VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

NÁVRH A REALIZACE ŘÍZENÍ A VIZUALIZACE DEMONSTRAČNÍ ROBOTICKÉ BUŇKY
DEMONSTRATION ROBOTIC CELL DESIGN AND IMPLEMENTATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Alexander Korotynskiy
AUTHOR

VEDOUCÍ PRÁCE Ing. Václav Kaczmarczyk, Ph.D.
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Alexander Korotynskiy ***ID:*** 186530

Ročník: 2 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Návrh a realizace řízení a vizualizace demonstrační robotické buňky

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s existujícím řešením demonstrační robotické buňky.
2. Navrhněte koncept povelování průmyslového robotu spočívající v dynamické tvorbě trasy.
3. Navrhněte vylepšení uživatelského rozhraní stroje včetně přenosu dat do nadřazeného systému.
4. Celé řešení realizujte a otestujte.
5. Řešení zadokumentujte.

DOPORUČENÁ LITERATURA:

Průběžné zprávy projektu "Kooperativní robotické platformy pro průmyslové aplikace".

Vaskaran Sarcar: Getting Started with Advanced C#

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 18.5.2022

Vedoucí práce: Ing. Václav Kaczmarczyk, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

Abstrakt

Táto diplomová práce se zabývá návrhem a realizací řízení a vizualizace demonstrační robotické buňky na vaření kávy. V rámci práce byly vybraný vhodné hardwarové prostředky a byly navržené elektrická schémata zapojení. Hlavní řídicí aplikace v jednodeskovém počítači Raspberry Pi je založena na platformě .NET a multiplatformním GUI frameworku Avalonia. Pro světelnou indikace procesu přípravy kávy byl realizován inteligentní maják na základě řídicí jednotky ESP8266 NodeMCU V2. Práce je taktéž věnována realizaci detekce stavů kávovaru s využitím řídicí desky T-ETH-POE ESP32-WROOM.

Klíčová slova

Kávovar, Robot, Raspberry Pi, RFID, ESP32-WROOM, ESP8266, NodeMCU, .NET, Avalonia, ModbusTCP, HTTP

Abstract

The topic of this thesis is the design and implementation of control and visualization of a demonstration robotic cell for coffee brewing. During the work, suitable hardware resources were selected, and electrical wiring diagrams were designed. The main control application on the single-board computer Raspberry Pi is based on the .NET platform and the Avalonia cross-platform GUI framework. An intelligent beacon based on the ESP8266 NodeMCU V2 control unit was implemented to perform light indication of the coffee brewing process. The thesis is also devoted to the implementation of coffee machine status detection using the control board T-ETH-POE ESP32-WROOM.

Keywords

Coffee machine, Robot, Raspberry Pi, RFID, ESP32-WROOM, ESP8266, NodeMCU, .NET, Avalonia, ModbusTCP, HTTP

Bibliografická citace

KOROTYNSKIY, Alexander. *Návrh a realizace řízení a vizualizace demonstrační robotické buňky* [online]. Brno, 2022 [cit. 2022-04-18]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/141630. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Václav Kaczmarczyk.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	Bc. Alexander Korotynskiy
VUT ID studenta:	186530
Typ práce:	Diplomová práce
Akademický rok:	2021/22
Téma závěrečné práce:	Návrh a realizace řízení a vizualizace demonstrační robotické buňky

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

.

V Brně dne: 28. dubna 2022 ----------

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Václavu Kaczmarczykovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: 28. dubna 2022 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V současné době dochází k velkému rozvoji v oblasti automatizace. Příprava kávy není výjimkou. Každodenně se ve světě vypije kolem 2 miliard šálků kávy. Může se zdát, že proces přípravy kávy už je maximálně automatizován, avšak hledají se jiné i způsoby řešení a realizace tohoto procesu.

Tato diplomová práce se zabývá návrhem a realizací řízení a vizualizace demonstrační robotické buňky na vaření kávy. Celý koncept demonstrační robotické buňky na vaření kávy spočívá v tom, že prostřednictvím přístupové karty bude uživateli umožněno si na dotykovém displeji vybrat a objednat kávu, následně robotická ruka automaticky uchopí kelímek a zmačkne příslušné tlačítko na kávovaru. Po nalití požadovaného typu nápoje vydá robot uživateli kelímek k odebrání.

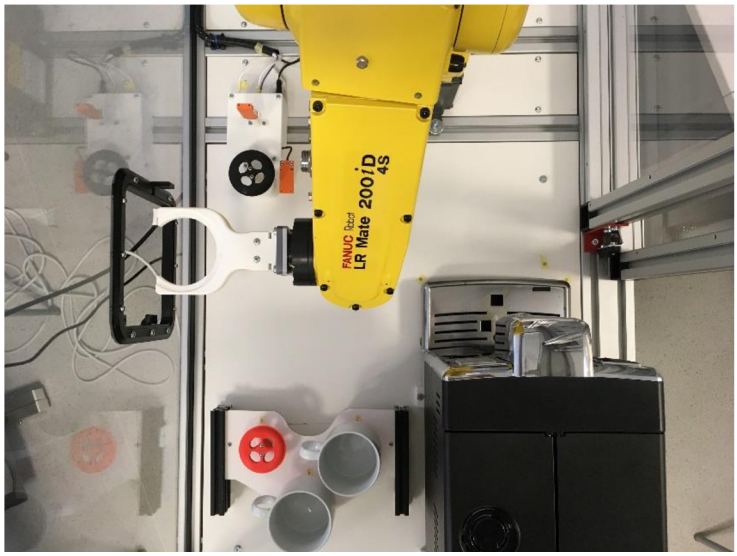
V rámci jednotlivých kapitol této práce je popsán proces návrhu a vývoje demonstrační robotické buňky. V první kapitole je detailně popsán návrh zlepšení stávající robotické buňky, včetně popisu stávajícího řešení. Ve druhé kapitole následuje teoretický úvod této práce. Ve třetí kapitole jsou popsané hardwarové prostředky, které byly vybrané a použité pro realizace této práce, včetně schémat zapojení. V následující kapitole je popsána softwarová realizace této práce, včetně zdůvodnění použitých programovacích technologií. V poslední kapitole je popsán průběh oživení realizovaného řešení.

1. NÁVRH ZLEPŠENÍ STÁVAJÍCÍ BUŇKY

V této kapitole bude popsáno stávající řešení demonstrační robotické buňky na vaření kávy, analýza požadavků této práce, navrhované změny pro splnění požadavků a návrh technického řešení.

1.1 Popis stávajícího řešení

V této podkapitole bude popsáno stávající řešení demonstrační robotické buňky na vaření kávy. Jedná se o buňku, součástí které jsou hlavně kávovar, robotická ruka značky Fanuc, stojan na hrnky, točna na hrnky, gripper a tlačítkový ovládací panel. Na obrázku 1.1 lze vidět stávající demonstrační robotickou buňku.

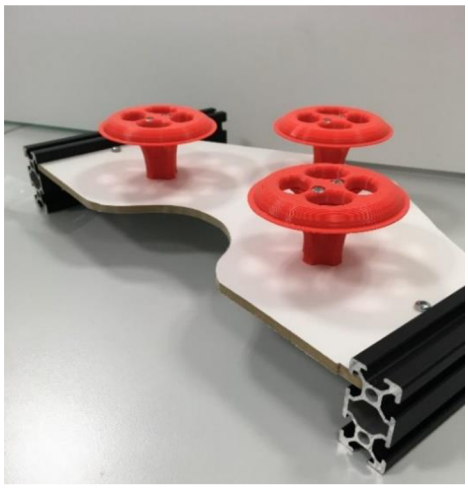


Obrázek 1.1 Stávající robotická buňka [1]

V rámci stávajícího řešení byl navržen a zrealizován stojan na 3 hrnky. Stojan byl sestaven z hliníkových profilů 20x20 a byl ukotven pomocí úchytů ke spodní podložce, na které stojí. Stojan obsahuje 3 podstavce pro hrnky, kde se každý podstavec skládá z klobouku a nohy. Klobouk byl navržen takovým způsobem, aby sedl na spodní část hrnku a tím byla zajištěna jeho stabilita. [1]

V předchozí bakalářské práci [1] byla navržená a realizována točna na hrnky. Točna byla navržena za účelem otáčet hrnky, aby robot vždy pracoval se správně natočeným hrnkem a nedocházelo tak ke kolizi ucha hrnku s kávovarem. Na obrázku 1.2 a) lze vidět točnu na hrnky a na obrázku 1.2 b) stojan na 3 hrnky. [1]

a)



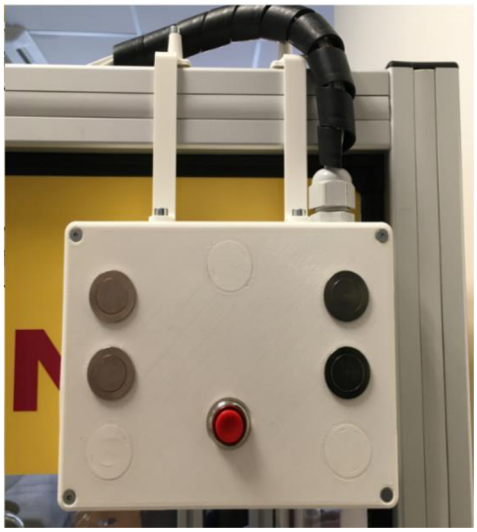
b)

Obrázek 1.2 Točna na hrnky (a) [1] a stojan na 3 hrnky (b) [1]

V rámci stávajícího řešení byl navržen a realizován i gripper, který byl následně uchycen na robotickou ruku a slouží tak ke manipulaci robotické ruky s hrnky, kde hrnek přímo zapadne do kleštin gripperu. [1]

V předchozí bakalářské práci byl také navržen ovládací panel na objednaní kávy. Jedná se o panel s mechanickými tlačítky, kde rozmístění tlačítek odpovídá rozložení tlačítek na samotném kávovaru s tím, že osmé tlačítko navíc slouží ke ukládání hrnků. V rámci bakalářské práce byly taktéž naprogramovány jednotlivé pohyby robota Fanuc přes teach pendant, které následně byly spojené do jednoho programu pro každý typ kávy. Následně pomocí tlačítek na ovládacím panelu se spouštěli programy v robotu Fanuc na přípravu příslušné kávy. Na obrázku 1.3 a) lze vidět gripper a na obrázku 1.3 b) tlačítkový ovládací panel. [1]

a)



b)

Obrázek 1.3 Gripper (a) [1] a tlačítkový ovládací panel (b) [1]

1.2 Analýza požadavků

Mezi požadavky této práce patří možnost dávkovat kávu do kelímků a vyhnut se tak řešení z předchozí práce, kde káva se nalívala do hrnků, pro které byl vyvinut stojan na tři hrnky, točna na hrnky a gripper pro uchycení hrnků. Dalším požadavkem byla možnost autentizace a autorizace uživatele prostřednictvím přístupové karty, což umožnuje implementovat omezený přístup pro použití robotické buňky na vaření kávy. Zároveň toto řešení umožňuje sbíraní statistických dat každého uživatele, jako například počet vypitých káv za nějakou dobu, ale i například kolik musí v budoucnu zaplatit za vypitou kávu. Následně bylo nutné realizovat vylepšení uživatelského rozhrání robotické buňky, což vedlo k opuštění myšlenky řešení z předchozí práce, kde k tomuto účelu byl vyvinut ovládací panel s mechanickými tlačítky. Dalším požadavkem práce bylo implementovat komunikaci nově navržené ovládací jednotky celé robotické buňky s robotickou rukou Fanuc. Taktéž bylo nutné realizovat přijímání a přenos dat do nadřazeného systému, kde se ukládají data o tom, zda uživatel má přístup ke kávovaru a statistická data každého uživatele, který má přístup k vaření kávy. V neposlední řadě zde byl požadavek na návrh a realizaci světelné indikace samotného procesu přípravy kávy. Mezi požadavky této práce taktéž patří detekce stavů kávovaru a možnost detekovat kdy je káva připravena.

1.3 Navrhované změny pro splnění nových požadavků

- Pro splnění prvního požadavku práce, tedy možnosti dávkování kávy z kávovaru do kelímků byl zakoupen a použit zásobník kelímků, do kterého je možné umístit až 50 kelímků. To na rozdíl od stávajícího řešení výrazně zvyšuje počet připravených káv bez zásahu člověka do robotické buňky. K tomuto účelu bylo potřeba vyvinout nový gripper na uchycení kelímků. Výhodou tohoto řešení také je to, že pro robotickou ruku se zmenší počet jednotlivých pohybů, což ve výsledků redukuje potřebný čas pro přípravu kávy.
- Pro možnost autentizace a autorizace uživatele prostřednictvím přístupové karty bylo navrženo použití RFID čtečky.
- V rámci splnění požadavku na vylepšení uživatelského rozhrání bylo dále navrženo použití dotykového displeje pro komunikaci s uživateli.
- V rámci splnění požadavků na komunikace s robotickou rukou Fanuc, přenos dat do nadřazeného systému a realizace uživatelského rozhrání bylo navrženo použít řídicí jednotku, pomocí které je možné tyto požadavky splnit.
- Pro splnění požadavku na světelnou indikaci procesu přípravy kávy bylo rozhodnuto realizovat inteligentní maják, který by měl být umístěn na demonstrační robotické buňce.
- Pro splnění požadavku na detekce stavů kávovaru bylo navrženo snímat displej kávovaru, na kterém lze poznat jeho aktuální stav.

- V rámci této práce bylo taktéž uvažováno o možnosti detekce uvařené kávy. Původním návrhem bylo realizace časovače, kde se čekalo určitou dobu v závislosti na typu kávy.
  Jednou z možností by byla realizace měření proudu procházejícím pumpou snímačem proudu, což by však vyžadovalo zásah do vnitřních obvodů samotného kávovaru. Tato možnost byla ale zamítnuta právě z důvodu nežádoucího zásahu do kávovaru.
  Jelikož pumpa kávovaru v čase přípravy kávy produkuje charakteristický zvuk, tak bylo navrženo a realizováno řešení měření zvuku tohoto pomocí mikrofonu. Po realizaci bylo ale zjištěno, že hlasitost kávovaru je porovnatelná s vnějšími zvuky, jako je například mluvení lidí vedle robotické buňky. Proto toto řešení bylo také zamítnuto z důvodu jeho nespolehlivosti.
  Dalším potenciálním řešením bylo detekce hmotnosti kelímku s kávou pomocí snímače hmotnosti. Z konstrukčních důvodu však toto řešení není realizovatelné. Kelímek je celou dobu ze spodu držen přísavkou na robotu a nelze tak snímač hmotnosti umístit. Z důvodu nedostatku místa pod nalévacím bodem kávovaru bylo zamítnuto i použití bezkontaktního snímače výšky hladiny, který by měl snímat výšku kávy v kelímku. Nepřesné měření by mohl dělat i škraloup u některých typu káv.
  Dalším potenciálním řešením by byla bezkontaktní detekce proudu kávy z kávovaru, ale řešení bylo taktéž zamítnuto z důvodu náhodného přerušování proudu kávy kávovarem, což neumožnuje zjištění přesného okamžiku dokončení přípravy kávy. Proto bylo rozhodnuto zůstat u původního návrhu a realizovat časovač v závislosti na vybrané uživatelem kávě.

1.4 Návrh technického řešení

Pro splnění požadavků této práce v první řádě bylo nutné vybrat hlavní řídicí jednotku celé demonstrační robotické buňky, která bude vyhovovat realizaci všech požadavků jak na hardwarové stráně, tak i na straně softwarové. Podle požadavků práce hlavní řídicí jednotka musí umožnovat připojení velkého dotykového displeje a realizace uživatelského rozhrání, které se bude zobrazovat na displeji. Řídicí jednotka musí také splňovat požadavky na komunikaci s nadřazeným systémem pro přijímání a ukládání dat do nadřazeného systému a umožnovat komunikace s robotem Fanuc.

Jako hlavní řídicí jednotku celé buňky bylo rozhodnuto použít jednodeskový počítač Raspberry Pi 4B, který umožnuje tyto požadavky splnit po hardwarové a softwarové stránce a je detailně popsán v podkapitole 3.3. Pro zobrazení uživatelského rozhrání bylo rozhodnuto použít Raspberry Pi 7‘‘ dotykový displej.

Pro možnost autentizace a autorizace uživatele prostřednictvím přístupové karty bylo rozhodnuto použít RFID čtečku MFRC522, a to jak z ekonomických důvodů, tak

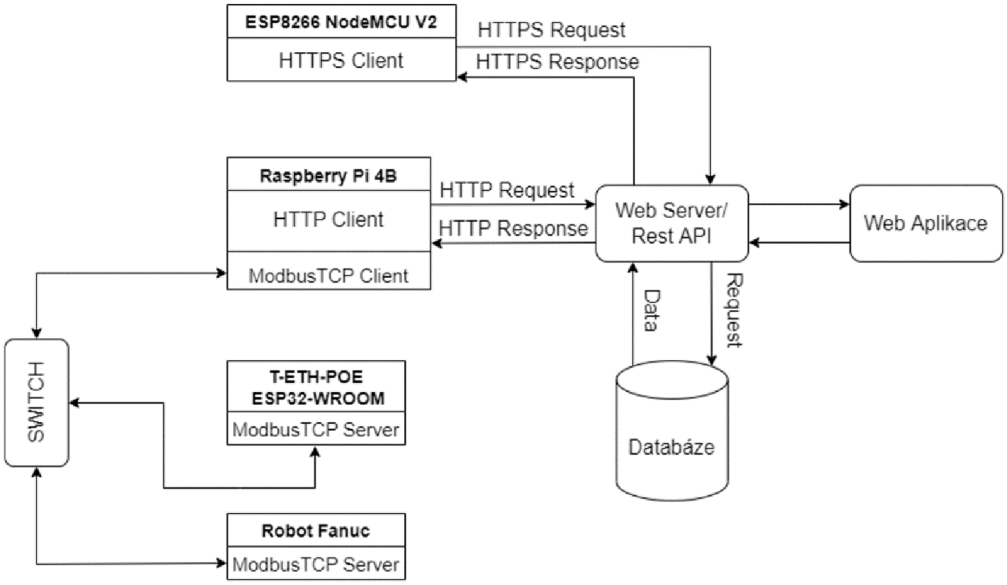
z pohledu komunikační kompatibility s hlavní řídicí jednotkou Raspberry Pi. Čtečka MFRC522 je detailně popsaná v podkapitole 3.4.

V rámci splnění požadavků na komunikaci hlavní řídicí jednotky s robotickou rukou Fanuc bylo rozhodnuto realizovat komunikaci prostřednictvím protokolu ModbusTCP, který je spolehlivý, univerzální a je možné jej v obou jednotkách implementovat (viz kapitola 2.2.5).

Pro splnění požadavku detekce stavů kávovaru bylo rozhodnuto snímat barvu displeje kávovaru snímačem barev. Tohoto cíle je možné dosáhnout i kamerou s použitím algoritmů počítačového vidění. Z ekonomického hlediska a vzhledem ke složitosti algoritmů počítačového vidění byl vybrán snímač barvy TCS3200 (detailně podkapitola 3.5). Zde byl kladen důraz na realizaci snímaní barvy displeje kávovaru jako modulární řešení, proto bylo nutné vybrat další řídicí jednotku pro tento účel. Tato řídicí deska musí umožnovat komunikaci s hlavní řídicí jednotkou Raspberry Pi prostřednictvím už navrženého komunikačního protokolu ModbusTCP. K tomuto účelu bylo rozhodnuto o použití desky T-ETH-POE ESP32-WROOM, která je detailně popsána v podkapitole 3.6.

Pro splnění požadavku na světelnou indikaci procesu přípravy kávy byl realizován inteligentní maják, jako samostatná modulární jednotka, základem které je řídicí deska ESP8266 NodeMCU V2 (detailně viz kapitola 3.7). Táto deska byla vybrána hlavně z důvodu umožňující komunikace s nadřazeným systémem pomocí HTTP/HTTPS protokolu.

Pomocí HTTP/HTTPS protokolu byla taktéž realizována komunikace hlavní řídicí jednotky Raspberry Pi 4 B s nadřazeným systémem (serverem). Blokové schéma komunikace celé demonstrační robotické buňky spolu s nadřazeným serverem lze vidět na obrázku 1.4.



Obrázek 1.4 Blokové schéma komunikace demonstrační robotické buňky

Hlavní řídicí jednotkou celé robotické buňky je Raspberry Pi 4 B, proto v této jednotce bylo nutné realizovat ModbusTCP klienta. V řídicí desce ETH-POE ESP32-WROOM bylo nutné realizovat ModbusTCP server pro její komunikaci hlavní řídicí jednotkou. V robotu Fanuc bylo nutné realizovat ModbusTCP server pro komunikaci robota s hlavní řídicí jednotkou Raspberry Pi 4 B. Jelikož bylo potřeba, aby se tři různá zařízení se nacházeli v jedné sítě, tak bylo nutné využít ethernet switch.

Zařízením byly přidělený následující statické IP adresy:

- IP adresa v hlavní řídicí jednotce Raspberry Pi 4 B byla nastavena na: 192.168.5.2
- IP adresa v robotu Fanuc byla nastavena na: 192.168.5.3
- IP adresa v řídicí desce T-ETH-POE ESP32-WROOM byla nastavena na: 192.168.5.5

V následujících kapitolách je detailně popsán průběh implementace celého technického řešení.

2. TEORIE

V této kapitole budou popsané komunikační možností komerčních robotů, zejména robota značky Fanuc. Budou také popsány komunikační knihovna a testovací aplikace, které byly dodány vedoucím této práce a byly následně využity v rámci této práce.

2.1 Řízení robotického systému

Standardním přístupem pro řízení průmyslového robotického systému je jeho povelování prostřednictvím jednoduchých binárních signálů, mezi které patří například povel pro vykonání určitého připraveného programu. Rovněž aktuální stav robotu (např. zda je spuštěn program či zda robot není v chybovém stavu) je možné vyčíst prostřednictvím binárních signálů z robot kontroléru. [2]

V rámci obslužného softwaru robotu bývá připraveno několik pevně daných programů (které se samozřejmě mohou parametrizovat v rámci možností kontroléru robotu a programovacího jazyka, kterým kontrolér disponuje). Výběr konkrétního programu pro spuštění lze rovněž realizovat nastavením vhodných úrovní na příslušné binární vstupy kontroléru. [2]

Vzhledem k tomu že typickým způsobem řízení robotu v robotické buňce je řízení za využití robot kontroléru, který umožňuje připojení široké škály periferií, neumožňují výrobci zpravidla vzdálené dynamické řízení jednotky. Pohyb robotu lze tedy ovlivnit zpravidla přímo z programu implementovaného uvnitř jeho řídicí jednotky. [2]

Přestože robot kontroléry jsou v dnešní době velmi výkonné výpočetní stanice s univerzálními operačními systémy a komunikačními rozhraními, jsou jejich možnosti výrobci záměrně omezovány kvůli specifickým licenčním politikám. Robotické systémy využívají zejména své Ethernet rozhraní k přímému programování robotů, nahrávání připravených programů, či spouštění konkrétního předem vytvořeného programu. Další možnosti jsou již omezovány a bývají zpřístupněny po dokoupení příslušného licenčního balíčku. V případě požadavku na komunikaci prostřednictvím rozhraní, které není v kontroléru fyzicky přítomno, nabízejí výrobci možnost dodat komunikační kartu. [2]

2.2 Analýza komunikačních možností komerčních robotů

Za účelem zápisu pozic do množiny bodů, skrze kterou má mechanická jednotka iterovat, bylo nutné vybrat vhodný komunikační protokol tak, aby byl co nejsnazší cestou dostupný v široké škále běžně dostupných robotických jednotek. Porovnání jednotlivých komunikačních protokolů a rozhraní jednotek posuzovaných výrobců jsou uvedena v tabulce 2.1:

Tabulka 2.1 Porovnání komunikačních rozhraní komerčních robotů [2]

	ABB	YAKSAWA	KUKA	FANUC	Kawasaki	EPSON
EtherNet/IP	x	x	x	x	x	x
DeviceNet	x	x	x	x	x	x
PROFIBUS DP	x	x	x	x	x	x
PROFINET	x	x	x	x	x	x
CC-Link	x			x	x	x
EtherCAT		x	x		x	x
MECHATROLINK-II		x				
MECHATROLINK-III		x				
Modbus TCP		x		x	x	
Modbus RTU		x				
Embedded Ethernet				x		
Fast Ethernet				x		
High Speed Serial Bus				x		
FL Net				x		
MT Connect server				x		
InterBUS			x		x	
CANOpen					x	
ControlNet					x	
VaranBUS			x			

Mezi nejběžnější vhodná komunikační rozhraní patří tato:

2.2.1 Ethernet/IP

Jedná se o široce používaný standard průmyslového Ethernetu, který byl vyvinut zejména pro průmyslovou automatizaci. Toto průmyslové řešení je plně kompatibilní se standardním Ethernetem TCP/IP, tedy umožňuje využití standardních technických a programových prostředků Ethernetu pro konfigurování a ovládání automatizačních prostředků.

V rámci sítě EtherNet/IP jsou jednotlivým uzlům přiřazeny předem definované typy zařízení se specifickými vlastnostmi a funkcemi (profily). Profily zařízení a aplikační vrstva EtherNetu/IP jsou tvořeny protokolem Common Industrial Protocol. Protokol CIP pracuje s objektovým modelem a využívá komunikaci na principu producent-konzument (producer-consumer). Použitím protokolu CIP se dosahuje interoperability mezi všemi sítěmi, které ho podporují – tedy sítěmi DeviceNet, ControlNet a EtherNet/IP. Mezi hlavní přednosti systému EtherNet/IP patří ucelený systém přenosu

dat metodou producent-konzument, koexistence s další komunikací v rámci sítě Ethernet a možnost využití běžné infrastruktury. [2]

2.2.2 DeviceNet

DeviceNet je celosvětově rozšířená průmyslová sběrnice. Většina výrobců průmyslových systémů nabízí komunikační rozhraní pro tuto sběrnici jako modulární kartu do svých zařízení. Komunikační standard používá pro svou funkci standard CAN (Controller Area Network) na úrovni linkové vrstvy OSI Modelu a standard Common Industrial Protocol pro pátou a vyšší vrstvy tohoto modelu. [2]

DeviceNet pracuje na principu poskytovatel/příjemce, což umožňuje podporu širokého spektra komunikačních hierarchií a možnosti prioritizace komunikace a přenosu dat. Síť lze nakonfigurovat v režimu Master-Slave nebo jako distribuovanou řídicí architekturu s typem komunikace peer-to-peer. Výhodou je napájení podřízených zařízení přímo po komunikační sběrnici. Sběrnice umožňuje přenos dat různými rychlostmi, avšak nejvýše 500 kb/s, umožňuje přímé napájení jednotek. Maximální velikost (rozlehlost) sítě je cca 500 m. [2]

Jak již bylo zmíněno v prvním odstavci, na vyšších vrstvách se používá Common Industrial Protocol, což je striktně objektově orientovaný protokol, kde každý objekt se skládá z atributů, služeb a chování. Dále je definováno několik obecných typů objektů – komunikační objekty, aplikačně-specifické objekty a případně objekty specifické pro konkrétního výrobce. [2]

2.2.3 EtherCAT

EtherCAT je průmyslová sběrnice založená opět na vrstvě Ethernet vyvinuta firmou Beckhoff Automation. Od ostatních podobných sběrnic se liší principem zpracování paketů. Pakety nejsou komunikovány tak, jak je běžné, tedy u ostatních sběrnic (tedy principem client-server). Namísto toho master zařízení vysílá do sítě uspořádané jako fyzický kruh jediný paket, který je přijat prvním slave zařízením na lince. Toto zařízení paket během přijímání zpracovává – vyčítá z něj a ukládá do něj svá data. Pak okamžitě odesílá do druhého fyzického rozhraní, kterým je slave 1 připojen ke slave 2. Poslední slave zařízení kruh uzavírá a odesílá datový rámec přímo k master zařízení. Každý node v síti EtherCAT je tedy vybaven dvěma fyzickými rozhraními. Každý paket může být rozdělen do několika sub-paketů, každý z nich může nabývat velikosti až 4 GB. Protokol dále umožňuje broadcast, multicast, či komunikaci mezi slave zařízeními, která však musí být iniciována master zařízením. [2]

Nevýhodou protokolu EtherCAT je jeho uzavřenost, nemožnost provozu na univerzálním hardware a jeho využití by znamenalo značné finanční náklady.

2.2.4 ProfiNET

ProfiNET je další z průmyslových sběrnic založených na základech průmyslového Ethernetu, který klade velký důraz na časově deterministickou komunikaci. Jedná se o

otevřený protokol definovaný IEC standardem umožňující přenos rychlostmi 10 či 100 Mbit/s. Nejčastěji se využívá pro komunikaci mezi PLC či DCS a vstupně/výstupními zařízeními. Komunikační model, který ProfiNET využívá je producent-konzument. Vzhledem k jeho otevřenosti by ProfiNET připadal v úvahu pro použití v rámci projektu. Tomu nahrává i fakt, že výukové laboratoře na FEKT UAMT VUT Brno jsou vybaveny množstvím zařízení kompatibilních s tímto komunikačním standardem. [2]

2.2.5 ModbusTCP

ModbusTCP je otevřený průmyslový komunikační protokol využívaný k povelování zařízení prostřednictvím rozhraní Ethernet. ModbusTCP stejně jako jeho dvojče Modbus RTU využívá systém sdílené paměti v podřízeném zařízení (serveru), ke kterým přistupuje nadřazené zařízení (klient) prostřednictvím příkazů pro její čtení či zápis.

ModbusTCP využívá klasický TCP/IP stack, což s sebou přináší řadu výhod, a samozřejmě také určité nevýhody. Mezi základní výhody se řadí to, že ModbusTCP lze implementovat na jakémkoli zařízení podporujícím TCP/IP, tedy na klasických počítačích, různých embedded platformách a dalších bez nutnosti dalších modifikací. Nejvýznamnější nevýhodou je pak fakt, že prostřednictvím ModbusTCP nelze z podstaty realizovat časově deterministickou komunikaci. I přesto, že pro ModbusTCP neexistují požadavky na speciální TCP/IP stack a zároveň se na rozdíl od ostatních popsaných řešení se jedná o otevřený standard, nelze říci, že by tento protokol byl favoritem mezi výrobci. [2]

Jeho oblíbenost v průmyslové sféře mu však nahrává, a byl proto i výrobci robotů akceptován a bývá na přání podporován. Jednoduchá implementace bez nutnosti hardwarových doplňků rovněž zajišťuje poněkud nižší cenu za softwarový balík zahrnující ModbusTCP ovladač pro robot kontrolér a licenci, než je tomu například v případě rozhraní ProfiNET. [2]

Data přístupná prostřednictvím ModbusTCP protokolu jsou uložená v jednom ze čtyř paměťových bloků, které definují typ a přístupová práva pro Client a Server zařízení, jak je to vidět v tabulce 2.2.

Tabulka 2.2 ModbusTCP data model

Memory Block	**Data Type**	**Client Access**	**Server Access**
Coils	Boolean	Read/Write	Read/Write
Discrete Inputs	Boolean	Read-only	Read/Write
Holding Registers	Unsigned Word	Read/Write	Read/Write
Input Registers	Unsigned Word	Read-only	Read/Write

V rámci této práce bylo rozhodnuto použít komunikační protokol ModbusTCP, který je podporován v robotické jednotce Fanuc. Tento protokol sice není nativně podporován všemi výrobci, ale lze jej na požádání s nízkými náklady do jednotek

doplnit. Tento protokol je rovněž univerzální a je tedy vhodnou volbou pro naši konkrétní realizaci a použití.

2.3 ModbusTCP implementace na straně hlavního řídicího systému

V rámci této podkapitoly jsou popsané testovací aplikace a komunikační knihovna, která byla dodána a následně zabudována do hlavní řídicí jednotky a stará se o efektivní propojení s robotem.

2.3.1 Komunikační knihovna

V rámci této podkapitoly je popsána komunikační knihovna, která je založena na otevřeném protokolu ModbusTCP. Její implementace byla umístěná do samostatné softwarové knihovny, která navenek poskytuje pouze tzv. API rozhraní. Toto rozhraní je již navrženo tak, aby jeho použití bylo snadné. Objekty a funkce, které jsou v API rozhraní definovány, jsou vhodně pojmenovány, a tedy do vysoké míry samo se popisující a snadno použitelné.

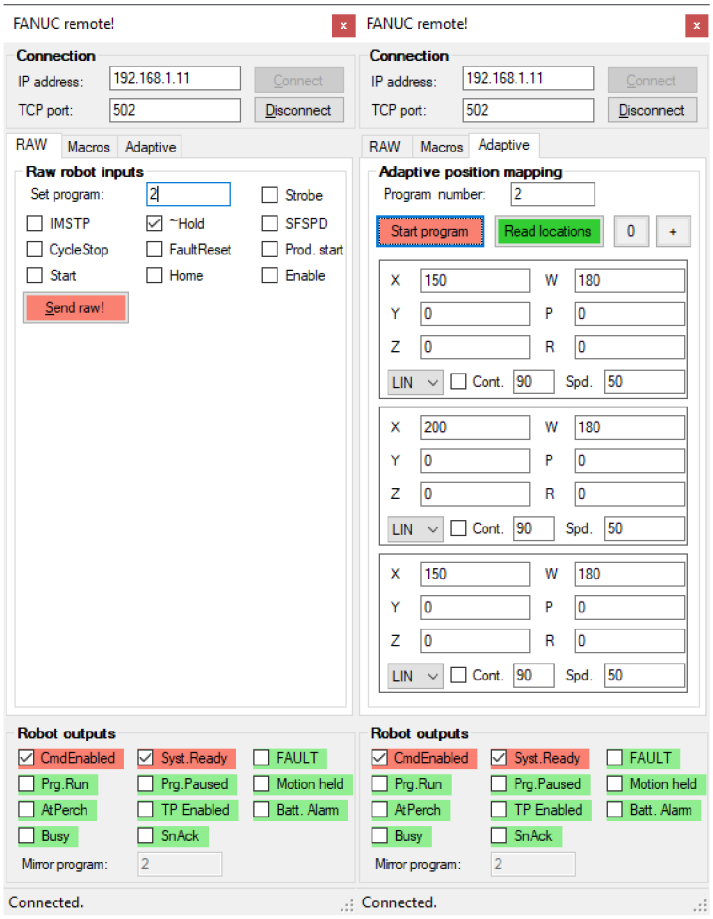
Knihovna byla navržena a implementována v prostředí .NET Framework a následně byla adaptovaná na platformu .NET CORE.

Knihovna definuje následující objekty:

- *InputDataModel* – jedná se o datový model, do kterého je možné nadefinovat všechny nutné parametry pro robota. Model implementuje také validaci hodnot a přepočet na Modbus registry.
- *OutputDataModel* – analogicky se jedná o datový model, z nějž je možné vyčíst všechny stavové informace, které robot poskytuje. Model opět implementuje přepočet specifických hodnot z Modbus registrů.
- *PointRegistr* – je datový model plně popisující jeden bod robotu. Tyto modely jsou využívány v rámci předchozích dvou objektů.
- *DcsBoundary* – je datový model, který umožňuje definovat hranici v prostoru, který bude v rámci knihovny brán jako robotem nepřekročitelný. V případě, že zadané hodnoty cesty robotu budou mimo tuto hranici, nebudou do robotu odeslány a dojde (v závislosti na nastavení) k vyvolání výjimky a ukončení zpracování příkazu, případně k ořezání hodnot tak, aby byly na hranici a jejich následné odeslání do robotu (hodí se zejména pro testovací účely, pro nasazení nebude využito).
- *FanucClient* – je vlastní kód knihovny, který implementuje funkce připojení k robotu, udržování spojení a přenos jednotlivých datových modelů mezi nadřazeným systémem a robotem. Kód rovněž implementuje některá důležitá makra, jako je např. spuštění programu, pro které je nutno vyslat postupně sekvenci příkazů. [2]

2.3.2 Testovací aplikace

V rámci této práce se používala dodána testovací desktop aplikace na testovaní a ladění komunikace s robotem Fanuc. Grafické rozhraní desktop aplikace lze vidět na obrázku 2.1. V této aplikaci je možné po připojení k robotu v reálném čase sledovat hodnoty jeho stavových bitů, a odesílat příkazy. Aplikace rovněž zjednodušuje testování náročnějších operací tím, že umožňuje konfigurovat a spouštět různá makra.



Obrázek 2.1 Screenshoty testovací aplikace

Aplikace po připojení automaticky udržuje spojení s robotickým kontrolérem, vyčítá a zrcadlí jeho stav (viz sekce *Robot outputs* na obrázku 2.1). Na záložce Raw robot inputs je možné odesílat do robotu jednoduché příkazy. Každou nastavenou kombinaci povelů je nutné společně odeslat tlačítkem Send raw. Záložka Macro pak obsahuje několik předdefinovaných komplexnějších příkazů, např. příkaz pro start programu, kterému musí předcházet uvedení robotu do korektního stavu, vymazání případného chybového příznaku apod. Poslední záložka – Adaptive position mapping pak umožňuje definovat několik bodů, které jsou vždy po stisku tlačítka Start program přehrány do robotu a je spuštěn pohyb robotu po trase definované právě těmito body. [2]

2.4 Komunikační protokol HTTP

Hypertext Transfer Protocol (HTTP), prostřednictvím kterého bude probíhat přenos dat mezi Raspberry Pi a nadřazeným systémem a taktéž mezi inteligentním majákem a nadřazeným systémem, je protokol aplikační vrstvy určený pro komunikaci s webovými servery. Protokol HTTP slouží pro přenos hypertextových dokumentů ve formátu XML, HTML a jiných typů souborů. Komunikační protokol HTTP předpokládá použití struktury přenosu dat klient-server. Klientská aplikace vytvoří požadavek, který následně odešle na server. Následně server požadavek zpracuje, vygeneruje odpověď a odešle ji zpátky klientovi. Poté klient může pokračovat v zasílaní dalších požadavků na server.

Samotný protokol HTTP neumožnuje šifrování nebo zabezpečení dat. K tomuto účelu se používá navíc kryptografický protokol prezentační vrstvy TLS (Transport Layer Security) nebo jeho předchůdce SSL (Secure Socket Layer). Tyto 2 protokoly spolu tvoří zabezpečený protokol HTTPS (Hypertext Transfer Protocol Secure). [3]

3. POPIS POUŽITÝCH HW PROSTŘEDKŮ

V této kapitole budou popsané hardwarové prostředky, který byly použité pro realizaci této práce, zejména robotická ruka Fanuc, hlavní řídicí jednotka celé robotické buňky Raspberry Pi 4 B, RFID čtečka karet MFRC522, snímač barvy TCS3200 a jiné. Na obrázku 3.1 lze vidět aktuální stav demonstrační robotické buňky.

Obrázek 3.1 Aktuální stav demonstrační robotické buňky

3.1 Kávovar

Pro vaření kávy uvnitř robotické buňky je umístěn kávovar Phillips Series 5000 EP5360/10, který lze vidět na obrázku 3.2 a). Kávovar umí připravit pět různých káv, jmenovitě: espresso, cappuccino, americano, latte Macchiato a pěněné mléko. K tomu je určeno pět tlačítek na předním krytu kávovaru, šesté tlačítko slouží pro vstup do menu a sedmé tlačítko uprostřed kávovaru slouží na jeho zapnutí/vypnutí. Kávovar má i displej jako uživatelské rozhraní, na kterém se ukazují informace o průběhu zahřívaní vody, vaření kávy a chyby, které mohou být uskutečněné přeplněním zásobníku na odpad v kávovaru nebo nedostatku vody či zrnkové kávy. Pro napájení kávovaru je potřeba 230 V AC napětí. [4]

3.2 Robot Fanuc

Uvnitř robotické buňky je umístěn robot Fanuc LR Mate 200iD/4S, který lze vidět na obrázku 3.2 b). Robot vykonává následující funkce: mačkaní tlačítek na kávovaru,

vybrání kelímků ze zásobníku a manipulace s kelímkem v různých prostorových bodech uvnitř buňky.

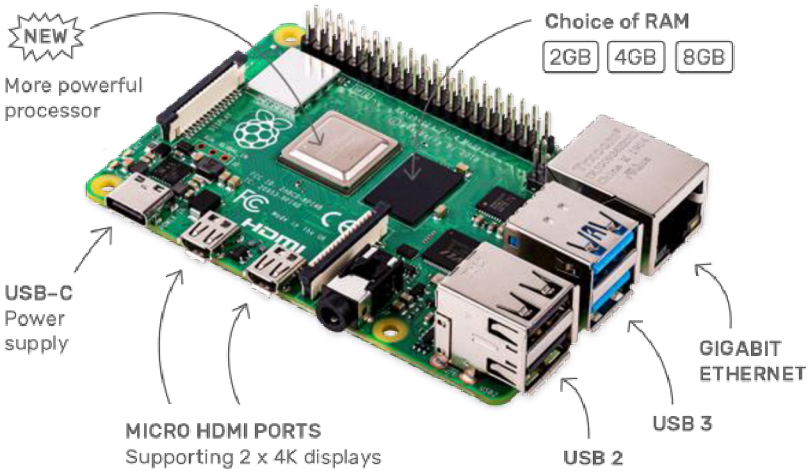
Robot Fanuc řady LR Mate 200iD/4S je kompaktní šestiosý robot s krátkým ramenem. Maximální dosah tohoto robota je 550 mm a maximální nosnost je 4 kg. K tomuto modelu robota Fanuc se nabízí i spektrům doplňků včetně integrované funkce inteligence, jako například vidění a snímání síly. Váha robota je 20 kg a je tak nejlehčí z řády LR Mate firmy Fanuc. [5]

a) b)

Obrázek 3.2 Kávovar Phillips Series 5000 EP5360/10 (a) [4] a Robotická ruka Fanuc LR Mate 200iD/4S (b) [5]

3.3 Hlavní řídicí jednotka Raspberry Pi 4B

Pro splnění požadavků této práce bylo nutné vybrat hlavní řídicí jednotku celé demonstrační robotické buňky. Jako hlavní řídicí jednotku celé buňky bylo rozhodnuto použít jednodeskový počítač Raspberry Pi od společnosti Raspberry Pi Foundation, který umožnuje tyto požadavky splnit po hardwarové a softwarové straně. Pro zobrazení uživatelského rozhrání a ovládání uživatelem celé robotické buňky bylo nutné také vybrat dotykový displej. Byl vybrán 7-palcový dotykový display s rozlišením 800x480 pixelů od stejné společnosti, který umožnuje požadavky práce splnit. Jako hlavní řídicí jednotka se vybral nejnovější model Raspberry Pi 4 B, který je ukázán na obrázku 3.3, z důvodu jeho nejlepších technických charakteristik ve srovnání s ostatními modely od společnosti Raspberry Pi Foundation.



Obrázek 3.3 Jednodeskový počítač Raspberry Pi 4 model B [6]

Raspberry Pi 4 B má procesor Broadcom BCM2711 ze řady čtyřjádrových 64 bitových procesorů Cortex-A72 (ARM v8) s frekvenci 1.5 GHz. Může obsahovat 2, 4 nebo 8 GB SDRAM. Z cenového hlediska byl vybrán model s 2 GB SDRAM. Raspberry Pi 4 B má celkově GPIO 40 pinů, na které je vyvedená například sériová komunikační rozhraní, jmenovitě UART, 2x I2C a 2x SPI. Raspberry Pi 4 B má Wi-Fi modul poskytující možnost komunikace na frekvencích 5.0 GHz a 2.4 GHz, Bluetooth modul (Bluetooth 5.0, BLE), Gigabit ethernet port, 2x USB porty 3.0, 2x USB porty 2.0, 2x micro-HDMI porty, display port, kamerový port a slot Micro SD pro načítání operačního systému a ukládání dat. Napájet tento model Raspberry Pi je možné přes USB-C konektor nebo přes k tomu dedikované piny. Minimální doporučené parametry napájecího zdroje je 5 V DC s maximálním výstupním proudem 3 A. [7]

3.4 RFID

Požadavkem práce bylo navrhnout a realizovat řešení pro možnost autentizace a autorizace uživatele prostřednictvím přístupové karty na použití demonstrační robotické buňky. Následně na základě tohoto řešení byly realizované různé statistické analýzy, jako například počet připravených káv za měsíc pro každého uživatele. Pro tento účel bylo rozhodnuto o použití RFID modulu na čtení Mifare karet. Mifare je ochranná známka společnosti NXP ze série čipů integrovaných obvodů používaných v bezkontaktních čipových kartách. V rámci testování jako Mifare karty se používali ISIC karty.

3.4.1 Princip RFID

RFID (Radio Frequency Identification – radiofrekvenční identifikace) je forma bezdrátové komunikace, která zahrnuje použití elektromagnetické nebo elektrostatické vazby v radiofrekvenční části elektromagnetického spektra a slouží k identifikaci objektu nebo osoby. RFID systém se skládá ze třech komponent: antény, transceiveru a

transpondéru. Pokud jsou anténa a transceiver kombinovány, označuji se jako RFID čtečka. Transpondér je elektronické zařízení, které po obdržení rádio-frekvenčního signálu pošle odvetný signál. Transpondér je umístěn v RFID štítku spolu s energetický nezávislou pamětí, do které lze data zapisovat a číst nebo pouze data číst. RFID čtečka je zařízení připojené k síti a využívá rádiové vlny k přenosu signálu, které aktivují štítek. Jakmile je štítek aktivován, tak pošle vlnu zpět do antény, kde se následně vlna konvertuje na použitelná data. [8]

Transpondéry se dělí do dvou základních skupin:

- *Aktivní* – mají vlastní zdroj energie a mohou přenášet data na větší vzdálenost
- *Pasivní* – získávají energii pro přenos dat z elektromagnetického pole čtecího zařízeni

RFID systémy lze rozdělit podle jejich pracovní frekvence, které ovlivňuje také i přenosový dosah komunikace mezi štítkem a čtečkou:

- *Nízkofrekvenční RFID systémy* – u těchto systému frekvenční pásmo je v rozmezí od 30 do 500 kHz. Typická pracovní frekvence je pro ně však 125 kHz.
- *Vysokofrekvenční RFID systémy* – u těchto systému frekvenční pásmo je v rozmezí od 3 do 30 MHz. Typická pracovní frekvence je pro ně však 13,56 MHz.
- *Ultra vysokofrekvenční RFID systémy* – u těchto systému frekvenční pásmo je v rozmezí od 300 do 960 MHz s typickou pracovní frekvencí 433 MHz.
- *Mikrovlnné RFID systémy* – pro tyto RFID systémy je typickou pracovní frekvenci 2,45 GHz. [8]

3.4.2 RFID MFRC522

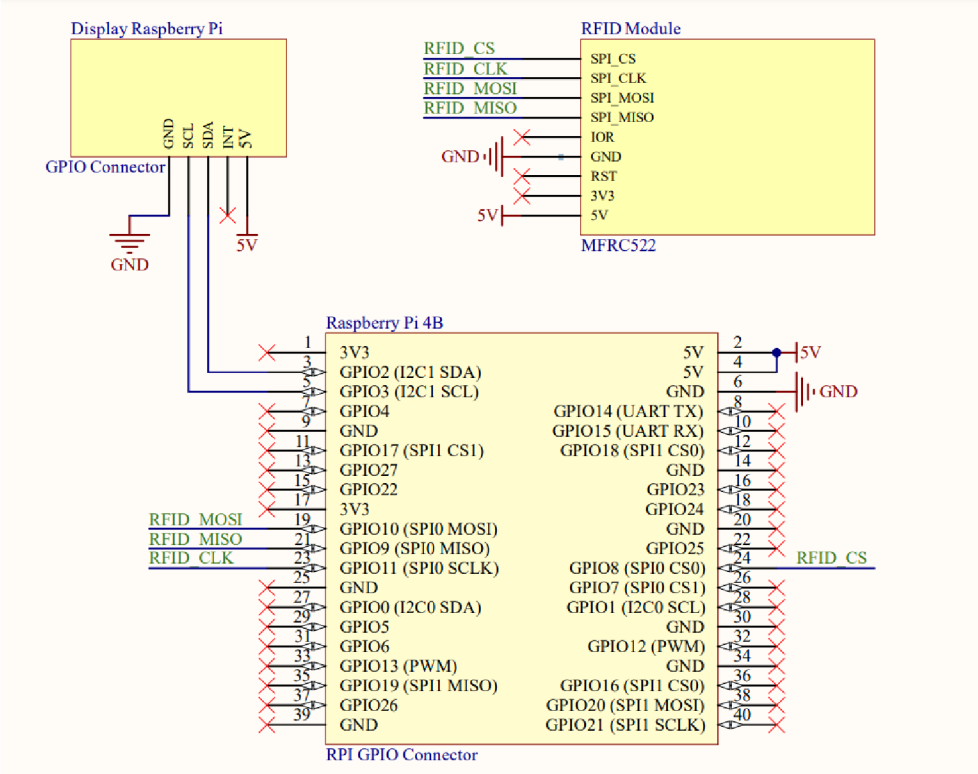
Pro splnění požadavku byla vybrána RFID čtečka karet MFRC522 s mikroprocesorem od společnosti NXP, kterou lze vidět na obrázku 3.4. RFID modul je zapojen do Raspberry Pi (obrázek 3.5) a spolu s displejem jsou umístěny v jedné navržené krabici na robotické buňce.

Obrázek 3.4 RFID modul MFRC522 [9]

MFRC522 je integrovaný obvod pro bezkontaktní čtení a zapisování informace pracující na frekvencí 13.56 MHz a využívá nejvyšší přenosové rychlosti Mifare karet, a to 848 kBd v oboru směrech. Čtečka MFRC522 podporuje ISO/IEC 14443A Mifare karty a NTAG. Napájet RFID modul je možný pomocí 3.3V nebo 5V. [10]

Pro komunikaci s jinými zařízení jsou k dispozici na modulu MFRC522 sériová komunikační rozhraní, jmenovitě UART, I2C a SPI. Přenosová rychlost dat RFID modulu pomocí komunikačního rozhraní SPI může dosahovat až 10 Mbit/s, I2C – až 3400 kBd, UART – 1228.8 kBd. Na základě této informace bylo rozhodnuto použít komunikační rozhraní SPI mezi Raspberry Pi a RFID modulem z důvodu nejvyšší rychlosti v porovnání s jinými komunikačními rozhraní. [10]

Schéma zapojení RFID modulu a displeje do Raspberry Pi lze vidět na obrázku 3.5.



Obrázek 3.5 Schéma zapojení RFID modulu a displeje do Raspberry Pi 4B

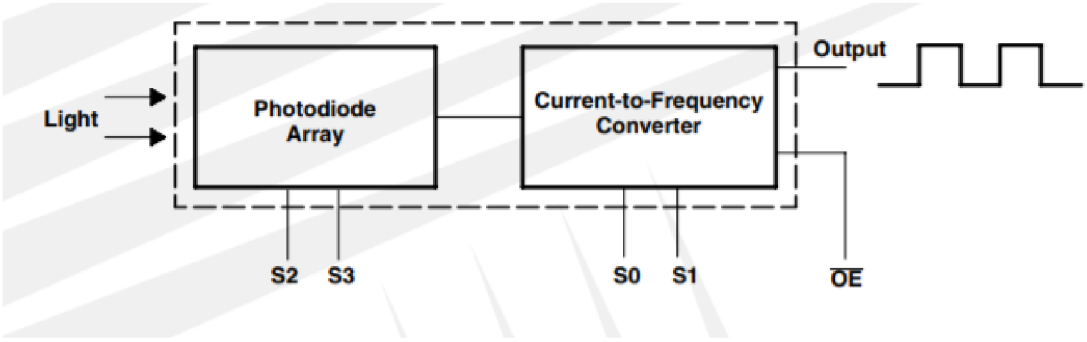
3.5 Snímač barvy TCS3200

Pro splnění požadavku detekce stavů kávovaru bylo rozhodnuto snímat barvu displeje kávovaru. Kávovar se do chybového stavu dostane v případě přeplnění zásobníku na odpad v kávovaru nebo nedostatku vody či zrnkové kávy na přípravu kávy. Ve funkčním stavu displej kávovaru svítí modře, v chybovém – červeně. Pokud kávovar je vypnutý, tak displej je černý. V případě, kdy kávovar je zapnutý, tak jeho displej může svítit dvěma barevnými stavy, které jsou základem barevného modelu RGB (Red, Green, Blue – červená, zelená, modrá). Barevnou detekci displeje kávovaru lze

realizovat pomocí kamery s následnými algoritmy počítačového vidění, ale z ekonomického hlediska a z hlediska jednoduchosti pozdější algoritmizace bylo rozhodnuto o použití detektoru barvy TCS3200, který lze vidět na obrázku 3.6.

Obrázek 3.6 Snímač barvy TCS3200 [11]

Snímač barvy TCS3200 převádí intenzitu světla na frekvenční signál. Základem jsou pole 8x8 křemíkových fotodiod a převodník proudu na frekvenci na jednom monolitickém CMOS integrovaném obvodu. Pole fotodiod obsahuje 16 fotodiod s modrými filtry, 16 fotodiod má zelené filtry, 16 fotodiod má červené filtry a ještě 16 fotodiod je bez filtrů. Výstupem ze snímače je obdélníkový signál se střídou 50 % a frekvencí přímo úměrnou intenzitě světla. Blokové schéma snímače barvy TCS3200 lze vidět na obrázku 3.7. [12]



Obrázek 3.7 Blokové schéma snímače barvy TCS3200 [12]

Pro napájení snímače je potřeba napětí v rozsahu od 2.7 V do 5.5 V. Na VCC pin snímače je přivedeno 3.3 V. Maximální napájecí proud je 2 mA. [12]

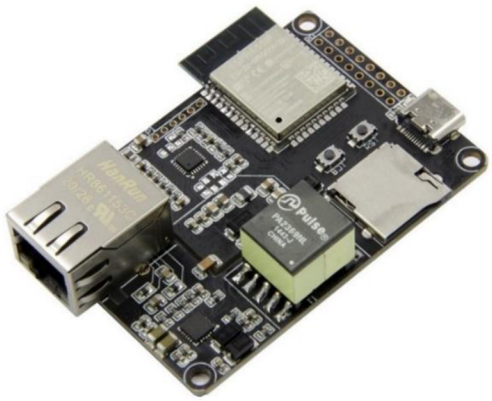
Výstupní frekvencí signálu ze snímače na pinu OUT lze nastavovat jako jednu ze tří přednastavených hodnot pomocí dvou ovládacích vstupních pinů S0 a S1. V případě nastavení pinů S0 a S1 v jedničce – výstupní frekvence signálu ze snímače bude typický 600 kHz. V případě nastavení pinu S0 v jedničce a pinu S1 v nule – výstupní frekvence signálu ze snímače bude typický 120 kHz. V případě nastavení pinu S0 v nule a pinu S1 v jedničce – výstupní frekvence signálu ze snímače bude typický 12 kHz. [12]

Piny S2 a S3 slouží k nastavování aktivní skupiny fotodiod (červena, zelená, modrá a bez filtrů). V případě nastavení pinů S2 a S3 v nule bude aktivní skupina fotodiod s červeným filtrem. V případě nastavení pinů S2 a S3 v jedničce bude aktivní skupina fotodiod se zeleným filtrem. V případě nastavení pinu S2 v nule a S3 v jedničce bude aktivní skupina fotodiod s modrým filtrem. V případě nastavení pinu S2 v jedničce a S3 v nule bude aktivní skupina fotodiod bez filtru. [12]

Pin LED slouží k ovládání přisvěcovacích led diod. Přisvěcovací diody byly však odpájený z konstrukčních důvodů, jelikož výška snímače měla být co nejmenší, aby snímač na displeji kávovaru nezasahoval do robotické ruky.

3.6 Řídicí jednotka T-ETH-POE ESP32-WROOM

V rámci této práce byl kladen požadavek na realizaci snímaní barvy displeje kávovaru jako modulární řešení v rámci celé demonstrační robotické buňky. Proto bylo nutné vybrat další řídicí jednotku, která by umožnila připojení snímače barvy TCS3200 a zároveň by umožnila komunikace s hlavní řídicí jednotkou Raspberry Pi 4 B prostřednictvím už navrženého komunikačního protokolu ModbusTCP. Bylo rozhodnuto jako řídicí jednotku použít desku T-ETH-POE ESP32-WROOM od společnosti LilyGO, která tyto požadavky plní. Desku T-ETH-POE ESP32-WROOM lze vidět na obrázku 3.8.



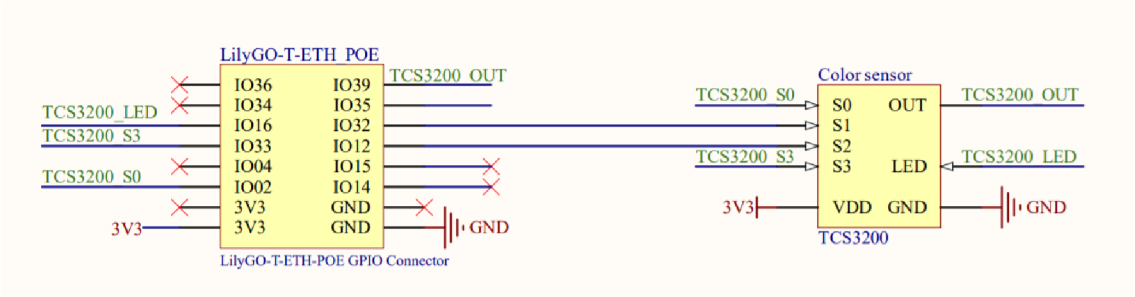
Obrázek 3.8 T-ETH-POE ESP32-WROOM [13]

Základem desky T-ETH-POE ESP32-WROOM je mikrokontroler ESP32-WROOM od společnosti Espressif a Ethernet čip LAN8720A, který umožňuje realizovat komunikace prostřednictvím komunikačního protokolu ModbusTCP s hlavní řídicí jednotkou celé robotické buňky Raspberry Pi 4 B. [13]

Výhodou této desky je také to, že umožňuje její napájení pomocí Ethernet (PoE), ale i pomocí Type-C USB 5 V. Pracovní napětí mikrokontroleru ESP32-WROOM je 2.7–3.6 V. Deska obsahuje slot pro SD kartu. Mikrokontroler ESP32-WROOM, který je použít na této desce má 4 MB FLASH pamětí a 520 kB SRAM pamětí. V sobě má také

Wi-Fi a Bluetooth čipy. Jelikož řídicím čipem desky T-ETH-POE ESP32-WROOM je čip ESP32-WROOM tak deska umožnuje používaní sériových komunikační rozhraní, jmenovitě UART, SPI, I2C, I2S. Pro nahrávání firmwaru do mikrokontroleru ESP32 byl použít USB TTL převodník CP2102. [13]

Na obrázku 3.9 lze vidět schéma zapojení detektoru barvy TCS3200 a řídicí desky T-ETH-POE ESP32-WROOM. Pro snímač barvy a řídicí desku byly speciálně navrženy a vytvořeny zapouzdření (pomocí 3D tisku), které byly následně uchycený na kávovaru.

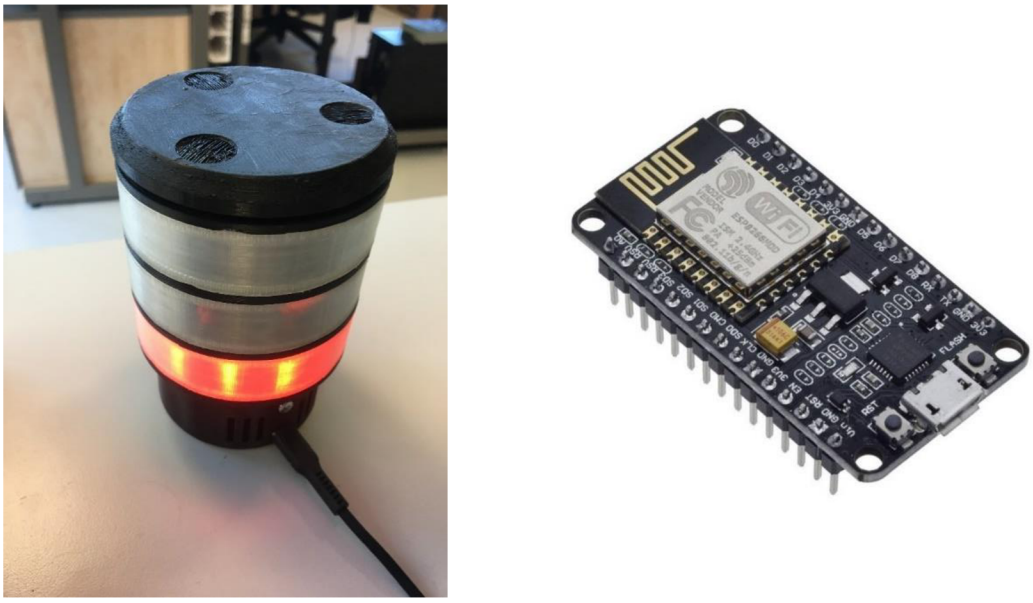


Obrázek 3.9 Schéma zapojení detektoru barvy TCS3200 do T-ETH-POE ESP32-WROOM

3.7 Inteligentní maják

Pro splnění požadavku této práce na světelnou indikaci procesu přípravy kávy v demonstrační robotické buňce byl navržen a realizován inteligentní maják. Požadavkem práce bylo realizovat inteligentní maják jako modulární prvek, aby jej bylo možné použít i v rámci jiných projektů. Cílem bylo realizovat maják s třemi patry LED, který by bylo možné ovládat nadřazeným systémem přes Wi-Fi prostřednictvím HTTP/HTTPS protokolu. V této podkapitole bude popsán inteligentní maják po jeho hardwarové stránce. Po softwarové stránce je inteligentní maják popsán v podkapitole 4.6.

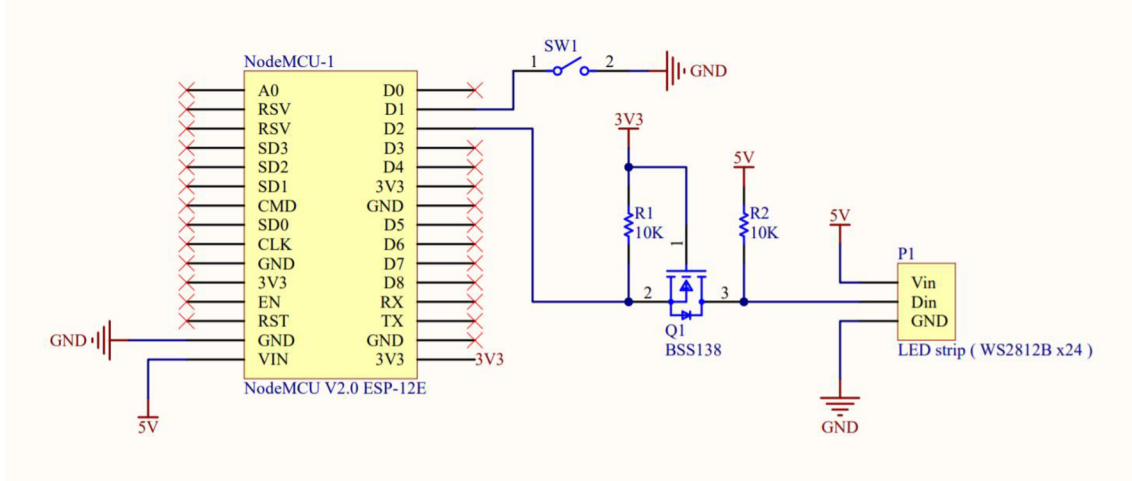
Krýt majáku byl realizován pomocí 3D tisku, kde se využili 2 typy PLA plastu – černý a průhledný. Zrealizovaný inteligentní maják lze vidět na obrázku 3.10 a). Pro realizaci inteligentního majáku bylo rozhodnuto jako řídicí jednotku použít desku ESP8266 NodeMCU V2, kterou lze vidět na obrázku 3.10 b).



a) b)

Obrázek 3.10 Zrealizovaný inteligentní maják (a) a ESP8266 NodeMCU V2 (b) [14]

NodeMCU V2 je založen na mikrokontroleru ESP8266 od společnosti Espressif. Řídicí deska NodeMCU V2 obsahuje čip CP2102, který je spojen s mikrokontrolerem ESP8266. Mikrokontroler ESP8266 má vestavěný bootloader, což umožnuje nahrávat program do čipu z počítače pomocí MicroUSB kabelu. Provozní napětí mikrokontroleru je 2.5–3.6 V. Pro napájení NodeMCU je potřeba 5–12 V DC a může se napájet pomocí MicroUSB konektoru nebo Vin a GND piny. ESP8266 má 1 analogový vstupní pin,11 digitálních I/O pinů a nemá žádný analogový výstup. ESP8266 má v sobě Wi-Fi čip a má taky několik sériových komunikačních rozhraní, jmenovitě 2x SPI, 1x I2C, 2x I2S, 2x UART. Mikrokontroler ESP8266, který je základem této desky má 4 MBFLASH pamětí a 64 kB SRAM pamětí. [15] [16]



Obrázek 3.11 Elektrické schéma zapojení inteligentního majáku

Pro realizaci inteligentního majáku bylo rozhodnuto využít adresovatelný LED pásek, který je sestaven z 24 LED s čipem WS2812B (pro každé „patro“ majáku 8 LED diod). NodeMCU V2 na svých výstupních pinech dává 3.3 V logického napětí. Čip WS2812B má 5 voltovou logiku. Proto pro synchronizaci logické úrovně byl navržen převodník logické úrovně na základě MOSFET tranzistoru BSS138. Bylo taktéž nutné přidat tlačítko pro přepínaní softwarových modů. Elektrické schéma zapojení majáku lze vidět na obrázku 3.11.

4. SOFTWAROVÁ REALIZACE

V této kapitole bude popsaná softwarová realizace řídicí aplikace v Raspberry Pi, softwarová implementace v řídicí desce T-ETH-POE ESP32-WROOM a softwarová implementace inteligentního majáku.

Požadavkem této práce bylo realizovat řídicí aplikaci v Raspberry Pi pomocí cross-platformy .NET a programovacího jazyka C# od společnosti Microsoft. V této kapitole budou také popsány programovací technologie, pomocí kterých byla řídicí aplikace realizována.

4.1 Platforma .NET

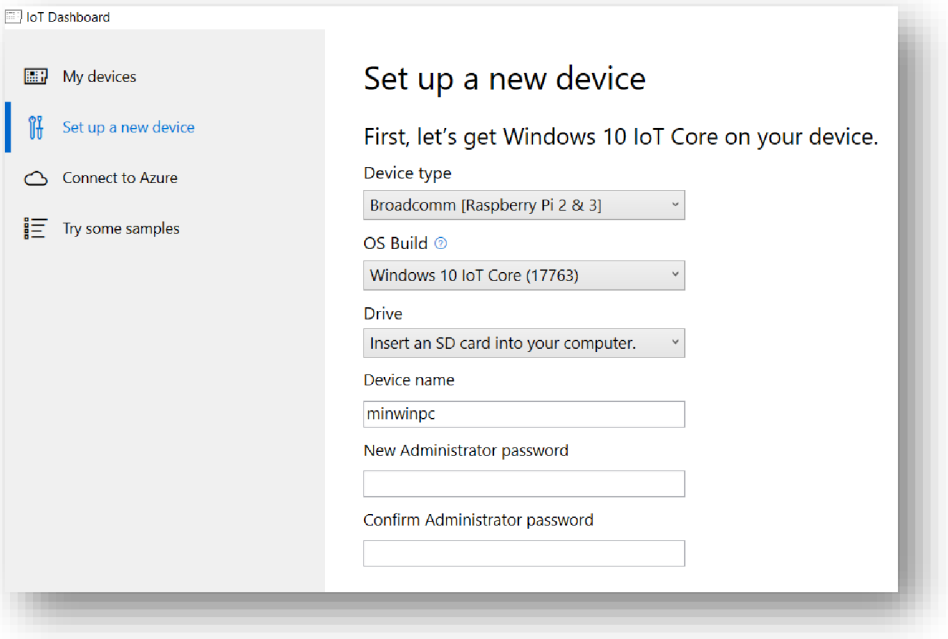
Platforma .NET je vývojová platforma s otevřeným zdrojovým kódem a byla vyvinutá na základě platformy .NET Framework, která na rozdíl od .NET může běžet pouze na OS Windows. Platforma .NET v sobě sjednotila .NET Core, Mono, .NET Framework a prezentuje se jako multiplatformní a modulární platforma. Modularitu platformě .NET zabezpečuje správce balíčků NuGet. Balíček NuGet je soubor *.zip* s *.nupkg* příponou, který obsahuje zkompilovaný kód v *DLL* formátu a jiné soubory související s kódem. [17]

Platforma .NET umožnuje vytváření mnoha druhů aplikací, jako například: webové aplikace, desktop aplikace, mobilní aplikace, cloudové aplikace, IoT aplikace a jiné. Platforma .NET umožnuje vytvářet aplikace pro různé operační systémy jako například: Windows, macOS, Linux, Android, IOS a jiné. Podporuje procesorové architektury jako x64, x86, ARM32 a ARM64. Integrovaná vývojová prostředí, které umožnují použití .NET platformy jsou Visual Studio, Visual Studio for Mac a Visual Studio Code. Platforma .NET také podporuje 3 programovacích jazyky: C#, F# a Visual Basic. Programovací jazyk C# je objektově orientovaný a bezpečný jazyk mající své kořeny v rodině jazyků C. V rámci této práce se použila verze .NET 5 jako aktuální verze od společnosti Microsoft. [17]

4.2 Výběr operačního systému pro Raspberry Pi 4 B

Jelikož platforma .NET je vyvinuta společnosti Microsoft tak bylo rozhodnuto o vyzkoušení operačního systém Windows 10 IoT Core pro Raspberry Pi 4 B. Windows 10 IoT Core je operační systém od společnosti Microsoft, který je optimalizován pro výkonově menší zařízení a je určen pro vestavěné systémy běžící na ARM nebo x86/x64 procesorech. [18]

V dalším kroku bylo nutné podle návodu provést stahování a nastavení Windows 10 IoT Core na Raspberry Pi a spojit se s počítačem, k čemuž je určena aplikace Windows 10 IoT Core Dashboard od Microsoftu. [19]



Obrázek 4.1 IoT Dashboard

Bylo zjištěno, že společnost Microsoft oficiálně nepodporuje Windows 10 IoT Core pro Raspberry Pi 4 B a po instalaci Windows 10 IoT Core na Raspberry Pi 4 B operační systém nefungoval. K dispozicí na vyzkoušení operačního systému byly také Raspberry Pi 3 B a Raspberry Pi 3 B+. V dalším kroku bylo zjištěno, že Microsoft také oficiálně nepodporuje Windows 10 IoT Core pro Raspberry Pi 3 B+. Jednu z verzí Windows 10 IoT Core, jmenovitě 17661, bylo možné úspěšně nainstalovat na Raspberry Pi 3 B+. Operační systém se spustil korektně, avšak následně na zařízení nefungovalo Bluetooth, Wi-Fi ale především touchscreen displej, který je důležitou komponentou pro tuto práci. [20] [21]

V následujícím kroku pomocí IoT Dashboard byla provedena instalace Windows 10 IoT Core na Raspberry Pi 3 B, na kterém operační systém už běžel bez jakýchkoliv problémů.

Na základě toho, že Windows 10 IoT Core se nedá použít pro novější generace Raspberry Pi bylo rozhodnuto se neomezovat na starší verzi jednodeskového počítače z hlediska výkonu a použít jiný operační systém, jmenovitě 32-bitovou Raspberry Pi OS, což je oficiální operační systém od společnosti Raspberry Pi Foundation založený na Debianu, který představuje jednu z distribucí Linuxu.

Po instalaci operačního systému Raspberry Pi OS pomocí následujícího příkazu v terminálu byl spuštěn skript, pomocí kterého platforma .NET byla nainstalovaná v Raspberry Pi: [22]

```
wget -O -
https://raw.githubusercontent.com/pjgpetecodes/dotnet5pi/master/instal
l.sh | sudo bash
```

4.3 Výběr vhodného GUI frameworku pro Raspberry Pi 4 B

V rámci této podkapitoly budou popsané GUI (Graphic User Interface – Grafické uživatelské rozhraní) frameworky, které je možné použít s platformou .NET mezi kterými je i framework Avalonia, který byl vybrán pro realizaci této práce.

4.3.1 Dostupná řešení od Microsoftu

Společnost Microsoft nabízí několik možností pro realizaci desktopových aplikací, zejména:

- *Windows Presentation Foundation (WPF)* - UI framework pro vytváření desktopových aplikaci pro Windows na platformě .NET.
- *Window Forms (WinForms)* – UI framework se snadno používanými moduly, které se přetahuji do pracovního okna aplikace. Je určený pro vytváření desktopových aplikaci pro Windows na platformě .NET.
- *Universal Windows Platform (UWP)* – umožnuje vývoj desktopových aplikací pro jakékoliv zařízení s Windows 10 a Windows 10 IoT Core na platformě .NET.
- *Xamarin* – slouží na vytváření vizuálních aplikací pro macOS, iOS, Android a Windows na platformě .NET. [23]

Žádný s těchto frameworku od společnosti Microsoft není určen pro běh desktopové aplikaci na Linuxu. Proto bylo nutné hledat jiný framework, který umožní běh desktopové aplikace na linuxovém Raspberry Pi OS a bude kompatibilní s platformou .NET.

4.3.2 Multiplatformní GUI frameworky

V této podkapitole jsou krátce popsané multiplatformní GUI frameworky, které se uvažovali pro realizace této práce a zdůvodněn výběr použitého frameworku.

- *Qml.NET* – UI framework, který je integrovaný s platformou .NET a může běžet na Linuxu, Windows a OSX. Pro vytváření uživatelského rozhrání slouží jazyky QML a C#. [24]
- *UNO Platform* – UI framework na vytváření webových a desktopových aplikací pro Windows, WebAssembly, iOS, macOS, Android a Linux na platformě .NET. Pro vytváření uživatelského rozhrání slouží jazyky C# a XAML. [25]
- *Avalonia* – UI framework, který umožnuje vytváření desktopových aplikací pro Windows, macOS, Linux, Raspberry Pi OS, iOS a Android na platformě .NET. Pro vytváření uživatelského rozhrání slouží jazyky C# a XAML. [26]

Bylo zamítnuto používat framework Qml.NET a to z důvodu jeho horší dokumentace v porovnání s jinými frameworky. Důvodem zamítnutí bylo také i to, že pro vytváření uživatelského rozhrání má sloužit jazyk QML a ne XAML. Snahou

bylo držet se podobných frameworků, které nabízí společnost Microsoft, u kterých se používá právě jazyk XAML. Při rozhodování mezi frameworky UNO Platform a Avalonia bylo rozhodnuto použít framework Avalonia a to z toho důvodu, že v dokumentaci frameworku byla uvedena možnost jeho použití pro Raspberry Pi OS.

4.3.3 Framework Avalonia

Avalonia je multiplatformní framework s otevřeným zdrojovým kódem pro .NET platformu. Avalonia má podporu Visual Studio pro vytváření souborů XAML a má také vizuální náhled ve Visual Studio. Je potřeba podotknout, že Avalonia není hotový produkt a je stále ve vývoji. Níže jsou uvedené některé z výhod tohoto frameworku:

- Avalonia je framework s otevřeným zdrojovým kódem a je to multiplatformní framework.
- Avalonia je napsána v .NET Standard a je tak 100% kompatibilní se všemi verzemi .NET Core a .NET Framework.
- Avalonia následuje balíky od Microsoftu (WPF a Silverlight) s jejich paradigma, jako například: vizuální a logické stromy, styly, ovládací šablony, šablony dat, vazby a související vzor MVVM (Model-View-ViewModel) a jiné.
- Avalonia má na rozdíl od JavaScriptu plnou výhodu silně typovaného jazyka a také výhodu jazyka kompilovaného do binárního kódu. [26]

Mezi oficiálně podporovanými OS platformy patří:

- Windows 8 a vyšší
- MacOS High Sierra 10.13 a vyšší
- Debian 9 a vyšší
- Ubuntu 16.5 a vyšší
- Fedora 30 a vyšší [27]

4.3.4 Programovací jazyk XAML

Programovací jazyk XAML (Extensible Application Markup Language) je deklarativní jazyk založený na jazyce XML a byl vyvinutý společnosti Microsoft. V programovacím modelu .NET jazyk XAML zjednodušuje vytváření uživatelského rozhraní pro aplikaci .NET. Je možné vytvářet grafické prvky uživatelského rozhraní v deklarativním kódu XAML a následně oddělit definici uživatelského rozhraní od běhové logiky programu pomocí souborů kódu na pozadí, které jsou připojeny ke značkám prostřednictvím definic dílčí třídy. Jazyk XAML umožnuje pracovní postup, ve kterém můžou jednotlivé strany samostatně pracovat na uživatelském rozhraní a logice aplikace nezávisle na sobě. [28]

4.4 Implementace řídicího programu v Raspberry Pi

V této kapitole bude popsána navržena aplikace pro řízení robotické buňky pro přípravu kávy. Aplikace byla realizována pomocí platformy .NET 5 a běží na hlavní řídicí jednotce robotické buňky – Raspberry Pi 4 B.

V rámci realizace aplikace byla využita řada NuGet balíčků (knihoven), které byly do projektu nainstalované prostřednictvím nástroje "NuGet Package Manager" (Manažer Nuget balíčků) a pomohli v realizaci projektu. Jedním z důležitých NuGet balíčků je balíček *Avalonia*, umožňující realizaci uživatelského rozhrání aplikace. V rámci této práce se použila verze 0.10.3, jako nejstabilnější verze v momentu implementace.

Další důležitou knihovnou je knihovna *EasyModbusTCP*, která umožnila komunikace prostřednictvím protokolu ModbusTCP na platformě .NET 5. V rámci této práce se použila její nejaktuálnější verze 5.6.0. Dalším použitým NuGet balíčkem je balík *Iot.Device.Bindings*, který poskytuje sadu vazeb IoT zařízení, které ke komunikaci s mikrokontrolerem používají standardní knihovnu *System.Device.Gpio.* V této práci byla použita verze 1.5.0 knihovny *Iot.Device.Bindings* jako nejaktuálnější v době realizace projektu a byla využita pro komunikaci jednodeskového počítače Raspberry Pi s RFID čtečkou karet MFRC522.

4.4.1 Popis programů pro robota Fanuc

V rámci předchozí bakalářské práce [1] byly naprogramovány jednotlivé pohyby robota Fanuc, ze kterých se složili programy na objednání určitých typů káv. V rámci této práce došlo ke změnám v návrhu vaření kávy, proto jednotlivé pohyby robota byly změněny a celkový počet programů je stanoven na 11ti. Tyto programy jsou pomocí dodané Modbus knihovny spouštěny z hlavní řídicí jednotky Raspberry Pi. Programy musejí být spouštěny v určitém pořadí, aby nedošlo k náhodné kolizi uvnitř robotické buňky.

V předchozí práci byla realizována detekce otevřených/zavřených dvířek na výdej kávy uživateli a detekce přítomnosti hrnku/kelímku v gripperu robota Fanuc pomocí koncových spínačů, které jsou zapojené ke kontroléru robota. Bylo rozhodnuto u tohoto návrhu zůstat a vyčítat stavy spínačů prostřednictvím Modbus TCP protokolu.

Následně zde tyto programy budou krátce popsané:

1) GET_STATE – tento program je určený na vyčtení stavů koncových spínačů, které jsou zapojené do kontroléru robota. V případě, že jsou dvířka robotické buňky otevřené a zároveň kelímek není v robotu, tak program vrátí 0. V případě, že dvířka robotické buňky jsou zavřené a zároveň kelímek není v robotu, tak program vrátí 1. V případě, že dvířka robotické buňky jsou otevřené a zároveň kelímek je v robotu, tak program vrátí 2. V případě, že dvířka robotické buňky jsou zavřené a zároveň kelímek je v robotu, tak program vrátí 3.

2) TAKE_CUP – tento program je určený k tomu, aby robot najel ke zásobníku kelímků a vzal si prázdný kelímek ze zásobníku.
3) TURN_ON_COFFEE_MACHINE – tento program je určený k tomu, aby robot zmačknul tlačítko na kávovaru na jeho zapnutí.
4) ESPRESSO – tento program je určený k tomu, aby robot zmačknul tlačítko na kávovaru pro přípravu espressa.
5) AMERICANO – tento program je určený k tomu, aby robot zmačknul tlačítko na kávovaru pro přípravu americana.
6) CAPPUCCINO – tento program je určený k tomu, aby robot zmačknul tlačítko na kávovaru pro přípravu cappuccina.
7) LATTEMACCHIATO – tento program je určený k tomu, aby robot zmačknul tlačítko na kávovaru pro přípravu lattemacchiata.
8) BABYCCINO – tento program je určený k tomu, aby robot zmačknul tlačítko na kávovaru na nalívaní mlíka.
9) GOTO_COFFEE_MACHINE – tento program je určený k tomu, aby robot najel s prázdným kelímkem pod kávovar a počkal si určitou dobu až se nalije káva do kelímku.
10) DISPENSING_POINT – tento program je určený k tomu, aby robot najel s kelímkem kávy k výdejnímu místu.
11) ZERO_POSITION – tento program je určený k tomu, aby se robot vrátil na svoji inicializační pozici.

4.4.2 Implementace algoritmu pro komunikace s RFID čtečkou

V rámci této práce pro splnění požadavku autentizace a autorizace uživatelů byl realizován algoritmus pro komunikaci hlavní řídicí jednotky Raspberry Pi s RFID čtečkou MFRC522. Byla realizovaná třída *RFID*, která obsahuje metodu *Initialize*, ve které proběhne nastavení a inicializace komunikace Raspberry Pi – MFRC522 prostřednictvím komunikačního rozhrání SPI. Následně byla realizovaná metoda *cardRead,* ve které se bude čekat až RFID čtečka nedetekuje Mifare kartu ISO 14443 typ A. Ve chvíli, kdy čtečkou bude detekována Mifare karta, tak se vyčte prvních 16 bytů z pamětí Mifare karty, ve kterých je uložená identifikační informace karty každého uživatele. Následně se uloží informace o tom, že Mifare karta byla přečtená.

```
private void cardRead()
{
    while (!wait4Close)
    {
        Var result = mfrc522.ListenToCardIso14443TypeA(
                  out Data106kbpsTypeA card, TimeSpan.FromSeconds(2));

        if (result)
        {
            mifare = new MifareCard(mfrc522, 0)
            {
```

```
                SerialNumber = card.NfcId,
                Capacity = MifareCardCapacity.Mifare1K,
                KeyB = MifareCard.DefaultKeyB.ToArray()
            };

            mifare.BlockNumber = 0;
            mifare.Command = MifareCardCommand.AuthenticationB;
            ret = mifare.RunMifareCardCommand();

            if (ret >= 0)
            {
                mifare.Command = MifareCardCommand.Read16Bytes;
                ret = mifare.RunMifareCardCommand();
            }

            var cardId = BitConverter.ToString(mifare.Data);
            CardReadFinished?.Invoke(cardId);
        }
    }
}
```

4.4.3 Implementace algoritmu pro komunikace s ESP32-WROOM

V rámci této práce byl realizován řídicí algoritmus pro komunikaci mezi hlavní řídicí jednotkou Raspberry Pi a řídicí jednotkou T-ETH-POE ESP32-WROOM prostřednictvím komunikačního protokolu ModbusTCP s cílem vyčtení informace o barvě displeje kávovaru. V hlavní řídicí jednotce byl realizován ModbusTCP klient, v řídicí jednotce T-ETH-POE ESP32-WROOM byl realizován ModbusTCP server, který je detailně popsán v podkapitole 4.5.

V programu byla realizována třída *ColorDetectionModbus*, která dědí základní třídu z dodané ModbusTCP knihovny *BaseModbusClient*. V této třídě byla především realizována metoda *colorDetectionFunction*, ve které se na začátku ověří, zdali je navázáno spojení s příslušným ModbusTCP serverem. Pokud spojení navázáno není, tak se ho klient pokusí navázat. Následně je jednobitový registr (coil) nastaven v jedničku, což pro server znamená, že musí vyčíst data ze snímače barvy. Ve chvíli, kdy je tento jednobitový registr v 0, tak měření proběhlo korektně a do 16bitového registru byla uložená hodnota, která odpovídá barvě displeje kávovaru a je možné si ji přečíst.

```
private void colorDetectionFunction()
{
    while (!wait4Stop)
    {
        try
        {
            if (GetState() == EState.DISCONNECTED)
            {
                Disconnect();
                Connect(ipAddress);
            }

            TransmitStateCoil(COLOR_DETECTION_COIL, true);
```

```
            if (!ReceiveStateCoil(COLOR_DETECTION_COIL)[0])
            {
                var value =
              ReceiveStateRegister(COLOR_DETECTION_VALUE_REGISTER)[0];
                ColorDetectionFinished?.Invoke(value);
            }
            Thread.Sleep(1000);
        }
        catch (Exception ex)
        {
            System.Console.WriteLine(ex.Message);
        }
    }
    Disconnect();
}
```

V případě, že vyčtená hodnota se bude rovnat 0, tak to znamená, že nastala neznáma chyba měření snímačem barvy a chyba pravděpodobně bude ve programovacím algoritmu na straně ESP32-WROOM. Pokud se vyčtená hodnota rovná 1, tak displej kávovaru svití modře a kávovar je přípraven na vaření. Pokud se vyčtená hodnota rovná 2, tak displej kávovaru svití červeně – chyba kávovaru. Pokud se vyčtená hodnota rovná 3, tak displej kávovaru nesvití – vypnutý kávovar.

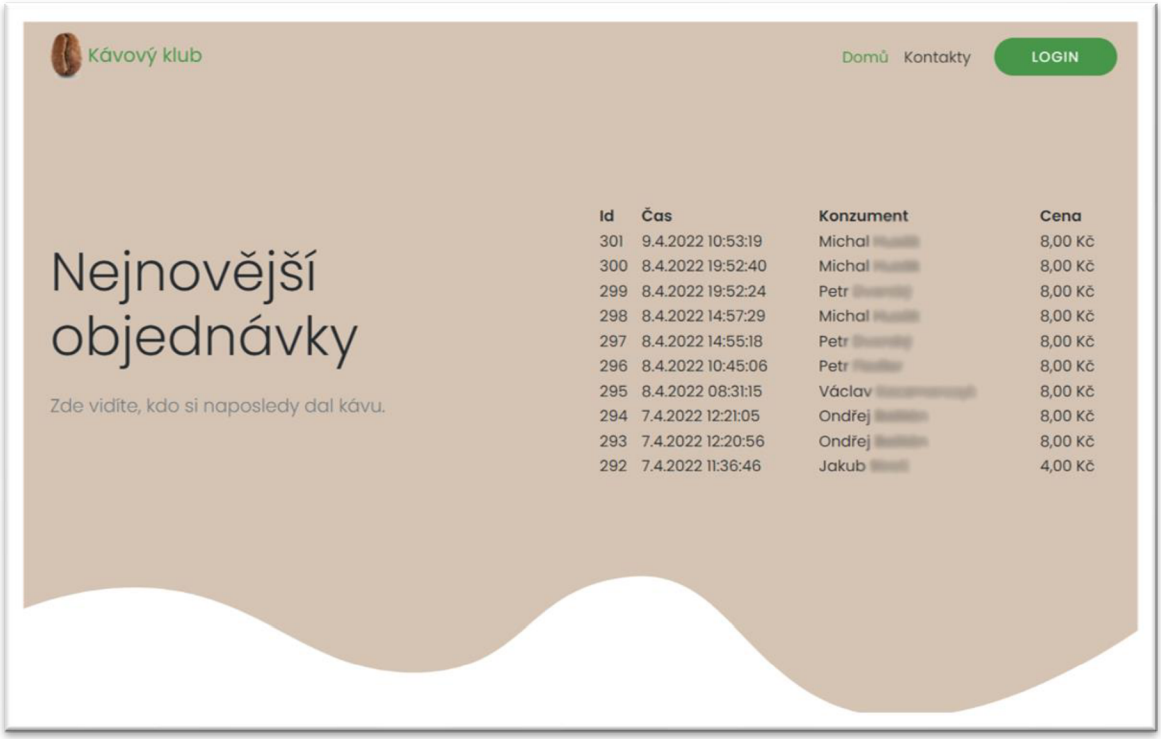
4.4.4 Implementace algoritmu pro komunikace s nadřazeným systémem

V rámci této práce na straně hlavní řídicí jednotky Raspberry Pi byla realizována komunikace s nadřazeným systémem (web serverem) prostřednictvím HTTP protokolu. Cílem bylo realizovat přijímání a přenos dat do nadřazeného systému, kde se ukládají data o tom, jestli uživatel má přístup ke kávovaru a statistická data každého uživatele, který má přístup k vaření kávy. Jako potvrzení funkčnosti realizované komunikace na obrázku 4.2 lze vidět webovou stránku, kde se zobrazuje aktuální informace o použití robotické buňky na vaření kávy.

Pro splnění požadavku byly realizovaný tři HTTP klienti, kde každý klient je implementován ve vlastní třídě, které jsou popsány níže:

1) *DoYouKnowClient* – v této třídě byla realizována metoda *Get*, která slouží k vyčtení informace z XML souboru uloženého na serveru. V metodě je definován objekt třídy *HttpClient*, který na základě známého URL odešle požadavek o této informaci na server a následně zpracuje odpověď. Metoda vrací řetězec s informaci, která je potom zobrazována v uživatelském rozhrání robotické buňky.
2) *StatisticsClient* – pro každého uživatele na serveru je vytvořen příslušný XML soubor, ve kterém se ukládá informace o uživateli, konto jeho karty, statistická data o počtu vypitých káv, čas objednání kávy atd. V této třídě byla realizována metoda *Get* s parametrem *cardId*. V metodě je definován objekt třídy *HttpClient*, který vyčte tuto informaci ze serveru na základě příslušného URL podle cardId uživatele. Metoda vrací instanci třídy *UserInformation*, ve které budou tyto data uložené.

3) *CoffeeClient* – v této třídě byla realizována metoda *Get*, ve které je na začátku definován objekt třídy *HttpClient*, který následně na základě cardId uživatele odešle informace na server o objednané kávě nebo nabití konta karty. Metoda vrací instanci třídy *UserInformation.*



Obrázek 4.2 Webová stránka nadřazeného systému

4.4.5 Popis hlavního řídicího algoritmu

V rámci této práce hlavní řídicí algoritmus se vyvíjel postupně. Proto bylo realizováno celkem 3 řídicích algoritmů, kde jeden z nich je finální. Další dva algoritmy umožnovali testování průběžně navržené aplikace. Každý z těchto řídicích algoritmů byl realizován jako stavový automat a v případě potřeby v souboru *App.axaml.cs* lze vybrat, který stavový automat se bude spouštět. V rámci této podkapitoly bude detailně popsán finální řídicí algoritmus aplikace.

```
//Zde si vybirame, ktery stavovy automat spustime.
//Program.StateMachine = new SimpleDemonstrationStateMachine();
//Program.StateMachine = new SimpleRfidStateMachine();
Program.StateMachine = new ModbusRfidStateMachine();
Program.StateMachine.Run();
```

První stavový automat byl realizován ve třídě *SimpleDemonstrationStateMachine.* Stavový automat obsahuje jenom 4 stavy a cílem tohoto stavového automatu bylo realizovat a otestovat aplikaci s jednou obrazovkou uživatelského rozhrání a ovládáním robota Fanuc přes GPIO Raspberry Pi pomocí relé modulů.

Další stavový automat byl realizován ve třídě *SimpleRfidStateMachine* a obsahuje 7 stavů a nejdůležitějším rozdílem od předchozího stavového automatu je hlavně implementace komunikace s RFID čtečkou karet MFRC522 a také přijímaní a přenos dat do nadřazeného systému.

Finální stavový automat byl realizován ve třídě *ModbusRfidStateMachine*. Tento stavový automat obsahuje 23 stavů. Na rozdíl od stavového automatu, který byl realizován ve třídě *SimpleRfidStateMachine,* ve finálním stavovém automatu byla navíc realizována komunikace pomocí protokolu ModbusTCP s robotem Fanuc a řídicí jednotkou T-ETH-POE ESP32-WROOM. Ve finálním stavovém automatu bylo také realizováno i kompletní uživatelské rozhrání, které je ukázáno v podkapitole 4.4.6.

Níže je detailně popsán finální stavový automat, grafické znázornění kterého, lze vidět v příloze A:

1) *Idle* – je to stav ve který program přejde z počátečního stavu *EndOfMakeBeverage* už s nainicializovanou RFID čtečkou a navázanou ModbusTCP komunikaci s robotem Fanuc a deskou T-ETH-POE ESP32-WROOM, pokud tyto zařízení budou v síti. V tomto stavu se zobrazí úvodní obrazovka uživatelského rozhrání a nastartuje se RFID čtečka, která bude čekat na detekci Mifare karty. Pokud RFID čtečka detekuje kartu, tak se přejde do dalšího stavu *RfidReadCompleted*.
2) *RfidReadCompleted* – V tomto stavu RFID čtečka ukončí detekci Mifare karet a proběhne ID kontrola detekované karty, jestli karta se nachází v databázi nadřazeného systému. Pokud karta se nenachází v systému tak program přejde do stavu *ErrorRfid*, jestli se nachází tak se uloží data o uživateli a přejde se do stavu *WaitingForCoffeeOrder*.
3) *WaitingForCoffeeOrder* – V tomto stavu se zobrazí obrazovka na objednání kávy, na které se načtou tlačítka pro výběr kávy a statistická data uživatele. V případě, že uživatelem je administrátor tak se zobrazí i tlačítko na nabití kreditu pro jakéhokoliv uživatele.

   Pokud se zmačkne tlačítko na nabití kreditu, tak se přejde do stavu *FillCredit*. Pokud se zmačkne tlačítko na zrušení objednávky nebo uplyne doba čekaní, která je 20 sekund, tak se přejde do stavu *Idle*.

   V případě, že se zmačkne tlačítko na objednání nějakého typu kávy, tak následně se bude rozhodovat podle toho, jestli robot Fanuc a deska T-ETH-POE ESP32-WROOM jsou připojené k Raspberry Pi prostřednictvím ModbusTCP protokolu. Jestli ano, tak se přejde do stavu *ColorSensorControl.* V případě, že není připojena deska T-ETH-POE ESP32-WROOM, tak se přejde do stavu *ErrorModbusESP*. Pokud není připojen robot Fanuc, což znamená manuální přípravu kávy, tak proběhne ukládání transakce do nadřazeného systému a v případě úspěchu se přejde do

stavu *ManualPrepareCoffee*, v případě neúspěchu se přejde do stavu *ErrorTransaction*.

4) *ColorSensorControl* – V tomto stavu proběhne komunikace s deskou T-ETH-POE ESP32-WROOM prostřednictvím protokolu ModbusTCP, kde se vyčte hodnota, která signalizuje barvu displeje kávovaru. Jestli kávovar je vypnuty, tak se přejde do stavu *TurnOnCoffeeMachine*. V případě, že kávovar má nějakou chybu, tak se přejde do stavu *ErrorCoffeeMachine*. Pokud kávovar je připraven vařit kávu, tak se přejde do stavu *TakeCup*. Jestli bude vyčtená chybová hodnota, tak na displeji na 5 sekund bude zobrazená obrazovka s příslušnou chybou a následně se přejde do stavu *Idle*.
5) *TurnOnCoffeeMachine* – V tomto stavu na začátku proběhne kontrola, jestli dvířka robotické buňky na vydání kávy jsou zavřené, a to prostřednictvím ModbusTCP komunikace s robotem Fanuc.
   Pokud dvířka jsou otevřené, tak na displeji se zobrazí obrazovka s příslušnou chybou a nezmizí, dokud dvířka nebudou zavřené.
   V případě, že dvířka jsou zavřené, tak na displeji se přepne obrazovka na informační s tím, že kávovar se zapíná. Následně prostřednictvím ModbusTCP protokolu se pošle do robota příkaz na zapnutí programu, kde robot musí zapnout kávovar. Jestli nastane chyba v komunikaci, tak program přejde do stavu *ErrorModbusFanuc*. Následně se bude čekat 30 sekund až kávovar se zapne a program přejde do stavu *ColorSensorControl*.
6) *TakeCup* – V tomto stavu stejně jako i v předchozím stavu *TurnOnCoffeeMachine* proběhne kontrola, jestli dvířka jsou zavřené a pokud jsou otevřené tak na displeji se zobrazí obrazovka s příslušnou chybou a nezmizí, dokud dvířka nebudou zavřené.
   Pokud dvířka jsou zavřené tak na informační obrazovce se jenom změní text, že si robot bere kelímek. Následně se do robota pošle příkaz, který spustí program na nabrání kelímku. Když nastane chyba v komunikaci, tak program přejde do stavu *ErrorModbusFanuc*. Po ukončení programu na nabrání kelímku prostřednictvím protokolu ModbusTCP se zjistí, jestli kelímek je v gripperu. Jestli kelímek v gripperu není, což znamená že kelímky došly, tak se přejde do stavu *ErrorNoCups*. Pokud kelímek v gripperu je, tak se přejde do stavu *StartMakeBeverage*.
7) *StartMakeBeverage* – V tomto stavu na začátku proběhne ukládání transakce do nadřazeného systému. V případě neúspěchu ukládání se přejde do stavu *ErrorTransaction*. Následně se provede nastavení parametrů pro další stavy na základě vybraného typu kávy a přejde se do stavu *PressCoffeeButton.*
8) *PressCoffeeButton* – V tomto stavu stejně jako i ve stavu *TurnOnCoffeeMachine* proběhne kontrola, jestli dvířka jsou zavřené a pokud

jsou otevřené, tak na displeji se zobrazí obrazovka s příslušnou chybou a nezmizí, dokud dvířka nebudou zavřené.

Pokud dvířka jsou zavřené, tak se na informační obrazovce pouze změní text, který typ kávy se připravuje. Následně se do robota pošle příkaz, který spustí program v robotu pro zmačknutí příslušného tlačítka na kávovaru. Jestli nastane chyba v komunikaci, tak program přejde do stavu *ErrorModbusFanuc*. Pokud chyba nenastane a v robotu se vykoná program, tak se přejde do stavu *WaitForCoffeeFinished.*

9) *WaitForCoffeeFinished* – V tomto stavu stejně jako i ve stavu *TurnOnCoffeeMachine* proběhne kontrola, jestli dvířka jsou zavřené a pokud jsou otevřené tak na displeji se zobrazí obrazovka s příslušnou chybou a nezmizí, dokud dvířka nebudou zavřené.

   Pokud dvířka jsou zavřené, tak se do robota pošle příkaz na spouštění programu, který je určený k tomu, aby robot najel s kelímkem pod kávovar. V případě, že nastane chyba v komunikaci, tak program přejde do stavu *ErrorModbusFanuc*. Pokud chyba v komunikaci nenastane a v robotu se vykoná program, tak se bude čekat příslušnou dobu, dokud se nenalije káva. Potom se přejde do stavu *EnjoyYourCoffee.*

10) *EnjoyYourCoffee* – V tomto stavu stejně jako i ve stavu *TurnOnCoffeeMachine* proběhne kontrola, jestli dvířka jsou zavřené a pokud jsou otevřené tak na displeji se zobrazí obrazovka s příslušnou chybou a nezmizí, dokud dvířka nebudou zavřené.

    Pokud dvířka jsou zavřené, tak na informační obrazovce se jenom změní text s tím, že káva je připravena a celkový počet připravených káv se zvýší o 1. Potom do robota se pošle příkaz na spouštění programu v robotu, který je určený k tomu, aby robot najel s kelímkem kávy k výdejnímu místu (dvířkám). V případě, že nastane chyba v komunikaci, tak program přejde do stavu *ErrorModbusFanuc*. Pokud chyba v komunikaci nenastane a v robotu se vykoná program, tak se bude čekat, dokud je v robotu kelímek a zároveň nejsou zavřený dvířka, což znamená, že kávu si uživatel zatím nevzal. Až uživatel si vezme kávu a budou uzavřené dvířka, tak program přejde do stavu *GoToZeroPosition.*

11) *GoToZeroPosition* – V tomto stavu stejně jako i ve stavu *TurnOnCoffeeMachine* proběhne kontrola, jestli dvířka jsou zavřené a pokud jsou otevřené tak na displeji se zobrazí obrazovka s příslušnou chybou a nezmizí, dokud dvířka nebudou zavřené.

    Pokud dvířka jsou zavřené tak do robota se pošle příkaz na spouštění programu, který je určený k tomu, aby robot se vrátil na svoji počáteční pozici. Jestli nastane chyba v komunikaci, tak program přejde do stavu

ErrorModbusFanuc. Pokud chyba nenastane a robot se vrátí na počáteční pozici, tak program přejde do stavu *EndOfMakeBeverage.*

12) *ManualPrepareCoffee* – V tomto se stavu změní obrazovka na informační s tím, že káva uživateli byla započtena a po 7 sekundách program přejde do stavu *EndOfMakeBeverage.*
13) *EndOfMakeBeverage* – je to výchozí stav celého stavového automatu, kde se „vynuluje“ výběr kávy a přejde se do stavu *Idle*.
14) *ErrorRfid* – V tomto stavu na začátku RFID čtečka ukončí detekci Mifare karet a změní se obrazovka na chybovou s tím, že přístup k uvaření kávy je zakázán. Následně po 15 sekundách přejde se do stavu *Idle.*
15) *ErrorCoffeeMachine* – V tomto se stavu na začátku změní obrazovka na chybovou s tím, že je potřeba si zkontrolovat kávovar a až uživatel si kávovar „opraví”, tak přejde se do stavu *Idle.*
16) *ErrorNoCups* – V tomto stavu se na začátku změní obrazovka na chybovou s tím, že nejsou kelímky. Následně po 5 sekundách se přejde do stavu *GoToZeroPosition.*
17) *ErrorModbusFanuc* – V tomto se stavu na začátku změní obrazovka na chybovou s tím, že nastala chyba v komunikaci s robotem Fanuc a po 5 sekundách se přejde do stavu *Idle.*
18) *ErrorModbusESP* – V tomto stavu se na začátku změní obrazovka na chybovou s tím, že nastala chyba v komunikaci s deskou ETH-POE ESP32-WROOM a po 5 sekundách se přejde do stavu *Idle.*
19) *ErrorTransaction* – V tomto stavu se na začátku změní obrazovka na chybovou s tím, že nastala chyba s ukládáním transakce do databáze nadřazeného systému a po 5 sekundách se přejde do stavu *Idle.*
20) *FillCredit* – V tomto stavu se na začátku zobrazí obrazovka na dobití kreditu uživatele a potom se nastartuje RFID čtečka, která bude čekat na detekci Mifare karty. Pokud RFID čtečka detekuje kartu, tak se přejde do dalšího stavu *FillProcess*.
21) *FillProcess* – V tomto stavu se na začátku skončí čtení karet RFID čtečkou. Následně pokud bylo na displeji vybráno o kolik se má nabít karta uživatele, tak proběhne ukládání transakce do nadřazeného systému. V případě úspěšné transakce se přejde do stavu *FillProcessFinised,* v případě neúspěšné transakce se přejde do stavu *FillProcessError.* Pokud na displeji nebylo vybráno o kolik se má karta uživatele nabít, tak se taktéž přejde do stavu *FillProcessError.*
22) *FillProcessError* – V tomto stavu se na začátku změní obrazovka na chybovou s tím, že nabití kreditu Mifare karty neproběhlo a po 5 sekundách se přejde do stavu *Idle.*

23) *FillProcessFinised* – V tomto stavu se na začátku změní obrazovka na informační s tím, že nabití kreditu Mifare karty proběhlo úspěšně a po 5 sekundách se přejde do stavu *Idle*.

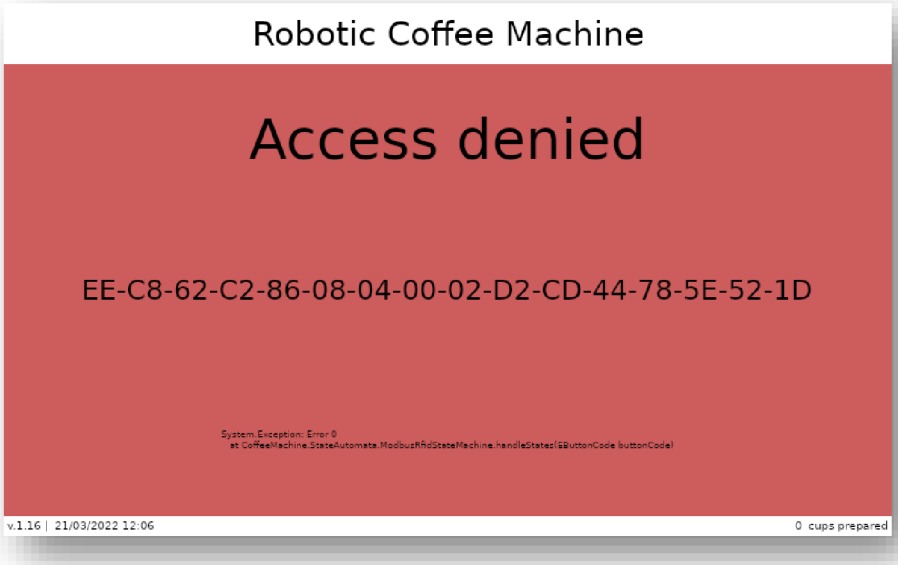
4.4.6 Uživatelské rozhrání robotické buňky

V rámci této práce bylo navrženo uživatelské rozhrání robotické buňky na platformě .NET pomocí frameworku Avalonia. V této podkapitole budou popsány a ukázany vytvořené aplikační obrazovky.

Na obrázku 4.3 lze vidět úvodní obrazovku navržené aplikace, kde jenabítka pro přiložení Mifare karty pro objednání kávy. Na všech obrazovkách aplikace je zobrazená její aktuální verze, dále aktuální datum a čas a v neposlední řadě počet připravených káv. Na této obrazovce se zobrazují různé věty, které se každých 5 vteřin mění a jsou uložené v databáze nadřazeného systému. Na obrazovce se také zobrazuje, zdali jsou v rámci komunikační sítě Modbus TCP připojené robot Fanuc a snímač barvy displeje kávovaru. Pokud je robot Fanuc zapnutý, tak proběhne uvaření kávy pomocí robota, ale v rámci aplikace je umožněno taktéž i manuální uvaření kávy, pokud robot je vypnutý.

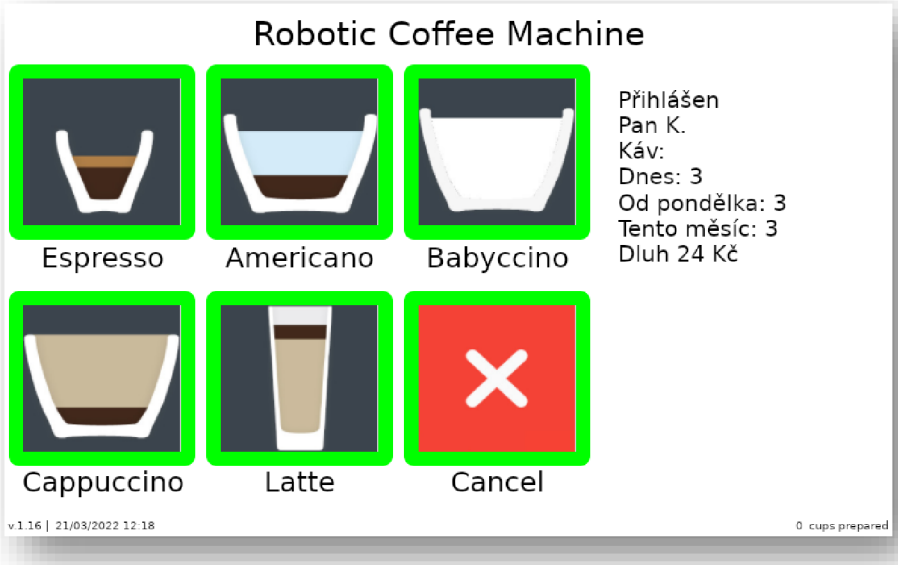
Obrázek 4.3 Aplikační úvodní obrazovka uživatelského rozhrání

Po přiložení Mifare karty k RFID čtečce proběhne její rozpoznávání a pokud se karta nenachází v databáze nadřazeného systému, tak aplikace neumožní objednání kávy a bude zobrazená chybová hláška, jak je to vidět na obrázku 4.4. Na obrazovce se zobrazí, že je přístup uživateli odepřen a zobrazí se identifikační číslo jeho karty, které případně musí být přidáno do databáze nadřazeného systému, pokud si uživatel bude chtít kávu objednávat. Tato hláška se bude ukazovat na displeji v průběhu 15 sekund, aby si uživatel stihl zaznamenat identifikační číslo své karty.



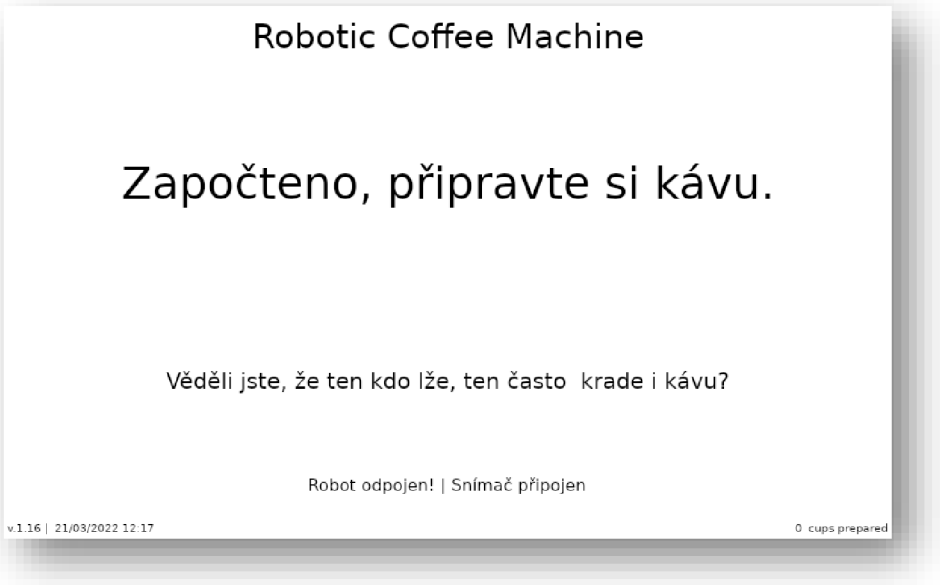
Obrázek 4.4 Aplikační obrazovka s chybovou hláškou neznáme Mifare karty

Pokud se karta uživatele nachází v databáze nadřazeného systému, tak jej aplikace pustí dále a umožní mu objednání kávy. Na obrázku 4.5 lze vidět obrazovku na objednání kávy. V levé částí obrazovky jsou umístěné tlačítka na objednání různých typů káv, mezi které patří i tlačítko na zrušení objednávky. V pravé částí obrazovky se ukazují statistická data uživatelé, kde je popsáno, kdo je přihlášen na objednání kávy, kolik káv vypil uživatel za den, v průběhu týdne a v průběhu měsíce a také i kolik peněz má na účtu nebo kolik peněz dluží.



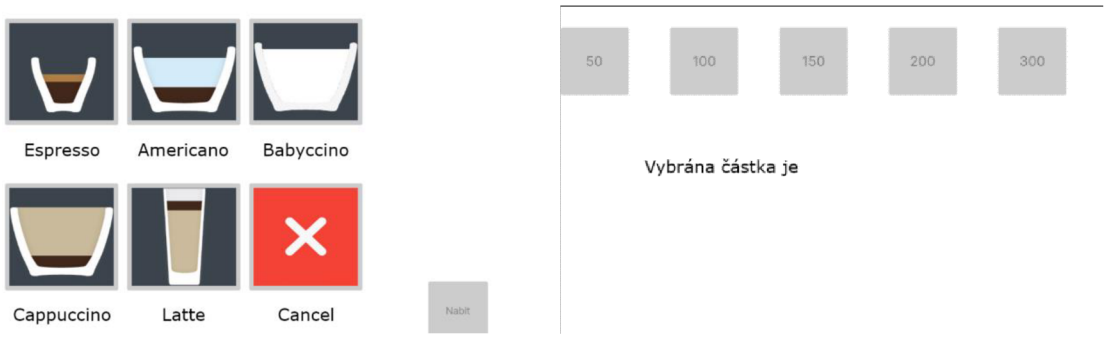
Obrázek 4.5 Aplikační obrazovka na objednání kávy

Po stisknutí tlačítka „Cancel“ aplikace se vrátí na úvodní obrazovku. Po zmačknutí tlačítka s výběrem kávy v případě, že je robot zapojený, tak na obrazovce bude uvedeno, která káva se připravuje. Pokud je robot odpojený, tak se na 7 sekund ukáže obrazovka, kterou lze vidět na obrázku 4.6 a následně se vrátí úvodní obrazovka.



Obrázek 4.6 Aplikační obrazovka po objednání kávy s odpojeným robotem

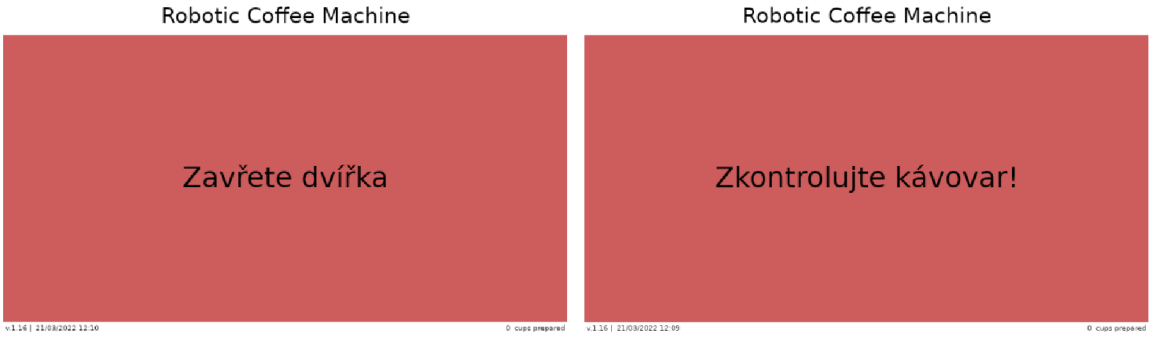
Aplikace byla navržená takovým způsobem, že dobití kreditu pro uživatele je umožněno pouze uživateli s administrátorskými právy. Pokud administrátor přiloží svoji Mifare kartu, tak na obrazovce vpravo dole se objeví tlačítko “Nabít”, jak je to vidět na obrázku 4.7 a). Po stisknutí toho tlačítka aplikace se přepne na obrazovku s výběrem kreditu na dobití. Tuto obrazovku lze vidět na obrázku 4.7 b). Následně uživatel může vybrat na kolik chce dobít svůj kredit a po přiložení jeho Mifare karty dobití proběhne.



a) b)

Obrázek 4.7 Aplikační obrazovka administrátora (a) aplikační obrazovka na dobití kreditu (b)

Na obrázku 4.8 pro příklad jsou uvedené aplikační obrazovky s různými chybovými hláškami, které mohou vzniknout v průběhu používaní robotického kávovaru. Zejména je uvedená chybová hláška "Zavřete dvířka", která může být způsobená otevřením dvířek v čase přípravy kávy robotickou rukou nebo hláška "Zkontrolujte kávovar" pokud došlo v kávovaru k přeplnění zásobníku na odpad nebo nedostatku vody či zrnkové kávy v kávovaru.



Obrázek 4.8 Aplikační obrazovky s chybovými hlášky

4.5 Implementace řídicího programu v ESP32-WROOM

V této kapitole bude popsán programovací algoritmus realizovány v mikrokontroleru ESP32-WROOM, který je základem desky T-ETH-POE ESP32-WROOM. Program byl realizován v programovacím jazyce C++ v rámci frameworku Arduino a pomocí platformy *PlatformIO,* která byla vyvinuta pro embedded programování.

Základem celého programu jsou tři třídy, které jsou popsány zde:

1) Třída *Ethernet* – tato třída obsahuje inicializační metodu, která umožnuje nakonfigurovat a zprovoznit ethernet komunikaci. IP adresa byla nastavena na 192.168.5.5.
2) Třída *ModbusTcp* – tato třída obsahuje také inicializační metodu, kde je provedená inicializace ModbusTCP Serveru včetně adres pro komunikaci s ModbusTCP Clientem, který byl realizován v hlavní řídicí jednotce Raspberry Pi.
3) Třída *ColorDetection* – tato třída slouží přímo ke barevné detekci displeje kávovaru snímačem barvy. Obsahuje inicializační metodu, kde je provedeno nastavení vstupních a výstupních pinů čipu ESP32-WROOM. Byla také realizovaná metoda *update*, která umožnuje přepínaní u snímače barvy TCS3200 skupin fotodiod s různými filtry (červený, zelený, modrý) a čtením výstupní frekvencí signálu ze snímače. Součásti této metody je také i algoritmus, který na základě šířky pulzu výstupní frekvence ze snímače barvy určuje barvu displeje kávovaru.

Následně je zde popsán algoritmus hlavní smyčky programu. Čeká se na požadavek spojení od ModbusTCP Clienta. Ve chvílí, kdy přijde požadavek, tak ho ModbusTCP

Server přijme a naváže se spojení. Následně se čeká až klient pošle požadavek na vyčtení dat o barvě displeje kávovaru, kde se musí jednobitový registr (coil) nastavit v jedničku. Až je jednobitový registr v jedničce, tak proběhne měření a do 16bitového registru (holding register) bude zapsána hodnota, která odpovídá barvě displeje kávovaru. Následně do stejného jednobitového registru bude zapsána nula, což pro klienta signalizuje dokončení měření a uložení příslušné hodnoty o barvě displeje do 16bitového registru.

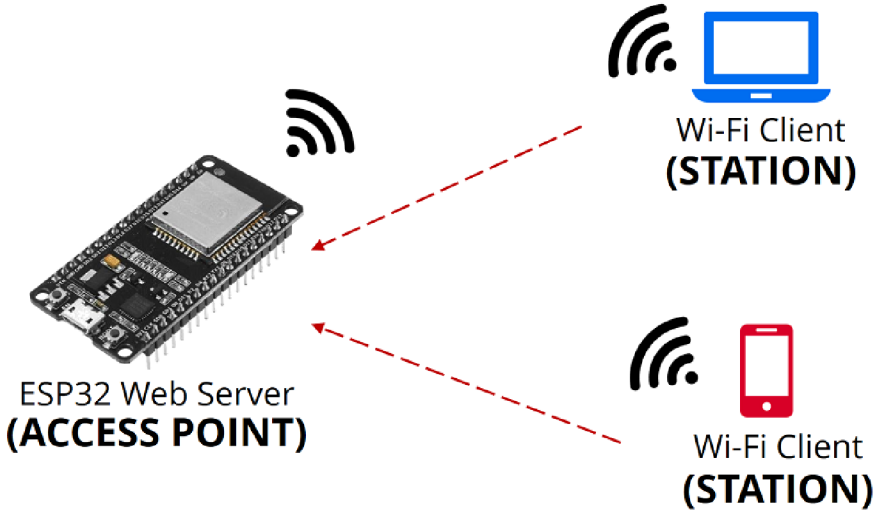
```
void loop()
{
    client = ethernet.ethServer.available();

    if (client)
    {
        Serial.println("[Client connected]");

        modbusTcp.modbusTCPServer.accept(client);

        while (client.connected())
        {
            if (client.available() > 0)
            {
                modbusTcp.modbusTCPServer.poll();

                if (modbusTcp.modbusTCPServer.coilRead(COLOR_DETECTION_COIL))
                {
                    colorDetection.update();

modbusTcp.modbusTCPServer.holdingRegisterWrite(COLOR_DETECTION_VALUE_REGISTER,
                    CoffeeMachine::ColorDetection::color);

                  modbusTcp.modbusTCPServer.coilWrite(COLOR_DETECTION_COIL,0);
                }
            }
        }

        client.stop();
        Serial.println("[Client disconnected]");
    }
}
```

4.6 Inteligentní maják

V této podkapitole bude popsán programovací algoritmus, který byl realizován v mikrokontroleru ESP8266 a je základem desky ESP8266 NodeMCU V2. Program byl taktéž realizován v programovacím jazyce C++ pomocí platformy *PlatformIO* a pomocí frameworku Arduino.

V rámci této práce bylo požadavkem, aby se inteligentní maják ovládal přes Wi-Fi a byl modulární a přenositelnou jednotkou, což znamená, že se může připojovat k různým Wi-Fi sítím v různých místnostech. Aby toto bylo možné, tak jedním z řešením je zadání jména a hesla Wi-Fi sítě přímo v kódu a následně i pro každou jinou Wi-Fi síť tyto údaje v kódu měnit a přehrávat program v mikrokontroleru. Jinou cestou je realizovat v mikrokontroleru režim přístupového bodu (Access Point), který umožní

připojení k mikrokontroleru pomocí jakékoliv Wi-Fi stanice, jako například mobil nebo počítač, jak je to vidět na obrázku 4.9, a následně přes realizovaný web server ukládat data nové Wi-Fi sítě do mikrokontroleru.



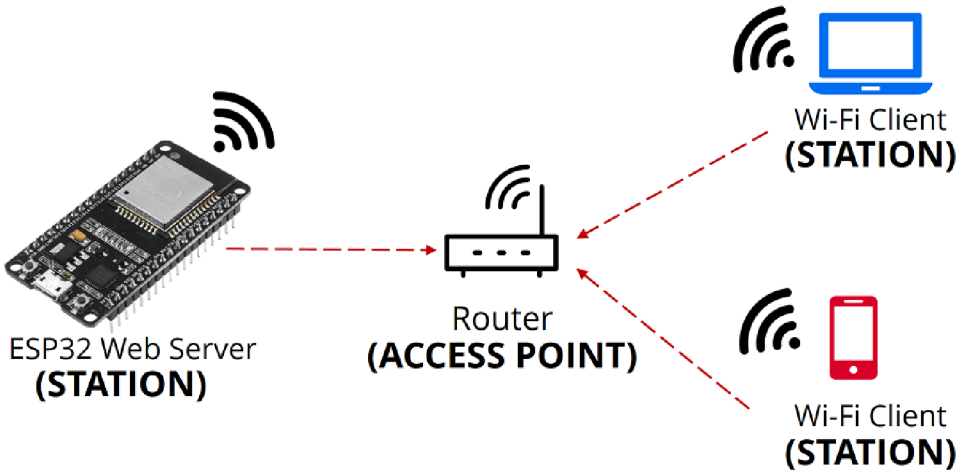
Obrázek 4.9 Mikrokontroler ESP jako přístupový bod [29]

Proto prvním krokem v rámci implementace algoritmu inteligentního majáku s mikrokontrolerem ESP8266 byla právě realizace režimu přístupového bodu. Hlavní funkcí této části algoritmu je funkce *accessPoint*, ve které proběhne spouštění a konfigurace přístupového bodu na IP adrese 192.168.1.1. Následně pomocí HTML a CSS byl realizován web server, ve kterém po spojení jakékoliv stanice s přístupovým bodem lze zadat jméno a heslo Wi-Fi sítě a také webovou adresu se kterou se bude komunikovat přes HTTPS protokol. Následně tyto parametry přes webové tlačítko budou uložené v nevolatilní pamětí mikrokontroleru. Při následujících spouštěních mikrokontroleru v režimu přístupového bodu tyto parametry se budou vyčítat z pamětí a zobrazovat se na web serveru. V případě potřeby lze parametry změnit a změny se automaticky uloží do pamětí.

```
void accessPoint()
{
    readEeprom();

    WiFi.mode(WIFI_AP);
    WiFi.softAPConfig(ipLocal, gateway, subnet);
    WiFi.softAP(ssidAccessPoint);

    delay(100);

    webServer.on("/", settingsPage);
      webServer.on("/ok", afterSubmitPage);
    webServer.begin();
}
```

V dalším kroku implementace s mikrokontrolerem ESP8266 byl realizován režim Wi-Fi stanice. Wi-Fi router se chová jako přístupový bod a mikrokontroler ESP8266, stejně jako například mobil nebo počítač, se chová jako Wi-Fi stanice, jak je to ukázáno na obrázku 4.10. Hlavní funkci této části algoritmu je funkce *WifiServer*, ve které na začátku proběhne vyčtení dat o jménu a heslu Wi-Fi sítě z pamětí mikrokontroleru, následně proběhne spojení s Wi-Fi síti.

```
void WifiServer()
{
    readEeprom();

    WiFi.mode(WIFI_STA);

    WiFi.begin(ssid, pass);

    // Wait for connection
    if ( WiFi.status() != WL_CONNECTED )
    {
      for (int i = 0; i < 20; i++)
      {
        delay ( 500 );
        if (WiFi.status() == WL_CONNECTED)
          break;
      }
    }
}
```



Obrázek 4.10 Mikrokontroler ESP jako Wi-Fi stanice [29]

V dalším kroku byl realizován HTTPS klient, který vyčítá data ze serveru. Odpovědí ze serveru je požadovaná barva majáku ve formátu textového řetězce. Byla realizovaná třída *Https*, která především obsahuje metodu *update*, ve které na začátku proběhne spouštění HTTPS klienta, následně bude odeslán požadavek na server a potom přijetí odpovědi ze serveru. Timeout na vyčtení dat byl nastaven na 3 sekundy.

```
void Https::update(String url, uint16_t timeout)
{
    _startMillis = millis();

    while((millis()                                                -
_startMillis < timeout)&&(WiFi.status()==WL_CONNECTED))
    {
        client.setInsecure();

        https.begin(client, url);

        httpCode = https.GET();

        if (httpCode > 0)
          payload = https.getString();

        https.end();
    }
}
```

Následně byl realizován algoritmus na ovládaní LED na základě vyčtených dat z HTTPS serveru. Algoritmus byl rozdělen na 2 části podle toho, který textový řetězec se objeví na serveru. Pokud první číslice řetězce je 0, tak je to první algoritmus, pokud první číslice je 1, tak je to druhý algoritmus. První algoritmus je určen k rozsvícení jenom jednoho patra LED inteligentního majáku určitou barvou a bude následně popsán na základě textového řetězce:

- "0,1" – červenou barvou bude rozsvíceno horní patro inteligentního majáku
- "0,2" – oranžovou barvou bude rozsvíceno druhé patro inteligentního majáku
- "0,3" – zelenou barvou bude rozsvíceno dolní patro inteligentního majáku

Druhý algoritmus je určen k rozsvícení LED jakoukoliv barvou a může být rozsvíceno jedno nebo více pater inteligentního majáku. Odpovědí ze serveru pro každé patro je požadovaná barva majáku v hexadecimálním formátu RRGGBBXX, kde RR GG BB jsou 0x00 – 0xFF hodnoty jednotlivých barevných složek RGB. XX může mít hodnotu 0x01 nebo 0x00, kde 1 znamená blikání LED a 0 svícení.

V posledním kroku byl realizován hlavní algoritmus, který určuje, v jakém režimu má inteligentní maják fungovat. Pokud je tlačítko majáku zmáčknuto a uživatel následně připojí maják k síti, tak se spustí režim přístupového bodu. Pokud tlačítko majáku zmačknuto není, tak se po připojení majáku do sítě maják spustí v režimu Wi-Fi stanice a po spojení s HTTPS serverem se rozsvítí LED podle zadaných hodnot.

```
void setup()
{
    pinMode(PIN_BUTTON, INPUT_PULLUP);

    rgbWS.begin();
    Serial.begin(9600);
    EEPROM.begin(226);

    ssidLength =        EEPROM.read(96);
```

```
    passwordLength =    EEPROM.read(97);
    urlLength =         EEPROM.read(225);

    isAccessPoint = !digitalRead(PIN_BUTTON);

    if (!isAccessPoint) WifiServer();
    else                accessPoint();
}

void loop()
{

    if (isAccessPoint)
        webServer.handleClient();
    else
    {
        httpsLightHouse.update(url, 3000);
        httpsLed();
    }
}
```

5. NASAZENÍ A PRŮBĚH OŽIVENÍ

Cela .NET aplikace se vyvíjela na PC v programovacím prostředí Visual Studio. Ale testování probíhalo na Raspberry Pi. Aby to bylo možné, tak bylo v první řádě nutné v konfiguračním menu Raspberry Pi OS povolit SPI rozhrání pro RFID čtečku a SSH (Secure Shell) pro přenos souboru s PC do paměťové karty Raspberry Pi. Následně bylo nutné na PC celou .NET aplikaci si zkompilovat a přenést zkompilované soubory, které se nacházejí ve složce *bin/Debug/net5.0*. Pro takovýto přenos souboru lze využít například aplikaci FileZilla. V následující podkapitole bude popsána realizace automatického spouštění navržené .NET aplikace v Raspberry Pi.

5.1 Automatické spouštění .NET GUI aplikace v Linuxu

V rámci této práce bylo realizováno automatické spouštění .NET aplikace na Raspberry Pi po bootování operačního systému. Jelikož navržená .NET aplikace obsahuje grafické uživatelské rozhrání, tak metody automatického spouštění přes *rc.local* a *autostart* soubory nejsou pro tento účel funkční. K tomuto účelu bylo potřeba využít systém *systemd*, který existuje ve většině linuxových distribucí, včetně Raspberry Pi OS. *Systemd* je metoda spouštění aplikací, kde je například možné, aby se spustili určité programy až po spouštění určitých služeb. [30]

Pro realizaci tohoto požadavku bylo v prvním kroku potřeba zjistit základní cestu .NET platformy na Raspberry Pi a to následně:

```
dotnet --info
```

Z odpovědi byla zjištěna základní cesta .NET - /opt/dotnet.

V dalším kroku bylo nutné vytvořit skriptovací soubor – službu v systému *systemd*, který bude automatický spouštět navrženou aplikaci. Soubor byl vytvořen následovně:

```
sudo nano /lib/systemd/system/coffee.service
```

Následně byl napsán skriptovací program, který umožnuje automatické spouštění navržené aplikace:

```
[Unit]
Description=Coffee Machine

[Service]
User=pi
Environment=DISPLAY=:0
Environment=XAUTHORITY=/home/pi/.Xauthority
Environment=DOTNET_ROOT=/opt/dotnet
ExecStart=dotnet
/home/pi/CoffeeMachine/bin/Debug/net5.0/CoffeeMachine.dll
Restart=always
RestartSec=10s
KillMode=process
TimeoutSec=infinity

[Install]
WantedBy=graphical.target
WantedBy=multi-user.target
```

Všechny služby v systému *systemd* mají určitá psací pravidla, příkazy, proměnné a kódová slova, podle kterých bylo postupováno při vytváření tohoto skriptu. Proměnné Description může být přiřazen jakýkoliv text a je určen jen pro popis daného skriptovacího souboru. [30]

V sekci [Service] je první proměnnou User, které bylo přiřazeno pi, což znamená, že se celý skript spustí až se Raspberry Pi přihlásí do uživatelé pi. Pomocí příkazu `sudo raspi-config` bylo nastaveno, aby se Raspberry Pi automatický po spuštění přihlásilo do uživatele pi bez jakéhokoliv hesla. Další proměnnou je Environment, které je v první řadě přiřazená proměnná DISPLAY, které bylo potřeba přiradit správnou adresu displeje a to – :0.0. Adresa displeje byla zjištěna pomocí příkazu `echo $DISPLAY`. Proměnné XAUTHORITY je potřeba přiřadit cestu k souboru .Xauthority, který se nachází v domovském adresáři každého uživatele, jmenovitě ve složce */home/pi*. Důležitou proměnnou pro .NET aplikaci je proměnná DOTNET_ROOT, které je potřeba přiřadit základní cestu .NET platformy. Proměnné ExecStart je potřeba přiřadit příkaz pomocí kterého lze spustit aplikaci. Pokud aplikace selže nebo se ukončí, tak k jejímu restartování je možné využít proměnné Restart a RestartSec, které jsou nastavené vždy na restartovaní po 10 sekundách v případě selhání nebo ukončení aplikace. Příkaz KillMode=process říká systému *systemd* aby ukončil všechny procesy spojené s aplikaci, pokud aplikace selže nebo se uzavře. Příkaz TimeoutSec=infinity znamená, že pokus o spouštění aplikace musí probíhat nekonečně dlouho. [30]

V sekci [Install] jsou 2 příkazy: WantedBy=graphical.target a WantedBy=multi-user.target, které vyžadují nutné spouštění těchto dvou služeb pro běh navržené služby *coffee.service*.

V dalším kroku je potřeba aby systém *systemd* navrženou službu rozpoznal K tomu je potřeba ve složce */lib/systemd/system* zadat následující příkaz:

```
sudo systemctl daemon-reload
```

Aby se navržená služba *coffee.service* spouštěla vždy po zapnutí Raspberry Pi, je nutné zadat další příkaz ve stejné složce:

```
sudo systemctl enable coffee.service
```

V rámci práce bylo implementováno taktéž i vypnutí kurzoru na dotykovém displeji. K tomuto účelu bylo nutné v souboru */etc/lightdm/lightdm.conf* ve skupině [Seat*] přidat příkaz *xserver-command = X -nocursor* a restartovat Raspberry Pi.

6. ZÁVĚR

Tato diplomová práce se zabývá návrhem a realizací řízení a vizualizace demonstrační robotické buňky na přípravu kávy, především tedy elektrickým návrhem včetně výběru vhodných hardwarových komponent a kompletním programovým návrhem pro zprovoznění celého navrženého řešení.

V první kapitole byla provedena analýza stávajícího řešení a analýza požadavků práce. Následně byly popsány navrhované změny pro splnění vytyčených požadavků, zejména došlo ke změně dávkování kávy do kelímku místo dávkování do hrnků, což umožnilo zrychlit přípravu kávy a zvětšit počet připravených káv bez zásahu operátora. V neposlední řádě byl taktéž popsán návrh celého technického řešení pro splnění požadavků práce.

Druhá kapitola obsahuje teoretický úvod, kde byl popsán princip řízení robotického systému, byla zde provedená analýza komunikačních možností komerčních robotů a v neposlední řádě byla popsána komunikační knihovna, která byla dodána pro realizaci této práce a následně sloužila pro komunikaci hlavní řídicí jednotky s robotickou rukou Fanuc.

V následující kapitole byly popsány hardwarové prostředky, které byly vybrané a použité pro realizaci této práce, včetně schémat elektrických zapojení. V této kapitole po hardwarové stráně byla popsaná realizace požadavků na řízení celé robotické buňky, autentizace a autorizace uživatele prostřednictvím přístupové kary, detekce stavů kávovaru, světelná indikce procesu přípravy kávy.

Ve čtvrté kapitole byla popsána celá softwarová realizace navrženého řešení a bylo zdůvodněno použití určitých programovacích technologií. Hlavní řídicí aplikace v Raspberry Pi 4 B byla realizována pomocí platformy .NET a multiplatformního GUI frameworku Avalonia. V této kapitole byla popsána implementace hlavního řídicího algoritmu, následně algoritmu pro komunikace Raspberry Pi s robotem Fanuc a řídicí deskou T-ETH-POE ESP32-WROOM prostřednictvím komunikačního protokolu ModbusTCP, algoritmu pro komunikace Raspberry Pi s RFID čtečkou karet, algoritmu pro komunikace Raspberry Pi s nadřazeným systémem prostřednictvím HTTP protokolu. Byla zde taktéž ukázána a popsána realizovaná vizualizace demonstrační robotické buňky. V neposlední řádě zde byly také popsány řídicí algoritmy pro splnění požadavků na detekci stavů kávovaru a světelné indikace procesu přípravy kávy.

V poslední kapitole byl popsán průběh oživení celé .NET aplikace na Raspberry Pi, včetně automatického spouštění této aplikace.

Výsledkem této diplomové práce je funkční zrealizované řešení, které umožňuje si uživateli prostřednictvím přístupové karty a dotykového displeje objednat kávu, která mu bude následně uvařena a robotickou rukou vydána v kelímku.

Demonstrační robotická buňka byla předvedena v roce 2022 na dne otevřených dveří FEKT VUT.

LITERATURA

[1] KLEMENT, Petr. NÁVRH A VIRTUÁLNÍ SIMULACE JEDNOÚČELOVÉHO STROJE [online]. Brno, 2021 [cit. 2022-04-18]. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=226249. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií.

[2] Návrh a vývoj univerzálního rozhraní pro integraci průmyslových robotů. Brno, 2020.

[3] HTTP (Hypertext Transfer Protocol). TechTarget [online]. San Francisco: Wesley Chai, 2021 [cit. 2022-04-18]. Dostupné z: https://www.techtarget.com/whatis/definition/HTTP-Hypertext-Transfer-Protocol

[4] Series 5000 Plně automatický kávovar. PHILIPS [online]. [cit. 2022-04-18]. Dostupné z: https://www.philips.cz/c-p/EP5360_10/series-5000-plne-automaticky-kavovar

[5] LR Mate 200iD/4S. FANUC [online]. [cit. 2022-04-18]. Dostupné z: https://www.fanuc.eu/cz/en/robots/robot-filter-page/lrmate-series/lrmate-200id-4s

[6] Raspberry Pi 4. Raspberry Pi [online]. Raspberry Pi Foundation [cit. 2022-04-18]. Dostupné z: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[7] Raspberry Pi 4 Tech Specs. Raspberry Pi [online]. Raspberry Pi Foundation [cit. 2022-04-18]. Dostupné z: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[8] RFID (radio frequency identification). TechTarget [online]. New Hampshire: Sarah Amsler, 2021 [cit. 2022-04-18]. Dostupné z: https://internetofthingsagenda.techtarget.com/definition/RFID-radio-frequency-identification

[9] RFID Reader/Writer. RobotDyn [online]. [cit. 2022-04-18]. Dostupné z: https://robotdyn.com/rfid-reader-writer-nfc-module-mfrc522.html

[10] MFRC522 Standard performance MIFARE and NTAG frontend. NXP [online]. NXP, 2016 [cit. 2022-04-18]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf

[11] Detektor barvy. Hadex [online]. [cit. 2022-04-18]. Dostupné z: https://www.hadex.cz/m532-detektor-barvy---arduino-modul-gy-031-s-tsc3200/?gclid=Cj0KCQiA2sqOBhCGARIsAPuPK0j7qeapJ1oRVnTn5EHlcG9TQydBOjpTVSZv4XAs5qedHUU4pfrdYWkaAkFNEALw_wcB

[12] TCS3200, TCS3210 PROGRAMMABLE COLOR LIGHT-TO-FREQUENCY CONVERTER. Mouser [online]. Texas advanced optoelectronic solutions, 2009 [cit. 2022-04-18]. Dostupné z: https://www.mouser.com/catalog/specsheets/tcs3200-e11.pdf

[13] TTGO T-Internet-POE ESP32-WROOM LAN8720A Chip Ethernet Adapter. LILYGO [online]. LILYGO [cit. 2022-04-18]. Dostupné z: http://www.lilygo.cn/prod_view.aspx?TypeId=50033&Id=1307

[14] NodeMCU v2 CP2102. Gleanntronics [online]. [cit. 2022-04-18]. Dostupné z: https://gleanntronics.ie/en/products/nodemcu-v2-cp2102-wifi-esp8266-lua-arduino-250.html
[15] ESP8266 NodeMCU Tutorial. DiyI0T [online]. 2021 [cit. 2022-04-18]. Dostupné z: https://diyi0t.com/esp8266-nodemcu-tutorial/
[16] ESP8266EX Datasheet. Espressif [online]. Espressif Systems, 2020 [cit. 2022-04-18]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
[17] What is .NET? Introduction and overview. Microsoft technical documentation [online]. Microsoft, 2022 [cit. 2022-04-18]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/core/introduction
[18] An overview of Windows 10 IoT Core. Microsoft technical documentation [online]. Microsoft, 2021 [cit. 2022-04-18]. Dostupné z: https://docs.microsoft.com/en-us/windows/iot-core/windows-iot-core
[19] Windows 10 IoT Core Dashboard. Microsoft technical documentation [online]. Microsoft, 2020 [cit. 2022-04-18]. Dostupné z: https://docs.microsoft.com/en-us/windows/iot-core/connect-your-device/iotdashboard
[20] Getting Started with Windows 10 IoT Core & Raspberry Pi 3B+. Developer Support [online]. Microsoft, 2019 [cit. 2022-04-18]. Dostupné z: https://devblogs.microsoft.com/premier-developer/getting-started-with-windows-10-iot-core-raspberry-pi-3b/
[21] Raspberry Pi 3B+ booting issues. Microsoft technical documentation [online]. Microsoft, 2020 [cit. 2022-04-18]. Dostupné z: https://docs.microsoft.com/en-us/windows/iot-core/troubleshooting?branch=master#raspberry-pi-3b-booting-issues
[22] Install and use Microsoft Dot NET 5 with the Raspberry Pi [online]. Pete Gallagher, 2020 [cit. 2022-04-18]. Dostupné z: https://www.petecodes.co.uk/install-and-use-microsoft-dot-net-5-with-the-raspberry-pi/
[23] .NET Desktop Apps [online]. Microsoft [cit. 2022-04-18]. Dostupné z: https://dotnet.microsoft.com/en-us/apps/desktop
[24] QmlNet [online]. [cit. 2022-04-18]. Dostupné z: https://github.com/qmlnet/qmlnet
[25] UNO Platform [online]. Uno Platform [cit. 2022-04-18]. Dostupné z: https://platform.uno/
[26] Avalonia [online]. Tallinn: AvaloniaUI OÜ [cit. 2022-04-18]. Dostupné z: https://avaloniaui.net/
[27] Avalonia Welcome [online]. AvaloniaUI OÜ [cit. 2022-04-18]. Dostupné z: https://docs.avaloniaui.net/
[28] XAML overview. Microsoft technical documentation [online]. Microsoft, 2021 [cit. 2022-04-18]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-6.0#what-is-xaml

[29] How to Set an ESP32 Access Point (AP) for Web Server. Random Nerd Tutorials [online]. [cit. 2022-04-18]. Dostupné z: https://randomnerdtutorials.com/esp32-access-point-ap-web-server/

[30] How to Run a Raspberry Pi Program on Startup. Sparkfun [online]. [cit. 2022-04-18]. Dostupné z: https://learn.sparkfun.com/tutorials/how-to-run-a-raspberry-pi-program-on-startup/all

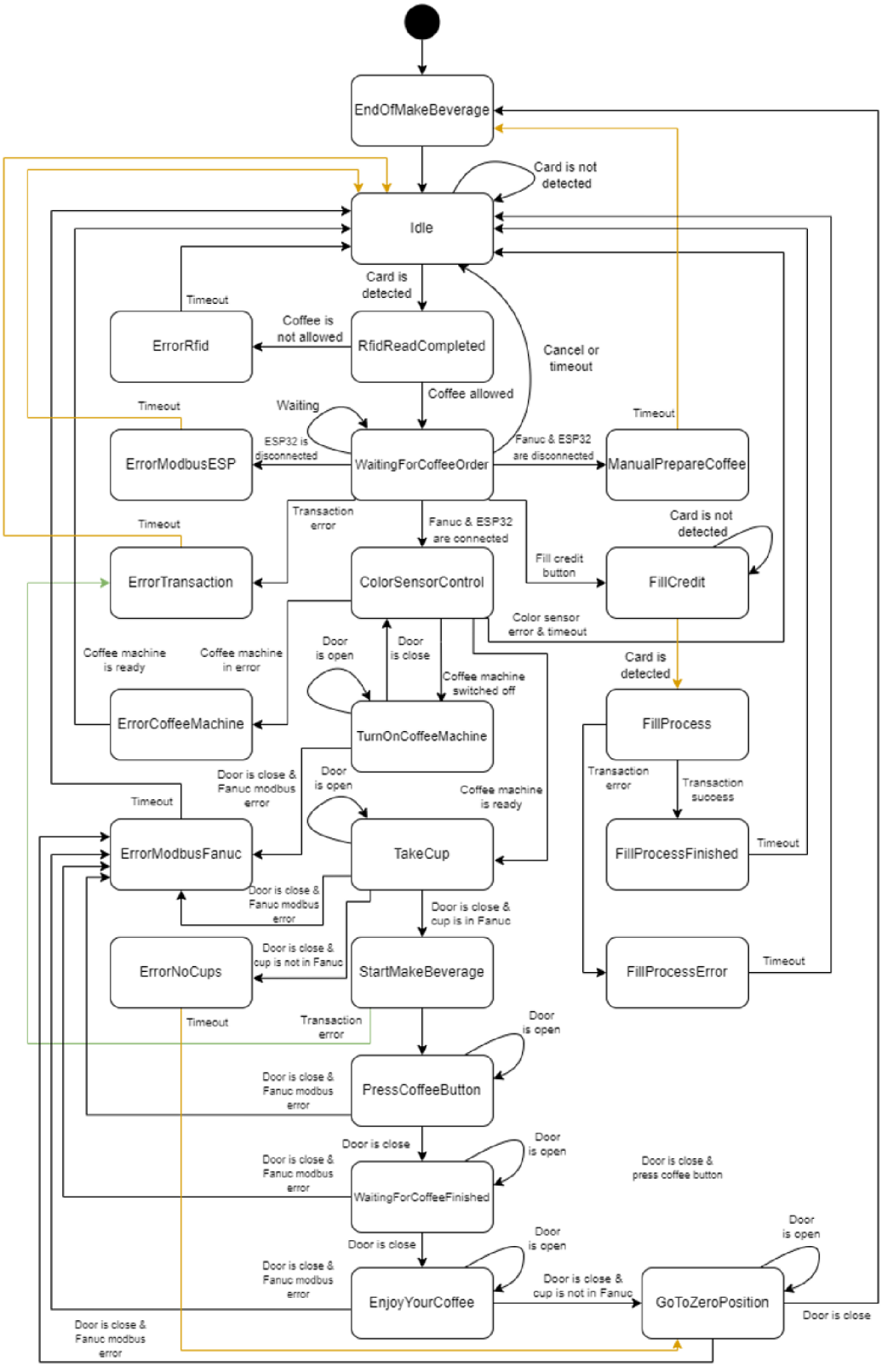
SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

FEKT	Fakulta elektrotechniky a komunikačních technologií
VUT	Vysoké učení technické v Brně
RFID	Radio Frequency Identification
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
TCP/IP	Transmission Control Protocol/Internet Protocol
CAN	Controller Area Network
IEC	International Electrotechnical Commission
PLC	Programmable Logic Controller
DCS	Distributed Control System
API	Application Programming Interface
XML	Extensible Markup Language
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
TLS	Transport Layer Security
SSL	Secure Sockets Layer
DC	Direct Current
SDRAM	Synchronous Dynamic Random Access Memory
SRAM	Static Random Access Memory
GPIO	General-purpose input/output
UART	Universal asynchronous receiver-transmitter
I2C	Inter-Integrated Circuit
I2S	Inter-IC Sound
SPI	Serial Peripheral Interface
USB	Universal Serial Bus
BLE	Bluetooth Low Energy
ISIC	International Student Identity Card
CMOS	Complementary Metal–Oxide–Semiconductor
LED	Light-Emitting Diode
DLL	Dynamic-link library
GUI	Graphic User Interface
IoT	Internet of Things

SEZNAM PŘÍLOH

Příloha A - Stavový automat hlavní řídicí aplikace



Příloha B - Obsah přiloženého CD/DVD

Přiložený CD/DVD obsahuje elektronickou verzi této diplomové práce, zdrojový kód navržené .NET aplikace pro Raspberry Pi, zdrojový kód pro řídicí desku T-ETH-POE ESP32-WROOM, zdrojový kód pro inteligentní maják.