Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Chytrý záhon pomocí platformy Arduino

Bc. Jakub Neidl

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jakub Neidl

Systémové inženýrství a informatika
Informatika

Název práce

Chytrý záhon pomocí platformy Arduino

Název anglicky

Automation of farming using Arduino platform

Cíle práce

Cílem diplomové práce je vytvoření rozšířitelného softwarového řešení pro automatizaci záhonků za využití běžně dostupných IoT komponentů. Mezi dílčí cíle práce patří navržení software pro ovládání záhonu vytvoření backendu a vytvoření frontendu.

Metodika

V diplomové práci bude provedena analýza požadavků konkrétního řešení. Na základě poznatků z teoretické části bude na bude navrhnuto vhodné a softwarové řešení. Využity budou mikrokontrolery, mikropočítače a senzory prostředí. Výsledné řešení bude na závěr zhodnoceno.

Doporučený rozsah práce

50-60 stran

Klíčová slova

IoT, SmartFarming, Arduino, automatizace záhonků, C++, ESP32, ESP8266, Java, Wiring, Spring framework, Vue.js, Chart.js

---

Doporučené zdroje informací

Banzi, Massimo, and Michael Shiloh. Getting started with Arduino. Mumbai: Shroff Publishers & Distributors Pvt. Ltd, 2015. ISBN 9789351109075.

Kurniawan, Agus. Internet of Things projects with ESP32 : build exciting and powerful IoT projects using the all-new Espressif ESP32. Birmingham, UK: Packt Publishing, 2019. ISBN 1789956870.

SELECKÝ, Matúš. Arduino: uživatelská příručka. Přeložil Martin HERODEK. Brno: Computer Press, 2016. ISBN 9788025148402.

Shovic, John C. Raspberry Pi IoT projects : prototyping experiments for makers. Berkeley, CA New York, NY: Apress,Distributed to the Book trade worldwide by Springer, 2016. ISBN 1484213785.

VIRIUS, Miroslav. Programování v C++: od základů k profesionálnímu použití. Praha: Grada Publishing, 2018. Myslíme v.. ISBN 9788027105021.

Předběžný termín obhajoby

2020/21 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 30. 3. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 30. 3. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 31. 03. 2021

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Chytrý záhon pomocí platformy Arduino" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31. 3. 2021 _____

Poděkování

Rád bych touto cestou poděkoval panu Ing. Marku Píckovi, Ph.D. za vedení mé diplomové práce, trpělivost, věcné připomínky, odborné rady a vstřícnost při konzultacích. Dále bych chtěl poděkovat své rodině za podporu při studiu.

Chytrý záhon pomocí platformy Arduino

Abstrakt

Tato diplomová práce se věnuje vývoji software pro chytrý záhon. Za pomocí mikrokontrolerů, krokových motorů a senzorů pro snímání dat z prostředí.

V teoretické části práce jsou pospány mikrokontrolery, jednotlivé komponenty a technologie, které jsou využité k realizaci projektu. Mezi komponenty patří Arduino, moduly ESP, senzor vlhkosti vzduchu, senzor teploty, senzor vlhkosti půdy, senzor teploty půdy. Mezi použité technologie patří framework Spring, framework Vue.js, framework Wiring.

V praktické části práce je zpracována analýza požadavků. Na základě této analýzy a poznatků získaných z teoretické části je vyvinuto řešení. Toto řešení je rozděleno do čtyř částí. Závěr práce je věnován ekonomickému zhodnocení, vyhodnocení úspěšnosti splnění požadavků a budoucímu vývoji.

Klíčová slova: Automatizace, ESP32, ESP8266, Arduino, Java, C++, Wiring, Spring, Vue.js, Chart.js, JavaScript

Automation of farming using Arduino platform

Abstract

This diploma thesis deals with the development of software for a smart garden. Using microcontrollers of stepper motors and sensors for reading data from the environment.

The theoretical part of the thesis describes microcontrollers, individual components, and technologies that are used to implement the project. Components include Arduino, ESP modules, humidity sensor, temperature sensor, soil humidity sensor, soil temperature sensor. The technologies used include the Spring framework, the Vue.js framework, and the Wiring framework.

In the practical part of the thesis, an analysis of requirements is processed. Based on the analysis of requirements and knowledge obtained from the theoretical part, a solution is developed. This solution is divided into four sections. The conclusion is devoted to economic evaluation, evaluation of the success of meeting the requirements, and future development.

Keywords: Automation, ESP32, ESP8266, Arduino, Java, C++, Wiring, Spring, Vue.js, Chart.js, JavaScript

Obsah

1 Úvod

S klesající cenou mikrokontrolerů je v dnešní době automatizace čím dál populárnější. Ať už se jedná o výrobní proces nebo rozsvěcení světel u vás doma. Jedním z populárních témat je automatizace pěstování rostlin. Příkladem může být pouhé pravidelné zalévání anebo komplexní udržování klimatu ve skleníku. S rostoucí popularitou 3D tiskáren roste i dostupnost jejich komponentů, například krokových motorů, hliníkových extruzí a jiných částí potřebných k tvorbě víceosých manipulátorů. Dostupnost těchto komponentů otevírá dveře dalším pohledům na automatizaci pěstování rostlin.

Ve své diplomové práci se věnuji vývoji a možnostem využití platformy Arduino a jiných komponentů v oblasti pěstování rostlin. V teoretické části jsem se zaměřil nejprve na mikrokontrolery, programování, krokové motory a senzory. V praktické části se věnuji vývoji software pro ovládání chytrého záhonu. Tento chytrý záhon se dokáže pohybovat po celé ploše vytyčeného záhonu ve třech osách, změřit pomocí senzorů například vlhkost a teplotu půdy. Tyto informace zapsat a vytvořit z nich přehledný graf. Pomocí webového rozhraní si uživatel může přesně rozvrhnout rozsazení jednotlivých rostlin, a tím i přizpůsobit dostatečné odstupy, aby se rostlina mohla rozrůstat. Ve webové aplikaci má uživatel možnost nastavit individuální podmínky pro každý typ rostlin. Což umožňuje pěstování více druhů rostlin s rozdílnými nároky na prostor a zalévání. Chytrý záhon měří u každé rostliny zvlášť její vlhkost půdy. Na základě daných informací následně i může zavlažit jednotlivé rostliny pomocí vodní pumpy. Uživatel může ovládat chytrý záhon skrze webové rozhraní, a tak rozhodovat i o provedení jednotlivých úkonů.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem diplomové práce je vytvoření rozšířitelného softwarového řešení pro automatizaci záhonků za využití běžně dostupných IoT komponentů. Mezi dílčí cíle práce patří navržení software pro ovládání záhonu vytvoření backendu a vytvoření frontendu.

2.2 Metodika

V diplomové práci bude provedena analýza požadavků konkrétního řešení. Na základě poznatků z teoretické části bude navrhnuto vhodné a softwarové řešení. Využity budou mikrokontrolery, mikropočítače a senzory prostředí. Výsledné řešení bude na závěr zhodnoceno.

3 Teoretická východiska

3.1 Mikrokontrolery

3.1.1 Arduino

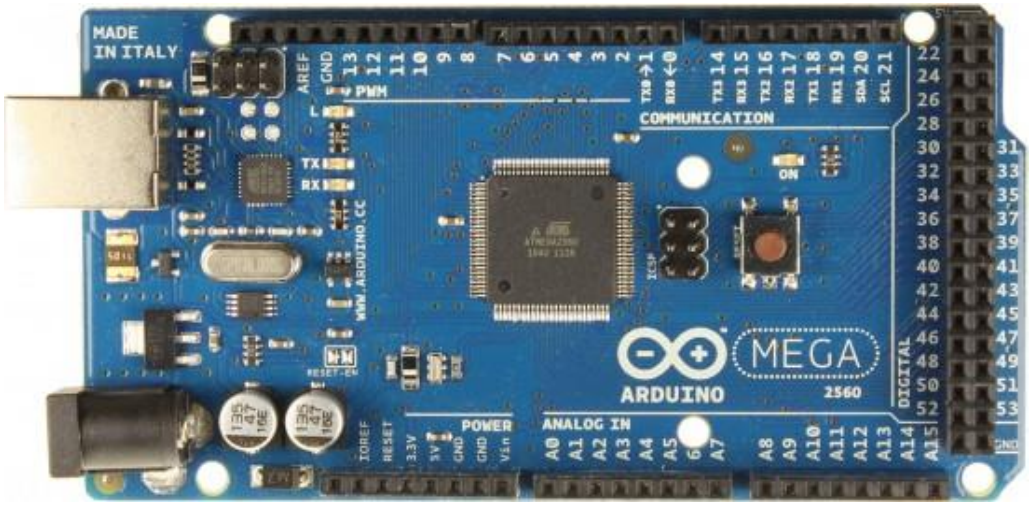
Arduino je open-source platforma, založená na mikrokontrolerech ATmega od firmy Atmel. V období 2005–2013 se prodalo 700 000 oficiálních zařízení Arduino. (1)

„Pomocí Arduina můžeme vytvářet interaktivní objekty. Arduino deska získává údaje od různých snímačů a senzorů (například snímač osvětlení, vzdálenosti nebo jen obyčejné tlačítko) a na základě těchto údajů ovládá nějaké výstupy (rozsvítí LED, zapne světlo nebo motor či jiný fyzický výstup). Aby Arduino deska vykonávala to, co potřebujeme, musíme vytvořit program pro Arduino mikrokontroler. Na to využijete programovací jazyk Arduino (založený na jazyce Wiring) a Arduino software (IDE), založené na prostředí Processig“ (2)

Platforma se hodí k rychlému prototypování a výrobě proof-of-concept řešení k prezentačním účelům. Velkým kladem této platformy je také její finanční dostupnost. Není příliš vhodná pro tvorbu koncových sériově vyráběných produktů. K jeho využití není potřeba hlubších znalostí principu elektroniky (1)

Arduino nabízí několik typů desek. Mezi nejznámější patří Arduino UNO, Mega, Leonardo a Micro. Výběr vhodné desky je nutné posoudit na základě požadavků projektu. Mezi hlavní rozhodující faktory patří počet analogových a digitálních vstupně-výstupních pinů, velikost paměti, rychlost mikroprocesoru a možnosti přidání rozšiřujících modulů. (1 str. 22)

Arduino Mega je deska designovaná pro projekty, které vyžadují vyšší počet pinů a jsou náročnější na hardware. Arduino mega nabízí 16 analogových a 54 digitálních vstupně-výstupních pinů, mikrokontroler ATmega2560, paměť o velikosti 32 KB. Je kompatibilní s většinou rozšiřujících modulů pro desku Arduino UNO.



Obrázek 1 Arduino Mega (3)

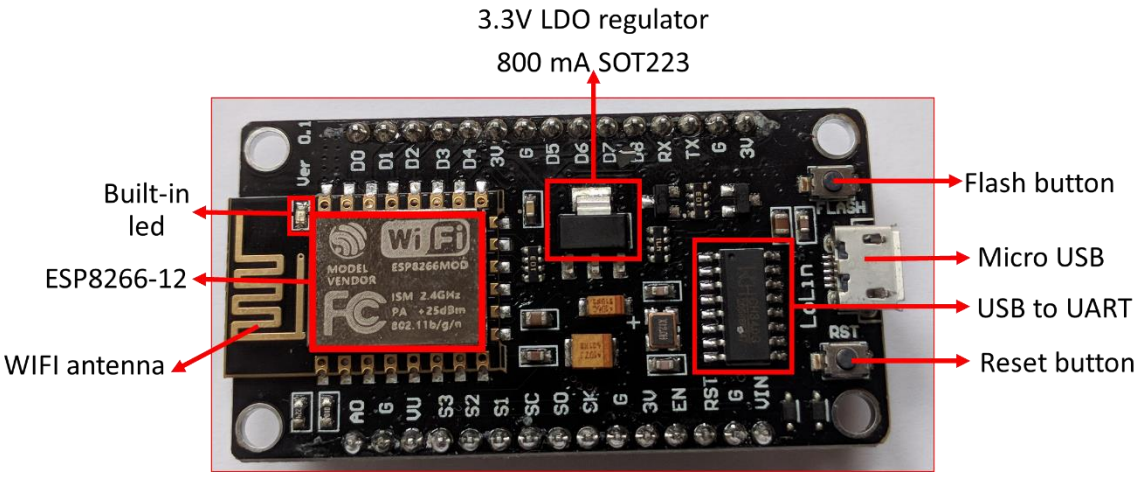
3.1.2 ESP 8266/32

Firma Espressif představila v roce 2014 nízkonákladový mikrokontroler, který nabízí WiFi konektivitu s označením ESP8266. V roce 2016 byl představen vylepšený modul ESP32. (4) ESP32 je následovník ES8266, ale to neznamená, že musí být náhradou. ESP32 nabízí rychlejší WiFi, CPU více GPIO pinů a podporu Bluetooth. (5)

Vlastnost	**ESP8266**	**ESP32**
MCU	Xtensa 1-Core 32-bitů	Xtensa 2-Core 32-bitů
Wi-Fi (rychlost up+down)	802.11 b/g/n, HT20 (max 130Mbit)	802.11 b/g/n, HT40 (max 300Mbit)
Bluetooth	Ne	v4.2
Frekvence	80-120 MHz	80-240 MHz

SRAM	160 kBajtů	512 KBajtů
GPIO	17	36
SPI / I2C / I2S / UART	2 / 1 / 2 / 2	4 / 2 / 2 / 2
ADC (Analog Digital Converter)	10-bitů	12-bitů
CAN (Controller Area Network)	X	✓
Cena	Cca 80 Kč	Cca 150 Kč
Představeno	2014	2016

Tabulka 1 Porovnání ESP32 a ESP8266 (Autor)



Obrázek 2 Vývojová deska ESP8266 (6)

3.2 Programování

3.2.1 Wiring

Jedná se o open-source framework pro programování široké škály mikrokontrolerů. (7) Tento framework byl vytvořen v roce 2003 jako součást diplomové práce na fakultě Interaction Design Institute of Ivrea (IDII) v Itálii. Cílem práce bylo ulehčení práce umělcům a designerům pracujícím s elektronikou. (1)

Program v jazyce Wiring se nazývá sketch. Sketch neboli skica se skládá ze dvou částí. Funkce **setup** a funkce **loop.** Setup se spouští pouze jednou, a to na začátku při spuštění programu. Využívá se pro inicializaci či nastavení objektů, knihoven anebo také pro nastavení vstupních a výstupních pinů. (8) Po skončení funkce setup, se spouští funkce loop. Cokoliv je obsahem této funkce se provádí cyklicky po sobě. Tato funkce umožňuje měnit data a využívat ji k ovládání Arduina. (9)

Arduino potřebuje obě tyto metody ve svém programu pro úspěšnou kompilaci a nahrání programu.

3.2.2 Java

Java je programovací jazyk nezávislý na platformě. Poprvé byla představena v roce 1995 společností Sun Microsystems. Jedná se objektově orientovaný jazyk. V tuto chvíli je to jeden z nejpopulárnějších programovacích jazyků. (10)

Java Maven

„Maven je rozšířený systém pro správu a sestavování aplikací postavených nad platformou Java. Jeho využitím odpadá závislost na konkrétním IDE, protože všechny informace potřebné ke kompilaci a sestavení programu jsou přímo obsaženy ve speciálních souborech ***pom.xml*** (POM = project object model). Systém Maven je již plně integrován do všech velkých IDE (Eclipse, Netbeans, IDEA) a tak je práce s ním opravdu velmi snadná i přes uživatelské rozhraní.“ (11)

Příklad nutné dependence pro vytvoření webové aplikace pomocí frameworku Spring:

```
<dependency>
     <groupId>org.springframework.boot</groupId>
     <artifactId>spring-boot-starter-web</artifactI
</dependency>
```

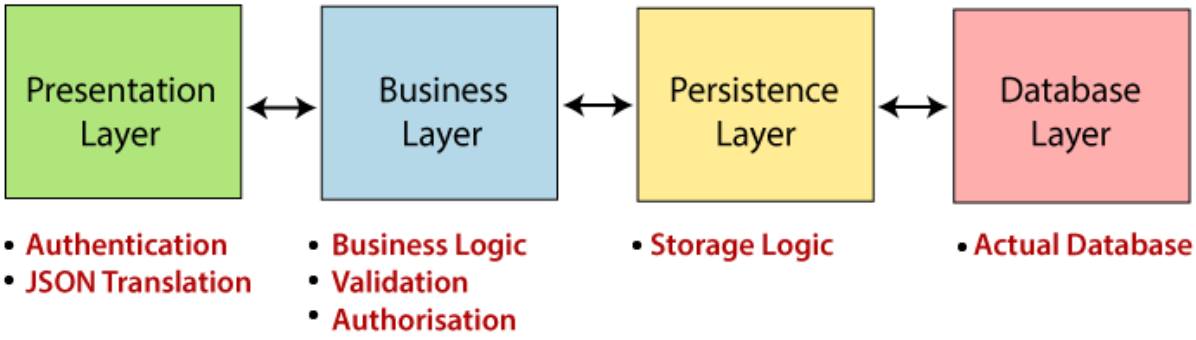
Každá dependence by měla obsahovat groupId, artifactId a verzi. U podřazených objektů není nutné specifikovat verzi. Díky tomu je pak možné projekty přenášet a spouštět na jiných zařízeních bez nutnosti ručně cokoliv nastavovat. (11)

Spring framework

Spring je vyvíjen firmou Pivotal a je k dispozici jako open-source framework. Cílem springu je zjednodušit vývoj aplikací odstraněním opakujícího se kódu nutného k vytvoření obvyklých funkcionalit. (12)

Spring je jeden z nejrozšířenějších enterprise frameworků. Nabízí spoustu funkcí a programovacích konvencí, které umožňují developerům psát program rychleji a jednodušeji. Spring boot je nadstavba frameworku Spring, která poskytuje další výhody. Je to standardní springová aplikace, kterou je možno rovnou spustit. Díky zabudovanému tomcat serveru, který se spouští spolu s aplikací. Toho je dosaženo principem convention above configuration**.** To znamená, že je nutné konfigurovat pouze věci, které chci mít konfigurované jinak, než je běžná praxe. (13)

Z hlediska budoucí udržitelnosti a použitelnosti kódu je dobré rozdělovat aplikaci do vrstev.



Obrázek 3 Spring architektura (14)

Prezentační vrstva je na vrchu architektury a skládá se z pohledů. Pohledy se starají o mapování objektů do formátu JSON a obráceně. Aplikační (Business) vrstva obsahuje logiku aplikace jako je validace, rozhodování, kalkulace, vyhodnocování, autorizaci. Vrstva persistence dat obsahuje logiku práce s databází. Zpracovává dotazy aplikace a překládá databázové entity na objekty a obráceně. Databázová vrstva se může skládat z databází jako je třeba PostgreSQL nebo H2. Může se skládat i z více databází najednou. Základní databázové operace CREATE, READ, UPDATE, DELETE se provádí na této vrstvě. (15) Jednotlivé vrstvy by měli vědět pouze o vrstvě pod sebou. Neměli by být schopny volat vrstvu nad sebou.

3.2.3 HTTP Požadavek

Komunikace webových aplikací funguje na základě posílání požadavků přes protokol HTTP. HTTP je komunikační protokol aplikační vrstvy. Používá se pro výměnu dat přes World Wide Web. Komunikaci začíná klient žádostí, typickým požadavkem bývá GET a cesta k webové stránce. Server poté vrátí odpověď s odpovídajícím stavovým kódem a tělem. V případě úspěšného požadavku typu GET na webovou stránku vrátí server stavový kód 200 a v těle HTML kód stránky, který je pak interpretován prohlížečem. (16)

Základní sada metod protokolu HTTP se skládá z příkazu GET, POST, DELETE. Příkaz GET načte požadované informace definované pomocí URL adresy serveru. (17) Příkladem URL u metody GET může být cesta „http://adresaserveru/sensors/humidityair“. Jedná se o dotaz na senzor s názvem „humidityair“. Struktura odpovědi následně záleží na logice serveru.

Metoda POST se používá v případě, že klient posílá informace na server. Nejčastěji to bývá vyplnění formuláře. Metoda DELETE se používá k mazání dat ze serveru. (17)

Požadavek se skládá z hlavičky a těla požadavku. Hlavička požadavku obsahuje typ metody, cestu na požadavek, informace o odesílateli, kódování a další. (18) Tělo požadavku v případě metody post může obsahovat objekt ve formátu JSON.

Odpověď se skládá také z hlavičky a těla odpovědi. V hlavičce odpovědi se nachází stavový kód, který reprezentuje úspěšnost požadavku. (19)

Nejpoužívanější stavové kódy jsou:

- Pro úspěšnou operaci
    - 200 OK
    - 201 Created
- Pro chybu na straně klienta
    - 400 – Bad request (Server nerozuměl požadavku.)
    - 404 – Not found (Server nemohl nalézt požadovaný zdroj.)
    - 408 – Request timout (Otevřené spojení je uzavřeno ze strany serveru buď z důvodu nevyužití spojení anebo dotaz trval moc dlouho.)
- Pro chybu na straně serveru
    - 500 – Internal Server Error (Na serveru nastala chyba.)

JSON

JavaScript Object Notation je formát pro výměnu dat. JSON reprezentuje JavaScriptový zápis objektů. Jeho výhodou je, že není složitý na strojové mapování a zároveň je jednoduše čitelný pro člověka. Používá se k předávání dat od klienta k serveru a naopak. Alternativou k JSONu je zápis ve formátu XML. (20)

Do formátu JSON lze uložit:

- Objekt
- Pole
- Číslo
- Řetězec znaků

- Boolean
- Null

Příklad požadavku POST pro vytvoření a uložení nového měření pro senzor temperature_air s tělem ve formátu JSON:

```
{
      "sensorName":"temperature_air",
      "sensorReadingSet":[
      {
      "reading": "30"
      }
]
}
```

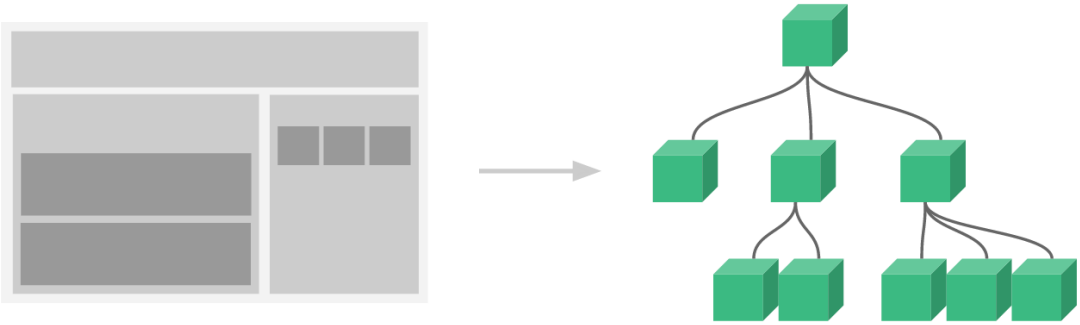
Alternativní zápis v XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <sensorName>temperature_air</sensorName>
  <sensorReadingSet>
    <element>
      <reading>30</reading>
    </element>
  </sensorReadingSet>
</root>
```

3.2.4 Vue.JS framework

„Vue.js je open-source JavaScriptový framework, který se využívá pro tvorbu uživatelských rozhraní. Vue.js je oproti jiným JS frameworkům více zaměřen na část view z MVC modelu. Také má oproti frameworkům jako je React nebo Angular daleko jednodušší syntax a je snazší ho přidat do projektu.“ (21)

Vue.js aplikaci je možné rozdělit na části a ty vyvíjet nezávisle. Jednotlivé části se nazývají komponenty. Komponenty je možné nasazovat postupně a libovolně. (22) „Každá komponenta funguje nezávisle a nese si vlastní HTML, CSS a JavaScript, díky čemuž může být vložena kamkoli a vždy se správně vykreslí.“ (22)



Obrázek 4 Vue.js Komponenty (23)

Reaktivita Vue spočívá ve sledování toku dat. Ve chvíli, kdy se změní definovaná proměnná změní se i její hodnota všude, kde je implementována.

```
<div id="app">
  {{ message }}
</div>

<script>
    const app = new Vue({
        el: '#app',
        data: {
            message: 'Hello Vue!'
        }
    })
</script>
```

V tomto případě kdykoliv bude změněna hodnota proměnné „message“ bude změněn i vypsaný text na stránce. (24)

Základní atributy Vue.js

Vue umožňuje manipulovat s daty a upravovat je v reálném čase pomocí vstupu od uživatele. Pomocí vytvoření dvoustranné vazby mezi vstupem a daty. Oboustrannou vazbu lze jednoduše vytvořit v prvcích html formuláře skrze atribut v-model. (25)

```
<input v-model="message" placeholder="edit me">
<p>Message is: {{ message }}</p>
```

Pro vytvoření jednostranné vazby mezi daty a komponentem nebo jiným prvkem html stránky se používá atribut **v-bind**.

```
<a v-bind:href="url"> ... </a>
```

K vizualizaci dat uložených v poli je možné použít atribut **v-for**, který je schopný vygenerovat html prvek nebo komponent pro každý objekt v poli. (26) Atribut **v-bind:key** by měl být vždy unikátní. V případě, že hodnota „message“ by byla duplikátní je nutné použít jako klíč index vypisovaného pole. (26)

```
<ul id="example-1">
  <li v-for="item in items" :key="item.message">
    {{ item.message }}
  </li>
</ul>
var example1 = new Vue({
  el: '#example-1',
  data: {
    items: [
      { message: 'Foo' },
```

```
    { message: 'Bar' }
  ]
 }
})
```

- Foo
- Bar

Obrázek 5 HTML výpis (Autor)

Responzivity webové aplikace lze také dosáhnout skrze použití atributu **v-if** a **v-else.** (27) Používají se k vygenerování komponentu nebo html prvku za určité podmínky.

```
<div v-if="Math.random() > 0.5">
  Now you see me
</div>
<div v-else>
  Now you don't
</div>
```

K vytvoření dynamického seznamu, který se po kliknutí otevírá a zavírá lze dosáhnout kombinací všech výše zmíněných atributu spolu s atributem **v-on:** který naslouchá událostem.

```
 <ul id="example-1">
   <li v-for="(item, index) in items" :key="index"
@click="toggle=!toggle">
     <p v-if="item.toggle">{{ item.message }}</p>
   </li>
 </ul>

var example1 = new Vue({
 el: "#example-1",
 data: {
   items: [
     { message: "Foo", toggle: true },
     { message: "Bar", toggle: true },
   ],
 },
});
```

- Foo
-

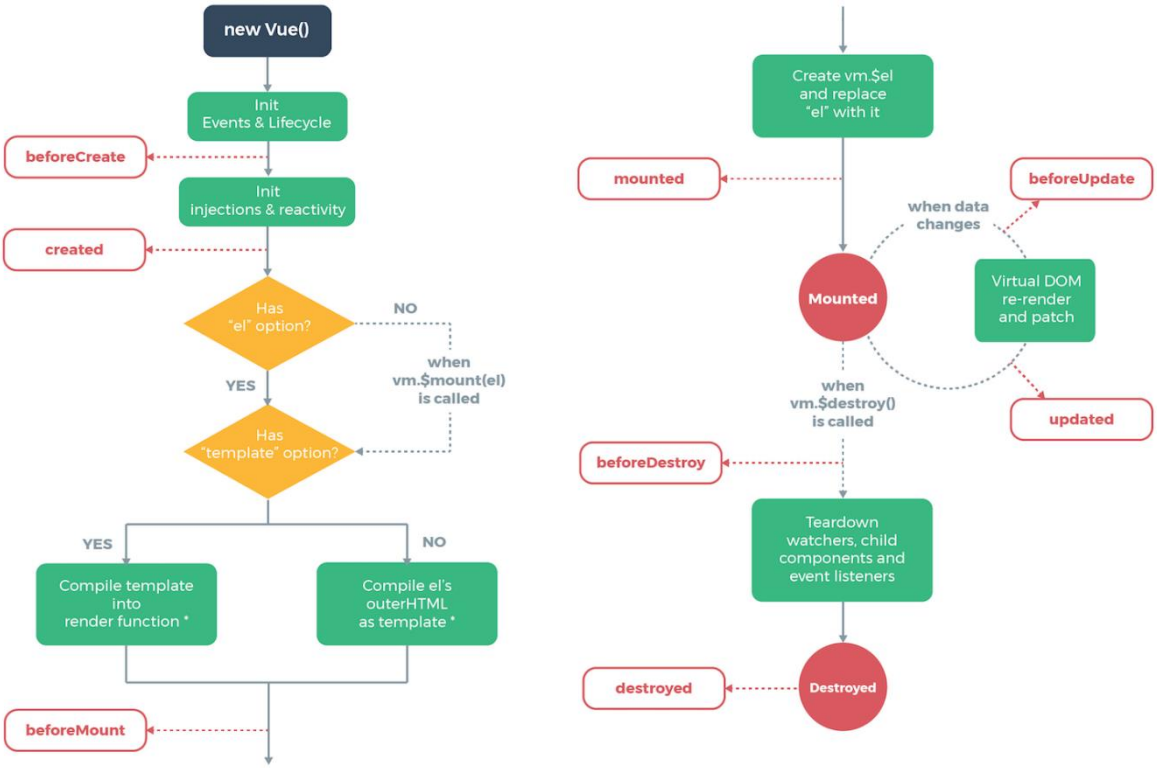
Obrázek 6 HTML výpis (Autor)

Životní cyklus Vue komponent

V červeně vyznačených sekcích na obrázku č. 7 se nacházejí tzv. „lifecyckle hooks“. Vyznačené události jsou metody, které se volají ve chvíli když komponent prochází danou částí životního cyklu.

- beforeCreate() proběhne pouze jednou, při inicalizaci komponentu,
- created() proběhne po incializaci objektu data,
- beforeMount() provede se před prvním vykreslením komponentu,
- mounted() proběhne po vykreslení komponentu,
- beforeUpdate() proběhne ve chvíli kdy dojde ke změne dat, ale předtím než je změna vykreslana,
- updated() proběhne při vykreslení komponentu po změně dat,

- beforeDestroy() proběhne před odstarnění komponentu,
- destroyed() proběhne jednou po odstranění komponentu (28)



Obrázek 7 Životní cyklus Vue komponenty (29)

Chart.js a Vue.js

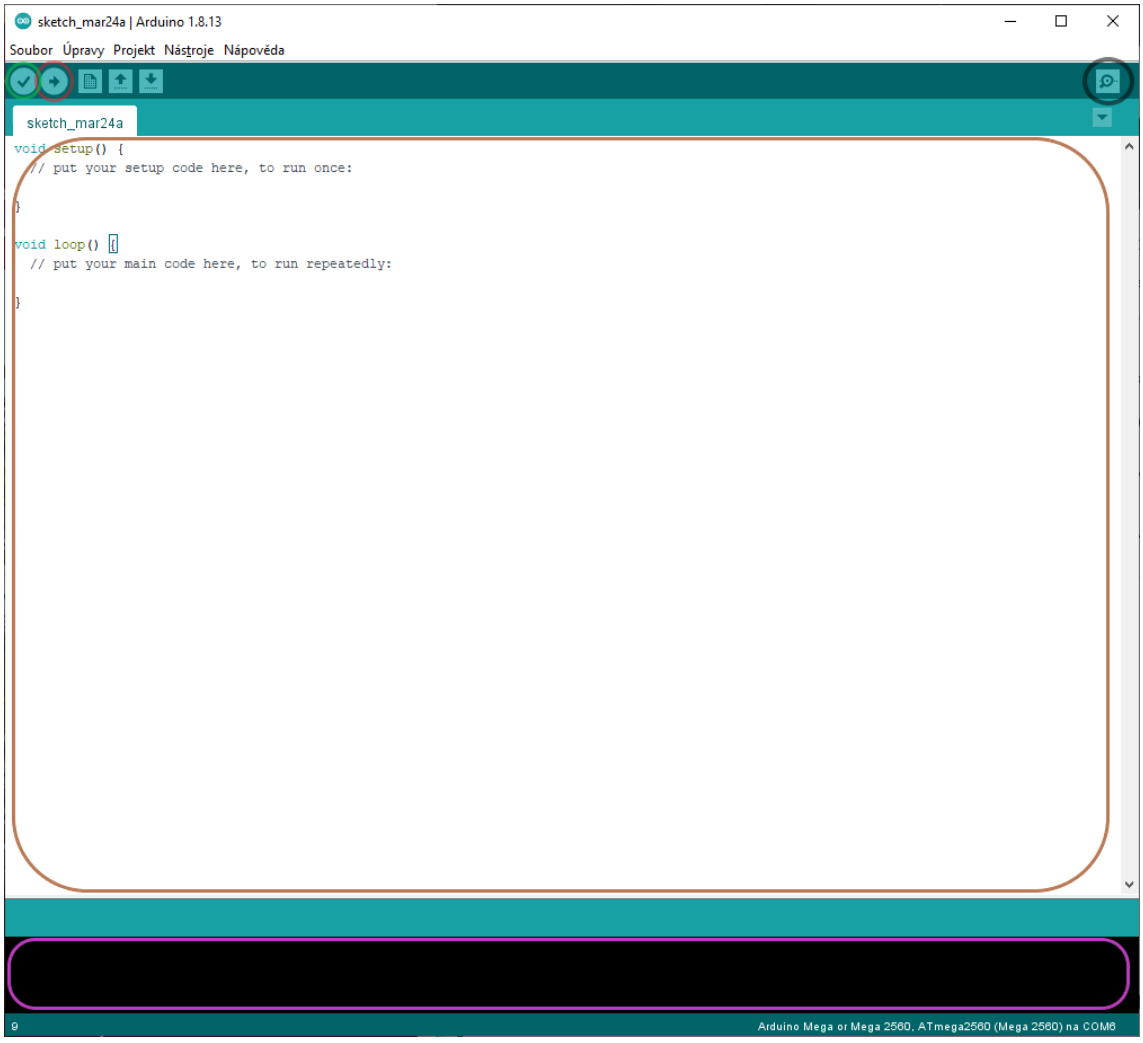
Chart.js je open source JavaScriptová knihovna, která je využívána k vizualizaci dat. Podporuje osm grafických typů zobrazení dat – sloupcové, čárové, bublinové, oblastní, koláčové, radarové, polární, korelační grafy. (30)

Pro zachování principu komponentů Vue je použit wrapper vue-charts, který umožňuje vytvořit z vybraného druhu grafu komponent a ten pak opakovaně vkládat na libovolná místa v aplikaci. (31)

3.2.5 Vývojová prostředí

Arduino IDE

Arduino Software IDE je open source vývojové prostředí navržené pro desky Arduino. (1) Je jedno z nejrozšířenějších vývojových prostředí pro platformu Arduino. Obsahuje editor, kompilátor, debugger, sériový monitor. Nahrát program do Arduina je možné jedním kliknutím na ikonu šipky. Program kód zkompiluje a nahraje na desku připojenou k počítači.



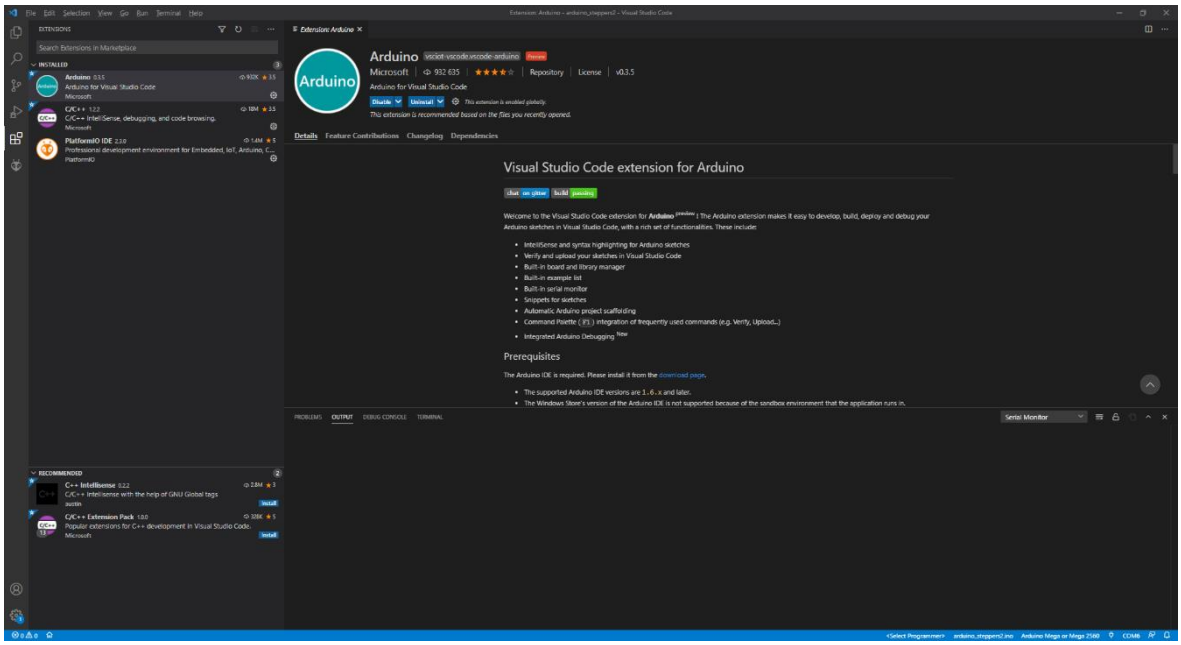
Obrázek 8 Arduino IDE (Autor)

V hnědě označené části Arduino IDE se nachází textový editor C/C++ s wiring knihovnami. Ve spodní fialově označené části se nachází stavové okno, které poskytuje informace o průběhu kompilace a nahrávání. V černě označeném pravém rohu se nachází

sériový monitor sloužící ke komunikaci s Arduinem po sériové lince. Zelený a červený kruh označují ikony sloužící k ověření kódu a nahrání kódu na desku Arduino. Alternativou k vývojovému prostředí Arduino je Visual Studio Code.

Visual Studio Code

Visual Studio Code je Open Source vývojové prostředí od společnosti Microsfot. Primárně určené pro editaci zdrojového kódu. Je veřejně dosputný pod licencí MIT. Tento software můžete používat zdarma k nekomerčním i komečním účelům. (32) Stejně jako Arduino IDE, IntelliJ IDEA je i VS Code k dispozici pro Windows, macOs a Linux.



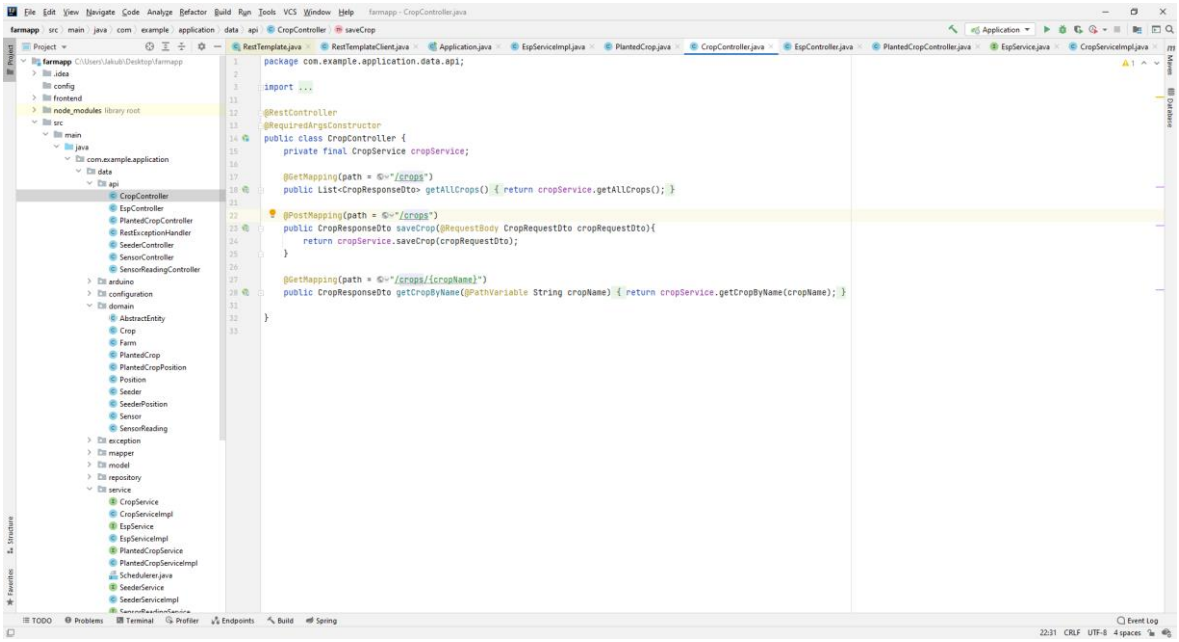
Obrázek 9 Visual Studio Code (Autor)

Díky otevřenosti tohoto vývojového prostředí jsou k dispozici stovky jednoduše nainstalovatelných doplňků. Jedním z nich je i doplněk pro Arduino. Po instalaci doplňku je třeba nastavit desku, programátor desky a baudrate. Oproti Arduino IDE je nastavení desky méně intuitivní.

IntelliJ IDEA

IntelliJ Idea je komerční produkt vytvořený firmou JetBrains, která sídlí v České republice. Mají v nabídce mnoho vývojářských nástrojů. IntelliJ IDEA je jeden

z nejznámějších a nejčastějších vývojových prostředích používaných mezi profesionálními vývojáři. Firma JetBrains nabízí placenou Ultimate edici, tak i komunitní, která je zdarma. (33)



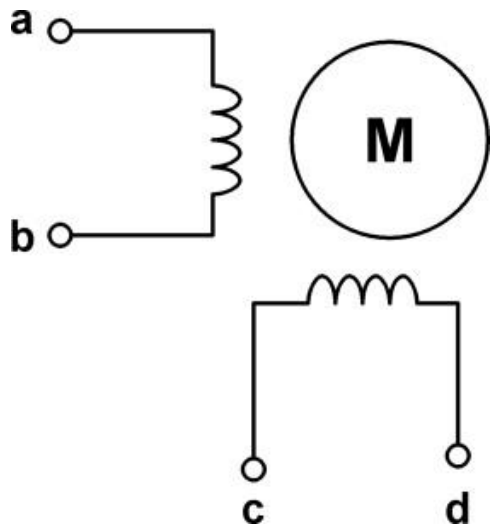
Obrázek 10 IntelliJ IDEA (Autor)

3.3 Hardware

3.3.1 Krokové motory

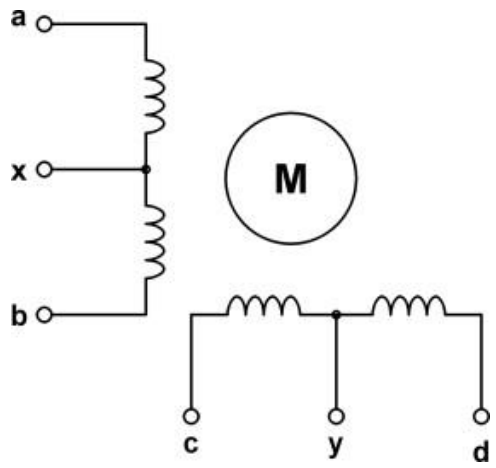
Krokové motory jsou složeny ze statoru a rotoru stejně jako stejnosměrný motor. Krokový motor rozděluje svůj pohyb mezi stejně velké menší pohyby, které nazýváme kroky. Počet kroků v jedné otáčce je daný úhlem kroku. (34) Nejtypičtější je 1,8 stupně. To znamená, že takový motor musí udělat 200 kroků, aby se otočil o 360 stupňů.

Krokové motory mohou dosahovat velice vysokých přesností. Proto jsou nejčastěji používané v 3D tiskárnách, CNC strojích, ale také třeba v Blu-Ray přehrávačích. (34) Krokové motory mají cívky organizované do skupin s názvem fáze. Podle stylu zapojení se krokové motory rozdělují na unipolární a bipolární.



Obrázek 11 Bipolární krokový motor (35)

Bipolární krokový motor má dvě identické na sobě nezávislé cívky. Využívá čtyř drátů. K změně směru pohybu je třeba změnit směr elektrického napětí na cívkách. Výhodou bipolárních krokových motorů je vysoký točivý moment. (35)

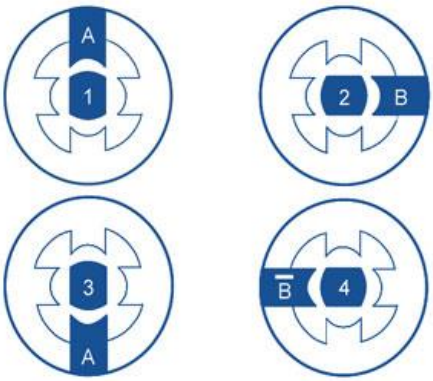


Obrázek 12 Unipolární krokový motor (34)

Unipolární krokové motory mají též dvě identické cívky, ale na rozdíl od bipolárních mají ve středu cívky napojený drát, který rozděluje cívku na dvě stejné. Většinou mají šest drátů, ale mohou mít také sedm nebo osm, podle počtu napojených drátů na dané cívky. Ke

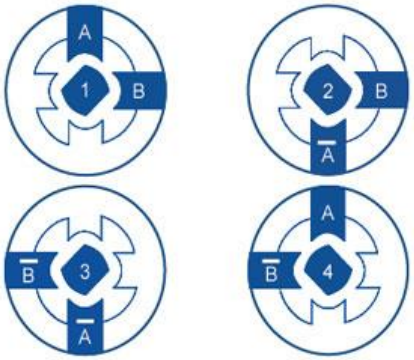
změně směru není třeba měnit směr napětí. Nabízí vyšší rychlost, ale nižší točivý moment než bipolární motory. (34)

Unipolární motory lze využívat i jako bipolární, pokud zapojíme jen dva hlavní vodiče od každé cívky. V případě obrázku č. 9 by to znamenalo zapojení pouze vodičů a b, c d. (36) Řídit krokové motory lze čtyřmi různými základními metodami. Vlnovým řízením, které se také nazývá jedno fázové řízení. Kdy spínáme jednu fázi po druhé a ta přitahuje rotor. (37)



Obrázek 13 Vlnové řízení (38)

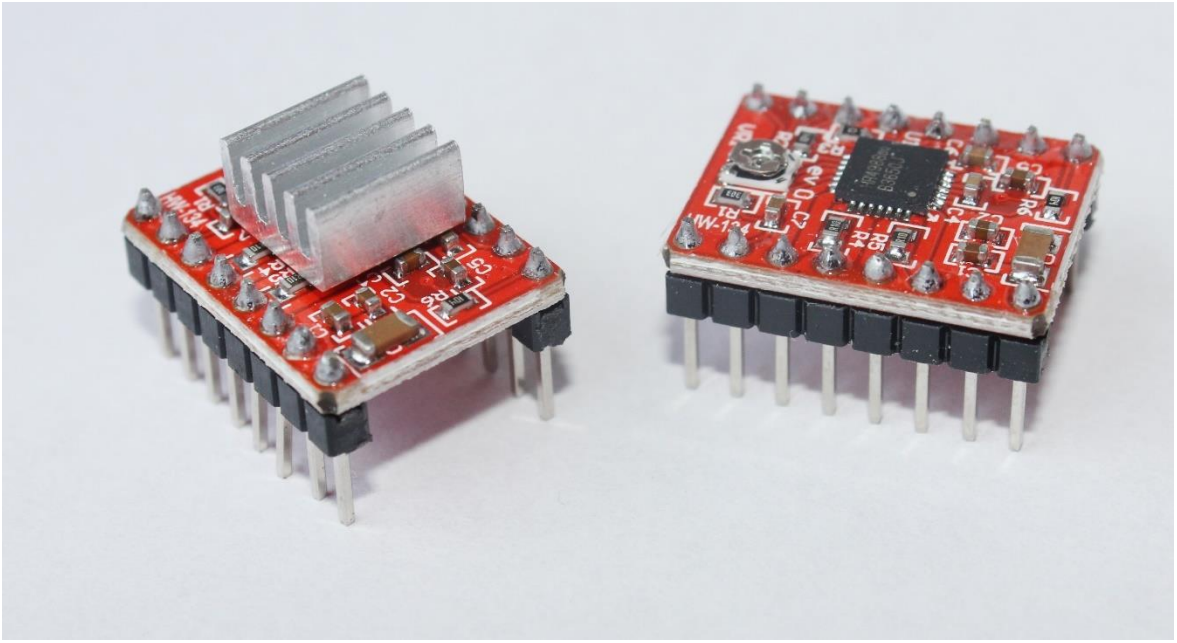
Dvoufázové řízení umožňuje vyšší točivý moment. Dvě sousedící fáze jsou sepnuty najednou a přitahují zuby rotoru k sobě. Při použití dvou fázového řízení je točivý moment až o 41% vyšší než u jednofázového. (37)



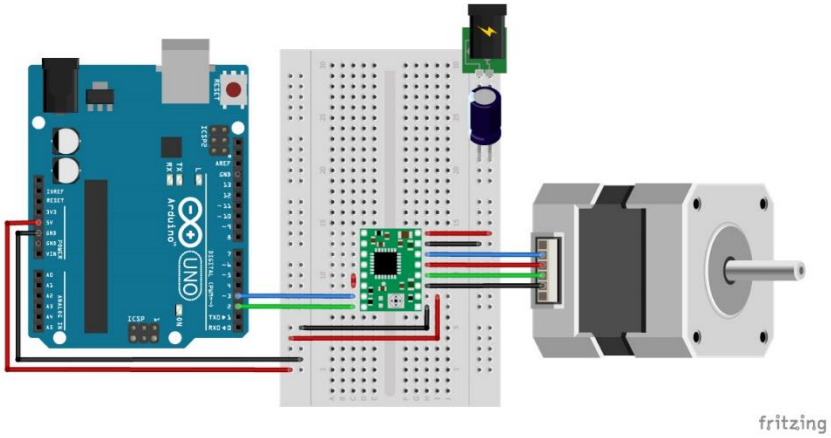
Obrázek 14 Dvoufázové řízení (39)

Řadiče krokových motorů

Pomocí řadičů krokových motorů lze ovládat jejich pohyb. Fungují na principu spínání jednotlivých cívek v pořadí, tak aby došlo k plynulému otáčení rotoru. (40)



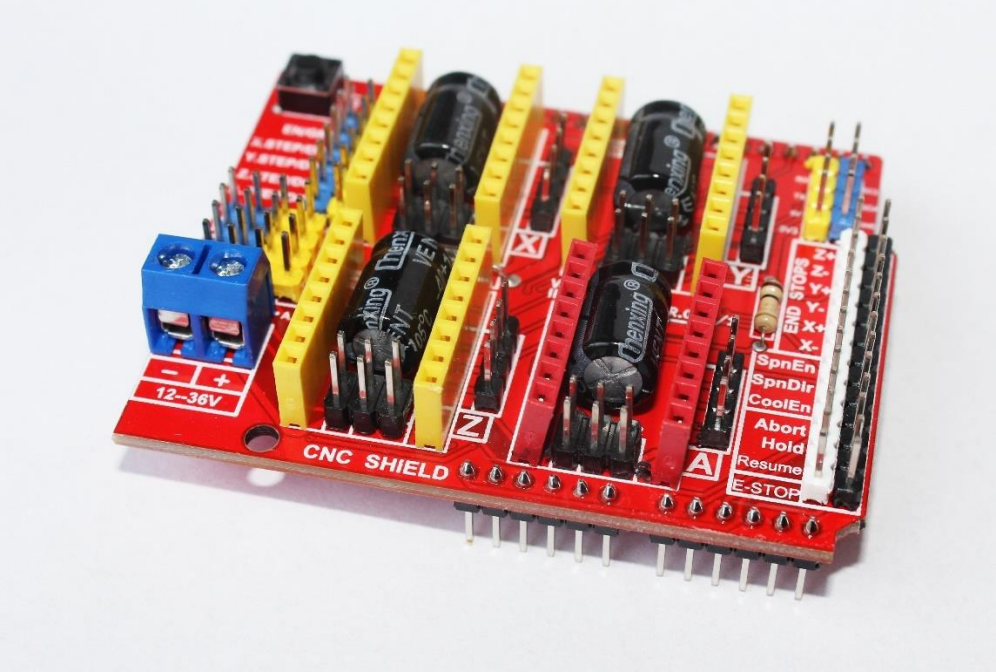
Obrázek 15 Řadiče krokových motorů A4988 (Autor)



Obrázek 16 Zapojení krokového motoru pomocí A4998 (41)

CNC sheild

Arduino lze rozšířit pomocí rozšiřujících externích modulů. Tyto moduly se nazývají „shieldy“. (1) CNC shield rozšíří Arduino o možnost připojení externího zdroje, který je zapotřebí k napájení krokových motorů a přidání obvodů potřebných k řízení krokových motorů. Tento rozšiřující modul ve verzi 3.0 umožňuje připojení až čtyř krokových motorů. Každý motor lze kontrolovat pouze za použití dvou digitálních pinů Arduina. Je velice populární volbou pro realizaci 3D tiskáren, CNC routerů a gravírovacích strojů. (42)

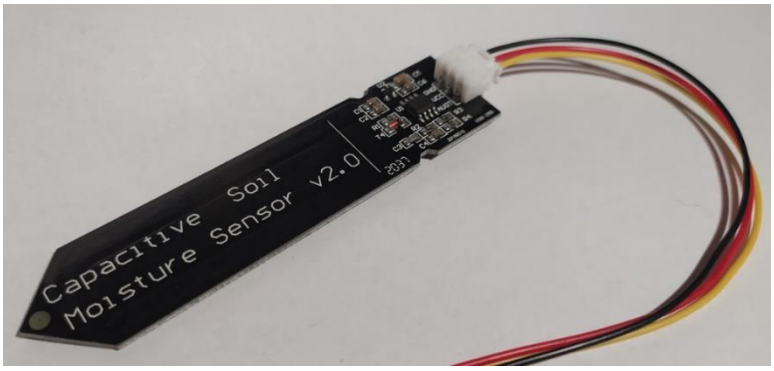


Obrázek 17 CNC Shield v3.0 (Autor)

3.3.2 Senzory

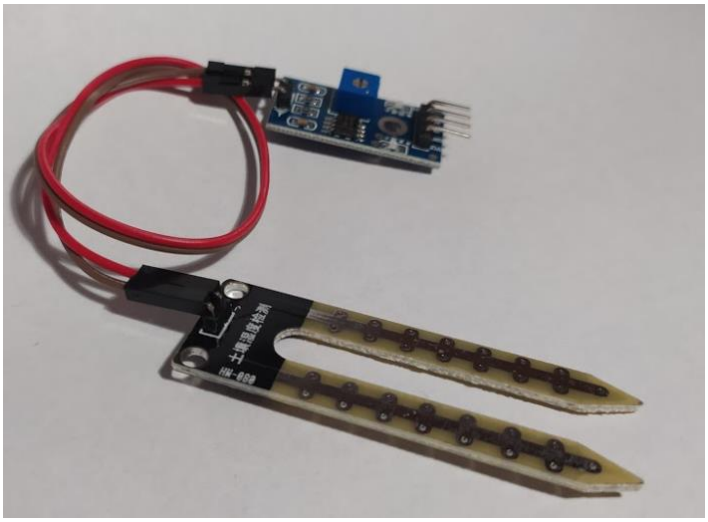
Vlhkoměr půdy

K monitorovaní vlhkosti půdy se nejčastěji používá kapacitní a odporový senzor. Oba druhy poskytují analogový výstup. (43) Kapacitní senzor nepoužívá styl dvou elektrod. Tento senzor nemá žádný kov vystavený na povrchu. Je zcela izolovaný. Izolované sondy senzoru tvoří kondenzátor. Při styku s vodou se mezi dvěma izolovanými elektrodami mění jejich kapacitní vlastnosti. (44)



Obrázek 18 Kapacitní senzor (Autor)

Odporový senzor vlhkosti půdy se skládá ze dvou elektrod. Funguje na principu měření změny napětí mezi anodou a katodou vůči výchozí hodnotě. Větší vlhkost půdy snižuje odpor mezi sondami, a tím je menší rozdíl napětí na výstupu děliče napětí oproti vstupu. (43) Je to velice jednoduchý a přímý systém měření.



Obrázek 19 Odporový vlhkoměr půdy (Autor)

Nevýhoda odporového senzoru je jeho životnost. Jednotlivé sondy nejsou izolované a díky průtoku stejnosměrného proudu dochází k elektrolýze. Kdyby však byly izolované, senzor by nemohl fungovat. A tak je spolehlivost senzoru časově omezená. (44)

Obrázek 20 Korozí zničený odporový senzor vlhkosti (45)

DHT11/DHT22

Senzory DHT11 a DHT22 snímají vlhkost vzduchu a teplotu vzduchu. Jejich výhodou je nízká cena a vysoká spolehlivost. DHT22 je oproti DHT11 dražší, ale co se týče specifikací lepší verzí. Výstupy senzorů jsou digitální. Zapojení u obou senzorů je stejné. Jejich nevýhodou oproti jiným senzorům je však pomalejší reakce na změnu prostředí. (46)

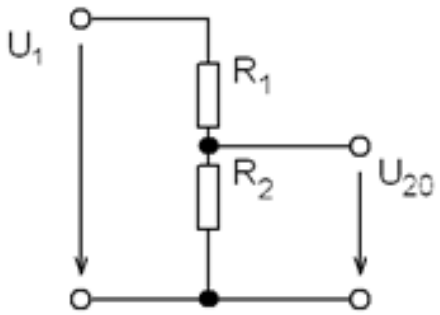
Typ	Teplotní rozsah	Přesnost měření teploty	Rozsah vlhkosti	Přesnost měření vlhkosti
DHT11	0 - 60°C	±2%	%5 to 95%	±5%
DHT22	-40 - 80°C	±0.5%	0 to 100%	±2%

Tabulka 2 Porovnání DHT11 a DHT22 (Autor)

Teploměr půdy

NTC teploměr (Negative Temperature Coefficient) je rezistor, který mění svůj odpor v závislosti na okolní teplotě. Odpor termistoru klesá při stoupající teplotě, a tak na výstupu děliče napětí čteme vyšší hodnotu napětí a tu poté interpretujeme pomocí dat z dokumentace senzoru. (47)

„Dělič napětí je zařízení, které umožňuje získání menšího napětí, než je svorkové napětí zdroje. Má buď pevné uspořádání k získání určitého konstantního napětí nebo má posuvný kontakt (odbočku), čímž se získá plynule proměnné napětí (tzv. *potenciometr*). Malý otočný potenciometr se užívá v elektronických zařízeních k jemnému nastavení napětí (tzv. *potenciometrický trimr*).“ (48)

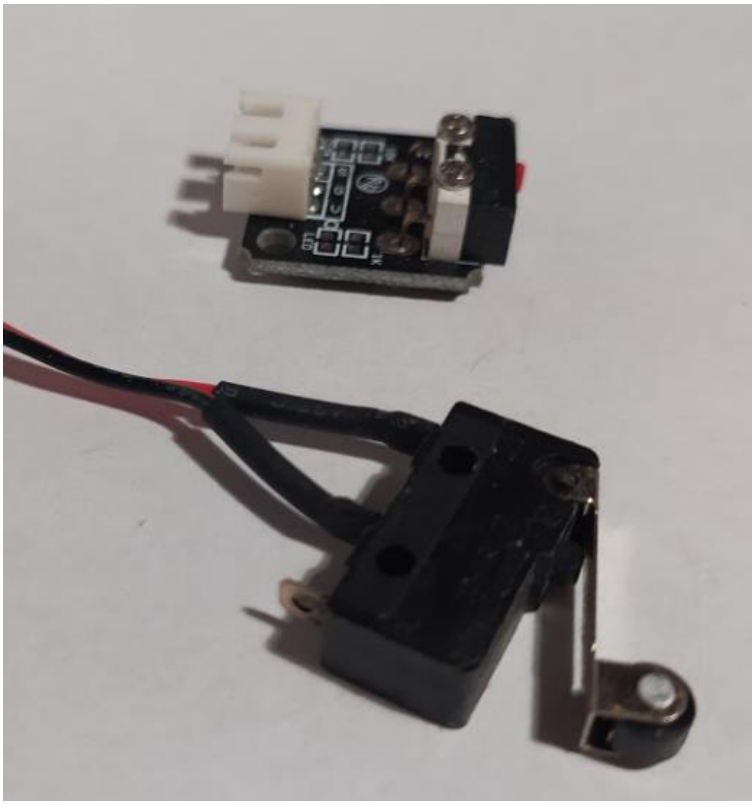


Obrázek 21 Schéma děliče napětí (49)

Mechanický koncový spínač

Krokové motory neznají svoji aktuální pozici. Na rozdíl od servo motorů, které mají omezený rozsah pohybu většinou na 180 stupňů. Rozsah pohybu je omezen, protože motor je spojen s potenciometrem, který se pohybuje zároveň s motorem, a tak podle odporu znají svou aktuální pozici. (50)

K zjištění aktuální pozice krokových motorů je třeba použít mezní spínače. Ty se mohou využít na začátku spuštění programu k zjištění počátku či konce osy po které se pohybuje. Plní tak referenční funkci, která umožňuje získání aktuální pozici na ose. (51)



Obrázek 22 Koncové spínače (Autor)

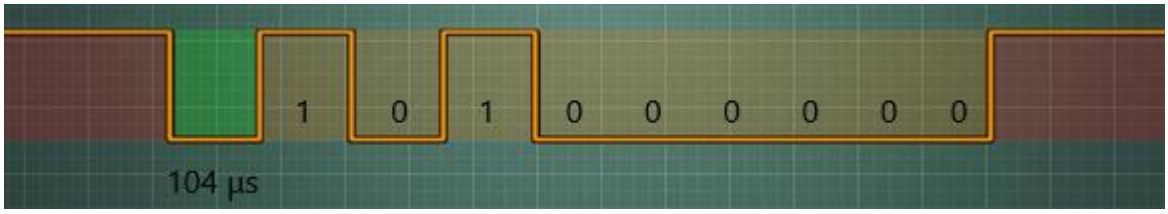
3.3.3 Sériová komunikace

Komunikace mezi dvěma zařízeními prostřednictvím posílání jednoho bitu po druhém přes jednu sběrnici. Sériovou komunikaci je možné dělit na synchronní a asynchronní. V případě synchronní komunikace zařízení, které vysílá a zařízení, které přijímá používají stejný časovač. Jedno zařízení (Master) řídí komunikaci a určuje druhému (Slave) kdy bude vysílat nebo přijímat. (52)

V případě asynchronní komunikace Arduino používá pro komunikaci s počítačem a jinými zařízeními rozhraní UART. Universal Asynchronus Reciver and Transmitor je rozhraní, které využívá pro komunikaci dvou drátů. Na rozdíl od synchronní komunikace každé zařízení využívá svůj vlastní časovač. Při přenosu je třeba určit, kdy přenos začíná a kdy končí. Toho se dosáhne pomocí prvního START bitu a END bitu. (52)

Baud je jednotka přenosu v asynchronní komunikaci. Je reprezentována bity za sekundu. Všechna zařízení, která se podílí na sériové komunikaci musí mít definovanou

stejnou rychlost komunikace. (52) Z rychlosti komunikace a pomocí startovního bitu můžeme pak začít číst vstup a následně ho interpretovat.



Obrázek 23 Komunikace po sériové lince (zpracováno zdroje (53))

4 Vlastní práce

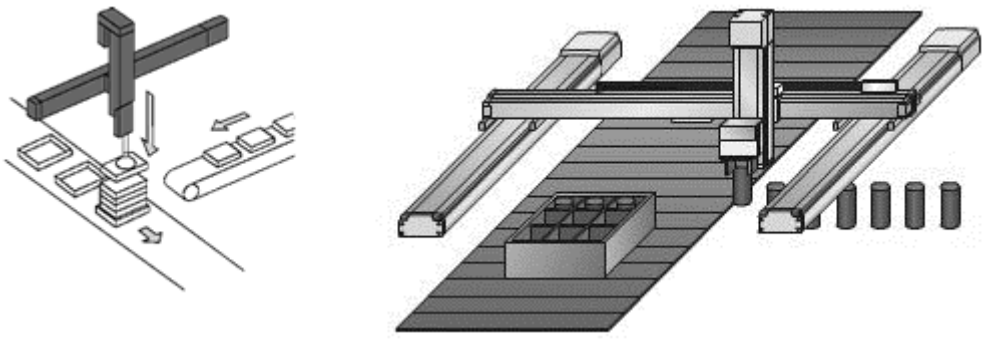
4.1 Analýza požadavků

Před realizací samotného návrhu je nutné definovat požadavky.

Mezi hlavní funkcionální požadavky patří:

- ovládání pohybu po jednotlivých osách,
- zavlažování rostlin jednotlivě,
- zavlažování všech rostlin,
- sběr dat ze senzorů,
- rozšiřitelnost o další senzory,
- měření vlhkosti pro každou rostlinu zvlášť,
- webová aplikace s uživatelským rozhraním pro manuální ovládání záhonu,
- grafické zobrazení dat přijatých ze senzorů ve webové aplikaci,
- zalévání rostlin na základě předdefinované hodnoty ideální vlhkosti,
- automatizace procesu měření vlhkosti a zalévání,
- bezdrátová komunikace mezi záhonem a serverem.

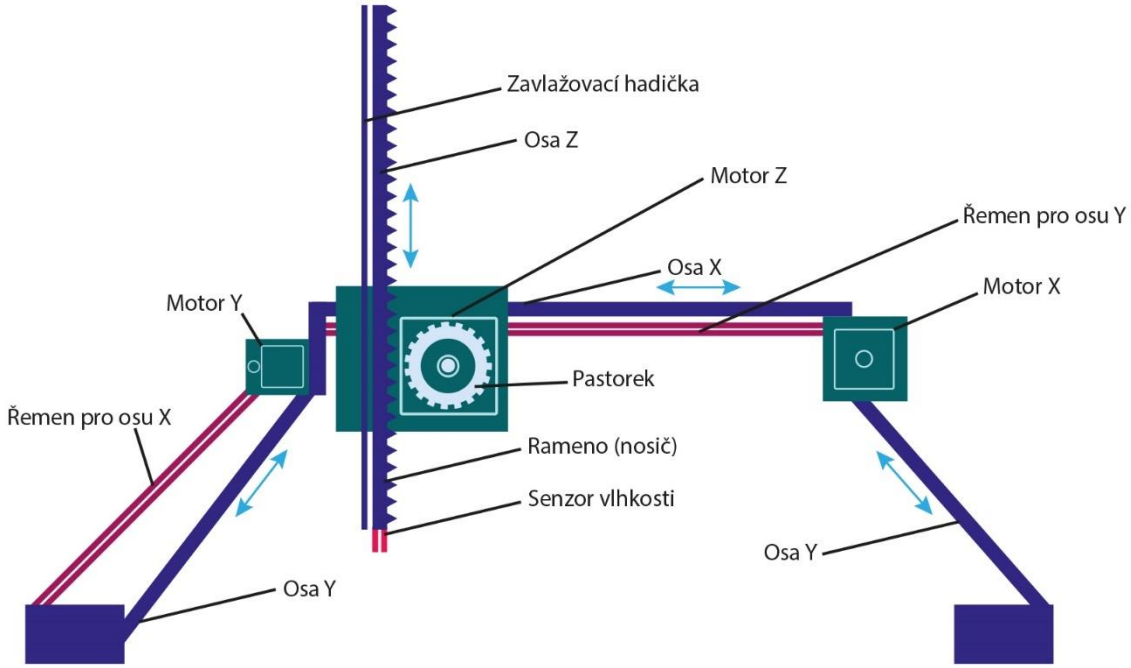
Zavlažování rostlin jednotlivě umožní zavlažení jen těch rostlin, které to budou potřebovat. Uživatel také bude schopný rozšířit rozhraní o libovolný počet senzorů. Mezi sledovanými daty by se měla nacházet vlhkost půdy pro jednotlivé rostliny zvlášť, teplota vzduchu, vlhkost vzduchu a teplota půdy. Výsledné řešení by pak mělo poskytovat data vhodná ke zlepšení pěstování jednotlivých rostlin. Data budou interpretována webovou aplikací, která bude poskytovat informace ve formě grafů. Ve webové aplikaci bude možné definovat typy rostlin a ty pak následně vkládat mezi zasazené rostliny. Webová aplikace bude reagovat dynamicky a zobrazovat pozice jednotlivých rostlin na záhonu.



Obrázek 24 tříosý manipulátor (54)

4.2 Návrh řešení

Pro splnění požadavku měření vlhkosti a zavlažování každé rostliny zvlášť je využit tříosý manipulátor vytvořený za pomocí hliníkových extruzí a krokových motorů.



Obrázek 25 Návrh konstrukce pro osazení záhonu (Autor)

Jako základní sada senzorů byly zvoleny následující senzory. Pro snímání vlhkosti vzduchu a teploty vzduchu senzor DHT22. K zjištění teploty půdy vodotěsný teploměr typu NCT a k měření vlhkosti půdy byl použit kapacitní i odporový senzor vlhkosti.

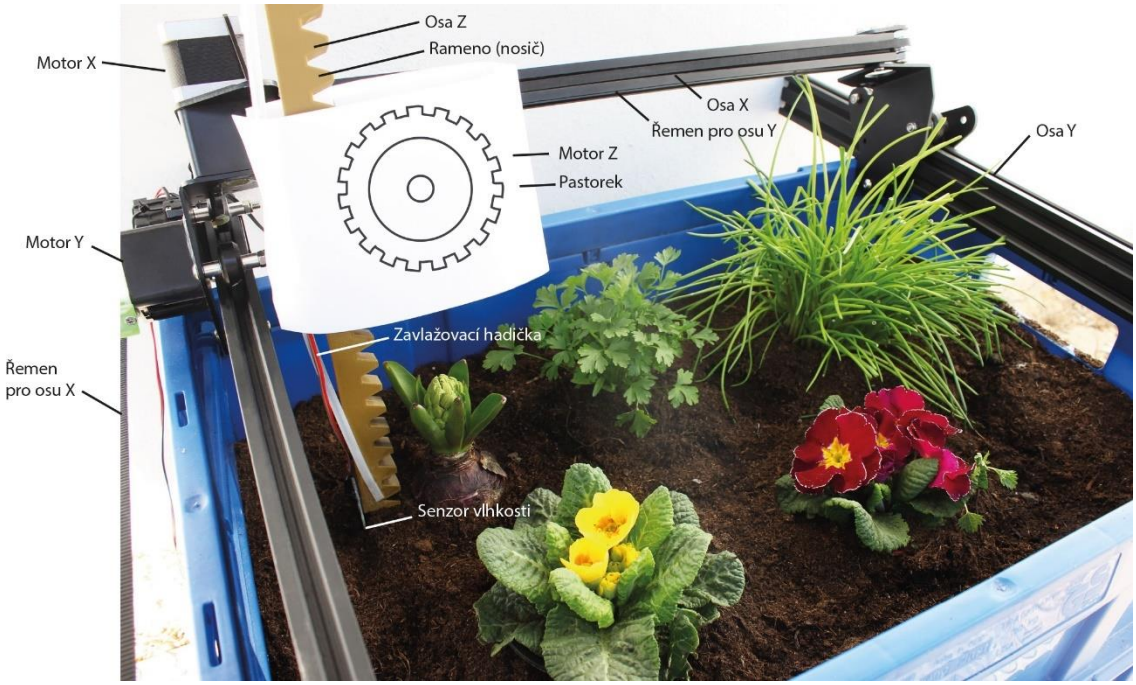
Pro získání dat ze senzorů, kontrolu pohybu a vodní pumpy bylo vybráno Arduino ve verzi Mega. Pro připojení do internetu byl zvolen modul ESP8266, který může být nahrazen i novějším modulem ESP32 v případě rozšíření nebo implementace vyššího zabezpečení, a tak zvýšení náročnosti na jeho hardware. Pro pohyb po osách X, Y, Z jsou použity bipolární krokové motory Nema 17 s přibližným točivým momentem cca 2.2 N/cm. Jejich pohyb řídí řadiče krokových motorů A4998 za pomocí rozšiřujícího modulu CNC Shield v3.

Arduino se stará čistě o ovládání jednotlivých komponentů, ale už neřeší hlubší logiku rozhodování. Arduino zpracovává základní úkony jako je pohyb, snímání dat ze

senzorů, spouštění vodní pumpy, vypnutí vodní pumpy. Nerozhoduje o tom, kdy jednotlivé úkony vykonat, jako například kde měřit, kdy a kde zalévat.

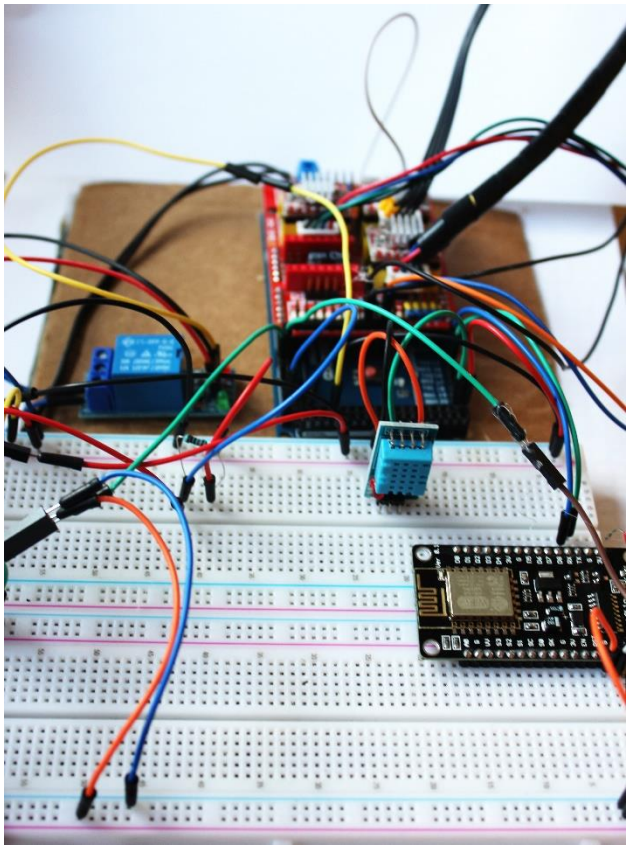
Tento „chytrý záhon“ umožňuje pěstování rostlin jednotlivě. Na základě zadaných hodnot udržuje u každé rostliny její požadovanou vlhkost půdy. Data ze senzorů zaznamenává a ty jsou interpretována ve formě grafů. Individuální zalévání, podle potřeby umožňuje pěstování více druhů rostlin v jednom záhonu. „Chytrý záhon“ kromě měření vlhkosti půdy u jednotlivé rostliny, zaznamenává obecnou vlhkost vzduchu, teplotu vzduchu i půdy.

Software byl testovaný na mnou vytvořeném prototypu. Tento prototyp jsem záměrně sestrojil z běžně dostupných IoT komponentů, což je výhodou i v pořizovací ceně.

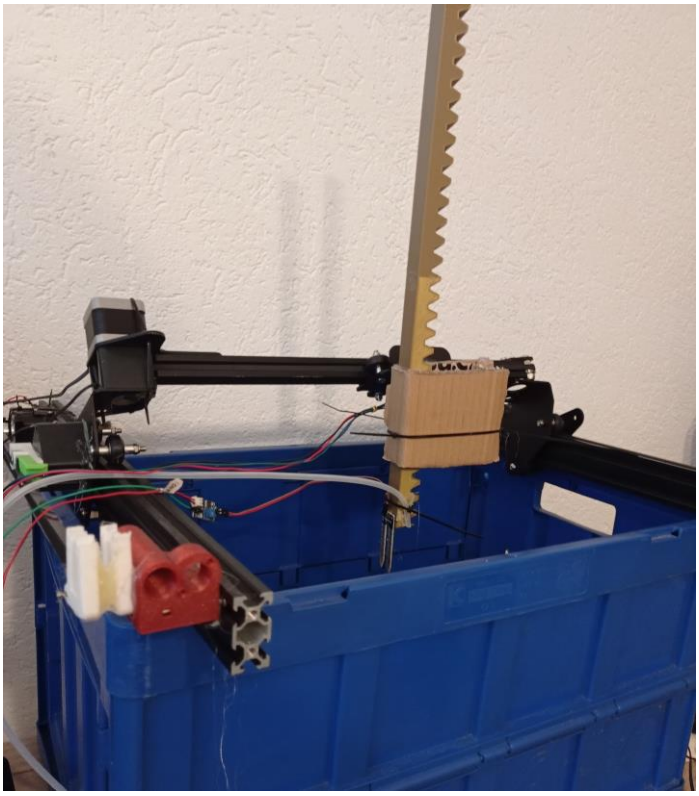


Obrázek 26 Prototyp "chytrého záhonu" při měření vlhkosti (Autor)

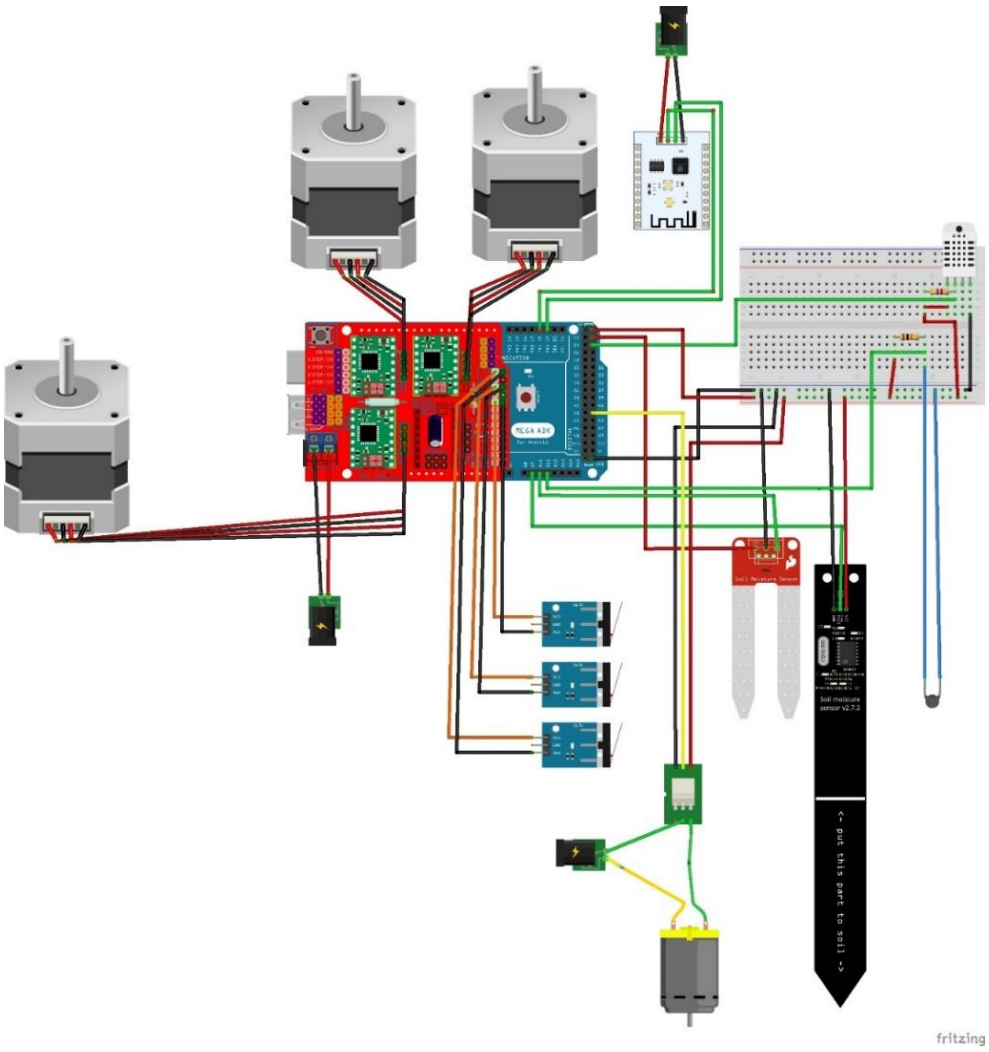
Řešení projektu je možné rozdělit do čtyř vrstev. Vrstva Audino, vrstva ESP, backend a frontend. Pomocí webové aplikace může uživatel přesně rozmístit rostliny na určité pozice, kde může zohlednit individuální rozpětí, které daná plodina potřebuje. Následným sběrem dat u jednotlivých rostlin je backend schopný vyhodnotit, kdy je potřeba rostlinu zalít. K zalití rostliny tento prototyp využívá vodní pumpy.



Obrázek 27 Zapojení komponentů během prototypování (Autor)



Obrázek 28 První zapojení všech komponentů (Autor)



Obrázek 29 Vizualizace jednotlivých komponentů na nepájivém poli (Autor)

4.2.1 Komunikace

Deska Arduino sama o sobě neumožňuje připojení do sítě. Proto byl vybrán modul ESP jako komunikační prostředník. ESP je fyzicky propojeno s Arduinem a komunikuje s ním přes sériovou linku pomocí rozhraní UART. ESP Zpracovává http požadavky od serveru a přeposílá je do Arduina.

Role serveru je sbírat a spravovat data získaná z Arduina. Na základě získaných dat poté rozhodnout o úkonech, které má Arduino provést. Komunikace probíhá pomocí předávání zpráv ve formátu JSON.

Mezi základní definované úkony patří pohyb, zasazení rostliny na pozici, zavlažení pozice, změření vlhkosti půdy na pozici, měření obecné vlhkosti půdy, měření teploty vzduchu, měření vlhkosti vzduchu a měření teploty půdy. Požadavek na vykonání úkonu je vždy ve formátu `"type":"název úkonu"`. Obsahem zprávy je i vše co je potřeba k vykonání daného úkonu. Po vykonání úkonu, Arduino odesílá potvrzující zprávu o dokončení.

Definování komunikace:
Požadavek na pohyb:
`{"type":"movement","positionX":"500","positionY":"100", "positionZ":"100"}`
Odpověď: `{"type":"response""}`

Zasazení rostliny:
`{"type":"plant","positionX":"500","positionY":"100", "positionZ":"50","seederPositionX":"0","seederPositionY":"0", "seederPositionZ":"50"}`
Odpověď: `{"type":"response""}`

Spuštění vodní pumpy: `{"type":"environment","request":"water"}`

Odpověď: `{"type":"response""}`

Získání vlhkosti půdy z ramenního senzoru:
`{"type":"environment","request":"soil_humidity_sensor"}`
Odpověď: `{"reading": "450", "value": "soil_humidity"}`

Získání obecné vlhkosti půdy:
`{"type":"environment","request":"soil_humidity_global"}`

Odpověď: `{"reading": "600", "value": "global_soil_humidity "}`

Získání teploty vzduchu:

```
{"type":"environment","request":"air_temperature"}
```

Odpověď: `{"reading": "752", "value": " °C "}`

Získání vlhkosti vzduchu:

```
{"type":"environment","request":"air_humidity"}
```

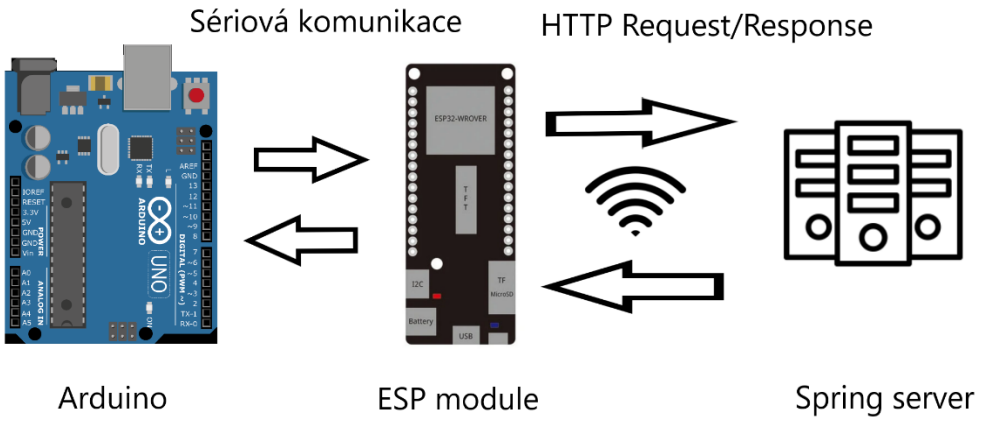
Odpověď: `{"reading": "50", "value": "%"}`

Získaní teploty půdy:

```
{"type":"environment","request":"soil_temp_sensor"}
```

Odpověď: `{"reading": "752", "value": " °C "}`

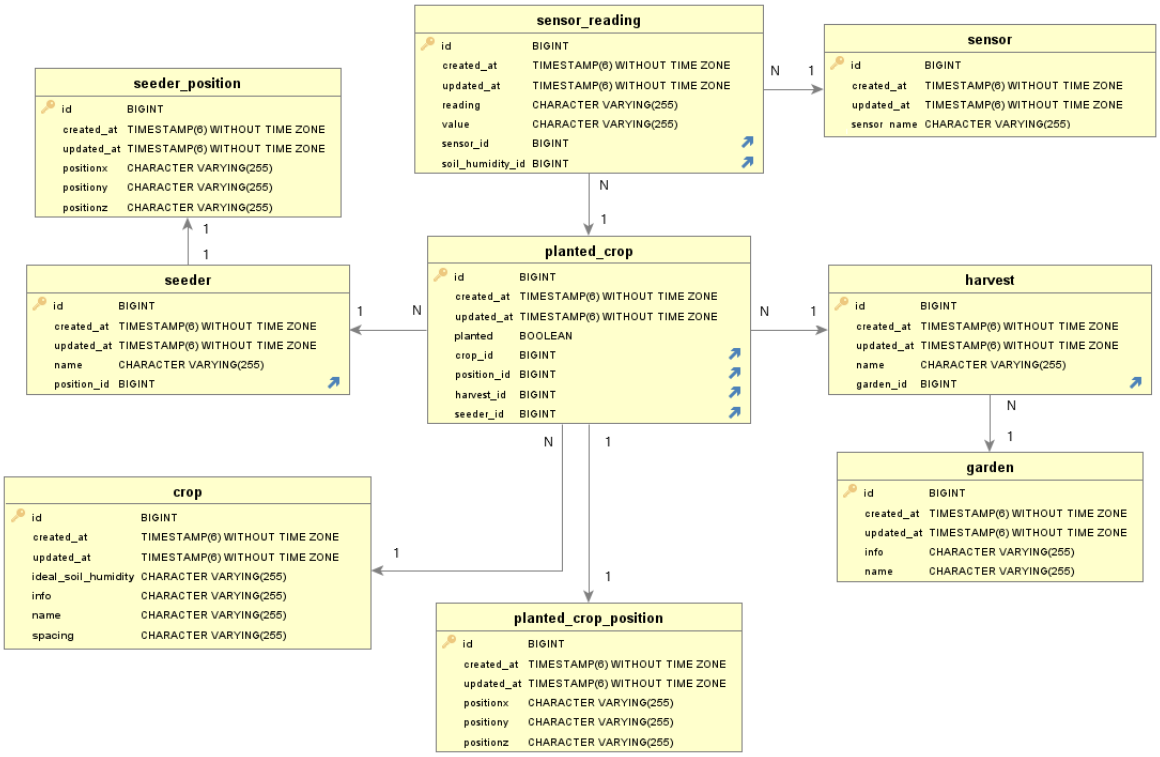
Zavlažení na pozici x500 a y100 lze dosáhnout kombinací požadavku `{"type":"movement","positionX":"500","positionY":"100"}`.
Ve chvíli, kdy server obdrží odpověď `{"type":"response""}` vyhodnotí, že byl úkon vykonán a nosič se nachází na požadované pozici. Server odešle další požadavek se zprávou ve formátu `{"type":"environment","request":"water"}`. Výsledkem je najetí nosiče nad požadovanou pozici a spuštění vodní pumpy. Doba zdělávání odpovídá času definovaném v programu Arduino.



Obrázek 30 Průběh komunikace (Autor)

4.2.2 ER diagram

Entitní model je vytvořen z tříd v balíku domain za pomocí frameworku Hibernate. Každá entita má atribut createdAt a updatedAt. Typu LocalDateTime. Hodnoty v těchto polích jsou generovány automaticky. Obě proměnné jsou naplněny datumem a časem při prvním zápisu do databáze. Proměnná updatedAt je přepsána aktuálním datumem a časem při každé úpravě entity.



Obrázek 31 Entitní model (Autor)

Entita garden reprezentuje záhon, který obsahuje atributy – jméno, info.

Entita harvest reprezentuje období od zasazení rostliny po její sklizení. Entita sensor reprezentuje senzor. Senzor obsahuje atributy jméno a cizí klíč odkazující na entitu sensor_reading.

Entita sensor_reading reprezentuje data čtená ze senzoru a obsahuje atributy value a reading. Atribut value obsahuje typ přečtené hodnoty vyjádřený symbolem např. % nebo °C. Pod atribut reading je uložena samotná hodnota naměřená senzorem.

Entita crop repererzntuje typ rostliny. Typ rostliny má jméno, idealní vlhkost půdy a místo které zabírá.

Entita seeder reprezentuje zásobník se sazenicemi. Tento zásobník je umístěn například na kraj záhonku. Pozice tohoto zásobníku je pak uložena v entitě seederPosition.

Entita plantedCrop reprezentuje zasazenou plodinu. Zasazená plodina má atribut planted typu boolean, která definuje jestli po přidání mezi zasazené rostliny již byla zasazesna. Zasazená rostlina má v sobě vazbu na typ rostliny. Díky této vazbě lze pak přistupovat k informacím o rostlině skze zasazenou rostlinu. Vazba s tabulkou sensor reading umožňuje měřit hodnotu vlhkosti pro každou rostlinu zvlášť. Abychom věděli kde je plodina zasazena, využívá vazby s tabulkou plantedCropPosition. Pozice by mohla být ukládána i přímo v entitě plantedCrop, ale jelikož se v implementaci projektu jedná o samostatný objekt je reprezentována samostatně.

Entita plantedCropPosition vytvořená na základě třídy PlantedCropPosition, která reprezentuje pozici zasazené rostliny.

Entita y seederPosition, vytvořená na základě třídy SeederPosition která reprezentuje pozici seedru.

Tímto datovým modelem můžeme sledovat data pro jednotlivá období (sklizně). Můžeme definovat více typů rostlin a z těch pak vybírat rostliny k sazbě. Samostatná tabulka pro senzory umožňuje přidání libovolného počtu senzorů a pro každou zasazenou rostlinu je možné sledovat vlhkost půdy zvlášť.

4.3 Implementace

4.3.1 Čtení dat ze senzorů

Teploměr NTC 10K

Získání teploty půdy z teploměru NTC 10k je řešeno pomocí funkce getSoilTemp(), která vrací výslednou teplotu ve stupních celsia.

Následující kód vychází z dokumentace senzoru. (55)

```
float getSoilTemp()
{
  float RT, VR, ln, TX, T0, VRT;

  T0 = 25 + 273.15;
  VRT = analogRead(A15);        //Acquisition analo
g value of VRT
  VRT = (5.00 / 1023.00) * VRT; //Conversion to vol
tage
  VR = VCC - VRT;
  RT = VRT / (VR / R); //Resistance of RT

  ln = log(RT / RT0);
  TX = (1 / ((ln / B) + (1 / T0))); //Temperature f
rom thermistorA

  TX = TX - 273.15; //Conversion to Celsius
  return TX;
}
```

Proměnná TX reprezentuje naměřenou hodnotu ve stupních Celsia.

Vlhkoměr DHT11

Pro čtení dat ze senzoru DHT11 je použita knihovna DHT. Pro práci se senzorem DHT je třeba na začátku programu definovat jeho typ a digitální pin ke kterému je připojen.

```
//---Air-Humidity-Temperature-Sensor---
//defining DHT sensor (humidity, temp) pins and typ
e
#define DHTPIN 22
#define DHTTYPE DHT11
```

```
//Sensors
//-Environment
DHT dht(DHTPIN, DHTTYPE);
```

K získání dat ze senzoru používáme metody třídy dht.readHumidity() a dht.readTemperature()

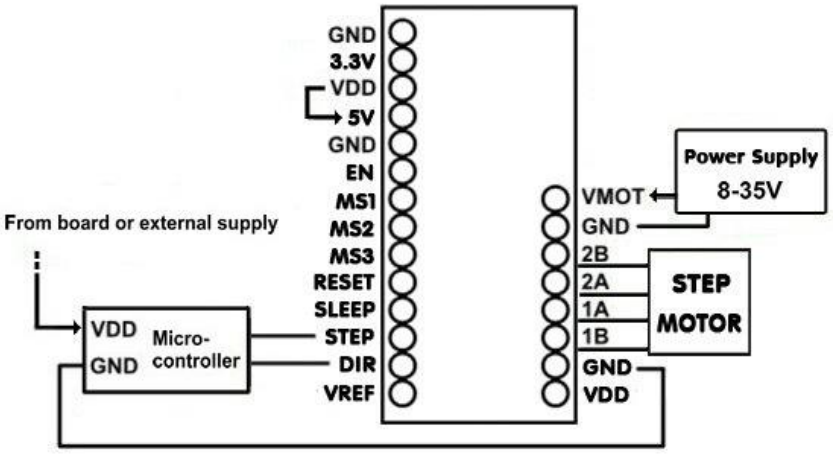
Senzory vlhkosti půdy

Odporový senzor vlhkosti půdy nám poskytuje analogovou hodnotu v přibližném rozpětí 300 až 1000. Kde 300 je hodnota čtení při kontaktu s vodou a 1000 při suchém čtení. (56) Hodnotu lze získat pomocí metody analogRead(SENSOR_PIN) a pinu ke kterému je senzor připojen. V případě kapacitního senzoru je analogová hodnota při čtení senzoru v přibližném rozsahu 250–600. Kde nižší hodnota reprezentuje vyšší vlhkost půdy. (44) Pro získání hodnoty vlhkosti v procentech je třeba senzor zkalibrovat. Zjištění načtené hodnoty za sucha a maximální načtené hodnoty při ponoření do vody pro každý ze senzorů. Převod na procenta je řešen pomocí funkce `map(value, fromLow, fromHigh, toLow, toHigh)`

```
map(analogRead(SOIL_GLOBAL_HUMIDITY_PIN), SOIL_GLOB
AL_HUMIDITY_MIN, SOIL_GLOBAL_HUMIDITY_MAX, 0, 100);
```

4.3.2 Pohyb motorů

Ovládání pohybu je možné dosáhnout skrze střídání vysokého a nízkého signálu na pinu ke kterému je připojen pin řadiče motoru s označením STEP. Směr otáčky se řídí pomocí signálu na pinu s názvem DIR. (57)



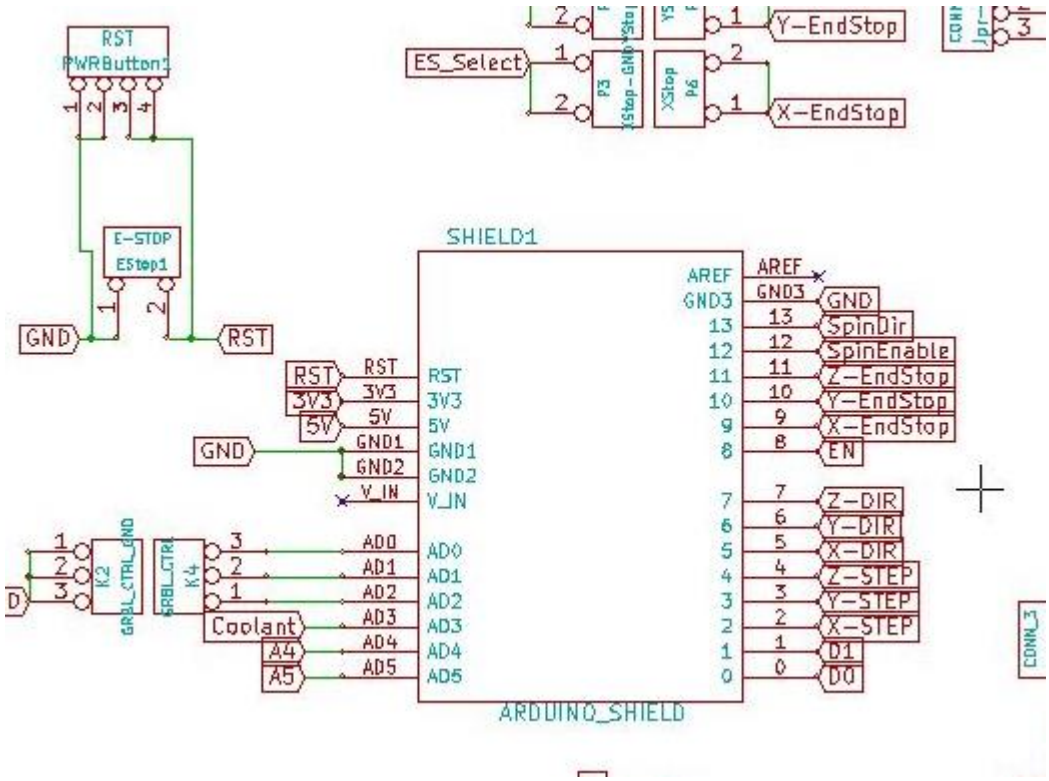
Obrázek 32 Zapojení A4988 (58)

Pro ovládání motorů byla zvolena knihovna AccelStepper, která umožňuje kontrolovat různé druhy krokových motorů. Nabízí objektově orientovanou kontrolu motorů. Pro každý motor lze vytvořit instanci třídy AccelStepper a tu pak kontrolovat. AccelStepper rozšiřuje standardní knihovnu Arduino Stepper například podporou nastavení akcelerace, zpomalení a umožnění ovládat více motorů současně. (59)

Inicializace motorů pomocí knihovny AccelStepper vypadá následovně:

```
//Stepper motors
AccelStepper stepperX(1, 2, 5);
AccelStepper stepperY(1, 3, 6);
AccelStepper stepperZ(1, 4, 7);
```

První parametr určuje způsob, jakým jsou řízeny motory. Číslo 1 odpovídá použití řadiče. Druhý parametr určuje, který pin je propojen s pinem STEP pro ovládání pohybu. Třetí poslední parametr určuje pin, který je propojen s pinem DIR pro ovládání směru.
V případě použití CNC Shieldu, lze tyto piny vyčíst z jeho dokumentace.



Obrázek 33 Rozložení pinů CNC Shield v3 (60)

K efektivnímu ovládání pohybu potřebujeme vědět, kde se aktuálně nacházíme na osách X, Y, Z. K tomu slouží funkce homeSteppers, která přesune nosič do výchozí pozice X0, Y0 a Z0. Všechen pohyb je pak prováděn relativně k této pozici.

```
void homeStepper(AccelStepper *stepper,
int endStopPin)
{
//Nastavení parametru motoru,
//Rychlost
stepper->setMaxSpeed(500);
//Akcelerace
stepper->setAcceleration(500);
//Počet kroků
stepper->move(-9999);

//Dokud není sepnut limitní spínač
//prováděj pohyb
while (!digitalRead(endStopPin))
{
stepper->run();
```

```
}
//Ve chvíli kdy je limitní spínač sepnut
//je nastavena tato pozice jako výchozí
//pozice 0
stepper->setCurrentPosition(0);
}
```

4.3.3 Knihovna ArduinoJSON

ArduinoJSON umožňuje rychle a efektivně parsovat řetězec znaků ve formátu JSON. Parsovaný řetězec uloží do dokumentu, se kterým lze poté pracovat jako s polem. Do dokumentu je možno rovnou ukládat příchozí data ze sériového vstupu. (61)

Mapování na proměnných pro zprávu může vypadat takto:

```
{"type":"movement","positionX":"500","positionY":"100",
"positionZ":"100"}
```

```
//Vytvoření dokumentu
DynamicJsonDocument doc(1024);

//Parsovani a uložení dat do dokumentu

deserializeJson(doc, Serial);

String type = doc[type];

Int positionX = doc[positionX];

Int positionY = doc[positionY];

Int positionZ = doc[positionZ];
```

Jednotlivé objekty v poli dokument je možné upravit a dokument opět serializovat a poslat skrze sériovou linku.

```
//Vymazání obsahu dokumentu

doc.clear()

//Vytvoření objektu type s hodnotou response

doc[type] = "response";

//serializace dokumentu a poslání skrze seriovou
//linku

serializeJson(serial, doc);
```

Odpověď Arduina po provedení výše zmíněného kódu pak vypadá takto:

```
{"type":"response"}
```

4.3.4 Rozhodování

Rozhodování a výsledná činnost je řešena pomocí podmínek ověřujících obsah v objektu „type“ v aktuální příchozí zprávě ze sériové linky. Toto se odehrává ve funkci loop(), kde je monitorován stav sériové linky. V případě nově příchozí zprávy je zpráva deserializována knihovnou ArduinoJSON. Poté je na základě předdefinovaných podmínek rozhodnuto, jaký kód bude vykonán. Po vykonání kódu Arduino pošle zpět zprávu do sériové linky s upravenou hodnotou objektu „type“ na hodnotu „response“. Což se dále interpretuje jako úspěšné dokončení požadavku.

```
loop(){
  //Wait for Serial input
  while (Serial2.available())
  {
    //save the input into message
    message = Serial2.readString();
    //message is ready to be parsed
    messageReady = true;
    //print it to serial for debuging purposes
    Serial.println(message);
  }
  if (messageReady)
  {
    DynamicJsonDocument doc(1024);
    DeserializationError error = deserializeJson(do
c, message);
    if (error)
    {
      Serial.print(F("deserializeJson() failed:"));
      Serial.println(error.c_str());
      messageReady = false;
      return;
    }

    if (doc["type"] == "movement")
    {
      stepperZ.moveTo(0);
      while (stepperZ.distanceToGo() != 0)
```

```
        {
          stepperZ.run();
        }

        int stepToGoX = doc["positionX"];
        int stepToGoY = doc["positionY"];
        int stepToGoZ = doc["positionZ"];

        stepperX.moveTo(stepToGoX);
        stepperY.moveTo(stepToGoY);
        stepperZ.moveTo(stepToGoZ);
        while (stepperX.distanceToGo() != 0 || steppe
rY.distanceToGo() != 0)
        {
          stepperX.run();
          stepperY.run();
        }
        while (stepperZ.distanceToGo() != 0)
        {
          stepperZ.run();
        }
 doc.clear();
      doc["type"] = "response";
      serializeJson(doc, Serial2);
     }
  }
}
```

Klíčové činnosti jako je třeba změření vlhkosti na určité pozici je řešeno kombinací zpráv. Nejdříve se pošle Arduinu zpráva s klíčovým slovem „movement“ a souřadnicemi. Po najetí na požadovanou pozici Arduino potvrdí dokončení. Poté se pošle zpráva typu „environment“ s hodnotou „request“ nastavenou na „soil_humidity_senzor“. Arduino po dokončení měření odpoví zprávou obsahující naměřené hodnoty.

```
if (doc["request"] == "soil_humidity_global")
      {
        doc.clear();
        doc["reading"] = analogRead(SOIL_GLOBAL_HUMIDITY_P
IN);
        doc["value"] = "soil_humidity";
        serializeJson(doc, Serial2);
      }
```

Odpověď Arduina s hodnotou ze senzoru pak vypadá následovně:

```
{
    "reading": "752",
    "value": "soil_humidity",
}
```

4.3.5 ESP8266

ESP je připojeno k Arduinu přes komunikační piny RX a TX. Na mikrokontroleru ESP8266 běží webový server, který zpracovává příchozí požadavky. K vytvoření webového serveru byla použita knihovna ESP8266 Web Server. Tělo přijatého požadavku přeposílá dále do Arduina a čeká na odpověď. Ve chvíli, kdy Arudino dokončí požadovaný úkon pošle odpověď modulu ESP a ten ho pak vrátí jako odpověď na požadavek hlavnímu serveru.

Aby mohl modul ESP přijímat http požadavky od hlavního serveru musí být připojený do stejné sítě. K připojení ESP modulu do sítě pomocí WiFi je použita knihovna ESP8266 WiFi. Pomocí této knihovny je možné se připojit na požadovanou WiFi síť. K tomu slouží metoda WiFi.begin('network-name', 'network-password'). Předávání zpráv z hlavního serveru je řešeno funcí handlePost().

```
void handlePost() {
  //clear the serial buffer
  serialFlush();

  if (server.hasArg("plain") == false) {
    return ;
  }
  //defining the body of the message
  String body = server.arg("plain");
  //deseriaze the body into a Json document
  deserializeJson(doc, body);
  //This part is here in case the document needs
editing
  //Serialize the document into serial
  serializeJson(doc, Serial);

  //Start counting time
  unsigned long currentTime = millis();

  //Wait for return message from arduino
  while (!Serial.available()) {
    //if there won't be a response in 50 seconds.
Close the connection
```

```
    if (millis() - currentTime > 50000) {
      server.send(408, "application/json", "");
    }
  }

  String returnMessage = "";
  returnMessage = Serial.readString();

  server.send(200, "application/json",
returnMessage);
}
```

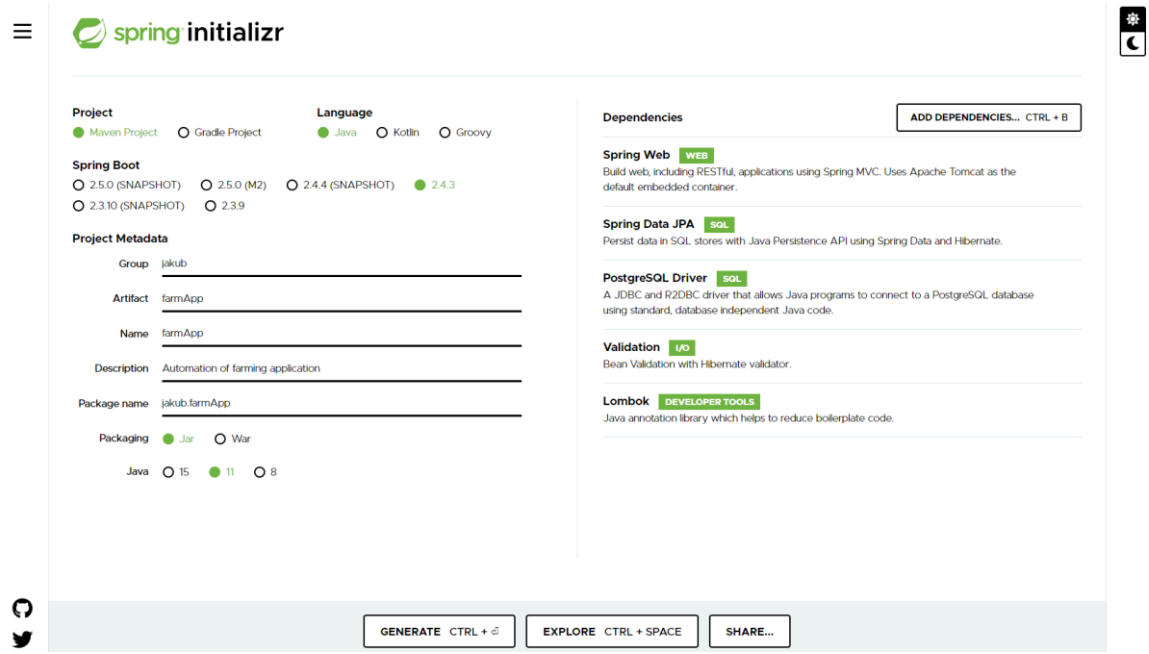
4.3.6 Webová aplikace

Backend

Cílem backendu je umožnit komplexní správu záhonu, a tak oddělit logiku od samotného Arduina. Vytvoření seznamu rostlin k zasazení, umožnit sazbu rostliny na určitou pozici, sbírat data ze senzorů a podle nich rozhodovat o úkonech. Komunikace s klientem je dosaženo skrze API endopointy.

Úkony lze rozdělit na obecné a specifické. Mezi obecné požadavky se řadí teplota vzduchu, vlhkost vzduchu, obecná vlhkost půdy. Obecné požadavky se netýkají přímo určité rostliny, týkají se všech rostlin. Mezi specifické požadavky se řadí zalévání rostliny a měření vlhkosti půdy rostlin. Specifického požadavku je dosaženo kombinací požadavků obecných.

Vytvoření projektu



Obrázek 34 Spring vytvoření projektu (Autor)

K vytvoření projektu byly vybrány závislosti Sping Web, Spring Data Jpa, Validation, PostgresSQL driver a Lombok.

Spring Web nám umožní vytvořit webovou aplikaci. Cílem Spring Data Jpa SQL je vytvořit abstrakci repositáře, a tak snížit nutnost opakujícího se kódu k implementaci datové vrstvy. Pomocí rozšíření repositáře interface CrudRepository<T,ID> je pak možné využívat metody findAll, findById, countByName atd.. bez nutnosti psát SQL dotazy. (62)

Validation I/O zjednodušuje validaci vstupů. Používá se skrze vkládání anotací přímo nad definici proměnné ve třídě. (63)

- @NotNull – hodnota nesmí být prázdná,
- @Min(18) – umisťuje se nad číselnou hodnotu. Hodnota musí být minimálně 18,
- @Max(18) - umisťuje se nad číselnou hodnotu. Musí být maximálně 18,
- @Size(min = 2, max = 30) umisťuje se nad řetězec znaků. Akceptuje minimální délku 2 a maximální 30.

PostgreSQL driver SQL je komponenta, která umožňuje aplikaci komunikovat s databází. (64) Lombok je plugin, který je možno implementovat do vývojového prostředí. Umožňuje zbavit se nutnosti psát opakující se kód. Například přístupové metody (getery, settery) pro jednotlivé proměnné tříd. Používá se skrze anotace nad třídou. (65)

Nejčastějšími anotace použité v projektu jsou:

`@AllArgsConstructor`

- vytvoří konstruktor se všemi argumenty

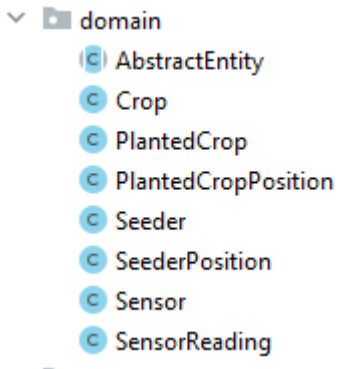
`@Data`

- vytvoří gettery a settery pro všechny proměnné typu private

`@ NoArgsConstructor`

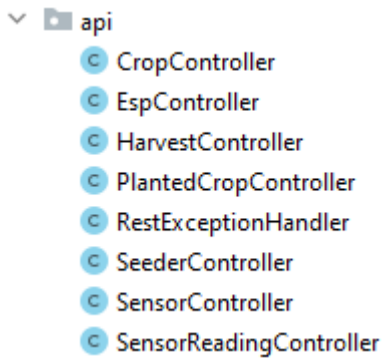
- vytvoří konstruktor bez argumentů

Pro ukládání dat byla zvolena databáze PostgreSQL alternativou k této databázi by mohla být databáze H2. Obě jsou to relační databáze, které splňují všechny požadavky nutné pro projekt. Zvolení databáze H2 by bylo vhodnější v případě, že bychom se rozhodli server provozovat na mikropočítačích. Například na Raspberry Pi.



Obrázek 35 Třídy tvořící datovou strukturu (Autor)

Rozhraní API



Obrázek 36 Controller (Autor)

Crop controller

`Get /crops` vrací kolekci všech instancí třídy Crop.

`Post /crops` vytvoří instanci plodiny a uloží ji mezi plodiny.

`Put /crops` najde plodinu podle názvu a upraví ji podle požadavku.

`Deltete /crops/{cropName}` najde plodinu podle názvu a vymaže ji.

PlantedCropController

`Get /plantedCrops` vrací kolekci všech instancí třídy PlantedCrops.

`Post /plantedCrops` vytvoří instanci třídy PlantedCrops a uloží ji mezi zasazené plodiny.

`Get /plantedCrops/humidity/{id}` vrátí kolekci naměřených hodnot vlhkosti na základě id zasazené rostliny.

`Delete /plantedCrops/{cropId}` najde instanci zasazené plodiny podle id a vymaže ji.

`Post /plantedCrops/harvest` vytvoří instanci třídy PlantedCrops a uloží ji mezi zasazené plodiny pod současnou sklizeň.

Get /plantedCrops/harvest/{harvestId} vrací kolekci instancí třídy PlantedCrops, které spadají pod danou sklizeň na základě id.

SeederControler

`Get /seeders` vrací kolekci všech seederů.

`Post /seeders` vytvoří instanci Seederu a uloží ho.

`Delete /seeders{seederName}` najde instanci Seederu podle názvu a vymaže ji.

SensorControler

`Get /allSensors` vrací kolekci všech senzorů.

`Get /sensor/{sensorName}` vrací pouze jeden senzor podle názvu.

`Post /newSensor` vytvoří nový senzor, pokud neexistuje. Existenci ověřuje na základě názvu senzoru.

`Post /newData` najde senzor podle názvu a do jeho seznamu sensorReadings uloží novou instanci sensorReading.

`Delete /Sensors/{sensorName}` najde instanci senzoru podle názvu a vymaže ho.

EspController

`Get /espHome` pošle Arduinu požadavek pro přesun nosiče do výchozí pozice X0, Y0, Z0.

`Get /espMovement` pošle Arduinu požadavek pro přesun na požadovanou pozici.

`Get /espPlant` pošle Arduinu požadavek o zasazení všech instancí PlantedCrops z aktuální sklizně, které ještě nebyli zasazeny a vrátí seznam všech zasazených rostlin.

`Get /chckSoilHumidtyAll` pošle Arduinu požadavek o změření vlhkosti na všech pozicích kde jsou zasazené rostliny z aktuální sklizně a vrací seznam všech zasazených rostlin.

`Get /waterLowHumidityCrops` pošle Arduinu požadavek o zavlažení zasazených rostlin z aktuální sklizně u kterých je poslední naměřená hodnota vlhkosti půdy nižší než ideální vlhkost půdy dané rostliny.

`Post /getReading/{sensorName}` pošle Arduinu požadavek o získání dat ze senzoru podle jména. Odpověď Arduina uloží mezi seznam naměřených hodnot vyžádaného senzoru.

`Post /getAllSensorReadingsSave` pošle Arduinu požadavek o získání dat ze všech senzorů. Pokud požadovaný senzor neexistuje v databázi vytvoří jeho instanci a přiřadí ji nové čtení mezi seznam naměřených hodnot a vše uloží.

HarvestController

`Get /getAllHarvests` vrací kolekci všech sklizní.

`Get /harvest/{id}` vrací pouze jednu sklizeň podle id.

`Post /harvests` vytvoří instanci sklizně a uloží ji.

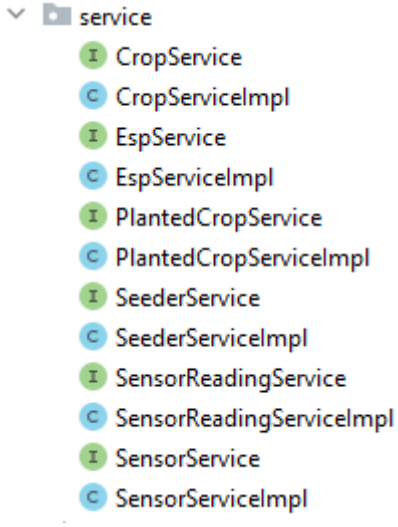
Implementované koncové body umožňují sběr dat a následné poskytnutí dat. Splňují tak požadavek na sběr dat. Třída SensorController umožňuje zpracovávat nejen data která přicházejí z Arduina. Vytvořit nový senzor, uložit ho a zapsat do jeho seznamu čtení, je možné pouhým požadavkem typu post na endopoint /newSensor. Tento endopointy slouží pro prvotní vytvoření senzoru. K senzorům jsou pak přiřazována data na základě jejich názvu.

Není možné mít dva senzory se stejným názvem. Pro uložení nového čtení ze senzoru slouží endpoint /newData. Tento způsob umožňuje připojit i jiná chytrá zařízení, která umí snímat prostředí a vizualizovat jejich data.

Zpráva ve formátu JSON pro uložení dat senzoru s názvem „humidity5“:

```
{
    "sensorName": "humidty5",
    "sensorReadings": [
                        {
                            "reading": "30",
                            "value": "%"
                        }
                        ]
}
```

Servisní třída



Obrázek 37 Servisní třídy (Autor)

Třída EspService má na starost převážně komunikaci mezi serverem a Arduinem. Implementuje následující metody:

```
public interface EspService {
    void moveToPosition(ArduinoMovementRequest
arduinoMovementRequest);
    void home();
    void plantCrop(PlantedCrop cropToPlant);
    List<PlantedCropResponseDto> plantAllCrops();
    void checkSoilHumidity(PlantedCrop
cropToCheck);
    List<PlantedCrop> checkSoilHumidityAll();
```

```
    void waterPlant(PlantedCrop plantToWater);
    void waterPlants();
    SensorReadingResponseDto
getSensorReading(String sensorName);
    void waterLowHumidityPlants();
    String getAllSensorReadings();
    void getAllSensorReadingsSave();
}
```

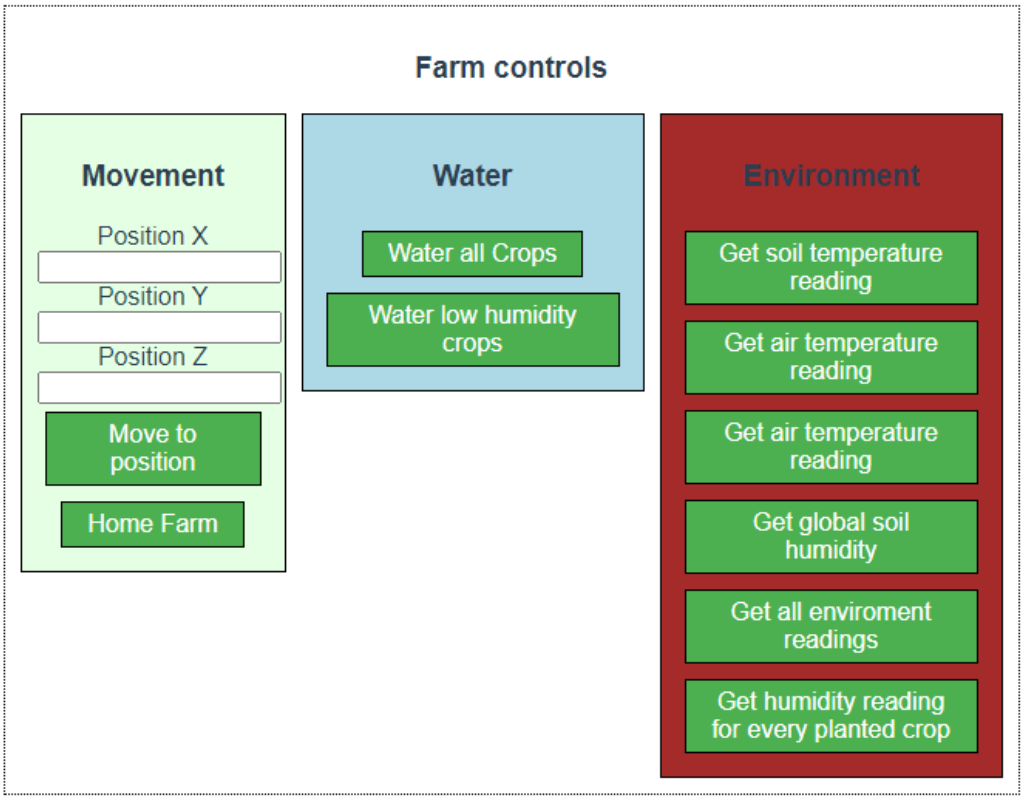
Automatizace měření vlhkosti a následného zavlažování rostlin je řešeno pomocí anotace `@Scheduled` s parametr `fixedDelay` nad metodou automate ve třídě AutomationService. Tato třída má v sobě implementaci EspService. Následující kód provede dvakrát denně získání dat ze senzorů a změří vlhkost půdy u jednotlivých plodin a podle přečtených hodnot je zavlaží.

```
@Service
@RequiredArgsConstructor
public class AutomationService {
private final EspService espService;

@Scheduled(fixedDelay = 43000*1000)
private void automate(){
    espService.getAllSensorReadingsSave();
    espService.checkSoilHumidityAll();
    espService.waterLowHumidityPlants();
}
}
```

Tímto způsobem je možné vytvářet jednotlivé metody, které mohou obsahovat různé kombinace úkonů a plánovat je na libovolný čas nebo intervaly.

4.3.7 Frontend

Frontend webové aplikace je vytvořený za pomocí frameworku Vue.js. Rozdělen je do tří sekcí. První sekcí je přehled záhonu pod záložkou Farm Overview, dalšími jsou Farm Control a Seeder. Sekce Farm Overview poskytuje kontrolu nad záhonem a skládá se z komponentů WaterControl, EnviromentControl, PlantedCropsTable, SensorDataChart, PlantedCropHumidityChart. Jednotlivé sekce webové aplikace využívají vytvořeného rozhraní API z kapitoly backend. Komponenty reagují dynamicky, lze je kliknutím zavírat či otevírat a zobrazit tak jen zrovna ty, které chceme využívat.



Obrázek 38 Komponenty k ovládání záhonu (Autor)

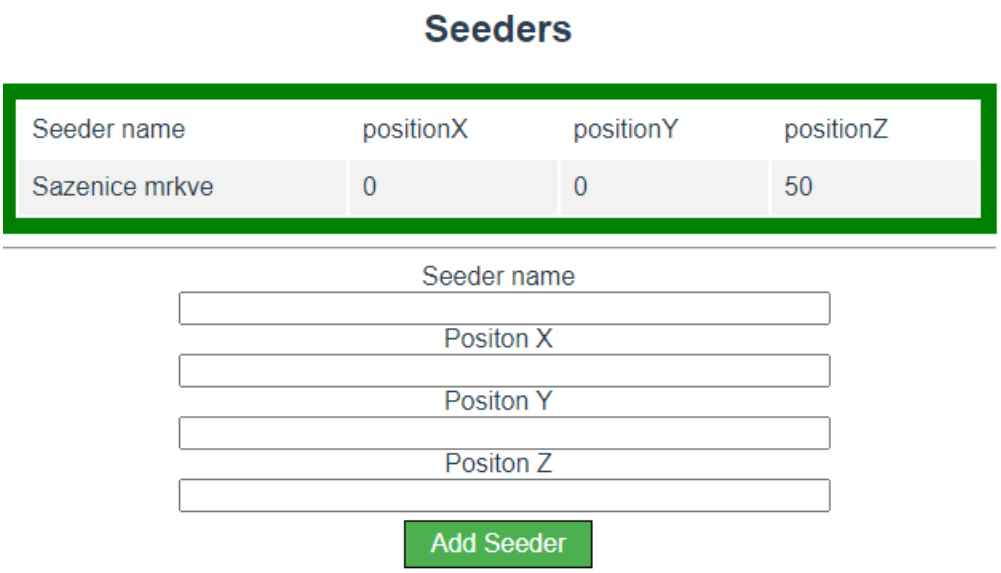
Ovládací panel pro pohyb nabízí možnost přesunout nosič na libovolnou pozici nebo se vrátit zpět do výchozí pozice. Panel Water umožňuje vynutit zalití všech rostlin anebo všech rostlin, u kterých je poslední naměřená hodnota vlhkosti nižší než jejich ideální. Panel Environment umožňuje vynutit získání nových dat ze senzorů v jinou než plánovanou dobu měření.

Obrázek 39 Přehled zasazených rostlin (Autor)

V pravé dolní části stránky Farm Overview se nachází přehled zasazených rostlin, pro každou novou zasazenou rostlinu je automaticky generován graf hodnot naměřené vlhkosti půdy na pozici, kde je rostlina zasazená. V levé dolní části stránky se nachází sada automaticky generovaných grafů, které se týkají senzorů připojených k záhonu. Pro každý nový senzor připojený k systému je automaticky vygenerován graf s jeho názvem a naměřenými hodnotami. K tvorbě grafů je použita knihovna Charts.js spolu s wrapprem vue-chartjs. Následující kód generuje grafy vlhkosti půdy pro jednotlivé zasazené rostliny:

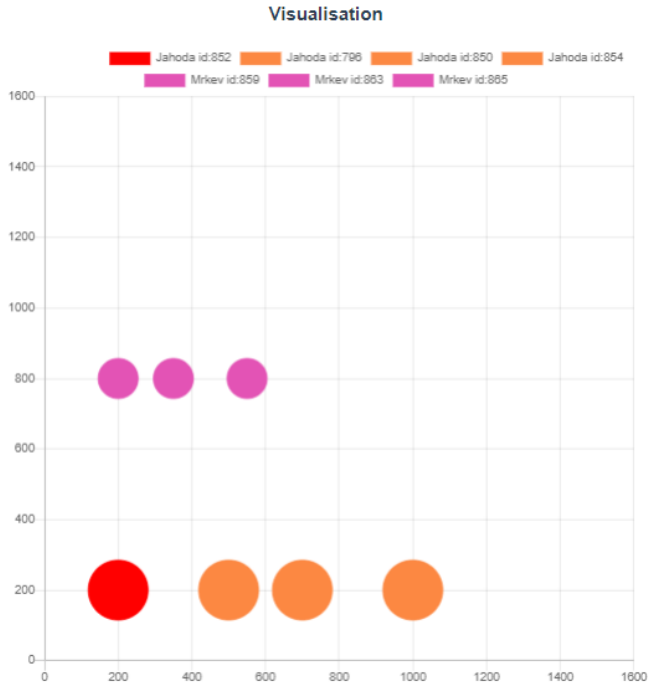
```
<template>
  <div>
    <h3 @click="hidden = !hidden" v-
if="hidden">Hide {{ heading }}</h3>
    <h3 @click="hidden = !hidden" v-
if="!hidden">Show {{ heading }}</h3>
    <div v-if="hidden">
      <div v-if="datacollection.length > 0">
        <ul>
          <li
            v-
for="(plantedCrop, index) in plantedCrops"
            :key="index"
            @click="
              datacollection[index].toggle = !datac
ollection[index].toggle
            "
          >
            <h2>{{ plantedCrop.crop.name}} id: {{pl
antedCrop.id}}</h2>
            <line-chart
              :options="options"
              v-if="datacollection[index].toggle"
              :chartData="datacollection[index]"
            ></line-chart>
          </li>
        </ul>
      </div>
      <div v-else>
        <h3>Loading data... or no data present</h3>
      </div>
    </div>
  </div>
</template>

<script>
import LineChart from "../components/LineChart.vue"
;
var url = "http://localhost:8081/plantedCrops";

export default {
  components: {
    LineChart,
  },

  data() {
    return {
      datacollection: [],
      tempReadings: [],
      tempLabels: [],
```

```
      hidden: true,
      loaded: false,
    };
  },
  mounted() {
    this.getData();
    setTimeout(() => this.prepareData(), 1000);
  },
  methods: {
    getData() {
      fetch(url)
        .then((response) => response.json()) // one
 extra step
        .then((data) => {
          this.plantedCrops = data;
        })
        .catch((error) => console.error(error));
    },

    prepareData() {
      this.plantedCrops.forEach((p) => {
        p.soilHumidityReadings.forEach((r) => {
          this.tempReadings.push(r.reading);
          this.tempLabels.push(r.createdAt);
        });
        this.datacollection.push({
          labels: this.tempLabels,
          datasets: [
            {
              label: p.crop.name + p.id,
              backgroundColor:
                "#" + Math.floor(Math.random() * 16
777216).toString(16),
              data: this.tempReadings,
            },
          ],
          toggle: true,
        });
        this.tempLabels = [];
        this.tempReadings = [];
      });
      this.loaded = true;
    },
  },
};
</script>
```



Obrázek 40 Definování pozic sazenic (Autor)

Sekce Seeder umožňuje definovat pozici se sazenicemi a uložit ji. Sekce Organise Farm nabízí možnost přidávat rostliny a z těch pak vybírat na jakou pozici budou zasazeny. Ve vrchní části stránky se nachízí vizualizace zasazených rostlin pomocí bublinového grafu. V grafu jsou zobrazeny jednotlivé rostliny ze seznamu zasazených rostlin. Pokud má bublina reprezentující rostlinu červenou barvu znamená to, že ještě nebyla zasazena a je ji možno zasadit pomocí tlačítka „Plant crops in queue“. Barva bubliny reprezetuje druh plodiny. To znamená, že bubliny reprezetnující zasazené plodiny stejného druhu mají stejnou barvu.



Obrázek 41 Vizualizace pozic zasazených rostlin (Autor)

Obrázek 42 Správa rostlin a jejich sazba (Autor)

5 Výsledky a diskuse

5.1 Ekonomické zhodnocení

Vytvořený software byl testovaný na mnou sestaveném prototypu. Záměrně jsem zvolil jednotlivé komponenty, tak aby byly cenově dostupné a běžně používané. Větší část ceny se dále odvíjí od velikosti realizovaného záhonu. Náklady jsou rozděleny do dvou sekcí. Fixní a variabilní. Fixní náklady se nemění v závislosti na velikosti záhonu. Tyto náklady se skládají převážně z komponent nutných pro provoz softwaru. Variabilní náklady jsou pak ty, na které má vliv velikost záhonu. Všechny níže uvedené ceny jsou s DPH 21 %.

Fixní náklady:

1x Klon Arduino mega	348 kč
1x IoT ESP8266 WIFI modul	148 Kč
3x Krokový motor Nema17	907 Kč
1x CNC Shield V3 3D	80 Kč
3x Řadič krokových motorů A4988	144 Kč
1x Jednokanálový modul 5 V relé	38 Kč
1x Kapacitní senzor vlhkosti	48 Kč
1x Odporový senzor vlhkosti	38 Kč
1x Vodní čerpadlo 2L/min	128 Kč
3D tisk ramene	200 Kč
3D tisk mechanismu osy Z	250 Kč
1x Hliníkový pojezd pro osu X	500 Kč
2x Hliníkový pojezd pro osu Y	1000 Kč
Celkem	3 829 Kč

Variabilní náklady:

Hliníkové profily pro rám	400 Kč za 1 m
Hadička průhledná 4 x 6 mm	12 Kč na 1 m^2 záhonu
Řemen GT2 se skelným vláknem	92 Kč za 1 m
Propojovací vodiče	60 Kč na 1 m^2 záhonu

V případě záhonu o rozloze 3 m2 by variabilní náklady činili 2184 Kč. Přibližné celkové náklady na uvedení záhonu o velikosti 3 m2 do provozu činí 5449 Kč. Do výsledné ceny nebyl započten hosting webové aplikace z hlediska možnosti využití hostingových služeb, které jsou zdarma anebo provozu serveru na vlastním zařízení.

V tuto chvíli se na českém trhu nenachází žádné komerční řešení chytrých záhonů, které by využívalo CNC manipulátorů.

5.2 Budoucí práce

V tuto chvíli systém závlahy rostlin zavlažuje rostlinu po přednastavený čas. Lepším řešením by bylo vytvoření systému závlahy, který by umožňoval zavlažit každou rostlinu tak, aby půda, která ji obklopuje dosáhla její ideální vlhkosti.

Měření vlhkosti u jednotlivých rostlin funguje správně do doby, než se rostlina rozroste do větších rozměrů, a tak zasahuje do místa kde pohyblivé rameno měří. Tímto může dojít k poškození rostliny. Tento problém bude vyřešen odsazením měřené pozice na základě stáří rostliny.

Velký potenciál chytrého záhonu nyní omezuje nemožnost výměny nástrojů. Rameno chytrého záhonu je osazeno pouze hadičkou k přívodu vody a senzorem pro měření vlhkosti. Na záhonu by se mohly nacházet různé odnímatelné nástroje, které by bylo možné měnit, a tak rozšířit stávající úkony o další. Mezi ně by mohlo patřit zasazení samotné rostliny nebo likvidace plevele pomocí těchto nástrojů.

Dalším rozšířením, které by umožnilo vyšší využitelnost chytrého záhonu je rozšíření o kameru a tím umožnit monitorování záhonu a rozhodování se na základě vizuálních podnětů za využití AI. Kamera by mohla být využita k detekci plevele a pohyblivé rameno k jeho následné likvidaci. Největším problémem a nejnákladnější částí je konstrukce záhonu, která by si spolu s návrhem jednotlivých nástrojů zasloužila svůj vlastní vývoj.

6 Závěr

Cílem diplomové práce bylo vytvořit software, který by umožňoval kontrolu chytrého záhonu. Cíl byl rozdělen do menších dílčích cílů, které splnily podmínky zadané v analýze požadavků. K dosažení cíle jsem využil běžně dostupných IoT komponentů. V současné době, kdy jsou chytré domácnosti a automatizace čím dál polárnější je jen otázkou času, než za nás bude pěstovat rostliny na zahradě stroj. Záměrně jsem zvolil jednotlivé komponenty, tak aby byly cenově dostupné a běžně využívané. Použil jsem Arduino, teplotní a vlhkostní snímač, vodní pumpu a pro pohyb po třech osách jsem využil krokových motorů. Dále jsem na konstrukci využil hliníkové extruze, které se také používají například ve 3D tiskárnách. Tento „chytrý záhon“ je možné kvůli své dostupnosti a přehlednému ovládání využít v domácnosti. Po dalších úpravách by ho bylo možné uplatnit i v halových zahradnictvích. Výsledné řešení odpovídá zadání definovanému v analýze požadavků. Software byl testovaný na mnou vytvořeném fyzickém prototypu. Systém byl tvořen s myšlenkou budoucích úprav a rozšíření.

Řešení projektu je rozděleno do čtyř vrstev. Vrstva Audino, vrstva ESP, backendu za pomoci frameworku Spring a frontend za pomoci frameworku vue.js. Jednotlivé vrstvy jsou od sebe oddělené a je možné je upravovat bez většího vlivu na ostatní vrstvy. To umožňuje rozšiřitelnost. Během vývoje se nenaskytly větší potíže. Zvolené technologie bych zhodnotil kladně.

Obrázek 43 Chytrý záhon (Autor)

Obrázek 44 Chytrý záhon (Autor)

7 Seznam použitých zdrojů

1. microcontroller (MCU). *IoT Agenda.* [Online] [Citace: 15. 2 2021.] https://internetofthingsagenda.techtarget.com/definition/microcontroller.

2. CO JE TO ARDUINO? *PRŮVODCE SVĚTEM ARDUINA.* [Online] [Citace: 24. 3 2021.] https://bastlirna.hwkitchen.cz/co-je-to-arduino/.

3. *Arduino: Uživatelská příručka.* Praha : Computer Press, 2017. 9788025148402.

4. Základní rozdíly čipů Espressif ESP8266 vs ESP32. *BLOG Jiří Vyorálek.* [Online] [Citace: 11. 3 2021.] https://blog.vyoralek.cz/iot/zakladni-rozdily-cipu-espressif-esp8266-vs-esp32/.

5. ESP8266 vs ESP32: What's the difference? *Wia Comunity.* [Online] [Citace: 11. 3 2021.] https://community.wia.io/d/53-esp8266-vs-esp32-what-s-the-difference.

6. *Wiring.* [Online] [Citace: 24. 3 2021.] http://wiring.org.co/.

7. setup(). *Arduino.cc.* [Online] [Citace: 24. 3 2021.] https://www.arduino.cc/reference/en/language/structure/sketch/setup/.

8. loop(). *Arduino.cc.* [Online] [Citace: 24. 3 2021.] https://www.arduino.cc/reference/en/language/structure/sketch/loop/.

9. Maven. *VoHo.eu.* [Online] http://voho.eu/wiki/maven/.

10. Spring and Open Source at the Pivotal Initiative. *Spring.io.* [Online] [Citace: 24. 3 2021.] https://spring.io/blog/2013/04/03/spring-and-open-source-at-the-pivotal-initiative/.

11. Úvod do Spring Boot frameworku pro Javu. *ITnetwork.cz.* [Online] [Citace: 11. 3 2021.] https://www.itnetwork.cz/java/spring-boot/uvod-do-spring-boot-frameworku-pro-javu.

12. Spring Boot Architecture and Workflow. *DZone.* [Online] [Citace: 24. 3 2021.] https://dzone.com/storage/temp/14265429-spring-boot-architecture.png.

13. Spring Boot Architecture. *Tutorial and example.* [Online] 12. 2 2020. [Citace: 11. 3 2021.] https://www.tutorialandexample.com/spring-boot-architecture/.

14. Hypertext Transfer Protocol -- HTTP/1.1. *Internet Engineering Task Force.* [Online] [Citace: 11. 3 2021.] https://tools.ietf.org/html/rfc2616.

15. Hyper Text Transfer Protocol. *Technická Univerzita Ostrava.* [Online] [Citace: 11. 3 2021.] http://www.cs.vsb.cz/grygarek/PS/kotasek/htp04.htm.

16. HTTP - Requests. *Tutorials point.* [Online] [Citace: 11. 3 2021.] https://www.tutorialspoint.com/http/http_requests.htm.

17. HTTP response status codes. *MDN Web Docs.* [Online] [Citace: 11. 3 2021.] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status.
18. JSON : jednotný formát pro výměnu dat. *Zdrojak.cz.* [Online] [Citace: 11. 3 2021.] https://zdrojak.cz/clanky/json-jednotny-format-pro-vymenu-dat/.
19. #1 Vue.js český návod zdarma. *StarkMedia .* [Online] [Citace: 24. 3 2021.] https://starkmedia.cz/blog/vue-js-zdarma-kurz-navod-cesky.
20. Vue.js český manuál a návody. [Online] [Citace: 24. 3 2021.] https://vue.baraja.cz/uvod-do-vue.
21. *Vue.js.* [Online] [Citace: 24. 03 2021.] https://vuejs.org/images/components.png.
22. Reactivity in Depth. *Vue.js.* [Online] [Citace: 24. 3 2021.] https://vuejs.org/v2/guide/reactivity.html.
23. Form Input Bindings. *vuejs.org.* [Online] [Citace: 24. 3 2021.] https://vuejs.org/v2/guide/forms.html.
24. List Rendering. *Vuejs.org.* [Online] [Citace: 24. 3 2021.] https://vuejs.org/v2/guide/list.html.
25. Conditional Rendering. *Vuejs.org.* [Online] [Citace: 24. 3 2021.] https://vuejs.org/v2/guide/conditional.html.
26. Samples. *chart.js.* [Online] [Citace: 27. 3 2021.] https://www.chartjs.org/.
27. Introduction. *vue-chartjs.* [Online] [Citace: 27. 3 2021.] https://vue-chartjs.org/guide/#introduction.
28. Why did we build Visual Studio Code? *Visual Studio Code.* [Online] [Citace: 11. 3 2021.] https://code.visualstudio.com/docs/editor/whyvscode.
29. Jet Brains. *IntelliJ IDEA.* [Online] [Citace: 11. 3 2021.] https://www.jetbrains.com/idea.
30. ScienceDirect. [Online] [Citace: 11. 3 2021.] https://www.sciencedirect.com/topics/computer-science/pin-connection.
31. Bipolar Stepper Motor. *ScienceDirect.* [Online] [Citace: 11. 3 2021.] https://www.sciencedirect.com/topics/engineering/bipolar-stepper-motor.
32. How to Drive a Stepper Motor. *DigKey.* [Online] 1. 10 2016. [Citace: 11. 3 21.] https://www.digikey.com/eewiki/display/Motley/How+to+Drive+a+Stepper+Motor.
33. Basics of Stepper Motors. *Oriental Motor.* [Online] [Citace: 11. 3 2021.] https://www.orientalmotor.com/stepper-motors/technology/stepper-motor-basics.html.

34. Stepper Motor System. *Orientalmotor.* [Online] [Citace: 24. 3 2021.] https://www.orientalmotor.com/images/stepper-motors/stepper-motor-wave-drive.jpg.
35. Stepper Motor System. *Orientalmotor.* [Online] [Citace: 24. 3 2021.] https://www.orientalmotor.com/images/stepper-motors/stepper-motor-2-phase-on.jpg.
36. What are stepper drives and how do they work? *Motion Control Tipis.* [Online] 29. 6 2015. [Citace: 11. 3 2021.] https://www.motioncontroltips.com/faq-what-are-stepper-drives-and-how-do-they-work/.
37. *Makerguides.com.* [Online] [Citace: 24. 3 2021.] https://www.makerguides.com/wp-content/uploads/2019/02/A4988-Arduino-stepper-motor-wiring-schematic-diagram-pinout-735x396.jpg.
38. CNC Shield V3. *electronicscomp.com.* [Online] [Citace: 24. 3 2021.] https://www.electronicscomp.com/cnc-shield-v3-3d-printer-a4988-expansion-board.
39. Soil Sensor. *Hardwario.* [Online] [Citace: 11. 3 2021.] https://www.hardwario.com/blog/2018-07-17-soil-sensor/.
40. Using Capacitive Soil Moisture Sensors on the Raspberry Pi. *SwitchDoc Labs.* [Online] [Citace: 24. 3 2021.] https://www.switchdoc.com/2020/06/tutorial-capacitive-moisture-sensor-grove/.
41. Is soil moisture sensor corrosion normal? *https://stackoverflow.com/.* [Online] [Citace: 24. 3 2021.] https://i.stack.imgur.com/J64vr.jpg.
42. LATEST OPEN TECH FROM SEEED STUDIO. *DHT11 vs DHT22* . [Online] [Citace: 11. 3 2021.] https://www.seeedstudio.com/blog/2020/04/20/dht11-vs-dht22-am2302-which-temperature-humidity-sensor-should-you-use/.
43. NTC Thermistor 10k. *Components 101.* [Online] [Citace: 11. 3 2021.] https://components101.com/resistors/ntc-thermistor-10k.
44. II. ZÁKLADNÍ ELEKTRICKÉ OBVODY. *Univerzita Palackého v Olomouci.* [Online] [Citace: 11. 3 2021.] https://ach.upol.cz/ucebnice2/obvody5.htm.
45. Dělič napětí zatížený a nezatížený. *mylms.cz.* [Online] [Citace: 24. 3 2021.] https://www.mylms.cz/wp-content/uploads/2012/01/3-1.png.
46. How Servo Motor Works & How To Control Servos using Arduino. *How To Mechatronics.* [Online] [Citace: 11. 3 2021.] https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/.
47. Endstops. *Marlin Firmware.* [Online] [Citace: 11. 3 2021.] https://marlinfw.org/docs/hardware/endstops.html.

48. SERIAL COMMUNICATIONS. *Electro noobs.* [Online] [Citace: 11. 3 2021.] http://electronoobs.com/eng_circuitos_tut36.php.
49. SERIAL COMMUNICATIONS. *http://electronoobs.com/.* [Online] [Citace: 24. 3 2021.] http://electronoobs.com/images/Circuitos/tut_36/serial_4.jpg.
50. Lineární pohony. *rem-technik.cz.* [Online] [Citace: 15. 3 2021.] https://www.rem-technik.cz/files/fck_userfiles/image/Fotky/IAI/viceose-systemy-pouziti.png.
51. Using a Thermistor. *adafruit.com.* [Online] [Citace: 25. 3 2021.] https://learn.adafruit.com/thermistor/using-a-thermistor.
52. Arduino Soil Moisture Sensor Calibration. *GREENSENSE.* [Online] [Citace: 25. 3 2021.] https://greensense.github.io/Blog/2017/02/17/Arduino-Soil-Moisture-Sensor-Calibration/.
53. A4988 Stepper Motor Driver Carrier Board. *GEEETECH.* [Online] [Citace: 25. 3 2021.] https://www.geeetech.com/wiki/index.php/A4988_Stepper_Motor_Driver_Carrier_Board.
54. A4988 Stepper Motor Driver Carrier Board. *GEEETECH.* [Online] [Citace: 25. 3 2021.] https://www.geeetech.com/wiki/index.php/A4988_Stepper_Motor_Driver_Carrier_Board.
55. AccelStepper. *Airspayce.com.* [Online] [Citace: 25. 3 2021.] http://www.airspayce.com/mikem/arduino/AccelStepper/.
56. CNC Sheild schematics. *Protoneer.co.nz.* [Online] [Citace: 25. 3 2021.] https://blog.protoneer.co.nz/wp-content/uploads/2013/07/Arduino-CNC-Shield-Scematics-V3.XX_.jpg.
57. What is Arduino? *arduino.cc.* [Online] [Citace: 24. 3 2021.] https://www.arduino.cc/en/guide/introduction.
58. Working with Spring Data Repositories. *spring.io.* [Online] [Citace: 24. 3 2021.] https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories.
59. Validating Form Input. *spring.io.* [Online] [Citace: 24. 3 2021.] https://spring.io/guides/gs/validating-form-input/.
60. Set up a Spring Boot Application With PostgreSQL. *DZone.* [Online] [Citace: 24. 3 20201.] https://dzone.com/articles/bounty-spring-boot-and-postgresql-database.
61. Common Java Object Functionality with Project Lombok. *infoworld.com.* [Online] [Citace: 24. 3 2021.] https://www.infoworld.com/article/2073566/common-java-object-functionality-with-project-lombok.html.

62. What is Arduino? *Arduino.cc.* [Online] Arduino. https://www.arduino.cc/en/guide/introduction.
63. The Untold History of Arduino by Hernando Barragán. *The Untold History of Arduino.* [Online] [Citace: 7. 3 2021.] https://arduinohistory.github.io/.
64. Shields. *Arduino.* [Online] [Citace: 11. 3 2021.] https://www.arduino.cc/en/Main/arduinoShields.
65. Soil Moisture Sensor Complete Guide to Use Soil Moisture Sensor . *Arduino.cc.* [Online] [Citace: 11. 3 2021.] https://create.arduino.cc/projecthub/electropeak/complete-guide-to-use-soil-moisture-sensor-w-examples-756b1f.
66. MAKE AN ARDUINO TEMPERATURE SENSOR. *Circuit Basics.* [Online] https://www.circuitbasics.com/arduino-thermistor-temperature-sensor-tutorial/.
67. vscode. *github.* [Online] [Citace: 11. 3 2021.] https://github.com/Microsoft/vscode.
68. Úvod do architektury MVC. *Zdrojak.cz.* [Online] [Citace: 24. 3 2021.] https://zdrojak.cz/clanky/uvod-do-architektury-mvc/.
69. Sending Pulse. *ScienceDirect.com* . [Online] [Citace: 24. 3 2021.] https://ars.els-cdn.com/content/image/3-s2.0-B9780080999340000077-f07-192-9780080999340.jpg?_.
70. Sending Pulse. *ScienceDirect.com* . [Online] [Citace: 24. 3 2021.] https://ars.els-cdn.com/content/image/3-s2.0-B9780080999241000071-f07-160-9780080999241.jpg?_.
71. The best JSON library for embedded C++. *ArduinoJson.* [Online] [Citace: 25. 3 2021.] https://arduinojson.org/.
72. AccelStepper. *Arduino.cc.* [Online] [Citace: 25. 3 2021.] https://www.arduino.cc/reference/en/libraries/accelstepper/.

Seznam obrázků

Seznam tabulek