|  |  |  |
| --- | --- | --- |
|  | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **Dávkovač krmiva** | | |
| Filip Dušek | | |
| ESP32-DevKitC Development Boards - Espressif Systems | Mouser | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2020/2021 | |

#### Poděkování

* *Chtěl bych poděkovat panu učiteli Ing. Petru Grussmannovi za cenné rady a věcné připomínky. Dále pak panu učiteli Mgr. Markovi Lučnému za pomoc s tvorbou web serveru a Mgr. Marceli Godovkému za vyřešení problematiky napájení.*

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. 2016

*podpis autora práce*

**ANOTACE**

*Cílem projektu bylo vytvořit automatický dávkovač krmiva pro morčata. Dávkovač by fungoval tak, že by dávkoval krmivo dle nastavených parametrů. Parametry jako: Čas spuštění, den, velikost dávky, by bylo možné nastavit na web serveru. Dávkování bude probíhat díky krokového motoru ovládaného ESP-32, Na kterém bude umístěn web server. Takže bude možné nastavit parametry na wifi síti přes mobilní telefon, počítač nebo jiné zařízení s přístupem k wifi.*

**Klíčová slova**

automatický dávkovač, web server, EPS-32,

OBSAH

[Poděkování 2](#_Toc60414804)

[Úvod 5](#_Toc60414805)

[1 Teoretická a metodická východiska 6](#_Toc60414806)

[2 Využité technologie 7](#_Toc60414807)

[2.1 Procesor 7](#_Toc60414808)

[2.2 Krokový motor 28BYJ-48 s řadičem 7](#_Toc60414809)

[2.3 Software 8](#_Toc60414810)

[2.3.1 Visual Studio Code a PlatformIO 8](#_Toc60414811)

[2.3.2 Knihovny 8](#_Toc60414812)

[2.3.3 Nástroje pro tvorbu webu 9](#_Toc60414813)

[2.3.4 KiCAD 9](#_Toc60414814)

[3 Způsoby řešení a použité postupy 10](#_Toc60414815)

[3.1 Schématické zapojení 10](#_Toc60414816)

[3.2 Napájení 10](#_Toc60414817)

[4 Způsoby řešení softwaru a použité postupy 11](#_Toc60414818)

[4.1 Připojení k internetu 11](#_Toc60414819)

[4.2 Získávání proměnných 12](#_Toc60414820)

[4.3 Posílání proměnných 13](#_Toc60414821)

[4.4 Výpis času 13](#_Toc60414822)

[4.5 Hlavní část 14](#_Toc60414823)

[4.6 Využití knihovny Stepper 15](#_Toc60414824)

[Závěr 16](#_Toc60414825)

[Seznam použitýCH INFORMAČNÍCH ZDROJů 17](#_Toc60414826)

Úvod

Cílem této práce bylo vytvořit funkční dávkovač krmiva s možností nastavení proměnných přes wifi.

Ze začátku bylo cílem pouhé nastavení délky intervalu mezi dávkami. Postupně však přicházely praktické návrhy na vylepšení. Z tohoto důvodu jsem pokračoval ve vývoji projektu a podle následné odezvy uživatelů upravoval systém.

Projekt má velmi jednoduchou hardwarovou stránku, která obsahuje pouze samotný krokový motorek, driver k motorku, externí napájení a ESP-32.  ESP-32 umožnuje nastavení dávkovače v podstatě kdekoliv v dosahu domácí wifi sítě, na všech zřízeních podporujících připojení k internetu.

V dokumentaci zmiňuji použité technologie a blíže popisuji princip fungování dávkovače. Dále se zaměřuji na softwarovou stránku, vysvětluji, na jakých principech je založen můj kód a věnuji pozornost problémům, které jsem musel řešit během vývoje svého projektu.

# Teoretická a metodická východiska

* Cílem tohoto projektu bylo programování mikrokontroleru a tvorba web serveru pro nastavení dávkovače. Na začátku projektu bylo velice důležité vybrat si správné technologie, které budou použity. První věc, kterou jsem musel udělat, bylo představit si, jak celé zařízení bude pracovat, jak to bude vypadat ve skutečnosti. Po utvoření představy bylo nutné vybrat si správný mikroprocesor, který celé zařízení bude řídit. Po zvážení jsem si zvolil procesor firmy espressif ESP-WROOM-32. Dalším krokem bylo zvolení správného motorku, který bude dávkovat samotné krmivo. Po zvážení jsem si zvolil stopový motorek: Krokový motor 28BYJ-48 s řadičem. Po nakreslení schématu zapojení celého zařízení bylo načase zakoupení všech potřebných součástek. Nejdříve jsem zapojení zkoušel na nepájivém kontaktním poli, kde jsem zkoušel správnou funkčnost zařízení. Když vše fungovalo správně na kontaktním poli. Po zasazení součástek do těchto otvorů už jen stačí propojit správně součástky podle schématu zapojení. Nyní byl ten správný čas pro vytvoření programu a následně web serveru. Když jsem vytvořil základ programu pro blikání LED přes tlačítko na web serveru bylo potřeba získávat z web serveru proměnné, které nám budou udávat samotnou funkci motorku, a časy dávek. Což bylo realizováno pomocí pana Lučného přes JavaScript. Poté už bylo jen třeba vyzkoušet v praxi a vytvořit “krabičku” která bude fungovat jako obal pro mikroprocesor, drátky a motorek.

# Využité technologie

## Procesor

Celý chod programu je řízen pomocí 32 bitového čipu ESP32 od firmy Espressif, který má implementovaný WiFi modul, a také Bluetooth ve verzi 4.2 s podporou BLE (Bluetooth Low Energy), který ale nespolupracuje se staršími zařízeními používající starší verzi bluetooth. Parametry :

  ● Napájení 3.3V/5V

 ● 36 pinů

● Flash paměť 16 MB

● SRAM 512 kB

● Frekvence 160 MHz

## Krokový motor 28BYJ-48 s řadičem

Pohon je zajištěn krokovým motorem, který je ovládán řadičem s tranzistorovým polem ULN2003. Zmíněný motor je unipolární a jedná se o elektromagnetické zařízení, které přeměňuje elektrické impulzy na mechanický pohyb. Mezi výhody tohoto krokového motoru patří snadné otočení o libovolný úhel, velká síla i v klidové pozici či okamžitá odezva při zastavení či spuštění motoru.

Typ motoru:28BYJ-48

Převodový poměr: 1 / 64

Úhel kroku:5.625 x 1 / 64

Průměr:28 mm

LED status A, B, C, D

Piny IN1, IN2, IN3, IN4

Napájení: 5VDC

Čip: UNL2003



## Software

### Visual Studio Code a PlatformIO

Programovací prostředí od firmy Microsoft. Podporuje programování v mnoha programovacích jazycích, jako například C, C++, C# aj. Můžeme zde vytvářet nejrůznější programy – od jednoduchých konzolových programů přes vizuální aplikace až po rozsáhlé webové projekty. Microsoft Visual Studio obsahuje mnoho nejrůznějších funkcí, které usnadňují programování.

### Knihovny

Jsou to složky se soubory, ve kterých je napsán nějaký zdrojový kód, tudíž nám umožňují mít přehlednější a jednodušší vlastní program. Ve svém kódu jsem použil tyto knihovny:

●NTPClient – Získávám skrze něj čas.

● SPIFFS.h – Knihovna pro nahrání HTML souboru apod. do flash paměti ESP32.

● ESPAsyncWebServer.h – Pro vytvoření web serveru na ESP32.

### Nástroje pro tvorbu webu

HTML (Hypertext Markup Language) je značkovací jazyk pro tvorbu webových stránek. CSS je zkratka pro kaskádové styly. Popis způsobu zobrazení elementů napsaných v jazycích HTML, XHTML nebo XML. JavaScript je multiplatformní objektově orientovaný skriptovací jazyk.

### KiCAD

Svobodný software pro projektování plošných spojů, tedy pro jednu z úloh při automatizaci projektování elektroniky, podporovaného projektování zaměřeného na projektování elektronických výrobků a jejich součástí.

# Způsoby řešení a použité postupy

## Schématické zapojení

Seznam nutných součástek

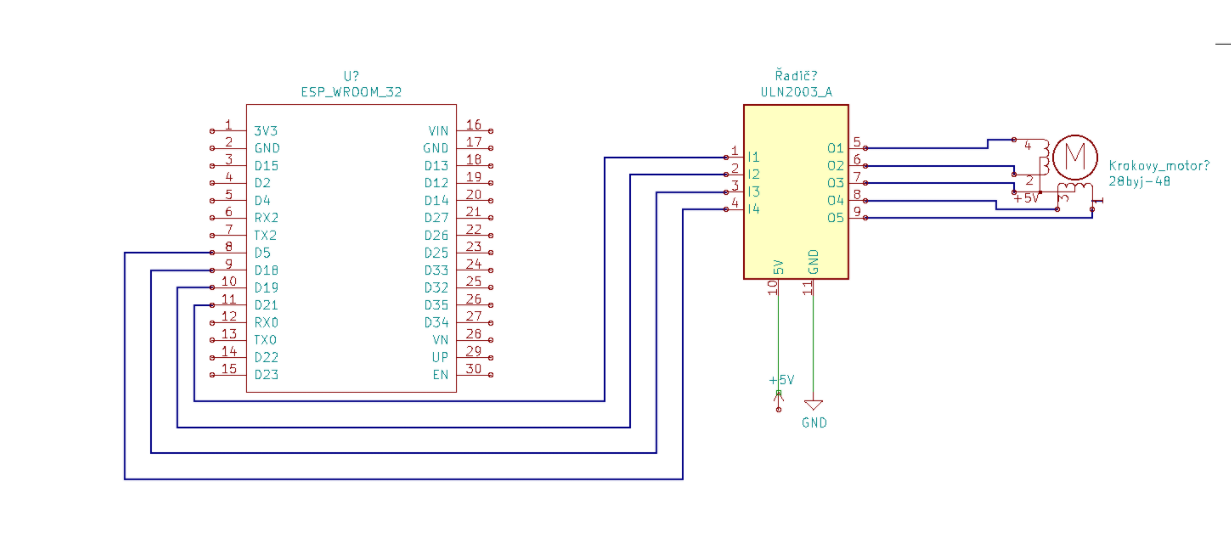
ESP-32

Krokový motor 28BYJ-48 s řadičem.

Pár drátků.

Externí napájení

USB do Mini USB kabel.



## Napájení

První větší problém který se mi nedařilo vyřešit bylo napájení krokového motoru. Který jsem z počátku řešil tím, že motorek napojím přímo na ESP-32, jenže to podle dokumentace motorku nedokáže pokrýt špičkové proudy.

“Uživatel musí dbát na výkonnost externího zdroje. Zdroj musí být schopný pokrýt špičkové proudy (cca 500 mA), které mohou v zátěži nastat.” Citace z dokumentace motorku.

Nakonec jsem požádal o pomoc pana učitele Mgr. Marcela Godovského, který mi daroval externí zdroj, který jsem následně využil.

## Tvorba obalu pro zařízení

# Způsoby řešení softwaru a použité postupy

## Připojení k internetu

Funkce pro připojení k internetu. Funkce používá globální řetězce ssid a password do kterých stačí uložit název domácí wifi sítě a heslo k ní. Samozřejmě jde to řešit elegantněji přes wifi mannager, ale vzhledem k plánovanému využití mého dávkovače jsem to nepoužil.

// Funkce pro pripojeni se k místní wifi síti

void pripojeni()

{

  //vypis připojováni

  Serial.printf("Connecting to %s ", ssid);

  //samotné pripojení pomocí ssid a hesla

  WiFi.begin(ssid, password);

  //dokud nebude pripojeno bude to psát tečky co 0.5s

  while (WiFi.status() != WL\_CONNECTED)

  {

    delay(500);

    Serial.print(".");

  }

  //vypis připojeno

  Serial.println(" CONNECTED");

  configTime(gmtOffset\_sec, daylightOffset\_sec, ntpServer);

  //vyvolání funkce pro získání času

  getTime();

  Serial.begin(115200);

  // Vyvolání SPIFFS

  if (!SPIFFS.begin(true))

  {

    Serial.println("An Error has occurred while mounting SPIFFS");

    return;

  }

}

## Konečné řešení webové otázky

Po připojení k internetu bylo třeba vyřešit podobu samotné webové stránky. První jsem se pokoušel řešit funkci na principu zadání intervalu mezi jednotlivými dávkami. Poté

## Získávání proměnných

Úplně na startu jsem si našel poměrně jednoduchý kód, který získává proměnné z jednoduchého formuláře, a zapisuje je do dané proměnné na ESP, tak že s nimi pak mohu manipulovat, funkci jsem si upravil k obrazu svému tím, že je ukládám do typedef struct takže je možné s nimi operovat po “řádcích a sloupcích” což mi do budoucna usnadňuje práci s nimi.

void ziskani()

{

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

    inputIndex = request->arg("i").toInt();

    Serial.println(inputIndex);

    request->send\_P(200, "text/plain", "{\"result\":\"ok\"}");

  });

  delay(10);

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

    data[inputIndex].hodiny = request->arg("hodiny");

    Serial.println(data[inputIndex].hodiny);

    request->send\_P(200, "text/plain", "{\"result\":\"ok\"}");

  });

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

    data[inputIndex].davka = request->arg("davka").toInt();

    Serial.println(data[inputIndex].davka);

    request->send\_P(200, "text/plain", "{\"result\":\"ok\"}");

  });

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

    data[inputIndex].datum = request->arg("datum");

    Serial.println(data[inputIndex].datum);

    request->send\_P(200, "text/plain", "{\"result\":\"ok\"}");

  });

## Posílání proměnných

Část kódu, kterou chci doplnit výše uvedený obrázek. Tato funkce získá z formuláře, který je možné rozšířit o další kolonky. Po stisknutí na tlačítko “Aktualizovat data” tato funkce opět projede celý formulář díky cyklu for, vezme z něj zadané proměnné a pošle je do předtím zmíněné funkce na ESP.

$('#send').on('click', function () {

    let i = 0;

    for (i = 0; i < index; i++) {

        var ho = $('#hodiny' + i).val();

        var dd = $('#davka' + i).val();

        var da = $('#datum' + i).val();

        console.log(i);

        console.log(ho);

        console.log(dd);

        console.log(da);

        $.post("/hodiny", { hodiny: ho });

        $.post("/davka", { davka: dd });

        $.post("/datum", { datum: da });

        $.post("/i", { i, i });

    }

});

## Výpis času

Taky bych zde chtěl zmínit problém s výpisem času na web serveru, který jsem řešil docela dlouho a vlastně jsem neuměl ani najít nějakou spolehlivou referenci nebo návod jak by to mohlo jít. Pan učitel Grussman mi zkoušel poradit že bych čas měl kopírovat do proměnné, která by měla mít podobu JSONu ale to se mi taky nepodařilo takto vyřešit. Nakonec jsem si našel referenci s části kódu níže. Ta mi dovolila uložit čas do proměnné ve tvaru char. A následně mi začal fungovat i výpis na web serveru.

strftime(porovnaniCas, 6, "%H:%M", &timeinfo);

## Hlavní část

Na obrázku je zachycena část kódu, která je vlastně hlavní částí mého projektu. Tato funkce mi porovnává proměnné získané z web serveru (které získávám jinou funkcí) s reálným časem, který výše v této funkci získávám pomocí NTP serveru. Pokud jsou srovnané hodnoty stejné pak se zapne funkce pro motorek, do které bude zaslána ještě velikost dávky, která určí množství krmiva. Zároveň je v ní výše zmíněný výpis na web server.

void printLocalTime()

{

  char den[15];

  String porovnani;

  String denporovnani;

  struct tm timeinfo;

  // Jestliže nebyl získán čas z NTP serveru napíše to hlášku že nebyl získan čas

  if (!getLocalTime(&timeinfo))

  {

    Serial.println("Selhání získání času.");

    return;

  }

  // Vypíše získaný čas v celé jeho podobě (den měsíc...)

  Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");

  // nakoopíruje mi do mojí proměnné současné hodiny a minuty

  strftime(porovnaniCas, 6, "%H:%M", &timeinfo);

  // to samé jako ta předchozí jenom nakopíruje pouze den

  strftime(den, 10, "%A", &timeinfo);

  porovnani = porovnaniCas;

  denporovnani = den;

  Serial.println(denporovnani);

  Serial.println(porovnani);

  int x = 0;

  // standartní funkce for která mi projede celé DATA od hodnoty 0 do hodnoty která je menší nebo rovna indexu.

  for (x = 0; x <= inputIndex; x++)

  {

    //jestliže je den který jsme vložili do web serveru ten stejný který je dnes

    if (denporovnani == data[x].datum)

    {

      // a zároven je i čas zadaný na web serveru ten stejný jako čas současný

      if (porovnani == data[x].hodiny)

      {

        // zapneme funkci motorek a zároven pro neopakování otevření zásobníku je zde nutný delay.

        motorek();

        Serial.println("Motorek jede!\n");

        delay(61000);

      }

    }

  }

}

## Využití knihovny Stepper

Jedná se o knihovnu, která ulehčuje práci s motorkem. Je určena pro práci s různými krokovými motory jak unipolárními tak bipolárními. K použití této knihovny je vyžadováno pouze vlastnit motorek, driver a mikrokontroler.

# Závěr

Úkolem bylo vytvořit automatický dávkovač krmiva, který bude dávkovat v daný na webu nastavitelný čas, den, a nastavenou dávku. Tím pádem by ho bylo možné ovládat ho přes jakékoliv zařízení na domácí wifi síti. Následně vytvořit exteriér a samotnou nádobu na krmivo.

Jistě je zde pár věcí, které bych mohl vylepšit. Jedním z možných vylepšení je například wifi manager pro snadnější konfiguraci místní sítě. Současné řešení je sice funkční, ale aby bylo možné změnit již nastavenou síť je nutné nahrát celý kód znovu. Další možné vylepšení se týká backendu, který není úplně dokonalé a zasloužilo by si pár vylepšení jako například při znovu načítání stránky se celá stránky uvede do původního stavu, ale proměnné se do aktualizace nezmění. Dále mnou udělaný exteriér a zásobník by mohly být lepší a krásněji udělané, například vytisknuté na 3D tiskárně.

Seznam použitýCH INFORMAČNÍCH ZDROJů

[] cplusplus.com - The C++ Resources Network. *cplusplus.com - The C++ Resources Network* [online]. Copyright © cplusplus.com, 2000 [cit. 29.12.2020]. Dostupné z: <https://www.cplusplus.com/>

[2] ESP32 Web Server using SPIFFS (SPI Flash File System) | Random Nerd Tutorials. *Random Nerd Tutorials | Learn ESP32, ESP8266, Arduino, and Raspberry Pi* [online]. Copyright © 2013 [cit. 29.12.2020]. Dostupné z: <https://randomnerdtutorials.com/esp32-web-server-spiffs-spi-flash-file-system/>

[3] Input Data on HTML Form ESP32/ESP8266 Web Server Arduino IDE | Random Nerd Tutorials. *Random Nerd Tutorials | Learn ESP32, ESP8266, Arduino, and Raspberry Pi* [online]. Copyright © 2013 [cit. 29.12.2020]. Dostupné z: <https://randomnerdtutorials.com/esp32-esp8266-input-data-html-form/>

[4] JavaScript Tutorial. *W3Schools Online Web Tutorials* [online]. Dostupné z: <https://www.w3schools.com/js/default.asp>

[5] HTML Tutorial. *W3Schools Online Web Tutorials* [online]. Dostupné z: <https://www.w3schools.com/html/default.asp>

[6] ArduinoJson: Efficient JSON serialization for embedded C++. *ArduinoJson: Efficient JSON serialization for embedded C++* [online]. Copyright © Benoît Blanchon [cit. 29.12.2020]. Dostupné z: <https://arduinojson.org/>

[7] ESP32 NTP Client-Server: Get Date and Time (Arduino IDE) – EmbedGyan Pvt Ltd. *EmbedGyan Pvt Ltd – We are a hybrid unit of resourcefulness*[online]. Dostupné z: <https://embedgyan.wordpress.com/2020/07/06/esp32-ntp-client-server-get-date-and-time-arduino-ide/>