VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

VZOROVÉ PŘÍKLADY V PROSTŘEDÍ ARMMBED

MODEL EXAMPLE IN ARMMBED ENVIRONMENT

AUTOR PRÁCE — Patrik Maraczek
AUTHOR

VEDOUCÍ PRÁCE — Ing. Daniel Zuth, Ph.D.
SUPERVISOR

BRNO 2017

ZADÁNÍ VŠKP 1

(tento list nahradíte oficiálním zadáním práce)

ABSTRAKT

Bakalářská práce obsahuje základní informace pro práci s mikrokontrolérem v prostředí ARMmbed. V práci je popsán mikrokontrolér, periferie čipu a práce obsahuje návod na obsluhu prostředí ARMmbed. Vše je následně ukázáno a vysvětleno na vzorových příkladech pro programování mikrokontroléru. Příklady se zabývají čtyřbitovým sedmisegmentovým displejem, LED diodami, sériovou komunikací a ultrazvukovým senzorem pro měření vzdálenosti SRF02.

ABSTRACT

The bachelor's thesis contains basic information for working with microcontroller properly in ARMmbed integrated development environment. Thesis describe microcontroller, microchips, chip's periphery, ARMmbed environment. Also manual is written for this environment. Four examples are solved at the end of thesis to explain how extended devices and sensors are working. Devices like 4-bit seven-segmented display, LED diodes, ultrasound sensor SRF02.

KLÍČOVÁ SLOVA

ARM, ARMmbed, mbed, STM, NUCLEO, vývojová deska, mikrokontrolér, mikročip, NUCLEO-F401RE, SRF02, čtyřbitový sedmisegmentový displej, LED, dioda

KEYWORDS

ARM, ARMmbed, mbed, STM, NUCLEO, development board, microcontroller, microchip, NUCLEO-F401RE, SRF02, four-bit seven-segmented display, LED, diode

BIBLIOGRAFICKÁ CITACE

MARACZEK, Patrik. *Vzorové příklady v prostředí ARMmbed*, Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky.

PODĚKOVÁNÍ

Děkuji tímto Ing. Danielu Zuthovi, Ph.D. za cenné připomínky a rady při vypracování této bakalářské práce.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. Daniela Zutha, Ph.D. a s použitím literatury uvedené v seznamu literatury.

V Brně dne 25. 5. 2017 ……

Patrik Maraczek

OBSAH

1 ÚVOD

Pro programování v oblastech automatizace, robotiky a průmyslové robotiky se v současné době často používají mikrokontroléry, které se programují v JSA (jazyk symbolických adres) nebo ve vyšších jazycích, např. C, Pascal, Matlab. Assembler převádí JSA na strojový kód. Programování v JSA je časově náročnější a složitější, takže se nejčastěji používají vyšší programovací jazyky, které jsou přehledné, efektivní a méně pracné. Praktické využití těchto řídících systémů je široké a téměř neomezené. Využívají se jak v domácnosti (automatická pračka, myčka, alarmy, hračky, apod.), tak v náročných aplikacích v leteckém a automobilovém průmyslu, strojírenství a dalších odvětvích. [1]

Oba principy mají jedno společné - zdrojový kód se píše na PC, kde je následně zkontrolován a přeložen. Přeložený kód se nahraje do flash paměti mikrokontroléru, pak dojde k resetu a další běh je již podle nově nahraného programu.

Tato práce se zabývá programy v jazyce C++ ve vývojovém prostředí ARMmbed.

1.1 Mikrokontrolér

Mikrokontrolér je minimalizovaný systém řídícího mikropočítače, který se začal vsazovat do jednoho čipu. Z čipu tak vznikla samostatná jednotka, která pro svoji funkci nepotřebuje další hardware. To je velký rozdíl oproti klasickým mikročipům, které ke své činnosti často potřebují velké množství dalších podpůrných obvodů. Čip je osazen mikroprocesorem, operační pamětí (RAM), pevnou pamětí (ROM), vstupními a výstupními obvody. Dále obsahuje periferie, časovač (synchronizace s reálným světem), A/D převodník, apod. [2]

Na rozdíl od klasického osobního počítače, u mikrokontroléru není zapotřebí vysoký výkon, protože se mikrokontrolér nasazuje z důvodu zjednodušení výroby a tím pádem snížení ceny. Není důvod nasazovat výkonnější mikrokontrolér, pro aplikaci, kterou zvládne řádově slabší mikrokontrolér. Z výše zmíněných důvodů firmy nabízí široký rozsah konfigurací mikrokontrolerů, lišících se v kapacitě pamětí (ROM, RAM), počtem vstupů, výstupů a použitelných periferií. [2]

1.2 Historie mikrokontroléru

Největšího aplikačního rozšíření mikropočítače přinesly firmy Intel a Motorola. Lze však zmínit i jiné firmy z velice širokého spektra výrobců mikročipů, jako jsou Toshiba Microchip, Atmel, Texas Instruments, National Semiconductors, Siemens, Philips, apod. Z historického hlediska řady Intel je patrná snaha o zvýšení kapacity pevné paměti a paměti RAM a o zrychlení provádění instrukcí. První mikroprocesor od firmy Intel vznikl v roce 1971. Pro ilustraci lze v Tab. 1 vidět rozdíly v historicky prvních mikropočítačích, vzniklých v letech 1976 – 1983. [2]

Tab. 1: Srovnání historicky vzniklých mikrokontrolérů [2]

Řada Intel MCS-48	Řada Intel MCS-51	Řada Intel MCS-96
(integrováno na čipu)	(integrováno na čipu)	(integrováno na čipu)
8bitová CPU	8bitová CPU	16bitová CPU
1/2/4 kB ROM	4/8 kB ROM	až 32kB ROM
64/128/256 bytů RAM	128/256 bytů RAM	až 744 bytů RAM
1 čítač/časovač	2 čítače/časovače	2 čítače/časovače a hlídací časovač (watchdog)
paralelní vstup/výstup	paralelní vstup/výstup	až 64 bitů číslicových vstupů/výstupů
8bitový převodník A/D	seriový vstup/výstup	až 5 seriových linek vstupu/výstupu
	až 6 zdrojů přerušení	A/D převodník 8/10 bitů s multiplexerem
		jednotka zpracování událostí
		jednotka zrychlené obsluhy přerušení
		až 14 zdrojů přerušení

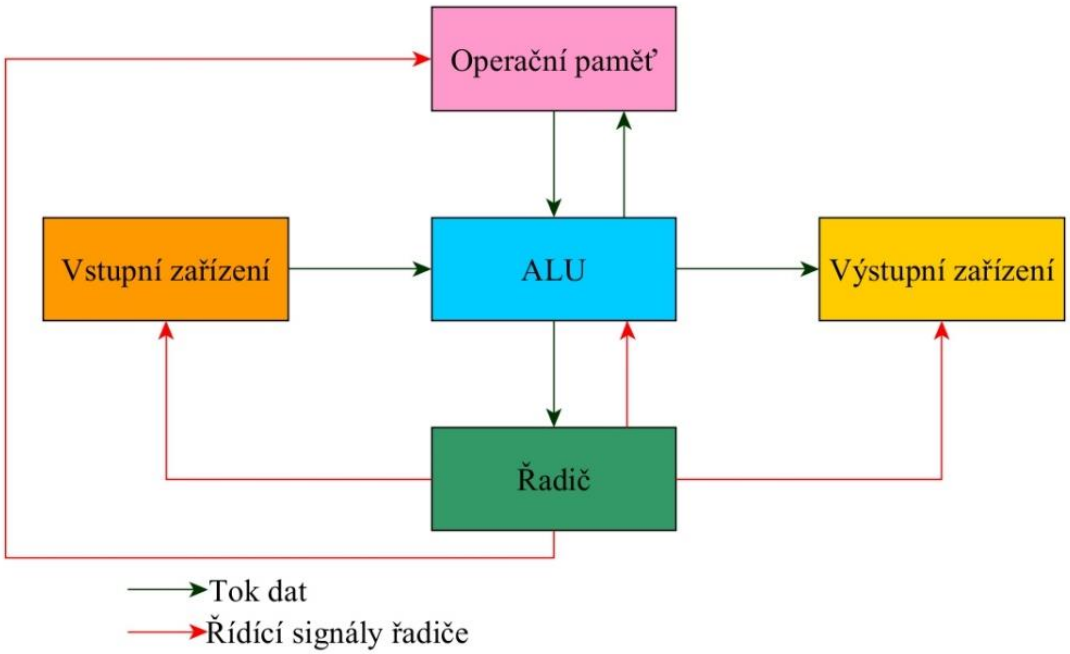
1.3 Architektura procesorů

Z historického hlediska existují dvě základní architektury mikroprocesorů, které předaly myšlenku dnešním moderním architekturám procesorů. Von Neumannova architektura a Harvardská architektura (též Harvardské schéma). [3]

1.3.1 Von Neumannova architektura

První průkopnickou myšlenku představil v roce 1945 americký vědec maďarského původu, matematik John von Neumannova. Zásadní změna v tehdejších koncepcích architektury procesorů byla, že programy a jejich data by se měla ukládat a uchovávat v jedné a téže paměti. Ve výsledku to znamená, že programy lze libovolně modifikovat a měnit, což ve výsledku vytváří neuvěřitelně flexibilní a všestranný počítač, i když s nevýhodou čistě sekvenčního postupu. To bylo později upraveno funkcí multitasking. Předchozí koncepce vycházely z představy, že program a data jsou nesmísitelná, takže co bylo jednou vytvořeno, nebylo možné změnit. Von Neumannova myšlenka ovlivnila celkový IT průmysl a lehce modifikovaný model se používá dodnes ve většině osobních počítačů. [3,4]

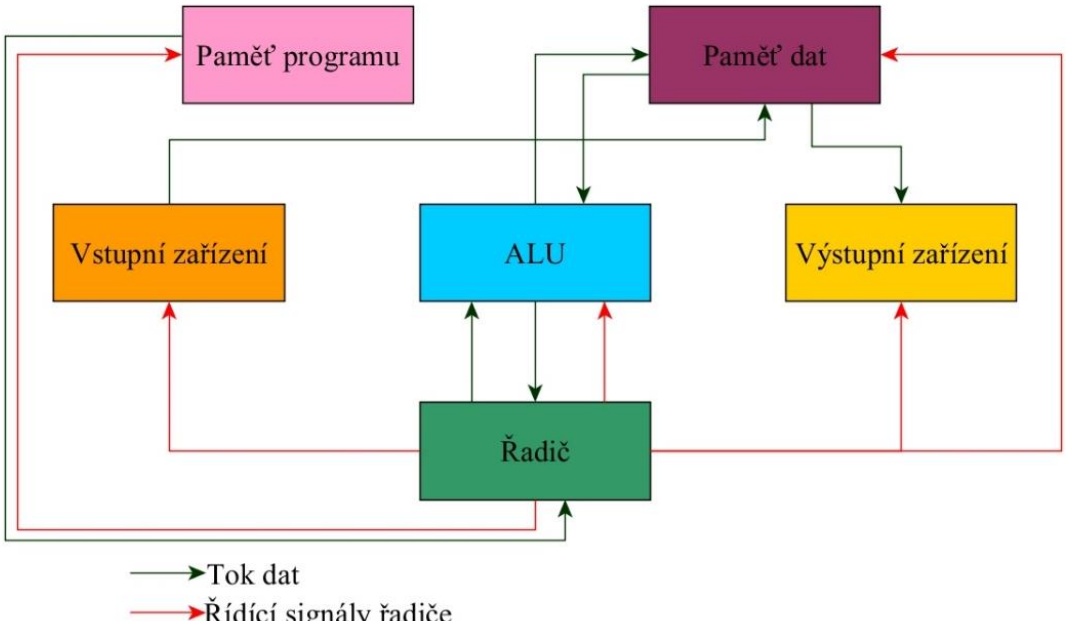
Von Neumannova architektura se skládá z pěti bloků zobrazených na Obr. 1. Operační paměť, kde se ukládá jednak program, jednak data, se kterými program pracuje. Programový řadič řídící celý počítač a vydávající povely. Aritmeticko-logická jednotka (také ALU), která vykonává aritmetické a logické operace. Poslední dva bloky jsou vstupní a výstupní zařízení, určená pro vstup programů a dat a pro výstup jednotlivých výsledků zpracovaných programem. [3,4]



Obr. 1: Schéma von Neumannovy architektury

1.3.2 Harvardská architektura

Druhým typem je Harvardská architektura, která vznikla v polovině 20. století, stejně jako von Neumannův model, avšak s jedním důležitým rozdílem. Paměť určená pro data je odlišná od paměti programu. Což má za následek celkové zrychlení počítače, protože je možné zároveň číst instrukce a pracovat s pamětí dat, ať už zapisování, nebo čtení. Velkým problémem této architektury je však princip, kdy je počítač stále ovládán jedním programem, tudíž není vhodný pro práci na obecných úlohách, které nejsou předem specifikované. Harvardská architektura (Obr. 2) byla oblíbena u počítačů na začátku sedmdesátých let a dodnes je používána právě v oblasti mikrokontrolerů. [5]



Obr. 2: Schéma Harvardské architektury

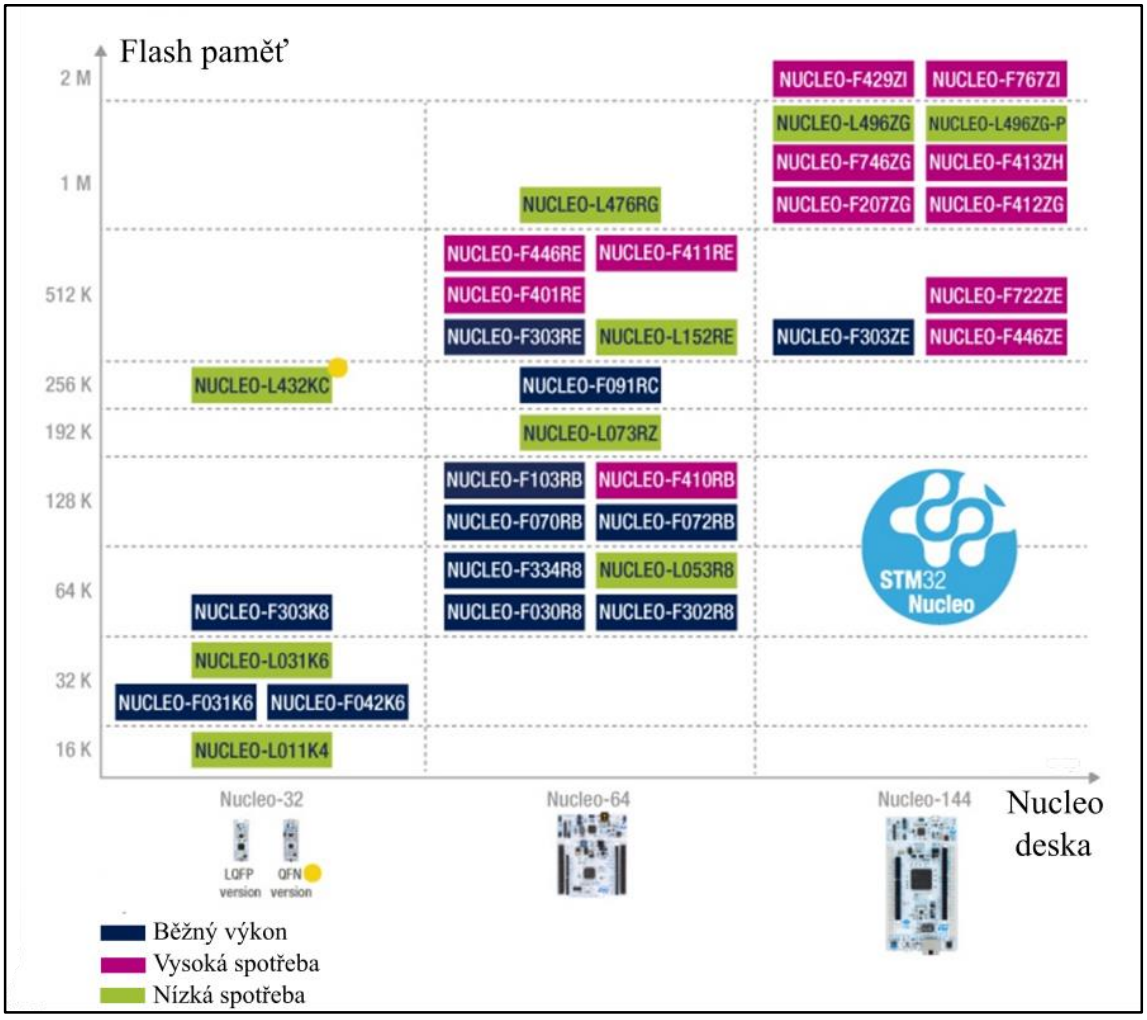
1.4 ARMové procesory

Jelikož se zabýváme prostředím ARMmbed, je třeba vysvětlit samotný pojem ARM. ARM (Advanced RISC Machines, dříve Acorn RISC Machines) je firma se sídlem v Anglii, vyrábějící ARMové procesory založené na RISC (Reduced Instruction Set Computing) architektuře. Tyto procesory mají zcela jiný typ architektury, než CISC (Complex Instruction Set Computing), který používají známí výrobci procesorů pro stolní počítače, Intel a AMD. RISC procesor je navržen pro běh menšího množství instrukcí, než jaký zvládne klasický CISC procesor. Instrukce definují, jaké příkazy mohou být programem poslány do procesoru. Jelikož zvládají méně instrukcí, mohou mít i vyšší frekvenci. Pokud se zredukuje počet instrukcí, které může procesor dostat, může být vytvořen i jednoduší obvod uvnitř čipu. Díky jednoduchému obvodu lze použít menší součástky s kratším elektrickým vedením. To dovoluje celkové zmenšení a menší spotřebu procesoru. [6]

Díky malým rozměrům a nízké spotřebě můžeme ARMové procesory najít po celém světě, v chytrých malých zařízeních, která fungují na baterie (automobily, většina „chytrých“ telefonů a také mikrokontroléry).

2 VÝVOJOVÝ KIT

Jelikož pracujeme ve vývojovém prostředí ARMmbed, musí být vybrána ARM architektura procesoru. Předložená práce se zabývá vývojovým kitem z řady STM32 Nucleo od výrobce STMicroelectronics, konkrétně NUCLEO-F401RE. Řada disponuje procesory ARM STM32 Cortex. Desky se liší ve spotřebě, v počtech pinů a v programovatelné paměti až 2 MB. Řada se skládá z mnoha typů desek (Obr. 3.).

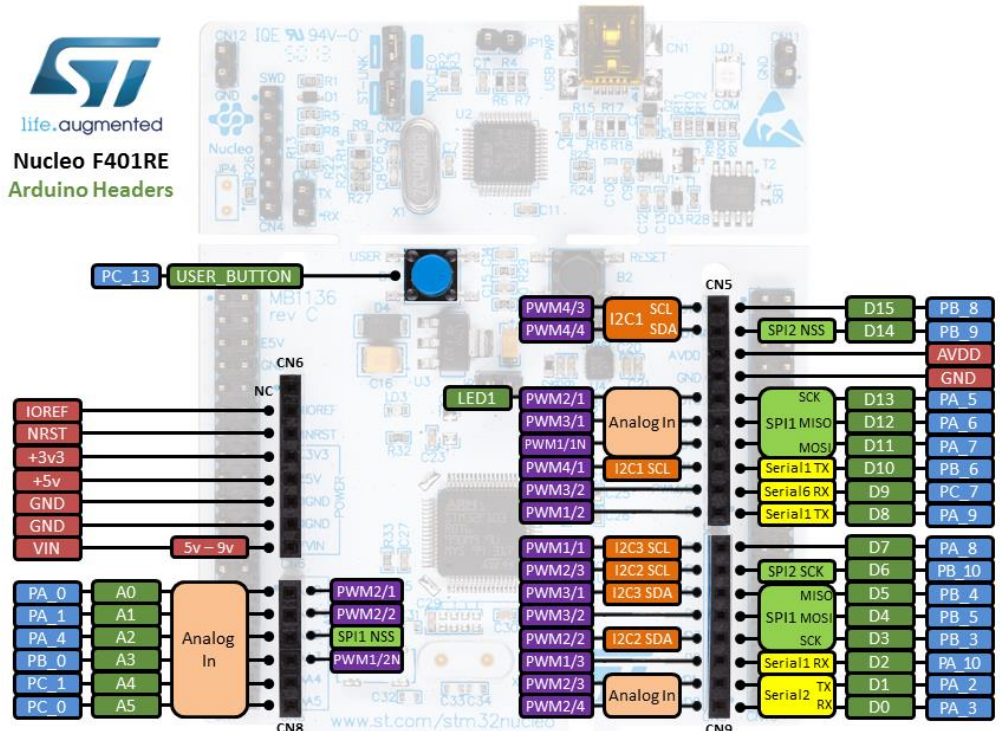


Obr. 3: Řada STM Nucleo [7]

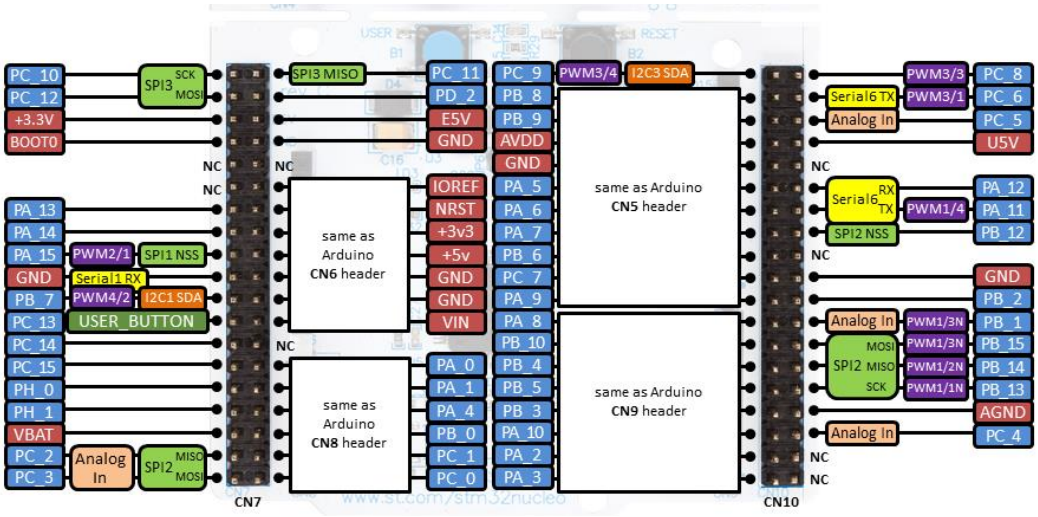
2.1 Vývojová deska STM NUCLEO-F401RE

Deska je optimalizovaná pro vysoký výkon, disponuje 512KB programovatelné flash paměti a 50 libovolně připojitelnými piny, tzv. GPIO. Na desku lze připojit celou řadu rozšiřitelných desek, které zajišťují extra funkci (například připojení přes WIFI, připojení motoru nebo reproduktoru, apod.). Díky pinům na vnitřní straně desky lze také připojit i rozšiřitelné desky Arduino Shield. Všechny piny jsou vyvedené na obvod desky.

Rozložení Arduino-kompatibilních konektorů a konektorů s rozšiřujícími piny na desce je znázorněno na Obr. 4 a 5.



Obr. 5: Arduino-kompatibilní konektory



Obr. 4: Konektory s rozšiřujícími piny [8]

Procesor na desce pracuje s 3,3 V logikou, ale zaznamenává logickou jedničku, i pokud je na vstupu až 5V, tzn., že procesor je 5 V tolerantní. Procesor na výstupu zvládne jen 3,3 V. Výhodou této voltové logiky na procesoru je její univerzálnost. Pokud je 5 V logika senzoru (např. SRF02 senzor, viz. kapitola 4.3.2), pak 3,3 V, které může procesor poslat na výstup, je pro senzor stále v rozmezí voltové tolerance logické jedničky. [9]

Samotná deska obsahuje dva čipy (Obr. 4). Čip v horní části je programátor, obsahující bootloader, druhý čip ve spodní části vývojové desky je ten, který se programuje. Programátor předává instrukce druhému čipu a obsahuje bootloader.

Bootloader je speciální program, pomocí kterého lze naflashovat daný čip (flash paměť čipu), tzn. kompletně vymazat a znovu nahrát nový program, nacházející se na daném čipu, konkrétně na programátoru. Bootloader se na čipu permanentně nachází na konci flash paměti. Druhý, programovatelný čip na desce, bootloader neobsahuje. [10]

Existují i jiné vývojové desky, například deska Arduino, která neobsahují dva čipy na desce a bootloader je permanentně nahrán na konci flash paměti. Poté, co bootloader nahraje danou aplikaci do flash pamět (stejná paměť, kde je uložen bootloader) začne procesor na první instrukci programu. V počítačovém světě je alternativou programátoru BIOS, který se po startu počítače spustí, a který nahraje do paměti RAM Windows nebo jiný operační systém. [10]

2.1.1 Parametry procesoru

V následující tabulce (Tab. 2) jsou zobrazeny parametry procesoru ARM Cortex-M4, který obsahuje vývojová deska STM NUCLEO-F401RE.

Tab. 2: Parametry procesoru [8]

Procesor	Základní informace	Řízení	Komunikace	Analogová část
ARM Cortex-M4	Maximální frekvence 83MHz	PWM	4x USART/UART	12 b ADC
	512 KB flash paměť	6x časovač	3x I²C	
	ARMv7	RTC	3x SPI	
	Harvardská architektura	2x Watchdog timer	SDIO	
	96 KB SRAM		USB 2.0	

2.1.2 Parametry desky

Vývojová deska má následující charakteristiky [8]:

- Dva typy rozšiřitelných desek
  - Arduino Uno Revision 3
  - STMicroelectronics Morpho piny
- ST-LINK/V2-1 debugování a programování
- Napájení desky přes USB VBUS nebo externí zdroj (3.3 V, 5 V, 7 - 12 V)
- Uživatelská LED
- Dvě tlačítka: USER and RESET
- Připojení přes USB přes Virtuální COM

2.2 Vybrané periferie čipu

2.2.1 Sériové sběrnice – komunikace komponent

Externí sériové sběrnice se používají pro přenos dat a komunikaci na krátkou vzdálenost mezi mikrokontrolérem a integrovanými obvody nebo pro přenos na vzdálenost v řádech metrů mezi jednotlivými mikropočítači – sběrnice CAN (Controller-area Network). Pro komunikace na krátkou vzdálenost – vedení nepřesahuje jednotky metrů – se nejčastěji používají sběrnice SPI (Serial Peripheral Interface) a I²C (Inter-Integrated Circuit), které

na rozdíl od ostatních externích portů disponují samostatným hodinovým signálem rozvedeným na každý uzel a umožňují oboustranný přenos mezi vícero uzly (zařízeními). Sériové sběrnice se používají zejména díky zmenšenému počtu adresních, datových a řídících vývodů. Nejčastěji obsahují tři až čtyři výstupy. Pro komunikaci mezi mikrokontrolérem a počítačem se používá UART (Universal Asynchronus Receiver Transmitter). [12]

2.2.1.1 SPI

Rozhraní SPI se používá k připojení externích pamětí, A/D převodníků a dalších obvodů a přídavných součástí. V předložené práci se jedná o čtyřbitový sedmisegmentový digitální display (více o displeji v kapitole 4.2). Jak bylo uvedeno výše, může být SPI sběrnice zapojena na dva i více uzlů. Pouze jeden obvod je hlavní (Master), většinou to bývá procesor, a ostatní obvody (Slaves) jsou pak spolu s hlavním připojeny čtyřmi vodiči:

1) Datový vstup MISO (Master In, Slave Out).
2) Datový výstup MOSI (Master Out, Slave In).
3) Výstup hodinového signálu SCK, který je připojen na všechny Slave SCK vstupy.
4) Vstup SS (Slave Select) připojen na každý Slave pro výběr obvodu. V případě mikrokontroléru je připojen přímo na Master (procesor) a lze tak snadno ovládat který uzel se kterým se má komunikovat.

Datový přenos probíhá mezi obvodem Master a jedním Slave obvodem. Data jsou posílána přes posuvné registry Master a Slave obvodu za stejné frekvence hodinovém signálu SCK. [12,13]

2.2.1.2 I²C

Na rozdíl od sběrnice SPI je sběrnice I²C typu multimaster. Musí řešit arbitraci a každý integrovaný obvod má svoji vlastní adresu. Jednotlivé stanice (Master/Slave) jsou propojeny jedním SDA datovým vodičem a hodinovým signálem SCL. Arbitrace pracuje metodou detekce kolize. Každá stanice během vysílání porovnává vysílané bity se stavem SDA. Pokud je zjištěna odchylka v očekávaném stavu, indikuje se kolize mezi stanicemi a stanice přestane vysílat. [13]

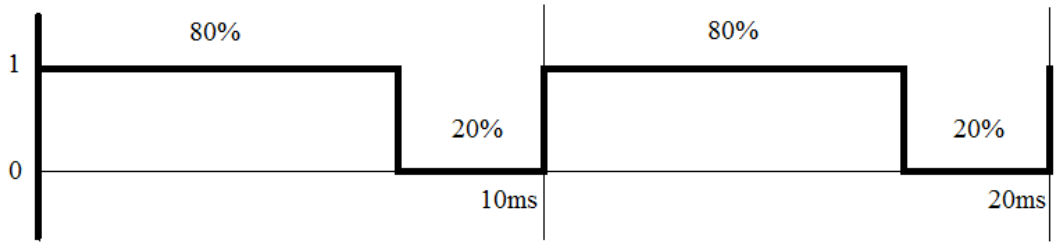
Všechny stanice mají přidělenou svoji vlastní sedmibitovou adresu. Po zachycení START podmínky každá stanice ověří vysílanou adresu sběrnice se svojí adresou a při shodě musí vyslat potvrzovací bit ACK. Poté začne přijímat/odesílat data. [13]

2.2.1.3 UART

UART je univerzální asynchronní příjmač/vysílač, který je klíčovou komponentou při sériové komunikaci mezi zařízením a počítačem. UART vezme byty dat a posílá je po jednom bitu v sériovém proudu. Na druhém konci je druhý UART, který znovu sestaví jednotlivé bity do bytů. [14]

2.2.2 PWM

PWM (Pulse Width Modulation) signál funguje na bázi střídání logické nuly a logické jedničky v určitém cyklu. Cyklus se zapisuje jako časová perioda v rozmezí sekund, milisekund, mikrosekund. Začíná vždy jedničkou. Poměr logické jedničky a nuly se nazývá střída. Střída 80 % znamená, že se střídá 80 % jedničky a 20 % nuly za daný interval (Obr. 6). Střídu lze také zapsat jako podíl periody. Při periodě 10 ms a střídě 2 ms bude v cyklu logická jednička 2 ms a poté logická nula 8 ms. Je nutné uvést, že PWM signál nemají všechny digitální vývody na desce. [15]



Obr. 6: Graf PWM signálu

PWM signál se například používá pro snižování/zvyšování jasu u LED diod. Lidské oko začne vnímat blikání jako plynulý svit při frekvenci okolo 30 ms. Takže při nastavení periody 10 ms a měněním střídy lidské oko zaznamenává změnu jasu. Příklad s použitím PWM signálu je řešen v kapitole 4.4. Obecně je PWM možnost předaní spojité informace pomoci dvoustavového výstupu.

Jako příklad z každodenní reality můžeme poukázat na starší CRT (Cathode Ray Tube) televizory, které fungovaly na frekvenci zhruba 30ms. Dnešní LCD (Liquid Crystal Display) obrazovky pracují s maximální zobrazovací frekvencí 10 ms.

2.2.2.1 Servo

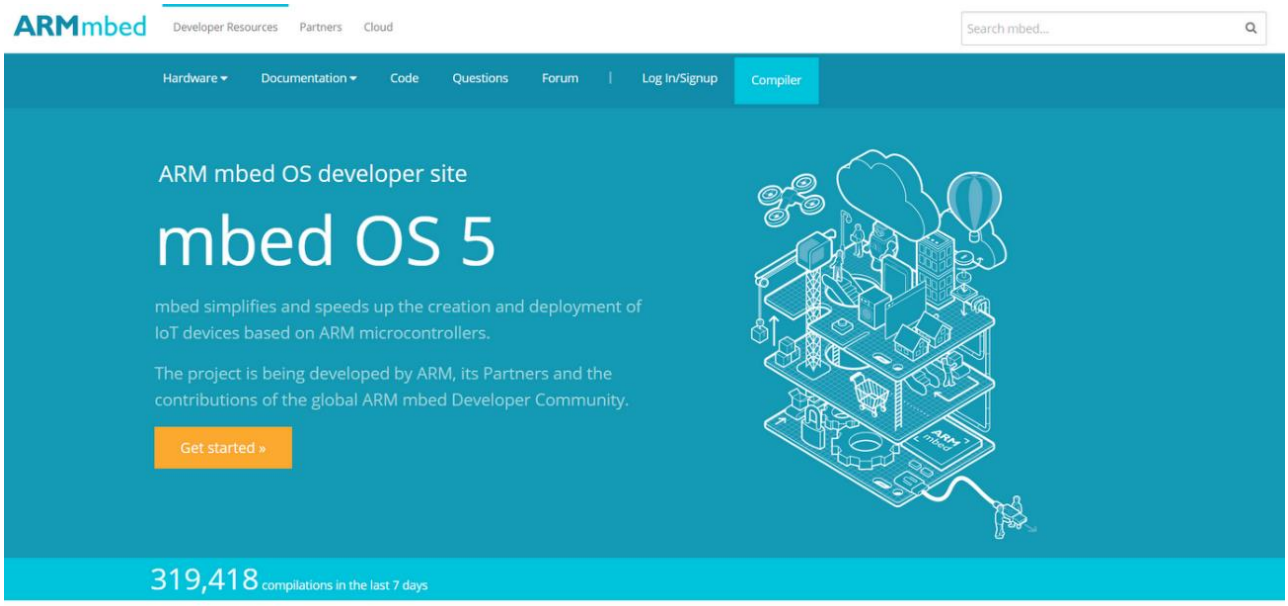
PWM signál se dá využít mnoha způsoby. Jedním z nich je ovládání servomotoru. Běžně se používá PWM signál o periodě 20 ms. Hodnota střídy pak určuje natočení servomotoru. Krajní polohy bývají 0,6 ms a 2,4 ms. Střední poloha 1,5 ms. [15]

3 PROSTŘEDÍ ARMMBED

3.1 Popis prostředí

Vývojové prostředí (dále jen IDE, Integrated Development Environment) mbed, z anglického slova embedded microcontroler (mikročip), je prostředí pro programátory vývojových desek a jejich komponentů. IDE se nachází na webové stránce [16] (Obr. 7). Prostředí je přístupné online, užívá cloudového uložiště, obsahuje technickou dokumentaci a návody k široké škále mikroprocesorů, senzorů a periferií. Dále jsou zde komunitní knihovny pro veřejnost, kde lze nahrát vytvořený program a nabídnout ho tak veřejnosti. S tím souvisí i možnost sdílení těchto programů/knihoven, které přináší zjednodušení týmové spolupráce na projektech. V této práci se všechny programy píšou právě v prostředí ARMmbed. Prostředí má tyto základní charakteristiky [11]:

- Standartní C/C++ zvýraznění syntaxe
- Python, Lua, JavaScript, XML, HTML, CSS zvýraznění syntaxe
- Zpět/vrátit operace
- Kopírování/vyjmutí/vložení textu
- Hledání v textu
- UTF-8/Unicode základní kódování.



Obr. 7: Hlavní stránka mbed OS developer [17]

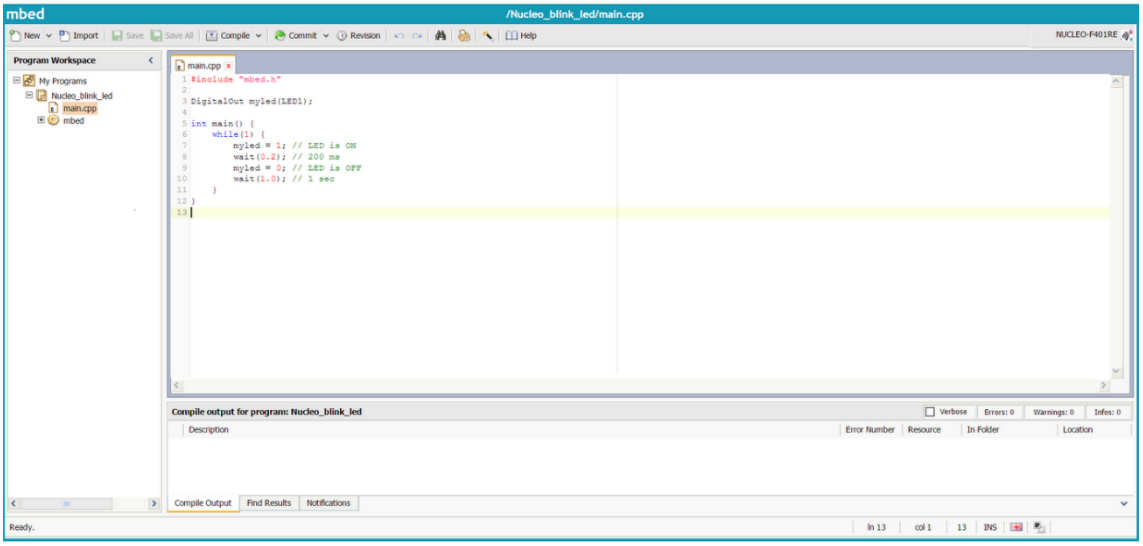
3.2 První program

Při prvním spuštění si uživatel musí stáhnout ovladač pro řadič, který je umístěn na desce. Volně ke stažení zde [18]. Při problémech je možné na stránce v záložce „Hardware → Boards“ najít příslušnou desku a postupovat podle návodu k ní přiloženému.

Při prvním programování je nutná registrace. Registraci nalezneme v pravém horním rohu. Po přihlášení klikneme na „COMPILER“. Nyní se nacházíme v hlavním

programu, kompilačním prostředí. Zde si v pravém horním rohu vybereme desku, na kterou budeme tvořit programy. V našem případě je to NUCLEO-F401RE. Nyní nám můžeme v levém horním rohu vytvořit nový projekt. IDE nám dá na výběr z předdefinovaných jednoduchých programů, pomocí kterých může uživatel snadno pochopit základní programování desek.

Jako příklad si vytvoříme program „Blinky LED test for the ST Nucleo boards“. Jedná se o jednoduchý program na rozsvěcování LED diody, nacházející se na desce. Nyní je v levém stromovém panelu vytvořen program s názvem „Nucleo_blink_led“, který obsahuje hlavní program „main.cpp“ a knihovnu „mbed.h“, která obsahuje předdefinované funkce procesoru. Na prvním řádku se nachází inicializace hlavní knihovny „mbed.h“. Dále je nadefinována LED dioda. Definice se musí shodovat s názvem LED diody, definovaným výrobcem desky (viz kapitola 2.1). Následuje hlavní část, kterou procesor vždy začíná číst jako první. Zde se pouze vytvořil nekonečný cyklus pomocí funkce „while(1)“, který na krátkou dobu zapíná diodu a pak ji vypíná. Poté co máme takto vytvořený program, provedeme kompilaci. Klikneme na „Compile“ v horní liště, čímž dojde ke kontrole chyb programu, a pokud se žádné chyby neobjeví, vygeneruje se výsledný soubor „*.bin“, se kterým už zvládne procesor na desce pracovat, a nabídne nám jeho stažení. Stažený soubor uložíme do uložiště procesoru. Uložiště je typu Mass Storage, takže se zobrazuje v počítači jako klasický flash disk. Po nahrání souboru do mikroprocesoru, je program automaticky nahrán a načten. Mikrokontrolér se restartuje a LED dioda začne blikat. IDE obsahující výše popisovaný první program zobrazeno na Obr. 8.



Obr. 8: Vývojové prostředí mbed [16]

4 VZOROVÉ PŘÍKLADY

V následující části práce jsou uvedeny čtyři vzorové příklady včetně zadání a postupu, podle kterého by měl být příklad schopen student bezproblémově vyřešit. Příklady jsou jednoduché, zaměřené na vysvětlení hlavních funkcí daných zařízení.

Úlohy jsou zaměřeny na základní a nejčastěji používané senzory a zařízení, se kterými se může student setkat. Obsahují seznámení se se sériovou komunikací, zobrazení čísel na čtyřbitovém sedmisegmentovém displeji, konkrétně stopek, měření vzdálenosti přes ultrazvukový senzor SRF02 a v poslední řadě vzorový příklad na propojení předchozích zařízení a RGB LED, připojenou přes PWM signál.

Každá úloha obsahuje zadání, teoretický úvod a postup. V teoretickém úvodu jsou uvedeny všechny potřebné teoretické informace k bezproblémovému vyřešení příkladu. Co není uvedeno zde, bylo již popsáno v předchozích kapitolách. Postup obsahuje podrobné informace řešení daného problému včetně vlastního kódu programu.

Všechny vzorové příklady, které budou dále popisovány, byly vytvořeny v prostředí ARMmbed v programovacím jazyce C++. Byla použita vývojová deska STM NUCLEO-F401RE. Funkční zdrojové kódy k jednotlivým příkladům naleznete v příloze práce.

4.1 Komunikace s mikrokontrolérem

4.1.1 Zadání

1. Seznamte se se sériovou linkou a libovolným programem na komunikaci s mikrokontrolérem.
2. Prostudujte komunitní knihovny pro danou problematiku.
3. Vytvořte v online prostředí mbed program, který bude po nahrání do paměti mikrokontroléru komunikovat přes sériovou linku s počítačem. Komunikace bude probíhat čistě jen na přeposlání zpět toho, co se pošle do mikroprocesoru.

4.1.2 Teoretický úvod

Sériová linka

Sériová linka, též UART, je universální asynchronní přijímač/vysílač. Linka má dva jednosměrné kanály, jeden pro příjem dat, druhý pro odesílání dat. Linka je asynchronní a oba konce linky musí být nakonfigurovány pro správný chod. Na vývojovém kitu NUCLEO-F401RE je zabudovaná sériová linka USB (Universal Serial Bus) pro jednoduchou komunikaci s ostatními zařízeními. [19, 20]

4.1.3 Postup

Než začneme psát náš program, je nutné mít v počítači nainstalovaný program, který dokáže číst, zobrazovat a zapisovat data do sériových COM portů. K tomu můžeme použít volně dostupný program Hercules, volně stažitelný z dostupné webové stránky [21]. Připojením desky k počítači, v našem případě přes USB, vznikne virtuální sériový port. V záložkách zvolíme „Serial“, vybereme COM, na kterém je připojena vývojová deska a počet bitů přenesených za sekundu baud, a po otevření portu začne komunikace mezi zařízeními.

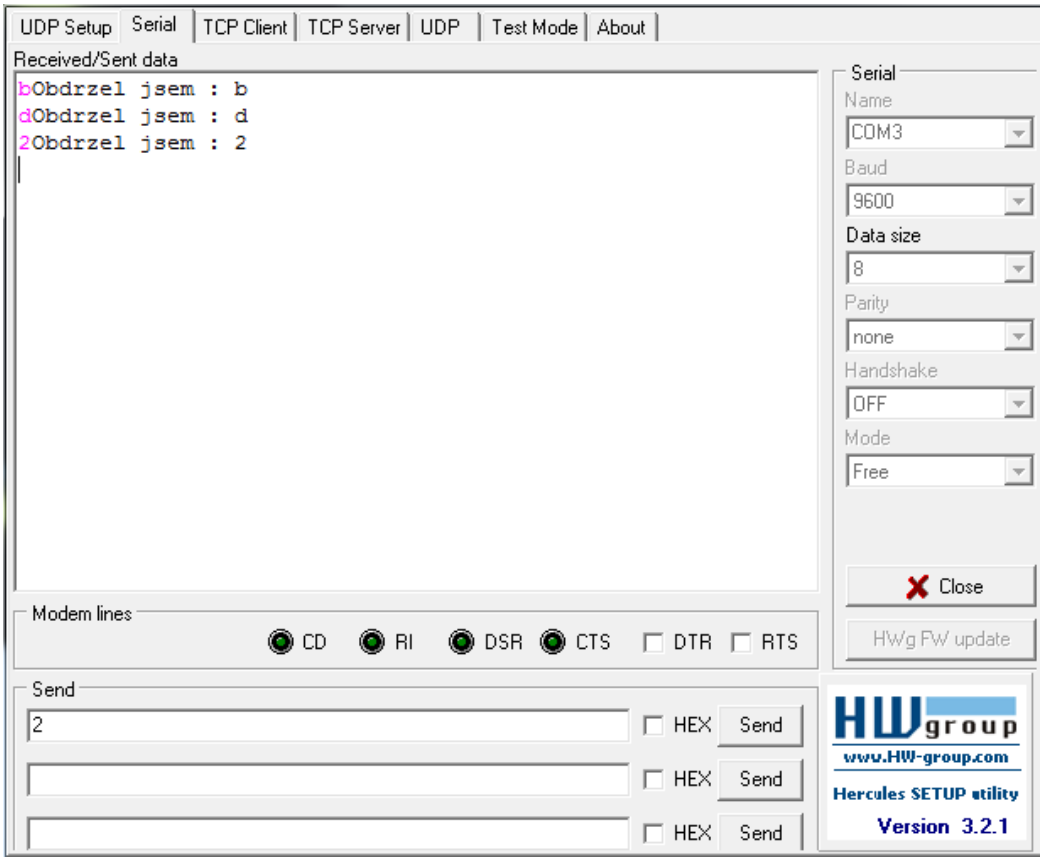
Začneme nastavením sériové linky. Pro USB port zadáme USBTX a USBRX. TX jako přijímání, RX odesílání. Nadefinujeme logickou proměnnou, která se bude měnit, pokud něco pošleme. A definujeme proměnou, do které se nám zapíše to, co přišlo. Vytvoříme funkci, která nám po přerušení od RX zapíše to, co přišlo:

```
void callback()
{
    rxZnak=uart.getc(); //nacteni znaku
    rxFlag=true; //vynulovani vlajky preruseni na rx
}
```

Nyní v hlavní části programu nastavíme rychlost sériové linky. Musí se shodovat s nastavenou rychlosti v programu Hercules. Nastavíme zavolání předchozí funkce, pokud dojde k přerušení od RX. A přepošleme zpět to co jsme zapsali:

```
int main()
{
    uart.baud(9600); //rychlost sériové linky
    uart.attach(&callback); //přerušeni na RX
    while(1) {
        if (rxFlag) { // přijato x znaku
            rxFlag=false;
                //poslat co jsem přijal
            uart.printf("Obdrzel jsem : %i\n", rxZnak);
        }
    }
}
```

Komunikace přes program Hercules lze vidět na Obr. 9.



Obr. 9: Vytvořený program v programu Hercules

4.2 Stopky

4.2.1 Zadání

1. Seznamte se se čtyřbitovým sedmisegmentovým digitálním displejem.
2. Prostudujte technickou dokumentaci a knihovnu pro ovládání tohoto displeje.
3. Zapojte digitální displej k vývojové desce NUCLEO podle návodu.
4. Ve vývojovém prostředí mbed vytvořte v jazyce C++ stopky fungující na zadaném displeji. Start stopek po zmáčknutí tlačítka na desce. Pauza po opětovném zmáčknutí. Stopky budou mít sekundy a minuty oddělené tečkou.

4.2.2 Teoretický úvod

Čtyřbitový sedmisegmentový displej

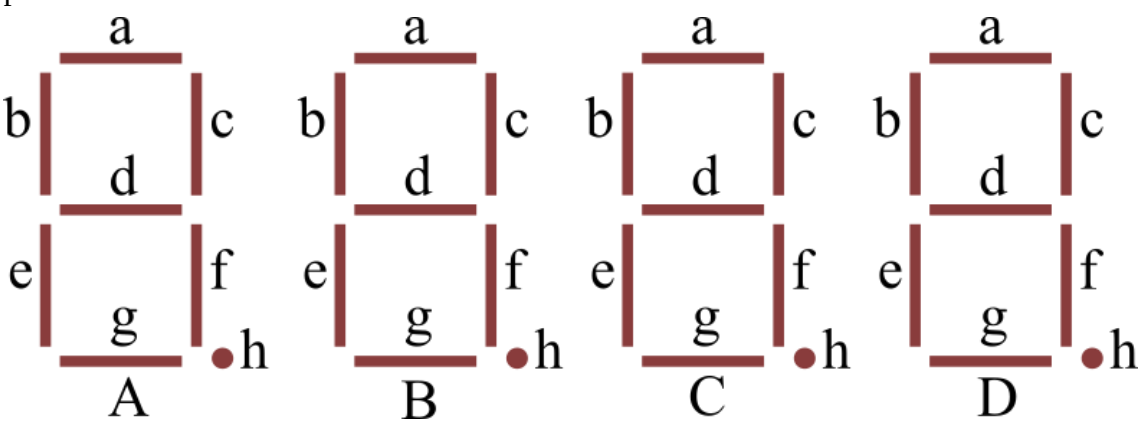
Základní vlastností segmentového displeje je, že v jeden okamžik může svítit pouze jedno pole. To je dáno ovládáním displeje přes dva posuvné registry. První registr určuje segmenty, které budou svítit. Druhý posuvný registr určuje, ve kterém ze čtyř polí budou segmenty svítit. Takže aby lidské oko nepoznalo, že se pole nerozsvěcují zároveň, musí se cyklus rozsvěcování polí vejít do intervalu zhruba 20 ms, který lidské oko ještě zaznamenává jako plynulý svit. [22]

Zadaný displej lze vidět na Obr. 10. Displej má SPI komunikační sběrnici a konkrétně 5 pinů uvedených níže:

- SCLK vstup hodinového signálu připojený k SCK výstupu na NUCLEO desce
- RCLK vstup dat připojený na libovolný digitální pin
- DIO datový výstup připojený na MOSI pin
- VCC napájení (5 V)
- GND zem

Obr. 10: Sedmisegmentový displej [23]

Přes sběrnici se do dvou posuvných registrů posílají dva byty. První byte určuje, které segmenty ze všech se mají rozsvítit. Druhý byte určuje, na kterém ze čtyř polí se má daná soustava segmentů, tvořící většinou číslo, rozsvítit. Na Obr. 11 je znázorněno schéma segmentů a polí displeje. Tabulka 3 udává hodnotu bytu pro výběr konkrétního pole.



Obr. 11: Schéma rozdělení segmentů a polí

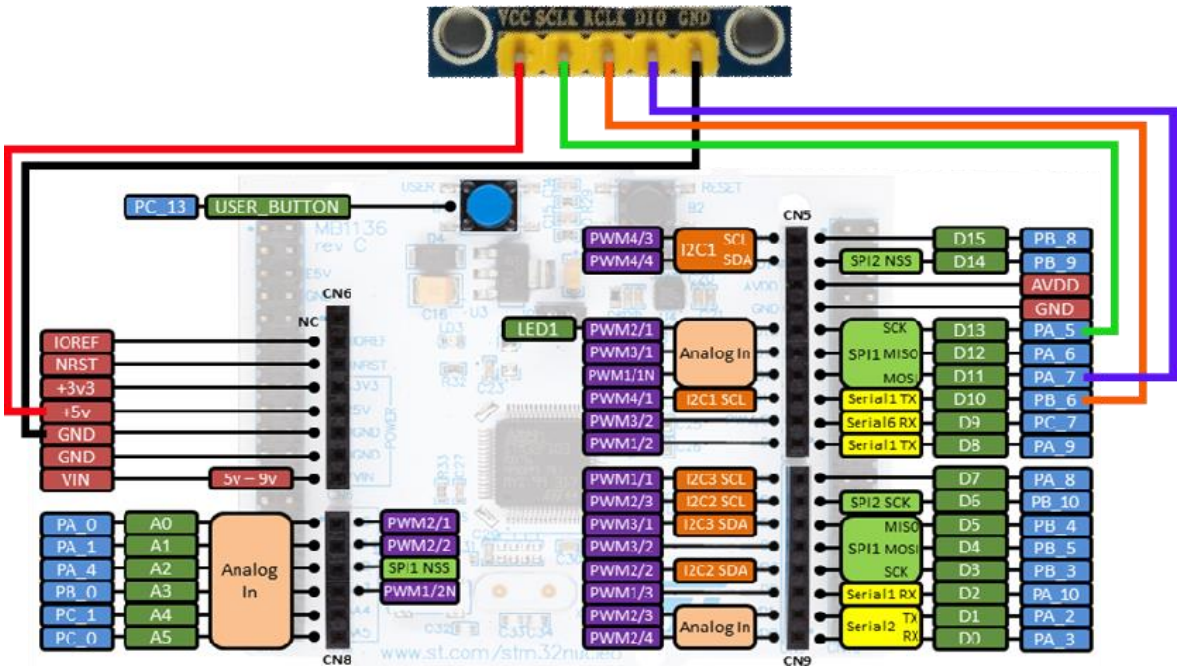
Pro zjednodušení a vysvětlení můžeme zaměnit jednotlivé bity za písmena, znázorňující jednotlivé segmenty: 0b d-b-e-g-e-c-a-h. Přiřadíme-li logickou jednotku pro zhasnutý segment a logickou nulu pro rozsvícený segment, odstraníme pomlčky, vznikne nám byte zapsaný v binární soustavě. Například byte, který rozsvítí číslo 5 je 0b00100101. [22]

Tab. 3: Tabulka druhého bytu pro jednotlivá pole

Pole	Byte
A	0b00010001
B	0b00001001
C	0b00000101
D	0b00000011

4.2.3 Postup

Nejprve je nutné správně zapojit jednotlivé piny displeje k NUCLEO desce. Provedení podle schématu na Obr. 12. RCLK lze propojit s libovolným digitálním pinem na desce.



Obr. 12: Schéma zapojení digitálního displeje

Nyní můžeme začít s psaním kódu. Stejně jako ve všech případech je nejprve zapotřebí nadefinovat připojené periferie:

```
//Definování SPI sběrnice, časovače, tlačítka
SPI spi(D11, D12, D13); // MOSI, MISO, SCLK
DigitalOut cs(D10); //RCLK
Ticker casovac;
InterruptIn event(USER_BUTTON);
```

Pro správnou funkci stopek potřebujeme vytvořit cyklus, který nám po zmáčknutí tlačítka začne každou sekundu zapisovat o jednotku vyšší hodnotu proměnných pro sekundu a minutu, a který, pokud se opět zmáčkne tlačítko, cyklus zastaví a zapíše poslední hodnoty proměnných.

Dalším krokem je vytvoření funkce, která přemění číslo od 0 po 9, včetně prázdného pole, na svítící segmenty. Výstupem je číslo, konkrétně zapsané v binární soustavě, o velikosti jednoho bytu. Poslední bit nic neznačí:

```
char dekod(char znak){
    char segmenty=0;
    switch (znak){    //prázdné pole, 0, 1, atd.
        case ' ': segmenty=0b11111111; break;
        case '0': segmenty=0b10000000; break;
        case '1': segmenty=0b11110011; break;
        . . . . . .
          //apod. až do čísla 9
     }
return segmenty;
}
```

Následuje určení pozice, na kterou má být první byte poslán. Opět vytvoříme funkci, kde na vstupu je hodnota od 0 po 4 a na výstupu bude druhý byte určující polohu. Výstupem je číslo o velikosti jednoho bytu. Nejnižší bit značí desetinnou tečku. Logická nula značí svit, logická jednička opak:

```
char digit(int pozice){
    char pozicedata;
    switch (pozice){
    case 0 : pozicedata=0b00010000; break;
    case 1 : pozicedata=0b00001001; break;
    case 2 : pozicedata=0b00000101; break;
    case 3 : pozicedata=0b00000011; break;
    }
return pozicedata;
}
```

Nyní vytvoříme funkci, která bude tyto dva byty posílat přes sběrnici do dvou registrů digitálního displeje, který podle toho bude spínat jednotlivé segmenty v jednotlivých polích.

```
void zobraz_cas(void) {
      char cislo_TXT [5];
      sprintf(cislo_TXT,"%02d%02d",min,sec);
      for(int i=0;i<4;i++)
      {
        int cekej=1;
        cs = 1;//nastavení posuvného registru pro čtení
        spi.write(dekod(cislo_TXT[i])); //odeslání 1. bytu
        spi.write(digit(i)); //odeslání 2. bytu
        cs = 0;//nastavení posuvného registru pro zápis
        wait_ms(cekej); //doba zobrazení segmentů v poli
      }
}
```

V poslední fázi je nutné spustit funkci, která bude každých 5 ms volat výše zmíněnou funkcí pro odesílání obou bytů. 5 ms bohatě stačí pro napohled plynulý svit segmentů.

```
casovac.attach(&zobraz_cas, 0.005);
```

4.3 Měření vzdálenosti

4.3.1 Zadání

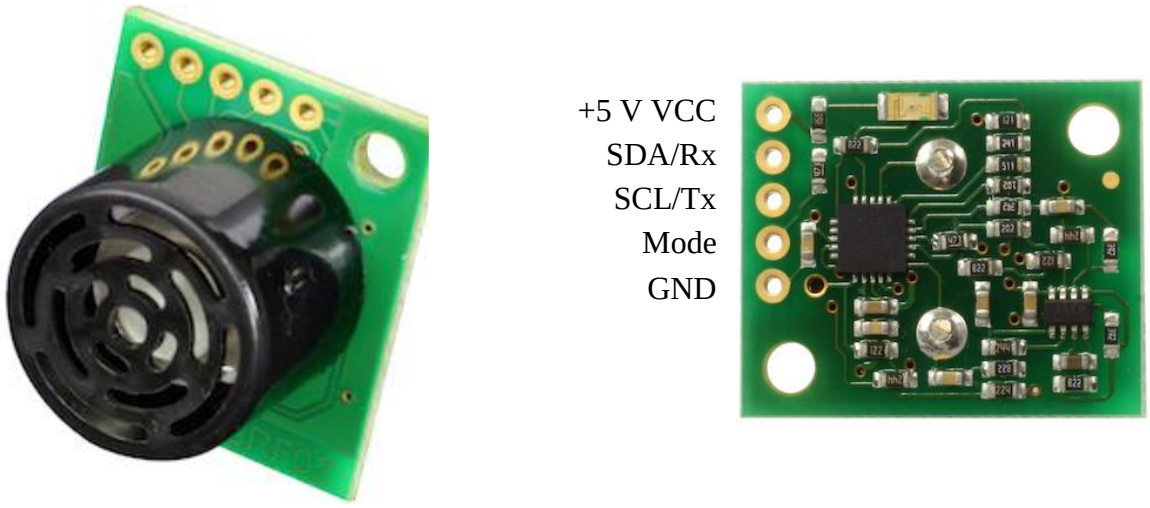
1. Seznamte se s ultrazvukovým senzorem SRF02 pro měření vzdálenosti.
2. Prostudujte technickou dokumentaci a knihovny pro ovládání tohoto senzoru.
3. Zapojte senzor SRF02 k vývojové desce NUCLEO podle návodu.
4. Ve vývojovém prostředí mbed vytvořte v jazyce C++ program, který bude měřit vzdálenost od senzoru v centimetrech a bude nám tuto hodnotu vypisovat na počítač.

4.3.2 Teoretický úvod

Ultrazvukový senzor SRF02

SRF02 je ultrazvukový dálkoměr s jedním snímačem připojený na desku s plošnými spoji. Obsahuje I²C sběrnici i sériové rozhraní. Na jednom rozhraní může být až 16 připojených senzorů. Protože SRF02 pracuje pouze s jedním snímačem, který data jednak vysílá, jednak přijímá, má značně větší minimální měřící vzdálenost než mají ostatní senzory se dvěma snímači. Minimální měřící vzdálenost se pohybuje v rozmezí 14-18 cm v závislosti na teplotě okolí. Maximální měřitelná vzdálenost se pohybuje poblíž 250 cm. Senzor může měřit v cm, palcích a mikrosekundách. [24]

Senzor SRF02 má 5 pinů. Připojení je znázorněno na Obr. 13. Pin „Mode“ nám přepíná mezi I²C a Serial módem. Chceme-li používat I²C sběrnici, jednoduše na pin „Mode“ nic nepřipojíme nebo ho připojíme na +5 V. Pro Serial funkci je zapotřebí „Mode“ pin připojit k 0 V Zemi (GND).



Obr. 13: Funkce pinů senzoru SRF02 [19]

Po připojení k napájení začne senzor vysílat svoji adresu pomocí blikání LED diody na zadní straně desky senzoru. Vždy začne s dlouhým zábleskem pro inicializaci a následují krátké záblesky v závislosti na aktuální adrese (Tab. 4). Každá z šestnácti adres pro Serial má ekvivalentní adresu pro I²C mód. Takže při změně I²C adresy se změní i Serial adresa. [24]

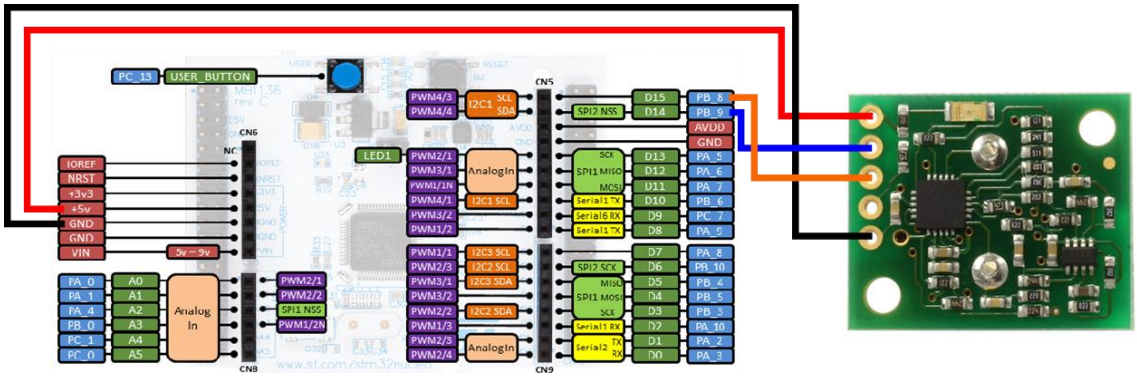
Tab. 4: Adresy senzoru SRF02 [24]

Serial adresa		I²C adresa		Dlouhé záblesky	Krátké záblesky
Decimální	Hex	Decimální	Hex		
0	00	224	E0	1	0
1	01	226	E2	1	1
2	02	228	E4	1	2
3	03	230	E6	1	3
4	04	232	E8	1	4
5	05	234	EA	1	5
6	06	236	EC	1	6
7	07	238	EE	1	7
8	08	240	F0	1	8
9	09	242	F2	1	9
10	0A	244	F4	1	10
11	0B	246	F6	1	11
12	0C	248	F8	1	12
13	0D	250	FA	1	13
14	0E	252	FC	1	14
15	0F	254	FE	1	15

Pro změnu adresy je nutné mít pouze jeden senzor zapojený na sběrnici. Dále stačí jen napsat tři příkazy ve specifickém pořadí zakončeném novou adresou senzoru. Pro příklad změny I²C adresy z defaultně zadané adresy 0xE0 na 0xF2 je třeba zapsat následující na adresu 0xE0; (0xA0, 0xAA, 0xA5, 0xF2). Vše je nutné poslat odděleně s krátkou prodlevou, např. 50 ms. [24]

4.3.3 Postup

Nejdříve zapojíme senzor k desce podle Obr. 14. Lze samozřejmě vybrat libovolný SLA/SDA pin.



Obr. 14: Schéma zapojení SRF02

Začneme definováním I²C sběrnice. Dále definujeme adresu pro zápis a adresu pro čtení ze senzoru:

```
I2C sensor(D14,D15); // SDA, SCL
 char addr=0xE0; // adresa senzoru
 char w_addr_ = addr; // nastavení zapisovací adresy senzoru
 char r_addr_ = w_addr_ + 1; //nastavení čtecí adresy
 //poslední bit v adrese rozlišuje čtení (1) a zápis (0)
```

Nyní si vytvoříme funkcí, kde deklarujeme pole o rozsahu 1x3. Do prvního pole zapíšeme příkaz pro příkazový registr senzoru a do druhého příkaz pro měření v centimetrech, palcích nebo mikrosekundách. Proměnnou odešleme přes I²C sběrnici. Počkáme minimálně 65 ms než začneme číst výsledky ze senzoru. Čtení výsledků ze senzoru probíhá následujícím způsobem – přes I²C sběrnici pošleme příkaz, který určí senzoru, co chceme číst; následně si to ze sběrnice přečteme a výsledkem této funkce je 16 bitová hodnota:

```
int getDistance(){
char data[2];
    // poslání CM příkazu do příkazového registru
    data[0] = 0x00; //CMD_REG
    data[1] = 0x51; //měření v CM
    int ack = sensor.write(w_addr_,data,2);
    wait_ms(70);
char reg = 0x02; //nejvyšší byte
    ack = sensor.write(w_addr_,&reg,1);
    ack = sensor.read(r_addr_,data,2);
return (data[0] << 8) | data[1]; //bitový posun
}
```

Nyní stačí pouze volat výše vytvořenou funkci getDistanceCm kdykoliv je to potřeba. Například každou půl sekundy pro neustálé měření. A nakonec je třeba zobrazit hodnoty na počítači (viz kapitola 4.1).

4.4 Měření vzdálenosti na displeji

4.4.1 Zadání

1. Seznamte se s PWM signálem a LED diodami.
2. Prostudujte komunitní knihovny zabývající se PWM signálem.
3. Zapojte senzor SFR02, čtyřbitový sedmisegmentový displej a RGB LED k vývojovému kitu.
4. Vytvořte program v online prostředí pro NUCLEO desku v jazyce C++, který bude měřit vzdálenost na senzoru SRF02 a tuto vzdálenost zobrazte na čtyřbitovém sedmisegmentovým displeji. Připojte k programu RGB LED diodu a vytvořte z barev libovolné signály v závislosti na naměřené vzdálenosti. Například červená barva může signalizovat vzdálenost menší než 20 cm, apod.

4.4.2 Teoretický úvod

RGB LED

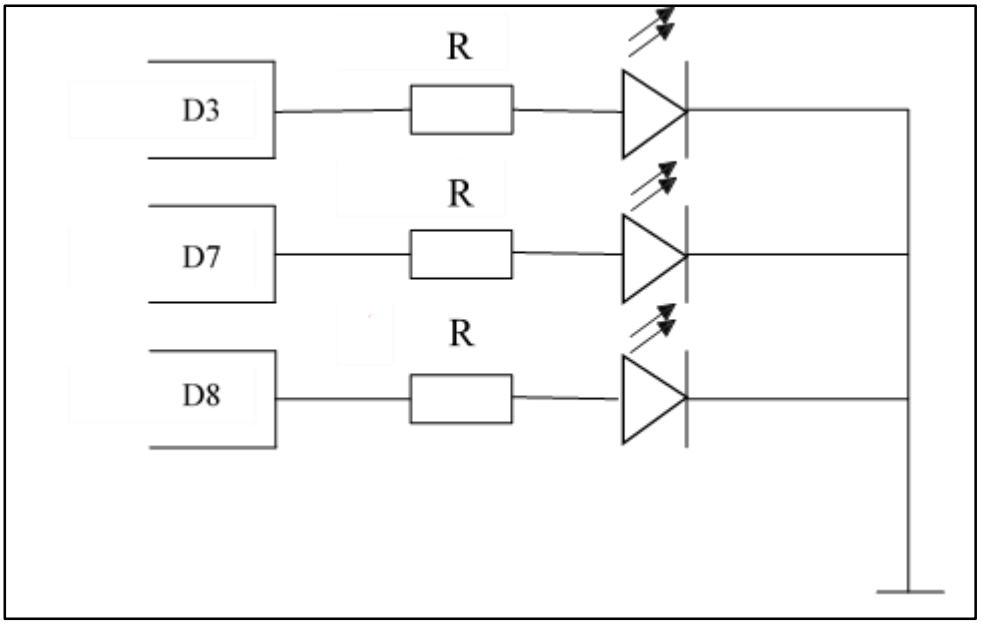
Vybraná RGB LED obsahuje 3 diody – červenou, zelenou a modrou. Kvůli většímu napětí z pinů je zapotřebí redukovat proud, který do diod vstupuje. Přesné charakteristiky diod lze vyčíst z katalogu výrobce dané diody, nebo lze použít charakteristiky z Tab. 5. U standartních LED je většinou proud roven 20 mA. Pro napětí na jednotlivých diodách lze použít vybrané hodnoty z tabulky č. 4. Použijeme napětí pro červeně svítící diodu, které je rovno 1,9 V. Napětí z pinů je 3,3 V. Nyní pomocí Ohmova zákona vypočítáme velikost předřadných odporů na červené diodě.

$$R = \frac{U_Z - U_D}{I} = \frac{3{,}3V - 1{,}9V}{0{,}02A} = 70\Omega$$

Tab. 5: Hodnoty napětí diod [25]

Dioda	Napětí [V]
Infračervená	1,6
Červená	1,9
Oranžová	2,2
Žluta	2,4
Zelená	2,6
Modrá	3,5
Bílá	3,5
Ultrafialová	3,5

Nyní lze zapojit rezistory do obvodu dle schéma na Obr. 15.



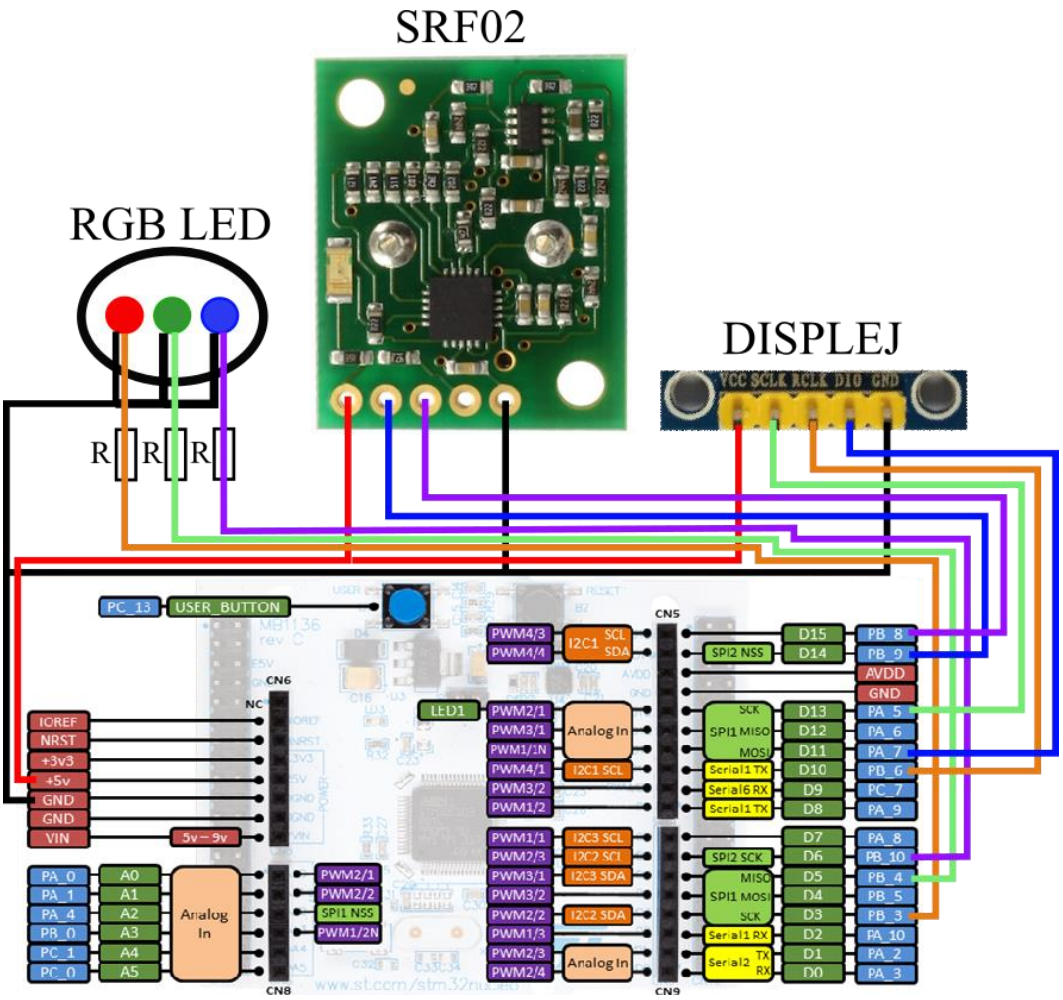
Obr. 15: Schéma zapojení diod

4.4.3 Postup

Poslední typ příkladu kombinuje postupy z předchozích dvou příkladů – Stopek a Měření vzdálenosti. Novinkou jsou právě LED diody, které jsme připojili k PWM výstupům. Byly tak připojeny pouze z hlediska větší bohatosti barev, které můžeme díky PWM signálům pestře měnit. Zapojíme senzor, RGB LED a 4bitový sedmisegmentový digitální displej k pinům na desce podle Obr. 16.

Následující postup je psán s ohledem na znalosti z předchozích příkladů. Postup je tento – zkompletujeme programy z předchozích příkladů dohromady a na digitálním displeji zobrazíme hodnotu ze senzoru. Uložíme si hodnotu ze senzoru pod nějakou proměnnou a tu následně budeme posílat do posuvných registrů displeje. Definujeme RGB diody jako PWM výstupy:

```
PwmOut R(D3);
PwmOut G(D6);
PwmOut B(D5);
```



Obr. 16: Schéma zapojení měřiče vzdálenosti na displeji

Vytvoříme funkci, kterou budeme následně spouštět každých 10ms. Ve funkci deklarujeme délku pulsu a periodu. Vhodnou volbou zadaných parametrů můžeme z RGB barev vytvořit barvu specifickou. Abychom funkci mohli spouštět každých 10ms, vytvoříme další časovač (jeden už máme vytvořen pro přenos dat do posuvného registru displeje):

```
Ticker tick; //deklarace časovače
int r=1;int g=11; int b=11;
void RGB() {
        R.period_ms(10); //perioda v ms
        G.period_ms(10);
        B.period_ms(10);
        R.pulsewidth_ms(r-1); //délka pulsu v ms
        G.pulsewidth_ms(g-1); //snížením snížíme jas
        B.pulsewidth_ms(b-1);
}
void cervena() { //tvorba vybrané barvy
  b=1; g=1;
  for (r=11;r>0;r--) {
  wait(0.1); } } //od nejjasnější červené až po nulový svit
```

Dále stačí v hlavní části programu spustit funkci nově vytvořeným časovačem a vytvořit cyklus, který bude číst hodnoty ze senzoru. Pokud budou naměřené hodnoty menší než například 20 cm, změní se barva LED. Takhle vypadá hlavní část programu:

```
int main() {
//odesílání dat do posuvného registru displeje, viz kap. 4.2
casovac.attach(&zobraz, 0.005);
tick.attach(&RGB,0.01);
     while(1) {
           //čtení hodnoty z displeje, viz kap. 4.2
           cislo = getDistanceCm();
           if(cislo<20)cervena(); //podmínky pro barvy
           if(cislo>19){g=11;r=1;b=11;};
           wait(1); //krátká prodleva před dalším měřením
    }
}
```

5 ZÁVĚR

Cílem předložené práce bylo vytvoření podkladů pro jednoduché seznámení se s mikrokontroléry ve vývojovém prostředí ARMmbed. Práce se zabývá popisem mikrokontroléru, jeho funkcí, jeho periferiemi a vývojovém prostředím ARMmbed. Dále jsou v práci vytvořeny vzorové příklady v jazyce C++ pro jednoduché pochopení programování mikrokontroléru a jeho přídavných zařízení.

Práce se dělí na čtyři části. První část práce se zabývá teoretickým úvodem v oblasti mikrokontrolerů – definuje mikrokontrolér, popisuje vývoj, architekturu a typy procesorů, používaných v mikrokontrolérech.

Ve druhé části je podrobně popsán vývojový kit, na kterém se následně programují vzorové příklady v poslední části práce. Definují se zde parametry desky a procesoru, nacházejícího se na desce. Jsou porovnány rozdíly s ostatními deskami stejného výrobce ST Microelectronics. Detailně jsou popsány všechny vývody na desce a vysvětleny funkce jednotlivých periferií, které jsou nejčastěji používány při komunikaci externích přídavných zařízení.

Třetí část se zabývá vývojovým prostředí ARMmbed, ve kterém se programují všechny programy, o kterých tato práce pojednává. Tato část je založena na seznámení se s vývojovým prostředím a následně provedením spuštění prvního programu, který je v prostředí předdefinován. Cílem je předejít problémům, které se mohou při prvním vstupu do prostředí vyskytnout, a snaha urychlit seznámení se s prostředím.

V poslední, nejdůležitější, části práce se postupně řeší jednotlivé vzorové příklady. Jsou zde uvedeny čtyři vzorové příklady včetně zadání a postupu. Úlohy jsou zaměřeny na základní a nejčastěji používané senzory a zařízení, se kterými se může student setkat. Obsahují seznámení se se sériovou komunikací, zobrazení čísel na čtyřbitovém sedmisegmentovém displeji, konkrétně stopek, měření vzdálenosti přes ultrazvukový senzor SRF02 a vzorový příklad na propojení předchozích zařízení a RGB LED, připojenou přes PWM signál. Každá úloha obsahuje zadání, teoretický úvod a postup. V teoretickém úvodu jsou uvedeny všechny potřebné teoretické informace k bezproblémovému vyřešení příkladu. Co není uvedeno zde, bylo již popsáno v předchozích kapitolách. Postup obsahuje podrobné informace řešení daného problému včetně vlastního kódu programu.

6 SEZNAM POUŽITÉ LITERATURY

[1] NOVÁK, Petr. *Mobilní roboty: pohony, senzory, řízení*. Praha: BEN - technická literatura, 2004, 247 s. ISBN 80-7300-141-1.

[2] Elektronika. *Fyzikální sekce Matematicko-fyzikální fakulty UK* [online]. Poslední změna 8. 5. 2014. [cit. 18. 05. 2017].
Dostupné z: http://physics.mff.cuni.cz/kfpp/skripta/elektronika/

[3] Jak pracuje počítač? *Root.cz - informace nejen ze světa Linuxu* [online].
Copyright © 1998 [cit. 24. 05. 2017].
Dostupné z: https://www.root.cz/clanky/jak-pracuje-pocitac/

[4] Von Neumannova architektura. *eArchiv.cz* [online]. [cit. 18. 05. 2017].
Dostupné z: http://www.earchiv.cz/a93/a321c120.php3

[5] Architektura počítače. *Otázky na státnice* [online]. [cit. 18. 05. 2017].
Dostupné z: http://michaelkuty.github.io/ssz-ai-hk-3/tech/2.html

[6] What is ARM? | *Android Central | Android Forums, News, Reviews, Help and Android Wallpapers* [online]. [cit. 18. 05. 2017].
Dostupné z: http://www.androidcentral.com/what-arm-cpu

[7] STM32 MCU Nucleo. *STMicroelectronics* [online]. Copyright © 2017 STMicroelectronics [cit. 15. 05. 2017].
Dostupné z: http://www.st.com/en/evaluation-tools/stm32-mcu-nucleo.html?querycriteria=productId=LN1847

[8] NUCLEO-F401RE. *Development Platform for Devices | mbed* [online]. Copyright © mbed [cit. 10. 05. 2017].
Dostupné z: https://developer.mbed.org/platforms/ST-Nucleo-F401RE/

[9] 5 V tolerant pins on Nucleo boards - Question. *Development Platform for Devices | mbed* [online]. Copyright © mbed [cit. 24. 05. 2017].
Dostupné z: https://developer.mbed.org/questions/4738/5V-tolerant-pins-on-Nucleo-boards/

[10] Bootloader v mikrokontrolérech AVR. *µArt.cz | Elektronika do posledního µA* [online]. [cit. 24. 05. 2017]. Dostupné z: http://uart.cz/721/bootloader-v-avr/

[11] MAŠTÁLKA, Daniel. *Výukový KIT pro platformu MBED* [online]. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií, 2016 [cit. 10. 05. 2017]. Dostupné z: http://hdl.handle.net/11012/59733

[12] DUDÁČEK, K. *Sériová rozhraní* [online]. Copyright © [cit. 23. 04. 2017].
Dostupné z: http://home.zcu.cz/~dudacek/NMS/Seriova_rozhrani.pdf

[13] Externí sériové sběrnice SPI a I²C. *Root.cz* [online]. Copyright © 1998 [cit. 23. 04. 2017].
Dostupné z: https://www.root.cz/clanky/externi-seriove-sbernice-spi-a-i2c/

[14] Serial and UART Tutorial. *The FreeBSD Project* [online]. [cit. 23. 05. 2017].
Dostupné z: https://www.freebsd.org/doc/en/articles/serial-uart/

[15] PWM. *Úvod* [online]. [cit. 23. 04. 2017]. Dostupné z: http://hw.zuth.cz/09hodina.html

[16] *Development Platform for Devices | mbed* [online]. Copyright © mbed [cit. 10. 05. 2017].
Dostupné z: https://developer.mbed.org

[17] *mbed* [online]. Copyright © ARM Ltd. Copyright 2017 [cit. 10. 05. 2017].
Dostupné z: https://www.mbed.com/en/

[18] NUCLEO-F401RE. *STMicroelectronics* [online]. Copyright © 2017 STMicroelectronics [cit. 24. 05. 2017].
Dostupné z: http://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-nucleo/nucleo-f401re.html

[19] Serial. *Development Platform for Devices | mbed* [online]. Copyright © mbed [cit. 10. 05. 2017]. Dostupné z: https://developer.mbed.org/handbook/Serial

[20] Serial and UART Tutorial. *The FreeBSD Project* [online]. [cit. 23. 05. 2017].
Dostupné z: https://www.freebsd.org/doc/en/articles/serial-uart/

[21] Aplikace Hercules SETUP. *Hw-group.com* [online]. [cit. 23. 04. 2017].
Dostupné z: http://www.hw-group.com/products/hercules/index_cz.html

[22] Posuvný registr 74HC595. *arduino8.cz* [online]. Copyright © 2015 Všechna práva vyhrazena. [cit. 10. 05. 2017].
Dostupné z: http://arduino8.webnode.cz/news/lekce-12-posuvny-registr-74hc595/

[23] 4 Digit display from Qifei - General - 43oh. *43oh - TI Microcontroller Forums, Projects and News* [online]. Copyright © 43oh, 2017 [cit. 24.05.2017].
Dostupné z: http://forum.43oh.com/topic/8596-4-digit-display-from-qifei/

[24] SRF02 Ultra sonic range finder. *Robot Electronics* [online]. [cit. 10.05.2017].
Dostupné z: http://www.robot-electronics.co.uk/htm/srf02tech.htm

[25] LED. *Pendatron.cz* [online]. [cit. 15.05.2017]. Dostupné z: http://pandatron.cz/?605&led

7 SEZNAM OBRÁZKŮ

8 SEZNAM TABULEK

9 SEZNAM ZKRATEK

10 PŘÍLOHY

Příloha 1: CD - elektronická verze bakalářská práce
- zdrojové kódy ke vzorovým příklad