Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

# Bakalářská práce

# Raspberry Pi jako řídící systém s displejem

Plzeň 2022 Pavel Třeštík

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

| | |
|---|---|
| Jméno a příjmení: | **Pavel TŘEŠTÍK** |
| Osobní číslo: | **A17B0380P** |
| Studijní program: | **B3902 Inženýrská informatika** |
| Studijní obor: | **Informatika** |
| Téma práce: | **Raspberry Pi jako řídící systém s displejem** |
| Zadávající katedra: | **Katedra informatiky a výpočetní techniky** |

## Zásady pro vypracování

1. Prostudujte problematiku užití Raspberry Pi jako základu řídícího systému a definujte případná omezení.
2. Navrhněte a realizujte vhodné hardwarové a softwarové řešení zahrnující možnost připojení různých čidel, paralelního LCD a klávesnice.
3. Implementujte jednoduchou řídící aplikaci s grafickým rozhraním a záznamem dat 24/7.
4. Ověřte funkci a časovou odezvu systému.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Tomáš Mainzer, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **4. října 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

L.S.

**Doc. Ing. Miloš Železný, Ph.D.**
děkan

**Doc. Ing. Přemysl Brada, MSc., Ph.D.**
vedoucí katedry

# Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 23. června 2022

Pavel Třeštík

# Abstract

This bachelor thesis is focused on the research of using a Raspberry Pi as a basis of a control system with a display. Based on the researched issues, a suitable solution is then proposed, including the use of various sensors, parallel LCD and a keyboard and their limitations. The last step of the thesis is to implement a simple control application that allows continuous recording of the data collected from used sensors.



# Abstrakt

Tato bakalářská práce je zaměřena na prozkoumání problematiky použití zařízení Raspberry Pi jako základ řídícího systému s displejem. Na základě prozkoumané problematiky je potom navrženo vhodné řešení zahrnující použití různých čidel, paralelního LCD a klávesnice a jejich omezení. Posledním krokem práce je implementace jednoduché řídící aplikace, která umožní nepřetržitý záznam dat nasbíraných z použitých čidel.

# Poděkování

Tímto děkuji panu Ing. Tomáši Mainzerovi, Ph.D. za odborné vedení bakalářské práce, rady a trpělivost při jejím vypracování.

# Obsah

# 1 Úvod

Řídící systémy jsou dnes téměř nepostradatelnou součástí v průmyslu, ale lze na ně narazit i v domácnosti. Jedná se o zařízení, která umožňují monitorovat, ovládat a samostatně řídit jiné zařízení a čidla. Příkladem z domácnosti může být například termostat[6]. V průmyslu se potom může jednat o terminál ovládající celou výrobní linku nebo její části.

V průmyslu se využívají různé stroje a zařízení již řadu let a některé produkty není možné bez jejich pomoci vyrobit. Stroje jsou přesnější a většinou rychlejší než lidé, proto je snaha o automatizaci, tedy nechat stroje udělat co možná největší část výrobního procesu. K jejich jednotlivému nebo i skupinovému ovládání pak slouží právě řídící systém.

Cílem této bakalářské práce je vytvořit řídící systém využitím micro počítače Raspberry Pi. Práce má tři hlavní úkoly, které by měla splnit. Prvním úkolem je prozkoumat, jaké schopnosti řídícího systému Raspberry Pi může poskytnout. Druhým úkolem je prozkoumat hardware potřebný k realizaci řídícího systému. V rámci této části je nutné vzít v potaz softwarovou podporu zvoleného hardwaru a jejich omezení. Třetím úkolem práce je implementovat aplikaci s grafickým uživatelským rozhraním, která umožní takovýto systém monitorovat a ovládat.

# 2 Využití Raspberry Pi jako základ řídícího systému

Raspberry Pi si už nějakou dobu razí cestu do průmyslu. Potenciálně má využití na mnoha pozicích a velká univerzálnost a výkon za nízkou cenu je dobrým lákadlem. Přibližně 44% všech Raspberry Pi je prodáváno průmyslům[3]. Produkty podobné cíli této práce již existují a například firma Comfile Technology nabízí několik variant produktů, využívající Compute Module 3+ jako základ. Tyto produkty se nazývají ComfilePi (označeny jako CPi).

Hlavním cílem práce je prozkoumat problematiku použití Raspberry Pi jako základ řídícího systému a následně navrhnout a vytvořit takovýto řídící systém. Výsledek práce by měl umožňovat monitorovat a ovládat čidla, jejich skupiny a jiné podsystémy. Aby toho práce dosáhla, musí se k Raspberry Pi připojit minimálně displej a klávesnice pro ovládání systému a minimálně jedno čidlo, které bude ovládáno či monitorováno. V budoucnu by systém také mohl být rozšířen o další čidla, například globální polohový systém známý jako GPS a nebo třeba kamera. Další rozšíření mají však už nižší prioritu a nemusí být součástí tohoto projektu.

Vzhledem k tomu, že řídící systémy jsou využívány hlavně v průmyslu, tak se musí, především při hardwarové realizaci, brát ohledy na omezení a je nutné tato omezení prozkoumat.

Požadavky na softwarovou část jsou takové, aby byl systém Raspberry Pi real-time, omezil zápis a čtení ze Secure Digital (SD) karty, na které se systém nachází a zajistil správnou funkčnost připojeného hardware. Další požadavek na software je navrhnout a implementovat aplikaci s uživatelským rozhraním, která bude zobrazovat přijímaná data, umožní ovládat stroj, či čidlo generující data a bude je zaznamenávat. Posledním nutným požadavkem na tuto aplikaci je, aby běžela nepřetržitě.

# 3 Možnosti Raspberry Pi jako řídící systém

Prvním úkolem práce je prozkoumat možnosti řídícího systému, které Raspberry Pi nabízí.

## 3.1 Hardware

V této sekci je seznámení a analýza různých hardware možností, které mohou být požadovány po řídícím systému. Při volbě hardware je potřeba vzít v potaz prostředí, ve kterém se bude řídící systém vyskytovat. Podle prostředí lze určit například teplotní limity nebo odolnost proti prachu a jiným částicím a tak dále. Na základě určených parametrů se poté vybírá vhodný hardware.

### 3.1.1 Raspberry Pi

Raspberry Pi Foundation je charitativní společnost sídlící ve Velké Británii. Jejich cílem bylo vytvořit levné a snadno dostupné jednodeskové (single-board) počítače, určené hlavně pro výuku počítačových technologií. Tento micro počítač ovšem získal na popularitě i v jiných oblastech než ve výuce a dnes je často používán hobbyisty a začíná se rozšiřovat i v průmyslu.

Raspberry Pi nabízí dva druhy micro počítačů. Raspberry Pi modely a Compute Module modely. Compute Module jsou zaměřeny pro průmyslové využití, ale potřebují Input/ Output (IO) desku, bez které nemají žádné použitelné vstupní ani výstupní rozhraní. Na následujícím obrázku (Obrázek 3.1) jsou pro porovnání uvedeny specifikace modelů Raspberry Pi 3 Model B+ a Compute Module 3+.

| Board | Raspberry Pi 3 Model B+ |
|---|---|
| Documentation | Product brief |
| Launch | March 2018 |
| Processor | BCM2837 |
| Core | Quad-core Cortex-A53, ARM v8 |
| Speed | 1.4GHz |
| RAM | 1GB, LPDDR2 |
| eMMC Flash | |
| GPIO | 40 pin |
| Ethernet | 300Mbps max |
| USB | 4 × USB 2.0 |
| HDMI ports | 1 × HDMI |
| Camera / display ports | CSI DSI |
| Wireless | 2.4GHz & 5.0GHz |
| Bluetooth | Bluetooth 4.2, BLE |
| PoE-enabled | Yes |
| SD support | MicroSD slot |
| Price | $35 |
| Status | LTB not before January 2026 |

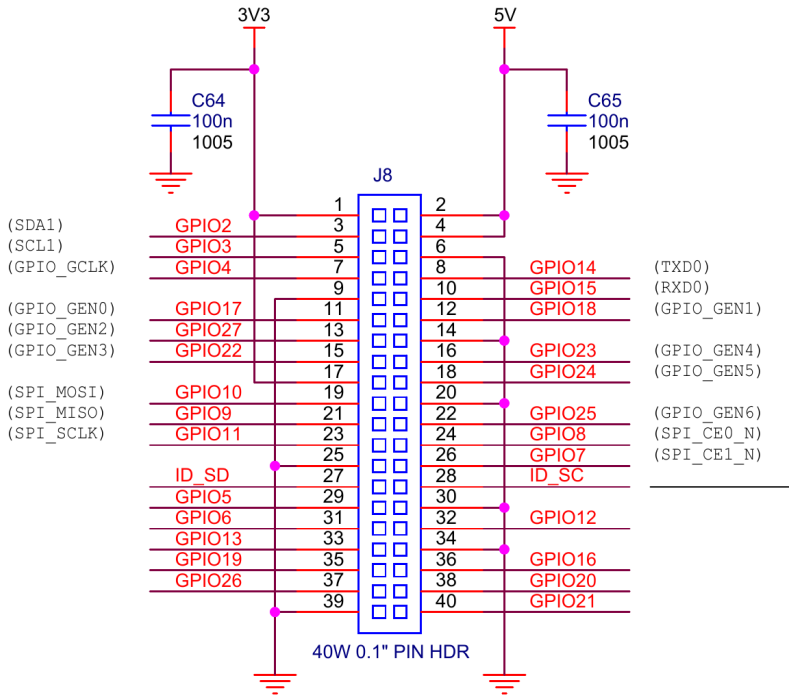
| Raspberry Pi Compute Module 3+ | Board |
|---|---|
| Datasheet | Documentation |
| February 2019 | Launch |
| BCM2837 | Processor |
| Quad-core Cortex-A53, ARM v8 | Core |
| 1.4GHz | Speed |
| 1GB LPDDR2 | RAM |
| 8GB, 16GB, 32GB or Lite (no eMMC) (depending on variant) | eMMC Flash |
| 46 via SODIMM | GPIO |
| | Ethernet |
| USB via SODIMM | USB |
| HDMI via SODIMM | HDMI ports |
| CSI / DSI via SODIMM | Camera / display ports |
| | Wireless |
| | Bluetooth |
| | PoE-enabled |
| n/a | SD support |
| $25 - $40 | Price |
| LTB not before January 2026 | Status |

Obrázek 3.1: Porovnání Raspberry Pi 3B+ a Compute Module 3+

### Raspberry Pi GPIO

General-Purpose Input/ Output (GPIO) je pinové rozhraní a je jedním z hlavních důvodů oblíbenosti Raspberry Pi produktů. Všechny modely momentálně dostupné Raspberry Pi nabízejí 40 pinové rozhraní. Na obrázku (Obrázek 3.2) je schéma těchto pinů. Některé piny, například 1, 2, 39, mají danou funkci a tu není možné změnit. Piny označené GPIOxx, jsou piny, které jsou programovatelné. Uživatel je může nastavit na vstup/ výstup nebo je využít pro přerušení[12]. Tyto GPIO piny mají často přednastavené funkce, jako například piny 19, 21, 23, které jsou určené pro Serial Peripheral Interface (SPI) sběrnici. Většina GPIO má alespoň jednu alternativní funkci

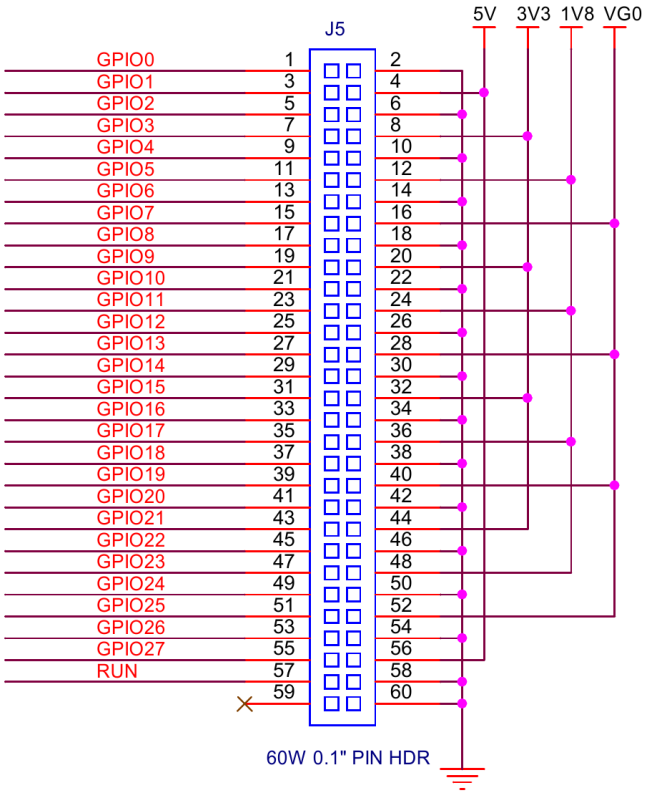
a tato funkce je volitelná uživatelem. Raspberry Pi modely mají celkem 26 GPIO z celkových 40 pinů.



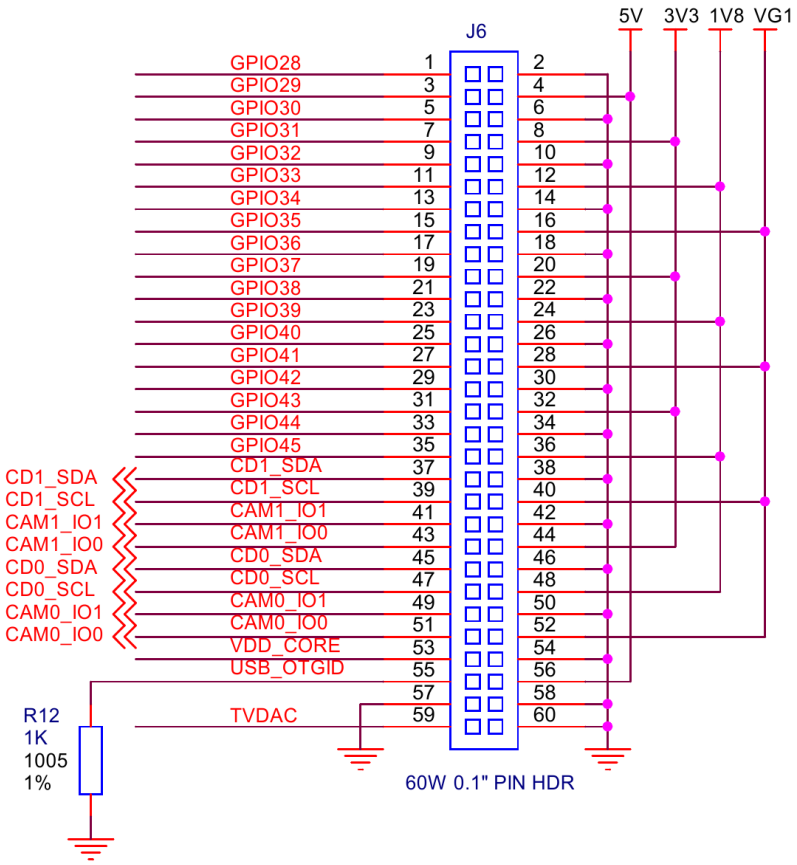

Obrázek 3.2: Raspberry Pi 3B+ pin header

### Compute Module IO Board GPIO

Compute Module IO Board má dvě 60ti pinové rozhraní. Z toho má dohromady 46 programovatelných GPIO. Prvních 26 GPIO má stejné funkce jako běžné Raspberry Pi, protože starší Raspberry Pi i Compute Module používají stejné procesory řady BCM283x. Tyto procesory mají velmi podobnou architekturu. To však neplatí pro nejnovější Raspberry Pi a Compute Module, jejichž architektura se už více liší. Nejnovější Compute Module 4 IO Board má už pouze standardní Raspberry Pi GPIO header se 40ti piny. Na obrázcích (Obrázek 3.3 a Obrázek 3.4) jsou schémata pinového rozhraní Compute Module IO Board pro modely Compute Module 1, 3, 3+.




Obrázek 3.3: Compute Module IO Board J5 headers




Obrázek 3.4: Compute Module IO Board J6 headers

### 3.1.2 Displej

Displej je nepostradatelnou součástí řídícího systému. Protože řídící systém musí informovat obsluhu o hodnotách nebo postupu procesu, je nutné tyto informace nějak zobrazit a umožnit tak jejich následnou kontrolu a ovládání.

V této práci jsou na displej kladeny velké nároky v několika oblastech. První oblastí je teplotní rozsah. Čím extrémnější teploty displej zvládne, tím lepší, ovšem není to hlavní charakteristika, kterou u displeje hledáme. Hlavní charakteristikou je svítivost (jas). Například běžné televizní obrazovky mají kolem 100 - 500 nitů svítivosti [2]. Takové obrazovky by ale mohly být těžko čitelné v jasně osvětlených prostorech. Proto hledaná svítivost displeje je kolem 1000 nit, což dělá displej téměř čitelný na slunci[2], to je ale určené i jinými vlastnostmi displeje, které už nejsou pro tuto práci tolik relevantní. V poslední řadě jednou ze žádaných vlastností je velikost displeje. Ačkoliv nebyla pevně stanovena přesná hranice, displej by měl mít maximálně do 7”

úhlopříčku.

Dále je třeba vyřešit způsob připojení displeje. Většina Raspberry Pi modelů umožňuje připojení displeje pomocí HDMI a novější modely pomocí mini HDMI. Pro běžné užití Raspberry Pi je HDMI nejlepší možnost připojení displeje, ale HDMI displeje mohou být drahé a pro tuto práci často nedostatečné, protože většina nesplňuje některý z výše uvedených požadavků. Raspberry Pi má další způsob připojení displeje a to pomocí Parallel Display Interface (DPI) přes pinové rozhraní. Díky tomuto je možné připojit více typů displejů, ale za cenu použití velkého počtu pinů.

DPI umožňuje připojit až 24-bitový paralelní RGB displej. Pro takzvané „true color“ je displej připojen v konfiguraci RGB24 (tedy 8 bitů pro každou barvu). Tato konfigurace využije všechny volné GPIO, jejichž primární funkce musí být pro použití DPI vypnuté. Displej je také možné připojit v konfiguracích RGB666 (6 bitů na každou barvu) a RGB565 (5 bitů pro dvě barvy a 6 bitů pro třetí). Výhoda použití RGB666 nebo RGB565 je, že zbydou některé piny volné a mohou být využity pro jiné účely. V následujícím obrázku (Obrázek 3.5) jsou zobrazeny možná připojení těchto konfigurací na pinové rozhraní.

| Mode | RGB bits | GPIO | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 |
| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 2 | 565 | - | - | - | - | - | - | - | - | 7 | 6 | 5 | 4 | 3 | 7 | 6 | 5 | 4 | 3 | 2 | 7 | 6 | 5 | 4 | 3 |
| 3 | 565 | - | - | - | 7 | 6 | 5 | 4 | 3 | - | - | 7 | 6 | 5 | 4 | 3 | 2 | - | - | - | 7 | 6 | 5 | 4 | 3 |
| 4 | 565 | - | - | 7 | 6 | 5 | 4 | 3 | - | - | - | 7 | 6 | 5 | 4 | 3 | 2 | - | - | 7 | 6 | 5 | 4 | 3 | - |
| 5 | 666 | - | - | - | - | - | - | 7 | 6 | 5 | 4 | 3 | 2 | 7 | 6 | 5 | 4 | 3 | 2 | 7 | 6 | 5 | 4 | 3 | 2 |
| 6 | 666 | - | - | 7 | 6 | 5 | 4 | 3 | 2 | - | - | 7 | 6 | 5 | 4 | 3 | 2 | - | - | 7 | 6 | 5 | 4 | 3 | 2 |
| 7 | 888 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Obrázek 3.5: Možnosti připojení DPI displeje

### 3.1.3 Klávesnice

Klávesnice je další nezbytná část ke složení řídícího systému. Asi největším problémem pro klávesnici v průmyslu je prach, vlhkost a další nečistoty vyskytující se v pracovním prostředí. Běžné mechanické klávesnice by v takovémto prostředí musely být zakryté nebo jinak izolované před okolními vlivy. Existuje typ fóliové klávesnice, někdy také nazývané membránové, která se skládá z několika vrstev a je velmi odolná proti prachu a vlhkostem, protože klávesnice je téměř jeden jednolitý kus.

Připojení takovéto klávesnice je realizováno pomocí IO pinů. Počet IO pro

připojení takovéto klávesnice ale není malý. Například pro připojení 4x3 nebo 4x4 maticové klávesnice je třeba využít 7 nebo 8 IO pinů. Tuto nevýhodu je možné alespoň částečně omezit přidáním mikrokontroléru, který může sloužit jako rozšíření IO pinů. Alternativní možností je připojit klávesnici pomocí USB.

### 3.1.4 Interní Sběrnice

Jsou sběrnice, které Raspberry Pi nabízí použitím GPIO pinů.

#### UART

Universal Asynchonous Reciever/ Transmitter (UART). Slouží pro asynchroní komunikaci mezi dvěma zařízeními. To znamená, že UART nepoužívá hodinový signál a k jednosměrné komunikaci stačí pouze jeden vodič. Pro obousměrnou komunikaci jsou třeba vodiče dva. Nevýhodou je, že obě zařízení musí mít dohodnuté parametry a komunikační protokol, aby nedošlo ke korupci dat.

#### $I^2C$

Jedná se o synchronní sériovou sběrnici. Umožňuje připojení více „master“ i „slave“ zařízení. Je využívána pro připojení periferií na krátké vzdálenosti, které nepotřebují vysokou rychlost přenosu.

Komunikuje pomocí zpráv. Zprávy mají START a STOP podmínku. V každé zprávě START podmínku následuje 7mi nebo 10ti bitová adresa, podle které ostatní zařízení připojená na sběrnici poznají, že jsou příjemcem zprávy. Dále se ve zprávě posílá Read/ Write bit, který určuje, jestli odesílající data posílá nebo žádá a také jsou v ní obsaženy datové rámce. Příjemce také posílá ACK/ NACK signály, takže vysílající ví, zda komunikace byla úspěšná.

Hlavními výhodami jsou pouze 2 vodiče nutné pro komunikace, ACK/ NACK signály a adresování, které umožňuje propojení více „master“ a „slave“ zařízení jednou sběrnicí.

Hlavními nevýhodami by potom byla přenosová rychlost, omezený počet zařízení, které lze adresovat 7mi nebo 10ti bity a složitost hardwaru, protože zařízení musí mít adresy. Navíc v případě více „master“ zařízení musí být schopný detekovat probíhající přenos. [4]

#### SPI

SPI je synchronní sériová sběrnice. Na rozdíl od $I^2C$ umožňuje připojit pouze více „slave“ zařízení a „master“ může být pouze jeden. Stejně jako $I^2C$ je využíváno ke komunikaci s periferiemi.

Komunikuje ve full duplex módu, což znamená, že „master“ i „slave“ má vlastní vodič do toho druhého. Používá tedy 4 vodiče v normálním zapojení. První pro komunikaci „master“ → „slave“ (Master Out Slave In - MOSI), druhý pro „slave“ → „master“ (Master In Slave Out - MISO), třetí pro hodiny a poslední je „Slave Select“ (SS) nebo také „Chip Select“ (CS) nebo „Chip Enable“ (CE). CS pin slouží k vybrání „slave“ zařízení, které má data přijímat/ posílat.

Výhodný je v tom, že může být až téměř dvakrát rychlejší než $I^2C$ a není limitovaný velikostí zprávy. Data může posílat bez přestání. Protože je to full duplex komunikace, může data posílat a zároveň přijímat. Je také jednoduché ji realizovat, díky CS pinu.

Nevýhodami je, že používá 4 vodiče (lze sloučit MISO a MOSI a udělat tak 3 vodičovou variantu) a více, kdy může teoreticky potřebovat navíc tolik vodičů kolik má „slave“ zařízení (každý „slave“ má vlastní CS). Další nevýhodou je, že na sběrnici může být pouze jeden „master“. V poslední řadě je potenciální problém sběrnice, protože nemá žádný způsob kontroly chyb.[5]

### 3.1.5 Průmyslové sběrnice

Sběrnice hojně využívané v průmyslu. Jejich připojení k Raspberry Pi je realizováno využitím interní sběrnice či USB.

#### RS-232

RS-232 je standard pro sériovou sběrnici zavedený v roce 1960. Podporuje synchronní i asynchronní přenosy. Je ale omezen nízkou přenosovou rychlostí a krátkou vzdáleností. Standard také nepodporuje propojení více zařízení jednou linkou, ale tuto nevýhodu je možné obejít (ačkoliv se tak narušuje tento standard). Dříve sloužil pro připojení do modemů, tiskáren a dalších periferních zařízení. Dnes už se tento standard téměř vytratil z oblasti osobních počítačů a jejich periferií, ale převážně díky jeho jednoduchosti se stále vyskytuje v průmyslu. [31]

Nejjednodušší zapojení RS-232 je pomocí 3 vodičů. První pro vysílání dat, druhý pro přijímání dat a třetí pro uzemnění. Složitěji pak jde RS-232

zapojit pomocí 5 vodičů, které přidávají Ready To Send a Clear To Send signály. [30]

Připojení této sběrnice k Raspberry Pi lze realizovat přes sběrnici UART na pinovém rozhraní. Další možností připojení by byla využití adaptéru RS-232 na USB. USB ale přidává odezvu navíc, což není ideální a proto je připojení přes UART preferované.

#### RS-485

RS-485 je standard pro sériovou komunikaci zavedený v roce 1983. Tento standard má větší přenosové rychlosti a delší vzdálenosti přenosu, než výše popsaný RS-232. Podporuje propojení více zařízení jednou linkou. Standard komunikuje pomocí diferenciálního signálu přenášeného přes kroucenou dvojlinku. To činní standard velmi odolný proti elektrickému rušení a dělá ho vhodným pro průmyslové využití. [32]

RS-485 je možné zapojit pomocí dvou vodičů. Jeden přenáší vysílaný signál a druhý přenáší inverzní vysílaný signál. Často bývá použit ještě třetí vodič, který vede signál uzemnění. Zařízení komunikující pomocí RS-485 používají tří stavovou logiku. Typicky „1“, „0“ a stav vysoké impedance. Díky stavu vysoké impedance je poměrně jednoduché detekovat, kdy je komunikační kanál volný a jiné zařízení může začít komunikovat.

Stejně jako RS-232 i RS-485 se k Raspberry Pi připojuje přes UART sběrnici. I zde je možnost připojení pomocí RS-485 na USB konvertoru. Připojení přes UART je zde také preferováno, ze stejného důvodu jako u RS-232.

#### CAN

CAN je seriová sběrnice podobná předchozím dvoum standrdům, uvedená v roce 1986. Rychlostí a vzdáleností přenosu je mezi RS-232 a RS-485. CAN je pravděpodobně nejrozšířenější typ sběrnice v automobilovém průmyslu.

Pro připojení CAN k Raspberry Pi je nutné použít pomocný integrovaný obvod nebo mikrokontrolér, protože Raspberry Pi nemá zabudovaný ovladač pro CAN.

### 3.1.6 Další periferie

K řídícímu systému může být žádané přidat další funkce. Například modul Real-Time Clock (RTC) pro přesný reálný čas. Nebo třeba GPS

modul pro určování lokace nebo teplotní čidlo, či jiné snímače hodící se pro určité prostředí.

Většinu přídavných modulů a čidel je možné připojit přes interní sběrnice. Pokud nová periferie nejde připojit rovnou přes interní sběrnici, tak je možné, že s velkou pravděpodobností bude existovat pomocný integrovaný obvod, který dovolí periferii komunikovat přes sériovou sběrnici. Například u GPS modulů se ale nabízí možnost využít i USB, které jsou snadno dostupné.

#### RTC

RTC je velice důležitou částí u systémů potřebující přesné časové údaje. Většina počítačů a kontrolérů má nějakým způsobem hodiny zabudované. Tyto hodiny ale nefungují, pokud dojde k výpadku napájení a po znovu spuštění systému mohou udávat špatný čas. RTC moduly mají baterii, díky které udržují správný čas i při výpadku napájení. Z tohoto důvodu jsou RTC velmi užitečné, ne-li přímo nezbytné, obzvlášť u real-time systému.

RTC lze připojit pomocí $I^2C$ nebo SPI sběrnice. Existují RTC moduly pro Raspberry Pi využívající $I^2C$. Příkladem může být třeba Adafruit DS1307 Real-Time Clock Assembled Breakout Board [1]. Nebyl nalezen již hotový produkt, který by byl deskou s integrovaným obvodem využívající SPI, ale byl nalezen integrovaný obvod, který potvrzuje možnost připojení RTC přes SPI [35].

### 3.1.7 Rozšíření pinového rozhraní

IO piny jsou velmi užitečné a pro tuto práci naprosto nezbytné. Alespoň u běžného Raspberry Pi je pravděpodobné, že bude potřeba rozšířit počet pinů. Pro rozšíření pinů je možné použít pomocné integrované obvody komunikující pomocí $I^2C$ sběrnice. Počet pinů, které je možné získat se liší podle použitého čipu, ale je možné získat alespoň 8 nebo 16 pinů [8]. Protože rozšíření je realizováno pomocí $I^2C$, je možné připojit více rozšiřovacích obvodů a tím ještě navýšit počet rozšiřujících pinů.

### 3.1.8 Pomocný procesor

Pomocným procesorem je myšlen mikrokontrolér, který je připojen k Raspberry Pi pomocí některé z interních sběrnic (SPI, $I^2C$, UART). Tento mikrokontrolér může sloužit jako prostředník mezi jednou nebo více periferiemi a Raspberry Pi. V podstatě se tímto způsobem dá rozšířit počet IO pinů.

### 3.1.9 Analogově digitální/ digitálně analogový převodník

Analogově digitální převodník (A/D převodník, ADC) je součástka převádějící analogový signál na digitální. Digitálně analogový převodník (D/A převodník, DAC) převádí digitální signál na analogový. Raspberry Pi nemá zabudovaný žádný ADC ani DAC. Pinové rozhraní komunikuje pouze pomocí digitálního signálu. Velké množství senzorů a čidel generují analogový signál, proto je nutné použít alespoň ADC (pro jednosměrnou komunikaci), pro použití těchto zařízení.

Pokud by byl použit mikrokontrolér pro komunikaci s některými periferiemi, tak analogová zařízení by mohla být připojena právě přes tento mikrokontrolér. Mikrokontroléry často mají ADC a některé mají i DAC.

Jiná možnost připojení ADC nebo DAC je použitím integrovaného obvodu. Tyto obvody lze často připojit přes SPI a $I^2C$ sběrnice.

## 3.2 Software

Podobně jako na hardware, tak i na software jsou kladeny nějaké požadavky. Tyto požadavky jsou kladeny především na operační systém.

### 3.2.1 Real-time systém

Koncept real-time systému je takový, že každá úloha musí být úspěšně splněna do určeného časového limitu. Pokud systém nedokáže najít správné řešení v určeném časovém limitu, tak nastává nepřípustná situace. Real-time systémy se potom dělí na „hard real-time“ nebo „soft real-time“ podle způsobu, kterým se s touto situací vypořádají.

V hard real-time systému je časový limit na získání výsledku absolutní. Pokud systém nezvládne vyprodukovat správný výsledek do časového limitu, tak systém selhává. Tento systém je tím pádem deterministický, neboli dokáže na každý vstup vygenerovat správný výstup. Skutečně hard real-time systémy neexistují ve velkém množství a existující mají převážně jednoduché, ale velmi důležité úkony.

V soft real-time systému je možné časový limit na získání výsledku přesáhnout, ale pouze v malé míře. Systém stále akceptuje výsledky vyprodukované po přesáhnutí časového limitu, ale s menší hodnotou. Čím déle systém přesáhne limit pro získání výsledku, tím menší hodnotu mají dosažená data.

Existuje ještě třetí možnost, jak se může real-time systém chovat při přesáhnutí časového limitu. Místo aby systém selhal, tak jednoduše znehodnotí data získané po přesáhnutí limitu a pokračuje ve své práci. Tento systém se označuje jako „firm real-time“. Nenaplnění časového limitu by se v tomto systému nemělo stávat velmi často.[7]

#### Real-time OS na Raspberry Pi

Real-time operačních systémů je celá řada pro různé platformy. Linuxový kernel prozatím není real-time, ale z části obsahuje implementaci real-time jádra. Linuxový kernel je poměrně jednoduché záplatou (z anglického patch) předělat na real-time kernel. Jeden z nejpopulárnějších a nejlehčích způsobů jak dosáhnout real-time operačního systému z Linux kernelu je nainstalovat PREEMPT_RT patch.

Pomocí PREEMPT_RT vznikne real-time kernel. Existuje ale jiný způsob implementace real-time systému a to je co-kernel. Systém má potom dvě oddělená jádra, kdy přidané jádro slouží k obsluze real-time.[11] Zástupcem tohoto přístupu je Xenomai.

Dále existují zástupci real-time systémů, kteří nejsou založeni na Linuxovém kernelu. Jedním takovýmto příkladem je ChibiOS/RT. Je to kompaktní a rychlý Real-Time Operating System (RTOS), podporující řadu především embedded architektur. Tento systém byl přenesený na Raspberry Pi a zatím byly implementovány tyto funkce: Port (GPIO), Serial, GPT (General-Purpose Timer), $I^2C$, SPI a PWM (Pulse-Width Modulation).

### 3.2.2 Read-only souborový systém

SD karty nejsou navrženy tak, aby běžely nepřetržitě.[33] Postupem času karta degraduje do takové míry, že není možné ji dále použít.

Dalším problémem je výpadek napájení během zápisu na kartu, což by mohlo způsobit poškození některých souborů a následek by mohl být, že se systém z karty již nenačte a může dojít ke ztrátě dat.

Read-only souborový systém je relativně jednoduché softwarové řešení na tyto problémy. Samozřejmě s sebou přináší jiné nevýhody. Například, systém trvale neuchovává logy. Cílem této práce je také implementovat aplikaci, která také bude zaznamenávat data, která nepůjdou zapsat na kartu souborového systému. Naštěstí se nabízí možnost připojení dalšího záznamového média nebo vytvoření nového diskového oddílu, na který bude zapisování povoleno.

# 4 Návrh řešení

Tato práce by měla implementovat příklad řídícího systému, který by mohl být užit ve zdravotnictví. Systém by měl umožnit sbírat data pacientů, přihlásit uživatele RFID čipem nebo přihlašovacími údaji a zobrazení nasbíraných dat. Sbíraná data mají být váhové údaje, která mohou být pořizována nepřetržitě. Zpracování nasbíraných dat už nebude součástí této práce. Práce by tedy měla být dále rozšířena, aby mohla sloužit k reálnému použití.

## 4.1 Hardware

Následuje stručný popis doporučeného hardware pro splnění požadavků řídícího systému této práce.

### 4.1.1 Raspberry Pi

Protože se navrhovaný řídící systém nebude zabývat řízením průmyslového kontrolního systému, tak není potřeba velkého počtu GPIO pinů, proto pro tuto práci stačí běžné Raspberry Pi modely. Práce tedy může být realizována použitím jakéhokoliv Raspberry Pi modelu. Dostupnými modely jsou [27]: Raspberry Pi - Pico, Zero, Zero W, Zero 2 W, 1 Model A+/B+, 3 Model A+/B/B+ a 4 Model B. Vzhledem k tomu, že většina GPIO pinů bude obsazena displejem, tak by bylo vhodné zvolit model, který umožňuje připojit zařízení i pomocí USB. To zužuje výběr na modely Raspberry Pi - 1 Model B+, 3 Model B/B+ a 4 Model B. Z těchto modelů je nejvýkonnější Raspberry Pi 4 Model B, který je pro tuto práci doporučen s ohledem na budoucnost, právě kvůli jeho výkonu, který umožňuje různé rozšíření bez nutnosti výměny Raspberry Pi - základu řídícího systému. Naproti tomuto doporučení může také být zvoleno například Raspberry Pi Zero, které sice neposkytuje více USB rozhraní, ale má menší spotřebu, celkovou fyzickou velikost a je cenově dostupnější.

#### Uložiště

K Raspberry Pi je také nutné mít micro SD kartu. Na této kartě je uložen operační systém, který Raspberry Pi používá a zároveň karta může sloužit k ukládání dat. SD karty ale nejsou spolehlivé, což je popsáno v sekci 3.2.2.

Jsou zde také popsána možná řešení, kterými jsou buď zapisovatelný oddíl na SD kartě nebo externí uložiště.

Výběr SD karty tedy závisí na velikosti a odolnosti. Oficiální dokumentace [29] pro Raspberry Pi doporučuje minimální velikost SD karty 8GB. Protože částí této práce je aplikace, která zaznamenává data získané z čidel, tak je doporučeno použít kartu s velikostí aspoň 32GB. Od karty je také očekáváno, že bude v nepřetržitém provozu, proto je vhodné, aby se jednalo o odolnou kartu. Takovéto karty jsou často vyráběny a používány pro průmysl.

### 4.1.2 Periferie

Doporučené periferie, potřebné k realizaci řídícího systému. Pro tuto práci je třeba zvolit displej, klávesnici, čtečku Radio-frequency identification (RFID) čipů a váhové čidlo.

Při výběru periferií, je potřeba vzít v potaz jejich připojení. Hlavním problémem zapojení je displej. V sekci 3.1.2 jsou popsány možné způsoby připojení displeje. Z těchto způsobů, je zřejmé, že zapojení DPI konfigurace RGB24, je naprosto nevhodné, protože použije všechny GPIO a nebude možné připojit jiné periferie (s vyjímkou USB). Nejvhodnější konfigurací je některý RGB565 Mode. Použitím konfigurace RGB565 zbyde 8 volných GPIO. RGB565 Mode je vybrán, podle funkcí, které poskytují volné GPIO. Tyto funkce mohou být nalezeny v datasheet, který přísluší procesoru zvoleného Raspberry Pi.

**Displej** Hlavní parametry pro volbu displeje značně omezují počet displejů splňujících požadavky. Většina displejů, které požadavky splňují se prodávají hromadně společnostem například pro výrobu navigací, což také komplikuje výběr. Vhodným displejem je třeba Tianma NL6448BC20-35F [22]. Tento displej splňuje požadavky na svítivost, teplotní rozsahy a velikost. Nevýhodou tohoto displeje je cena, která je i při koupi od třetí strany několika násobně vyšší než samotné Raspberry Pi.

**Klávesnice** Ačkoliv existují USB klávesnice, splňující požadavky na prostředí, tedy jsou vlhkosti odolné, prachu odolné a nárazu vzdorné, tak pro tuto práci je doporučena jednolitá membránová, maticová klávesnice nebo keypad. Zástupcem takovéto klávesnice je například Songhe 4 x 4 Matrix Array Membrane Switch Keypad [16].

**RFID čtečka** Pro splnění návrhu je potřeba možnosti přihlašování načtením RFID čipu. V přístupových řídících systémech jsou RFID čtečky často používané pomocí Wiegand rozhraní. Toto rozhraní je způsob přípojení a komunikace s periferiemi jako RFID čtečka, čtečka otisku prstů či čtečka duhovky. Problémem je, že toto rozhraní vyžaduje rychlé odezvy (v rámci mikrosekund) a jak bylo zmíněno na začátku této sekce, tak preferovaný způsob připojení periferií je pomocí sběrnice SPI. Z těchto důvodů je doporučen modul RFID RC522 [28].

**Váhový modul** Poslední žádanou periferií je váhový modul. Jedním z nejpoužívanějších způsobů, jak je váhový modul realizován, je pomocí analogově digitálního převodníku HX711 [14]. Je to velmi přesný 24-bitový převodník, který je často využívaný právě pro váhová čidla. HX711 na vstupu přijímá napěťové signály a jejich hodnoty vystupují v digitální podobě. Připojením libovolného váhového čidla na HX711 pak vznikne velmi přesný váhový modul. Jedná se o levné a již otestované řešení. Pro práci je tedy doporučeno použít například produkt XFW HX711 [13] a váhové čidlo SEN-10245 [34], které spolu vytvoří vážící modul, který lze připojit k mikrokontroléru či řídícímu systému pomocí 2 datových vodičů, napájení a uzemnění.

## 4.2 Software

V této sekci je doporučen systémový software. Protože se zde stále jedná o návrh řešení, tak není možné předem určit, jaký hardware a software bude vybrán. Z tohoto důvod zde není navržena požadovaná řídící aplikace, která bude navržena, až bude znám použitý hardware a systémový software.

### 4.2.1 Operační systém

Prvním nezbytným výběrem software je operační systém. Raspberry Pi je micro počítač a proto potřebuje operační systém, aby fungoval. Výběr operačního systému souvisí s požadavky, které jsou na systém kladeny a účelem užití Raspberry Pi. Požadavky této práce již byly zmíněny, ale pro připomenutí se jedná o real-time a read-only možnosti. Užití Raspberry Pi je jako systém s displejem, takže vhodnou volbou je systém s grafickým rozhraním. Raspberry Pi ovšem nemají dedikované grafické čipy, takže je dobré vybrat málo náročné grafické rozhraní. V potaz je také třeba vzít podporu hardware přímo na Raspberry Pi desce.

S těmito požadavky, je navržen Raspberry Pi OS (RPOS). RPOS je oficiální systém pro Raspberry Pi, poskytován, udržován a vyvíjen přímo od

Raspberry Pi Foundation. Tento systém je založený na Linuxové distribuci Debian a je nabízeno několik variant a verzí. Následující seznam popisuje nabízené verze a varianty v době psaní této práce (červen 2022).

- **Raspberry Pi OS** - verze Linuxového kernelu: 5.15. Debian verze: 11 (bullseye). Tento systém je nabízen ve variantách 32-bit a 64-bit.
- **Raspberry Pi OS (Legacy)** - verze Linuxového kernelu: 5.10. Debian verze: 10 (buster). Tento systém je nabízen pouze ve variantě 32-bit.
- **Raspberry Pi OS Desktop** - verze Linuxového kernelu: 4.19. Debian verze: 10 (buster). Tento systém je nabízen pouze ve variantě 32-bit a je zaměřen pro Desktopové zařízení, proto není relevantní pro tuto práci.

RPOS a RPOS (Legacy) se dále dělí na:

- **with desktop** - systém má grafické rozhraní a základní software, který je očekávaný pro základní užití Raspberry Pi jako běžného počítače.
- **Lite** - systém nemá grafické rozhraní a má pouze minimum základního software.

Pro výběr systému je tedy třeba nejdřív vybrat RPOS nebo RPOS (Legacy) a následně variantu „with Desktop“ nebo „Lite“.

### Úpravy operačního systému

S operačním systémem souvisí jeho probírané úpravy. Protože RPOS není real-time systém, je třeba zajistit real-time odezvy jiným způsobem. Doporučeným způsobem je probíraný PREEMPT_RT patch.

Pro read-only úpravu existuje několik způsobů jak jí dosáhnout na Linuxovém systému. Problémem společným pro všechny způsoby je, že většina programů předpokládá zapisovatelnost systému a může vést ke kompletní či částečné ztrátě funkcí těchto programů. Způsob, kterým se tento problém aspoň částečně řeší, je použití Temporary File System (`tmpfs`). `Tmpfs` umožňuje připojit oddíly, které se jeví jako fyzické, ale ve skutečnosti jsou alokované v operační paměti.

Jednou z výhod RPOS je, že na Raspberry Pi je read-only poměrně žádaná vlastnost, takže přímo do `raspi-config`[1] byla přidána možnost

[1] raspi-config je nastavovací menu, které RPOS poskytuje jednoduchou možnost správy základních vlastností systému.

zapnout takzvaný OverlayFS, který udělá systém read-only. OverlayFS je implementací principu union mount. Union mount je připojení několika adresářů/ oddílů do jednoho. Výsledek se poté jeví jako kombinace všech adresářů. OverlayFS tímto způsobem připojí fyzické uložiště jako read-only a zároveň připojí `tmpfs` oddíl do stejného adresáře. Tímto se uživateli systém jeví jako zapisovatelný, i přes to, že data nejsou uložena persistentně. Z těchto důvodů je tato konfigurace doporučena jako read-only úprava pro tuto práci.

# 5 Použité řešení

V této kapitole jsou popsána vybraná hardwarová a softwarová řešení. Dále je v rámci této kapitoly popsán návrh a implementace aplikace řídícího systému, která byla vytvořená v rámci této práce. Kapitola je zakončena dosaženými výsledky a testy.

## 5.1 Hardware

Vybraný hardware, který byl poskytnut a hardware, který byl reálně použit k realizování práce.

### 5.1.1 Raspberry Pi

K realizaci bylo poskytnuto Raspberry Pi Zero. Tento model nemá možnost wireless komunikace a má pouze limitovaný počet typických vstupů/ výstupů. Zařízení má následující fyzická rozhraní: HDMI, 2 micro USB, slot na micro SD kartu a GPIO header. 1 micro USB ovšem slouží pouze k napájení, reálně je tedy k použití pouze 1 micro USB. Kvůli absenci wireless komunikace se k zařízení přistupuje pomocí Secure Shell Protocol (SSH) přes USB kabel. Tento přístup je ale náročné zprovoznit a navázaná komunikace bývá nestabilní. Z tohoto důvodu je práce nakonec realizována na zařízení Raspberry Pi Zero W, což je téměř totožné s Raspberry Pi Zero s tím rozdílem, že je možné se tímto zařízením připojit k WiFi.

Během realizace práce bylo použito několik karet. Příkladem použité karty je Kingston Canvas Go! Plus microSD Memory Card 32GB [17]. Toto je pouze obyčejná karta, ale pro vývoj a testování je dostačující. Pro dlouhodobý provoz je takováto karta nevhodná, protože je větší šance, že bude dříve poškozena a dojde ke ztrátě dat. Kartu je nutné před použitím připravit. Na kartu musí být předem nainstalován operační systém, aby Raspberry Pi bylo s touto kartou funkční. Kvůli tomu, že se read-only systémový požadavek vylučuje s potřebou řídící aplikace zapisovat data, tak je třeba na kartě připravit nový oddíl, který bude na rozdíl od zbytku karty zapisovatelný. Postup přípravy SD karty je popsán v příloze 6.

### 5.1.2 Periferie

Použitými periferiemi jsou:

- Displej - 24” monitor BenQ s rozlišením 1920x1080. Monitor je připojen pomocí HDMI. Důvodem použití běžného monitoru je nedostatek součástek pro připojení doporučeného displeje nebo podobné alternativy.
- RFID RC522 čtečka, která je přes sběrnici SPI.
- ADC HX711, které je používáno s váhovým čidlem a využívá 2 volné GPIO.
- Membránová maticová klávesnice (resp. keypad) stejná jako doporučená. Klávesnice je připojena pomocí mikrokontroléru STM8S103F3P6, který je stejně jako RFID čtečka připojen přes SPI. „Cena“ připojení klávesnice je tedy pouze jeden GPIO pin, který slouží jako CS.

Ačkoliv je použit normální monitor, tak práce stále byla realizována s předpokladem, že k Raspberry Pi bude připojen displej pomocí DPI. V sekci 4.1.2 je řečeno, že je třeba zvolit vhodnou konfiguraci DPI. Zvolenou konfigurací je RGB565 Mode 3 (viz Obrázek 3.5). Důvodem je, že jako jediná na volných GPIO poskytuje sběrnici. Touto sběrnicí je SPI, proto některé vybrané periferie používají SPI. Dohromady je tedy použitých 16 GPIO na DPI, 5 GPIO sběrnicí SPI (SCK, MISO, MOSI, CS1 - RC522, CS2 - klávesnice), 2 GPIO používá HX711 a zbývá 1 nepoužitý GPIO.

Na následujícím schéma (Obrázek 5.1) je znázorněno připojení použitých periferií na pinovém rozhraní. Ve schématu jsou znázorněny pouze GPIO piny, GROUND a POWER piny znázorněny nejsou. Způsob nastavení Raspberry Pi, aby se použilo toto zapojení, je popsán v příloze 6.


Obrázek 5.1: Použité GPIO

## 5.2 Vybraný systémový software

V této části je vybrán systémový software. Systémovým softwarem jsou myšleny operační systém, real-time úprava a read-only úprava. Software na kterém záleží implementovaná aplikace je popsán v sekci aplikace 5.3.2.

### 5.2.1 Operační systém

Byla zvolena varianta doporučeného RPOS. Zvolená varianta je `RPOS (Legacy) with Desktop`. Ke zvolení `RPOS (Legacy)` vedly dva hlavní důvody. Prvním důvodem je, že RPOS od Debian verze 11 (bullseye), již neposkytuje některé starší 32-bitové knihovny, které byly dostupné v adresáři `/opt/vc`. Druhým důvodem je, že od stejné verze již v systému není výchozí uživatel `pi`. Následkem je to, že není možné se k Raspberry Pi připojit vzdáleně, pokud není vytvořen nový uživatel, což pravděpodobně nelze udělat bez fyzického připojení klávesnice k zařízení.

Varianta `with Desktop` byla zvolena, protože pro variantu `Lite` je třeba nainstalovat desktopové prostředí. Zvoleným desktopovým prostředím by

bylo Lightweight X11 Desktop Environment (LXDE), které by bylo zvoleno kvůli šetrnosti s dostupnými zdroji. Varianta `with Desktop` ale již má desktopové prostředí, které je modifikovanou variantou prostředí LXDE.

### 5.2.2 Real-time úprava systému

Zvoleným řešením je doporučený PREEMPT_RT patch do kernelu. Výhodami je poměrně snadný instalační proces a jeho zdánlivá spolehlivost. Za výhodu také může být považováno to, že pouze upravuje Linuxový kernel, tedy nemění strukturu celého systému.

Tento patch ale nelze aplikovat na již používaný Linuxový kernel. Patch se musí aplikovat na zdrojové soubory kernelu. To znamená, že je třeba zkompilovat kernel s upravenými zdrojovými soubory pro úspěšné aplikování PREEMPT_RT úpravy. Protože architektura Raspberry Pi je jiná od běžných desktopových počítačů, proto je třeba zajistit, aby kernel byl zkompilován pro správnou architekturu. To lze zajistit buď překladem kernelu přímo na Raspberry Pi nebo takzvanou cross-kompilací. Cross-kompilace znamená, že zdrojové soubory jsou přeloženy, aby fungovaly na zařízení s architekturou rozdílnou od architektury, na které jsou soubory překládány. Cross-kompilace kernelu s PREEMPT_RT úpravou je popsána v příloze 6.

Kernel také musí být nasazen. Tato záležitost je popsána v příloze 6.

### 5.2.3 Read-only úprava systému

Pro práci je zvolen doporučený způsob aplikace read-only úpravy. Důvodem je, že ostatní zkoumané metody se pouze lehce odlišují od úpravy, kterou provádí OverlayFS. Navzdory tomu, že se ostatní metody odlišují pouze lehce, tak žádná jiná zkoušená metoda nefungovala stejným způsobem jako OverlayFS.

Byl také nalezen potenciální problém s OverlayFS, který vyžaduje testování. OverlayFS umožňuje uživateli vytvářet a zapisovat soubory, protože se stále jeví jako zapisovatelný. Tyto zápisy jsou ale prováděny do dočasného `tmpfs` uložiště. Problém nastává při opakovaných nebo velkých zápisech, protože OverlayFS nejspíš neomezuje maximální velikost dočasného uložiště. Pokud bude tento problém potvrzen, tak je teoreticky možné vyřešit jej upravením zdrojového kódu OverlayFS skriptu.

#### Příprava před read-only úpravou

Před zapnutím úpravy je třeba zajistit, že jsou provedeny veškeré instalace závislostí a programů, které jsou po systému požadovány.[1] Pro tuto práci je tedy před zapnutím read-only třeba nasadit real-time úpravu a zařídit funkčnost aplikace. To znamená nainstalování jejich závislostí a překlad aplikace samotné, tento postup je popsán v příloze 6.

#### Zapnutí/ vypnutí read-only úpravy

Jak již bylo zmíněno, tak přímo do `raspi-config` byla přidána možnost zapnutí a vypnutí OverlayFS. Toto nastavení se nachází pod menu `Performance options` → `Overlay File System`. Po zapnutí OverlayFS je uživatel vyzván, zda-li chce připojit oddíl `/boot` jako read-only. Zde se zvolí ano.

V případě nutnosti je možné read-only nastavení vypnout použitím stejného menu. Při vypnutí read-only jsou nutné dva restarty zařízení.

## 5.3 Řídící aplikace

Posledním cílem této práce bylo implementovat jednoduchou řídící aplikaci. Tato aplikace je postavena na práci s připojenými periferiemi. Ty mohou být nalezeny v sekci 5.1.2. Klávesnice je použita k navigaci aplikací a zadávání. RFID čtečka zjednodušuje možnost přihlášení. Nejdůležitější periferií je váhový modul, jehož výsledky jsou ukládány a mohou být dále zpracovávány, ale to už nebylo řešeno v rámci této práce.

### 5.3.1 Analýza a výběr vhodných knihoven

Aplikace potřebuje komunikovat s periferiemi a uživatelem. Pro komunikaci s uživatelem je třeba grafického uživatelského rozhraní (GUI). Pro komunikaci s RFID čtečkou a klávesnicí je třeba zajistit SPI rozhraní. Pro komunikaci s HX711 jsou potřeba 2 volné GPIO, které jsou použity pro sběr dat z čidla, ale nepoužívají žádnou sběrnici. Pro ukládání dat je použita MariaDB. Ta funguje jako databázový server a pro ukládání a čtení dat je potřeba rozhraní, které slouží jako klient připojený, k tomuto serveru. Dalšími věcmi, které mohou být užitečné, je logování a konfigurace aplikace pomocí konfiguračního souboru.

[1] V případě zapomenutí některé závislosti, či nastavení, je možné read-only úpravu vypnout.

### Knihovna grafického uživatelského rozhraní

Protože Raspberry Pi Zero (W) je jedním z nejslabších, tak grafická knihovna byla vybírána především s ohledem na výkon. Další méně důležitým bodem byly funkce nabízené knihovnou. Z počátku výběru byly zvažovány knihovny jako Qt či wxWidgets, ale tyto knihovny zcela nevyhovují hlavnímu požadavku. Po více průzkumu se novým zúženým výběrem staly dvě knihovny. Dear ImGui (dále jen ImGui)[23] a Light and Versatile Graphics Library (LVGL)[18]. Tyto dvě knihovny reprezentují dva typy realizací grafických knihoven, těmi jsou „immediate mode“ (ImGui) a „retained mode“ (LVGL). Rozdílem mezi těmito typy je, že immediate mode okamžitě vykresluje všechny elementy na obrazovku, zatímco retained mode uchovává stav grafického rozhraní odděleně od zbytku aplikace a vykresluje vše v jeden moment. Následujícím rozdílem mezi těmito knihovnami je, že ImGui nabízí řadu různých backendů, aby se rozhraní dalo použít na více platformách. To ale znamená, že je třeba mít systém s Desktopovým prostředím (mezi známé Linuxové zástupce patří: KDE, Gnome..). LVGL využívá Linuxový frame buffer. Frame buffer je obraz reprezentován bitmapou v operační paměti a není tedy potřeba Desktopového prostředí.

Vybranou knihovnou se stalo ImGui. LVGL je více zaměřené na embedded zařízení a je tedy šetrnější se zdroji, proto vypadá jako lepší volba než ImGui. Hlavním důvodem proč nakonec bylo zvoleno ImGui je ten, že ImGui nabízí širokou řadu funkcí a nebylo třeba upravovat funkčnosti knihovny.

### Knihovny komunikace s čidly

Klávesnice a RFID čtečka jsou připojeny sběrnicí SPI. Komunikaci s SPI umí obstarávat přímo RPOS. Pokud je SPI zapnuto, vzniká device rozhraní jako například `/dev/spidev0.0`. Přes toto rozhraní je možné komunikovat s připojeným zařízením pomocí `ioctl`. Díky tomuto je možné SPI zařízení používat i bez použití dalších knihoven. V této práci je pro zjednodušení použita knihovna `spidev-lib` [21]. Tato knihovna vnitřně využívá právě `ioctl`.

RFID čtečka sice komunikuje přes SPI, ale proces načítání karet je poměrně složitý. Pro Raspberry Pi není mnoho knihoven pro použitý typ čtečky (RC522). Žádná z existujích knihoven nefungovala „out of the box“. Knihovna RFID čtečky nakonec vychází z knihovny [24], která ale není použita celá. Z této knihovny jsou použité pouze části kódu, které byly třeba k čtení karet. Použitý kód z této knihovny byl upraven, aby pro komunikaci používal výše zmíněnou `spidev-lib`.

Pro ADC HX711 je použita knihovna `Raspberry Pi HX711 C++ Library`

[9]. Tato knihovna zařizuje komunikaci s čidlem a přidává řadu funkcí, pro práci s daty z čidla. Velké množství přidaných funkcí je také důvod, proč knihovna byla vybrána, protože značně ulehčuje práci s naměřenými daty. Pro komunikaci s čidlem knihovna závisí na knihovně `lgpio`[15], kterou je třeba nainstalovat. V knihovně HX711 se nachází skript, který závislosti (tedy lgpio) instaluje.

#### Další knihovny

**spdlog** Je rychlá a snadno použitelná knihovna pro logování. Knihovnu je možné nainstalovat nebo použít jako header-only. Při použití jako header-only může značně zvyšovat čas kompilace. Pro většinu Linuxových systémů je dostupná pomocí package managerů. Zdroje a dokumentace jsou dostupné z platformy GitHub [10].

**libconfini** Jedná se o parser konfiguračních souborů v jazyce C. Knihovna podporuje typy boolean, čísel, textu a polí. Dále podporuje sekce, podsekce, zápisy na více řádek, komentáře a další. Knihovna neumí zapisovat do konfiguračních souborů, pouze je číst. Knihovnu je možné získat z platformy GitHub [19].

**MariaDB Connector/C** Knihovna sloužící jako klientské rozhraní mezi MariaDB databází a jazykem C. Pro většinu Linuxových systémů je dostupná pomocí package managerů. Zdroje a dokumentace jsou dostupné z oficiální stránky [20].

### 5.3.2 Závislosti

Aplikace závisí na knihovnách v předešlé sekci (sekce 5.3.1). Některé tyto knihovny mají vlastní závislosti. ImGui závisí na knihovnách použitých backendů. Pro tuto aplikaci jsou využity backendy `OpenGL` a `SDL2`. Další závislostí, která již byla zmíněna je MariaDB, která je používána pro ukládání dat z aplikace. Pro úspěšný překlad a běh aplikace je potřeba mít nainstalované všechny tyto závislosti. Většina závislostí může být nainstalována RPOS package managerem, ale detailnější popis instalace závislostí je v příloze 6.

### 5.3.3 Implementace řídící aplikace

Důležitou věcí, která by měla být zajištěna před prvním spuštěním je připravený databázový server. Připravený databázový server je takový, který

má založenou databázi a ve které jsou tabulky používané aplikací. Aplikace používá pouze 2 tabulky, kterými jsou `cs_users` a `cs_measurements`. Zbylé parametry databáze, jako její jméno, či používaný uživatel mohou být vybrány libovolně, ale je zapotřebí je nastavit v konfiguračním souboru aplikace. Zakládací SQL skripty mohou být nalezeny v podadresáři zdrojových souborů `sql`. Jsou zde skripty pro vytvoření databáze `control_system` s uživatelem `pi`, založení tabulek a vytvoření testovacích dat.

Konfigurační soubor aplikace je druhou věcí, kterou je třeba zajistit. Načtení tohoto souboru je první věc, kterou aplikace provádí při spuštění. V tomto souboru je možné nastavit věci jako konfigurace SPI, připojení k databázi, používaný font, používaný jazyk a další. Po načtení konfiguračního souboru se aplikace pokouší o načtení sekundárního konfiguračního souboru. Tento soubor už není povinný, ale je použit pro hodnoty, které může měnit aplikace. Hlavní konfigurační soubor je aplikací pouze čten a upraven může být pouze manuálně. Umístění sekundárního konfig. souboru je definováno v hlavním konfig. souboru a je očekáváno, že sekundární konfig. soubor je umístěn na zapisovatelném oddílu.

Další důležitou činností je načtení takzvaného slovníku. Slovníkem je myšlen soubor, který obsahuje řádky ve formátu **KEY=Hodnota**, a v aplikaci jsou potom použity hodnoty těchto klíčů. Slovníkový soubor se musí jmenovat jako `dictionary_lang`, kde `lang` je nahrazeno za zkratku jazyka. Příkladem tohoto je třeba `dictionary_en`, což je slovník anglického jazyka poskytnutý s touto prací. Používaný jazyk a umístění slovníků jsou definovány v hlavním konfig. souboru. Pokud se nepovede slovník načíst, tak aplikace skončí, ale pokud ve slovníku pouze chybí některý klíč, tak je použit tento klíč místo hodnoty.

Po načtení slovníku je provedena inicializace grafického kontextu. To je následováno načtením fontů, respektive jednoho fontu, který je definovaný v konfig. souboru, ale je načten v několika velikostech. V dalším kroku se inicializují čidla, jejichž čtení je puštěno v novém vlákně (s vyjímkou HX711) a nakonec se naváže spojení s databází.

### Koncept aplikace

Aplikace je implementována v jazyce C++ a je inspirována architekturou Model-View-Controller (MVC). Ve zdrojovém adresáři se nachází podadresáře:

- **gui** - V tomto adresáři se nachází definice obrazovek GUI a logika s nimi.

- **logic** - Soubory v tomto adresáři poskytují logiku k obsluze použitých periferií a řízení aplikace.
- **model** - Obsahuje soubory, které poskytují struktury pro funkci aplikace a logiku nad nimi.
- **sql** - Obsahuje SQL skripty pro přípravu databáze, tabulek a testovacích dat.

Na Obrázku 5.2 je znázorněn koncept aplikace. Po výše popsaných přípravách (sekci 5.3.3) aplikace dosáhla hlavního řídícího cyklu, který je vidět na obrázku. Tento cyklus obstarává většinu logiky aplikace a také je řídícím cyklem GUI. Na začátku tohoto cyklu jsou odbavovány eventy. Jsou odbavovány eventy 3 typů. Nejdříve jsou eventy RFID čtečky. Další v pořadí jsou eventy, které jsou vytvořené periferní klávesnicí, připojenou přes SPI (dále jen SPI klávesnice). V poslední řadě jsou odbaveny eventy knihovny `SDL2`, které mohou být navíc předány GUI knihovně, která na základě těchto eventů manipuluje GUI. Po odbavení eventů následuje začátek nového snímku (frame) GUI. Zde jsou volány vykreslovací funkce jednotlivých prvků. Frame je poté ukončen a vykreslen na obrazovku a začíná nová iterace.

Tento řídící cyklus běží v hlavním vlákně aplikace. Z obrázku je ale možné vidět, že paralelně běží vlákna SPI klávesnice a RFID čtečky a uživatelskou akcí může být zapnuto i vlákno pro měření HX711, které navíc zapíná vlákno pro sledování probíhajícího měření.

Vlákno SPI klávesnice v určitých časových intervalech (konstanta `READ_INTERVAL`) čte z SPI. Při psaní této práce má konstanta hodnotu 20000, což je interval čtení SPI v mikrosekundách. Každých zhruba 20ms je tedy přečtena hodnota a pokud není nulová, tak je vytvořen event struktury `kb_event`. Tento event je vložen do klávesnicové `event_queue`. Hlavní vlákno aplikace z této queue eventy vyjímá a obsluhuje je.

Vlákno RFID čtečky funguje stejným principem jako SPI klávesnice. Časový interval (konstanta `RFID_READ_INTERVAL`) ve kterém je SPI čteno je zde 100000 (tedy 100ms). Stejně jako u SPI klávesnice je při úspěšném načtení karty vytvořena event struktura `rfid_event`, která je vložena do `event_queue`[2]. Stejně jako u SPI klávesnice, hlavní vlákno aplikace vyjímá eventy z této queue a vykonává jejich obsluhu.

Pokud uživatel vyžádá akci měření nebo je spuštěn test používající HX711, tak je měření zapnuto v novém vlákně. Protože použitá knihovna je blokující, tak si toto vlákno navíc vytvoří ještě jedno vlákno, které slouží jako pozorovatel měřených hodnot. Vlákno pozorující měření počítá dočasné hod-

[2]Tato `event_queue` se nachází v jiném `namespace`, proto může mít stejný název.

noty a nastavuje je do proměnných, které jsou zobrazeny v GUI. Poté co skončí měření, tak je zastaveno vlákno pozorovatele. Celkový výsledek měření je uložen do instance třídy `measurement` a do databáze. V případě testování jsou výsledky měření uloženy pouze do proměnných, které přísluší prováděnému testování. Do databáze jsou ukládány naměřené hodnoty v binární podobě, průměr a medián ze všech hodnot, průměr a medián hodnot, které jsou normalizované po určitém počtu vzorků (konstanta `X_SAMPLES`, hodnota 5) a také metadata o měření jako například délka měření, začátek a konec měření a další. Naměřené hodnoty jsou datového typu `double` a nejdelší měření co aplikace umožňuje je 5000 vzorků (konstanty `HX_CONT_SAMPLES` - použita při nepřetržitém měření a `S2_MAX_SAMPLES_IN` - maximální limit měření určitého množství vzorků). Za předpokladu, že `double` je 8B, tak i s metadaty lze předpokládat, že velikost jednoho nejdelšího měření bude mít maximálně 50kB. V případě nepřetržitého měření, je takovéto měření uloženo jako jednou částí z celého měření a těchto částí může být neomezené množství. Databáze používá pro naměřená data typ `MEDIUM_BLOB`, který může mít velikost až přibližně 16,6MB. Aplikace momentálně neukládá takto velká měření, ale pokud by byla potřeba ukládat objemnější měření, tak je možné použít `LONGBLOB`.

**Funkce hlavního řídícího cyklu** Na začátku této sekce bylo popsáno řízení aplikace cyklem v hlavním vlákně. Je zde řečeno, že řídící cyklus nejdříve obsluhuje eventy. Tyto obsluhy jsou implementovány v souboru `logic/event_handler.cpp`.

Obsluha RFID eventu je `handle_rfid_event(rfid_event* event)`. Aplikace umožňuje přihlásit uživatele a pokud má přihlášený uživatel roli `zaměstnanec` nebo `administrátor`, tak umožňuje přihlásit takzvaného `poduživatele`. Poduživatel je označení uživatele s rolí `pacient` a pro tohoto uživatele jsou vykonávána měření. Vykonaná měření patří poduživateli, ale jsou ukládána tak, že uživatel byl ten kdo měření provedl. Při obsluze RFID eventu je tedy zkontrolováno, zda-li se může přihlásit uživatel. Pokud je uživatel již přihlášen, tak je zkontrolováno, zda-li uživatel může mít poduživatele a zda-li nemá již přihlášeného poduživatele. Pokud nelze přihlásit uživatele ani poduživatele, tak je event ignorován, jinak je poslán dotaz do databáze, zda-li je RFID čip registrovaný a pokud je RFID čip nalezen, tak je (pod)uživatel přihlášen.

Obsluhou eventu SPI klávesnice je `handle_raw_event(kb_event* raw_event)`. Klávesnice funguje ve 2 režimech, kterými jsou `NAVIGATION` a `INPUT`, který se dále dělí podle typu vstupu na `INT`, `FLOAT`, `TEXT`. Podle toho v jakém režimu klávesnice je, je event obsloužen. Existuje ale jedna vyjímka

a tou je klávesa s hodnotou **0x2000**, která má na klávesnici znak # a je rezervovaná pro funkci `RETURN`. Tato funkce slouží k aktivaci prvků GUI a tím i přepínání mezi zmíněnými režimy. Klávesa je rezervovaná pro všechny možné typy vstupů klávesnice. Pokud je tedy obsluhovaný event jinou klávesou (nebo kombinací stisku více kláves), tak je volána jeho příslušná obsluha, která typicky vytváří `SDL2` eventy. Obsluha eventů obou režimů je podobná. Pro obě obsluhy je třeba zjistit odpovídající `SDL2` klávesu, aby bylo možné nasimulovat event stisknutí klávesy na běžné klávesnici. K tomuto je využita funkce `determine_scancode(uint16_t keycode, uint8_t cycle)`, která podle parametrů hledá klávesu v mapě příslušné obsluhy. Jsou definovány následující mapy kláves: `navigation_keymap`, `int_in_keymap`, `float_in_keymap`, `text_in_keymap`. Přidání či změna klávesy může být provedena pouze změnou těchto map. Po získání odpovídající `SDL2` klávesy jsou poté vytvořeny příslušné `SDL2` eventy, aby byl nasimulován stisk klávesy. Například pro navigaci jsou vytvořeny `SDL2` eventy zmáčknutí a puštění klávesnicové šipky. Pro aktivaci tlačítka ale musí navíc být vytvořen event `SDL2` textového vstupu s klávesou `SPACEBAR`.

Aplikace také umožňuje reagovat na eventy knihovny `SDL2` obslužnou funkcí `handle_sdl_event(const SDL_Event* event)`. Tato obsluha je použita hlavně při používání běžné klávesnice, aby bylo možné aspoň částečně nasimulovat ovládání aplikace pomocí SPI klávesnice. Jedinou důležitou věcí v této obsluze je „workaround“ pro resetování zvýraznění GUI prvku. ImGui si pamatuje pozici zvýraznění a při změně obrazovky je zvýrazněn GUI prvek s pozicí nejblíže předchozímu zvýraznění. Tento workaround funguje tak, že po změně obrazovky je místo klávesy šipky nahoru, či dolu vytvořen nový event s klávesou `PAGE_UP`, což způsobí zvýraznění prvního GUI prvku na obrazovce. Kromě tohoto „workaround“ je zde například obsloužen event klávesy `Right_Alt`, který simuluje # na SPI klávesnici.

Hlavní cyklus dostal do kroku renderování GUI. To se skládá z obrazovek, které jsou definované v souboru `screen_definitions.cpp` pomocí funkcí například `define_screen_1(std::vector<std::unique_ptr <gui_element»& elems)`. Jednotlivé GUI prvky jsou zjednodušeny pomocí vlastních tříd, které jsou implementovány v adresáři **gui**. Tyto třídy obalují funkce ImGui knihovny, aby byla snažší definice obrazovek. V `screen_definitions.cpp` se nachází i obslužné akce pro prvky, které obslužnou akci potřebují. Vytvořené prvky jsou vkládány do standardní struktury `std::vector`, který je předán parametrem při definici obrazovky. Všechny prvky, které jsou takto přidané, musí dědit od třídy `gui_element`, která vyžaduje překrytí funkce `void render_element()`. Toto je potom využito ve správci obrazovek - třída `screen_manager`. Tato třída uchovává

definované obrazovky, momentálně zvolenou obrazovku a poskytuje funkce k vykreslení a manipulaci obrazovek. Třída používá strukturu `std::unordered_map` k držení prvků obrazovky. Klíčem je číslo obrazovky. Při vytváření této třídy je tato struktura inicializována pro předurčený počet obrazovek, které budou pro aplikaci dostupné (konstanty `DEBUG_SCREENS` s hodnotou 5 a `REAL_SCREENS` s hodnotou 6). Pro přidání nové obrazovky je tedy třeba provést následující kroky:

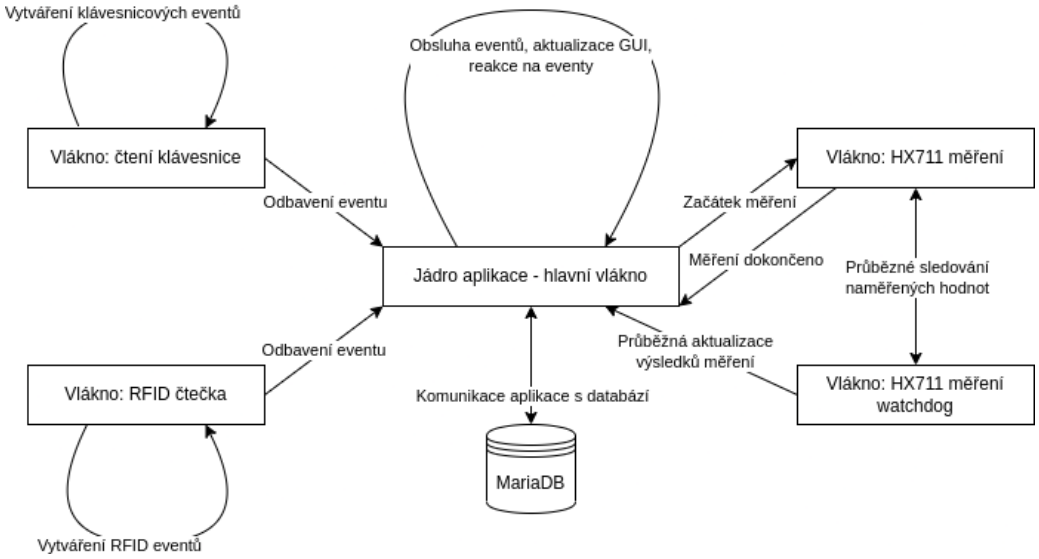
1. Ve `screen_definitions.cpp` vytvořit novou funkci definující prvky obrazovky podle příkladu `define_screen_1(std::vector<std::unique_ptr<gui_element>>& elems)`.

2. Změnit konstantu `REAL_SCREENS` případně `DEBUG_SCREENS` na nový počet používaných obrazovek.

3. Přidat obrazovku do konstruktoru `screen_manager` podle příkladu `define_screen_1(available_screens.at(LOGIN_SCREEN))`.

4. Pokud je po obrazovce požadováno, aby bylo možné měnit její prvky za běhu, tak je také třeba přidat ve funkci `refresh_screen_elements(uint8_t screen)` větev přepínače, podle příkladu `case 1: define_screen_1(*elems); break;`.

Tímto hlavní cyklus došel ke konci, kde jsou pouze volány knihovní funkce k vykreslení GUI na obrazovku. Jednou z významných funkcí, která ale nebyla probrána je třída `db_driver`. Tato třída poskytuje funkce pro získání a ukládání dat do databáze. Tyto funkce jsou často volány v reakci na některý z eventů.




Obrázek 5.2: Koncept řídící aplikace

### 5.3.4 Možné rozšíření aplikace

Aplikace je poměrně prostá a nabízí pouze limitovaný počet funkcí. Některé věci také nejsou dotaženy do konce, nebo by mohlo být vyžadováno trochu jiné chování. Jedním vhodným rozšířením by byly grafy k výsledkům měření. Dalším by byl export dat z databáze do souboru, aby bylo možné přenést výsledky vybraných měření do jiných aplikací nebo na jiný systém. Celkový vzhled GUI by také mohl být vylepšen. Také by mohla být vylepšena lokalizace GUI, protože momentálně není podporováno UTF-8, ačkoliv ImGui podporu umožňuje. Lokalizace dále může být vylepšena načítáním jazyků za běhu aplikace a v případě chybějícího klíče by byl použit výchozí jazyk, který by měl být vždy načten a obsahovat všechny použité klíče.

## 5.4 Testování a výsledky

Poslední částí práce je provést testy a popsat některé dosažené výsledky.

### 5.4.1 Testování real-time odezvy systému

Pro testování real-time systémů existuje sada RT-testů [36]. Odezvy byly testovány použitím „cyklického“ testu (cyclictest) z této sady. Testování bylo provedeno na RPOS bez vytížení procesoru a s ním. Následně bylo stejné testování zopakováno na RPOS s PREEMPT_RT úpravou kernelu. Byla

snaha provést testy i pro Xenomai (zmíněno v sekci 3.2.1), ale zde se objevil problém s podporou pro používané Raspberry Pi. I přes to byl pokus tyto testy provést na Raspberry Pi 4B (4B je zkratkou pro označení 4 Modle B), který by měl být podporován pro Xenomai. Pokus se zřejmě nepovedl, protože výsledky pro Xenomai jsou ve většině případech horší, než neupravený RPOS kernel. Není zřejmé, zda-li jsou tyto výsledky zaviněny nesprávným nasazením Xenomai nebo tím, že Xenomai nefunguje spolehlivě na Raspberry Pi zařízeních. V následujících tabulkách jsou nejhorší (Tabulka 5.1) a průměrné (Tabulka 5.2) odezvy v mikrosekundách [µs]. RPOS zde znamená neupravený systém. „load“ znamená, že test byl proveden s vytíženým procesorem.

| | **Core 1** | **Core 2** | **Core 3** | **Core 4** |
|---|---|---|---|---|
| **RPOS**[µs] | 273 | 300 | 411 | 136 |
| **RPOS load**[µs] | 235 | 103 | 824 | 256 |
| **PREEMPT_RT**[µs] | 69 | 63 | 55 | 72 |
| **PREEMPT_RT load**[µs] | 32 | 37 | 37 | 34 |
| **Xenomai**[µs] | 266 | 170 | 729 | 1301 |
| **Xenomai load**[µs] | 234 | 146 | 4864 | 732 |

Tabulka 5.1: Tabulka nejhorších odezev cyklických (RT) testů z Raspberry Pi 4B

| | **Core 1** | **Core 2** | **Core 3** | **Core 4** |
|---|---|---|---|---|
| **RPOS**[µs] | 21 | 21 | 21 | 21 |
| **RPOS load**[µs] | 7 | 7 | 7 | 7 |
| **PREEMPT_RT**[µs] | 22 | 22 | 22 | 22 |
| **PREEMPT_RT load**[µs] | 7 | 7 | 7 | 7 |
| **Xenomai**[µs] | 67 | 63 | 58 | 58 |
| **Xenomai load**[µs] | 66 | 61 | 50 | 51 |

Tabulka 5.2: Tabulka průměrných odezev cyklických (RT) testů z Raspberry Pi 4B

### 5.4.2 Testování pomocí aplikace

Následující testy jsou vykonané využitím implementované aplikace.

### Počet vzorků z HX711

HX711 snímá data rychlostí 80Hz. Tedy z čidla by mělo být přečteno zhruba 80 vzorků za vteřinu. Tento test je implementován v aplikaci dvěma způsoby a je přístupný z administrátorské obrazovky. V této implementaci testu je měřena doba získání určitého počtu vzorků (konstanta `HX_TEST_SAMP_COUNT`). Měření je provedeno celkově desetkrát a poslední čtyři měření mají navíc přičtenou konstantu `HX_TEST_SAMP_COUNT` k předchozímu počtu prvků. V Tabulce 5.3 jsou výsledky tohoto měření s vytížením procesoru a bez něj. Druhá implementace je obdobou předchozí, ale funguje opačně. Tedy je měřeno kolik vzorků je přečteno za určitou dobu (konstanta `HX_TEST_TIME_COUNT`). Stejně jako první implementace je toto měření provedeno desetkrát a poslední čtyři iterace mají zvyšující se dobu měření. Výsledky tohoto testu jsou v Tabulce 5.4 a také byly provedeny s vytížením procesoru a bez něj.

Z Tabulky 5.3 vychází, že 80 vzorků trvá průměrně **1456.1 ms bez vytížení procesoru** a **1305.3 ms s vytíženým procesorem**. Z Tabulky 5.4 vychází, že 80 vzorků trvá průměrně **1379.3 ms bez vytížení procesoru** a **1518.7 ms s vytíženým procesorem**. Jako celkový výsledek testování je možné usoudit, že z HX711 není snímáno 80 vzorků za vteřinu. Podle dosažených výsledků je snímáno něco mezi 50-60 vzorky. Zajímavým úkazem je to, že při zatížení procesoru bylo ve většině iterací nasbíráno více vzorků, než při měření bez zátěže. Není zřejmý důvod tohoto úkazu, ale na základě testů jejichž výsledky jsou v Tabulkách 5.1 a 5.2, se lze domnívat, že při vytížení procesoru má systém kratší odezvy, ačkoliv tato domněnka zní neintuitivně.

| **Počet vzorků** | **Čas [ms]** | **Čas s vytížením CPU [ms]** |
|---|---|---|
| 160 | 3045 | 2595 |
| 160 | 3075 | 2778 |
| 160 | 2889 | 2822 |
| 160 | 2507 | 2723 |
| 160 | 2936 | 2242 |
| 160 | 2831 | 2453 |
| 320 | 5470 | 5676 |
| 480 | 9106 | 7829 |
| 640 | 11683 | 9919 |
| 800 | 14702 | 13175 |

Tabulka 5.3: Tabulka časů měření pro určitý počet vzorků

| Čas [s] | Počet vzorků | Počet vzorků vytížením CPU |
|---|---|---|
| 2 | 112 | 113 |
| 2 | 104 | 137 |
| 2 | 106 | 115 |
| 2 | 92 | 133 |
| 2 | 98 | 118 |
| 2 | 102 | 123 |
| 4 | 223 | 226 |
| 6 | 340 | 375 |
| 8 | 473 | 493 |
| 10 | 557 | 597 |

Tabulka 5.4: Tabulka počtů naměřených vzorků za určitý čas

### Prodleva reakcí klávesnice

Klávesa je čtena po určitém časovém intervalu (konstanta `READ_INTERVAL`), který je v době psaní této práce stanoven na 20000 µs. Interval je implementován voláním `usleep()`. Tento test slouží k zjištění prodlevy mezi stisky (čtení) klávesy. Test je přístupný z administrátorské obrazovky. V aplikaci je nastavena proměnná `bool kb_testing`. Když je tato proměnná `true`, tak v obsluze SPI komunikace dojde ke změření času od předchozí obsluhy a je vytvořen klávesnicový event, který má nastavený bit (`KB_FLAGS_TEST`). Tato implementace měří pouze prodlevy čtení SPI. Díky vytvoření eventu s bitem `KB_FLAGS_TEST`, je ale možné test rozšířit, aby testoval i prodlevu mezi vytvořením a obsloužením eventu. Změřená odezva je vypsána do logů na obrazovce na úrovni **info** a každých 200 vzorků (konstanta `RESPONSES_AVG_CNT`) je na úrovni **warning** vypsán průměr, minimum a maximum těchto vzorků. V tabulce 5.5 jsou minimální, maximální a průměrné odezvy. Tyto odezvy byly naměřeny pro vlákno s normální prioritou a nejvyšší (RT) prioritou. Navíc byla tato měření provedena i s vytíženým procesorem (load).

Z tabulky je zřejmé, že výsledky s RT prioritou mají nižší odezvy klávesnice. To je nejvíce význačné na maximálních odezvách, ale i na průměrných odezvách je vidět poměrný rozdíl. Pro uživatele jsou tyto výsledky naprosto nepostřehnutelné, ale v porovnání s odezvami cyklických testů z Tabulek 5.1 a 5.2 jsou odezvy několikanásobně-krát horší, než odezvy očekávané na základě těchto tabulek.

| | **Minimální** | **Maximální** | **Průměrná** |
|---|---|---|---|
| **Odezvy**[µs] | 20525 | 39285 | 21886 |
| **Odezvy load**[µs] | 20530 | 53376 | 22942 |
| **Odezvy RT**[µs] | 20332 | 22595 | 20828 |
| **Odezvy RT load**[µs] | 20358 | 22693 | 20831 |

Tabulka 5.5: Výsledky testování odezev klávesnice

# 6 Závěr

Tato bakalářská práce prozkoumala možnosti, které Raspberry Pi nabízí jako řídící systém. Z tohoto průzkumu vyšlo, že Raspberry Pi lze použít jako základ řídícího systému, ale má své výhody i nevýhody, které záleží na přesných požadavcích řídícího systému. Hlavní obecnou výhodou je velká univerzálnost Raspberry Pi, díky které lze Raspberry Pi použít pro téměř jakýkoliv účel. Tato výhoda ale souvisí s hlavní obecnou nevýhodou. Tou je složitost implementace vybraného řešení. Ačkoliv existující řídící systémy nemají tak velkou univerzálnost, tak je většinou mnohem snažší a levnější je nastavit a uvést do provozu. Kdyby Raspberry Pi mělo takovéto systémy nahradit, tak by bylo třeba nejdříve implementovat celé řešení řídícího systému, což je časově náročný proces a to by se projevilo i na celkové ceně takového systému.

S provedeným průzkumem následovala další část práce a to návrh jednoduchého vážícího řídícího systému. Požadavkem na tento systém bylo vybrat displej, RFID čtečku, klávesnici, vážící modul a vhodný software, pro zajištění real-time odezev zvolených periferií. Návrh tohoto systému byl dokončen implementací aplikace, využívající vybraná řešení. Vytvořená aplikace ukládá naměřená data a počítá z nich průměr a medián, ale data dále nezpracovává. Aplikace také umožňuje uživateli zobrazit vypočítaný průměr, medián a další data, jako třeba kdy měření proběhlo a jak dlouho trvalo. Dále poskytuje funkce pro své testování, které mohou sloužit jako testování řídícího systému. Pro zaručení real-time odezev byl použit PREEMPT_RT patch do Linuxového jádra.

Pro reálné použití by bylo vhodné aplikaci dále rozšířit. Potenciálně nejužitečnějšími rozšířeními by byla možnost exportu naměřených dat, aby bylo možné tato data přenést na jiný systém pro další zpracování, nebo přidání grafů, protože samotný průměr a medián z více než několika hodinového měření, není dostatečně vypovídající.

# Přehled zkratek

- **GPS** - Global Positioning System
- **SD karta** - Secure Digital karta
- **IO** - Input/ Output
- **GPIO** - General-Purpose Input/ Output
- **SPI** - Serial Peripheral Interface
- **HDMI** - High-Definition Multimedia Interface
- **DPI** - Parallel Display Interface
- **RGB** - Red Green Blue
- **UART** - Universal Asynchonous Reciever/ Transmitter
- **$I^2C$** - Inter-Integrated Circuit
- **ACK** - Acknowledged
- **NACK** - Not Acknowledged
- **MISO** - Master In Slave Out
- **MOSI** - Master Out Slave In
- **SS** - Slave Select
- **CS** - Chip Select
- **CE** - Chip Enable
- **USB** - Universal Serial Bus
- **RS** - Recommended Standard
- **CAN** - Controller Area Network
- **RTC** - Real-Time Clock
- **ADC** - Analog-to-Digital Converter
- **DAC** - Digital-to-Analog Converter

- **RTOS** - Real-Time Operating System
- **GPT** - General-Purpose Timer
- **PWM** - Pulse-Width Modulation
- **RFID** - Radio-frequency identification
- **RPOS** - Raspberry Pi OS
- **tmpfs** - Temporary File System
- **GUI** - Graphical User Interface
- **SSH** - Secure Shell Protocol

# Literatura


[1] *Adafruit DS1307 Real Time Clock Assembled Breakout Board* [online]. https://www.adafruit.com, 2020. [cit. 2022/6/16]. Dostupné z: https://www.adafruit.com/product/3296.

[2] Braun, A. *What Is a Nit of Screen Brightness and How Many Do You Need?* [online]. https://www.maketecheasier.com, 2020. [cit. 2022/5/27]. Dostupné z: https://www.maketecheasier.com/what-is-nit-of-screen-brightness/.

[3] Buckley, I. *Raspberry Eyes up the Industrial Automation Sector* [online]. 2020. [cit. 2022/6/6]. Dostupné z: https://www.makeuseof.com/raspberry-pi-industrial-support/.

[4] Campbell, S. *Basics of the I2C communication protocol* [online]. https://www.circuitbasics.com, 2016. [cit. 2022/5/28]. Dostupné z: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/.

[5] Campbell, S. *Basics of the SPI communication protocol* [online]. https://www.circuitbasics.com, 2016. [cit. 2022/5/28]. Dostupné z: https://www.circuitbasics.com/basics-of-the-spi-communication-protocol.

[6] *Control system* [online]. https://en.wikipedia.org, 2020. [cit. 2022/5/27]. Dostupné z: https://en.wikipedia.org/wiki/Control_system.

[7] Delaney, S. F. *RealPi - A Real Time Operating System on the Raspberry Pi* [online]. https://www.cse.unr.edu, 2018. [cit. 2022/6/6]. Dostupné z: https://www.cse.unr.edu/~fredh/papers/thesis/075-delaney/thesis.pdf.

[8] DiCola, T. *Using MCP23008 & MCP23017 with CircuitPython* [online]. https://learn.adafruit.com/, 2020. [cit. 2022/6/16]. Dostupné z: https://learn.adafruit.com/using-mcp23008-mcp23017-with-circuitpython/overview.

[9] endail, D. R. *Raspberry Pi HX711 C++ Library* [online]. 2020. [cit. 2022/6/16]. Dostupné z: https://github.com/endail/hx711.

[10] gabime, G. M. *spdlog* [online]. 2022. [cit. 2022/6/21]. Dostupné z: https://github.com/gabime/spdlog.

[11] Gonzalez, D. R. *Design and Evaluation of a Real-Time Sensor Monitor System on Raspberry Pi using Xenomai* [online]. https://kth.diva-portal.org, 2019. [cit. 2022/6/6]. Dostupné z: https://kth.diva-portal.org/smash/get/diva2:1381368/FULLTEXT01.pdf.

[12] *Compute Module Attaching and Enabling Peripherals Guide* [online]. Raspberry Pi, 2020. [cit. 2022/6/8]. Dostupné z: https://www.raspberrypi.org/documentation/hardware/computemodule/cm-peri-sw-guide.md.

[13] *Goose electronic HX711 module weighing sensor 24 AD module pressure sensor /SCM,DIY preferred for Arduino XFW HX711* [online]. https://www.pcb-hero.com, 2020. [cit. 2022/6/16]. Dostupné z: https://www.pcb-hero.com/products/423572.

[14] *HX711 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales* [online]. https://microcontrollerslab.com/, 2022. [cit. 2022/6/21]. Dostupné z: https://microcontrollerslab.com/hx711-adc-weigh-scales/.

[15] joan2937. *Knihovna lgpio* [online]. 2021. [cit. 2022/6/16]. Dostupné z: http://abyz.me.uk/lg/py_lgpio.html.

[16] *Keypad - Songhe 4 x 4 Matrix Array Membrane Switch Keypad* [online]. https://www.amazon.com, 2020. [cit. 2022/6/16]. Dostupné z: https://www.amazon.com/Matrix-Membrane-Switch-Keyboard-Arduino/dp/B07THCLGCZ.

[17] Kingston. *Canvas Go! Plus microSD Memory Card* [online]. https://www.kingston.com/, 2022. [cit. 2022/6/21]. Dostupné z: https://www.kingston.com/en/memory-cards/canvas-go-plus-microsd-card.

[18] *Light and Versatile Graphics Library* [online]. 2022. [cit. 2022/6/21]. Dostupné z: https://lvgl.io/.

[19] madmurphy. *Knihovna libconfini* [online]. 2021. [cit. 2022/6/16]. Dostupné z: https://github.com/madmurphy/libconfini/.

[20] *MariaDB C & C++ Connectors* [online]. 2022. [cit. 2022/6/21]. Dostupné z: https://mariadb.com/kb/en/mariadb-connector-c/.

[21] milekium, E. K. *Knihovna spidev-lib* [online]. 2020. [cit. 2022/6/16]. Dostupné z: https://github.com/milekium/spidev-lib.

[22] *Display - NL6448BC20-35F* [online]. http://usa.tianma.com/, 2020. [cit. 2022/6/16]. Dostupné z: http://usa.tianma.com/products-technology/product/nl6448bc20-35f.

[23] OCORNUT. *Dear ImGui* [online]. 2022. [cit. 2022/6/21]. Dostupné z: https://github.com/ocornut/imgui.

[24] PAULVHA. *Knihovna rfid-rc522* [online]. 2020. [cit. 2022/6/16]. Dostupné z: https://github.com/paulvha/rfid-rc522.

[25] *The Linux kernel (Custom Kernel)* [online]. 2022. [cit. 2022/6/21]. Dostupné z: https://www.raspberrypi.com/documentation/computers/linux_kernel.html.

[26] *Operating system images* [online]. 2022. [cit. 2022/6/21]. Dostupné z: https://www.raspberrypi.com/software/operating-systems/.

[27] *Raspberry Pi Produkty* [online]. https://www.raspberrypi.com, 2022. [cit. 2022/6/21]. Dostupné z: https://www.raspberrypi.com/products/.

[28] *RC522 RFID Module* [online]. https://components101.com, 2020. [cit. 2022/6/16]. Dostupné z: https://components101.com/wireless/rc522-rfid-module.

[29] *Raspberry Pi Documentation - Getting Started* [online]. https://www.raspberrypi.com/, 2022. [cit. 2022/6/21]. Dostupné z: https://www.raspberrypi.com/documentation/computers/getting-started.html.

[30] *RS-232 3-wire* [online]. https://en.wikipedia.org, 2020. [cit. 2022/5/29]. Dostupné z: https://en.wikipedia.org/wiki/RS-232#3-wire_and_5-wire_RS-232.

[31] *RS-232* [online]. https://en.wikipedia.org, 2020. [cit. 2022/5/29]. Dostupné z: https://en.wikipedia.org/wiki/RS-232#Data_and_control_signals.

[32] *RS-485* [online]. https://en.wikipedia.org, 2020. [cit. 2022/5/29]. Dostupné z: https://en.wikipedia.org/wiki/RS-485.

[33] SCHALLWIG, A. *Make your Raspberry Pi file system read-only (Raspbian Buster)* [online]. https://medium.com, 2019. [cit. 2022/6/10]. Dostupné z: https://medium.com/swlh/make-your-raspberry-pi-file-system-read-only-raspbian-buster-c558694de79.

[34] *Váhové čidlo SEN-10245* [online]. https://www.digikey.com, 2022. [cit. 2022/6/21]. Dostupné z: https://www.digikey.com/en/products/detail/sparkfun-electronics/SEN-10245/5843757.

[35] *RTC - AB-RTCMC-32.768kHz-EOA9-S3-DBT* [online]. https://eu.mouser.com/, 2020. [cit. 2022/6/16]. Dostupné z: https://eu.mouser.com/ProductDetail/ABRACON/AB-RTCMC-32.768kHz-EOA9-S3-DBT?qs=w%2Fv1CP2dgqp54Pcc4INOzw%3D%3D.

[36] Williams, C. – Kacur, J. *RT-Tests* [online]. 2019. [cit. 2022/6/16]. Dostupné z: https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/rt-tests.

# Příloha A: Obsah archivu

- Text_prace
  - img
- Aplikace_a_knihovny
  - bin - Adresář obsahuje spustitelný soubor aplikace, který je spustitelný na Raspberry Pi.
  - doc - html - Vygenerovaná dokumentace.
  - fonts - Adresář pro fonty, používané aplikací.
  - include - Adresář hlavičkových souborů.
  - languages - Adresář obsahuje soubory, které slouží k poskytnutí lokalizace aplikaci.
  - lib - Knihovny třetích stran.
    - hx711
    - imgui
    - libconfini
    - spidev-lib
  - src - Zdrojové soubory aplikace.
  - templates - Konfigurační soubory aplikace a nástroje cmake.
  - rt-kernel - Obsahuje archiv Linuxového kernelu 5.10.120-rt70 s aplikovaným PREEMPT_RT.

# Příloha B: Uživatelská příručka

Aplikace se skládá z jednoduchých elementů, kvůli potřebě snadné navigace klávesnicí, která má pouze 16 kláves. Použitá klávesnice je na Obrázku 6.1. Podle toho, v jakém režimu je zadávání aplikace, tak tlačítka mají jiné funkce. Na Obrázku 6.2, lze ve spodní části obrázku vidět lištu, která říká „Mode: NAVIGATION CAPS: No Shift: No“. Mode, neboli režim určuje aktuální funkci kláves. Tyto režimy jsou dohromady 4 a dělí se na: `NAVIGATION`, `INPUT INT`, `INPUT FLOAT`, `INPUT TEXT`.

V režimu `NAVIGATION` slouží klávesy **1-9** k přepínání obrazovek. Pokud se po stisku klávesy obrazovka nemění, tak to může znamenat, že tato obrazovka není použita. Klávesy **A-D** (ve stejném pořadí: nahoru, doprava, doleva, dolu) slouží pro navigaci výběru elementu uživatelského rozhraní. Klávesa **0** slouží k vrácení se na předchozí obrazovku. Klávesa # slouží k aktivaci zvoleného elementu. V případě, že se jedná o tlačítko, tak je aktivována funkce tlačítka. Pokud je vybraný element vstup, tak se klávesnice přepne do režimu `INPUT`.

Režim `INPUT` je rozdělen na pod-režimy `INT`, `FLOAT` a `TEXT`. V případě `INPUT INT` jsou klávesami **0-9** zadávána čísla a klávesou **B** může být vloženo mínus. V případě `INPUT FLOAT` jsou použity stejné klávesy jako `INPUT INT`, ale navíc je možné klávesou * vložit desetinnou čárku. `INPUT TEXT` umožňuje zadávat řadu textových znaků, ale kvůli malému počtu kláves zadávání funguje podobným způsobem jako na tlačítkových telefonech, kdy opakovaným stiskem klávesy jsou cykleny možnosti dané klávesy. Rozvržení této klávesnice je znázorněno v tabulce 6.1. Pokud si uživatel přeje zadat velké písmeno je třeba zapnout režim `CAPS`, což je provedeno klávesou **C**, tento režim je vypnut opakovaným stisknutím klávesy. Klávesnice také umožňuje klávesou **B** zapnout modifikaci režimem `SHIFT`. Tento režim momentálně umožňuje zapsat podtržítko klávesou pro mínus. Pro všechny pod-režimy platí, že klávesou **A** je `BACKSPACE`, tedy smazání předchozího znaku. Klávesou # je potvrzen zadaný vstup a režim klávesnice je opět přepnut do `NAVIGATION`.

| | | | |
|---|---|---|---|
| 1 | 2,A,B,C | 3,D,E,F | BACKSPACE |
| 4,G,H,I | 5,J,K,L | 6,M,N,O | LSHIFT |
| 7,P,Q,R,S | 8,T,U,V | 9,W,X,Y,Z | CAPSLOCK |
| *.- | 0 | # | |

Tabulka 6.1: Schéma vstupů klávesnice v režimu `INPUT TEXT`

Na Obrázku 6.2, je dále výzva k přihlášení pomocí RFID čtečky nebo zadáním uživatelských údajů. V horní části obrázku je stavová lišta, která zobrazuje:

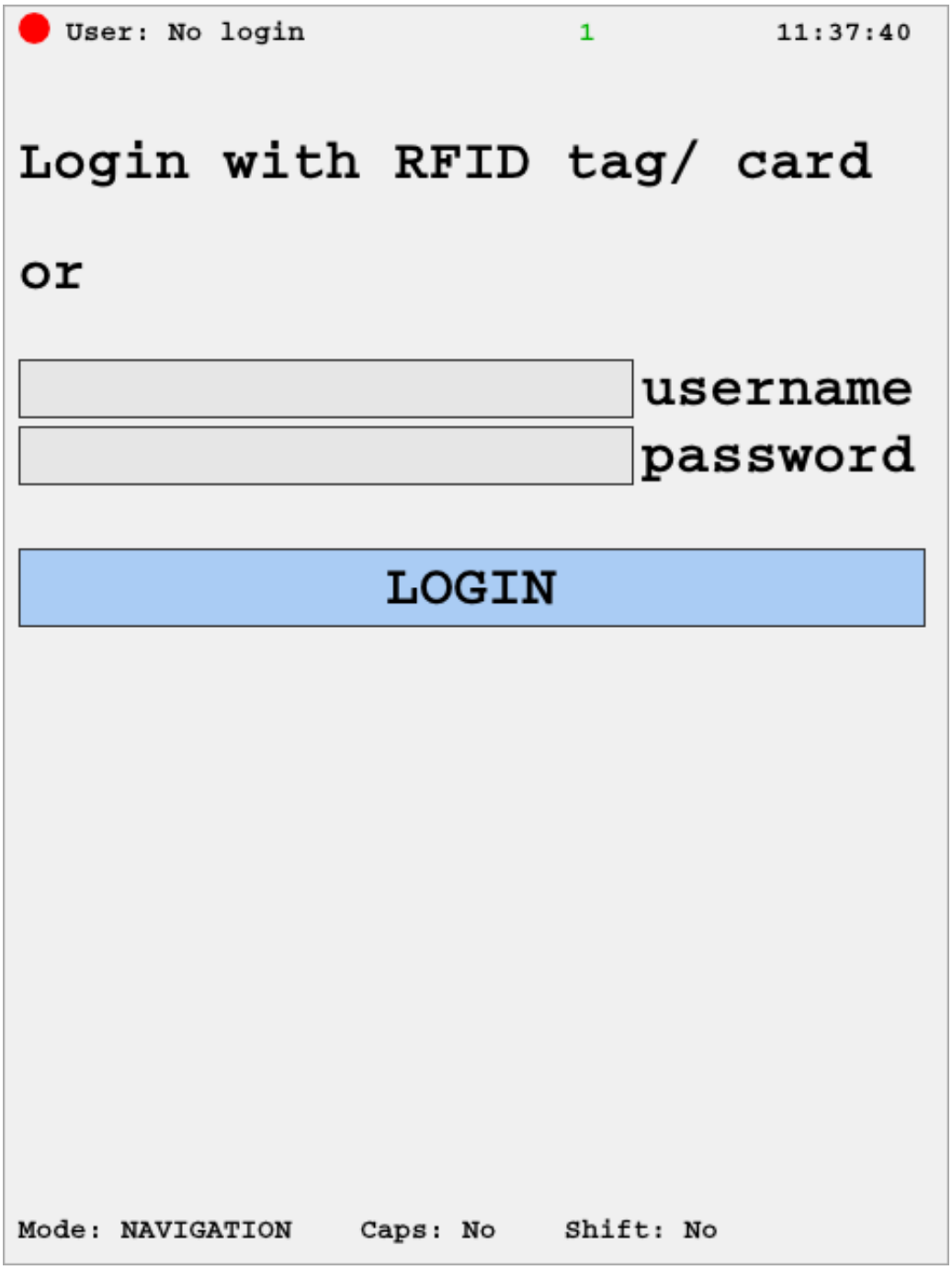
1. Přihlašovací uživatelské jméno
2. Aktivní obrazovku
3. Čas systému

Aplikace dále uživateli umožňuje:

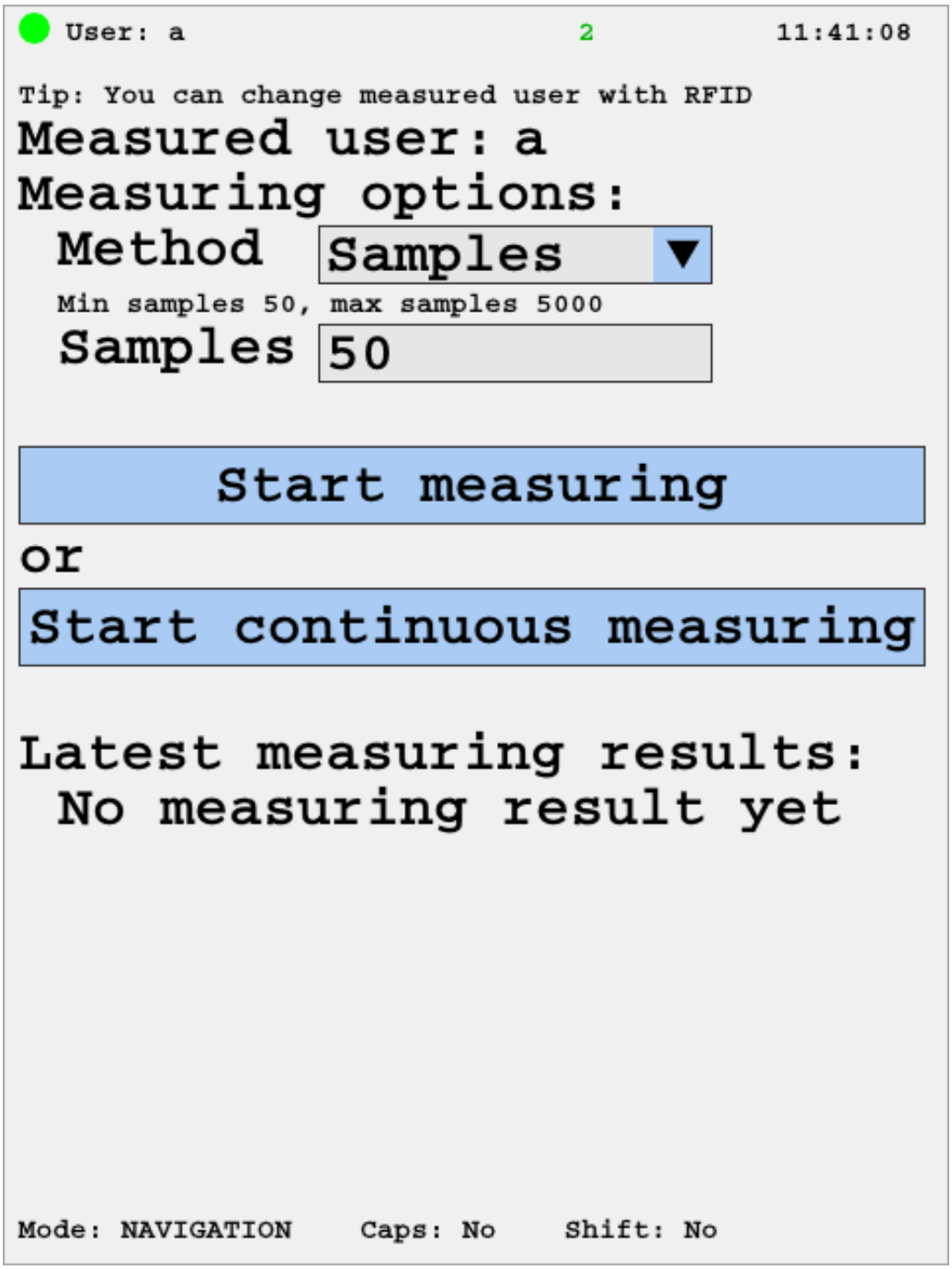
- Měření váhovým čidlem. Tato funkce je dostupná na obrazovce 2. Pokud je přihlášený uživatel zaměstnanec nebo administrátor, tak může přihlásit takzvaného „poduživatele“ (subuser) a vykonané měření jsou poté prováděna pro tohoto „poduživatele“.
- Prohlížení již provedených měření. Na obrazovce 3 může být měření vybráno. Pokud je přihlášen „poduživatel“, tak jsou zde na výběr jeho měření.
- Detail provedeného měření. Na obrazovce 4 je zobrazen detail měření, které bylo zvoleno na obrazovce 3.
- Výběr administrátorské akce. Obrazovka 5 je dostupná pouze administrátorům a slouží k výběru akce. Akcemi jsou kalibrace HX711, přidání nového uživatele, testování HX711 pomocí počtu vzorků, testování HX711 pomocí času měření a testování odezev klávesnice.
- Obrazovka 6 slouží k provedení akce, vybrané na 5 obrazovce.

Stavová lišta na vrchu a spodku obrazovky je přítomna na všech obrazovkách.

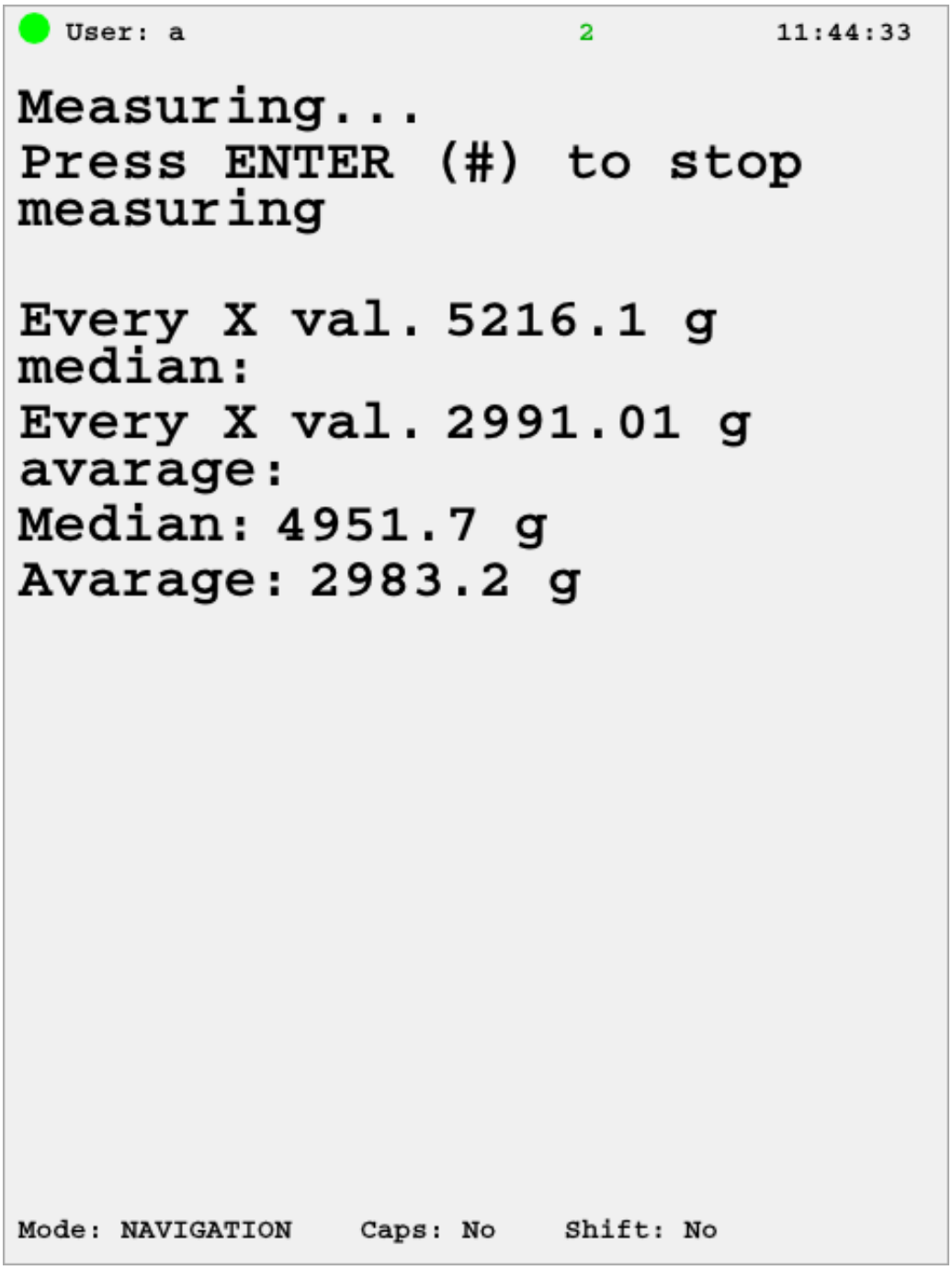

Obrázek 6.1: Použitá klávesnice



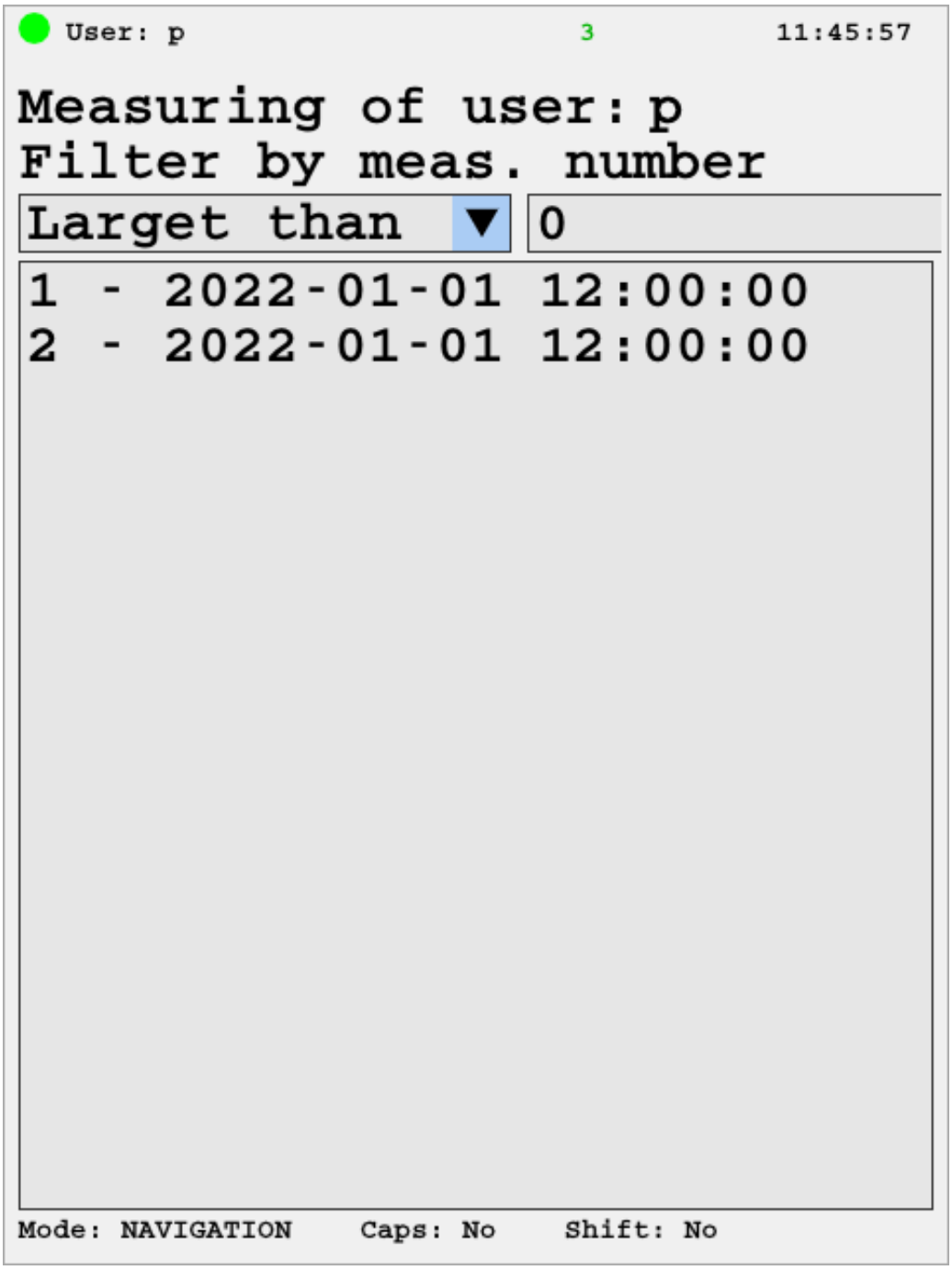

Obrázek 6.2: Obrazovka 1: přihlašování



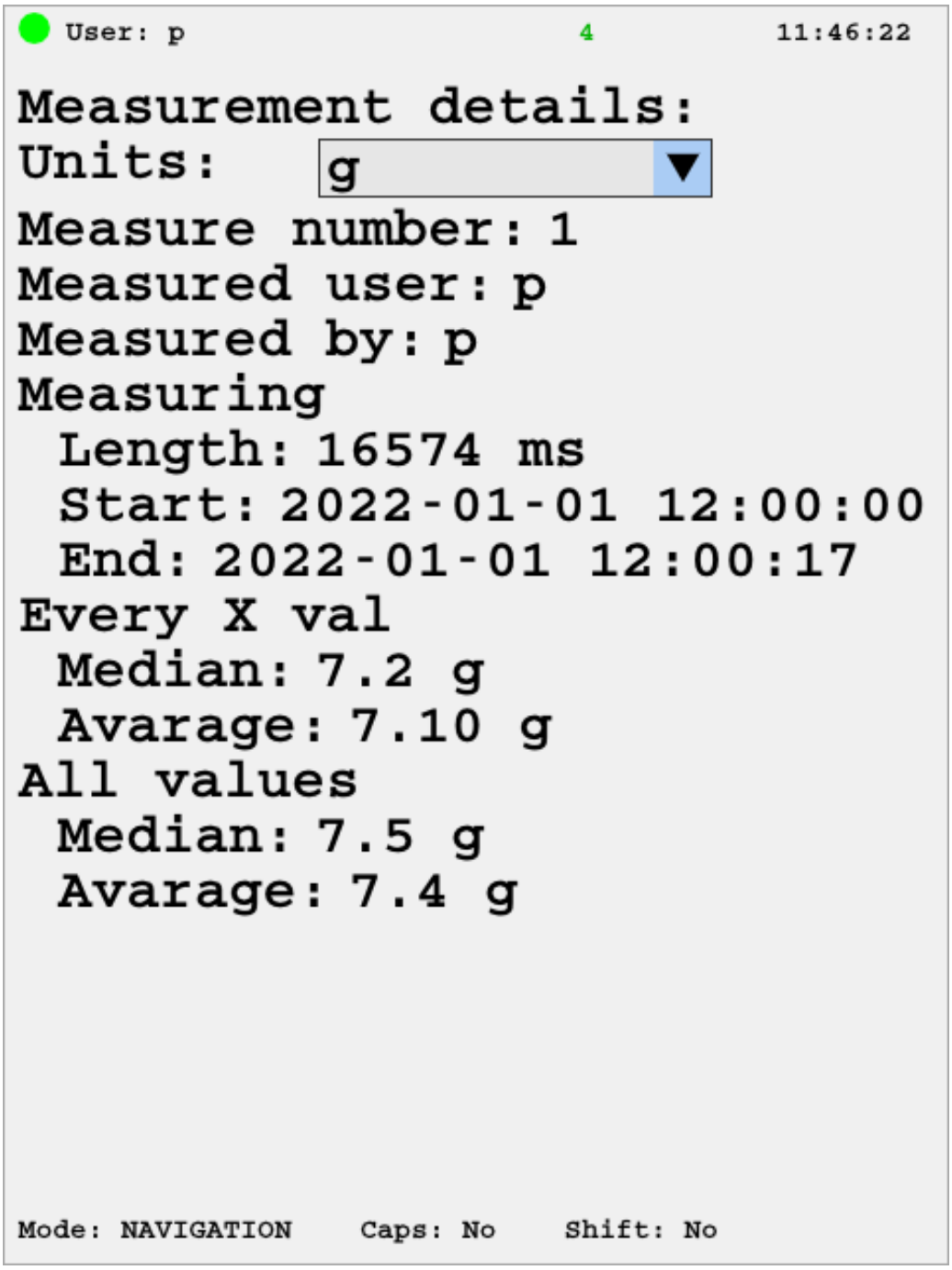

Obrázek 6.3: Obrazovka 2: výběr parametrů měření




Obrázek 6.4: Obrazovka 2: probíhající měření




Obrázek 6.5: Obrazovka 3: výběr měření




Obrázek 6.6: Obrazovka 4: detail měření

# Příloha C: Instalační příručka

## C.1: Stručný popis instalace

Následující kroky jsou pouze stručným popisem a všechny kroky jsou detailně popsané v následujících sekcích.

1. Instalace zvoleného systému (RPOS (Legacy) with Desktop) na SD kartu.
2. Po rozšíření filesystému instalace, zmenšit oddíl root (/) adresáře. Z uvolněného místa vytvořit nový oddíl, který bude zapisovatelný.
3. Zkompilovat kernel s PREEMPT_RT a nasadit ho.
4. Nainstalovat závislosti, upravit konfigurace a přeložit aplikaci (pokud není používána poskytnutá binární aplikace).
5. Ověřit funkčnost aplikace.
6. Zapnout read-only úpravu systému.

## C.2: Instalace na SD kartu

Postup přípravy SD karty a instalace RPOS (Legacy) with Desktop v systému Linux.

1. Stažení obrazu operačního systému [26] do zvoleného adresáře.[1]
2. Rozbalení archivu s obrazem.

   ```
   cd ~/Download
   unxz nazev_stazeneho_archivu.xz
   ```

3. Formátování karty.[2]

   ```
   sudo mkfs.ext4 /dev/mmcblk0
   ```

4. Text_prace

[1] V tomto návodu je použit adresář Download v domovském adresáři uživatele `~/Download`

[2] Název blokového zařízení se může lišit. V toho návodu byl název `/dev/mmcblk0`

- img

5. Aplikace_a_knihovny
   - bin - Adresář obsahuje spustitelný soubor aplikace, který je určen pro Raspberry Pi.
   - doc - html - Vygenerovaná dokumentace.
   - fonts - Adresář pro fonty, používané aplikací.
   - include - Adresář hlavičkových souborů.
   - languages - Adresář obsahuje soubory, které slouží k poskytnutí lokalizace aplikaci.
   - lib - Knihovny třetích stran.
     - hx711
     - imgui
     - libconfini
     - spidev-lib
   - src - Zdrojové soubory aplikace.
   - templates - Konfigurační soubory aplikace a nástroje cmake.
   - rt-kernel - Obsahuje archiv Linuxového kernelu 5.10.120-rt70 s aplikovaným PREEMPT_RT.

6. Nahrání zvoleného systému na kartu. V případě RPOS se vytvoří 2 oddíly.[2]

   ```
   sudo dd if="nazev_obrazu" of="/dev/mmcblk0"
       bs=4M conf=fsync
   ```

7. Vložení karty do Raspberry Pi a nechat systém načíst. Automaticky by mělo dojít k rozšíření root (/) oddílu na zbytek karty.

8. Vyjmout kartu a vložit ji zpět do systému se čtečkou. Nyní je třeba vytvořit oddíl, který později bude read-write, zatímco zbytek systému bude read-only. Oddíl je vytvořen zmenšením root (/) oddílu. Tento krok je možné udělat pomocí příkazové řádky, ale na Linuxu může být příjemnější použít software jako například gparted. Kdyby byl tento krok vykonán před automatický rozšířením, tak pak může dojít k chybě automatického rozšíření.

9. Připravit připojení k systému. Tento krok má více částí. První částí je připojit `/boot` jednotku RPOS.[3] Poté zde vytvořit soubor `ssh`, čímž se zapne SSH. Nakonec vytvořit soubor `wpa_supplicant.conf` pro připojení k WiFi.

   ```
   sudo mount /dev/mmcblk0p1 /mnt/boot
   cd /mnt
   touch boot/ssh
   touch boot/wpa_supplicant.conf
   ```

   Následně upravit obsah souboru `wpa_supplicant.conf`, který by měl obsahovat následující:

   ```
   update_config=1
   ctrl_interface=/var/run/wpa_supplicant

   network={
       ssid="jmeno_vasi_site"
       psk="heslo_vasi_site"
   }
   ```

10. Karta je nyní připravena. Po vložení do Raspberry Pi by se měl načíst systém, ke kterému je možné se připojit pomocí SSH na uživatele `pi`.

## C.3: Nastavení Raspberry Pi po instalaci

V této sekci jsou konfigurace Raspberry Pi systému a některých služeb. Ne všechna nastavení jsou povinná, ale části označené jako „[Doporučené]“ jsou vhodné provést pro snažší práci s Raspberry Pi. Některé konfigurace také vyžadují restart, a proto jsou označeny jako „[Restart]“. Pokud je prováděno více těchto konfigurací, tak stačí restartovat jednou po provedení všech nastavení.

1. - [Doporučené] - Nainstalování preferovaného editoru.

   ```
   sudo apt install vim
   ```

2. - [Doporučené] - Nastavení systémové proměnné TERM. Terminál přes SSH může způsobovat problémy při mazání znaků. Toto je také doporučeno přidat na konec `/etc/bash.bashrc`, aby se tato proměnná nastavila automaticky pro každý terminál.

[3] Blokové zařízení `/dev/mmcblk0p1` a adresář `/mnt/boot` se mohou lišit.

```
export TERM=vt100
```

3. - [Restart] - Nastavení vybraného DPI RGB565 (viz sekce 5.1.2). Do souboru `/boot/config.txt` přidat řádek `dpi_output_format=0x13`. DPI může vyžadovat více nastavení, ale protože při práci nebyl reálně použit DPI displej, tak není možné určit další nastavení.

4. - [Restart] - Nastavení SPI rozhraní. Do souboru `/boot/config.txt` přidat řádek `dtoverlay=spi0-2cs,cs0_pin=19,cs1_pin=26`.

5. - [Doporučené][Restart] - Vypnutí spořiče obrazovky. Do souboru `/etc/xdg/lxsession/LXDE-pi/autostart` je třeba přidat následující řádky.[4]

```
@xset s noblank
@xset s off
@xset -dpms
```

6. - [Restart] - Vypnutí uvítací hlášky a upozornění SSH na defaultní heslo uživatele.

```
sudo rm /etc/xdg/autostart/piwiz.desktop
sudo rm /etc/xdg/lxsession/LXDE-pi/sshpwd.sh
```

## C.4: Cross-kompilace kernelu

Následuje postup, jakým byl kompilován kernel pro tuto práci. Kernel byl kompilován ve virtuálním stroji s Linuxovým systémem Debian 11 a postupuje podle oficiální dokumentace [25].

1. Příprava závislostí pro konfiguraci a kompilaci kernelu.

```
sudo apt install git bc bison flex libssl-dev make
    libc6-dev libncurses5-dev
    crossbuild-essential-armhf
```

2. Příprava cílových adresářů kompilace.

```
mkdir rpi-kernel && cd rpi-kernel
mkdir -p rt-kernel/boot/overlays
```

[4]Pokud spořič obrazovky stále funguje, tak může pomoct přidat tyto řádky i do souboru `/etc/xdg/lxsession/LXDE/autostart`

3. Získání zdrojových kódů kernelu.[5]

```
git clone --depth=1 --branch \
    rpi-5.10.y https://github.com/raspberrypi/linux
```

4. Získání PREEMPT_RT patch a jeho aplikace.[6]

```
cd linux/
wget https://mirrors.edge.kernel.org/pub/linux/
   kernel/projects/rt/5.10/patch-5.10.120-rt70.
   patch.xz
unxz patch-5.10.120-rt70.patch.xz
cat patch-5.10.120-rt70.patch | patch -p1
```

5. Konfigurace kernelu.

```
KERNEL=kernel
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
    bcmrpi_defconfig
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
    menuconfig
```

   Menuconfig je libcurses CLI menu. Pro zapnutí PREEMPT_RT je třeba v tomto menu zapnout následující možnost. `General setup -> Fully preemptible kernel (real-time) -> Save as .config -> Exit`

6. Kompilace kernelu. Důležitým parametrem je „-j 28“. Tento parametr specifikuje, kolik „jobs“ může běžet současně. Dokumentace doporučuje výsledek programu „nproc“ * 1.5. V tomto případě jsou použita všechna vlákna, které jsou poskytnuté virtuálnímu stroji.

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
    zImage modules dtbs -j 28
```

7. Složení a příprava přeložených souborů k přenosu na Raspberry Pi.

```
cp arch/arm/boot/zImage ~/rpi-kernel/rt-kernel/
    boot/kernel-preempt.img
cp arch/arm/boot/dts/*.dtb* ~/rpi-kernel/rt-kernel/boot/
cp arch/arm/boot/dts/overlays/*.dtb* ~/bp-kernel/
```

[5] V této práci je použit Raspberry Pi Linux kernel verze 5.10.

[6] Je třeba vybrat patch s verzí odpovídající verzi kernelu z předchozího kroku nebo nejbližší dostupnou verzi.

```
    rt-kernel/boot/overlays
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
    INSTALL_MOD_PATH=~/rpi-kernel/rt-kernel
    modules_install -j 28
cd ~/rpi-kernel/rt-kernel/lib/modules/5.10.120-rt70+/
rm build source
```

8. Vytvoření přenositelného archivu kernelu.

```
cd ~/rpi-kernel
tar cJvf rt-kernel.tar.xz rt-kernel/
```

Těmito kroky vznikl archiv kernelu s aplikovaným PREEMPT_RT. Tento archiv je nyní připraven k přenosu a aplikaci na Raspberry Pi.

## C.5: Nasazení kernelu

Pro tento krok by měl být připravený archiv s Linux kernelem s aplikovaným PREEMPT_RT patch. Následuje postup nasazení tohoto kernelu na Raspberry Pi.

1. Přesunout archiv na běžící Raspberry Pi. Může být použit například program `scp`.

2. Rozbalit archiv v adresáři, kde se archiv nachází.

```
tar xJvf rt-kernel.tar.xz
```

3. Přesunout nebo zkopírovat rozbalené adresáře `boot` a `lib` do `/boot` a `/lib` na Raspberry Pi. [7]

```
sudo cp -r rt-kernel/boot/* /boot/
sudo cp -r rt-kernel/lib/* /lib/
```

4. Aktivovat nový obraz kernelu. Tento krok má 2 možnosti.

   (a) Přejmenovat `/boot/kernel.img` na `/boot/kernel.img.bak` pro zachování zálohy stávajícího kernelu. Následně přejmenovat `/boot/kernel-preempt.img` na `/boot/kernel.img`. [8]

[7] Je doporučeno zkontrolovat, že byl obsah adresáře `lib` správně přesunut. V podadresáři `modules` musí být podadresář se jménem odpovídající verzi nového kernelu.

[8] `/boot/kernel-preempt.img` je název obrazu kernelu, který je používán v těchto postupech, ale není závazný. Název tohoto obrazu může být téměř cokoliv.

(b) Názvy souborů můžou být zachovány. Do souboru `/boot/configtxt` přidat řádku `kernel=kernel-preempt.rt`.[8]

5. Restartovat Raspberry Pi. Po restartu ověřit, že je načtený správný kernel. Ověření může být provedeno příkazem:

   ```
   uname -a
   ```

   Očekávaným výstupem tohoto příkazu je řádka, která obsahuje řetězec PREEMPT_RT. Například:

   ```
   Linux raspberrypi 5.10.110-rt70+ #1 PREEMPT_RT Sun
    Jun 12 13:05:04 EDT 2022 armv6l GNU/Linux
   ```

# C.6: Instalace a konfigurace závislostí

V této sekci jsou popsány instalace a konfigurace služeb či knihoven potřebných ke spuštění nebo překladu řídící aplikace.

## Instalace

1. Instalace závislostí, které jsou nutné pro spuštění aplikace. Předpokladem je, že je použit RPOS (Legacy) with Desktop.

   (a) Knihovna pro HX711. Tato knihovna je odevzdána s touto prací, ale musí být nainstalována.

   ```
   cd Aplikace_a_knihovny/lib/hx711
   sudo ./install-deps.sh
   make && sudo make install
   ```

   (b) Instalace databázového severu MariaDB.

   ```
   sudo apt install mariadb-server
   ```

2. Instalace závislostí, které jsou potřebné pro překlad aplikace.

   ```
   sudo apt install cmake libspdlog-dev libsdl2-dev
       libmariadb-dev-compat
   ```

## Konfigurace

Ze závislostí v této sekci je třeba konfigurovat pouze MariaDB. Důvodem je, že databáze musí být zapisovatelná, i po tom co bude nasazena OverlayFS. Díky tmpfs stačí přesunout jen datový adresář. Postup je následující.

1. Nejdříve je třeba zastavit službu.

   ```
   sudo systemctl stop mariadb
   ```

2. Je předpokládáno, že je již nastaven zapisovatelný oddíl, který je připojen k `/media/writable_partition`. Na tomto oddílu je vytvořen adresář pro data.

   ```
   sudo mkdir /media/writable_partition/mysqld/
   ```

3. Nyní je třeba přesunout nebo zkopírovat existující adresář do nového.

   ```
   sudo rsync -rav /var/lib/mysql
       /media/writable_partition/mysqld
   ```

4. Po přesunutí nebo zkopírování adresáře je třeba nastavit nový adresář v konfiguračním souboru. V souboru `/etc/mysql/mariadb.conf.d/50-server.cnf` je třeba změnit následující řádku.

   ```
   datadir = /var/lib/mysql
   ```

   na

   ```
   datadir = /media/writable_partition/mysqld
   ```

5. MariaDB by nyní měla jít spustit a její datový adresář by měl být na zapisovatelném oddílu. Lokaci datového adresáře databáze je také možné ověřit pomocí dotazu `SELECT @@datadir;`

   ```
   sudo systemctl stop mariadb
   systemctl status mariadb
   ```

6. Pokud databáze ještě nebyla připravena pro řídící aplikaci tak je třeba připravit databázi, uživatele a tabulky. Toto je možné udělat pomocí poskytnutého skriptu, který navíc vytvoří testovací data. Nejsou-li požadována testovací data nebo je potřeba použít jinou databázi či jiného uživatele, tak jsou poskytnuty oddělené skripty pro založení databáze s uživatelem a tabulek.

   ```
   sudo mysql < src/sql/database_setup_in_one.sql
   ```

# C.7: Spuštění a překlad aplikace

Aplikace je překládána pomocí nástroje **cmake**. Tento nástroj k překladu používá sadu instrukcí popsaných v CMakeLists.txt. S odevzdáním jsou poskytnuty dva soubory CMakeLists.txt (CMakeLists.txt_desktop a CMakeLists.txt_rpi), ale zde je použit CMakeLists.txt_rpi, protože druhý je určený pro desktopový počítač, kde aplikace není plně podporována, takže se jedná spíše o vývojovou verzi. Za předpokladu, že jsou nainstalovány závislosti, tak je postup následující.

1. Je otevřený terminál, který je v kořenovém adresáři aplikace. To znamená, že příkaz `ls`, mimo jiné, vylistuje adresář `src` a `CMakeLists.txt`. Následujícími příkazy se vytvoří adresář build a provede se konfigurace cmake.

   ```
   mkdir build
   cmake -DCMAKE_BUILD_TYPE=Release -S . -B build/
   ```

2. Nyní je aplikaci možné přeložit.

   ```
   cmake --build build/
   ```

Po dokončení kompilace v adresáři build vznikne spustitelný soubor **rpi_control_system**. Tento soubor je spustitelný, ale pro jeho spuštění je navíc třeba konfiguračního souboru. Jako výchozí aplikace hledá soubor **./app_config.conf**. Aplikace se ale nepustí, pokud tento soubor není nalezen. Soubor je také možné předat aplikaci pomocí přepínače **-c** a aplikaci je poté možné pustit z libovolného místa. Spuštění aplikace potom může vypadat například: `./rpi_control_system -c ~/.config/cs_config.conf`.

V konfiguračním souboru jsou ale použity cesty k fontům, jazykům a doplňujícímu konfiguračnímu souboru. Pokud jsou tyto cesty zadávány relativně, tak je třeba zajistit, že cesty v konfiguračním souboru odpovídají relativním cestám z pozice, ze které je aplikace pouštěna.