VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

WINDOWS 10 IOT NA PLATFORMĚ RASPBERRY PI 2

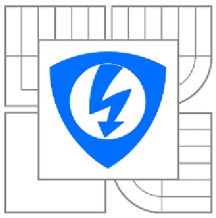
WINDOWS 10 IOT ON RASPBERRY PI 2 COMPUTER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Vojtěch Prachař
AUTHOR

VEDOUCÍ PRÁCE — Ing. Aleš Jelínek
SUPERVISOR

BRNO 2016



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Vojtěch Prachař	***ID:***	164374
Ročník:	3	***Akademický rok:***	2015/2016

NÁZEV TÉMATU:

Windows 10 IoT na platformě Raspberry Pi 2

POKYNY PRO VYPRACOVÁNÍ:

1. Vypracujte rešerši možností Raspberry Pi2 a Windows 10 IoT.
2. Prakticky otestujte všechny dostupné rozhraní Raspberry Pi 2.
3. Proveďte praktické testování Windows 10 IoT z hlediska požadavků automatizace a robotiky.
4. Vytvořte wiki stránky obsahující tutoriály k předešlým bodů.
5. Vypracujte dokumentaci a zhodnoťte získané výsledky.

DOPORUČENÁ LITERATURA:

MICROSOFT. Develop Windows 10 IoT apps on Raspberry Pi 2 and Arduino [online]. 2015 [cit. 2015-09-18]. Dostupné z: https://dev.windows.com/en-us/iot

Termín zadání:	8. 2. 2016	***Termín odevzdání:***	23. 5. 2016

Vedoucí práce: Ing. Aleš Jelínek
Konzultanti semestrální práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Účelem této semestrální práce je seznámení s Raspberry Pi 2, operačním systémem Windows 10 IoT a vývojářskými nástroji pro tuto platformu. V rámci práce byl představen hardware i software a detailně popsáno zprovoznění vývojářských nástrojů. Dále byly vyzkoušeny některé ukázkové aplikace a komunikační protokoly. V posledním bodě byly otestovány rozhraní důležitá z pohledu robotiky. V závěru je tato platforma zhodnocena pro možnost použití v domácí automatizaci a robotice.

Klíčová slova

Raspberry Pi 2, SPI, UART, I2C, automatizace, robotika, Windows 10, IoT, Visual Studio

Abstract

The purpose of this thesis is to learn about the Raspberry Pi 2, Windows 10 IoT and development tools for this platform. In this work is presented hardware and software, and stated in detail the launching of development tools. In next part some sample applications and communication protocols are tested. In last part tested important ports for robotics. At the end, this platform is evaluated for the possibility of usage in home automation and robotics.

Keywords

Raspberry Pi 2, SPI, UART, I2C, automation, robotics, Windows 10, IoT, Visual Studio

Bibliografická citace:

PRACHAŘ, V. *Windows 10 IoT na platformě Raspberry Pi 2*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. 42s. Vedoucí bakalářské práce byl Ing. Aleš Jelínek

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Windows 10 IoT na platformě Raspberry Pi 2 jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **23. května 2016** .....

podpis autora

Poděkování

Rád bych poděkoval panu Ing. Aleši Jelínkovi za jeho rady a vedení bakalářské práce. Také bych zde poděkoval Ing. Miroslavu Uhrovi za jeho pomoc při měření.

V Brně dne: **23. května 2016**

.....
podpis autora

OBSAH

1. ÚVOD

V dnešní době, kdy máme spoustu „inteligentních“ spotřebičů v domácnosti, se začíná rozšiřovat trend připojovat zařízení k internetu. Tyto spotřebiče se poté nazývají zkratkou IoT (internet of things). V této skupině nalezneme velké spotřebiče, jako je televize, pračka, lednice, ale i malé spotřebiče, jako je například termostat, nebo i obyčejná žárovka [1]. Při vývoji nových spotřebičů a software je však nutné mít nějakou univerzální platformu, na které budeme moci zkoušet různé konfigurace hardware a testovat námi vyvíjený software. Pro tento účel nalezneme spoustu různých vývojových desek, které podporují většinou OS Linux. Nalezneme zde taky Raspberry Pi 2, na kterém můžeme provozovat OS Linux a od léta 2015 Windows 10 IoT, pro který je uzpůsobeno i Visual studio 2015, které Microsoft dal vývojářům zdarma [2], [3].

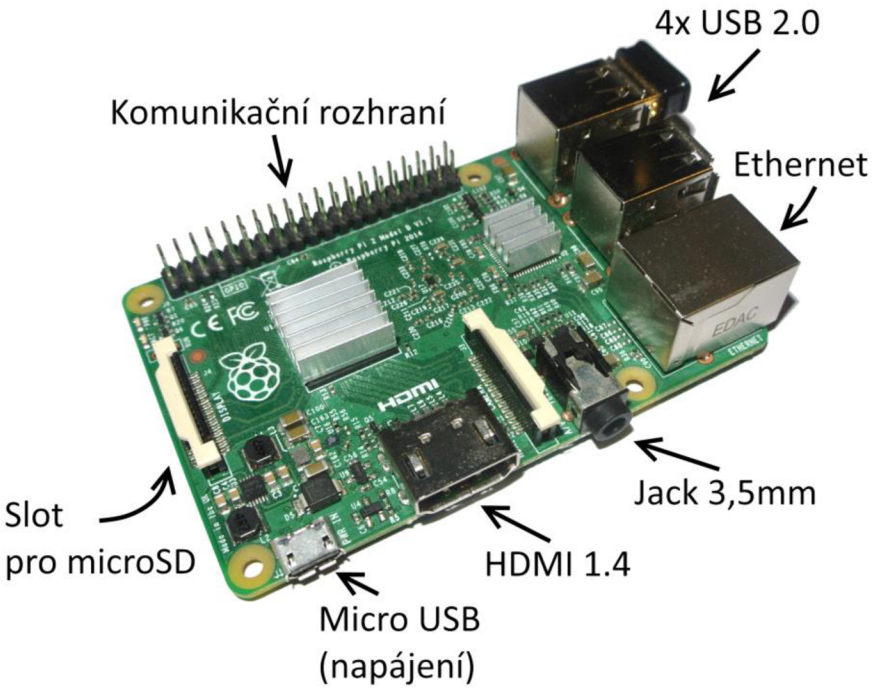
Má bakalářská práce se zabývá právě Raspberry Pi 2 ve spojení s Windows 10. Je zde popsán tento minipočítač, hlavně jeho rozhraní. U některých rozhraní byla taky otestována reálná přenosová rychlost. Je zde vysvětlen celý postup instalace operačního systému. Dále jsou popsány potřebné nástroje pro vývoj. Jejich zprovoznění a propojení s Raspberry Pi 2 pro nahrání programu a debuggování. Dále jsou v práci ukázány některé ukázkové aplikace s jejich rozborem. V poslední části je rozebrána vhodnost této kombinace pro domácí automatizaci a robotiku spolu s testy potřebných parametrů k tomuto použití.

2. RASPBERRY PI 2

Raspberry Pi 2 je již 2. generace[1] tohoto minipočítače o velikosti kreditní karty, který je osazen starším ARM procesorem Broadcom BCM2836 s grafickým jádrem VideoCore IV. Jedná se o čtyř jádrový procesor taktovaný na 900 MHz, který lze však úpravou konfiguračního souboru přetaktovat i na 1,1 GHz, kdy při této frekvenci je nutné zvýšit napájecí napětí procesoru. K procesoru je připojena 1 GB paměť RAM, typu DDR2. Hlavním důvodem proč deska není osazena novějšími, případně výkonnějšími komponenty je jeho cena, kterou se výrobce snažil udržet co nejníže. To se mu také povedlo, jelikož cena je pod 1000 Kč.

Deska je vybavena jedním micro USB konektorem pro napájení, kde je doporučen 1,5 A zdroj, reálně však záleží na připojených komponentech a lze použít i slabší. Další standartní rozhraní, které se zde nachází, jsou 4x USB 2.0 a RJ-45 které jsou připojeny přes integrovaný USB 2.0 Hub and 10/100 Ethernet Controller „LAN9514“ a dále HDMI 1.4 a 3.5 mm jack. U okraje desky se nalézají piny komunikačního rozhraní. Na spodní straně desky je poté umístěn slot na microSD kartu, která je určena na OS a data. Deska je také vybavena čtyřmi otvory pro snadnou montáž [4], [5].

Zajímavostí tohoto minipočítače je, že je vyroben ve Spojeném Království.

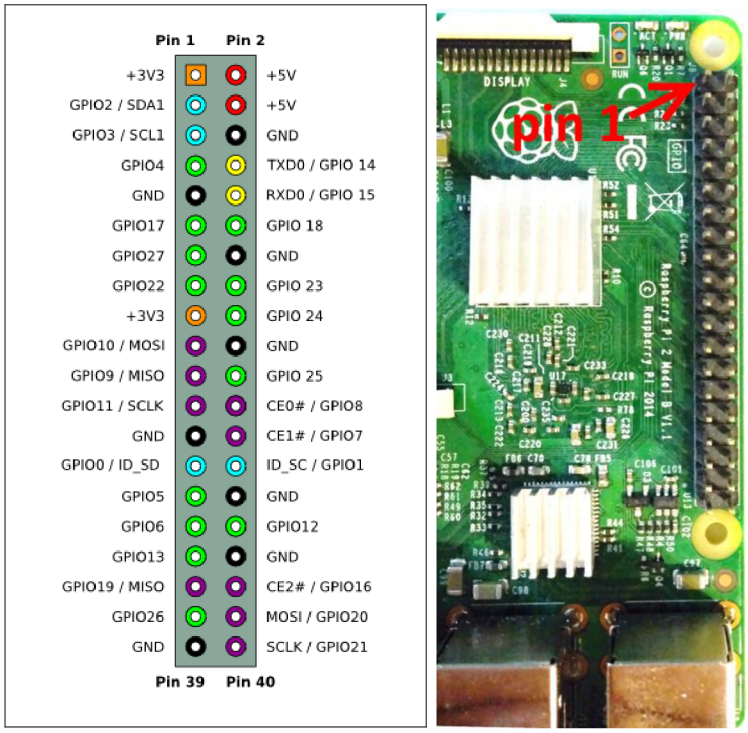


Obrázek 2.1: Raspberry Pi 2.

[1] V únoru 2016 byla představena 3. generace, která je osazena výkonnějším procesorem, který je na rozdíl od 2. generace 64 bitový. Dále má navíc wi-fi a bluetooth modul. Tato nová generace je ve všem zpětně kompatibilní s 2. generací, takže i s Windows 10 IoT.[6]

2.1. Komunikační rozhraní 1

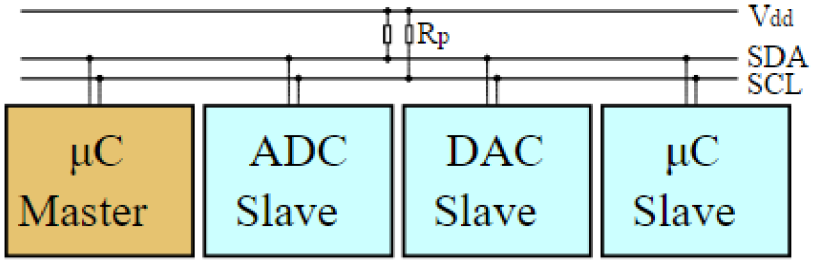
Hlavním portem pro vývoj, který se nalézá na desce je port GPIO. Tento port má 40 pinů, které jsou rozděleny pro různé funkce. 27 pinů je vstupně/výstupních se vstupním/výstupním proudem maximálně 2 – 16 mA, dle nastavení. Většina těchto pinů ale má i další funkce v podobě podpory různých komunikačních protokolů. Nebezpečím tohoto portu však je, že nepracuje na očekávané 5 V logice, ale využívá 3,3 V logiku. Dále se zde nachází dvojice 3,3 V a 5 V pinů pro napájení periferií a také 8 zemnících pinů [7].



Obrázek 2.2: Konektor GPIO [7].

2.1.1. I2C

Pro komunikaci přes protokol I2C jsou přizpůsobeny 2 piny. Pin č. 3, na kterém se nachází SDA (Data) a na pinu č. 5 SCL (Clock) a samozřejmě napájení na pinu č. 1. Podporovány jsou podle Microsoftu dvě rychlosti: 100 kbit/s (StandardMode) a 400 kbit/s (FastMode).



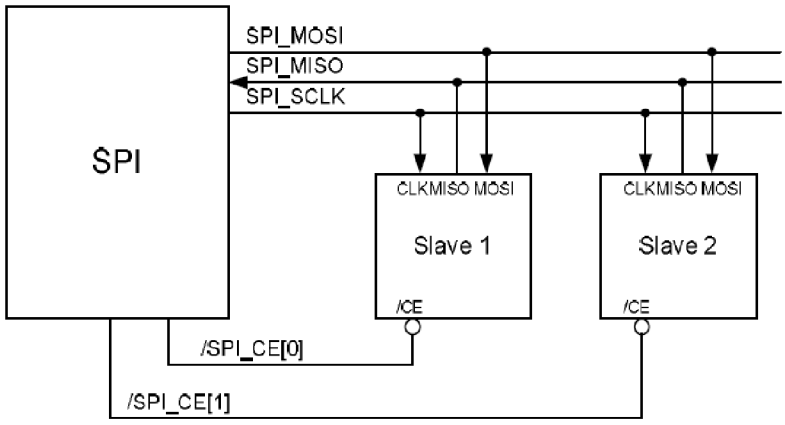
Obrázek 2.3: I2C – připojení periferií [8].

2.1.1. SPI

Ke komunikaci přes protokol SPI se zde nachází dvojice portů. První z nich na pinu č. 19 MOSI, č. 21 MISO, č. 23 SCLK a na pinech č. 24 a 16 CS0,1. Druhý z nich je poté na pinech č. 38 MOSI, č. 35 MISO a č. 40 SCLK. Maximální rychlost (frekvence) je 125 MHz a minimální 30,5 kHz [9].

cdiv	speed
2	125.0 MHz
4	62.5 MHz
8	31.2 MHz
16	15.6 MHz
32	7.8 MHz
64	3.9 MHz
128	1953 kHz
256	976 kHz
512	488 kHz
1024	244 kHz
2048	122 kHz
4096	61 kHz
8192	30.5 kHz

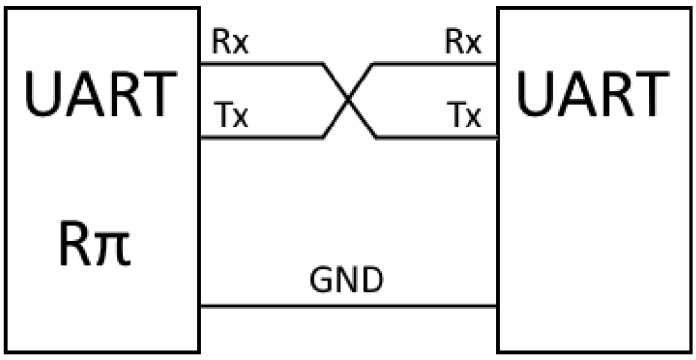
Tabulka 2.1: Oficiálně podporované rychlosti [9].



Obrázek 2.4: SPI MASTER - připojení periferií [10].

2.1.2. UART

Komunikace pomocí protokolu UART (sériový port) je možná přes pin č. 8 TXD (Output) a pin č. 10 RXD (Input). Maximální přenosová rychlost je 31,25 kBd a minimální rychlost je 476 Bd [11].



Obrázek 2.5: Zapojení UART.

2.2. Komunikační rozhraní 2

Měření bylo prováděno volně dostupným programem *Diskspd Utility v2.0.15* [12] od firmy Microsoft. Byla měřena rychlost čtení a zápisu 100 MB souborů po dobu 5ti minut pro každé měření. Toto nastavení zajistilo rozlišení 0,33 MB/s, což je pro informační měření dostatečné. Při měření rychlosti USB flash disku a SD karty byl tento program spuštěn přímo na Raspberry Pi 2. Při měření rychlosti ethernetu byl program spuštěn na počítači Asus UX305UA s 1Gb ethernetovou kartou a diskem s rychlostí čtení i zápisu více než 400MB/s, což zajišťuje dostatek výkonu, aby nebyl výsledek zkreslen.
Parametry spuštěného programu pro měření rychlosti čtení:
diskspd.exe -b100M -d300 -o1 -t1 -h -w0 -Z500M -c500M C:\test.dat
Parametry spuštěného programu pro měření rychlosti zápisu:
diskspd.exe -b100M -d300 -o1 -t1 -h -w100 -Z500M -c500M C:\test.dat

-b Velikost bloku dat.
-d Délka testování v sekundách.
-o Hloubka fronty.
-t Počet vláken pro měření
-h Zakázání ukládání do mezipaměti.
-w Procentní vyjádření zápisu. (příklad: -w25 = 25% zápis, 75% čtení)
-Z Velikost vyrovnávací paměti pro uložení náhodných dat k testu.
-c Velikost testovacího souboru.

2.2.1. SD karta

Typ SD karty (formát souborů)	Čtení [MB/s]	Zápis [MB/s]
Sony CD-HC class 10 (NTFS)	11,00	9,33
Kingston SD-HC class 10 (NTFS)	10,67	8,00

Tabulka 2.2: Test přenosové rychlosti SD karty.

Typ SD karty (formát souborů)	Čtení [MB/s]	Zápis [MB/s]
Sony CD-HC class 10 (NTFS)	19,33	12,67
Kingston SD-HC class 10 (NTFS)	35,33	9,67

Tabulka 2.3: Test přenosové rychlosti SD karty v PC – referneční meření.

Z výsledků měření SD karet je vidět, že rychlost čtení i zápisu se pohybuje kolem rychlosti 10 MB/s. Při testu rychlosti čtení byla rezerva v rychlosti karty dostatečná, v testu rychlosti zápisu je ale možné, že měla na výsledek vliv i rychlost SD karty. Bohužel ale nebyla pro testování dostupná rychlejší karta.

2.2.2. USB

Čtyři USB 2.0 rozhraní a Ethernet jsou k procesoru připojeny pomocí USB 2.0 HUBu, což by mohlo snižovat reálnou přenosovou rychlost. Proto byla, kromě samotné rychlosti připojeného zařízení, také otestována v další kapitole rychlost mezi ethernetem a tímto zařízením v USB.

Typ flash disku (formát souborů)	Čtení [MB/s]	Zápis [MB/s]
Kingston HyperX FURY 16GB (FAT32)	20,33	29,33
Kingston DataTraveler SE9 G2 32GB (FAT32)	20,33	13,33

Tabulka 2.4: Test přenosové rychlosti flash disku.

Typ flash disku (formát souborů)	Čtení [MB/s]	Zápis [MB/s]
Kingston HyperX FURY 16GB (FAT32)	36,00	33,00
Kingston DataTraveler SE9 G2 32GB (FAT32)	35,00	13,33

Tabulka 2.5: Test přenosové rychlosti flash disku v PC – referenční měření.

V případě flash disků byla všechna zařízení dostatečně rychlá pro nezkreslené měření, kromě rychlosti zápisu na druhý z flash disků. Rychlost byla změřena několikrát, u všech měření, ale nedošlo ke změně. Bylo také vyzkoušeno chování při zatížení procesoru a ani v tomto případě se rychlost nezměnila. Rychlost čtení je 20,33 MB/s, zvláštností ale je rychlost zápisu, která je v tomto případě vyšší: 29,33 MB/s.

2.2.1. Ethernet

Test rychlosti ethernetu byl proveden vůči USB flash disku i SD kartě, aby se ověřila omezení USB HUBu, na který je ethernet připojen. Toto měření bylo prováděno z PC přes protokol SMB2.

Typ flash disku (formát souborů)	Čtení [MB/s]	Zápis [MB/s]
Kingston HyperX FURY 16GB (FAT32)	10,67	11,00
Kingston DataTraveler SE9 G2 32GB (FAT32)	10,67	10,67

Tabulka 2.6: Test přenosové rychlosti flash disku přes ethernet.

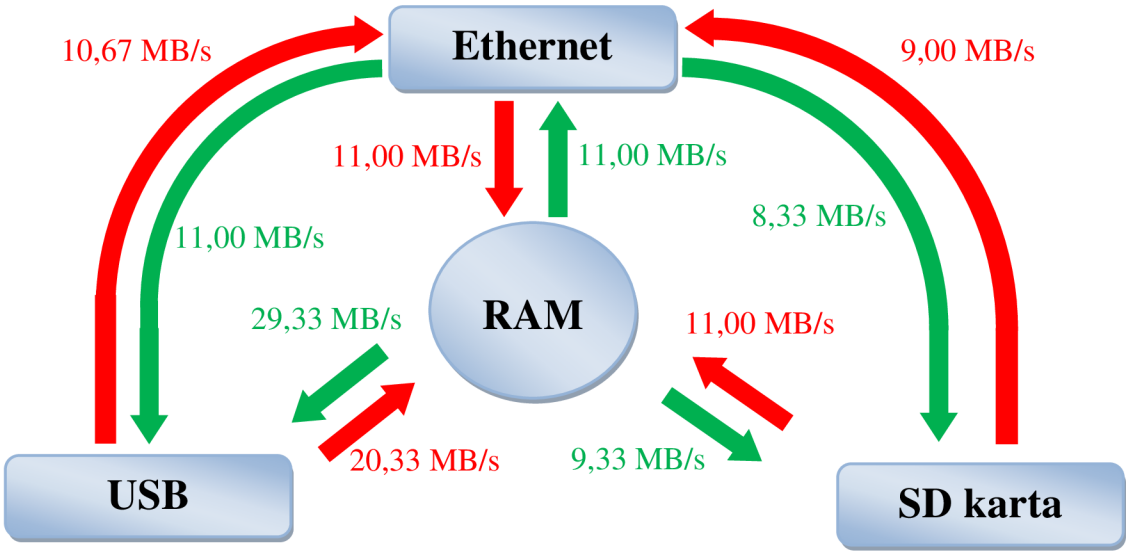
Typ SD karty (formát souborů)	Čtení [MB/s]	Zápis [MB/s]
Sony CD-HC class 10 (NTFS)	9,00	8,33
Kingston SD-HC class 10 (NTFS)	8,33	6,67

Tabulka 2.7: Test přenosové rychlosti SD karty přes ethernet.

Typ SSD (formát souborů)	Čtení [MB/s]	Zápis [MB/s]
SanDis SD7SN3Q256G1002 (NTFS)	11,00	11,00

Tabulka 2.8: Test přenosové rychlosti ethernetu vůči paměti RAM.

Změřená přenosová rychlost flash disku se blíží maximální rychlosti 100 Mbit ethernetu. Z toho lze vyvodit, že přenos není USB hubem nijak omezen.



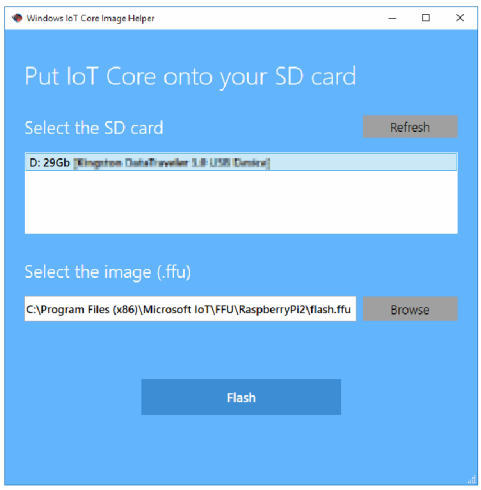
Obrázek 2.6: Grafické zpracování naměřených hodnot přenosové rychlosti.

3. WINDOWS 10 IOT

Windows 10 IoT byl vydán v červenci roku 2015, jako součást jednotného balíku Windows. Jako hlavní výhoda je prezentována možnost tvorby univerzálních aplikací, které běží na klasických desktopech, mobilech a také na zařízení IoT. K využití pro PC nestandardních rozhraní je však nutné přidat knihovnu. Při spuštění této verze se ale nezobrazí standartní uživatelské prostředí. Nachází se zde jen plocha se základními informacemi o připojení k síti a také možnost nastavení sítě. Další ovládání je možné až přes ethernet, pomocí internetového prohlížeče, nebo přes PowerShell [13].

3.1. Instalace

Potřebná aplikace k instalaci Windows jsou na internetových stránkách Microsoftu [14], kde jsou zdarma ke stažení. Stáhne se soubor *iso*, ve kterém je instalační soubor. Po jeho spuštění se nainstalují dvě aplikace. První z nich je WindowsIoTCoreWatcher. Ten slouží ke zjištění IP adresy Raspberry a také k ověření jeho připojení k PC. Druhým programem je WindowsIoTImageHelper, který slouží k instalaci OS na SD kartu. Při jeho spuštění se otevře následující okno (Obrázek 3.1), kde se vybere SD karta, na kterou má být OS nainstalovat (minimální doporučená velikost je 8 GB) a také obraz s Windows 10 IoT. Nepříjemností při následném prvním spuštěním Raspberry je jeho dlouhá zdánlivá nečinnost, při které se zdá, jako by se počítač zasekl, neboť první spuštění trvá přibližně 3 minuty.



Obrázek 3.1: Instalace Windows 10 IoT.

3.2. Konfigurační soubor

SD karta se systémem obsahuje soubor *config.txt*, který slouží pro jednoduché nastavení některých parametrů, jako například rozlišení, paměť GPU, a také možnost přetaktování. Výhodou tohoto nastavení je, že i když uživateli dovolí změny taktů a jiné úpravy, tak ve většině případů nedovolí uživateli počítač poškodit.

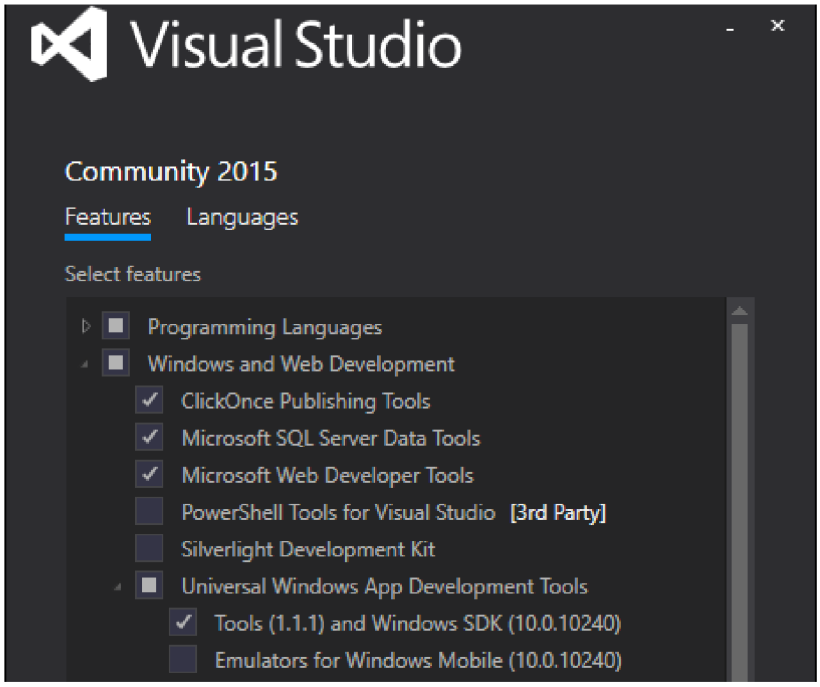
Parametr:	Výchozí hodnota:	Popis:
gpu_mem	32	Velikost paměti RAM přiřazené GPU [MB].
disable_overscan	1	Vypnutí overscan.
init_uart_clock	16000000	Frekvence hodinového signálu pro UART [Hz].
hdmi_group	2	Skupina rozlišení, které je možno použít.
disable_l2cache	0	Vypnutí L2 cache.
force_turbo	0	Zruší dynamický takt, pro maximální výkon (možná ztráta záruky).
over_voltage	0	Zvýšení napětí GPU (krok 0,025 V). Výchozí hodnota je rovna 1,2 V[2].
arm_freq	900	Frekvence ARM [MHz].
gpu_freq	250	Frekvence GPU [MHz].
temp_limit	85	Ochrana proti přehřátí [°C].
decode_MPG2	X	Licence pro MPEG-2.
decode_WVC1	X	Licence pro VC-1.

Tabulka 3.1: Ukázka z možností nastavení [15].

[2] Hodnota 1,2 V odpovídá v textovém souboru číslu 0

4. VISUAL STUDIO

Pro vývoj aplikací pro Raspberry Pi se systémem Windows 10 IoT je nutné mít i na počítači nainstalovaný Windows 10. Další podmínkou je intalace Visual Studia Community. Hlavní výhodou této edice Visual Studia je, že od roku 2015 je ke stažení ze stránek Microsoftu [2] zcela zdarma.
Následně při instalaci je potřeba změnit typ instalace z Deuafalt na Custom. V dalším bodě se přidá vývojářský nástroj pro vývoj universálních aplikací, který se nachází na cestě: Windows and Web Development -> Universal Windows App Development Tools -> Tools and Windows SDK. Tento nástroj je nutné přidat, jelikož univerzální aplikace je jedna ze dvou možných typů aplikací, které lze na Windows 10 IoT spustit.



Obrázek 4.1: Instalace Visual studia.

Po dokončení instalace se ještě musí zapnout vývojářský režim na počítači. Ten se nachází v nastavení, v záložce Aktualizace a zabezpečení – Pro vývojáře. Zde se změní nastavení z *Instalace aplikací bokem* na *Vývojářský režim* a potvrdí [16]. Posledním bodem před začátkem psaní programu je instalace rozšíření pro vývoj aplikací běžících na pozadí, které je ke stažení ze stránek Microsoftu [17].

4.1. Založení projektu

Založení projektu se nachází v nabídce *File – New – Project...* . V dalším kroku se vybere typ aplikace k vytvoření. Pro Windows 10 IoT lze vybrat dva typy projektů.

4.1.1. Headless

Tento typ projektu běží v zařízení jako vlákno na pozadí. Je k nalezení pod názvem *Background Application(IoT),* pod záložkou Templates – Visual C# – Windows – Windows IoT Core. Tímto se vytvoří předpřipravená šablona pro aplikaci.

4.1.2. Headed

Druhý typ se nachází pod názvem Univerzální aplikace Windows, v záložce *Universal*. Jedná se o aplikaci s uživatelským rozhraním. Nevýhodou této aplikace je, že může na Windows 10 IoT běžet současně jen jedna.

Mezi těmito dvěma módy lze přepínat přes PowerShell[3] i samotný Windwos 10 IoT. Pokud je potřeba spouštět aplikace s uživatelským rozhraním, nastaví se Headed (výchozí nastavení). Pokud potřeba není, nastaví se Headless. [18]
Po vybrání z jednoho typu aplikace je potřeba ještě přidat referenci na Windows IoT Extension. Ta se přidá vpravo v kartě Solution Explorer kliknutím pravým tlačítkem na References->Add Reference. Zde již stačí zadat název reference do vyhledávače.

Bylo by dobré ještě krátce představit soubor *Package.appxmanifest.* Jedná se o XML soubor, ve kterém jsou obsaženy informace, které systém potřebuje. Jedná se například o závislosti mezi balíky, požadované schopnosti, vizuální prvky, atd.. Tento soubor je digitálně podepsán a musí jej obsahovat každý balík aplikací. [19]

[3] Příkazy pro nastavení přes PowerShell:

- ***setbootoption.exe*** Zobrazení aktuálního nastavení.
- setbootoption.exe headless Nastavení na Headless.
- setbootoption.exe headed Nastavení na Headed.

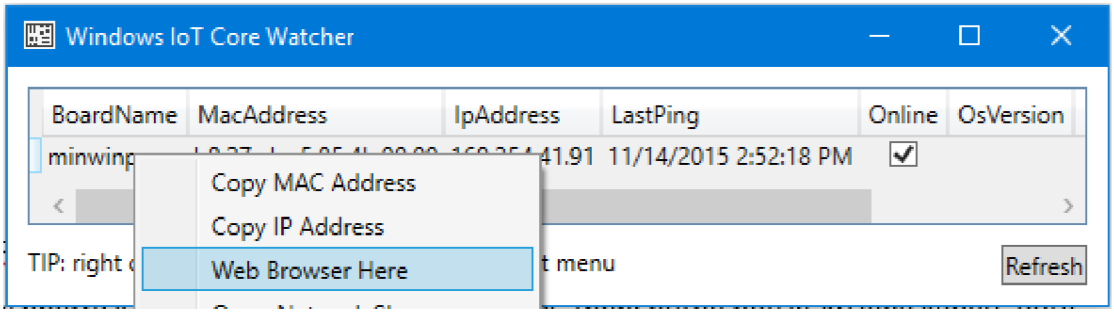
4.2. Vytvoření aplikace připravené k instalaci

Pro instalaci aplikace do zařízení je nutné vytvořit soubor, který bude možné nainstalovat. Tento soubor se vytvoří kliknutím pravým tlačítkem na projekt v Solution Exploreru. Zde se dále vybere možnost *Store* a *Create App Packages*. Tímto se otevře nové okno. Zde se vybere vytvoření aplikace, která není pro Windows store. V dalším kroku se již zobrazí nabídka, kde se vyberou podrobnosti o aplikaci, jako je cesta kam se mají výsledné soubory uložit, verze programu a architektura, pro kterou se má aplikace vytvořit. Soubory vytvořené aplikace se poté nachází ve složce, která byla vybrána.

4.3. Připojení k Raspberry Pi

4.3.1. Webové nastavení [20]

Připojení Raspberry Pi 2 k PC je realizováno pomocí ethernetového kabelu, kterým se může připojit Raspberry k síťovému prvku, nebo přímo k PC (není nutné použít křížený kabel). Dále pomocí programu „WindowsIoTCoreWatcher“, který se nainstaloval dříve, při tvorbě SD karty, se zobrazí připojené Raspberry a také IP adresa. Kliknutím na tuto adresu pravým tlačítkem se zobrazí nabídka, kde je také možnost otevření webového rozhraní („Web browser here“).



Obrázek 4.2: Otevření nastavení pomocí webového prohlížeče.

Přihlašovací údaje [20]:
Jméno: Administrator
Heslo: p@ssw0rd

Zde se poté nachází nastavení, seznam spuštěných aplikací a jiné podrobnosti. Přes webové rozhraní je také možné instalovat aplikace. Tato možnost se nachází na adrese: <IP Adresa zařízení>:8080/AppManager.htm, kde se nahrají dříve vytvořené soubory aplikace a spustí se instalace tlačítkem Go.

4.3.2. Připojení Visual studia

Připojení Visual studia k Raspberry je nutné pro ladění programu, případně i pro nahrání jeho finální verze. K tomu slouží ikonka se zelenou šipkou (Start Debugging). Předtím se však musí ještě vybrat, že aplikace má být přeložena pro ARM procesor a klikne se na Remote Machine.
Po kliknutí na Remote Machine se zobrazí okno pro připojení, kde ve většině případů by mělo být Raspberry automaticky detekováno, to se bohužel občas nestane a je jej nutné připojit manuálně zadáním IP adresy.
Teď již je možné začít ladit aplikace, případně při zvolení „Release“ nahrát hotovou aplikaci do Raspberry. Musíme však vědět, že při debuggingu jede aplikace velmi pomalu.

4.3.3. PowerShell [22]

Jelikož v aktuální verzi OS[4] lze přes webové rozhraní aplikace jen instalovat, tak je zde také ukázáno, jak nastavit spuštění aplikace po startu.

- V prvním kroku se spustí **PowerShell** jako správce.
- Příkazem ***net start WinRM*** se spustí služba pro vzdálené připojení
- Dalším příkazem se povolí připojení zařízení s jeho IP adresou. ***Set-Item WSMan:\localhost\Client\TrustedHosts -Value <IP Adrr>***
- Následně se připojí zařízení. Bude potřebovat heslo, které je: P@ssw0rd. ***Enter-PSSession -ComputerName <IP> -Credential <IP>\Administrator***

Tímto se připojilo k Raspberry Pi a je jej možné vzdáleně ovládat.

Je možné zjistit, které aplikace jsou nastavené pro spuštění při startu, vypsat seznam nainstalovaných aplikací a také přidávat a mazat aplikace které mají být spuštěny při startu. [23]

IotStartup list	Seznam nainstalovaných aplikací.
IotStartup add headed [MyApp] ***IotStartup add headless [Task1]***	Přidání aplikace spouštěné při startu s informací o typu (headed | headless).
IotStartup remove headed [MyApp] ***IotStartup remove headless [Task1]***	Odebrání aplikace spouštěné při startu s informací o typu (headed | headless).
IotStartup startup	Seznam aplikací spouštěných při startu.

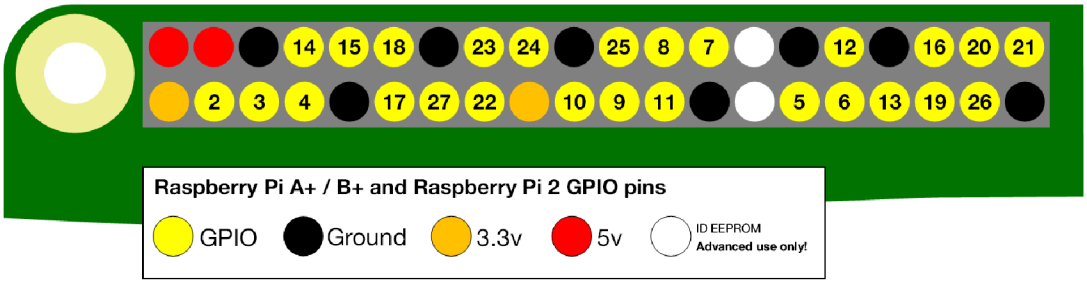
[4] Verze: 10.0.10556.0

5. UKÁZKOVÉ PŘÍKLADY [24][5]

V této kapitole je rozebráno několik ukázkových příkladů, které jsou ke stažení na oficiálních stránkách Microsoftu.

5.1. Blikání LED diody (GPIO)

Pro ukázku funkčnosti GPIO, neboli vstupně/výstupních pinů byl vybrán ukázkový program *BlinkyHeadless,* který rozbliká LED. Dále je také ukázáno rozšíření o vstupní tlačítko. Pokud jsou piny nastaveny jako vstupní, je možné je nastavit jak *pull-up, tak i pull-down* dle potřeby. Dále je lze nastavit jako výstupní piny, kdy na nich dostaneme 3,3 V.



Obrázek 5.1: Rozložení pinů GPIO na desce s jejich číselnými adresami.

Po otevření projektu nalezneme hlavní program pod názvem *startuptask.cs*.

Na začátku souboru je vidět přidaná knihovna `Windows.Devices.Gpio`. Tato knihovna umožňuje právě práci se vstupy a výstupy. O několik řádků níže se nachází třída *StartupTask : IBackgroundTask*, ve které je tělo programu.

První část programu patří deklaraci a inicializaci:

```
BackgroundTaskDeferral deferral;
private GpioPinValue value = GpioPinValue.High;
private const int LED_PIN = 5;
private GpioPin pin;
private ThreadPoolTimer timer;
```

Jako první je zde deklarována proměnná *value* typu *GlioPinValue*, která je inicializována na hodnotu: GpioPinValue. High. Na druhém řádku je inicializována proměnná LED_PIN typu const int, do které je přiřazeno číslo pinu se kterým se bude pracovat (viz Obrázek 5.1). Dále je deklarována třída GpioPin, pomocí které je pin ovládán. Na konec je deklarován timer, který umožní blikání.

[5] Testováno ve verzi 10.0.10240

```
public void Run(IBackgroundTaskInstance taskInstance)
        {
            deferral = taskInstance.GetDeferral();
            InitGPIO();
            timer = ThreadPoolTimer.CreatePeriodicTimer(Timer_Tick,
TimeSpan.FromMilliseconds(500));
        }
```

Při založení projektu je již vytvořena funkce Run, která je volána po spuštění programu. V této funkci se nachází přiřazení třídy *taskInstance.GetDeferral()*, která zaručí, že proces nebude zastaven, nebo pozastaven před dokončením programu. Dále je volána inicializace GPIO (viz dále) a nakonec je zde vytvořen timer, který v nastavené periodě volá funkci s názvem Timer_tick.

```
private void InitGPIO()
        {
            pin = GpioController.GetDefault().OpenPin(LED_PIN);
            pin.Write(GpioPinValue.High);
            pin.SetDriveMode(GpioPinDriveMode.Output);
        }
```

V inicializaci GPIO se nachází na prvním řádku přiřazení zvoleného pinu do proměnné třídy *GpioPin*. Poté je do této třídy zapsána hodnota pinu a nakonec je pin nastaven jako výstupní.

```
private void Timer_Tick(ThreadPoolTimer timer)
        {
            value = (value == GpioPinValue.High) ? GpioPinValue.Low
: GpioPinValue.High;
            pin.Write(value);
        }
```

Funkce, která provádí blikání LED je až na konci, kde při zavolání mění hodnotu pinu na hodnotu opačnou a tu následně zapíše.

5.2. Komunikace přes I2C

Pro ukázku funkčnosti I2C komunikace a grafického rozhraní byl vytvořen program *Blinky*_e, který je založen na ukázkovém programu *I2C Port Expander*. Tento projekt je ve funkčnosti a částečně i kódu podobný kapitole 5.1 s tím rozdílem, že LED je připojena pomocí port expandéru, který komunikuje pomocí protokolu I2C a dále je stav LED zobrazen i na obrazovce.

První část programu patří deklaraci a inicializaci

```
private byte bitMask = 0x01;
private const byte PORT_EXPANDER_I2C_ADDRESS = 0x20;
private const byte PORT_EXPANDER_IODIR_REGISTER_ADDRESS = 0x06;
private const byte PORT_EXPANDER_GP_REGISTER_ADDRESS = 0x00;
private const byte PORT_EXPANDER_OLAT_REGISTER_ADDRESS = 0x02;
private byte iodirRegister;
private byte gpioRegister;
private byte olatRegister;
private I2cDevice i2cPortExpander;

private SolidColorBrush redBrush =
new SolidColorBrush(Windows.UI.Colors.Red);
private SolidColorBrush grayBrush =
new SolidColorBrush(Windows.UI.Colors.LightGray);
```

Zde je deklarována a inicializována maska pro práci s prvním výstupem. Musí zde být nastavena I2C adresa expandéru, v tomto případě 0x20. Dále se zde nachází adresa registru pro nastavení pinu na vstup, nebo výstup. Adresa registru pro čtení hodnoty vstupů (v této ukázce není použit) a registru pro změnu hodnoty výstupu. Poté jsou zde ještě inicializovány byty pro uložení kopie registrů do Raspberry. Nakonec jsou ještě definovány barvy pro vizualizaci.

```
string deviceSelector = I2cDevice.GetDeviceSelector();
var i2cDeviceControllers =
   await DeviceInformation.FindAllAsync(deviceSelector);
var i2cSettings = new
I2cConnectionSettings(PORT_EXPANDER_I2C_ADDRESS);
i2cSettings.BusSpeed = I2cBusSpeed.FastMode;
i2cPortExpander =
   await I2cDevice.FromIdAsync(i2cDeviceControllers[0].Id,
i2cSettings);
```

Dále nalezneme inicializaci rozhraní, do které musí být zadána jen proměnná s I2C adresou expandéru a také se zde zvolí rychlost sběrnice (FastMode/StandardMode).

```
i2cPortExpander.WriteRead(new byte[]
            { PORT_EXPANDER_IODIR_REGISTER_ADDRESS },
            i2CReadBuffer);
iodirRegister = i2CReadBuffer[0];
```

Následně je vytvořena lokální kopie registrů, podobně jako výše, pro všechny tři registry (případně dle potřeby).

```
iodirRegister = bitMask;
i2CWriteBuffer = new byte[] {PORT_EXPANDER_IODIR_REGISTER_ADDRESS,
            iodirRegister };
i2cPortExpander.Write(i2CWriteBuffer);
```

Poté je nastaven pin dle masky jako výstup a ostatní piny jako vstupní.

```
private void Timer_Tick(object sender, object e)
        {
olatRegister = (byte)(((olatRegister & 0x01) == 0x01) ?
olatRegister &
            (byte)(bitMask ^ 0xFF) : olatRegister | bitMask);
i2cPortExpander.Write(new byte[] {
      PORT_EXPANDER_OLAT_REGISTER_ADDRESS, olatRegister }); }
```

V poslední části je zajištěno blikání LED, připojené na expandér. To je zajištěno tím, že při „tiknutí“ časovače, je pomocí masky zapsána do registru hodnota pro zapnutí, případně vypnutí daného výstupu. Tento registr je poté poslán pomocí funkce `i2cPortExpander.Write`, do které je zadán jako parametr adresa požadovaného registru připojeného zařízení.

V této části byly zobrazeny jen vybrané části kódu, pro lepší orientaci a pochopení je nutné se podívat přímo do programu v příloze, kde je kód kompletní.

5.3. Komunikace přes UART[6]

Komunikace přes UART byla otestována vůči počítači, do kterého byl připojen převodník USB to UART. Pro test byl vybrán program *SerialSample*.

```
<Capabilities>
  <DeviceCapability Name="serialcommunication">
    <Device Id="any">
      <Function Type="name:serialPort"/>
    </Device>
  </DeviceCapability>
</Capabilities>
```

Prvním krokem, pro práci se sériovým portem, je přidání kódu výše do souboru *Package.appxmanifest*. Pro zobrazení kódu se na tento soubor klikne pravým tlačítkem a zvolí *View Code*. Přidáním tohoto kódu je zpřístupněna možnost komunikace přes sériový port.

```
private SerialDevice serialPort = null;
DataWriter dataWriteObject = null;
DataReader dataReaderObject = null;
private CancellationTokenSource ReadCancellationTokenSource;
```

Na začátku kódu je inicializaci, kde první řádek inicializuje proměnnou, pomocí které budou nastavovány parametry sériového portu. Následně jsou vytvořeny proměnné pro vstupní a výstupní data, ke kterým se v následujícím kódu přiřadí vstupní a výstupní stream sériového portu. Poslední proměnná slouží k informaci o ukončení příjmu dat.

```
serialPort = await SerialDevice.FromIdAsync(entry.Id);
serialPort.WriteTimeout = TimeSpan.FromMilliseconds(1000);
serialPort.ReadTimeout = TimeSpan.FromMilliseconds(1000);
serialPort.BaudRate = 9600;
serialPort.Parity = SerialParity.None;
serialPort.StopBits = SerialStopBitCount.One;
serialPort.DataBits = 8;
```

Dále je ukázáno nastavení sériového portu. První je do proměnné *serialport* přiřazeno, s kterým portem se bude pracovat. Na dalších dvou řádcích je nastavena maximální doba pro zápis a čtení. Na konec je ještě nastavena přenosová rychlost, paritní bit, ukončovací bit a počet datových bitů.

[6] Testováno ve verzi: 10.0.10556.0

```
Task<UInt32> loadAsyncTask;
uint ReadBufferLength = 1024;
cancellationToken.ThrowIfCancellationRequested();
dataReaderObject.InputStreamOptions = InputStreamOptions.Partial;
loadAsyncTask =dataReaderObject.LoadAsync(ReadBufferLength)
                .AsTask(cancellationToken);
UInt32 bytesRead = await loadAsyncTask;
if (bytesRead > 0)
{
rcvdText.Text = dataReaderObject.ReadString(bytesRead);
status.Text = "\nBytes read successfully!";
}
```

Pokud je připojen sériový port, je vytvořen asynchronní task výše. Poté je vytvořen buffer o velikosti 1024*sizeof(uint). Následně je nastaven token, který signalizuje ukončení tasku, na dalším řádku se nastavuje proměnná, která bude reagovat na paritní bit (přijmutí jednoho byte). Následně je již nastaven a spuštěn samotný asynchronní task, který čeká na data z *serialPort.InputStream* a ukládá je do proměnné, která je následně vypsána na obrazovku.

```
Task<UInt32> storeAsyncTask;
dataWriteObject.WriteString(sendText.Text);
storeAsyncTask = dataWriteObject.StoreAsync().AsTask();
```

Zápis je proveden obdobně jako čtení. Je též vytvořen asynchronní task a následně je načten text k odeslání do objektu *dataWrite*. Zápis je asynchronním taskem dokončen (kvůli rozsahu programu byly popsány jen části, dle mého úsudku, důležité).

5.4. Komunikace přes SPI

Pro vyzkoušení sběrnice SPI, byl vybrán program *PotentiometerSensor*, který komunikuje s A/D převodníkem. Na A/D převodník je připojen potenciometr a na něm nastavená hodnota je zobrazována na obrazovce.

Deklarace a inicializace proměnných pro komunikaci přes SPI

```
private const string SPI_CONTROLLER_NAME = "SPI0";
private const Int32 SPI_CHIP_SELECT_LINE = 0;
private SpiDevice SpiADC;
```

Zde je inicializován název rozhraní, pomocí kterého se bude komunikovat, jelikož Raspberry má dvě sběrnice SPI. Na druhém řádku je poté ještě adresace zařízení na sběrnici (CS0).

```
var settings = new SpiConnectionSettings(SPI_CHIP_SELECT_LINE);
settings.ClockFrequency = 500000;
settings.Mode = SpiMode.Mode0;
string spiAqs = SpiDevice.GetDeviceSelector(SPI_CONTROLLER_NAME);
var deviceInfo = await DeviceInformation.FindAllAsync(spiAqs);
SpiADC = await SpiDevice.FromIdAsync(deviceInfo[0].Id, settings);
```

Dále je již inicializace samotné komunikace. Na prvních třech řádcích je nastavena adresa zařízení, dále frekvence a poté mód hodinového signálu. Dále ještě, volba komunikace pomocí SPI1 a nakonec je vše zapsáno do proměnné SpiADC.

```
byte[] readBuffer = new byte[3];
byte[] writeBuffer = new byte[3] { 0x00, 0x00, 0x00 };
writeBuffer[0] = MCP3208_CONFIG; //0x06
SpiADC.TransferFullDuplex(writeBuffer, readBuffer);
adcValue = convertToInt(readBuffer);
```

Pro vyčtení hodnoty z A/D převodníku jsou vytvořeny (read a write). Do write bufferu je zapsána hodnota 0x06 (start bit + adresa prvního vstupu). Která je plně dulplexní komunikací odeslána. Po příchodu hodnoty z A/D převodníku je ještě hodnota převedena na INT, kvůli zobrazení.

5.5. Ethernetová komunikace

Pro otestování ethernetové komunikace byl vybrán program, který přidává programu z kapitoly 5.1. možnost ovládání pomocí webového rozhraní tím, že vytváří server. Tato aplikace se jmenuje *App2App WebServer* a je typu *BackgroundTask*.

```
<Capabilities>
  <Capability Name="internetClient" />
  <Capability Name="internetClientServer" />
</Capabilities>
```

K vytvoření serveru se musí přidat do schopností (capability), v souboru *Package.appxmanifest*, část kódu viz výše.

```
listener = new StreamSocketListener();
listener.BindServiceNameAsync(port.ToString());
listener.ConnectionReceived += (s, e) =>
```

Dále je vytvořen `StreamSocketListener`, který naslouchá příchozím spojením. Poté je v něm nastaven port, který bude používán a na posledním řádku je indikátor obdržení spojení.

```
StringBuilder request = new StringBuilder();
using (IInputStream input = socket.InputStream)
   {
    byte[] data = new byte[BufferSize];
    IBuffer buffer = data.AsBuffer();
    uint dataRead = BufferSize;
    while (dataRead == BufferSize)
     {
      await input.ReadAsync(buffer, BufferSize,
                                    InputStreamOptions.Partial);
      request.Append(Encoding.UTF8.GetString(data, 0,
data.Length));
      dataRead = buffer.Length;
     }
   }
```

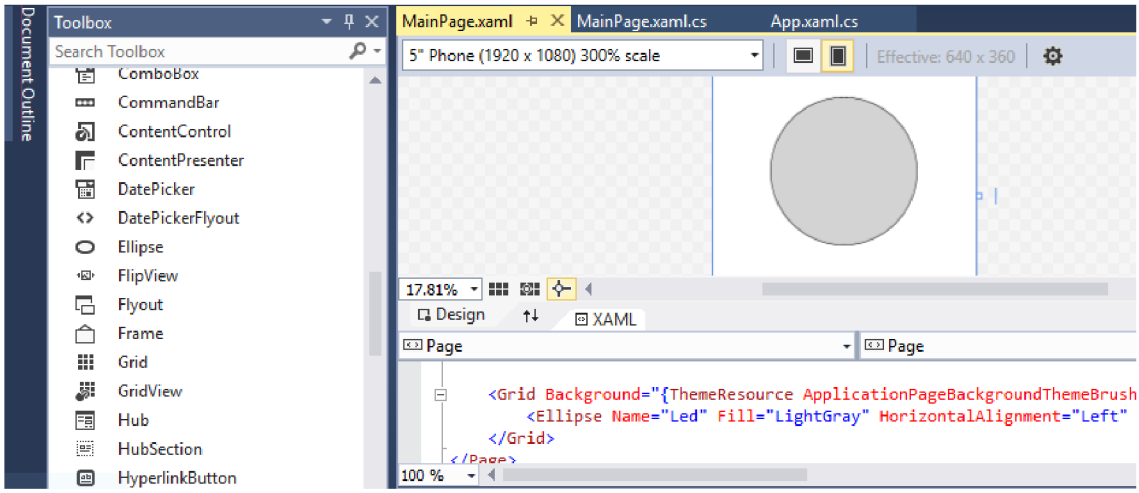
Pro čtení je vytvořena proměnná pro uložení řetězce, dále je vstupní stream přiřazen do proměnné input. Následně jsou vytvořeny proměnné pro uložení byte a buffer. Poté je již čekáno na příchozí data, která jsou po příchodu převedena do správného formátu a uložena.

```
using (IOutputStream output = socket.OutputStream)
{
 using (Stream resp = output.AsStreamForWrite())
  {
   string header = String.Format("HTTP/1.1 200 OK\r\n" +
                                 "Content-Length: {0}\r\n" +
                                 "Connection: close\r\n\r\n",
                                  stream.Length);
   byte[] headerArray = Encoding.UTF8.GetBytes(header);
   await resp.WriteAsync(headerArray, 0, headerArray.Length);
   await resp.FlushAsync();
  }
}
```

Zápis je prováděn obdobně jako čtení. První je výstupní stream přiřazen do proměnné output a následně je do proměnné resp přiřazen stream pro zápis. Následně je uložena odesílaná hlavička HTML souboru do proměnné header. Tato hlavička je překódována do bytů. Poté je zapsána na výstup. Na konec je vymazána vyrovnávací paměť tohoto streamu.

5.1. Grafické rozhraní

Tvorba grafického rozhraní je uzpůsobena pro co nejjednodušší tvorbu grafiky u různých zařízení. Všechny potřebné prvky jsou připraveny v nabídce a stačí je jen přetáhnout na pracovní plochu, kde je s nimi možno dále pracovat.



Obrázek 5.2: Tvorba grafiky.

S grafickým rozhraním se pracuje v souboru *MainPage.xaml (z programu z předchozí kapitoly)*, zde po vybrání z toolboxu požadovaného prvku a jeho vložení na pracovní plochu se prvek zobrazí také níže v kódu a podle jeho jména s ním je možné pracovat dále v kódu. Např. změna barvy výplně, viz níže.

```
Led.Fill = (((olatRegister & 0x01) == 0x01) ? redBrush :
```

6. VÝSLEDKY TESTOVÁNÍ PARAMETRŮ PRO ÚČELY ROBOTIKY

Pro zjištění vhodnosti Windows 10 IoT s Raspberry Pi 2 k použití v robotice a domácí automatizaci byla otestována přesnost časování, jelikož se jedná o jeden z nejdůležitějších parametrů pro robotiku. Jako doplňkový test byla otestována rychlost reakce na přerušení.

6.1. Přesnost časování

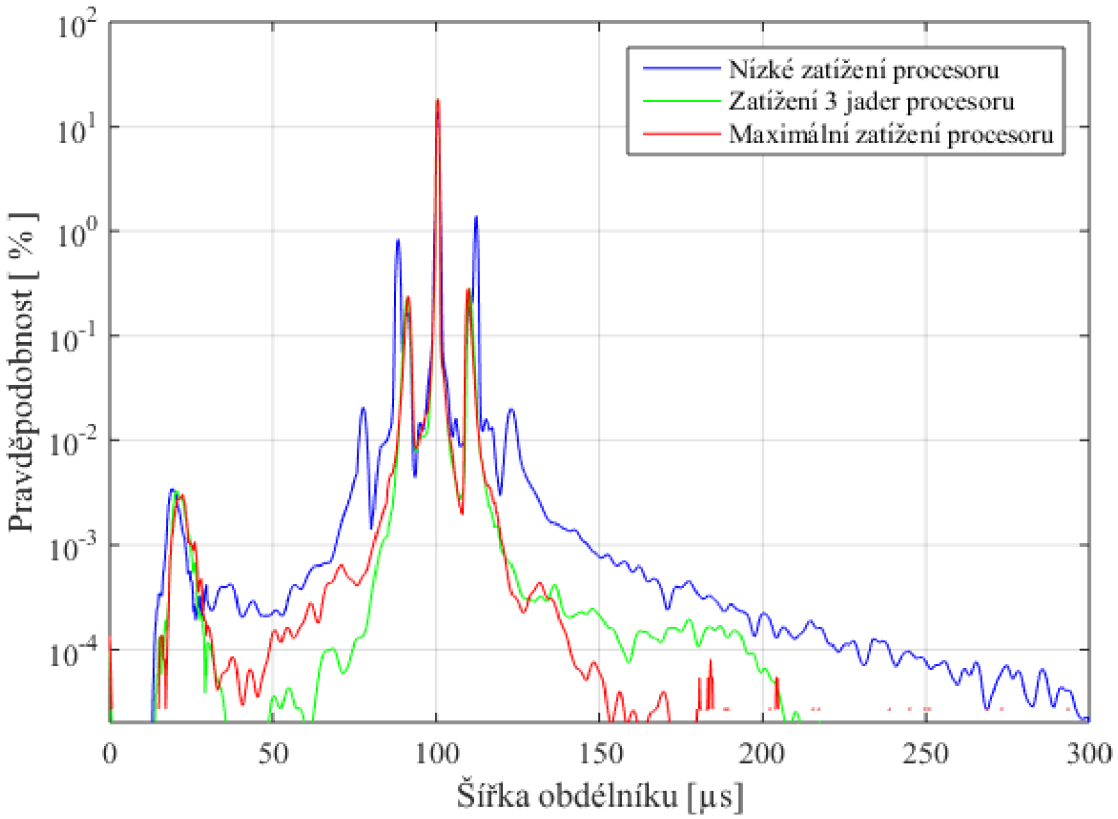
K určení přesnosti časování byl vytvořen program, který zahrnuje dva typy časovačů. Prvním z nich je „*threadpooltimer*“, se kterým je možné dosáhnout přesnosti max. ~16 ms, což je způsobeno časovou základnou, která běží na 64 Hz. Jako druhý byl vytvořen přesný časovač podle doporučení Microsoftu [25]. Tyto časovače byly testovány při třech stavech procesoru. Při prvním testu byl procesor bez zátěže (minimální dosažitelné zatížení), druhý test byl proveden při maximálním zatížení všech jader procesoru. Poslední test byl proveden při maximální zátěži tří ze čtyř jader.

6.1.1. Přesný časovač [25]

U přesného časovače byla pro měření nastavena šířka obdélníkového signálu na 100 µs (měřeno od nástupné hrany k sestupné). Jako nejvíce vypovídající parametr bylo použito procentní vyjádření, kolik času celkově strávil výstup ve špatném stavu (byl delší než 100 µs + 15%) vůči celkovému času měření. Tímto bylo vypočteno, že šířku větší než 115 µs měl puls při minimálním zatížením v 0,7 % času, při zatížení 3 jader v 0,3 % času a při maximálním zatížení v 24,8 % času, z celkové délky měřeni. Dále byl hlavní vrchol aproximován normálním rozložením, viz tabulka 6.1.

Velikost zatížení	**Normální rozložení**		**Medián [µs]**	**Modus [µs]**	**Minimální čas [µs]**	**Maximální čas [µs]**
	Střed [µs]	Rozptyl [µs]				
Minimální	100,49	0,43	100,50	100,50	13,45	1730,80
Maximální tří jader	100,45	0,31	100,48	100,48	0,10	$5{,}42\times10^{4}$
Maximální	100,45	0,34	100,43	100,35	0,10	$2{,}35\times10^{5}$

Tabulka 6.1: Tabulka změřených parametrů časovače.



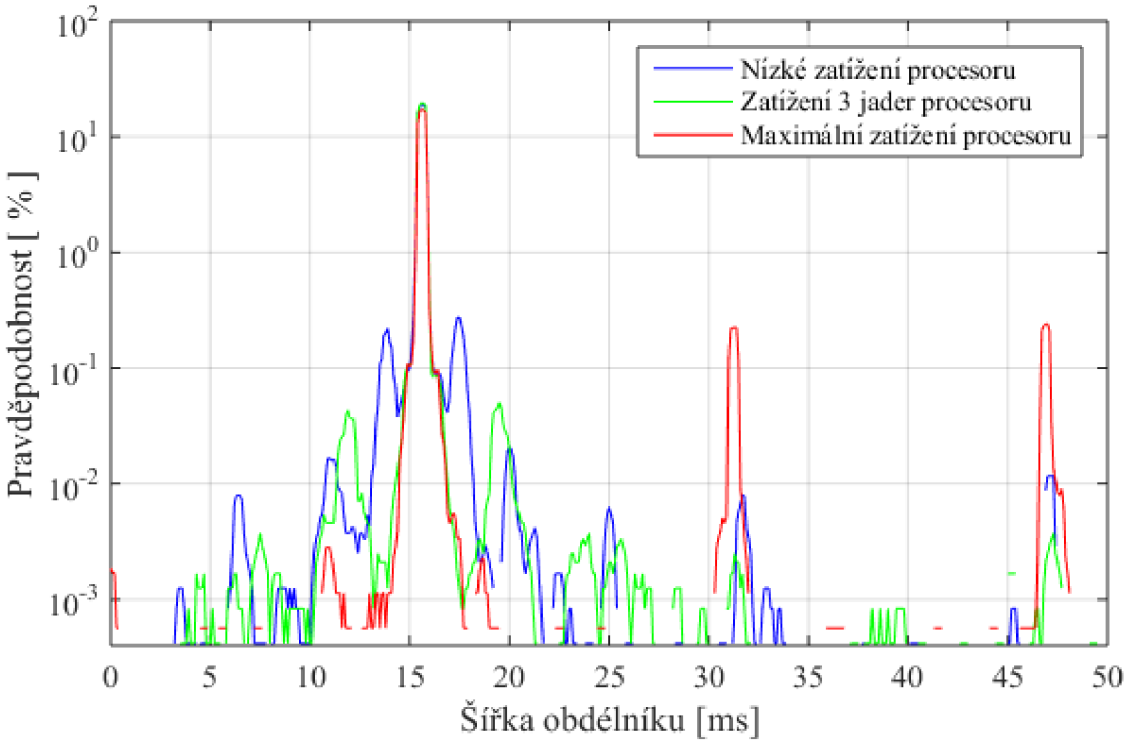
Graf 6.1: Rozložení pravděpodobnosti u časovače s vysokou přesností (vodorovná osa rozdělena na intervaly o šířce 0,2 µs).

6.1.2. Threadpooltimer

U „threadpooltimeru“ byla nastavena šířka obdélníkového signálu nastavena na 15 ms (měřeno od hrany k hraně signálu). Toto nastavení však není možné reálně splnit a je vnitřně použit nejbližší násobek čísla: 1/64 s. Z grafu je také vidět, že i chyby se drží častěji těchto násobků.

Velikost zatížení	Medián [ms]	Modus [ms]	Minimální čas [ms]	Maximální čas [ms]	Obdélníky do šířky 23,43 ms [%]
Minimální	15,62	15,62	3,38	65,39	99,82
Maximální tří jader	15,62	15,62	3,68	78,23	99,84
Maximální	15,63	15,64	$1{,}00\times10^{-4}$	132,67	90,00

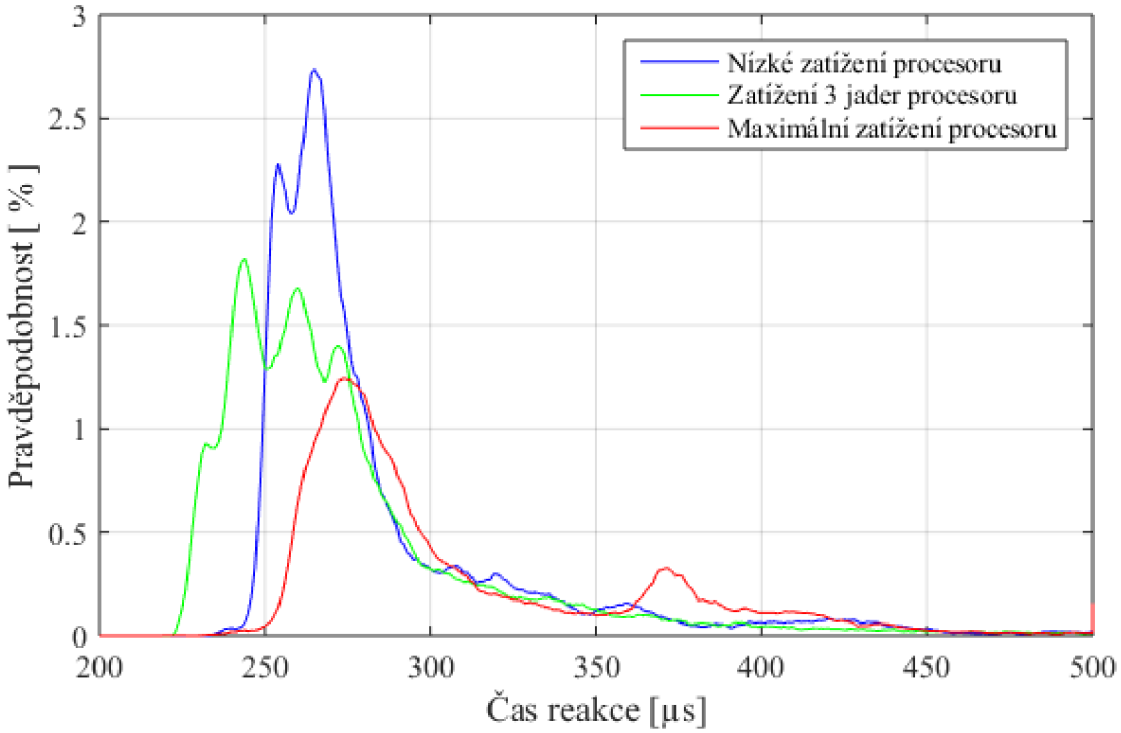
Tabulka 6.2: Tabulka změřených parametrů časovače.



Graf 6.2: Rozložení pravděpodobnosti u threadpooltimeru (vodorovná osa rozdělena na intervaly o šířce 0,1 ms).

6.2. Rychlost reakce na přerušení

V rámci měření, byla také otestována rychlost, s jakou je operační systém schopen zareagovat na přerušení a na jeho základě změnit stav výstupní hodnoty na jednom z pinů. Toto měření je však pouze orientační, protože do něj zasahuje také rychlost, s jakou dokáže ovladač změnit stav výstupu.

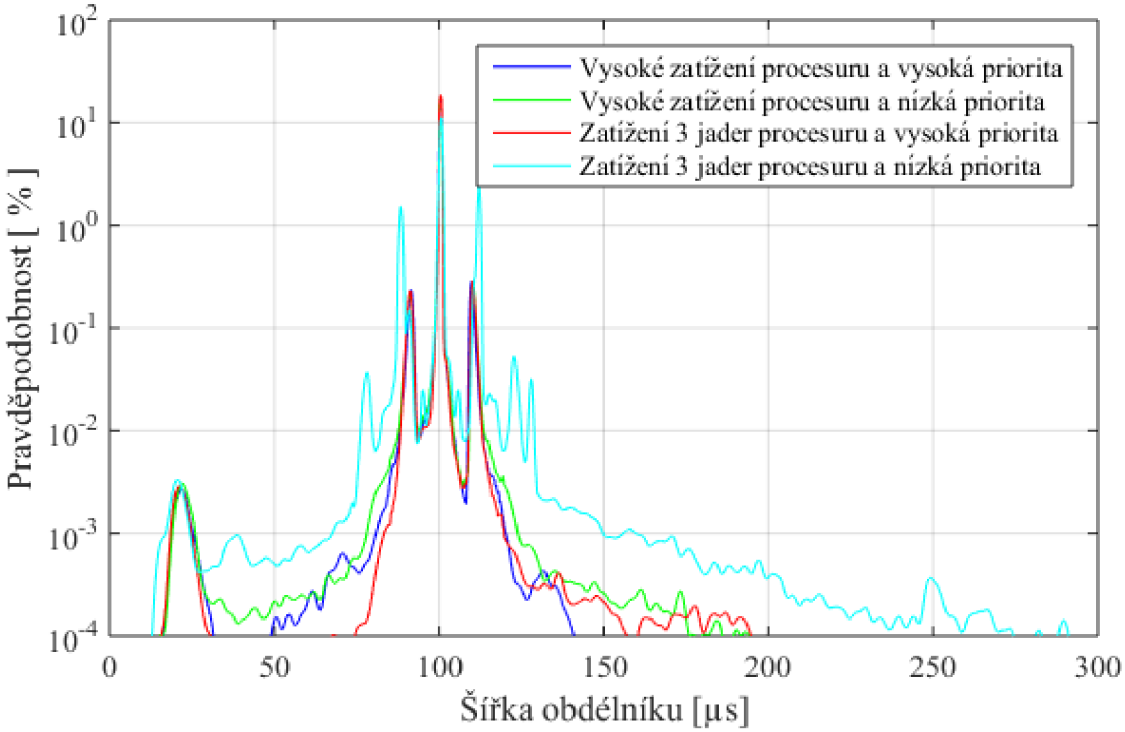


Graf 6.3: Rozložení pravděpodobnosti u reakce na přerušení (vodorovná osa rozdělena na intervaly o šířce 1 µs).

Z měření bylo zjištěno, že reakce na přerušení je při nezatíženém procesoru a při zatížení 3 jader ze čtyř přibližně 230 – 300 µs, a do 500 µs zareaguje systém na přerušení s pravděpodobností 96%. Při zatížení všech jader procesoru se rychlost reakce na přerušení zhorší a pravděpodobnost reakce na přerušení do 500 µs je jen 72%.

6.3. Možnosti priorit (přiblížení k real-time)

Při zkoumání možností priorit bylo zjištěno, že je možné změnit prioritu vlákna, která má 3 úrovně: Low, Normal a High. V rámci vlákna lze dále nastavit několik priorit pro vykonávání asynchronních operací. Byly hledány další možnosti zvýšení priority a tím přiblížení k real-time, avšak nebyly nalezeny. Například syntaxe „System.Diagnostics.Process“, pomocí které by bylo možné zvýšit prioritu, není v UWP podporována. [26] V operačním systému lze prioritu zvýšit, avšak není možné z něj vytvořit real-time operační systém, pro tyto účely je Windows Embedded.



Graf 6.4: Rozložení pravděpodobnosti s různými prioritami (vodorovná osa rozdělena na intervaly o šířce 0,2 µs).

7. WEBOVÁ STRÁNKA

Jelikož jedním z hlavních účelů této bakalářské práce je usnadnit začátek vývoje na této platformě, byla vytvořena webová stránka. Na této webové stránce se nachází velmi zjednodušený návod na instalaci Windows 10 IoT. Dále je zde popsána instalace Visual studia a informace potřebné pro vytvoření a spuštění aplikace na Raspberry Pi 2. Také jsou zde krátce popsány komunikační rozhraní s drobnostmi, na které si nový uživatel musí dát pozor.

Tato stránka je zveřejněna na internetové adrese:

sites.google.com/a/vutbr.cz/kambot/doc/sw/rpiwin

8. ZÁVĚR

Na základě získaných informací byla vypracována rešerše o Raspberry Pi 2 a jeho možnostech s nainstalovanými Windows 10 IoT. Byla popsána instalace jak Windows, tak i Visual studia, s důrazem na nestandardní drobnosti. Dále byly ukázány typy aplikací, které lze v této verzi Windows spustit. Jak tyto aplikace debugovat na zařízení přímo z notebooku, jak ji nahrát na Raspberry Pi 2 a jak nastavit její spuštění při startu.

Byly také popsány vstupně-výstupní piny GPIO a komunikační rozhraní SPI, I2C, UART. Na tyto rozhraní, navíc s ethernetovým portem, byly vyzkoušeny a popsány ukázkové aplikace, které prověřily jejich funkčnost.

Pro zjištění vhodnosti pro použití v domácí automatizaci a robotice byla otestována přenosová rychlost USB rozhraní a SD karty a také rychlost mezi ethernetem a USB/SD kartou. Dále byla také ověřena přesnost časování a rychlost reakce na přerušení. Na základě těchto testů bylo zjištěno, že Raspberry Pi 2 s Windows 10 IoT je vhodné pro domácí automatizaci, pro robotiku ale méně. Důvodem je, že pro robotiku dosahuje malé přesnosti časování i při nastavené nejvyšší prioritě. Pro domácí automatizaci tak přesné časování nutné není a má spoustu vstupů a výstupů s podporou komunikačních protokolů. Dále má dostatečně rychlé USB a Ethernetové rozhraní (~10MB/S) a uživatelsky přívětivé programátorské prostředí.

Nejdůležitější informace pro začátek vývoje na Raspberry Pi 2 s Windows 10 IoT byly také zpracovány ve formě webové stránky.

9. SEZNAM POUŽITÉ LITERATURY

[1] LiFi Labs, Inc. [online] [cit. 7. 11. 2015]. Dostupné z WWW: <www.lifx.com>

[2] Microsoft s.r.o. [online] [cit. 7. 11. 2015]. Visual Studio Community. Dostupné z WWW: <www.visualstudio.com/products/visual-studio-community-vs>

[3] Microsoft s.r.o. [online] [cit. 7. 11. 2015]. The Internet of your things. Dostupné z WWW: <dev.windows.com/en-us/iot>

[4] Raspberry Pi Foundation. [online] [cit. 7. 11. 2015]. Raspberry Pi 2 model b. Dostupné z WWW: <www.raspberrypi.org/products/raspberry-pi-2-model-b/>

[5] Embedded Linux Wiki. [online] [cit. 7. 11. 2015]. RPi Hardware. Dostupné z WWW: <elinux.org/RPi_Hardware>

[6] Raspberry Pi Foundation. [online] [cit. 20. 3. 2016]. Raspberry Pi 3 model b. Dostupné z WWW: < www.raspberrypi.org/products/raspberry-pi-3-model-b/ >

[7] Embedded Linux Wiki. [online] [cit. 7. 11. 2015]. RPi Low-level peripherals. Dostupné z WWW: <http://elinux.org/RPi_Low-level_peripherals>

[8] Wikipedia. [online] [cit. 7. 11. 2015]. I²C. Dostupné z WWW: < cs.wikipedia.org/wiki/I²C >

[9] Raspberry Pi Foundation. [pdf] [cit. 7. 11. 2015]. BCM2835-ARM-Peripherals (str. 21). Dostupné z WWW: < www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf>

[10] Raspberry Pi Foundation. [pdf] [cit. 7. 11. 2015]. BCM2835-ARM-Peripherals (str. 148). Dostupné z WWW: < www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf>

[11] Raspberry Pi Foundation. [pdf] [cit. 7. 11. 2015]. BCM2835-ARM-Peripherals (str. 11). Dostupné z WWW: < www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf>

[12] Microsoft s.r.o. [online] [cit. 26. 4. 2016]. Diskspd Utility. Dostupné z WWW: < gallery.technet.microsoft.com/DiskSpd-a-robust-storage-6cd2f223 >

[13] Microsoft s.r.o. [online] [cit. 7. 11. 2015]. Windows 10 Coming to Raspberry Pi. Dostupné z WWW: < blogs.windows.com/buildingapps/2015/02/02/ windows-10-coming-to-raspberry-pi-2/>

[14] Microsoft s.r.o. [online] [cit. 7. 11. 2015]. Downloads and Tools. Dostupné z WWW: <ms-iot.github.io/content/en-US/Downloads.htm>

[15] Embedded Linux Wiki. [online] [cit. 7. 11. 2015]. RPiconfig. Dostupné z WWW: <elinux.org/index.php?title=RPiconfig>

[16] Microsoft s.r.o. [online] [cit. 7. 11. 2015]. Set up your PC. Dostupné z WWW: <ms-iot.github.io/content/en-US/win10/SetupPCRPI.htm>

[17] Microsoft s.r.o. [online] [cit. 7. 11. 2015]. Windows IoT Core Project Templates. Dostupné z WWW: < visualstudiogallery.msdn.microsoft.com/ 55b357e1-a533-43ad-82a5-a88ac4b01dec >

[18] Microsoft s.r.o. [online] [cit. 2. 2. 2016]. Windows IoT Core Project Templates. Dostupné z WWW: < ms-iot.github.io/content/en-US/win10/ HeadlessMode.htm >

[19] Microsoft s.r.o. [online] [cit. 2. 1. 2016]. App package manifest. Dostupné z WWW: < msdn.microsoft.com/en-us/library/windows/ apps/br211474.aspx >

[20] Microsoft s.r.o. [online] [cit. 7. 11. 2015]. Using WiFi on your Windows 10 IoT Core device. Dostupné z WWW: < ms-iot.github.io/content/en-US/win10/SetupWiFi.htm>

[21] Microsoft s.r.o. [online] [cit. 7. 11. 2015]. Blinky Sample. Dostupné z WWW: < ms-iot.github.io/content/en-US/win10/samples/Blinky.htm >

[22] Microsoft s.r.o. [online] [cit. 1. 3. 2016]. Using PowerShell. Dostupné z WWW: < ms-iot.github.io/content/en-US/win10/samples/ PowerShell.htm >

[23] Microsoft s.r.o. [online] [cit. 1. 3. 2016]. Windows Core Command Line Utils. Dostupné z WWW: < ms-iot.github.io/content/en-US/win10/ tools/CommandLineUtils.htm >

[24] Microsoft s.r.o. [online] [cit. 7. 11. 2015]. Docs and Samples. Dostupné z WWW: < ms-iot.github.io/content/en-US/win10/StartCoding.htm >

[25] GitHub, Inc. [online] [cit. 20. 2. 2016]. ServoMotorBasics. Dostupné z WWW: < github.com/ms-iot/samples/blob/develop/ServoMotorBasics/CS/ ServoMotorBasics/StartupTask.cs >

[26] Microsoft s.r.o. [online] [cit. 15. 2. 2016]. Get process in UWP. Dostupné z WWW: < social.msdn.microsoft.com/Forums/en-US/487290bd-bbf1-456c-a599-f909791a198d/uwp-get-process-in-uwp?forum=wpdevelop >

10. SEZNAM OBRÁZKŮ

11. SEZNAM TABULEK

12. SEZNAM GRAFŮ

13. SEZNAM PŘÍLOH

Příloha 1: Ukázkový program Blinky_e.
Příloha 2: Ukázkový program BlinkyHeadless.
Příloha 3: Ukázkový program PotentiometerSensor.
Příloha 4: Ukázkový program SerialSample.
Příloha 5: Ukázkový program App2App WebServer.
Příloha 6: Software Windows_10_IoT_Core_RPi2.msi.
Příloha 7: Testovací program časování a přerušení Timers.
Příloha 8: Testovací program pro zatížení procesoru LOADCPU.
Příloha 9: Naměřená data přesnosti časování.