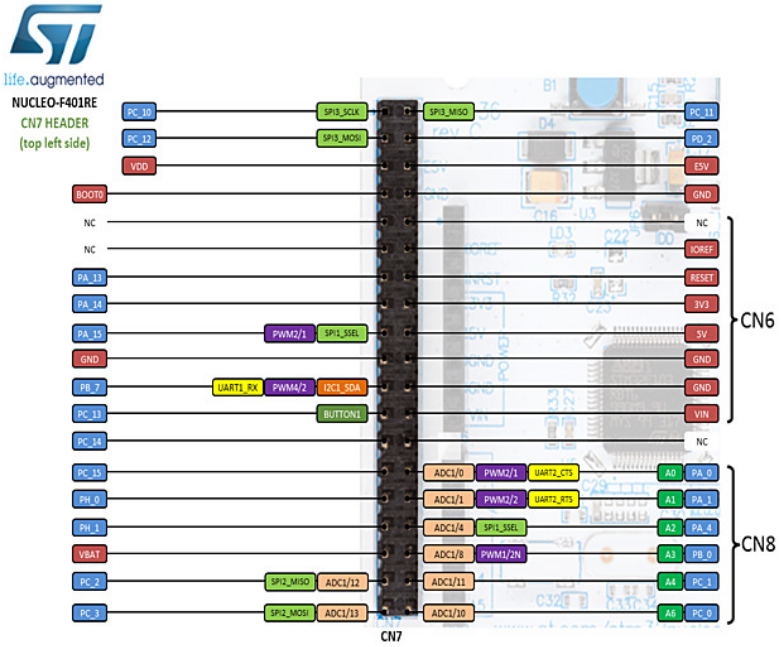
Byl zvolen senzor s jedním pinem pro přenos dat, pomocí AD převodníku, je potřeba tedy najít AD převodník jako periferii jednoho z pinů.

Dle obrázku lze zvolit hned několik variant. V tomto případů jde o pin PA_1, tedy Port A pin 1.

V .h souboru je vytvořena struktura GPIO pro HAL knihovny.

```
GPIO_InitTypeDef  cGPIOAD;
```

V .c souboru pak nastavena samotná konfigurace struktury.



Obrázek 9.1: Nucleo-F401RE PinOut

```
//Inicializase Pinu
cGPIOAD.Mode  = GPIO_MODE_ANALOG;     //Použití periferie AD převodníku
cGPIOAD.Pull  = GPIO_NOPULL;          //Definice typu vstupu pinu
cGPIOAD.Speed = GPIO_SPEED_FREQ_HIGH; //Nastavení rychlosti snímání dat
cGPIOAD.Pin   = GPIO_PIN_1;           //Určení pinu
__HAL_RCC_GPIOA_CLK_ENABLE();         //Zapnutí časovače na portu A
HAL_GPIO_Init(GPIOA, &cGPIOAD);       //Zapnutí nastaveného pinu na portu A
```

Tím došlo k aktivaci pinu pro komunikaci a přiřazení periferie AD převodníku.

Nyní je třeba nastavit samotný AD převodník. K tomu je určena struktura Handler HAL knihovny. V .h souboru je implementována struktura ADC Handler pro HAL knihovny.

```
ADC_HandleTypeDef hSensorAD;
```

V .c souboru pak nastavena samotná konfigurace struktury.

```
__HAL_RCC_ADC1_CLK_ENABLE(); //Zapnutí časovače pro periferii ADC1
hSensorAD.Instance = ADC1; //Periferie, pro kterou je Handler použitý
```

Další nastavení dle použitého senzoru:

```
hSensorAD.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
hSensorAD.Init.Resolution = ADC_RESOLUTION_12B;
```

```
hSensorAD.Init.DataAlign = ADC_DATAALIGN_RIGHT;
hSensorAD.Init.EOCSelection = ADC_EOC_SINGLE_CONV;

if (HAL_ADC_Init(&hSensorAD) != HAL_OK)
```

Samotná HAL funkce pro inicializaci AD převodníku. Podmínka je přidaná pro detekci chybné inicializace. Tělo podmínky tedy má obsahovat algorytmus spuštěný při vzniklém problému.

V případě AD převodníku je třeba nakonfigurovat kanál. V tomto případě je použit kanál 2, nicméně kanály se číslují od nuly, proto „ADC_CHANNEL_1".

```
ADC_ChannelConfTypeDef cSensorAnalogInChannel;
cSensorAnalogInChannel.Channel = ADC_CHANNEL_1;
cSensorAnalogInChannel.Rank = 1;
cSensorAnalogInChannel.SamplingTime = ADC_SAMPLETIME_28CYCLES;

if (HAL_ADC_ConfigChannel(&hSensorAD, &cSensorAnalogInChannel) != HAL_OK)
```

Opět funkce na spuštění kanálu s odkazem na Handler a podmínkou pro ošetření chybné inicializace. Inicializační funkce pro digitální vstup je jen GPIO část, neboť pro čtení hodnoty 0/1 není třeba žádných periferií.

Inicializace I2C a SPI obsahují část GPIO a nastavení samotných periferií, tak jako tomu je u AD převodníku. Kanál není potřeba, tedy se nekonfiguruje. Pro senzory s těmito složitějšími sběrnicemi jsou ale typicky charakteristické inicializační funkce nutné pro konfiguraci samotného senzoru například nastavení módu měření, přesnosti dat apod. Toto nastavení by mělo být součástí inicializační funkce, která se programuje pomocí funkcí vysvětlených v následující podkapitole.

Funkce pro inicializaci senzoru tedy může vypadat následovně:

- SensorI2C.h

```
#define CHIP_ADDR (uint16_t)0x76    // Adresa senzoru
GPIO_InitTypeDef cGPIOI2C;          // struktura GPIO
I2C_HandleTypeDef HSensorI2C;       // struktura Handler
```

- SensorI2C.c

```
void InitSensorI2C(){

...             // Inicializace GPIO

...             // Inicializace I2C

    HAL_StatusTypeDef sensorI2CDone = HAL_OK;
    uint8_t value = 0x00;
    sensorI2CDone = HAL_I2C_Mem_Read
        (&HSensorI2C, CHIP_ADDR << 1, 0xD0, 1, &value, 1, 1000);

                // Přečtení hodnoty z registru 0xD0 do value

    if ((sensorI2CDone != HAL_OK)||(value != 0x58))

                // Pokud je senzor aktivní value == 0x58

    {
                 // ERROR
    }

    value = 0xB6;
    sensorI2CDone = HAL_I2C_Mem_Write
        (&HSensorI2C, CHIP_ADDR << 1, 0xE0, 1, &value, 1, 1000);

                // Zápis do registru 0xE0, nastavení módu měření

    if (sensorI2CDone != HAL_OK)

                // Kontrola zda zápis proběhl dle očekávání

    {
                // ERROR
    }

}
```

9.1.2 Nastavení senzorů

Další důležitou funkci v senzorových souborech jsem nazval GetData a má dva úkoly. Prvním úkolem je získat data ze senzoru, druhým je zapsání dat do senzorového bufferu. Každá sběrnice má odlišný postup a princip sbírání dat, proto je rozeberu všechny. Tato část obsahuje mnoho funkcí z knihovny HAL, které se pokusím dostatečně vysvětlit.

Digitální vstup

V případě digitálního vstupu prakticky stačí jen zjistit, jaká hodnota je připojena na datovém pinu.

```
HAL_GPIO_ReadPin(DigiGPIO , GPIO_InitStructDigi.Pin)
```

Tato funkce umožňuje přečíst stav pinu. Prvním argumentem je GPIO struktura z inicializační funkce a druhým argumentem je samotný pin, ze kterého informaci čtu. Výstup funkce můžeme porovnat s „GPIO_PIN_SET", což substituuje logickou 1.

Analogový vstup

V případě AD převodníku nejdříve spustíme AD konverzi příkazem:

```
HAL_ADC_Start(&hSensorAD);
HAL_ADC_PollForConversion(&hSensorAD, 1000);
```

Argumentem funkce je odkaz na strukturu Handler z inicializační funkce, druhý argument je doba čekání na odpověď v milisekundách. Změřenou hodnotu pak načteme příkazem:

```
HAL_ADC_GetValue(&hSensorAD);
```

I2C

I2C komunikace probíhá prostřednictvím čtení a zapisování dat přímo do registrů senzoru skrze jeho adresu. Pro komunikaci po sběrnici I2C jsou použity níže uvedené funkce.

```
HAL_I2C_Mem_Write(&HSensorI2C, CHIP_ADDR << 1, 0xE0, 1, &value, 1, 1000);
HAL_I2C_Mem_Read(&HSensorI2C, CHIP_ADDR << 1, 0xFA, 1, &data, 3, 1000);
```

Jak již jejich název napovídá, Write slouží pro zápis dat do a Read pro čtení dat z registrů senzoru. Prvním argumentem je odkaz na strukturu Handler, druhým je adresa senzoru posunutá o jeden bit doleva, třetím adresa registru pro zápis/čtení, čtvrtým velikost paměti, pátým odkaz na buffer pro data, šestým velikost zapisovaných/čtených dat a sedmým doba (v ms), po kterou se má modul snažit navázat spojení.

SPI

SPI komunikace sice nepotřebuje adresy senzorů, ale místo nich využívá princip Chip Select. Každý senzor má tedy vlastní vodič zajišťující otevření komunikace s daným senzorem. Principem komunikace je okamžitá odpověď na přijatý požadavek. Pro čtení dat tedy nejdříve musíme aktivovat daný senzor. V mém případě to znamená přivést na PB_10 logickou 0.

```
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10, GPIO_PIN_RESET);
```

Prvním argumentem je Port použitého pinu, druhým pak číslo pinu. Na třetím místě je cílený stav, logická 1/0, tedy GPIO_PIN_SET / GPIO_PIN_RESET.

```
HAL_SPI_TransmitReceive(&HSensorSPI, rxBuf, txBuf, 1 + count, 1000);
```

Tato funkce zajišťuje komunikaci se senzorem. Prvním argumentem je struktura Handler, druhým odkaz na buffer pro přijatá data, třetím pak odkaz na buffer pro vysílaná data, čtvrtým je velikost přenášených dat (důležité je nezapomenout, že se jedná o komunikaci Half duplex, tedy musím sečíst velikost odesílaných a velikost přijímaných dat, pátým pak čas, jak dlouho se má modul snažit o spojení.

Senzam senzorů

Program obsahuje jeden konfigurační soubor, ve kterém je seznam všech použitých senzorů včetně odkazů na důležité funkce. SensorConfig.c / .h V hlavičkovém souboru je nutné nastavit správný počet použitých senzorů. Ve zdrojovém kódu se nachází definice seznamu (struktura) senzorů složená z několika částí:

```
typedef struct {
    char Name[20];
    char ID[8];
    EnSensStatus Status;
    EnDataType DataTypeSensor;
    EnDataType DataTypePaket;
    _SensInit Init;
    _SensGetData GetData;
    _SensValueConv ValueConv;
} SensBase;
```

- Name

  Celé jméno senzoru, které se objeví v tabulce počítače

- ID

  Originální kombinace znaků, pro identifikaci v rámci komunikace

- Status

  Enum odpovídající statusu senzoru

- DataTypeSensor

  Datový typ hodnoty v senzorovém bufferu

- DataTypePaket

  Datový typ hodnoty v paketu

- Init

  Odkaz na inicializační funkci

- GetData

  Odkaz na fuknci pro získání dat

- ValueConv

  Odkaz na funkci pro psaní dat do paketu vytvořenou uživatelem

Správně zapsaný senzor pak vypadá třeba následovně:

```
{"Digital", "Digi\0x00", EnOn, Enbool, Enstring,
InitSensorDigi, GetDataSensorDigi, DataConvDigi}
```

```
#include "Def0Global.h"

SensBase SensorConfig[Sensor_Count] =
{
        // Sensor (AD)
        {"AnalogDigital", "AD\0x00", EnOn, Enint32, Enint32, InitSensorAD, GetDataSensorAD, NULL},

        // Sensor (Digi)
        {"Digital", "Digi\0x00", EnOn, Enbool, Enstring, InitSensorDigi, GetDataSensorDigi, DataConvDigi},

        // Sensor (I2C)
        {"IdvaC", "IC\0x00", EnOff, Enint32, Enint32, InitSensorI2C, GetDataSensorI2C, NULL},

        // Sensor (SPI)
        {"SPI", "SPI\0x00", EnOn, Enfloat3D, Enfloat3D, InitSensorSPI, GetDataSensorSPI, NULL},

        // Další pomocné konfigurace senzorů

        // Sensor (Int)
//{"Integer", "INT\0x00", EnOn, Enint32, Enbyte, InitSensor1, GetDataSensor1, NULL},

        // Sensor (Float)
//{"Float", "FLT", EnOn, Enfloat, Enbyte, InitSensor2, GetDataSensor2, NULL},

        // Sensor (Byte)
//{"Byte", "BYTE", EnOn, Enbyte, Enbyte, InitSensor3, GetDataSensor3, NULL},

        // Sensor (String)
//{"String", "STR", EnOn, Enbyte, Enstring, InitSensor3, GetDataSensor3, NULL},
```

Obrázek 9.2: Konfigurace senzorů

Status

Jak již bylo zmíněno, položka status uchovává stav senzoru.

```
typedef enum{
    EnOff = 0,
    EnOn = 1,
    EnErr = 2
} EnSensStatus;
```

- EnOff

  Stav odpovídající poloze vypnuto se používá v případě, že senzor nepoužíváme. Senzor s tímto statusem je ignorován při tvorbě datového paketu a jeho hodnoty tedy nejsou odeslány.

- EnOn

  Senzor ve stavu zapnuto je aktivní a data z něj se odesílají.

- EnErr

  Status odpovídající chybě. Lze využít jako identifikaci chybné inicializace, či čtení dat.

Datové typy

Správně zapsané datové typy jsou důležité, aby modul byl schopný data řádně zpracovat a vložit do paketu.

```
typedef enum{
    Enbyte = 0,
    Enbool = 1,
    Enint32 = 2,
    Enfloat = 3,
    Enstring = 4,
    Enint3D = 5,
    Enfloat3D = 6
} EnDataType;
```

DataTypePaket je zároveň odesílán prostřednictvím INFO paketu. Jeho úkolem je správné vytvoření tabulky pro zobrazení hodnot v koncovém zařízení.

Konverzní funkce

Pro zapsání dat ze senzorového bufferu do paketu se používají předdefinované funkce. Uživatel si ale tyto funkce může nakonfigurovat sám a tím tak změnit typ dat v paketu na základě snímaných dat.

Jedna ze vzorových funkcí je nakonfigurována pro SensorDigi v jeho .c souboru. Jedná se o detektor pohybu. Tato funkce na základě dat v senzorovém bufferu (logická 0/1) propíše do paketu „KLID" / „POHYB".

Všechny následující funkce mají společný první argument „name". Tento argument se používá pro vytvoření nové senzorové části v paketu. Vzhledem k tomu, že program sestavuje tyto části samostatně, měl by uživatel vždy použít pro první argument „NULL".

Druhý argument typicky obsahuje hodnotu daného datového typu. Tato hodnota se tedy skutečně propíše to paketu.

- INT:

```
PaketOutAddNameInt(char* name, int32_t value)
```

- Float:

```
PaketOutAddNameFloat(char* name, float value, uint8_t decPlaces)
```

Třetí argument udává počet použitých desetinných míst.

V následujících funkcích obsahuje druhý argument hodnotu formou odkazu na buffer.

Třetí argument obsahuje pozici v bufferu, od které jsou platné hodnoty.

Čtvrtý argument pak počet hodnot bufferu pro zapsání do paketu.

- Bytes:

```
PaketOutAddNameBytes(char* name, uint8_t *value, uint8_t pos, uint8_t count)
```

- String:

```
PaketOutAddNameString(char* name, char *value, uint8_t pos, uint8_t count)
```

Nastavení komunikace

Za konfiguraci komunikací je zodpovědný soubor CommunConfig.c / .h.

V hlavičkovém souboru se nastavuje identifikační číslo modulu a frekvence odesílání dat.

```
#define Modul_ID            1   // Adresa (ID) modulu
```

ID se musí změnit, pokud síť obsahuje více než jeden modul a musí být pro každý modul v síti originální. Identifikační číslo musí být v rozsahu 1 - 99.

```
#define Modul_DataSendTime  1000 // Perioda odeslání dat v ms
```

Frekvence automatického odesílání dat je nastavitelná pomocí periody (v ms). Soubor dále obsahuje pomocné nastavení pro zjednodušení ladění komunikací.

Soubor se zdrojovým kódem obsahuje funkce pro inicializaci a odeslání dat. Výběr použité komunikace se provede zakomentováním všech ostatních.

```
void CmnInit(){
    CmnTypeUSARTInit();             // inicializace USART
    //CmnTypeBTInit();              // inicializace Bluetooth
}


void CmnSend(){
    CmnTypeUSARTSend(NULL, 0);      // Odeslání dat UART
    //CmnTypeBTSend(NULL, 0);       // Odeslání dat Bluetooth
}
```

V případě předchozí konfigurace je tedy vybrána komunikace pomocí USART.

Kapitola 10

Závěr

V této bakalářské práci byly zhodnoceny některé komerčně dostupné možnosti pro vývoj experimentálních hraček a rehabilitačních pomůcek. Na základě provedené rešerše byl navržen postup sestavení programu pro snadnou tvorbu experimentálních senzorových zařízení. V rámci návrhu řešení byl sestaven jednotný protokol a struktura pro přenos dat bez ohledu na použitou komunikační technologii. Zdrojový kód byl naprogramován pro procesory ARM s konfigurací základních sběrnic (ADC, I2C, SPI) pro připojení senzorů s možností snadné implementace dalších sběrnic. Do programu byly plně implementovány komunikační kanály (USART a Bluetooth) umožňující přenos dat do koncového zařízení.Program byl navržen pro snadné rozšíření o další senzorové sběrnice i komunikační kanály. Pro zobrazení dat a jejich další zpracování byla naprogramována multiplatformní desktopová aplikace pomocí jazyka C# (.net core). V rámci práce je dostupný i návod pro nastavení programu a snadnou konfiguraci senzorů či komunikačních protokolů včetně vzorových souborů. Pro demonstrační účely vznikl senzorový modul s procesorem ARM využívající všechny implementované sběrnice.

Výsledkem práce jsou funkční knihovny pro jednoduché sestavení experimentálního senzorového modulu s desktopovou aplikací, jednoduchým návodem a možností snadného rozšíření. Výsledek práce je tedy vhodný pro použití v domácím či laboratorním prostředí jako rychlá pomoc při testování nových senzorů, komunikací nebo částí projektů.

Dalším krokem po úplném dokončení projektu dle zadání může být rozšíření podporovaných sběrnic, optimalizace programu modulu z hlediska úspory energie a implementace bateriového napájení. Za pokročilé možnosti zdokonalení lze považovat šifrování komunikace či možnost tvorby senzorových sítí prostřednictvím metody Daisy chain. Do aplikace koncového zařízení by v budoucnu mohla být přidána funkce tvoření grafů či sledování trendů snímaných hodnot, nebo sdílení dat pomocí cloudu.

Vzhledem k časové náročnosti projektu se mi nepodařilo realizovat plnou portaci programového vybavení senzorového modulu na platformu Arduino a s tím případně i další komunikační kanály. Celková práce mi zabrala mnohem více času, než jsem prvotně předpokládal. Bylo nutno prostudovat větší množství materiálů a vytvořit více programového kódu, než se na první pohled zdálo.

Z důvodu přecenění svých schopností v oblasti programování jsem nebyl schopen realizovat zadanou mobilní aplikaci. Tedy i přes mou veškterou snahu a současně výpomoc vedoucího práce se nepodařilo tento typ aplikace dotáhnout do zdárného konce.

Literatura

[1] NOVIELLO, Carmine *Mastering STM32 [PDF]*, 2018 Itálie, Leanpub

[2] OWSIAK, Tom *Beginning C 7 HAnds-On - The Core Language [PDF]*, 2017, Packt, ISBN-13: 978-1788294263

[3] LIBERTY, Jesse *Windows 10 Development with XAML and C 7*, 2017, Apress, ISBN-13: 978-1484229330

[4] HAL *User Manual*, https://www.st.com/content/ccc/resource/technical/document/user_manual/97/4d/5f/9a/ed/e4/4e/66/DM00132099.pdf/files/DM00132099.pdf/jcr:content/translations/en.DM00132099.pdf, [Online, navštíveno 3. 1. 2020]

[5] STM32 Nucleo-F401RE *User Manual*, 2021 https://www.st.com/resource/en/data_brief/nucleo-f401re.pdf [Online, navštíveno 3. 1. 2020]

[6] STM32CubeIDE, https://www.st.com/en/development-tools/stm32cubeide.htmloverview, [Online, navštíveno 3. 1. 2020]

[7] Getting Started with STM32 https://www.digikey.com/en/maker/projects/getting-started-with-stm32-introduction-to-stm32cubeide/6a6c60a670c447abb90fd0fd78008697 [Online, navštíveno 3. 1. 2020]

[8] JUŘÍK, David. *Experimentální senzorové moduly pro tvorbu hraček a domácího monitorování*[online]. Praha, 2020. Bakalářská práce. České vysoké učení technické, Fakulta elektrotechnická, Katedra kybernetiky. Ing. Petr Novák, Ph.D. Dostupné z: https://dspace.cvut.cz/handle/10467/87605

CITACE OBRÁZKŮ:

[9] Lego Mindstorms *Robot-Advance* [online]. Francie: A2O Distribution, 2019 [cit. 2022-1-3]. Dostupné z: https://www.robot-advance.com/userfiles/www.robot-advance.com/images/lego-mindstorms-NXT-education.jpg

[10] LittleBits Core-Electronics [online]. Australia: Core Electronics Pty Ltd, 2019 [cit. 2022-1-3]. Dostupné z: https://core-electronics.com.au/media/catalog/product

[11] STM32 Arm-Cortex Deep Blue [online]. Egypt: Deep Blue, 2018 [cit. 2022-1-3]. Dostupné z: https://deepbluembedded.com/getting-started-with-stm32-arm-cortex-mcus

[12] Arduino family Predictable Designs [online]. USA: Predictable Designs LCC, 2021 [cit. 2022-1-3]. Dostupné z: https://predictabledesigns.com/how-to-choose-the-best-development-kit-the-ultimate-guide-for-beginners/

[13] Nucleo family.Code Inside Out [online]. USA: Code Inside Out 2020 [cit. 2022-1-3]. Dostupné z: https://www.codeinsideout.com/blog/stm32/intro/

[14] Blokové schéma měřícího systému David Juřík [online]. Praha: Juřík David, Experimentální senzorové moduly pro tvorbu hraček a domácího monitorování, 2020 [cit. 2022-1-3]. Dostupné z: https://dspace.cvut.cz/handle/10467/87605

[15] STM32 Nucleo-F401RE PinOut *ARM Holdings* [online]. USA: ARM Holdings, 1990 [cit. 2022-1-3]. Dostupné z: https://os.mbed.com/platforms/ST-Nucleo-F401RE/