**СОФИЙСКА ПРОФЕСИОНАЛНА ГИМНАЗИЯ ПО ЕЛЕКТРОНИКА** **„ДЖОН АТАНАСОВ“**

**Д И П Л О М Е Н П Р О Е К Т**

**Тема “**Проектиране и изработка на смарт управление за поливна система**“**

**Практическа част: „**Изработка на смарт управление за поливна система**“**

Дипломант: Ивайло Христов Иванов 12в клас

Професия: 481030 „ Приложен програмист“

Специалност: 4810301 „Приложно програмиране“

Ръководител на дипломен проект: инж. Любица Димитрова

Дипломант:

*/подпис/*

Ръководител на Дипломен проект:

*/подпис/*

София

2022

# Съдържание

[Съдържание 2](#_Toc102116227)

[Списък на използваните термини и съкращения 5](#_Toc102116228)

[Увод 8](#_Toc102116229)

[Изложение 10](#_Toc102116230)

[Глава I. Анализ 10](#_Toc102116231)

[1.1 Първи сегмент 11](#_Toc102116232)

[1.2 Втори сегмент 12](#_Toc102116233)

[1.3 Трети сегмент 13](#_Toc102116234)

[1.4 Четвърти сегмент 14](#_Toc102116235)

[Предимства на смарт управлението за поливна система 15](#_Toc102116236)

[Глава II. Резултати от измервания и експерименти 16](#_Toc102116237)

[2.1 Експеримент 1 16](#_Toc102116238)

[2.2 Експеримент 2 18](#_Toc102116239)

[2.3 Експеримент 3 19](#_Toc102116240)

[Глава III. Избор на софтуерна архитектура 20](#_Toc102116241)

[3.1 Java 20](#_Toc102116242)

[3.2 IDE – IntelliJ 21](#_Toc102116243)

[3.3 Android Studio 23](#_Toc102116244)

[3.4 Arduino Софтуер 23](#_Toc102116245)

[3.5 База данни – PostgreSQL 24](#_Toc102116246)

[3.6 Git - GitHub 26](#_Toc102116247)

[3.7 Spring Framework 26](#_Toc102116248)

[3.8 Postman 27](#_Toc102116249)

[3.9 PuTTY 27](#_Toc102116250)

[3.10 FileZilla 27](#_Toc102116251)

[3.11 VNC Viewer 28](#_Toc102116252)

[3.12 Raspberry Pi Imager 28](#_Toc102116253)

[3.13 Tasmota 29](#_Toc102116254)

[3.14 ESP Easy 29](#_Toc102116255)

[Глава IV. Избор на хардуерни компоненти 30](#_Toc102116256)

[4.1 Raspberry Pi - Raspberry Pi Zero W 30](#_Toc102116257)

[4.2 Микроконтролер ESP8266 WeMos D1 Mini 31](#_Toc102116258)

[4.3 Sonoff Basic R2 DIY Smart Switch 31](#_Toc102116259)

[4.4 Електромагнитен вентил - Tork t-gp 104 32](#_Toc102116260)

[4.5 Сензор за температура и влажност на въздуха - DHT22 33](#_Toc102116261)

[4.6 Сензор за налягане - BMP280 33](#_Toc102116262)

[4.7 Сензор за влажност на почвата - Capacitive Soil Moisture Sensor v1.0 34](#_Toc102116263)

[4.8 FTDI - UART TTL FT232RL 34](#_Toc102116264)

[4.9 Литиево-йонни батерии - INR18650-MH1 35](#_Toc102116265)

[4.10 Модул за зареждане на литиева батерия - TP4056 35](#_Toc102116266)

[4.11 Соларен панел - LIGHTEX 5W 9V 36](#_Toc102116267)

[Глава V. Разработка на софтуер 37](#_Toc102116268)

[5.1 Back-end – Spring Framework 37](#_Toc102116269)

[5.2 Front-end – Android 44](#_Toc102116270)

[Глава VI. База данни 46](#_Toc102116271)

[6.1 Таблица 1 – “Configuration” 46](#_Toc102116272)

[6.2 Таблица 2 – “Valve” 47](#_Toc102116273)

[6.3 Таблица 3 – “Watering\_Hours” 47](#_Toc102116274)

[6.4 Таблица 4 – “Users” 48](#_Toc102116275)

[6.5 Таблица 5 – “Logs” 48](#_Toc102116276)

[6.6 Таблица 6 – “Devices” 49](#_Toc102116277)

[6.7 Таблица 7 – “Sensors” 49](#_Toc102116278)

[6.8 Таблица 8 – “Sensor\_Types” 50](#_Toc102116279)

[6.9 Таблица 9 – “Sensor\_Data” 50](#_Toc102116280)

[Глава VII. Разработка на хардуер 51](#_Toc102116281)

[7.1 Разработка на “Първи сегмент” 51](#_Toc102116282)

[7.2 Разработка на “Втори сегмент” 51](#_Toc102116283)

[7.3 Разработка на “Трети сегмент” 51](#_Toc102116284)

[7.4 Разработка на “Четвърти сегмент” 52](#_Toc102116285)

[Глава VIII. Стартиране на системата от свързани устройства 53](#_Toc102116286)

[8.1 Подготвяне на Raspberry Pi Zero W 53](#_Toc102116287)

[8.2 Инсталиране на Java и Maven 53](#_Toc102116288)

[8.3 Инсталиране на функционалния код от GitHub 54](#_Toc102116289)

[8.4 Подготовяне на микроконтролер ESP8266 WeMos D1 Mini 54](#_Toc102116290)

[8.5 Подготвяне на Sonoff Basic R2 DIY Smart Switch 55](#_Toc102116291)

[8.6 Инсталиране на Android приложение 55](#_Toc102116292)

[Заключение 56](#_Toc102116293)

[Практическа част 57](#_Toc102116294)

[Списък на използваните фигури и диаграми 58](#_Toc102116295)

[Използвана литература 61](#_Toc102116296)

[Печатни издания 61](#_Toc102116297)

[Интернет адреси 61](#_Toc102116298)

[Полезни адреси 62](#_Toc102116299)

# Списък на използваните термини и съкращения

1Back-end - Отнася се до секциите от кода, които му позволяват да се изпълнява, но не са видими за потребителя. Съхранява и има достъп до по-голямата част от данните и оперативния синтаксис. Един или повече езици за програмиране обикновено се използват в кода.

2Front-end - Включва целия софтуер или хардуер, който е част от потребителският интерфейс. Човешките или цифровите потребители взаимодействат директно с различни аспекти на предния край на програмата включително: въведени от потребителя данни, бутони, програми, уебсайтове и други функции.

3Wi-Fi (Wireless Fidelity) - Безжичната технология, която свързва компютри, таблети, смартфони и други устройства към интернет, е известна като Wi-Fi. Тя е радиосигнал, предаван от безжичен рутер до близко устройство, което преобразува сигнала в данни, които могат да се видят и използват. Устройството предава радиосигнал обратно към рутера, който се свѝѝързва с интернет по проводник или кабел.

4Demo - демонстрация на продукт или техника.

5Cron - В подобни на Unix операционни системи, помощната програма Cron, известна още като Cron job, е планировчик на задания. Това е инструмент, използван от потребители, които настройват и поддържат софтуерни среди, за да планират задания (команди или скриптове), които да се изпълняват в определени часове, дни или интервали. Въпреки че, общото му предназначение го прави удобен за задачи като изтегляне на файлове от интернет и имейл на редовни интервали, най-често се използва за автоматизиране на поддръжката или администрирането на система. Cron е най-подходящ за планиране на повторяеми задачи.

6GPIO - Съкращение за: вход/изход с общо предназначение. Това е често срещан метод за свързване на микроконтролери към други електронни устройства. Може да се използва със сензори, диоди, дисплеи и модули. Позволява предоставяне на външно захранване, дистанционно управление на свързани устройства, излъчване на повече данни и дефиниране на съдържание за персонализирани Bluetooth данни.

7IDE - Интегрирана среда за разработка

8GUI - Графичният потребителски интерфейси е стандарт за дизайн и е ориентиран към потребителя в програмирането на софтуерни приложения, предоставяйки възможността за интуитивна работа с компютри и други електронни устройства чрез директно манипулиране на графични икони като бутони, ленти за превъртане, прозорци, раздели, менюта, курсори, посочващото устройство на мишката и др. Много съвременни графични потребителски интерфейси включват сензорен екран и възможности за взаимодействие с гласови команди.

9Сериен монитор - Серийният монитор е връзката между компютър и Arduino. Той позволява да се изпращат и получават текстови съобщения, което е полезно за отстраняване на грешки и контрол с помощта на клавиатура.

10БД – База от данни.

11СУБД - Система за управление на бази данни.

12SQL – Structured Query Language или Език за структурирани запитвания, е популярен език за програмиране, предназначен за създаване, достъпване, видоизменяне, извличане, манипулиране и обработка на данни от системи за управление на бази данни.

13API - API е съкращение на приложно-програмен интерфейс (Application Programming Interface) и най-просто казано е набор от програмен код, който позволява предаване на данни между един софтуерен продукт и друг. Образно казано, предоставя един по-абстрактен и опростен план за разработчика на приложения, който би му спестил изучаването на няколко различни слоя от системи зад интерфейса. По този начин се достига ефективност и бързина при адаптирането на нови софтуерни технологии.

14SSH - Мрежови протокол, позволяващ криптирано предаване на данни. Най-често се използва за изпълняване на команди на отдалечена машина, прехвърляне на файлове от една машина на друга и самото ѝ менажиране. Предоставя високо ниво на автентикация и сигурност по време на комуникацията през незащитена връзка.

15Telnet - Мрежови протокол, който се използва в интернет или локална мрежа с цел да осигури двупосочно интерактивно текстово ориентирано комуникационно средство с помощта на виртуално терминално свързване.

16FTP - Протоколът за прехвърляне на файлове (FTP) е стандартизиран мрежови протокол, използван за прехвърляне на файлове между клиент и сървър през интернет мрежа. FTP е проектиран да насърчава споделянето на файлове на всички видове компютри. Неговата силна страна е надеждното и ефективно групово прехвърляне на файлове.

17LAN - Local Area Network или Локална мрежа е вид малка компютърна мрежа, обслужваща компютри и други устройства, свързани помежду си.

18RAM – Динамична памет с произволен (непосредствен) достъп.

19CRUD функции – Те обхващат всички основни функции, които трябва да са имплементирани в приложенията. Всяка една от буквите на акронима е свързана със SQL и означава съответното: "C" e "Insert", "R" e "Select", "U" e "Update" и "D" e "Delete". Съществуват вариации на този модела, но всяка една от тях включва тези четири основни функционалности под една или друга форма.

20Колоната за идентичност - Това е колоната, която е съставена от генерирани подред числа на създаване.

21Външен ключ - Външният ключ е колона, използвана в база от данни за свързване на данни между таблици.

# Увод

В нашето натоварено ежедневие, често се налага да сме далече от нашите градини, а животът на всяко растение зависи пряко от равномерното и правилно напояване. Друг проблем който човек изпитва е несигурността дали момента за поливане е удобен за насъжденията, поради някаква природна характеристика като: висока температура, влажност на почвата или въздуха, неочакван валеж или нещо друго. Именно поради тази причина изграждането на ефективна напоителна система има решаващо значение. Тук идва на помощ умното устройство което е насочено към подпомагане, улесняване, автоматизиране, и контрол над работата с поливна система независимо от нейното местоположение, начин на грижа и природни условия. По този начин се редуцира разхода на вода, спестявайки също време и пари. Дистанционният достъп позволява да разглеждаме, управляваме и контролираме програматора за напояване от телефона.

Проекта е изграден напълно чрез Java и е насочен към концепцията за “IoT” (Internet of Things – Интернет на Нещата), която набира голяма популярност, тъй като решенията в тази сфера улесняват много нашето ежедневие.

Селското стопанство е основен кандидат за тоталната “IoT” революция, поради земята и огромното количество реколта, която се нуждае от наблюдение и грижа. Интернет на нещата позволява не само да се вземат по-интелигентни решения, но и да се автоматизират много от процесите, извършвани ежедневно от човешка ръка. Нивата на влагата може да бъдат наблюдавани, за да се избегне преполиване на растенията, както и засушаване, а въз основа на данни за състоянието на почвата и температурата може да се намали рискът от загуба и да се подобри растежа им.

Други иновации и предимства, които тази сфера притежава са точните актуализации в реално време относно консумацията на енергия, събитията и системите за управление на мрежата. Ползите за наблюдение и оптимизация, които идват с внедряването на “IoT”, осигуряват повече ранни предупреждения, по-голям контрол върху разпределението и използването на енергия и подобрена ефективност, сравнение с такива които биват управлявани и наблюдавани само от хора.

Очаква се Интернет на нещата да продължи да расте и да се подобрява през следващите години. Така избора на избраната тема се превръща в голяма иновация, която има за цел да подпомага много хора които имат нужда от автоматична поддръжка на растения. Също така системата е рядко срещана и има голямо поле за развитие и подобряване.

“Смарт управлението за поливна система” е съвкупност от 4 сегмента, които са лесни за инсталация и преместване на нови места. Най-основните стъпките за поставяне на инсталацията са общо 4 на брой:

1. Включване на електромагнитен вентил и смарт реле към водоснабдителна, токова инсталация и мрежа.
2. Свързване на Raspberry Pi Zero W и неговите модули към мрежа.
3. Поставяне на кутията със соларен панел, устройствата за мерене на температура и влажност на въздуха, налягане и забиване на сензора за влажност на почвата на предназначеното място за поливане в близост до мрежа.
4. Сваляне и свързване на Android приложението за телефон към мрежа.

В следващите глави от документацията ще обясним и разгледаме:

1. Анализ – тук ще направим разбивка на основната същност на проекта, начина на работа, цели и неговите предимства.
2. Резултати и експерименти – разглеждаме всички експерименти които сме осъществили по време на разработката на проекта и какво сме научили от тях.
3. Избор на софтуерна архитектура – показваме кои са най-основните софтуерни продукти които са използвани за разработката на системата.
4. Избор на хардуерни компоненти – ще изброим всички използвани хардуерни компоненти.
5. Разработка на софтуер – е процеса на разработка и работа на 1Back-end-а и 2Front-end-а в нашата системата.
6. Разработка на База данни – е процеса на разработка и работа на нашата База от данни.
7. Разработка на Хардуерни компоненти – обясняваме процеса на разработка и работа на нашите хардуерни компоненти.
8. Стартиране на системата от свързани устройства – ще разгледаме основните стъпки за инициализиране на системата.
9. Използвана литература – тук ще разгледаме цялата документация, която сме използвали по време на осъществяване на документацията и проекта.

# Изложение

## Глава I. Анализ

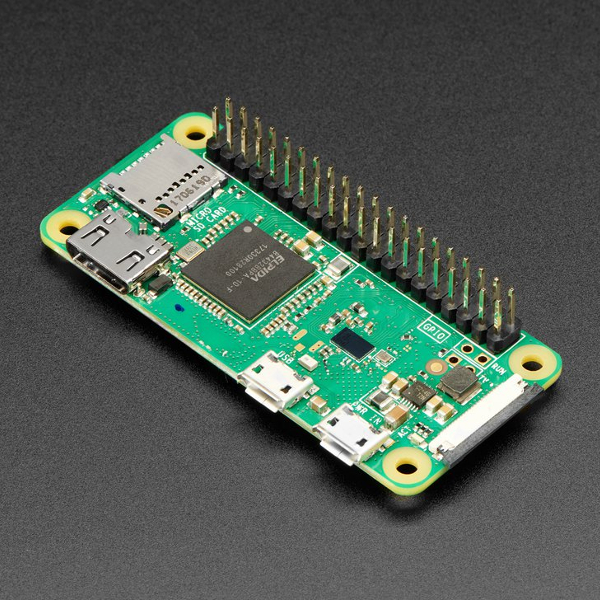
Фигура 1 – Диаграма на дипломната работа

Всеки желае за перфектна визия и продуктивност на своята градина, но при ръчните поливни системи може да се стигне до усложнения, неточности и забавяне. Целта на устройството за “Смарт управление на поливна система” е да улесни и подпомогне процеса на поддръжката на селско-стопански площи нуждаещи се от постоянна грижа, да намали консумацията на вода и енергия и да съкрати нуждата от човешка ръка.

За да може една автоматизирана напоителна система да заработи ефективно обаче, има много работа преди изграждането ѝ, за това е важно след инсталацията ѝ да се направят правилните настройки.

Проекта изпълнява своите финкции посредством множество сензорни проверки и данни зададени от клиента, но за да ги обясним, първоначално трябва да се запознаем с идеята като едно цяло.

### Първи сегмент

Проекта се разделя на 4 основно несвързани сегмента. Първият се състои от:

* Raspberry pi zero w.

Той е мозъкът на нашия проект и има за цел основно да контролира и управлява целия процес на работа. Сегмента трябва да се постави в близост до мрежа, по която чрез 3Wi-Fi да се свърже. Интересното тук е, че не е нужно да има интернет връзка която да преминава през системата. Проекта работи абсолютно офлайн и единственото което му е необходимо е захранване и данни въведени от потребителя за правилна и точна работа.

Фигура 2 – Raspberry Pi Zero W

Когато нашите модули биват поставени и включени към електрически поток, имаме нужда да свържем Raspberry pi към съответната мрежа, като въведем името и паролата на 3Wi-Fi канала, а след това да калибрираме стойност за работа по наш избор, при нужда. Когато инициализираме системата за първи път, ще има данни зададени по подразбиране, но ако трябва да ги променим по наше желание можем лесно да го направим през Android приложението. Стойностите са 7 на брой:

* Максимална възможна температура на въздуха за поливане,
* Максимална възможна влажност на въздуха за поливане,
* Максимално възможно налягане за поливане,
* Максимална възможна влажност на почвата за поливане,
* График на поливане (Брой на поливания на ден),
* Времетраене на поливането,
* Час и минути на поливането.

След като данните са въведени и всички останали сегменти са поставени, когато системата се инициализира, тя извършва проверки дали получените от сензорите стойности отговарят на зададените от потребителя или не такива. Ако са в границите, започва автоматично поливане, ако не отговарят, се изчаква 30 минути и отново се проверява докато получената информация не е валидна. Ако данните не са в границите до следващото поливане или 2 часа след първия опит за такова, програмата автоматично се изключва.

### Втори сегмент

Graphical user interface, application

Description automatically generatedDiagram, application

Description automatically generatedВторият сегмент е предната част на нашата система и единствената връзка на клиента с нея. Това е Android приложение, изградено чрез Android Studio, което има за роля да приема нуждите на клиента свързани с начина и времето на поливане и да ги обновява в базата данни. Също така може да показва появили се грешки в кода по време на работа на програмата.

Фигура 3 – Главна страница на Android приложение

Фигура 4 – Страница за проверка на връзката – Android приложение

Приложението е с изчистен и лесно достъпен дизайн. Единственото нещо което е нужно, за да се използва е:

* Да се свали от интернет,
* Телефонът да е свързан към същата мрежа, към която е и “Първият сегмент”.

Чрез него може да се редактират и задават всичките 7 данни за работа на системата, да се инициализира или спира поливането ръчно при нужда от клиент, да се видят данните от сензорите динамично и да се проверява за грешки в кода по време на работа на устройството.

### Трети сегмент

Третата част, която играe много основна роля и е изградена от:

* Електромагнитен вентил (Solenoid valve на английски),
* Смарт 3Wi-Fi реле.

Електромагнитния вентил или просто крана се закача за водоснабдителната система и пропуска или спира притока на вода само когато му се подаде ток при нужда. Това се случва посредством смарт 3Wi-Fi реле, което получава сигнали по мрежа от “Първият сегмент”, когато системата е завършила своите проверки и трябва да стартира поливане. Аналогично приключва или се прекратява поливане.

Крана трябва да се закачи към тръба на водопровода през която да минава вода. След това се сваля софтуерът на релето, свързва се и се поставя в близост до електромагнитният вентил. Накрая трябва да се свърже към маршрутизатора и ток.

Този сегмент е гръбнакът или центърът на нашата система за “смарт управление на поливна система“. Чрез него ние може да контролираме и управляваме напояването и количеството вода което се използва в определено време от деня.

A picture containing indoor, white

Description automatically generatedA picture containing indoor, camera

Description automatically generated

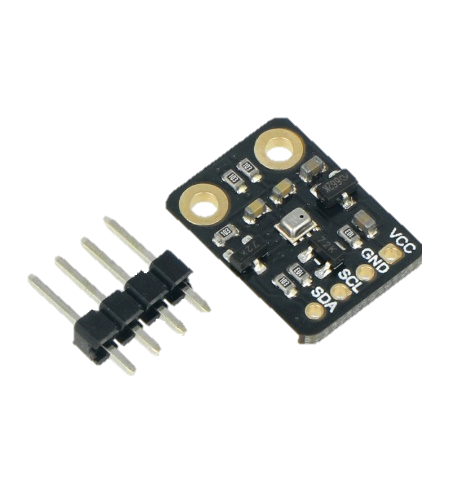
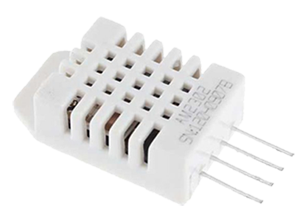
Фигура – Смарт 3Wi-Fi реле (Sonoff)

Фигура – Електромагнитен вентил

### Четвърти сегмент

Четвъртият сегмент е съставен от:

* Микроконтролер ESP8266 WeMos D1 Mini,
* Сeнзор за температура и влажност на въздуха (DHT22),
* Сензор за влажност на почвата (Capacitive Soil Moisture Sensor v1.0),
* Сензор за налягане (BMP280),
* Литиево-йонни батерии,
* Модул за зареждане на литиева батерия,
* Соларен панел.



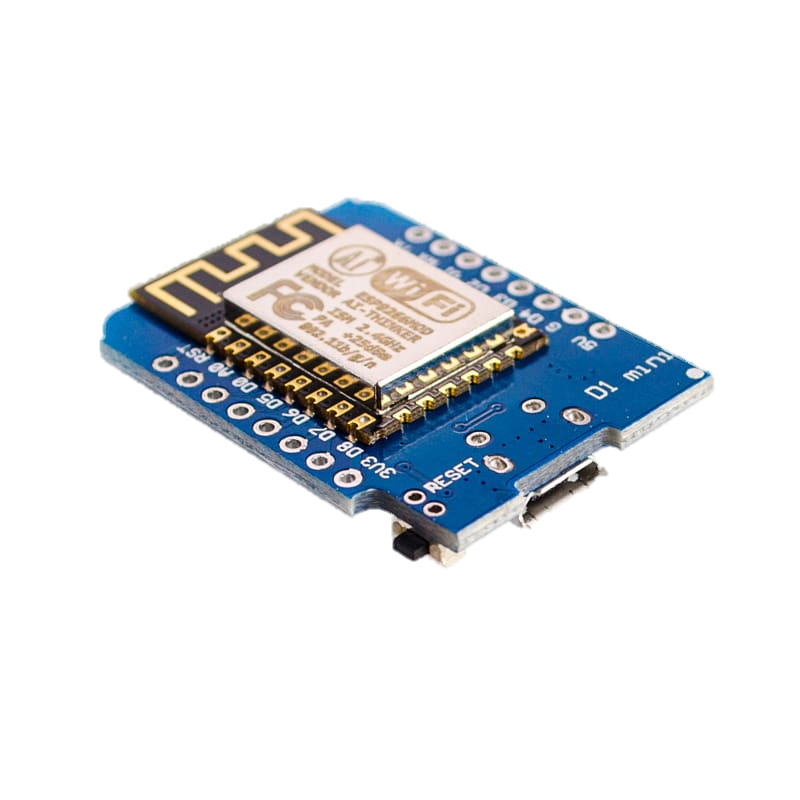
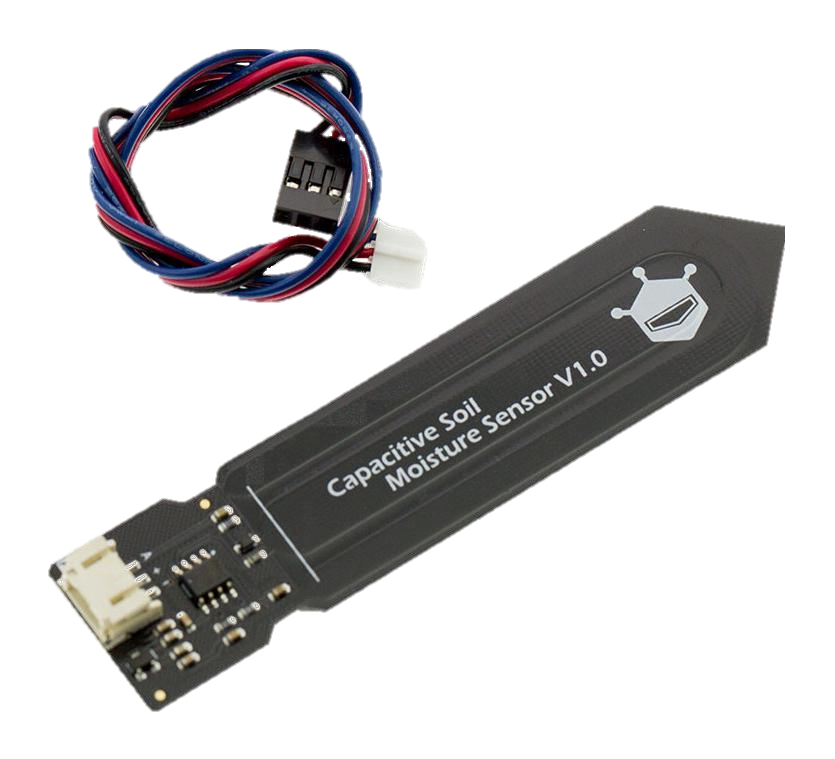
Фигура 7 – DHT22

Фигура 8 – Сензор BMP280

Целта на четвъртата част от нашия проект е да даде точна и коректна информация за влажността на почвата, температурата и влажността на въздуха и налягането в района за поливане, за да не се получава наводняване и изгаряне на растенията. Частите се поставят на място в близост до нашата мрежа.

Главните елементи тук са сензорите които изпращат данни към микроконтролера D1 mini. Неговата работа е основна за този сегмент, защото препраща получените стойности по 3Wi-Fi мрежата към “Първия сегмент” който ги обработва и използва.

Тъй като устройствата трябва да са поставени на мястото на поливане, където може да има липса на постоянно захранване, е възможно частите да бъдат поддържани от соларен панел, батерии и модул за зареждане на литиеви батерии.



Фигура – Микроконтролер D1 mini

Фигура – Сензор за влажност на почвата

### Предимства на смарт управлението за поливна система

След като разгледахме цялостната работа и всичките сегменти на системата можем

да разгледаме и кои са нейните основни предимства:

1. При наличие на смарт управление за поливна система няма да се налага ежедневно да се осигурява човек, който да полива. Чрез поливната система икономисваме време и пари.
2. Ако данните от сензорите за влажност на въздуха и почвата са твърде високи програмата може автоматично да пропусне напояване, като по този начин се редуцира и разхода на вода.
3. Градината може да се полива дори и в най-неудобните часове като рано сутрин или късно вечер, което я прави по-прецизна, с точност до минута.
4. Широката гама от сензори позволява правилен, точен и автоматичен контрол върху напояването на градината, като е и усъвършенствана система за следене и получаване на информация от околната среда.
5. Устройството има датчик за налягане, който автоматично не инициализира или спира поливането, ако има опасност от валеж.
6. Чрез Android приложението се настройва времето, за което ще се полива дадена площ. В зависимост от сезона и изискванията на растителността времето за поливане може да се промени.

## Глава II. Резултати от измервания и експерименти

### Експеримент 1

Първият експеримент който направихме е 4demo версия на цялостния проект.

Той се състоеше от:

* Raspberry Pi Zero W,
* Сензор за температура и влажност – DHT11,
* Led лампичка,
* Резистор,
* Пиезоелектричен активен сигнален звънец (Buzzer).

A picture containing toilet, indoor

Description automatically generatedКогато този проект за първи път започна да се изгражда на Python, неговата идея беше да се интегрира като основен и да работи като главна логика която да управлява начина, часа и времетраенето на поливането и записването на грешки появили се по време на работа, но срещнахме много спънки и проблеми по пътя, които биха направили програмата много несигурна и податлива на грешки. Също така по този начин на изграждане би имало много малко и трудни начини за подобрение, развиване и поддържане.

Фигура - Разработка на "Експеримент 1"

Начина на постигане на идеята беше изцяло изграден върху Raspberry Pi на Python и съдържаше съвкупност от 3 файла:

* Python файл,
* Текстов файл,
* 5Cron файл (Cron tab или Cron job).

Text

Description automatically generatedPython файла съдържаше бизнес логиката за работа на програмата, текстовия файл имаше роля на място за съхранение на грешки и съобщения изпращан от системата, а 5Cron за планиране на поливане.

Фигура – 5Cron файл (Експеримент 1)

Логиката на програмата беше кратка и проста, тя работеше главно на основата за

Text

Description automatically generatedText

Description automatically generated6GPIO и активираше сензорите, Led лампичката, сигналния звънец и крана за поливане като подаваше ток към тях, когато часът съвпадне с този от 5Cron файла.

Фигура – Първа част от код на експеримент 1

Фигура – Втора част от код на експеримент 1

### Експеримент 2

Този експеримент беше проведен чрез:

* Сензор за влажност на почвата - Capacitive Soil Moisture Sensor v1.0,
* Микроконтролер ESP8266 WeMos D1 Mini,
* Чаша с вода.

Опита се проведе, за да се измерят максималните и минималните стойности, които изпраща сензора за влажност на почвата, за да може да се направят нужните изчисления и да се открият оптималните стойности на които той работи, а накрая и да се зададат като данни по подразбиране (“Първи сегмент”).

Експеримента се изпълни като на първо място сензора за влажност на почвата и ESP8266 WeMos D1 Mini се свържат чрез джъмпер кабели по между си. След като тази стъпка се изпълни, микроконтролера се скача към компютър и се добавя код, който чете получените стойности и ги показва на екран. Впоследствие сензора постепенно се потапя в чаша с вода, като се следят стойностите които изпраща и се записват. Накрая се отбелязват най-ниската и най-високата получена стойност и им се намира средно-аритметичното.

Резултат от измерването:

* Минимални стойности – 404
* Максимални стойност – 784
* A picture containing outdoor, grass, tree, person

  Description automatically generatedСредно-аритметично – 594

Фигура 15 – Експеримент 2

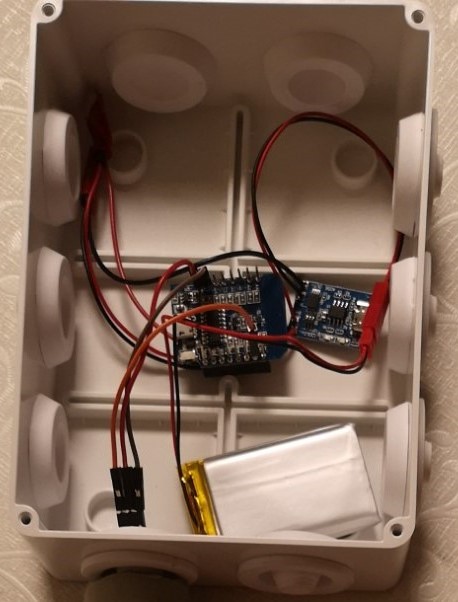
### Експеримент 3

Този експеримент целеше да изчисли времето за което “Четвърти сегмент” може да работи на малка батерия. Той се състоеше от:

* Литиево полимерова батерия - 3,7V 1100mAh,
* Микроконтролер ESP8266 WeMos D1 Mini,
* Сензор за влажност на почвата - Capacitive Soil Moisture Sensor v1.0,
* Платка за зареждане на литиева батерия - TP4056.

Експеримента се изпълни като на първо място се запоиха джъмпер кабелчета за батерията и платката за зареждане на литиева батерия. След това всички модули се свързват по между си и системата се пуска. Последната стъпка беше да се проследи времето през което сегмента е активен.

Резултат от измерването:

* Време на работа – 2 часа и 30 минути

Фигура – Експеримент 3

## Глава III. Избор на софтуерна архитектура

### Java

Java е с общо предназначение, базиран на класове, обектно-ориентиран език за програмиране, предназначен да има по-малки зависимости за изпълнение. Това е компютърна платформа за разработка на приложения. Следователно Java е бърза, сигурна и надеждна.

Програмистът пише програма на разбираем от човека език, наречен изходен код, а компютрите разбират само едно нещо, което се нарича машинен език, който работи на ниво процесор. Следователно машината никога не разбира текста написан от разработчика. Затова имаме нужда от език за програмиране, който да разбира изходен код и да го превежда в машинен, за да може да изпълним програмата.

Тази функция се изпълняват в следните 3 компонента на платформата Java:

#### Java Development Kit (JDK)

Това е кросплатформена среда за разработка на софтуер, която предлага колекция от инструменти и библиотеки, необходими за направа на базирани на Java софтуерни приложения. Примери за включен софтуер са виртуалната машина, компилатор, инструменти за наблюдение на производителността, дебъгер и други помощни програми.

#### Java Runtime Environment (JRE)

JRE е набор от компоненти за стартиране и изпълнение на Java приложения. Той е част от JDK. Състои се от виртуална машина (JVM), библиотеки с класове на Java, технологии за внедряване и др. Той обаче не идва с инструменти за разработка – те са част от JDK.

#### Java Virtual Machine (JVM)

JVM има две основни функции: позволява на Java програмите да работят на всяко устройство или операционна система и да управлява и оптимизира програмната памет. Той действа като двигател по време на изпълнение за стартиране на приложения и е това което Logo, company name

Description automatically generatedизвиква основният метод, присъстващ в кода. JVM е част от JRE (Java Runtime Environment).

Фигура – Java

### IDE – IntelliJ

Интегрираната среда за разработка (7IDE) е софтуер, който комбинира общи инструменти за разработчици на софтуер в единен графичен потребителски интерфейс (8GUI) за изграждане на програми. Тя повишава производителността на програмистите, като комбинира общи задачи за разработка на софтуер в един инструмент, като например автоматично предсказване и дописване на кода. 7IDE обикновено се състои от:

#### Редактор на изходен код

Редактор на изходен код е програма за текстов редактор, създадена специално за редактиране на код на компютърни програми. Може да е самостоятелно приложение, да е вградено в интегрирана среда за разработка (7IDE) или уеб браузър. Редакторите на изходен код са основен инструмент за програмиране, неговите функции са:

* подчертаване на синтаксиса с визуални знаци,
* осигуряване на специфично за езика автоматично довършване,
* проверка за грешки при писането на кода.

#### Инструменти за автоматично изграждане на изходни приложения

Автоматизацията на изграждане е процесът на създаване на софтуер без пряка човешка намеса. Задачите, които традиционно са били отговорност на разработчика се стандартизират, за да се превърнат в скриптови, повтарящи се, автоматизирани стъпки за придвижване на нов софтуер напред до окончателната му форма.

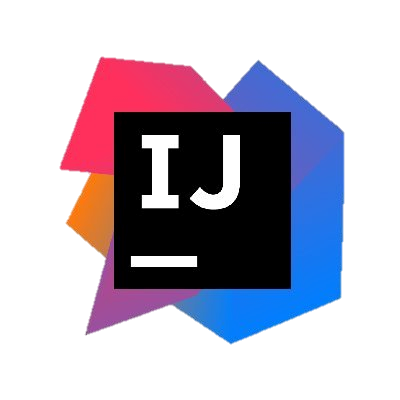
Автоматизацията в 7IDE е процесът на дигитализиране на създаването на софтуерна компилация и свързаните с нея задачи, включително:

* компилиране на компютърен изходен код в двоичен код,
* пакетиране на двоичен код,
* провеждане на автоматизирани тестове.

#### Дебъгер

Дебъгерът е компютърна програма, използвана от програмистите за тестване и отстраняване на грешки в код. Тя използва симулатори с набор от инструкции, вместо да изпълнява системата директно на процесора, за да постигне по-високо ниво на контрол върху нейното изпълнение. Това позволява на дебъгерите да спрат програмата според специфични условия. Използването на симулатори обаче намалява скоростта на изпълнение.

Дебъгерите позволяват на програмиста да проследява текущите си операции и да наблюдава промените в компютърните ресурси (най-често областите на паметта, регистри на процесора и устройства за съхранение, използвани от системата или операционната система на компютъра), които могат да показват неправилно функциониращ код. Повечето програми за отстраняване на грешки включват възможността за стартиране или спиране на програма в определени точки и промяна на съдържанието на паметта или регистър, за да се въвеждат избрани тестови данни по време на работа, които могат да са причина за неправилно изпълнение.

IntelliJ IDEA е базирана на Java интегрирана среда за разработка (7IDE), която има за цел да увеличи максимално производителността на разработчиците. Тя се грижи за ежедневните и повтарящи се задачи, като осигурява интелигентно завършване, статичен анализ и рефакторинг на код, което позволява на програмистите да се концентрират върху по-приятните аспекти на разработката на софтуер.

Фигура 18 – IntelliJ IDEA

### Android Studio

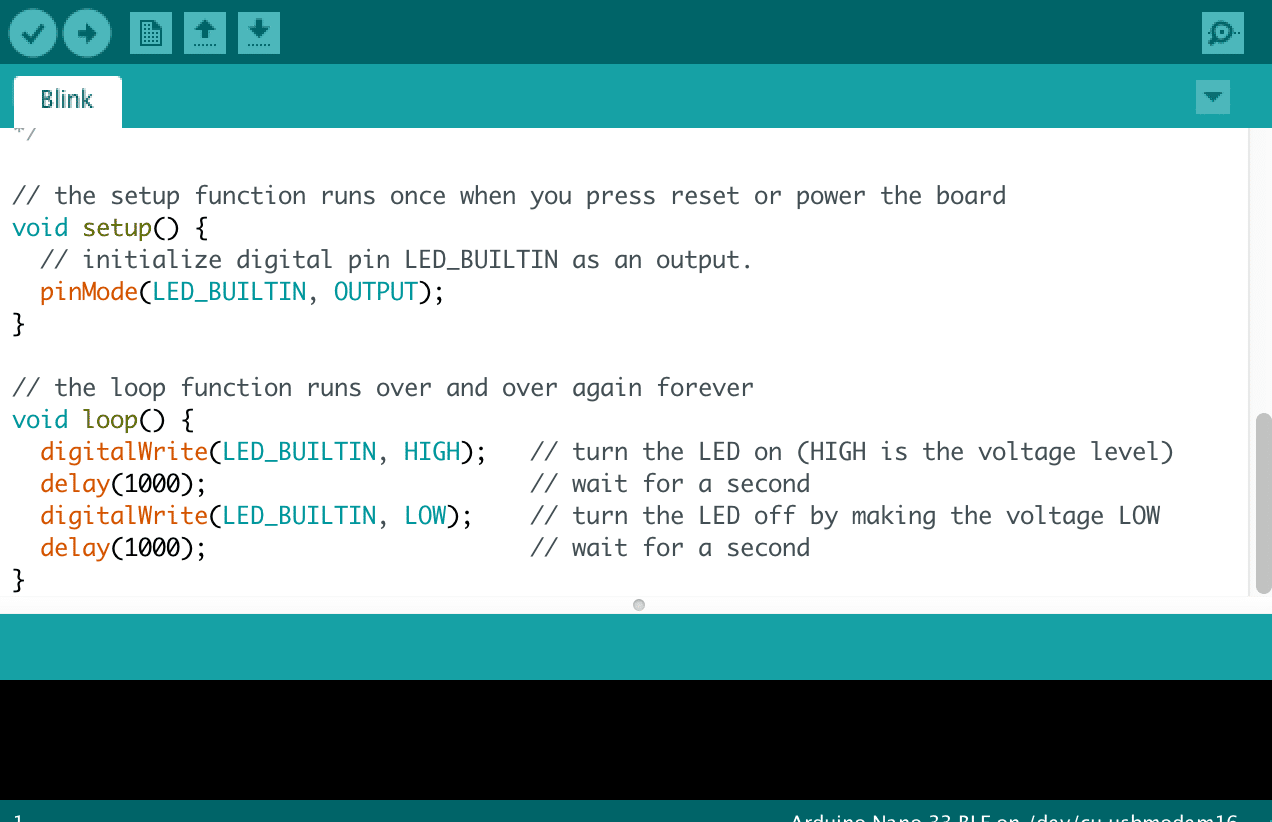


Android Studio (базирана на IntelliJ IDEA) е официалната интегрирана среда за разработка (7IDE) на приложения за Android в допълнение към мощния редактор на код и инструменти за разработчици на IntelliJ. Тя предлага функции, които подобряват производителността при изграждане на приложения, като например:

Фигура 19 – Android Studio

* Гъвкава система за изграждане, базирана на Gradle,
* Виртуално устройство с Android (Емулатор), което е бързо и богато на функции, за стартиране и отстраняване на грешки в приложения,
* Единна среда, в която можете да разработват всички устройства с Android,
* Прилагане на промени, без да има нужда от рестартиране на приложение,
* Интеграция с GitHub, за помощ при импортиране на код,
* Обширни инструменти и рамки за тестване.

### Arduino Софтуер

Софтуерът Arduino (7IDE) включва текстов редактор за писане на код, зона за съобщения, която показва грешки и предоставя обратна връзка по време на съхранение и експортиране, текстова конзола, която включва подробни съобщения за проблеми и друга информация, серия от менюта, лента с инструменти с бутони за основни функции, чрез които може да се изрязва и поставя, да се търси и заменя код, да се валидират и качват програми, да се генерират, отварят и запазват скици и да се отваря 9серийния монитор. Софтуерът се свързва с хардуера на Arduino, което му позволява да качва и комуникира със системи чрез “скици”. Това са програми, създадени със софтуера, изграждат се с текстовия редактор и се запазват с разширението “.ino”.

Фигура 20 – Проста функция на Arduino софтуер

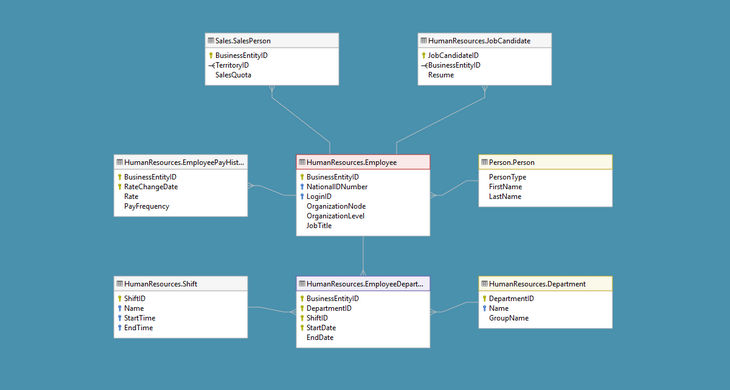
### База данни – PostgreSQL

База от данни (10БД) е организирана съвкупност от взаимосвързани данни, разположени в един или повече файла, отразяващи състоянието и промените в параметри на обекти, наблюдавани от предметна област. Те служат като единна информационна среда за приложения, чрез които се автоматизира процеса на управление. Данните са организирани така, че да има възможност за добавяне на нови, модифициране на съществуващи и провеждане на успешно информационно търсене. Организацията на базата данни трябва да осигури минимално количество излишък на информация, за да позволи използването и по оптимален начин за едно или няколко приложения. Информацията трябва да бъде структурирана по начин, който да предвижда възможност за по-нататъшно нарастване на системата.

База данни е цялата информация, която се въвежда, съхранява, обработва в компютър и се представя под определена форма на потребителя. Целият софтуер, който осигурява тези манипулации с информация, се нарича система за управление на бази данни (11СУБД). Съвременните 11СУБД позволяват на потребителите автоматично да обработват данни в абстрактни термини, които са независими от това как се съхраняват физически в компютър.

В теорията на базата данни съществува стандартна концепция за „ниво на абстракция на представяне на данни“. Те могат да се разглеждат на ниво битове, записани на външен носител, от една страна, а от друга, тази информация представлява напълно различна абстракция за потребителя, като например: потребителско име, продукт и т.н. Всъщност съществуват само 10БД които се разглеждат на първото ниво на физическа абстракция. Останалите са създадени изкуствено за удобство на потребителите. Те са концептуални и представителни и се осигуряват чрез програмни средства, работещи с данните на високо ниво.

За да се направи обработката и заявката на данни ефективно, най-често срещаните типове 10БД, използвани днес, често се моделират в редове и колони в последователност от таблици. След това информацията може да се достъпи, управлява, актуализира, контролира, модифицира и организира с лекота. За писане и запитване на данни повечето бази данни използват структуриран език за заявки - 12SQL.



Фигура 21 – Графична схема на обектно ориентирана база от данни

Съществуват няколко вида 10БД:

* Йерархични бази от данни,
* Мрежови бази от данни,
* Релационни бази от данни,
* Обектно ориентирани бази от данни.

PostgreSQL е обектно-релационна база данни, която използва 12SQL езика, комбиниран с много функции целящи да помогнат на разработчиците да изграждат приложения, на администраторите да защитят целостта на данните и да изградят устойчиви на грешки среди, които да помогнат за управление на информация, без значение колко голям или малък е наборът от такива. PostgreSQL е силно разширяем, например: може да дефинира свои собствени типове данни, да създава персонализирани функции, да пише код от различни езици за програмиране, без да прекомпилира 10БД. Много от функциите, изисквани от стандарта 12SQL, се поддържат, макар и понякога с леко различен синтаксис или функция.

Фигура 22 – PostgreSQL

*\*

### Git - GitHub

GitHub е уебсайт и облачна услуга, която помага на разработчиците да съхраняват, управляват, проследяват и контролират промени в техния код. Базира се на Git системите за контрол и управление на версиите, които поддържат функционалностите и съхраняват модификациите в централно хранилище. Това позволява на разработчиците лесно и бързо да си сътрудничат, като изтеглят текущата версия на софтуера, правят промени и след това ги качват. Тези нови подобрения могат да се разглеждат, изтеглят и допринасят от всеки разработчик. След обновяване на хранилището, в него се запазват старите версии на променените файлове. Така тези файлове могат да бъдат възстановени при необходимост. Системата за контрол на версиите също така следи за конфликти – различни промени на различните потребители, които ползват хранилището.

Фигура 23 – GitHub

Git е предпочитаната система за контрол на версиите от повечето разработчици, тъй като има множество предимства пред другите налични системи. Той съхранява промените на файловете по- ефективно и гарантира по-добра цялост на файла.

### Spring Framework

A picture containing logo

Description automatically generatedSpring Framework е Java платформа с отворен код, използвана за създаване на микро услуга, която осигурява цялостна инфраструктурна поддръжка за разработване на самостоятелно, стабилно и производствено приложение, много лесно и много бързо, което можете веднага да стартирате.

Фигура 24 – Spring Framework

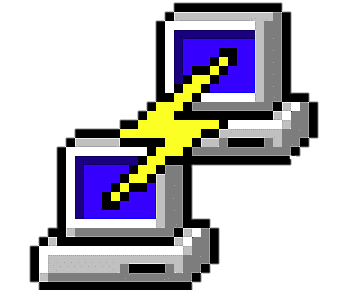
### Postman



Postman е HTTP платформа за изграждане, разработване, проектиране, тестване и използване на 13API, използвайки графичен потребителски интерфейс. Приложението опростява всички стъпки от жизнения цикъл на 13API и ги рационализира, за да може те да се създават по-бързо и по-добре.

Фигура 25 – Postman

### PuTTY

PuTTY е безплатна реализация на 14SSH и алтернатива на 15Telnet клиенти за компютри с Windows. Неговата цел е да осигури двупосочно интерактивно текстово ориентирано комуникационно средство с помощта на виртуално терминално свързване. Основното предимство което има, е че осигурява сигурна, криптирана връзка с отдалечена система. PuTTY поддържа няколко мрежови протокола и е изградена основно от:

Фигура – PuTTY

* емулатор на терминал,
* серийна конзола,
* приложение за мрежово прехвърляне на файлове.

### FileZilla



FileZilla е безплатна за използване, кросплатформена 16FTP помощна програма, която позволява на потребителя да прехвърля файлове от локален компютър към отдалечен компютър. Тя се състои от Client и Server.

16FTP клиент е програма, предназначена за прехвърляне на файлове между два компютъра. С помощта на клиента потребителите могат да качват, изтеглят, изтриват, преименуват, преместват и копират файлове на отдалечен сървър. Докато 16FTP може да бъде достъпен чрез терминал, предпочитат се програми с графичен потребителски интерфейс, а Filezilla представлява точно такова, лесно за използване мултиплатформено решение.

Фигура – FileZilla

### VNC Viewer

VNC означава виртуални мрежови изчисления. Това е графична система за споделяне на екрана, която е създадена за дистанционно управление на друг компютър. Тук екранът, клавиатурата и мишката на компютъра могат да се използват от разстояние, от отдалечен потребител на вторично устройство като се предават актуализациите на графичния екран по мрежа, сякаш седят точно пред него.

 VNC е независим от платформата – има клиенти и сървъри за много 8GUI базирани операционни системи и за Java. Множество клиенти могат да се свързват към сървъра едновременно. Тази технология включват дистанционна техническа поддръжка и достъп до файлове на работния компютър от домашния или обратно.

Фигура – VNC Viewer

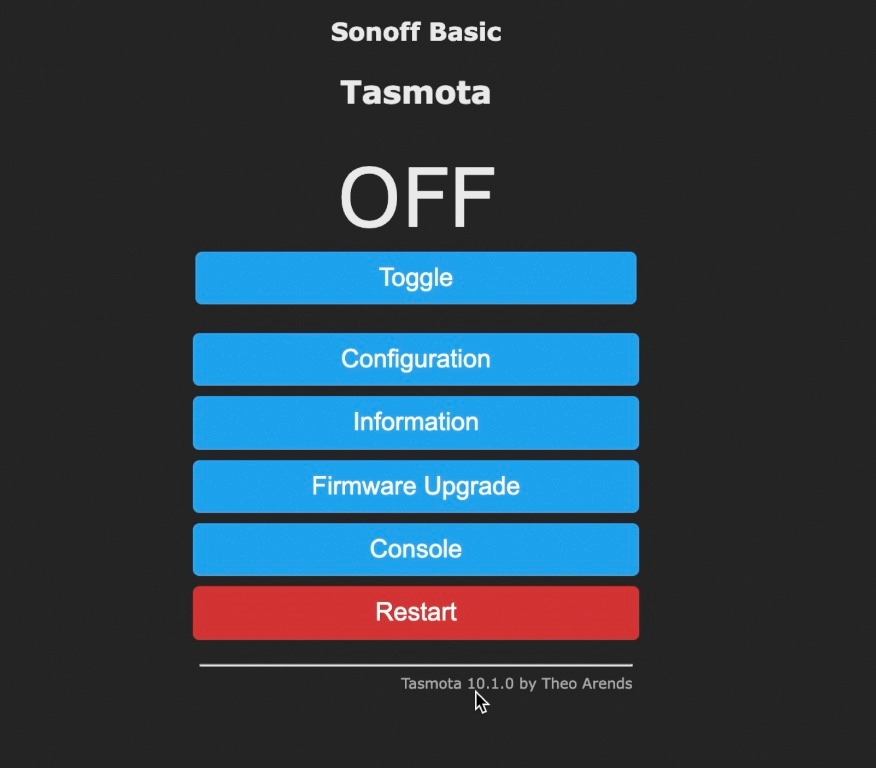
### Raspberry Pi Imager

Graphical user interface

Description automatically generatedRaspberry Pi Imager е бързият и лесен начин за инсталиране на операционна система за Raspberry Pi и други такива на microSD карта, готова за използване. Това прави процеса на изобразяване много по-лесен. Той централизира всички най-разпространени инструменти и софтуер за изображения, от които се нуждаете, в един пакет.

Фигура – Raspberry Pi Imager

### Tasmota

Tasmota е персонализирана система с отворен код, която работи на всяко интелигентно домашно устройство, което използва ESP 3Wi-Fi чип. Първоначално системата стартира като персонализиран вграден софтуер за Sonoff устройства, но след това се трансформира в мощен инструмент за контрол на всичко в ESP чипа. Tasmota идва с вградени драйвери за управление на много популярни сензори и чипове. Освен всичко това, тя е лека и лесна за инсталиране и има няколко метода за това.

Фигура – Tasmota

### ESP Easy

ESP Easy позволява създаване на сензорни възли с безжична връзка към локална мрежа чрез 3Wi-Fi, без да се пише никакъв код. Системата се използва за превръщане на модули в прости многофункционални сензорни и задвижващи устройства на платформи за домашна автоматизация. След като се зареди на хардуера, конфигурацията на ESP Easy е изцяло базирана на уеб интерфейс и се използва основно върху компоненти ESP8266.

Фигура – ESP Easy

## Глава IV. Избор на хардуерни компоненти

### Raspberry Pi - Raspberry Pi Zero W

**Характеристики на Raspberry Pi Zero W:**

* 802.11 b/g/n wireless 17LAN
* Bluetooth 4.1
* 1GHz, едноядрово централен процесор
* 512MB 18RAM
* Mini HDMI порт и Micro-USB порт
* Micro USB порт за захранване

Фигура – Raspberry Pi Zero W

* 40 6GPIO пин конектори
* Порт за камера

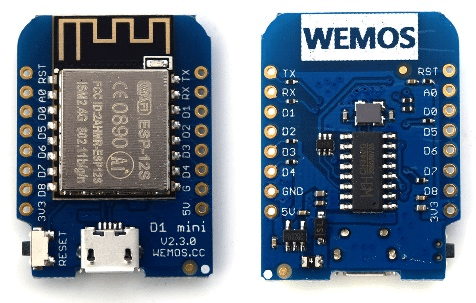
Raspberry Pi е малък едноплатков компютър. Чрез свързване на периферни устройства като клавиатура, мишка и дисплей, той може да действа като мини персонален лаптоп. Използва се в направата на изображения и видео в реално време, базирани на IoT приложения и приложения за роботика. Raspberry Pi е по-бавен от нормален компютър, но може да предостави всички очаквани функции или способности при ниска консумация на енергия и нещо повече, тъй като осигурява достъп до вградения в чипа хардуер, т.е. 6GPIO за разработване на приложение. Така можем да свързваме устройства като светодиоди, двигатели, сензори и т.н. и да ги контролираме.

Официалната операционната система е достъпна за безплатно използване. Тя е ефективно оптимизирана и има 8GUI, който включва инструменти за сърфиране, програмиране на Python, офис системи, игри и др.

Raspberry Pi Zero W е 8-мият подред официално произведен джобен компютър. Платката е с по-малки размери и е по-лека от кредитна карта. Тя е изградена от 1GHz, едноядрен процесор и 512GB 18RAM. Има два микро USB порта, единият който се използва за захранване на платката, и един мини HDMI порт. В тях може да се добавят всякакви устройства. Интегриран е и конектор за камера, в случай че има нужда от такава. Най-голямото предимство, което този, 8-мо поколение мини компютър, притежава пред предците си, освен малките си размери, са Bluetooth 4.0 и 802.11n 3Wi-Fi чиповете, чрез които може да се свързват различни устройства и системи.

### Микроконтролер ESP8266 WeMos D1 Mini

**Характеристики на ESP8266 WeMos D1 Mini:**

* Работно напрежение: 3,3v
* 6GPIO портове: 11
* Аналогов вход: 1 (max. 3,3v)
* Тактова честота: 80MHz/160MHz
* Флаш памет: 4Mb
* Съвместима с Arduino
* Размери:  34мм x 26мм
* Тегло: 9гр.

Фигура – ESP8266 WeMos D1 Mini

WeMos D1 Mini е базиран на микроконтролера ESP8266, проектиран и произведен от Espressif Systems в Шанхай. ESP8266 има вградена зареждаща програма, която улеснява флашването на платката с програмен код, който създавате. D1 Mini е невероятно гъвкав, защото е евтин, с активиран 3WiFi и напълно съвместим с платформата Arduino.

### Sonoff Basic R2 DIY Smart Switch

**Характеристики на Sonoff Basic R2 DIY Smart Switch:**

* Захранване: 90 - 250V AC
* Макс.ток: 10А
* Макс.мощност: 2200W (неиндуктивен товар)
* WiFi стандарт: IEEE 802.11 b/g/n
* Размери: 88 mm x 38 mm x 22 mm
* Тегло: 46 g

Фигура – Sonoff Basic R2 DIY Smart Switch

Този модул е предназначен за дистанционно управление на различни устройства, електроуреди и др. в домовете и офисите. В допълнение към дистанционното управление е възможно да се зададат предварително, времена за включване и изключване на устройството. Функции като споделени контроли и интелигентни сцени предоставят допълнителни опции за управление. Лесно може да се добавят и управляват различни устройства с приложения за Android.

### Електромагнитен вентил - Tork t-gp 104



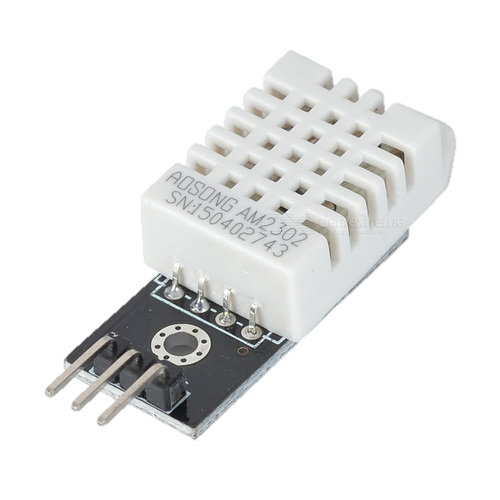
**Характеристики на Tork t-gp 104:**

* Нормално затворен
* Мощност: 220 V
* Налягане: 16 бара
* Защита: IP 65
* Температура на издържане: 130 °C
* Тегло: 0,80 кг.

Фигура – Tork t-gp 104

Електромагнитен вентил е електромеханичен клапан, който обикновено се използва за контролиране на потока на течност или газ. Клапанът има магнит, който представлява електрическа намотка с подвижно феромагнитно ядро (бутало) в центъра. В положение на покой буталото затваря малък отвор. Когато електрически ток мине през намотката се създава магнитно поле. Това поле упражнява сила нагоре върху буталото, отваряйки отвора. Вентилите могат да бъдат нормално затворени или нормално отворени, което зависи от това в какво състояние е буталото (вдигнато или свалено), когато не минава ток през системата.

### Сензор за температура и влажност на въздуха - DHT22

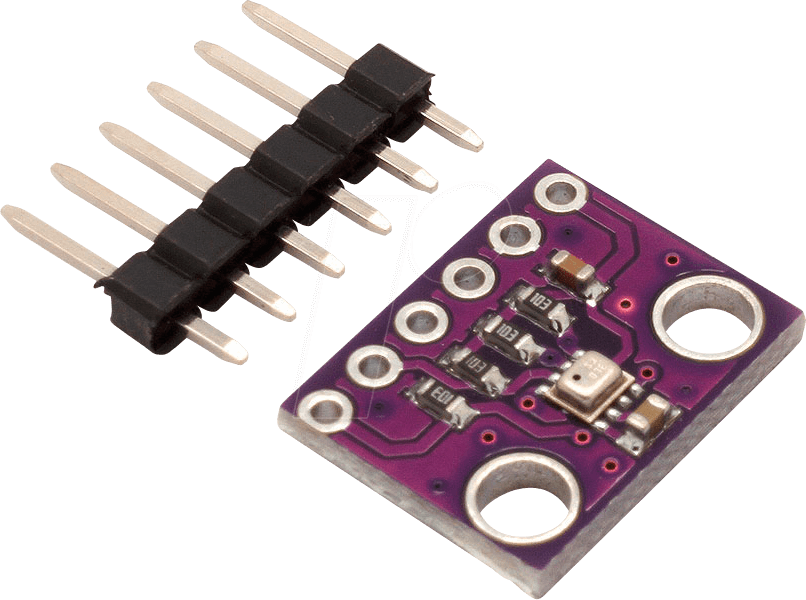
**Характеристики на DHT22:**

* Захранване: 3,3V – 5V
* Измерване влажност: от 0% до 99%, с точност ±2% (при 25°)
* Измерване температура: от -40° до 80° C, с точност ± 0,5°
* Честота на опресняване: до 1Hz (1 сек)
* Разстояние на предаване: 20м

Фигура – DHT22

DHT22 е калибриран модул за температура и влажност. Той използва капацитивен сензор за влажност и термистор (резистор, чието съпротивление се влияе от температурата на околната среда) за измерване на околния въздух и изхвърля цифров сигнал върху щифта за данни. Неговите сензорни елементи са свързани с 8-битов едночипов компютър. Малкият размер, ниската консумация и голямото разстояние на предаване (20 м) позволяват на модула да бъде включен в много проекти. DHT22 измерва температура от -40° до 80° C (точност ± 0,5°) и относителна влажност на въздуха в диапазон 0% – 99% (точност ±2%).

### Сензор за налягане - BMP280

**Характеристики на BMP280:**

* Размери: 19mm x 18mm
* Тегло: 1 гр
* Захранване: 3V – 5V
* Налягане: 300 hPa – 1100 hPa с точност ±1 hPa
* Надморска височина: до 9000m с точност ±1m

Фигура – BMP280

* Температура: -20°C до +85°C, точност ±1°C в диапазон от 0°C до 65°C

BMP280 е сензор за абсолютно барометрично налягане, който е приложим във всякакви мобилни приложения. Малките размери и ниската консумация на енергия позволяват внедряване в устройства, захранвани от батерии. Модула е базиран на пиезорезистивна технология, характеризираща се с висока точност и линейност, както и дългосрочна стабилност и висока устойчивост. Устройството е оптимизирано по отношение на консумация на енергия, разделителна способност и производителност на филтъра.

### Сензор за влажност на почвата - Capacitive Soil Moisture Sensor v1.0

**Характеристики на Capacitive Soil Moisture Sensor v1.0:**

* Работно напрежение: 3,3 ~ 5,5 V
* Работен ток: 5mA
* Интерфейс: PH2.0-3P
* Размери: 3,86 x 0,905 (Д x Ш)
* Тегло: 15гр

Фигура – Capacitive Soil Moisture Sensor v1.0

Сензорът за влажност на почвата измерва нивата на влажност чрез капацитивно наблюдение. Този модул е изработен от устойчив на корозия материал, включва вграден регулатор на напрежението, който му дава диапазон на работно напрежение от 3,3 ~ 5,5V и е идеален за микроконтролери с ниско напрежение.

### FTDI - UART TTL FT232RL

**Характеристики на UART TTL FT232RL:**



* Работно напрежение:  3.3V и 5V
* Индикация за комуникация (RX и TX) чрез два светодиода
* Размери: 36мм х 18мм

Фигура – UART TTL FT232RL

* Тегло: 4гр

Платката е компактна и се свързва към компютър чрез USB конектор. Отговорността на този хардуер е да каже на вашия компютър какво представлява даден модул, така че вашият компютър да може да зареди правилния драйвер, логика и буфери за него, както и да управлява транзакциите с данни с компютъра след това. Или по-просто казано служи за връзка или програмиране на различни устройства, които нямат вграден начин за връзка.

### Литиево-йонни батерии - INR18650-MH1

A picture containing text, battery, electronics

Description automatically generated

**Характеристики на литиево-йонни батерии:**

* Размер на клетката: MR18650
* Нормално напрежение: 3.6V
* Капацитет: 3200mAh
* Размери на корпуса: 18.4mm x 65.1mm
* Максимален ток: 10A
* Зареждаема литиево-йонна батерия

Фигура 40 – Литиево-йонни батерии

Литиево-йонни зареждаеми батерии за захранване.

### Модул за зареждане на литиева батерия - TP4056

A close-up of a computer chip

Description automatically generated with medium confidence**Характеристики на модул за зареждане на литиева батерия:**

* Метод на зареждане: линейно зареждане
* Заряден ток: 1A (регулируем)
* Точност на зареждане: 1.5
* Входно напрежение: 4.5V-5.5V
* Напрежение при пълно зареждане: 4.2V
* Работна температура: от -10 ° C до +85 ° C

Фигура 41 – Модул за зареждане на литиева батерия

TP4056 е линейно зарядно устройство с постоянен ток и напрежение за единична клетка литиево-йонни батерии. Ниският му брой външни компоненти прави модула идеално подходящ за преносими приложения. Напрежението на зареждане е фиксирано на 4.2V, а токът на зареждане може да се програмира външно с резистор. TP4056 автоматично прекратява цикъла на зареждане, когато зарядният ток спадне до 1/10-та от програмираната стойност след достигане на крайното плаващо напрежение. Други функции включват блокиране под високо напрежение и автоматично презареждане.



### Соларен панел - LIGHTEX 5W 9V

**Характеристики на соларен панел**

* Материал на соларната клетка: поликристален силикон
* Размери: 259 x 192 x 17 mm
* Оптимално работно напрежение: 9.2V
* Оптимален работен ток: 540mA

Фигура 42 – Соларен панел

Соларните панели са екологични източници на енергия. Известни са още като фотоволтаични системи. Те превръщат енергията на слънцето в електрическа чрез преобразуване на светлината от слънцето, която се състои от частици енергия, наречени "фотони", което може да се използва за захранване.

## Глава V. Разработка на софтуер

В тази глава ще обясним написаният софтуер. Всички основни раздели от кода съдържат 10 на брой класа, но всеки един от тях е подобен на останалите в съответната част, затова ще обясним само 1 от тях.

### Back-end – Spring Framework

1Back-end-а се дели на 5 основни части, като 2 от тях се разделят: първата на два подраздела, а втората на един. Така общата структура става 8 отделения. Той е написан чрез Spring Framework-а и създава HTTP заявки, които взимат информация от Базата данни.

#### Общ код за всеки клас

##### Библиотеки

Text

Description automatically generatedБиблиотеките в езиците за програмиране са колекции от предварително написан код, който потребителите могат да използват за оптимизиране на своите задачите.

Фигура 43 – Библиотеки в Java

##### Конструктор

Конструкторът е специален метод на клас или структура в обектно-ориентирано програмиране, който инициализира новосъздадения обект от този тип. Всеки път, когато се създава обект, той се извиква автоматично. Конструкторът е като метод, който обикновено има същото име като класа и може да се използва за задаване на стойностите на членовете на обект, както по подразбиране, така и на дефинирани от потребителя стойности.

#### Първа част – Конфигурации

Text

Description automatically generated Папката с конфигурации съдържа два класа в нея. Първият е LocalDateConverter и неговата цел е да конвертира типа данни LocalDate чрез формати във String.

Фигура 44 – LocalDateConverter

Text

Description automatically generatedВторият клас е ScheduledTasks и той се грижи определен клас да се извиква на интервал който му е зададен.

Фигура 45 – ScheduledTasks

Анотация Scheduled отговаря за интервала в който класа вътре се извиква, в нашият случай всяка секунда. След това създаваме метод restClient. В неговото тяло имаме 2 реда код. Първият създава обект от класа RestClient, а вторият извиква метода get.

#### Втора част – Rest контролери

Rest контролерите са класовете които ни позволяват да извикваме логиката на определен адрес.

Text

Description automatically generated

Фигура 46 – Първа част на Rest контролер

Text

Description automatically generatedПървата част от кода започва с 2 анотации. Първата анотира класа като Rest контролер, а втората насочва към URL-а, от който може да се достъпи. Следващият ред създава обект от ConfigurationService класа, в който се съдържа логиката на програмата. Последните 3 реда от тази част са конструктор.

Фигура 47 – Втора част от Rest контролер

Text

Description automatically generatedВтората част от един контролер съдържа 19CRUD операции. Има два различни “Get” метода, първият взима всички данни от една таблица в база данни, а вторият взима само тези на определено id. „Create“ метода създава и добавя информация в таблица на базата данни, а „Update” може да я променя на съответния номер. “Delete” се използва за изтриване на данни отново на определено id.

Фигура 48 – Трета част на Rest контролер

Третата част са всички персонализирани заявки. Те се пишат, когато има нужда от взимане или изпращане на данни по определен начин или URL с анотацията @RequestMapping. В този случай извикваме getActiveTime метода.

#### Трета част – Entity

Text

Description automatically generated

Entity класовете са огледално код на таблиците в базата данни.

Фигура 49 – Първа част на Entity класовете

Първата част от Entity класа са редовете в една таблица от базата данни. Те се анотират чрез @Column. @Id и @GeneratedValue (генерира всяко следващо число автоматично) са специфични за Id-то във всяка таблица. Връзките между таблиците използват @OneToMany, @ManyToOne и @ManyToMany според вида им. Имат и анотации @JsonIgnore и @JsonIgnoreProperties, които се грижи върнатият резултат от заявката да не е рекурсивен/безкраен.

Фигура 50 – Втора част на Entity класа

Text

Description automatically generated

Втората част на кода са прости конструктори, ”getter” и “setter” методи от които да може да се достъпват данните.

#### Четвърта част – Repository

Text

Description automatically generatedRepository класовете са връзката между базата данни и нашите Entity-та. Те удължават JpaRepository, който е специфично разширение на хранилището за JPA. В този клас могат и да се пишат Derived Query Methods. Те позволяват да се изпълняват различни видове заявки за извличане на данни от базата данни, като може или да се използва името на метода, за да се извлече директно заявка или ръчно да се дефинира собствена, като се използва анотацията @Query.

Фигура 51 – Repository

#### Пета част – Service

Text

Description automatically generatedService-ите са Interface-и. В езика за програмиране Java се дефинира като план на клас, който съдържа статични константи и абстрактни методи и е механизъм за постигане на абстракция. С други думи, те могат да имат само методи, които нямат тяло и променливи.

Фигура 52 – Service

#### Шеста част – Implementation

Implementation класовете съдържат логиката на програмата и имплементират Service.

Text

Description automatically generatedПървите редове код са прости. Създава се обект от класа configurationRepository, после се инициализира чрез конструктор и му се създава “Get” метод при нужда на достъпване.

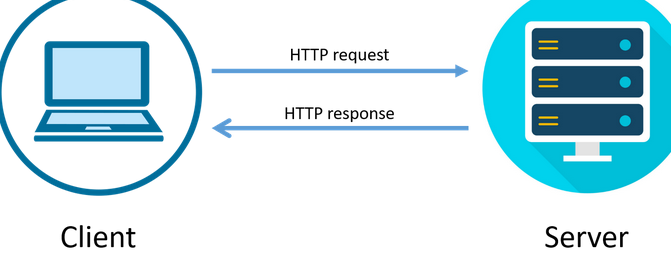
Фигура 53 – Първа част от Implementation класа

Втората част на кода съдържа методите с логиката на нашите 19CRUD операции и персонализираният getActiveTime. В първият “Get” метод се взимат всички данни от дадена таблица. На вторият логиката взима информацията само по определено Id, като прави проверка дали съществува такова. В “Create” се използва вече създаден метод “save”, който е от JpaRepository за запазване на данни в таблица, а в “Update” се използва “setter” от configuration entity класа, накрая се повтаря логиката от предишният метод за запазване. Text

Description automatically generated“Delete” получава Id на което да изтрие информацията и го прави чрез deleteById процеса.

Фигура 54 – Втора част от Implementation класа

#### Седма част – Spring Client

Spring Client (клиент) кода изпълнява функцията за връзка със Server (сървър). Тази функционалност е създадена, за да може да се получават изпратените данни от сензорите в “четвърти сегмент” към главната логика и от нея към Android приложението и базата от данни.

Фигура 55 – HTTP request and response

#### Осма част – System Application

Text

Description automatically generatedТози клас е основният в Java приложение и той инициализира нашата програма.

Фигура 56 – System Application

System Application има 2 анотации: @SpringBootApplication и @EnableScheduling. Първата се използва за стартиране на Spring приложение от основния метод. Тя автоматично сканира конфигурационните класове и стартира приложението. С втората можем да активираме конфигурационният ScheduledTasks клас в приложението.

Следващата стъпка е да създадем нашият SystemApplication клас, главният ни main метод и да пуснем програмата чрез специализираната “run” команда.

### Front-end – Android

Android приложението е връзката с нашият потребител и системната ни логика. През него може да се задава продължителността на поливане, часовете на поливане, максималните и минималните данни от сензори при които може да се осъществи поливане и тн. То е изградено от:

* Java класове,
* Xml файлове.

Java класовете са логиката на всяка страница в приложението. Те предимно управляват данните които се показват, функционалностите на бутоните и полетата за показване и писане, местоположението на елементите на всяка страница, новите стойности, които потребителя задава и тн. XML файловете са подобни на HTML, но за разлика от тях, те са чувствителни към малки и големи букви, изисква се всеки маркер да е затворен правилно и запазват празното пространство. В нашето приложение се използват за изобразяването на модулите и тяхното манипулиране от клиента.

Приложението е изградено на няколко страници като всяка една от тях си има собствена цел и идея. Те са 6 на брой:

* Страница за проверка на връзката с мрежата,
* Главна страница, която разпределя към останалите,
* Страница за проверка на моментните стойности от сензорите,
* Страница за проверка на грешки появили се при момента на работа на системата,
* Страница за конфигуриране на данните за поливане,
* Страница за настройване на максималните и минималните данни за поливане от сензорите.

## A picture containing chart Description automatically generatedГлава VI. База данни

Фигура 57 – Структура на база данни

Graphical user interface, text, application

Description automatically generated

### Таблица 1 – “Configuration”

Базата от данни е структурирана от 9 публични таблици. Първата има 11 полета:

* Configuration\_id – тип Integer. 20Колоната за идентичност,
* Active\_time – тип Integer. Тя съдържа времето за поливане,
* Полета за всеки ден от седмицата – тип boolean. Чрез тях се запазват дните през които да се полива,
* Configuration\_changed\_by – тип Integer. В нея се съхранява потребителя който за последно е променил конфигурацията за поливане,

Фигура 58 – Таблица "Configuration"

* Valve\_id – тип Integer. 21Външен ключ към таблицата “Valve”

Тази таблица има за цел да запазва основните конфигурации и данни за поливане. Тя е свързана към таблицата “Valve” с връзка ManyToOne и таблицата “Watering\_hours” с OneToMany.

### **Таблица 2 – “Valve”**

Graphical user interface, application

Description automatically generated

Втората таблица има 2 полета:

* Valve\_id – тип Integer. 20Колоната за идентичност,
* Valve\_name – тип character varying. Съхранява името на крана.

Фигура 59 – Таблица "Valve"

Тази таблица пази информация за добавените кранове в системата. Тя е свързана към таблицата “Configuration” с връзка OneToMany.

### Таблица 3 – “Watering\_Hours”

Graphical user interface, text, application

Description automatically generatedТази таблица има 3 полета:

* Watering\_hours\_id – тип Integer. 20Колоната за идентичност,
* Time – тип character varying. Колоната запизва часа на поливане
* Configuration\_id – тип Integer. 21Външeн ключ към таблицата “Configuration”

Фигура – Таблица "Watering\_hours"

Третата таблица запазва часа за поливане на градината. Тя е свързана към таблицата “Configuration” с връзка ManyToOne.

### Таблица 4 – “Users”

Четвъртата таблица е изградена от 6 полета:

* Graphical user interface, text, application

  Description automatically generatedUser\_id – тип Integer. 20Колоната за идентичност,
* User\_full\_name – тип character varying. Тук се пази пълното име на потребителя,
* User\_login - тип character varying. Тук се пази потребителското име на клиента,
* User\_pass - тип character varying. Тук се пази паролата на потребителя,
* User\_create\_date – тип timestamp without time zone. Съхранява датата и часа на създаване на потребителския акаунт,
* User\_admin – тип boolean. Пази дали потребителят е admin.

Фигура 61 – Таблица "Users"

Тази таблица пази данните за регистрираните потребители в системата. Тя е свързана към таблицата “Logs” с връзка OneToMany и “Devices” с OneToMany.

### Таблица 5 – “Logs”

Петата е изградено то 5 полета:Graphical user interface, text, application, chat or text message, email

Description automatically generated

* Log\_id – тип Integer. 20Колоната за идентичност,
* Log\_date - тип timestamp without time zone. Тук се запазва дата и часа на създаване на записа,
* Log\_description – тип character varying. Това е информацията за записа,
* Log\_level – тип character varying. Тук се записва на какво ниво е записа: “error” или “warning”,
* User\_id – тип Integer. 21Външeн ключ към таблицата “Users“.

Фигура 62 – Таблица "Logs"

Тази таблица запазва информацията от всички записи които са се появили по време на работа на програмата. Тя е свързана към таблицата “Users” с връзка ManyToOne.

### Таблица 6 – “Devices”

Graphical user interface, text, application, email

Description automatically generatedШестата таблица от нашата база от данни съдържа 4 полета:

* Device\_id – тип Integer. 20Колоната за идентичност,
* Device\_name – тип character varying. Тук се пази името на устройство,
* Device\_location – тип character varying. Тук се съдържа местоположението на устройството.
* User\_id – тип Integer. 21Външeн ключ към таблицата “Users“.

Фигура 63 – Таблица "Devices"

Шестата таблица запазва типа на всички устройства и тяхното местоположение. Тя е свързана към таблицата “Sensors” с връзка OneToMany и “Users” с ManyToOne.

### Таблица 7 – “Sensors”

Graphical user interface, text, application

Description automatically generated

Седмата таблица е изградено от:

* Sensor\_id – тип Integer. 20Колоната за идентичност,
* Sensor\_type\_id - тип Integer. Това поле е връзка с таблицата “Sensor\_Type“.
* Devices\_id – тип Integer. 21Външeн ключ към таблицата “Devices“.

Фигура – Таблица "Sensors"

Таблицата има за цел да запазва сензорите и техният “device” тип, тя е свързана към таблицата “Sensor\_Types” с връзка OneToMany и към “Devices” чрез OneToMany.

### Таблица 8 – “Sensor\_Types”

Graphical user interface, text, application

Description automatically generated

Осмата таблица съдържа 3 полета:

* Sensor\_type\_id – тип Integer. 20Колоната за идентичност,
* Sensor\_type – тип character varying. Тук се пази типа на сензора,
* Data\_type – тип character varying. Тук се съдържа типа на сензора.

Таблица запазва типа на сензорите, тя е свързана към таблицата “Sensors” с връзка OneToMany.

Фигура 65 – Таблица "Sensor\_Types"

### Таблица 9 – “Sensor\_Data”

Graphical user interface, text, application, email

Description automatically generated

Последната таблица е изградена от 4 колони:

* Sensor\_data\_id – тип Integer. 20Колоната за идентичност,
* Sensor\_value – тип character varying. Колона за запазване на резултата върнат от сензор,
* Sensor\_value\_date – тип timestamp without time zone. Колона за запазване на времето на получаване на данните,

Фигура – Таблица "Sensor\_Data"

* Sensor\_id – тип Integer. 21Външeн ключ към таблицата “Sensors”.

Тази таблица запазва стойностите и часа на данните изпратени от сензорите. Тя е свързана към таблицата “Sensors” с ManyToOne.

## Глава VII. Разработка на хардуер

### Разработка на “Първи сегмент”

Първият сегмент няма физическа връзка, тук единственото свързване е с мрежата.

### Разработка на “Втори сегмент”

Вторият сегмент няма физическа връзка, тук единственото свързване е с мрежата.

### Разработка на “Трети сегмент”

A circuit board with wires

Description automatically generated with low confidenceВ електромагнитният кран бяха запоени 3 жици. В последствие те се свързват към смарт 3Wi-Fi релето, което също е запоявано. То пък на свое място е скачено към ток и мрежата.

Фигура – Запояване на Sonoff Basic R2 DIY Smart Switch



Фигура – Свързан електромагнитен вентил

### Разработка на “Четвърти сегмент”

Изграждането на “Четвъртият сегмент” започна със запояването на модула за зареждане на литиеви батерии. Към него се запоиха 4 жици като предназначението на 2 от тях е да се включат към батерията, а останалите към микроконтролера ESP8266.

A close-up of some wires

Description automatically generated with low confidenceA picture containing wall, indoor

Description automatically generatedСлед като батерията се скачи, следващата стъпка от разработката е микроконтролера и модула за зареждане да направят връзка и да се установи токова мрежа. Когато това се изпълни, сензорите се закачат към микроконтролера чрез джъмпер кабелчета по своите места на платката и софтуерът се инициализира.

Фигура – Запояване на модул за зареждане на литиеви батерии

Фигура – Процес на разработка на "Четвърти сегмент"

## Глава VIII. Стартиране на системата от свързани устройства

### Подготвяне на Raspberry Pi Zero W

Началната стъпка за стартиране на системата е да се използва Raspberry Pi Zero W или високи модели от Raspberry Pi 3 Model B (2016) и по-късно, тъй като те имат 3Wi-Fi модул. Първо трябва да се настрои микрокомпютърът и да се свърже към мрежата. Това се случва посредством [онлайн урок](https://desertbot.io/blog/headless-pi-zero-w-wifi-setup-windows)[1]. След като е свързан, ще трябва да разберем IP адресът му от списъка с устройства, свързани към маршрутизатора. Щом тази стъпка е изпълнена трябва да резервираме въпросният адрес.

### Инсталиране на Java и Maven

* Graphical user interface, text, application

  Description automatically generatedПървата стъпка е да достъпим Raspberry Pi. Това се случва като отворим “PuTTY” сесия и в полето за “Host name” въведем IP-то на нашият микрокомпютър.

Фигура 71 – PuTTY сесия

* След като сме въвели данните за връзка трябва да въведем името и паролата на Raspberry Pi, които по подразбиране са pi и raspberry.
* След това трябва да инсталираме Java 8. Това се случва посредством [онлайн урок](https://phoenixnap.com/kb/install-java-raspberry-pi)[2].
* Инсталираме и Maven, отново посредством [онлайн урок](https://xianic.net/2015/02/21/installing-maven-on-the-raspberry-pi/)[3].

### Инсталиране на функционалния код от GitHub

* Достъпваме Raspberry Pi като отворим “PuTTY” сесия и в полето за “Host name” въведем IP-то на нашият микрокомпютър.
* След като сме въвели данните за връзка трябва да въведем името и паролата на Raspberry Pi, които по подразбиране са pi и raspberry.
* Кода се инсталира от [публичното хранилище](https://github.com/Initaro/watering_system_spring) в Github, чрез командата git clone и URL-а на хранилището.

Фигура 72 – Git clone команда

* Навигира се до папката с проекта.

Фигура 73 – Команда за навигиране до папката с проекта

* Проекта се компилира с командата “mvn clean install”, после се достига до директорията “target”. където се изпълнява командата “java -jar demo-0.0.1-SNAPSHOT.jar”, която инициализира нашият проект.

Фигура – Команда за изпълняване на проекта

### Подготовяне на микроконтролер ESP8266 WeMos D1 Mini

Микроконтролера трябва да се свърже към компютър чрез кабел USB към micro USB A. След това се сваля и отваря Arduino софтуерът. След като стъпката е изпълнена имаме нужда да свалим библиотеките за управление на сензорите, чрез LibraryManager който се намира под Tools > Library Manager или с комбинация от бутоните - Ctrl + Shift + I. Те са 7 на брой:

* Adafruit BMP280 Library
* Adafruit BusIO
* Adafruit Circuit Playground
* Adafruit GFX Library
* Adafruit Unified Sensir
* BMP280\_DEV
* DHT sensor library

Имаме нужда да свалим и Board с който ще работим. Това може да стане като натиснем Tools > Board > Board Manager и инсталираме esp8266 Board. Щом е изпълнена тази стъпка трябва да отидем до Tools > Board > ESP8266 Boards и да натиснем LOLIN(WEMOS) D1 R2 & mini.

След като сме подготвили нашето 7IDE и сме свързали микроконтролера, трябва да вземем Arduino кода от [публичното хранилище](https://github.com/Initaro/watering_system_spring) в GitHub и да го сложим в приложението. Накрая трябва само да натиснем “Upload” бутона и модула е готов за използване.

### Подготвяне на Sonoff Basic R2 DIY Smart Switch

Вграждането на софтуер на Sonoff Basic R2 DIY Smart Switch се случва чрез FTDI - UART TTL FT232RL и [онлайн урок](https://www.youtube.com/watch?v=H6YUWkeSZNU)[4].

### Инсталиране на Android приложение

Android приложението за телефон се сваля в 6 прости стъпки:

* Инсталирайте системата от [публичното хранилище](https://github.com/Initaro/watering_system_spring) в Github.
* Навигирайте до директорията “Android”.
* Там трябва да се намира изпълним файл с разширение “.apk”.
* Инсталирайте го върху вашето устройство.
* Свържете телефона към мрежата в която са и останалите модули.
* Отворете приложението.

# Заключение

Земеделието и селскостопанската индустрия се развиват с изключително бързи темпове и поддържането става все по-сложно и трудно, затова разработената дипломна работа на тема: “Проектиране и изработка на смарт управление за поливна система” представлява цялостен и завършен проект, отразяващ автоматизацията и проектирането на система за автоматично поливане. По време на изработване, тази система премина през два етапа.

Първият представлява 4demo на проекта, анализ на същността, характеристиките и архитектурата на проекта и неговите компоненти и определяне на основните изисквания, на които трябва да отговаря. Това изследване доказва все по-голямо интегриране на Интернет на Нещата във все повече сфери от реалния живот и бизнеса, както и ползата от използването му за различни приложения.

След съгласуване и одобрение на проекта за автоматизирана поливна система се преминава към неговото изпълнение. Вторият етап се състоеше от проектирането и изграждане на системата за “Проектиране и изработка на смарт управление за поливна система”. По време на нея бяха изработени 1Back-end, 2Front-end и Базата от данни като за тяхното реализиране са използвани техническите средства – Java и 12SQL. Проекта се изгради на сегментен принцип, което позволява разширяване на нейната функционалност и добавяне на нови модули, в зависимост от потребностите, които могат да възникнат в последствие.

Автоматизирането на поливната система превръща трудна и понякога досадна задача в детска играчка!

# Практическа част

Тема “Проектиране и изработка на смарт управление за поливна система“

Системата от свързани устройства покрива всички изисквания и функционалности, описани в заданието. Употребени са всички описани модули. Използването на системата е лесно за управление и не изисква почти никакви специални умения. Към нея могат да бъдат добавени много и различни умни устройства, които да бъдат контролирани от мобилното приложение.

Устройството за “Смарт управление на поливна система” е приложима в много различни напоителни системи. Oсновни ползи като: автоматична и лесна поддръжката на градини и селско-стопански площи нуждаещи се от постоянна грижа, намаляване на консумацията на вода и енергия и съкращаване на нуждата от човешка ръка са реализирани. Използването на системата е интуитивно и осигурява на всеки потребител бърз, модерен и удобен начин за поливането на своята градина.

Тази версия на системата има следните възможности за развитие:

* добавяне на допълнителен език на интерфейса и съдържанието,
* търсачка и филтриране на данните,
* повече функционалности за по разнообразно и автоматизирано поливане,
* да се контролира часа и времето за поливане в отделните дни от седмицата,
* защита на системата от нарушители.

# Списък на използваните фигури и диаграми

[Фигура 1 – Диаграма на дипломната работа 10](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116418)

[Фигура 2 – Raspberry Pi Zero W 11](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116419)

[Фигура 3 – Главна страница на Android приложение 12](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116420)

[Фигура 4 – Страница за проверка на връзката – Android приложение 12](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116421)

[Фигура 5 – Смарт 3Wi-Fi реле (Sonoff) 13](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116422)

[Фигура 6 – Електромагнитен вентил 13](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116423)

[Фигура 7 – DHT22 14](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116424)

[Фигура 8 – Сензор BMP280 14](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116425)

[Фигура 9 – Микроконтролер D1 mini 15](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116426)

[Фигура 10 – Сензор за влажност на почвата 15](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116427)

[Фигура 11 - Разработка на "Експеримент 1" 16](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116428)

[Фигура 12 – 5Cron файл (Експеримент 1) 17](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116429)

[Фигура 13 – Първа част от код на експеримент 1 17](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116430)

[Фигура 14 – Втора част от код на експеримент 1 17](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116431)

[Фигура 15 – Експеримент 2 18](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116432)

[Фигура 16 – Експеримент 3 19](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116433)

[Фигура 17 – Java 21](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116434)

[Фигура 18 – IntelliJ IDEA 22](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116435)

[Фигура 19 – Android Studio 23](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116436)

[Фигура 20 – Проста функция на Arduino софтуер 23](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116437)

[Фигура 21 – Графична схема на обектно ориентирана база от данни 25](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116438)

[Фигура 22 – PostgreSQL 25](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116439)

[Фигура 23 – GitHub 26](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116440)

[Фигура 24 – Spring Framework 26](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116441)

[Фигура 25 – Postman 27](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116442)

[Фигура 26 – PuTTY 27](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116443)

[Фигура 27 – FileZilla 27](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116444)

[Фигура 28 – VNC Viewer 28](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116445)

[Фигура 29 – Raspberry Pi Imager 28](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116446)

[Фигура 30 – Tasmota 29](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116447)

[Фигура 31 – ESP Easy 29](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116448)

[Фигура 32 – Raspberry Pi Zero W 30](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116449)

[Фигура 33 – ESP8266 WeMos D1 Mini 31](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116450)

[Фигура 34 – Sonoff Basic R2 DIY Smart Switch 31](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116451)

[Фигура 35 – Tork t-gp 104 32](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116452)

[Фигура 36 – DHT22 33](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116453)

[Фигура 37 – BMP280 33](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116454)

[Фигура 38 – Capacitive Soil Moisture Sensor v1.0 34](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116455)

[Фигура 39 – UART TTL FT232RL 34](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116456)

[Фигура 40 – Литиево-йонни батерии 35](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116457)

[Фигура 41 – Модул за зареждане на литиева батерия 35](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116458)

[Фигура 42 – Соларен панел 36](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116459)

[Фигура 43 – Библиотеки в Java 37](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116460)

[Фигура 44 – LocalDateConverter 38](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116461)

[Фигура 45 – ScheduledTasks 38](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116462)

[Фигура 46 – Първа част на Rest контролер 39](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116463)

[Фигура 47 – Втора част от Rest контролер 39](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116464)

[Фигура 48 – Трета част на Rest контролер 39](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116465)

[Фигура 49 – Първа част на Entity класовете 40](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116466)

[Фигура 50 – Втора част на Entity класа 41](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116467)

[Фигура 51 – Repository 41](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116468)

[Фигура 52 – Service 42](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116469)

[Фигура 53 – Първа част от Implementation класа 42](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116470)

[Фигура 54 – Втора част от Implementation класа 43](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116471)

[Фигура 55 – HTTP request and response 43](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116472)

[Фигура 56 – System Application 44](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116473)

[Фигура 57 – Структура на база данни 46](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116474)

[Фигура 58 – Таблица "Configuration" 46](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116475)

[Фигура 59 – Таблица "Valve" 47](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116476)

[Фигура 60 – Таблица "Watering\_hours" 47](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116477)

[Фигура 61 – Таблица "Users" 48](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116478)

[Фигура 62 – Таблица "Logs" 48](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116479)

[Фигура 63 – Таблица "Devices" 49](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116480)

[Фигура 64 – Таблица "Sensors" 49](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116481)

[Фигура 65 – Таблица "Sensor\_Types" 50](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116482)

[Фигура 66 – Таблица "Sensor\_Data" 50](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116483)

[Фигура 67 – Запояване на Sonoff Basic R2 DIY Smart Switch 51](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116484)

[Фигура 68 – Свързан електромагнитен вентил 51](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116485)

[Фигура 69 – Запояване на модул за зареждане на литиеви батерии 52](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116486)

[Фигура 70 – Процес на разработка на "Четвърти сегмент" 52](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116487)

[Фигура 71 – PuTTY сесия 53](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116488)

[Фигура 72 – Git clone команда 54](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116489)

[Фигура 73 – Команда за навигиране до папката с проекта 54](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116490)

[Фигура 74 – Команда за изпълняване на проекта 54](file:///C:\Users\ivi1sf3\Desktop\Поливна%20система%20-%20документация.docx#_Toc102116491)

# Използвана литература

## Печатни издания

1. Cay S. Horstmann. Core Java Volume 1-Fundamentals. 2018 г.
2. Robert C. Martin, Michael C. Feathers, Timothy R. Ottinger. Clean Code. A Handbook of Agile Software Craftsmanship. 2008 г.

## Интернет адреси

1. [Официална страница на Raspberry Pi](https://www.raspberrypi.org/)
2. [Официална страница на IntelliJ](https://www.jetbrains.com/idea/)
3. [Официална страница на Android Studio](https://developer.android.com/)
4. [Официална страница на Java Oracle](https://www.java.com/en/)
5. [Официална страница на Arduino](https://www.arduino.cc/)
6. [Официална страница на PostgreSQL](https://www.postgresql.org/)
7. [Официална страница на GitHub](https://github.com/)
8. [Официална страница на Spring Framework](https://spring.io/)
9. [Официална страница на Postman](https://www.postman.com/)
10. [Официална документация на PuTTY](https://documentation.help/PuTTY/)
11. [Официална документация на FileZilla](https://wiki.filezilla-project.org/Documentation)
12. [Официална страница на VNC Viewer](https://www.realvnc.com/en/)
13. [Официална страница на Raspberry Pi Imager](https://www.raspberrypi.com/software/)
14. [Официална документация на Tasmota](https://tasmota.github.io/docs/)
15. [Официална документация на ESP Easy](https://espeasy.readthedocs.io/en/latest/)
16. [Официална страница на Raspberry Pi Zero W](https://www.raspberrypi.com/products/raspberry-pi-zero-w/)
17. [Официална страница на ESP8266 Wemos D1 Mini](https://www.wemos.cc/en/latest/d1/d1_mini.html)
18. [Официална страница на Sonoff](https://sonoff.tech/)
19. [Официална страница на Sonoff Basic R2 DIY Smart Switch](https://sonoff.tech/product/diy-smart-switch/basicr2/)
20. [Официална страница на DHT22](https://learn.adafruit.com/dht)
21. [Официална страница на BMP280](https://www.bosch-sensortec.com/products/environmental-sensors/pressure-sensors/bmp280/)
22. [Официална страница на DFRobot](https://www.dfrobot.com/)
23. [Официална страница на Capacitive Soil Moisture Sensor V1.0](https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193)
24. [Официална документация на TP4056](https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf)
25. [Официална страница на Wikipedia](https://www.wikipedia.org/)
26. [Официална страница на Stack Overflow](https://stackoverflow.com/)
27. [Официална страница на W3Schools](https://www.w3schools.com/)
28. [Официална страница на JavaTpoint](https://www.javatpoint.com/)
29. [Официална страница на Baeldung](https://www.baeldung.com/)
30. [Официална страница на TutorialsPoint](https://www.tutorialspoint.com/index.htm)

## Полезни адреси

1. Публично хранилище на проекта – <https://github.com/Initaro/watering_system_spring>
2. Онлайн урок 1[1] - <https://desertbot.io/blog/headless-pi-zero-w-wifi-setup-windows>
3. Онлайт урок 2[2] – <https://phoenixnap.com/kb/install-java-raspberry-pi>
4. Онлайн урок 3[3] – <https://xianic.net/2015/02/21/installing-maven-on-the-raspberry-pi/>
5. Онлайн урок 4[4] – <https://www.youtube.com/watch?v=H6YUWkeSZNU>