Univerzita Jana Evangelisty Purkyně
v Ústí nad Labem
Přírodovědecká fakulta

Řízení domácnosti pomocí PLC a následná vizualizace pomocí webového serveru

BAKALÁŘSKÁ PRÁCE

Vypracoval: František Oplt
Vedoucí práce: Ing. Petr Haberzettl

Studijní program: Informační systémy
Studijní obor: Aplikovaná informatika

ÚSTÍ NAD LABEM 2018

zde vložte zadání!!!

Prohlášení

Prohlašuji, že jsem tuto diplomovou/bakalárskou práci vypracoval samostatně a použil jen pramenů, které cituji a uvádím v přiloženém seznamu literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona c. 121/2000 Sb., ve znění zákona c. 81/2005 Sb., autorský zákon, zejména se skutečností, že Univerzita Jana Evangelisty Purkyně v Ústí nad Labem má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Jana Evangelisty Purkyně v Ústí nad Labem oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladu, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

V Ústí nad Labem dne 15. dubna 2018 Podpis: .....

Děkuji vedoucímu práce doc. Pafnutijovi Snědldítětikaši, Ph.D.
za neocenitelné rady a pomoc při tvorbě bakalářské práce.

ŘÍZENÍ DOMÁCNOSTI POMOCÍ PLC A NÁSLEDNÁ VIZUALIZACE POMOCÍ WEBOVÉHO SERVERU

Abstrakt:

Bakalářská práce se zaměřuje na možnosti ovládání domácnosti pomocí webového rozhraní. Řízení domácnosti je realizováno pomocí programovatelného logického automatu (PLC). Jako PLC je použit nejnovější kontrolér od firmy Siemens, a to SIMATIC s7-1200. Pro ovládání je použit webový server. Webový server je navržen tak, aby se dal snadno použít i pro komunikaci s jinými zařízeními zabývajícími se domácí automatizací. Vzhled webového serveru je navržen tak aby umožnoval snadné ovládání pomocí mobilního telefonu.

Klíčová slova: PLC, automatizace, webový server

CONTROLLING HOME VIA PLC AND VISUALIZATION USING WEB SERVER

Abstract:

The bachelor thesis focuses on the possibilities of controlling the household through the web interface. Household control is implemented using a programmable logic controller (PLC). As the PLC is use the latest Siemens controller SIMATIC s7-1200. For controlling household is used web server. The web server is designed to be easy to use for communicating with other home automation devices. The appearance of the web server is designed to allow easy control over your mobile phone.

Keywords: PLC, automation, web server

Obsah

1. Úvod

Ve své bakalářské práci se zabývám možností ovládání některých prvků v domácnosti jako je například osvětlení nebo vytápění pohodlně pomocí webového aplikace. Ze začátku jsem se vydal cestou vývoje vlastního řešení postaveném na technologii asp.net. To se postupem času ukázalo jako špatná cesta. Nakonec jsem zvolil již existující platformu, do které jsem doprogramoval potřebné věci.
První část práce se zabývá stručným popisem použitých technologií. Popisuje použité komunikační protokoly a seznamuje čtenáře s problematikou programovatelných automatů.
Druhá část popisuje použitý webový server. Popisuje způsob, jakým byl vybrán hardware určený pro plynulý chod aplikace. Dále je zde popsán způsob výběru webového serveru pro vizualizaci domácnosti.
Třetí část se zabývá návrhem celého systému. Je zde popsáno, co všechno a v jakém rozsahu bude moci být ovládáno přes web. Dále je zde popsána topologie sítě, způsob komunikace webového serveru s PLC a rozvrženo pořadí jednotlivých prací.
Čtvrtá část se pak zabývá implementací programu pro PLC a kódu pro webový server.
Poslední kapitola se zabývá testováním celého systému.

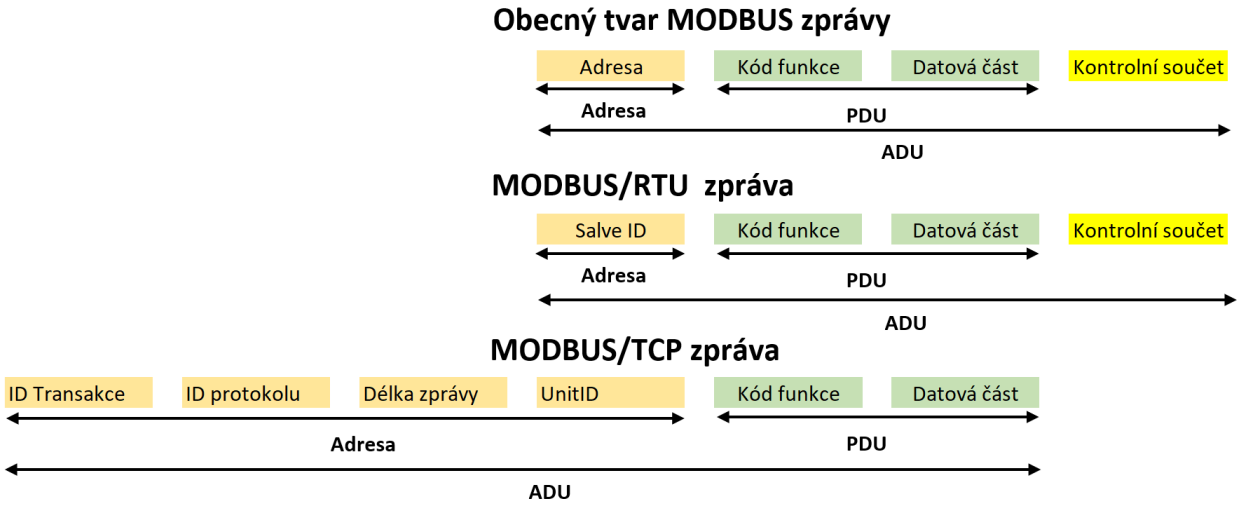
2. Popis použitých technologií

2.1. Modbus

MODBUS je komunikační protokol na úrovni aplikační vrstvy modelu ISO/OSI, který umožňuje komunikaci typu Master-Slave. Na sběrnici je jeden Master (v případě MODBUS TCP i více) a několik Slave zařízení. Master posíla dotazy na Slave ražízení a ty odpovídají. Matster bývá většinou řídící prvek například průmyslové PC nebo PLC. Slave bývají většinou čidla, měřící přístroje nebo PLC. Jako přenosová média MODBUS používá Ethernet (TCP/IP) nebo sériový přenos(RS-232, RS485, optické vlákno, atd.).

2.1.1. Popis protokolu

MODBUS definuje strukturu zprávy na úrovni protokolu(PDU - Protocol Data Unit) nezávislé na typu přenosového média. V závislosti na typu sítě je pak PDU rozšířena o další části a tvoří tak zprávu na aplikační úrovni (ADU - Application Data Unit)
Kód funkce udává zařízení typu slave (nebo Server) jakou operaci má provést. Rozsah je 0x01 až 0xff, přičemž kódy 0x80 až 0xff jsou vyhrazeny pro oznámení chyby.



Obrázek 2.1.: Tvar modbus zprávy

2.1.2. Datový model

Datový model je založen na tabulkách s charakteristickým významem. Jsou definovány čtyři základní tabulky Cívky, Diskrétní vstupy, Vstupní registry a Uchovávající registry. Mapování tabulek do adresního prostoru zařízení je závislá na konkrétním zařízení . Každá z tabulek může mít vlastní adresní prostor nebo se něj mohou dělit. Každá z tabulek může mít 65536 položek, ale z důvodu zpětné kompatibility bývá adresní prostor rozdělen na bloky o velikosti 10000 položek.

Tabulka	**datový typ**	**Přístup**	**Adresa (SIMATIC)**
Cívky (*Coils*)	bool	Čtení/Zápis	1 až 9999
Diskrétní vstupy(*Discrete Inputs*)	bool	Čtení	10001 až 19999
Vstupní registry (*Input Registers*)	WORD	Čtení	30000 až 39999
Uchovávací registry (*Holding Registers*)	WORD	Čtení/Zápis	40001 až 49999

Tabulka 2.1.: Datový model protokolu Modbus

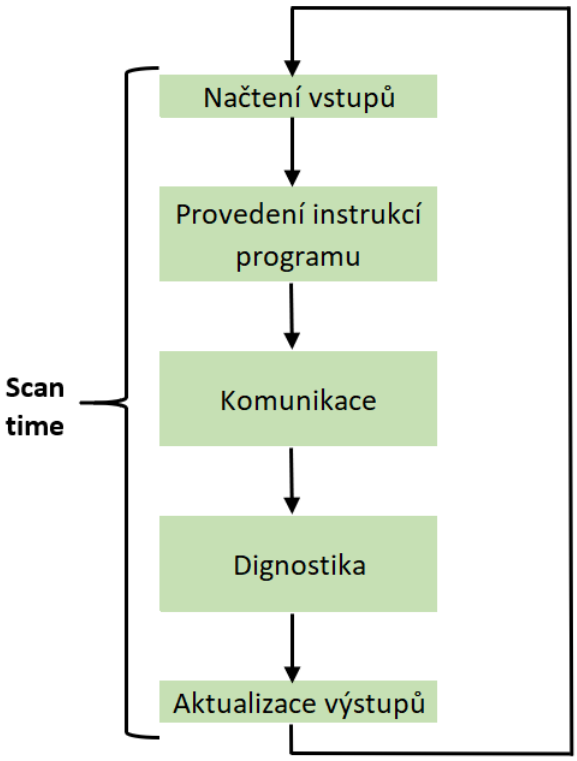
2.2. Programovatelný automat (PLC)

PLC (Programmable Logic Controller) neboli programovatelný logický automat je malý průmyslový počítač přizpůsobený pro automatizaci procesů v reálném čase jako například montážní linky, robotické zařízení nebo jakoukoliv činnost vyžadující vysokou spolehlivost, snadné programování a diagnostiku poruch procesu. Hlavní rozdíl mezi PLC a klasickými počítači je ten že PLC jsou určena pro náročné podmínky (jako je prach, vlhkost, chlad nebo teplo) a nabízí vstupy a výstupy (I/O) pro připojení senzorů a pohonů. Vstupy můžeme mít analogové (snímaní teploty, tlaku, vlhkosti, ...) nebo binární (koncové snímače, tlačítka). Výstupy výstup můžeme připojit např. sirény, elektromotory, solenoidy, relé nebo analogové výstupy.

Program v PLC se provádí v cyklech. Na začátku každého cyklu se stav všech fyzických vstupů zkopíruje do oblasti paměti někdy nazývané "I/O Image Table", která je přístupná procesoru. Procesor poté čte instrukce programu a provádí jednotlivé operace s daty. Po přečtení poslední instrukce procesor zpracuje požadavky z komunikací (modbus , s7), provede diagnostiku a aktualizuje výstupy. Délka celého procesu se nazývá scan time. [1]

2.2.1. SIMATIC S7-1200

SIMATIC S7-1200 je modulární mikrokontrolér pro menší a střední automatizační úlohy. Kontroler se skládá z napájecího zdroje, samotné řídící jednotky a vstupně/výstupních mo-



Obrázek 2.2.: scan time

dulů pro digitální a analogové signály. Toto PLC je vybaveno 1MB pamětí pro uživatelský program a 25KB operační paměti.
Paměťový prostor kontroleru je rozdělen na pět částí:

- Process image input (I) - Na začátku každého cyklu je do této paměti zkopírován stav všech fyzických vstupů.

- Process image output (Q) - Na začátku každého cyklu je obsah této paměti zkopírován na fyzický výstup.

- Bit Memory (M) - Data zapsána do Bit memory jsou přístupná odkudkoliv z uživatelského programu.

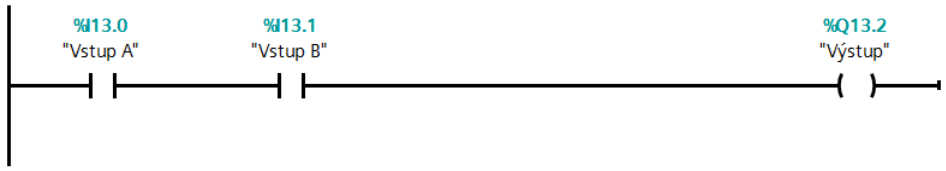
- Temp memory (L) - Kdykoliv je zavolán některý blok kódu je mu přiřazena část této paměti.

- Data block (DB) - Paměť určená pro ukládání dat.

Pro programování kontroleru S7-1200 lze použít tři programovací jazyky LAD, FBD a SCL.

LAD

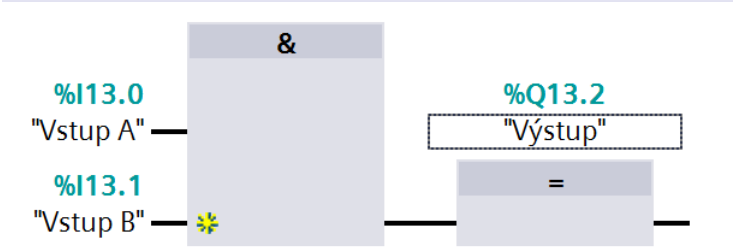
LAD neboli ladder logika je vizuální programovací jazyk založený na konceptu reprezentace programu pomocí reléové logiky. Program v jazyku LAD je zleva i zprava ohraničena svislými čarami, které reprezentují napájení. Mezi nimi je pak samotná logika programu.



Obrázek 2.3.: Funkce AND zapsána pomocí jazyka LAD

FBD

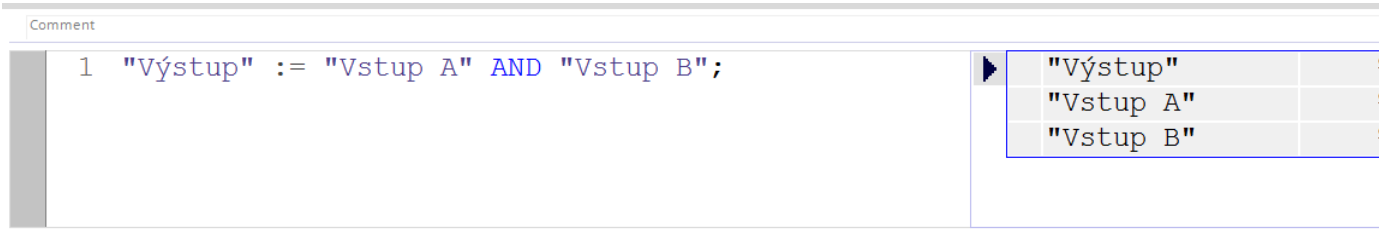
FBD nebo-li Function Block Diagram je jazyk který popisuje program jako soubor vzájemně propojených grafických bloků obdobně jak je tomu u elektronických obvodovích diagramech.



Obrázek 2.4.: Funkce AND zapsána pomocí jazyka FBD

SCL

SCL je jazyk vhodný pro programování složitých algoritmů a aritmetických funkcí nebo pro zpracová dat. SCL kombinuje prvky známé z vyšších programovacích jazyků jako jsou například cykly nebo větvení a prvků známích z jazyků typických pro PLC jako je například adresování vstupů a výstupů.



Obrázek 2.5.: Funkce AND zapsána pomocí jazyka SCL

2.3. Z-Wave

Z-Wave je bezdrátová technologie vyvinutá pro inteligentní řešení ovládání domácnosti. Umožňuje komunikaci mezi elektronickými zařízeními v domácnosti, jejich vzdálenou kontrolu a ovládání. Pro ovládání Z-Wave zařízení je zapotřebí řídící jednotka tzv. Z-Wave Gateway. V současné době Z-Wave podporuje více než 1700 výrobků od více než 450 výrobců. V Evropě používá Z-Wave frekvenci 868.42 MHz. Maximální přenosová rychlost je 100 Kbit/s. Maximální vzdálenost mezi dvěma zařízeními je 100 metrů. Pokud zařízení nemá přímé spojení s řídící jednotku může komunikovat až přes 4 další z-wave zařízení která fungují jako z-wave repeater celková vzdálenost pro komunikace je maximálně 200 metrů.

2.4. 1-Wire

Tato sběrnice byla navržena firmou Dallas Semiconductor. Umožnuje připojit zařízení k řídící jednotce pomocí dvou vodičů. Při komunikaci na sběrnici se využívá model Master-Slave kde master je pouze jeden a bývá to většinou mikroprocesor. Slave zařízení může být na sběrnici připojeno více. Pro zapojení 1-wire zařízení se většinou používají tři vodiče. Napájecí vodič, datový vodič a zem. Datový vodič je spojen s napájecím vodičem pomocí 5kΩ odporu.
Komunikaci zahajuje vždy master a to tak, že datový vodič uzemní (to způsobí že se na vodiči objeví logická 0) a v tomto stavu ho drží minimálně 480 mikrosekund. Poté datový vodič uvolní a naslouchá. Díky odporu se datový vodič vrátí do stavu log. 1. Pokud je na sběrnici připojeno nějaké zařízení tak tuto změnu detekuje a uvede datový vodič po dobu 60-240 mikrosekund do stavu log. 0 a tím se ohlásí řídící jednotce. Pokud se zařízení ohlásí tak master může vysílat nebo přijímat data.
Data jsou posílána v časových intervalech dlouhých 60 mikrosekund. V každém intervalu je poslán jeden bit. Pokud chce master poslat log. 1 tak na 1 – 15 mikrosekund vyšle na datový vodič log. 0 a zbytek intervalu ponechá log. 1. Posílání log. 0 probíhá tak že master pošle na datový vodič log.0 a ponechá ji tam po celý interval.
Čtení probíhá tak, že master vyšle na datový vodič log. 0 a to po dobu 1 mikrosekundy. Poté může zařízení buď to vysílat log. 1 nebo log. 0.

Každé 1-wire zařízení obsahuje unikátní 64bitový identifikační kód. Pokud máme na síti více zařízení, musí se před posláním dat nejprve poslat adresa zařízení, pro které jsou data určena.

2.5. ESP8266

ESP8266 je označení pro řadu mikropočítačů vyráběných společností Espressif Systém. Série ESP8266 v současné době obsahuje dva čipy ESP8266EX a ESP8285.
ESP8266EX je Soc, který integruje 32bitový mikroprocesor Tensilica, standartní digitální rozhraní (digitální vstupní a výstupní piny, různé typy sběrnic), anténu a modul pro řízení spotřeby. Obsahuje 2,4 GHz WiFi s podporou standardů 802.11 b/g/n.
ESP8285 je variantou ESP8266 s 1024 KB Flash pamětí.
Při použití ESP8266 máme k dispozici 32bit mikroprocesor s frekvencí 80 MHz 80 KB operační paměti. Dále je vybaven Flash pamětí pro ukládání programu. Velikost této paměti se pohybuje od 521 KB do 4 MB. Dále je k dispozici 16 Vstupně/výstupních pinů, SPI, I²C a UART sběrnice.
Programovací jazyk pro vytváření programů pro EPS8266 je závislý na použitém firmwaru. Mezi nejpoužívanější firmware patří: NodeMCU (využívá jazyk LUA), Arduino (C++) a MicroPython.

2.6. MQTT

MQTT je jednoduchý komunikační protokol určený pro internet věcí. Původně byl navržen firmou IBM, ale dnes již ve správě konsorcia Eclipse foundation. Protokol na předávání zpráv mezi klienty prostřednictvím centrálního serveru (tzv. broker). Server sbírá zprávy od poskytovatele zpráv (tzv. publisher) a předává je klientům které o ně mají zájem (tzv. subscribers). Jeden broker může mít libovolné množství poskytovatelů zpráv. Jakýkoliv klient může být jak Publisher, tak Subsriber. Publisher bývá snímač nebo měřící jednotka, která posílá naměřená data na server a subscriber tvoří řídící jednotka, která tyto hodnoty zpracovává.

Zprávy jsou rozděleny do témat (topic), přičemž každá zpráva patří právě do jednoto tématu. Témata jsou hierarchická a oddělená lomítky. Obsah zpráv není nijak definován, obsahem zprávy mohu tak být data libovolného formátu. Velikost jedné zprávy je v současné době omezena na 256 MB.

3. Specifikace a volba webového serveru

Běh webového serveru zajišťuje jednodeskový počítač Raspberry Pi 3. Počítač je osazen čtyřjádrovým procesorem Broadcom BCM2837 s taktem 1.2 GHz a operační pamětí 1GB. Do počítačové sítě je připojen pomocí Ethernetu, který je integrován na desce počítače a podporuje maximální rychlost 100Mbit/s, která je v našem případě dostatečná. Důvodem pro výběr Raspberri Pi byla nízká cena, a především vysoká dostupnost.
U Raspberry pi 3 máme na výběr ze dvou operačních systému Windows 10 a Linux. Při testování se distribuce Windows 10 Iot určená speciálně pro Rasbberry Pi ukázala jako nevhodná, především kvůli vysoké nestabilitě. Byl tedy použit operační systém Raspbian. Jedná se oficiální Linuxovou distribuci pro Raspberry Pi.

3.1. Volba webového serveru

PPři tvorbě webového serveru pro vizualizaci jsou na výběr celkem tři možnosti, jak postupovat:

- Naprogramovat si vlastní webový server
- použít SCADA systém
- použít již existující platformu pro domácí automatizaci

Výhoda SCADy je v tom že je stavěná na propojení s PLC a má v sobě už všechny potřebné protokoly. SCADA systémy jsou učeny pro vizualizaci procesů v průmyslu, a z toho taky vyplývá jejich nevýhoda pro použití doma a to je vysoká cena.
Nejideálněji řešení se nabízí použít už některou z vytvořených platforem pro domácí automatizaci. Taková to platforma by měla mít následující vlastnosti:

- Responzivní vzhled
- snadná tvorba pluginů
- aktivní komunita
- kvalitní dokumentace

Výše zmíněným podmínkám vyhovují dvě platformy. Home Assistant a OpenHAB. Obě tyto platformy jsou rovnocenné. Nakonec byl zvolen Home Assistant a to z důvodu podpory více pluginů a rychlejšího vývoje.

3.2. Home Assistant

Home Assistant je open-source platforma určená pro domácí automatizaci. Home assistant využívá programovací jazyk python3. Díky tomu může běžet na všech operačních systémech, které mají potporu pythonu3. Rozšířením pro Home Assistant se říká komponenty. V aktuální verzi (0.66.1) je k dispozici 1036 komponent.

3.2.1. Instalace

Pro instalaci nejnovější verze Home Assistantu je požadovýna minimální verze pythonu 3.5.3. Déle je doporučeno nainstalovat Home Assistant do virtuálního prostředí.

- Vytvoření virtuálního prostředí

```
1        $ python3 -m venv homeassistant
2
```

- otevení vituálního prostředí

```
1        $ cd homeassistant
2
```

- Aktivace virtuálního prostředí

```
1        $  source bin/activate
2
```

- instalace knihovny wheel

```
1      $ python3 -m pip install wheel
2
```

- instalace Home Assistantu

```
1        $ python3 -m pip install homeassistant
2
```

- Spuštění

```
1        $ hass
2
```

3.2.2. Konfigurace a přidávání komponent

Home Assistant ukládá konfiguraci do souboru s názvem *configure.yaml*. Tento soubor se vygeneruje automaticky při prvním spuštění aplikace. Některé věci je možné upravovat přímo z uživatelského rozhraní, ostatní vyžadují ruční úpravu konfiguračního soubor. Pro editaci konfigurační souboru je dobré použít textový editor s podporou YAML syntaxe například Visual Studio Code nebo Atom. Pokud je třeba například přidat senzor, který bude generovat náhodné hodnoty, stačí do konfiguračního souboru přidat následují dva řádky:

```
1  senzor:
2      platform: random
3
```

Seznam všech komponent a vzorové příklady jak je použít je k nalezení na officiálních stránkách Home Assistantu.

3.2.3. Psaní automatizací

Home Assistant nabízí tvorbu tzv. automatizací což je prvek který lze přirovnat například k tvorbě eventů ve vyšších programovacích jazycích. Automatice se skládají ze tří části: Trigger, Condition a Action.
Trigger (česky spouštěč) popisuje událost, která pokud nastane tak dojde ke spuštění automatizace. Může se jednat například o osobu, která přijde domu. To je v Home Assistentu reprezentováno změnou stavu u dané osoby z „not_home“ na „home“.
Conditions (česky podmínky) jsou volitelné testy, které mohou omezit automatizaci, aby fungovala jenom v konkrétním případě. Například budeme chtít reagovat, pouze pokud je po západu slunce.
Třetí a poslední částí je **Action** (česky akce). Zde se definuje akce, která se provede při spuštění triggeru a zároveň splnění všech doplňujících podmínek. Může se jednat například o změnu teploty na termostatu.
Níže uvedený kód je příklad zápisu automatizace, která pokud osoba přijde domu po západu slunce tak rozsvítí světla. Pro detekci je zde použita komponenta device_tracker, která sleduje zařízení připojená k WiFi.

```
1   automation:
2     alias: Zapni svetla kdyz prijdu domu
3     trigger:
4       platforn: state
5         entity_id = device_tracker.frantisek_mobil
6         from: 'not_home'
7         to: 'home'
8     condition:
9       condition: sun
10      event: sunset
```

```
        offset: "-1:00:00"
      action:
        service: light.turn_on

```

3.2.4. Psaní scriptů

Scripty jsou sekvence akcí, které Home Assistant vykoná. Po vytvoření je script k dispozici jako samostatná entita, ale může být také vložený do automatizací. Stejně jako automatizace se i scripty píší ve značkovacím jazyce YAML. Základní struktura scriptů je seznam klíčů a k nim přiřazených hodnot, které obsahují jednotlivé akce.
Níže je příklad scriptu, který svém zavolání zapne světla v koupelně a poté pošle upozornění s textem „Světla v koupelně byla rozsvícena“. Pokud script obsahuje pouze jednu akci tak můžeme element „sequence“ vynechat.

```
Script:
  Muj_prvni_script:
    Sequence:
      - Service: light.turn_on
        Data:
          Entity_id: light.svetla_koupelna
      - Service: notify.notify
        Data:
          Message: "Svetla v koupelne byla rozsvicena

```

3.2.5. Tvorba vlastních rozšíření

I když Home Assitant v základu obsahuje více než tisíc již vytvořených komponent, ne vždy tam najdeme to co potřebujeme. Proto není špatné se alespoň trochu seznámit s tvorbou vlastních komponent. Každé rozšíření musí obsahovat objekt DOMAIN. V něm je uložen název komponenty a měl by se shodovat s názvem souboru. Dále soubor musí obsahovat metodu *setup(hass, config)*. Tato metoda se volá pouze jednou a to při načtení komponenty. Jak už název napovídá uvnitř této metody se provádí inicializace našeho rozšíření. Množství parametrů metody setup se muže lišit v závislosti na typu rozšíření, ale parametry hass a config jsou přítomny vždy.
Základní objekty:

- *hass* - tento objekt obsahuje instanci Home Assistantu
- *hass.config* - v tomto objektu je uložena základní konfigurace Home Assistantu jako například umístění konfiguračního souboru

- *hass.states* - objekt umožňující měnit stavi jednotlivých entit (např. světel) nebo sledovat jestli byli změněny
- *hass.bus* - tento objekt umožňuje aktivovat nebo sledovat eventy
- *hass.services* - v tomto objektu můžeme registrovat služby

Ve níže uvedeném kódu je příklad komponenty, která reprezentuje teplotní čidlo.

```
from homeassistant.const import TEMP_CELSIUS
from homeassistant.helpers.entity import Entity
import random
DOMAIN = 'example'
#Inicializace platformy
def setup_platform(hass, config, add_devices, discovery_info=None):

  #Vytvoreni dvou instanci tridy ExampleSenzor (Vytvoreni dvou cidel)
  cidlo1 = ExampleSenzor()
  cidlo2 = ExampleSenzor()
  #objekt add_divice slouzi k registraci entit.
  add_devices([cidlo1 , cidlo2])

#Trida ExampleSensor reprezentuje entitu senzoru
class ExampleSensor(Entity):

  def __init__(self,name):
    #Inicializace senzoru.
    self._name = name
    self._state = None

  @property
  def name(self):
      #Tato metoda vraci nazev cidla
    return self._name

  @property
  def state(self):
    # Metoda vraci stav cidla. Napr. pokud se jedna o teplotni cidlo, tak vraci
    posledni namerenou hodnotu
    return self._state

  @property
  def unit_of_measurement(self):
    # Tato metoda vraci jednotky ve kterych cidlo provadi mereni. TEMP_CELSIUS je
    konstanta Home Assistantu
    return TEMP_CELSIUS

  def update(self):
    #Po zavolani teto metody dojde k aktualizaci merene hodnoty. Zaroven je to jedina
     metoda, ktera by mela ziskavat data pro Home Assistant
    self._state = random.randint(1,50)
```

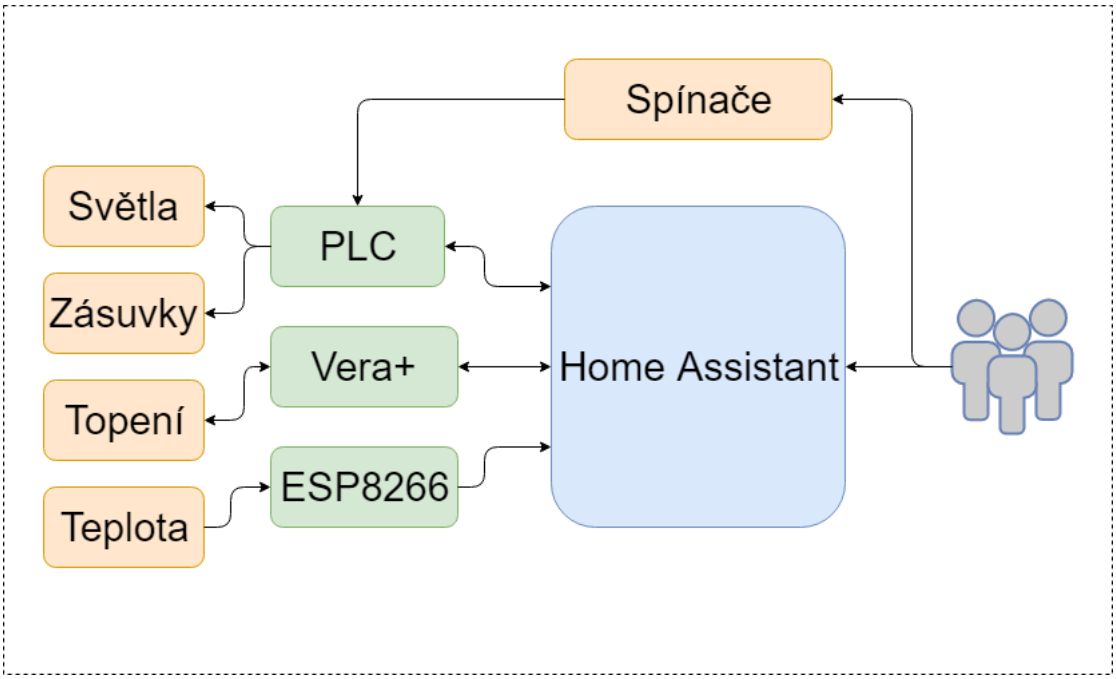
4. Návrh systému a způsob provedení

4.1. Popis provedení

V rámci této bakalářské práce bude realizována automatizace prvního patra rodiného domu. Všechna světla a některé zásuvky bude možné ovládat z webového rozhraní. Světla na wc a koupelně se budou automaticky po hodině vypínat.
Na schodišti budou umístěna dvě difuzní čidla pro detekci pohybu, která budou zajišťovat automatické rozsvícení světel pokud bude zaznamenán pohyb. Světla na schodišti se automaticky po třiceti minutách vypnou.
V obývacím pokoji a na chodbě bude umístěno intimní osvětlení, které se každý večer ve stanovený čas zapne a druhý den ráno v určité době vypne. Bude zde také umístěno teplotní čidlo pro měření pokojové teploty.
Na radiátorech budou umístěny termostatické hlavice, které budou regulavat teplotu v místnosti.

4.2. Topologie

Světla a zásuvky budou ovládané pomocí PLC. To se bude dát ovládat dvěma způsoby. První způsobem je pomocí webové rozhraní Home Asssistantu. Druhý způsob je ovládání pomocí spínačů umístěných na zdi. To nám zaručí, že v případě výpadku serveru bude možné pořád ovládat světla.
Ovládání vytábění bude realizováno pomocí termostatických hlavic od firmy Danfoss. Ty budou komunikovat s řídící jednotkou Vera Plus pomocí technologie Z-Wave. Vera Plus bude sloužit jako gateway pro všechna Z-Wave zařízení. Komunikace mezi Home Assistantem a Vera Plus bude realizováno pomocí Ethernetu. Pro vyčítaní dat z Vera+ bude použito již hotové rozšíření pro Home Assistant.
Měření teploty realizováno pomocí čidel DS18B20, které komunikují pomocí sběrnice 1-Wire. Data z čidel budou vyčítána pomocí ESP8266 a dále posílaná do Home Assistantu pomocí protokolu MQTT.



Obrázek 4.1.: Topologie

4.3. Pracovní postup

Jako první je třeba připravit nový rozvaděč ve kterém bude umístěno PLC, napájecí obvody a jističe. Poté co bude nový rozvaděč na místě dojde k demontáži stávající elektroinstalace. V rámci nové elektroinstalace budou veškeré spínače a osvětlení svedeno do rozvaděče a napojeny na PLC. Jakmile bude elektroinstalace hotova bude vytvořen program pro PLC. Po dokončení PLC programu přijde na řadu instalace a konfiguraci Home Assistantu. Jakmile bude odladěna komunikace mezi PLC a Home Assistantem přijde na řadu zprovoznění vytápění.

4.4. Značení I/O

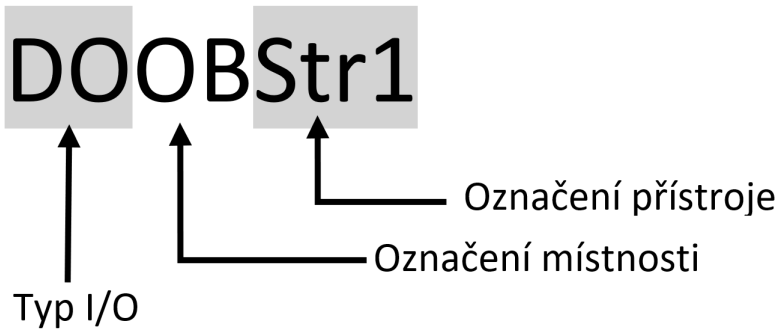
Digitální vstupy a výstupy jsou v PLC reprezentovány proměnou typu Bool a hardwarovou adresou. Adresa se skládá ze tří částí, první část tvoří písmeno, které nám říká jestli se jedná o vstup (I) nebo výstup (Q), následuje číslo Bytu v paměti a následně číslo bitu. Například I9.5 znamená že se jedná o digitální vstup, který najdeme v pátém bitu v devátém bajtu.
Analogové vstupy a výstupy jsou reprezentovány proměnou typu Word a jejich hardwarová adresa je ve tvaru IW10. První písmeno udává jestli se jedná o vstup (I) nebo výstup (Q), druhé písmeno nám říká že jde o Word tzn. Analogovou hodnotu a číslo označuje adresu v paměti kde je hodnota uložena.
Před zahájením psaní programu je dobré si nejdříve vytvořit IO List, který bude obsahovat

zkratka	úplný název	zkratka	úplný název
DI	digitální vstup	Spi	Spínač
DO	digitální výstup	Str	Stropní světlo
AI	analogový vstup	Ste	Nástněnné osvětlení
AO	analogový výstup	Led	LED osvětlení
OB	Obývací pokoj	Cid	pohybové čidlo
LO	Ložnice	Zas	Zásuvka
PR	Pracovna		
KO	Koupelna		
WC	Záchod		
BA	Balkón		

Tabulka 4.1.: Seznam zkratek

seznam všech vstupů a výstupů, datový typ, adresu a popřípadě komentář. Dále je dobré si stanovit metodiku pojmenování IO tak aby název každého IO byl v rámci programu unikátní a aby se z něj dalo na první pohled zjistit k čemu slouží.

zvolená metodika



Obrázek 4.2.: Značení přístojů

První dvě písmena v názvu určují typ I/O. Poté následují dvě písmena s onznačením místnosti. Počet posledních znaků není pevně stanoven, obsahuje minimálně tři písmena s označením přístroje a pokud je v místnosti více stejných přístrojů tak je zde ještě číslo přístroje.

4.5. Přenos dat mezi PLC a serverem Home assistant

Na výstupní svorky PLC máme připojeno celkem 28 zařízení. Všechna zařízení se ovládají pomocí binárních signálů, k ovládání každého zařízení nám tedy bude stačit pouze jeden bit. Velikost holding registru je 16 bitů. Pro přenos ovládání všech zařízení nám tedy budou stačit dva registry. Přenos dat z PLC do Home assistantu bude obstarávat protokol Modbus TCP. PLC bude sloužit jako Modbus server a Home assistant jako klient. Aby PLC mohlo fungovat jako Modbus server stačí použít už připravený bloček MB_SERVER, který je součástí standartní knihovny v prostředí TIA PORTAL V14. Po vložení stačí pouze nastavit potřebné parametry.
Na straně Home assistantu je situace komplikovanější. Home assistant sice obsahuje rozšíření pro práci s protokolem Modbus, to však umí do jednoho holding registru zapsat pouze informaci o stavu jednoho zařízení. Při použití tohoto rozšíření bychom tedy potřebovali 28 registrů. Takové množství dat by způsobilo viditelné zpomalení celého programu.

4.5.1. Návrh modbus rozšíření pro Home Assistant

Rozšíření je složeno ze dvou soborů *plc_modbus.py* a *modbus_sw.py*.

plc_modbus.py

Tento soubor obsahuje vše potřebné pro komunikaci s PLC pomocí protokolu ModbusTCP. Pro komunikaci je použita knihovna *pymodbus*.
setup_platform - rozšíření bude poskytovat Home Assistentu dvě služby. Službu *handle_write* po jejímž zavolaní boudou do PLC zapsány všechny registry ve kterých došlo ke změně hodnoty registru. Dále bude poskytovat službu *handle_read*, která má za úkol získat skutečný stav registrů uložených v PLC.

RegisterHandler - Třída představuje jeden register. Tato třída nekomunikuje s PLC přímo, ale využívá při tom metody třídy *Communication*

- **register_number** - v tomto objektu je uloženo číslo registru v PLC, kterému tento registr odpovídá.
- **ip** - Ip adresa PLC ve kterém je registr uložen.
- **port** - Port přes který se má komunikovat s PLC (V PLC je pro každý registr vyhrazen jeden port, aby mohla komunikace probíhat paralelně).
- **reg_value** - 16bitová hodnota která představuje číselnou hodnotu registru.

- **status_change** - Informace o tom jestli od poslední synchronizace s PLC došlo ke změně registru.
- **write_reg()** - Slouží pro zápis registru do PLC. Zápis probíhá pouze v případě, že proměná (status_value) má hodnotu *True* (došlo ke změně v registru)
- **read_reg()** - Metoda pro čtení dat z PLC. Díky této metodě je zajištštěno, že *RegisterHandler* a příslušný registr v PLC budou mít stejnou hodnotu
- **set_bit()** - Pokud dojde ke stisku tlačítka na webovém rozhraní, tak tlačítko zavolá tuto metodu a ta změní hodnotu příslušného bitu v proměné *reg_value*
- **status()** - Tato metoda vrací aktuální stav konkrétního bitu v proměnné *reg_val*

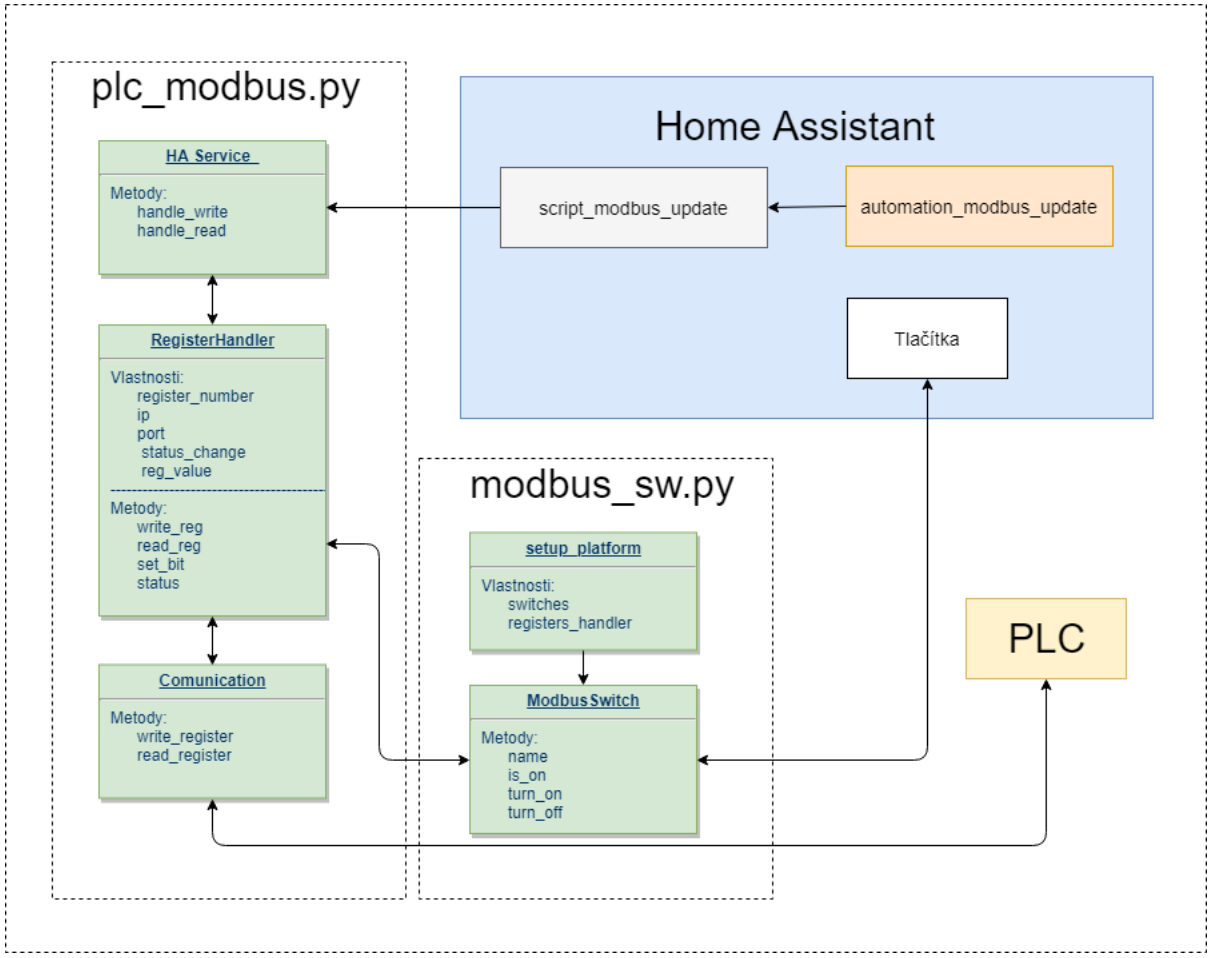
Communication - Tato třída má na starosti přenos dat mezi Home Assistantem a PLC. K přenosu dat využívá knihovnu *pymodbus*. Třída obsahuje metodu *write_register* pro zápis hodnoty jednoho registru do PLC. A metodu *read_register* pro přečtení hodnoty jednoho registru z PLC. Pokud by v budoucnu došlo ke změně komunikačního protokolu. Například z protokolu Modbus TCP na S7 tak stačí změna imlementace této třídy.

modbus_sw.py

Soubor modbus_sw.py obsahuje metody a třídy potřebné pro implementaci spínačů napojených na komunikaci Modbus TCP.
setup_platform - Tato metoda se zavolá při spuštění Home Assistantu a má na starosti vytvoření tlačítek. Vytváří instace třídy *RegisterHandler*. Každé instanci přidá ip adresu PLC, číslo registru a port. Dále přiřadí tlačítka k jednotlivým registrům.
ModbusSwitch - Tato třída reprezentuje jeden spínač. Obsahuje tyto metody:

- **name** - Metoda vrací název tlačítka
- **is_on** - Tato metoda vrací aktuální stav tlačítka
- **turn_on** - Zapne tlačítko
- **turn_off** - Vypne tlačítko

Pro synchronizaci dat slouží script *script_modbus_update* tento script volá metody *handle_write* a *handle_read*. Volání scriptu každou vteřinu zajištuje automatizace *automation_modbus_upda*



Obrázek 4.3.: modbus komunikace s HA

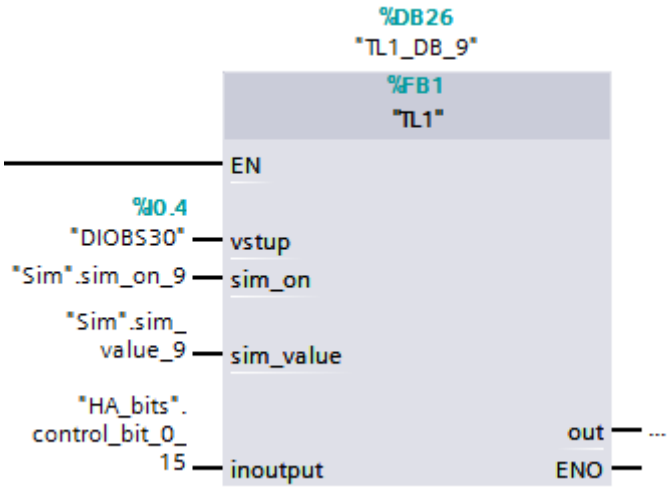
5. Implementace

5.1. Ovládání výstupů

Jeden ze základních požadavků na systém je, aby se veškeré spotřebiče připojené k PLC dali ovládat jak místně tak pomocí webové stránky. V programu jsou pro tento problém implementovány dva bločky.

5.1.1. TL1

První z nich TL1 slouží pro ovládání jednoho zařízení pomocí jednoho hardwarového vstupu a pomocí komunikace.



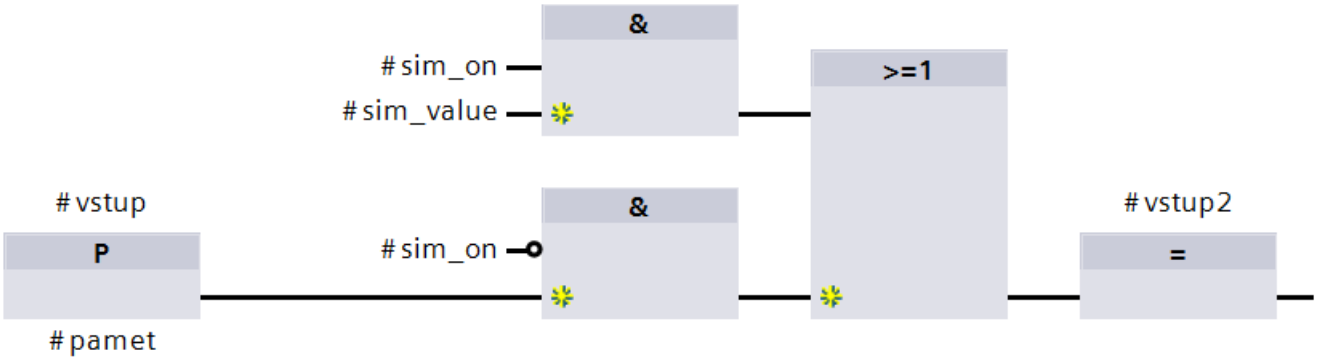
Obrázek 5.1.: TL1

Bloček umožňuje simulovat hardwarový vstup, to je užitečné zejména v případě že potřebujeme otestovat funkčnost ještě před tím, než připojíme hardwarový vstup, nebo pokud se vstup nachází na rozšiřující kartě, která z nějakého důvodu není k dispozici.
Výstupní signál můžeme připojit dvěma způsoby. Připojit hardwarový výstup přímo na vý-

stup bločku (*out*), nebo to udělat později v programu a použít k tomu proměnou která je přiřazená k *inoutput*.

Vnitřní impementace TL1

Pro přehlednost je vnitřní struktura TL1 rozdělena na čtyři části. V první části se hodnota z *inoutput* přiřadí k proměnné *pametBit*.
Druhá část obsahuje implementaci režimu simulace. Pokud je *sim_on* nastaven na True tak se do proměnné *vstup2* uloží hodnota *sim_value*. Pokud ne uloží se tam hodnota proměnné *pamet*. První bloček z leva nám zajistí to, že se hardwarový vstup uloží do proměnné *pamet*. Písmeno P znamená, že do proměnné *pamet* se uloží True pouze tehdy mění-li se signál na hardwarovém vstupu z False na True (náběžnou hranu). Poslední část programu obsahuje

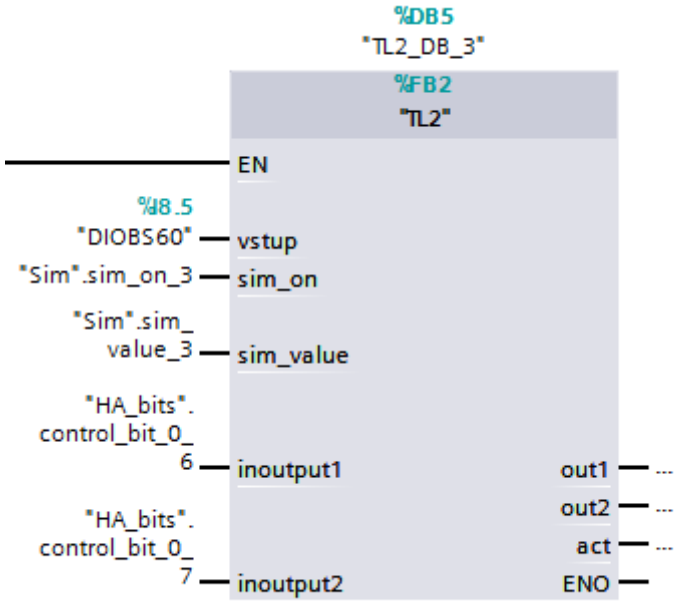


Obrázek 5.2.: TL1 simulace

logiku pro ovládání výstupu. Ta je realizována pomocí funkce XOR

5.1.2. TL2

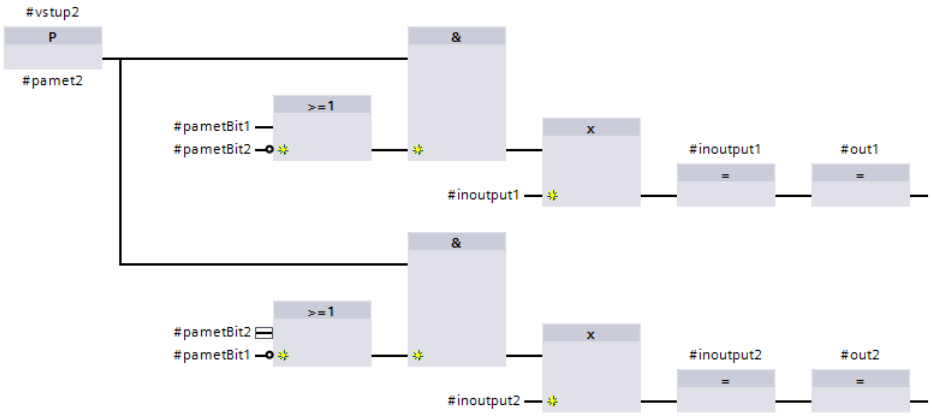
Druhý bloček s názvem TL2 slouží k ovládání dvou zařízení pomocí jednoho tlačítka, ale zároveň umožnuje přes webové rozhraní ovládat každé zařízení zvlášť. Od TL1 se liší tím, že vyžaduje dvě proměnné pro komunikaci s webovým serverem, a kromě dvou výstupů pro zařízení (out1 a out2) obsahuje ještě výstup act. Ten nabývá hodnoty True pokud je aktivní alespoň jeden z výstupů.



Obrázek 5.3.: TL2

Vnitřní impementace TL2

Pro lepší přehlednost je program rozdělen na tři části. Z nichž první dvě jsou totožné s TL1. třetí část, která obsahuje logiku se od logiky pro TL1 liší tím, že musí zajistit aby pokud je jeden z výstupů aktivní a dojde ke stisknutí tlačítka musí se oba výstupy nastavit na hodnotu False.



Obrázek 5.4.: TL2 logika

5.2. Implementace Modbus rozšíření

5.2.1. RegisterHandler

```
class RegisterHandler:
  register_number = 0
  ip = "192.168.1.5"
  port = 502
  reg_value = 15
  change = False
  def __init__(self,register,ip,port):
    self.register_number = register
    self.ip = ip
    self.port = port
    self.reg_value = Comunication.readRegister(self.ip, self.port, self.
    register_number)

  def write_reg(self):
    if self.change:
      Comunication.writeRegister(self.ip, self.port, self.register_number, self.
    reg_value)
      self.change = False

  def status(self,index):
    mask = 1<<index # spocita masku
    value = mask & self.reg_value # zjisti hodnotu pozadovan ho bitu
    if value == 0:
      return False
    else:
      return True

  def read_reg(self):
    if not self.change:
      self.reg_value = Comunication.readRegister(self.ip, self.port, self.
    register_number)

  def set_bit(self,index,x):
    mask = 1 << index  # Spocita masku
    self.reg_value &= ~mask  #  Smaze hodnotu po adovan ho bitu
    if x:
      self.reg_value |= mask  # Pokud X = True tak nastavi po adovan  bit na 1
      self.change = True
```

5.2.2. Comunication

Tato třída implementuje kumunikaci s PLC pomocí protokolu Modbus TCP. Komunikace probíhá tak, že se naváže spojení s PLC. Poté se provede zápis/čtení registru do/z PLC. Po přenosu dat se spojení se PLC ukončí.

```
class Comunication:
  def writeRegister(ip, port,register, value):
    client = ModbusTcpClient(ip,port)
    client.write_register(register, value)
    client.close()
    return True

  def readRegister(ip, port,register):
    client = ModbusTcpClient(ip, port)
    result = client.read_holding_registers(register, 1)
    client.close()
    return result.getRegister(0)
```

5.2.3. ModbusSwitch

Implementace třídy *ModbusSwitch* odpovídá svými metodami požadavkům na spínače, které jsou uvedeny v dokumentaci Home Assistantu. Pokud dojde ke stisknutí spínače zavolá se metoda *turn_on* popřípadě *turn_off*. Tyto metody změní příslušný bit ve svém registru. Každý spínač obsahuje odkaz na registr, ve kterém se nachází.

```
class ModbusSwitch(SwitchDevice):

  def __init__(self, name, register,index):
    self._name = name
    self.register = register
    self.index = index
    self._state = False

  @property
  def name(self):
    return self._name

  @property
  def is_on(self):
    self._state = self.register.status(self.index)
    return self._state

  def turn_on(self):
    self.register.set_bit(self.index,1)
    self.is_on

```

```
22
23   def turn_off(self):
24     self.register.set_bit(self.index, 0)
25     self.is_on
```

5.2.4. Služby

Při zavolání služby *handle_read* nebo *handle_write* dojde k postupnému zavolání metod *write_reg* nebo *read_reg* u všech registrů.

```
1   def setup(hass, config):
2
3     def handle_Write(call):
4       for r in registers:
5         r.write_reg()
6
7     def handle_Read(call):
8       for r in registers:
9         r.read_reg()
10
11 hass.services.register(DOMAIN, 'plc_Write', handle_Write)
12 hass.services.register(DOMAIN, 'plc_Read', handle_Read)
13 return True
```

5.2.5. Inicializace

Kromě standartních vstupních parametrů *hass* a *config* vstupuje do metody *setup_platform* ještě *add_device*. Tento objekt slouží k registraci nových zařízení. V poli *switches* jsou uloženy všechny spínače.
První cyklus vygeneruje dva registry. Oba budou mít stejnou IP adresu (jsou umístěny na stejném PLC). Každý z registrů komunikuje s PLC na jiném portu.
Ve druhém cyklu, se pro každý registr vytvoří 16 spínačů. Názvy spínačů jsou ve tvaru switch.x_y. Kde *x* představuje číslo registru a *y* představuje číslo bitu.

```
1   def setup_platform(hass, config, add_devices, discovery_info=None):
2     registers = plcModbus.registers
3     switches = []
4     #simatic s7-1200 - prvni patro
5     for i in range(0,2):
6       r = plcModbus.RegisterHandler(i,"192.168.0.11", 502+i)
7       registers.append(r)
8       for spinac in range(0,16):
9         # _LOGGER.info(a)
10        switches.append(ModbusSwitch(str(i)+"_"+str(spinac),r,spinac))
11
```

```
12      add_devices(switches)
```

5.3. Konfigurace Home Assistantu

5.3.1. skupiny

Home Assistant umožňuje sdružovat entity do skupin. Tyto skupiny se pak zobrazují na stejné kartě. V našem případě vytvoříme pro každou místnost samostatnou skupinu. Níže je ukázka, jak vypadají skupiny pro WC a koupelnu.

```
1  wc:
2    name: Zachod
3    entities:
4      - switch.1_9
5
6  koupelna:
7    name: Koupelna
8    entities:
9      - switch.1_7
10     - switch.1_8
11
```

5.3.2. Úprava Entit

Protože entita s názvem *switch.1_9* nám sice řekne na kterém bitu a ve kterém registru se tlačítko nachází, ale pro každodenní používání to není příliš vhodné pojmenování. Naštěstí Home Assistant umožňuje ke každé entitě přiřadit název, který se bude zobrazovat na webovém rozhraní. Kromě názvu jde změnit i ikona. Například pro entitu, která ovládá světlo je dobré, aby se zobrazovala ikona žárovky.

```
1 switch.1_9:
2   friendly_name: Svetlo
3   icon: mdi:lightbulb-on-outline
4
```

5.3.3. Konfigurační soubor

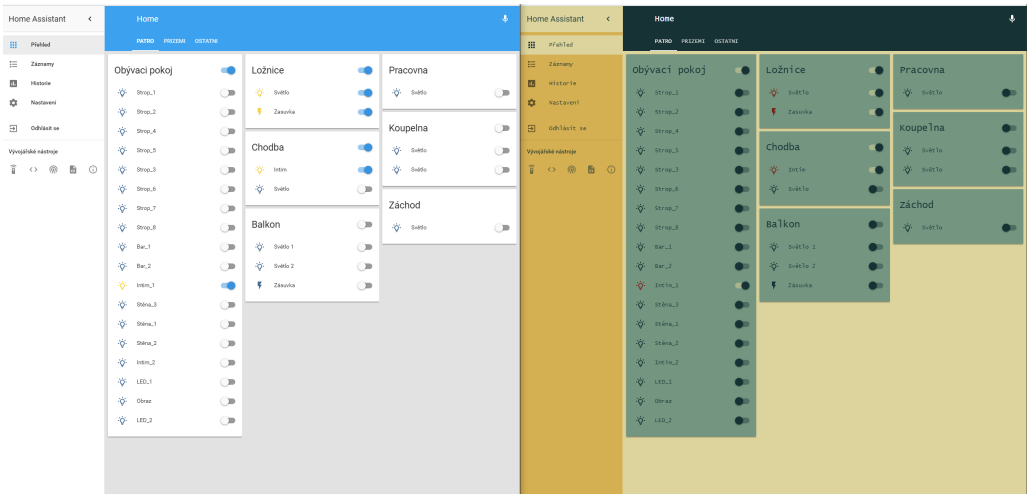
Do konfiguračního souboru bylo přidáno několik komponent. Komponenta *plc_modbus* poskytuje služby pro komunikaci s PLC. Spínače, které jsou vygenerovány pomocí komponenty *modbus_sw* mají nastavený scan interval na dobu jedné sekundy. Díky tomu bude

Home Assistant aktualizovat stav spínačů každou vteřinu. V aplikaci jsou použity tři druhy senzorů. První z nich využívá protokol MQTT pro vyčítání teploty v obývacím pokoji. Druhý slouží pro měření výkonu procesoru. To je zde především z důvodu testování, jestli Raspberri Pi 3 má dostatečný výkon pro Home Assistant. Třetí typ senzorů slouží pro měření aktuálního času.
Komunikace se zařízením Vera+ je uskutečněno pomocí komponenty *vera*, která dokáže automaticky detekovat a zobrazit všechna zařízení připojená k Vera Plus. Poslední přidaná komponenta je vstup pro zadávání času zapnutí/vypnutí intimního osvětlení.

```
plc_modbus:

switch:
  - platform: modbus_sw
    scan_interval: 1

sensor:
   - platform: mqtt
     state_topic: "prvniPatro/obyvak/teplota"
   - platform: systemmonitor
      resources:
      - type: processor_use
   - platform: time_date
  display_options:
     - 'time'
     - 'date'
     - 'date_time'
     - 'beat'
vera:
  vera_controller_url: http://192.168.1.105:3480/

input_datetime:
  chodba_od:
    name: Zapnuti svetel
    has_date: false
    has_time: true

```

5.3.4. Změna vzhledu

Protože ne každému vyhovuje výchozí barévné schéma Home Assistantu, přidal jsem možnost měnit vzhled. Pro změnu vzhledu se používají předem připravená témata.
Pro vytváení témat slouží soubor *themes.yaml*.



Obrázek 5.5.: příklad změny vzhledu

5.4. Scripty a automatizace

Pro vytváření Automatizací a Scriptů se mi osvědčilo používat editor integrovaný do webového rozhraní Home Assistantu. Jeho hlavní výhodou je, že se nemusí řešit syntaxe YAML. Díky tomu zvládne napsat jednoduché scripty nebo automatizace i člověk, který nemá s YAML žádné zkušenosti.

Obrázek 5.6.: Editor pro vytváření scriptů

5.4.1. Intimní osvětlení

Automatizace na zapínání/vypínaní intimního osvětlení je řešena tak, že se porovnává aktuální systémový čas s časem zapnutí/vypnutí intimního osvětlení. Pokud se časy rovnají

zapne/vypne se příslušný spínač.

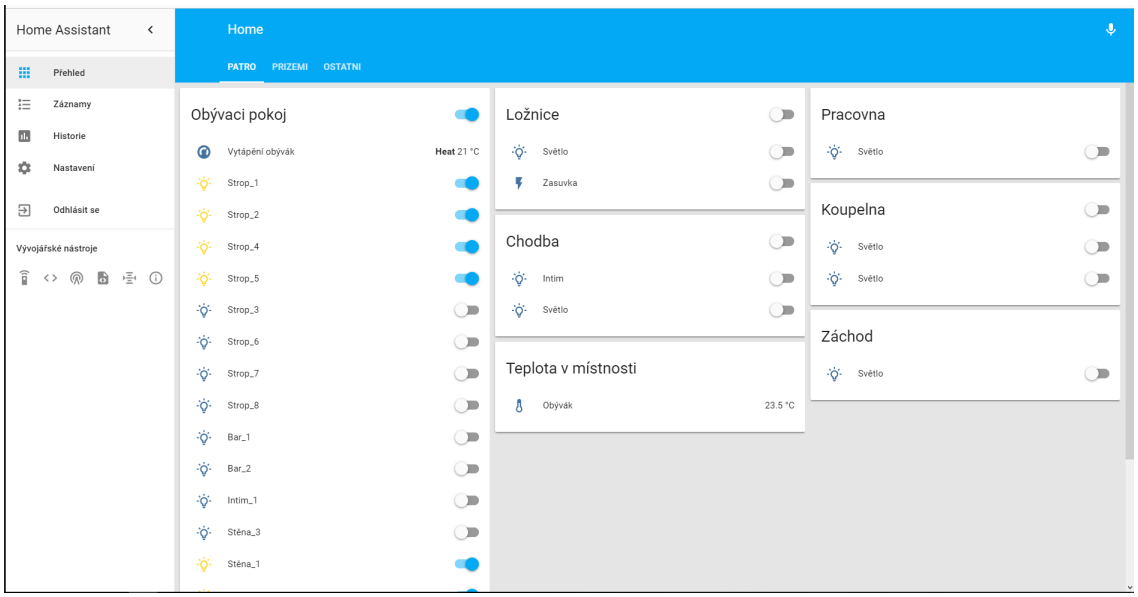
```
- action:
  - alias: zapnuti_intim1_obyvak
    data:
    entity_id: switch.0_10
    service: switch.turn_on
      condition: []
      id: '1522672482244'
      trigger:
      - platform: template
        value_template: '{{ states(''sensor.time'') == (states.input_datetime.
    obyvak_intim1_od.attributes.timestamp| int | timestamp_custom(''%H:%M'', False))
    }}'


```

5.4.2. Modbus

Synchronizace dat s PLC je realizováno pomocí scriptu *CteniDatPlc*. Ten nejprve zavolá službu *plc_write*, která zapíše registry, ve kterých došlo ke změně. Poté co se dokončí zápis registrů do PLC, dojde k zavolání služby *plc_read*, která načte skutečný stav registrů.
O pravidelné spouštění scriptu se stará automatizace s názvem *update_modbus*. Jako trigger slouží systémový event *time_change*, který se vyvolá každou vteřinu. Dále je zde podmínka, aby automatizace běžela pouze pokud není spuštěn script *CteniDatPlc*. Pokud je tato podmínka splněna tak se zavolá script pro aktualizaci registrů.

```
    alias: CteniDatPlc
    sequence:
     service: plc_modbus.plc_write
      service: plc_modbus.plc_read

```

```
- action:
  - service: script.1522670956418
 alias: update_modbus
 condition:
  - condition: state
   entity_id: script.1522670956418
   state: 'off'
 id: '1522671186544'
 trigger:
  - event_data: {}
   event_type: time_changed
   platform: event

```



Obrázek 5.7.: Home Assistant

6. Ověření v praxi

Testování bylo zahájeno 1. března 2018. Během testování se postupně zdokonaloval způsob komunikace s PLC. Jako první byla použita varianta, kde každý spínač funguje jako samostatná entita a o komunikaci s PLC se staral sám (neexistovala třída RegisterHandler). To se ukázalo jako velice nepraktické řešení, a to z toho důvodu, že doba od zmáčknutí spínače po rozsvícení příslušného světla trvali někdy i několik vteřin.
Další z testovaných variant využívala buffer. Tento buffer představoval kopii jednoho PLC registru. Do tohoto bufferu byla automaticky každé dvě vteřiny uložena aktuální hodnota registru v PLC. O zápis do PLC se stále staral každý spínač sám. Tato verze už byla značně rychlejší. Především díky tomu, že o čtení dat už se nemusel start každý spínač samostatně. Místo toho bylo prováděno hromadně pro všechny spínače najednou. Nevýhodou tohoto řešení byl zápis. To se projevovalo zejména, když se uživatel pokusil vypnout více světel najednou. Kód byl v takovém případě nepředvídatelný. A tak ve výsledku se některá světla vypla a některá ne.
Nakonec byla použita varianta popsaná v kapitole 3.5.1. Která již netrpí žádným z nedostatků předchozích variant.
Během testovacího období nebyly zaznamenány žádné pády nebo jiné problémy, které by bránili používání Home Assistantu.

7. Závěr

Cílem této bakalářské práce bylo umožnit ovládání osvětlení, některých zásuvek a vytápění pomocí webového rozhraní. Tohoto cíle se jednoznačně dosáhnout povedlo. Celý systém se ukázal jako velice stabilní. A k dnešnímu datu (14.4.2018) nebyl zaznamenám, žádný problém, se stabilitou. Celý systém je navržen tak aby se dal snadno rozšířit. Díky tomu je možné přidat další místnosti nebo i celé patro s minimálními úpravami kódu. Ne vše se ale povedlo. Raspberry Pi se postupem času ukázalo jako špatná volba. I když výkonem zatím stačí, tak v případě přidání dalšího patra už nebude jeho výkon stačit. Bude tak potřeba zakoupit silnější počítač pro provozování Home Assistantu.

Veškeré zdrojové kódy, které v rámci této bakalářské práce vnikly jsou dostupné na přiloženém DVD. Z důvodu bezpečnosti byli z kódu odebrány veškeré historické záznamy a přístupové klíče. Kromě zdrojových kódů je k dispozici i krátké video s ukázkou funkčnosti.

S výsledkem bakalářské práce jsem velmi spokojen.

Seznam použité literatury

[1] BERGER, Hans. *Automating with SIMATIC S7-1200: configuring, programming and testing with STEP 7 Basic, visualization with HMI Basic*. 2., enl. and rev. ed. Erlangen: Publicis Publ, 2013. ISBN 9783895783852.

[2] BLAJA AUTOMATION PORTAL. *BLAJA AUTOMATION PORTAL* [online]. [cit. 2017-10-24]. Dostupné z: https://www.blaja.cz

[3] STENERSON, Jon a David DEEG. *Siemens Step 7 (Tia Portal) Programming, a Practical Approach*. 2015-07. nevim pico: Createspace Independent Publishing Platform, 2015. ISBN 9781515220541.

[4] SCHWARTZ, Marco. *Internet of Things with ESP8266* [online]. 1. Packt Publishing Limited, 2016 [cit. 2018-01-14]. ISBN 978-1786468024.

[5] Home Assitant Cookbook. *Home Assitant* [online]. [cit. 2018-04-14]. Dostupné z: https://www.home-assistant.io/cookbook/

[6] *Home Assitant Development Guide* [online]. [cit. 2018-02-14]. Dostupné z: https://www.home-assistant.io/developers/

[7] GAY, Warren. *Raspberry Pi hardware reference* [online]. New York, NY: Apress, 2014 [cit. 2018-04-14]. Technology in action series. ISBN 978-1-484208-00-7.

Seznam obrázků

A. Obsah přiloženého CD

soubor/adresář	obsah
bakalarska-prace.pdf	originální text diplomové práce