|  |  |  |
| --- | --- | --- |
|  | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **Ovládání žaluzií** | | |
| Pavel Košnar | | |
|  | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2022/2023 | |

#### Poděkování

Chtěl bych poděkovat panu učiteli Ing. Petru Grussmannovi za pomoc s projektem a jeho náležité připomínky. Také děkuji svému otci za pomoc se složením hardwaru a sponzoring.

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité   
informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové   
a umělecké škole v Opavě, Praskova 399/8.

V Opavě 31. 12. 2022

*podpis autora práce*

**ANOTACE**

Práce slouží k bezdrátovému ovládání žaluzií pomocí esp8266. Vše funguje přes MQTT server Home Assistanta. V Home Assistantu se po úspěšném spojení objeví tlačítka k ovládání. Lze ovládat posun žaluzií nahoru či dolů a zastavit v jakoukoli chvíli. Je také možné žaluzie naklápět. Všechny informace o probíhajících procesích a stavu žaluzií lze pozorovat v MQTT záložce Home Assistanta, kde je také možné žaluzie ovládat pomocí příkazů. Pokud uživatel nemá k Home Assistantu přístup, může zařízení ovládat přes webovou stránku. Na webové stránce však není stav žaluzií.

OBSAH

[Úvod 5](#_Toc370246085)

[1 Teoretická a metodická východiska 6](#_Toc370246086)

[2 Využité technologie 7](#_Toc370246087)

[3 Způsoby řešení a použité postupy 8](#_Toc370246088)

[4 Výsledky řešení, výstupy, uživatelský manuál 9](#_Toc370246089)

[Závěr 10](#_Toc370246090)

[Seznam použitýCH INFORMAČNÍCH ZDROJů 11](#_Toc370246091)

[Seznam příloh 12](#_Toc370246092)

Úvod

Mým cílem pro závěrečný projekt bylo posunout se v rámci informatiky zase o kousek dále. Jelikož jsem moc nerozuměl elektrotechnice, rozhodl jsem se rozvinout své znalosti právě v tomto oboru. Bezdrátové ovládání žaluzií se tedy jevilo jako nejrozumnější téma. Projekt totiž obsahuje nejen elektrotechniku, ale i programování v jazyce c++, komunikaci se serverem a logické přemýšlení.

Původně jsem pro svůj projekt chtěl vyvinout ještě vlastní aplikaci, ale z časových důvodů se tak nestalo.

Chtěl jsem se také naučit psát efektivnější funkce, což si myslím, že se mi postupem času povedlo. V kódu projektu je možné vidět postupné zlepšení, co se týče nejen efektivity, ale i rychlosti a jednoduchosti.

Dále v dokumentaci popisuji postupy a využité technologie.

# Teoretická a metodická východiska

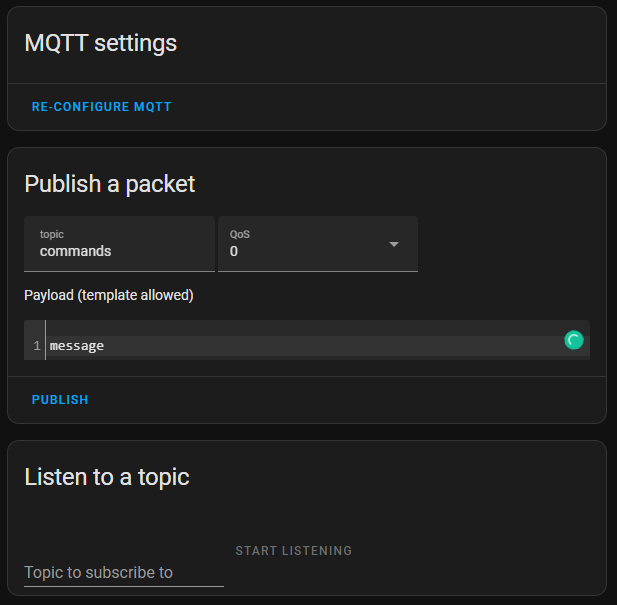
## Využitý hardware

Základním a nejdůležitějším hardwarem mého projektu je esp8266 – Wi-Fi mikročip se zabudovaným TCP/IP síťovým softwarem. Dále 80x50cm žaluzie s \_\_\_ elektromotorem a dvě spínací relé.

## Home Assistant a MQTT server

Home Assistant poskytuje vhodné prostředí pro operace se zařízeními pro chytrou domácnost. Z esp8266 se navazuje spojení s Home Assistantem přes MQTT server, který je běžně dostupný v HA Add-ons. Přes Home Assistant lze také MQTT konfigurovat, posílat a přijímat zprávy.

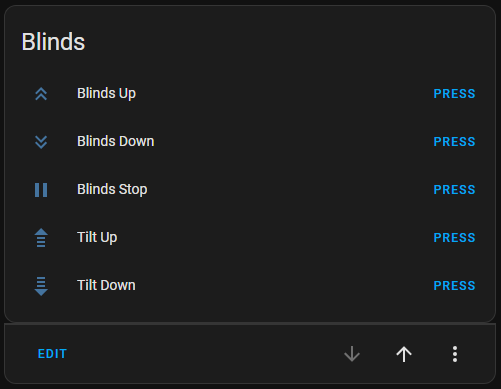
MQTT v Home Assistantu vypadá následovně:



Obrázek 1.1: MQTT v HA

## Knihovna ArduinoHA

Pro připojení esp8266 k Home Assistantu je zapotřebí knihovna ArduinoHA. Není sice ani z daleka jediná, ovšem pro můj projekt se jevila jako nejvhodnější volba. Knihovna totiž kromě připojení k MQTT umožňuje také vytvářet vlastní tlačítka. Veškeré informace o tlačítkách posílá serveru v podobě json formátu. Home Assistant je poté schopen přijímané informace proměnit v tlačítka, které se zobrazují na Overview (také Dashboard) stránce.



Obrázek 1.2: Tlačítka v Home Assistantu

# Využité technologie

## PlatformIO

Nejoblíbenější IDE řešení pro Microsoft Visual Studio Code. Uživatelsky přívětivé a rozšiřitelné integrované vývojové prostředí se sadou profesionálních vývojových nástrojů, které poskytuje moderní a výkonné funkce pro urychlení a zároveň zjednodušení vytváření a dodávání produktů.

Přes PlatformIO lze jednoduše komunikovat s esp8266 a to přímo ve Visual Studio Code. Byla to tedy nejpohodlnější volba pro můj projekt.

## Home Assistant

Home Assistant je bezplatný a open-source software pro domácí automatizaci navržený jako centrální řídicí systém pro inteligentní domácí zařízení se zaměřením na místní ovládání a soukromí. Lze k němu přistupovat prostřednictvím webového uživatelského rozhraní pomocí doprovodných aplikací pro Android a iOS nebo pomocí hlasových příkazů prostřednictvím podporovaného virtuálního asistenta, jako je Google Assistant nebo Amazon Alexa.

## MQTT

MQTT je standardní protokol pro zasílání zpráv OASIS pro internet věcí (Internet of Things). Je navržen jako extrémně lehký přenos zpráv pro publikování/odběr, který je ideální pro připojení vzdálených zařízení. MQTT se dnes používá v celé řadě průmyslových odvětví.

## Little FS

Little FS je vysoce integrovaný souborový systém v Mbed OS, optimalizován pro práci s omezeným množstvím RAM a ROM. Vyhýbá se rekurzi, omezuje dynamickou paměť na konfigurovatelné vyrovnávací paměti a v žádném okamžiku neukládá celý úložný blok do paměti RAM.

Little FS jsem využil pro práci s html, css a javascript soubory v paměti.

# Způsoby řešení a použité postupy

## Založení projektu

Než jsem mohl začít s psaním kódu projektu, bylo zapotřebí si stáhnout a nainstalovat mnoho programů. První a nejdůležitější z nich je Visual Studio Code. Ve VSCode jsem potřeboval různá rozšíření (např. Platformio, C/C++ a Python, ten jsem potřeboval stáhnout i přímo do počítače). Dále jsem si nainstaloval a nakonfiguroval Home Assistant přes Virtual Box. Byla také potřeba nainstalovat všechny knihovny, které jsem k projektu použil. To ovšem bylo díky PlatformIO velmi jednoduché, jelikož umožňuje automatické nainstalování knihoven po jejich připsání do „platformio.ini“ souboru. Po všech úspěšných instalacích jsem ještě napsal krátký kód pro rozsvícení LEDek, abych se ujistil, že vše funguje, jak má.

## Webová stránka

Webová stránka projektu slouží ke snadnému a rychlému přístupu k ovládání zařízení. Má proto velmi jednoduchý vzhled a obsahuje pouze pět tlačítek – nahoru, dolů, naklopení nahoru a dolů a zastavit.

### Vzhled stránky



Obrázek 3.1: Vzhled webové stránky

Stránka je velmi jednoduchá a stručná. Kromě nadpisu a záhlaví obsahuje pouze 5 tlačítek.

### Knihovna ESPAsyncWebServer

Každé z tlačítek přesměruje uživatele na určitou stránku. Uživatel sice změnu nepozná, ale zařízení dostane potřebnou informaci k provedení určitého úkonu.

Ukázka kódu tlačítek pro přesměrování:

<div class="container p-3 my-3 border bg-dark text-center"

style="margin-top: 0 !important; margin-bottom: 0 !important;">

<p><a href="/up"><button class="btn btn-success active btn-block" id="up">UP</button></a></p>

<p><a href="/down"><button class="btn btn-success active btn-block" id="down">DOWN</button></a></p>

<p><a href="/stop"><button class="btn btn-danger active btn-block" id="stop">STOP</button></a></p>

<p><a href="/tilt-up"><button class="btn btn-primary active btn-block" id="tiltUp">TILT UP</button></a></p>

<p><a href="/tilt-down"><button class="btn btn-primary active btn-block" id="tiltDown">TILT DOWN</button></a></p>

</div>

Knihovna ESPAsyncWebServer dokáže poznat, zda se uživatel nachází v určitém adresáři stránky. Pro každý z jednotlivých odkazů je tedy přiřazena funkce, která se spustí ve chvíli, kdy uživatel klikne na tlačítko.

Ukázka kódu vyvolání funkce z jednotlivých odkazů:

server.on("/up", HTTP\_GET, [](AsyncWebServerRequest \*request){

    movement(goUp, "up");

    request->send(LittleFS, "/index.html", String());

});

server.on("/down", HTTP\_GET, [](AsyncWebServerRequest \*request){

    movement(goDown, "down");

    request->send(LittleFS, "/index.html", String());

});

server.on("/stop", HTTP\_GET, [](AsyncWebServerRequest \*request){

    stopMovement("stop");

    request->send(LittleFS, "/index.html", String());

});

## Funkce k ovládání

### Nahoru a dolů

Funkce nahoru nebo dolů nejprve zjistí, v jaké poloze se žaluzie nachází, poté do motoru posílá signál pro pohyb. Funkce také zjistí, jestli už žaluzie dorazily na konec – tedy úplně nahoru nebo úplně dolů. V tomto případě přestane signál posílat.

Funkce je rozdělena na dvě části:

První nastaví proměnné na potřebné hodnoty pro pohyb požadovaným směrem a pošle zprávu MQTT serveru, že se žaluzie začnou posouvat určitým směrem.

Ukázka kódu funkce:

void movement(bool direction, const char\* topic) {

  if (direction == true) {

    stopMovement(topic);

  } else {

    start\_time = millis();

    stop = false;

    if (topic == "up") {

      goUp = true;

      goDown = false;

      digitalWrite(pinDown, LOW);

      digitalWrite(pinUp, HIGH);

      tiltFully = "up";

      mqtt.publish(movement\_topic, topic, true);

    }

    else if (topic == "down") {

      goDown = true;

      goUp = false;

      digitalWrite(pinUp, LOW);

      digitalWrite(pinDown, HIGH);

      tiltFully = "down";

      mqtt.publish(movement\_topic, topic, true);

    }

  }

}

Druhá běží v loopu, kontroluje a aktualizuje polohu žaluzií. Každých 5 minut a při pohybu nahoru či dolů také posílá zprávu MQTT serveru s aktuální polohou v procentech.

Ukázka kódu funkce:

void state() {

  if (!stop) {

    if (goUp == true) {

      if (millis() > start\_time + 1000) {

        current\_position += 100 / path\_length;

        if (current\_position > 100) {

          current\_position = 100;

        }

        if (current\_position == 100) {

          stopMovement("top");

        } else {

          start\_time = millis();

        }

      }

    }

    else if (goDown == true) {

      if (millis() > start\_time + 1000) {

        current\_position -= 100 / path\_length;

        if (current\_position < 0) {

          current\_position = 0;

        }

        if (current\_position == 0) {

          stopMovement("bottom");

        } else {

          start\_time = millis();

        }

      }

    }

    if (goUp == true || goDown == true) {

      if (millis() > state\_time + 1000) {

        state\_time = millis();

        state\_message = itoa(current\_position, buffer, 10);

        mqtt.publish(state\_topic, state\_message, true);

      }

    } else {

      if (millis() > state\_time + 300000) {

        state\_time = millis();

        state\_message = itoa(current\_position, buffer, 10);

        mqtt.publish(state\_topic, state\_message, true);

      }

    }

  }

}

### Zastavit

Tato funkce je velmi jednoduchá a stručná. Zahrnuje pouze nastavení proměnných, přestane posílat signál do motoru a pošle zprávu MQTT serveru s důvodem zastavení. Důvody mohou být stisknutí tlačítka nebo dojezd žaluzií úplně nahoru či dolů.

Ukázka kódu funkce:

void stopMovement(const char\* reason) {

  goUp = false;

  goDown = false;

  stop = true;

  tiltDirection = "stop";

  digitalWrite(pinUp, LOW);

  digitalWrite(pinDown, LOW);

  mqtt.publish(movement\_topic, reason, true);

}

### Naklopení nahoru a dolů

Napsat vhodnou funkci pro naklápění žaluzií byl asi ten nejsložitější úkol, co se týče logiky. Nejen, že žaluzie nelze naklápět, zatímco jsou v pohybu, ale taky je před každým pohybem nahoru či dolů potřeba žaluzie naklopit do vhodné polohy pro pohyb daným směrem. Ještě je potřeba neustále kontrolovat úhel naklopení žaluzií, protože uživatel může proces kdykoliv přerušit tlačítkem „stop“.

Problém jsem vyřešil rozdělením do tří funkcí:

První je opět jednoduchá – zase jde pouze o nastavení potřebných proměnných a poslání signálu do motoru.

Ukázka kódu funkce:

void startTilting(String direction) {

  tiltDirection = direction;

  tiltTime = millis() + 400;

  pauseTime = tiltTime + 800;

  if (direction == "up") {

    digitalWrite(pinUp, HIGH);

  }

  else if (direction == "down") {

    digitalWrite(pinDown, HIGH);

  }

}

Druhá funkce dohlíží na správný chod naklápění – 400ms posílat signál do motoru, 800ms pauza. Vždy při pauze pošle zprávu MQTT serveru s aktuální polohou naklopení v procentech.

Ukázka kódu funkce:

void tiltMovement(String tilt) {

  if (tilt != "stop" && stop) {

    if (tiltTime < millis() && pauseTime > millis()) {

      tiltTime = pauseTime + 400;

      if (tilt == "up") {

        digitalWrite(pinUp, LOW);

        tilt\_position += 10;

      }

      else if (tilt == "down") {

        digitalWrite(pinDown, LOW);

        tilt\_position -= 10;

      }

      if (tilt\_position >= 100) {

        tiltDirection = "stop";

        tilt\_position = 100;

      }

      else if (tilt\_position <= 0) {

        tiltDirection = "stop";

        tilt\_position = 0;

      }

tilt\_message = itoa(tilt\_position, buffer, 10);

mqtt.publish(tilt\_topic, tilt\_message, true);

    }

    else if (pauseTime < millis()) {

      pauseTime = tiltTime + 800;

      if (tilt == "up") {

        digitalWrite(pinUp, HIGH);

      }

      else if (tilt == "down") {

        digitalWrite(pinDown, HIGH);

      }

    }

  }

}

Třetí funkce pracuje pouze v případě, že byl zadán příkaz posunout žaluzie nahoru či dolů. Má za úkol naklopit žaluzie do správné polohy pro pohyb daným směrem. Při započetí i při dokončení naklápění informuje MQTT server.

Ukázka kódu funkce:

void fullTilt() {

  if (tiltFully == "up" && current\_position >= 100) {

    tiltFully = "no";

  }

  else if (tiltFully == "down" && current\_position <= 0) {

    tiltFully = "no";

  }

  if (tiltFully == "up") {

    if (!stop && tilt\_position < 100) {

      stop = true;

      fullTiltTime = millis();

      mqtt.publish(movement\_topic, "Tilting up...", true);

    }

  }

  else if (tiltFully == "down") {

    if (!stop && tilt\_position > 0) {

      stop = true;

      fullTiltTime = millis();

      mqtt.publish(movement\_topic, "Tilting down...", true);

    }

  }

  if (stop) {

    if (tiltFully == "up" && fullTiltTime + 400 < millis()) {

      tilt\_position += 10;

      if (tilt\_position > 100) {

        tilt\_position = 100;

      }

      fullTiltTime = millis();

    }

    else if (tiltFully == "down" && fullTiltTime + 400 < millis()) {

      tilt\_position -= 10;

      if (tilt\_position < 0) {

        tilt\_position = 0;

      }

      fullTiltTime = millis();

    }

    if (tilt\_position >= 100 && tiltFully == "up" ||

tilt\_position <= 0 && tiltFully == "down") {

      stop = false;

      tiltFully = "no";

      mqtt.publish(movement\_topic, "Done", true);

    }

  }

}

### Naklopení o jeden stupeň

Velmi stručná a jednoduchá funkce. Funguje podobně jako funkce naklápění, ale posune se pouze o kousek a zůstane stát.

Tvořena ze dvou funkcí, kdy první opět jen nastavuje proměnné, druhá žaluzie naklopí a poté zastaví.

Ukázka kódu obou funkcí:

void startStepTilt(String direction) {

  stepTiltDirection = direction;

  stepTiltTime = millis() + 800;

  if (direction == "up") {

    digitalWrite(pinUp, HIGH);

  }

  else if (direction == "down") {

    digitalWrite(pinDown, HIGH);

  }

}

void stepTilt(String direction) {

  if (direction != "none") {

    if (stepTiltTime < millis()) {

      stepTiltDirection = "none";

      if (direction == "up") {

        tilt\_position += 20;

        if (tilt\_position > 100) {

          tilt\_position = 100;

        }

        digitalWrite(pinUp, LOW);

      }

      if (direction == "down") {

        tilt\_position -= 20;

        if (tilt\_position < 0) {

          tilt\_position = 0;

        }

        digitalWrite(pinDown, LOW);

      }

      tilt\_message = itoa(tilt\_position, buffer, 10);

      mqtt.publish(tilt\_topic, tilt\_message, true);

    }

  }

}

# Výsledky řešení, výstupy, uživatelský manuál

Text čtvrté kapitoly

* výčet splněných a nesplněných cílů, obrázky (schémata, vzorce apod.) z finálního provedení, prokázání funkčnosti, výsledné parametry výrobku apod.
* podle zaměření a charakteru práce je třeba volit vhodný nadpis pro tuto kapitolu, je samozřejmě možné i rozdělení na více kapitol (např. Uživatelské rozhraní internetové aplikace; Administrace internetové aplikace…)

# Závěr

Text závěru

* povinná část,
* shrnuje výsledky, hodnotí splnění cíle práce, uvádí možnost uplatnění řešení v praxi a nastínění případných dalších budoucích vylepšení
* kapitola se nečísluje (stejné jako úvod)

Seznam použitýCH INFORMAČNÍCH ZDROJů

[1] BOHMAN, Ludvík. Zákon o pojistné smlouvě. Praha: Linde Praha a. s., 2004. 381 s. ISBN80-7201-504-4

[2] DUCHÁČKOVÁ, Eva. Principy pojištění a pojišťovnictví. 3. aktualizované vydání. Praha: Ekopress 2009. 224 s. ISBN 978-80-86929-51-4

[3] KUBALA, Petr. Planetární dvojcata - Věda a technika (Český rozhlas) [online].   
Č. 2000-2008, poslední revize 19. 3. 2008 [cit. 2008-03-20].  
<http://www.rozhlas.cz/veda/vesmir/\_zprava/435849>.

[4] KULDOVÁ, O., FLEISCHMANNOVÁ, E. Metodická příručka k technice administrativy a obchodní korespondence. 1.vyd. Praha: Fortuna 1998. 111 s.   
ISBN 80-7168-574-7. Kapitola 6, Metody nácviku psaní hmatovou metodou,   
s. 28-29.

[5] VLACH, J. JE Temelín a zásobování teplem. Energetika, 2001, roč. 51, č. 3, s. 84 -85. ISSN 0375-8842.

* musí zahrnovat všechny prameny, knihy, internetové odkazy a další studijní podklady, z nichž jsme čerpali;
* kapitola se nečísluje a zde končí číslování stránek práce;
* jednotlivé publikace se uvádějí v abecedním pořadí podle příjmení autorů a iniciál jeho jména, který se píše za čárkou;
* příjmení autora se píše velkými písmeny;
* název publikace se zvýrazňuje kurzívou;
* jestliže jsou uvedeni více než tři autoři, je možné vypsat hlavního autora s poznámkou „a kol.“(a kolektiv).