Софийски университет "Св. Климент Охридски"

Факултет по математика и информатика

Учебна дисциплина „Компютърни системи за управление на роботи“

Курсов проект

на

Димо Апостолов, фак. № 62099

Камен Вакавчиев, фак. № 81712

Георги Маджаров, фак. № 81593

Тема: „Проектиране и разработване на Система за отглеждане на растения „

Научен ръководител

доц. д-р Симеон Цветанов

София 2020

Съдържание

[Увод 3](#_Toc36674193)

[1 Въведение 4](#_Toc36674194)

[2 Проектиране 6](#_Toc36674195)

[3 Разработване 14](#_Toc36674196)

[4 Експерименти и тестове 17](#_Toc36674197)

[Заключение 19](#_Toc36674198)

[Източници 20](#_Toc36674199)

[Приложение 20](#_Toc36674200)

# Увод

Целта на проекта е да се проектира и разработи система за автоматизирано поливане на растения която да улесни агрономи, бутинари и обикновените хора да отглеждат растения. С помощта на нашата система те ще знаят колко светлина и вода са необходими за растението, също така няма да се налага да го поливат ръчно. Системата сама ще засича дали растението има подходяща влага на почвата и при необходимост ще включи помпа която ще полее растението. Всички тези данни ще са достъпни от мобилно приложение. От приложението потребителя ще може да управлява необходимата влага на почвата, да види нивото на водата във резервоара на помпата и след колко дни ще трябва да напълни резервоара, ще може да наблюдава както текущи така и предишни стойности на светлина и влага на растението.

# Въведение

Отглеждането на растения не винаги е лесно. На някои хора им се отдава, а други въобще нямат „a green thumb“. Факт е че дори и да ти се отдава не винаги можеш да предложиш оптималните условия на растенията. Може би не разбираш нищо от отглеждането на растения, може би се опитваш да си отглеждаш градина на село докато живееш в града и да нямаш възможността да ходиш всяка вечер след работа да ги огледаш и да ги полееш. Или пък заминаваш на почивка и няма да можеш да се грижиш активно за растенията. Дори може да си по-възрастен и да нямаш силите да разнасяш тежка кофа с вода и канче. Няма значение каква е твоята ситуация, важното е че проблема винаги се свежда до невъзможността да се грижиш адекватно за флората която си избрал да отглеждаш. Нашият проект би помогнал с това затруднение. Той може да те отмени в няколко от най-основните дейности необходими за грижата за растенията.

Първото нещо което човек се сеща, когато го попитат как би отглеждал едно растение, е да го полива редовно. Нашият проект помага точно със тази дейност. Използваме датчик за влага на почвата който засича кога растението трябва да се полее. Ако засече прекалено ниска влага, се включва помпа която взима вода от резервоар и полива растението. Естествено всяко растение е различно и изисква различна влага на почвата. За това сме направили мобилно приложение от което потребителя да контролира от колко вода се нуждае растението, по този начин той променя границата при която сензора ще каже на помпата да се включи. По този начин премахваме необходимостта от ръчно поливане на растенията.

Но това заменя непрекъснатото наблюдение дали на растението му е необходимо вода с непрекъснатото наблюдение на нивото на оставащата вода в резервоара. Този проблем беше решен като сложихме датчик за дистанция на капака на резервоара. Той отчита нивото на водата и информацията се показва във реално време на приложението. Също така показва и за колко време ще стигне оставащата вода. Преди да се пусне в употреба трябва датчика да се постави на капака на празен съд, който ще служи за резервоар, за да се направи първоначално засичане на дълбочината на съда.

Освен по влаголюбивост, растенията се делят и по светлолюбивост. С този аспект от градинарството се справихме като добавихме сензор за светлина. Прочетените от него стойности можем да видим в приложението.Така ако забележим че растението не се справя много добре , ще знаем че се нуждае от повече слънчева светлина и ще го преместим.

Споменахме по-горе за мобилно приложени което е част от системата. Това приложение има два „екрана“: Now и Statistics. На екрана Now може да се наблюдават текущите измервания на всички датчици в проценти:

каква е влажността на почвата, колко вода има в резервоара и каква светлина се засича от датчика за светлина. Освен това (както споменахме по-горе) има бутон UPDATE с който можем да определим thresh hold при който ще се включи помпата. На екрана Statistics има drop down меню от което потребителя може да избере число от 1 до 10. Тези стойности са за колко дни назад да се включват отчетените стойности от датчиците в статистиката. След като се избере броя дни можем да видим средно аритметичните стойности за влажността на почвата, колко вода е използвана дневно и колко слънчево е било. Също така показва за колко дни ще стигне оставащата вода в резервоара.

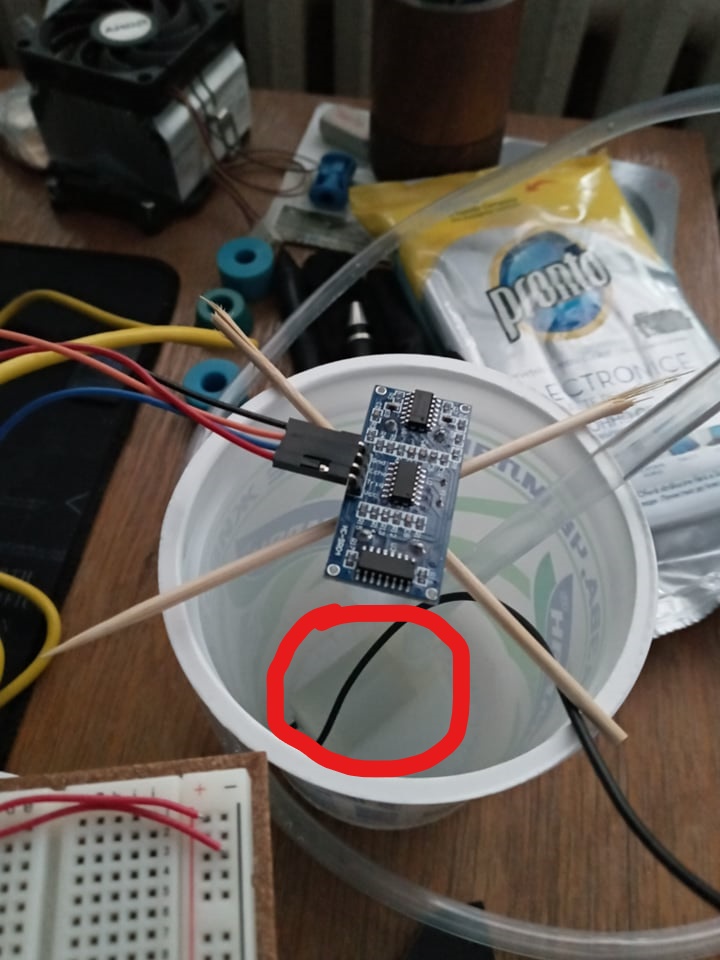
# Проектиране

Идеята ни дойде от желанието ни да улесни хората които не могат да се справят сами с отглеждането както на къщни така и на градински растения.

Започнахме само с идеята за поливане на растенията. Решението на този проблем беше лесно. Свързахме Arduino UNO с датчик за влажност на почватa и реле за управление на помпа за вода.

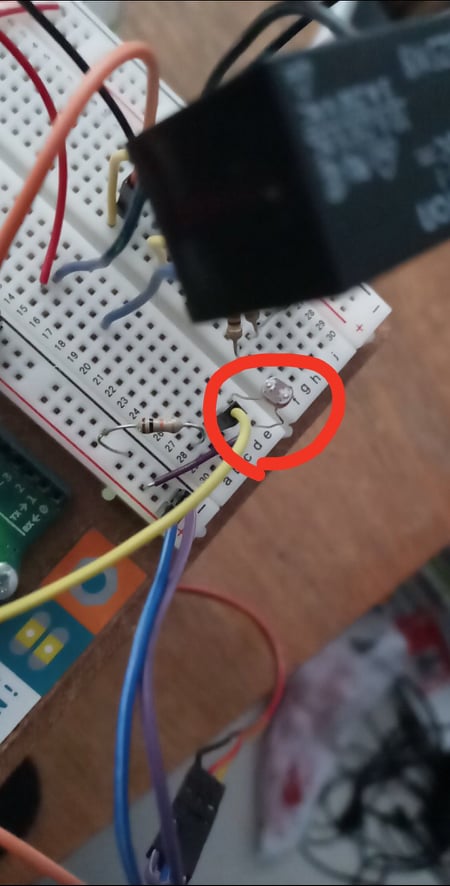


Естествено, всяко растение е различно и изисква различно ниво на влага на почвата. Заради това проектирахме приложение от което да се контролира thresh hold за поливане.



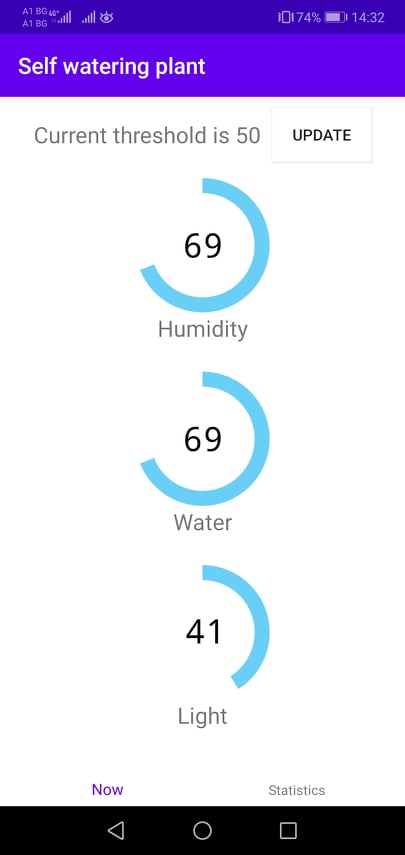
Но то въведе нов въпрос, а именно следенето на водата в резервоара. Решихме този проблем като добавихме сензор за дистанция който ще отчита разстоянието от себе си до водата. Но тогава се запитахме, дали винаги ще използваме резервоари с едни и същи височини, за това добавихме калибрация при първо пускане на системата която изисква резервоара да бъде празен при пускането на системата. Добавихме възможността да се изчислява за колко дни ще ни стигне оставащата вода в резервоара.

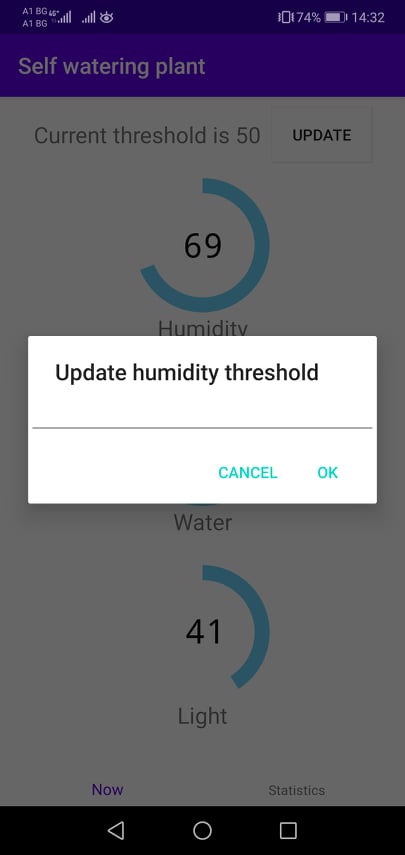
Освен по влаголюбивост, растенията се различават и по своите изисквания относно светлината. Поради таза причина решихме да добавим датчик за светлина.



Така можем да следим колко светлина поема растението и дали трябва да го преместим на по-светло или на по-тъмно място.

И понеже трябва да се следи прогреса на растението решихме в приложението да показваме информация както за текущите стойности така и средните стойности за максимум 10 дни назад.





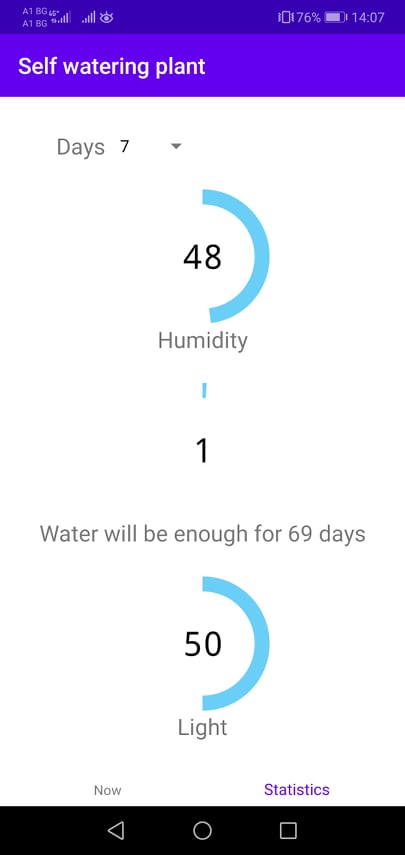
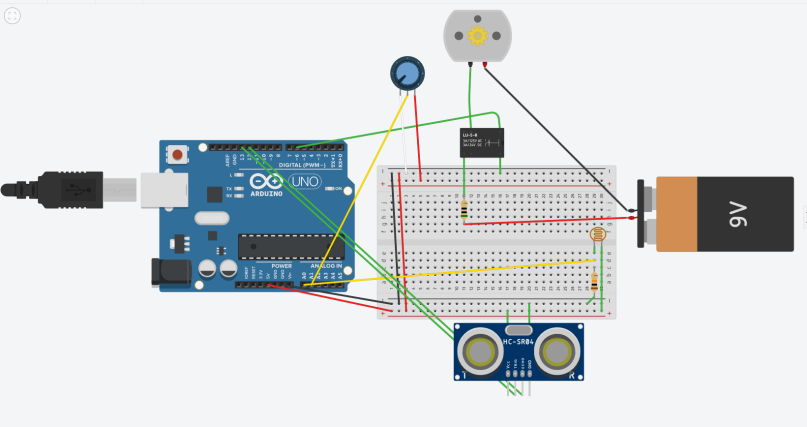


Схема на системата:



Фиг 1

Сензора за управление на мотор е свързан с реле което свързва мотора с 9V батерия. Използваме допълнителна батерия понеже мощността на Arduino-то не е достатъчна да захрани мотора.

Сензора за влага и сензора за светлина са директно вързан за аналоговия вход на Arduino-то.

Ултразвуковият сензора за разстояние (дълбочина), който използва 2 пинчета, 1 за четене(echo) и 1 за писане(trig). То излъчва ултра звукова вълна с едната „колонка“ и гледа за колко време ще стигне до другата. И на база времето изчислява разстоянието.



# Разработване

За разработването на сървърната част сме използвали Spring boot като е имплементиран MVC архитектурен шаблон. Самото приложение е качено на cloud платформата Heroku. Heroku ни предоставя URL , спрямо който сме дефинирали API за комуникацията на сървъра с останлите части на проекта. Базата от данни, която ползваме за съхранение на информация е postgresql като се възползваме от готов add-on на Heroku, който дава възможност за съхранение на 10000 реда в таблици.

Сега малко по-подробно за самата реализация на сървъра и необходимите конфигурации, за да заработят успешно нещата заедно. В Spring изполваме следното dependency- spring-boot-starter-data-jpa и по-точно няколко негови анотации - @Entity, @Table ( оказва, че за даденият POJO клас съществува таблица в базата от данни ) и още няколко други като @Id, @Column, @GeneratedValue , чрез които задаваме какви да са имената на колоните, кое е ид-то и ключа в таблицата и каква е стратегията за неговото генериране. Имаме интерфейс IDataRepository, който наследява интерфейс отново от JPA – JpaRepository , от който получаваме наготово основни операции като save, findAll и други за таблицата получена от горе дефинираният клас Data в базата данни. Също така сме написали и няколко допълнителни метода към това repository като getMostCurrentData , който ни връща най-близкото по дата entry в базата, getAvgHumidity(Integer days), който ни връща средната стойност на влажност в почвата за определен брой дни - days назад от текущата дата и още няколко метода. След това имаме един layer от service клас, който има чрез dependency injection(@Autowired) „инжектирано“ това repository и който service просто извиква методите на repository-то. Накрая имаме rest controller , в който сме определели API-то за комуникация със сървъра. На самият controller имаме mapping „/data“, който ще бъде сложен пред всеки един от следните методи:

• @GetMapping()

public List<DataDTO> getAllData()

• @GetMapping(value = "/currentData")

public DataDTO getCurrentData()

• @GetMapping(value = "/currentThreshold")

public Integer getCurrentThreshold()

• @GetMapping(value = "/statistics/{days}")

public StatisticsDTO getStatistics(@PathVariable("days") Integer days)

• @PostMapping(value = "/add", consumes = MediaType.APPLICATION\_JSON\_VALUE)

public void addData(@RequestBody DataDTO dataDTO)

• @GetMapping(value = "/updateThreshold/{threshold}")

public boolean updateThreshold(@PathVariable("threshold") Integer threshold)

Това са всички методи или API за комуникация на останлите части от проекти чрез сървъра и базата от данни. Техните имена са достатъчно описателни и лесно се разбира, чрез кой mapping какво можем да направим.

За да успее Spring да разбере по-какъв начин да използва вградената в Heroku postgresql база е нужно да направим малко конфигурация. Първо трябва да си добавим dependency за postgresql driver. След това си създаваме проект в Heroku и в сайта отваряме настройките на проекта и ще дефинираме няколко Config Vars :

• SPRING\_DATASOURCE\_URL = jdbc:postgresql://YourPostgresHerokuHostNameHere:5432/YourPostgresHerokuDatabaseNameHere

• SPRING\_DATASOURCE\_USERNAME = YourPostgresHerokuUserNameHere

• SPRING\_DATASOURCE\_PASSWORD = YourPostgresHerokuPasswordHere

• SPRING\_DATASOURCE\_DRIVER-CLASS-NAME = org.postgresql.Driver

• SPRING\_JPA\_DATABASE-PLATFORM = org.hibernate.dialect.PostgreSQLDialect

• SPRING\_JPA\_HIBERNATE\_DDL-AUTO =u pdate

Необходимите данни за инциализацията на променливите можем да открием например от data.heroku.com. След това deploy-ваме кода на Spring приложението на Heroku и с това работата по сървъра приключва.

За реализирането на android сме използваме отново java. За комуникация чрез сървъра сме използвали retrofit2, като отново имаме един дефиниран интерфейс WebService със следните методи:

• @GET("/data")

Call<List<Data>> getData()

• @GET("/data/currentData")

Call<Data> getMostCurrentData()

• @GET("/data/statistics/{days}")

Call<Statistics> getStatistics(@Path("days") Integer days)

• @GET("/data/updateThreshold/{threshold}")

Call<Boolean> updateThreshold(@Path("threshold") Integer threshold)

И след това чрез Builder design pattern си построяваме Retrofit обект, който използваме за да си направим инстанция на WebService, чрез която правим call-ове към сървъра :

new Retrofit.Builder()

.baseUrl("https://mysterious-brushlands-97617.herokuapp.com/")

.addConverterFactory(GsonConverterFactory.create())

.build()

webService = retrofit.create(WebService.class);

Приложението има два екрана, един, който показва данните в текущо време и още един показващ статистика за определен брой дни. За преминаването от единия екран към другия се използва Navigation, създали сме граф с тези два екрана и връзка между тях, като имаме BottomNavigationView с два бутона now и statistics и при натискането на някой от двата бутона се прехвърляме съответно на този екран.

За показване на стойностите влажност, оставаща вода в резорвара и светлина сме използвали външна библиотека, която е добавена към gradle файла така:

implementation 'com.akexorcist:RoundCornerProgressBar:2.0.3'

От нея използваме CircleProgressView, което представлява кръг, който се запълва по посока часовниковата стрелка толкова, колкото е зададената стойност, например за зададена стойност 50, ще бъде запълнен на 50 процента или половината.

Имаме възможност и да променяме желаният threshold на влага, което е реализирано чрез Dialog. В този диалог задаваме нова стойност на threshold, след което тя се изпраща към сървъра и при успешно обновяване на стойността се взима отново цялата информация от сървъра, в случай, че нещо е променено от преди нашата промяна.

# Експерименти и тестове

Тестови сценарии, Анализ на резултатите.

Първи тест:

- при дадено стайно растение в саксия

- първоначалната влага почвата 57%

- зададен thresh hold 69% влага

- напоява се за 10 сек.

Втори тест:

- при дадено стайно растение в саксия

- първоначалната влага почвата 45%

- зададен thresh hold 60% влага

- преди достигане на този thresh hold, променяме го на 50%

- помпата спира на 50% вместо да продължи до 60%

Трети тест:

* при резервоар 1л. с цилиндър висок 10см.
* с оставаща вода на височина 7см.
* среден статистически разход от 100мл. на ден
* изчислява че ще стигне за 7 дни.

Четвърти тест:

- при дадено стайно растение в саксия

- поставено на място с 23% светлина

- редовно поливано за подходящия за него thresh hold

- то все още увяхва

- след като го преместихме в стая с 48% светлина

- то се управи

Специфика на хардуера:

-Arduino Uno

-HC-SR04

-фото резистор

-Сензор за влага версия 1.2 (1)

- OMRON G5LE-1 5VDC

- Смартфон с Android 6.0+

# Заключение

Успяхме да създадем система която да помага в отглеждането на растения и да следи всичките релевантни показатели свързани със тази дейност. Резултатите са обещаващи, но винаги има място за подобрение.

Относно бъдещото развитие на проекта, има няколко идеи които бихме имплементирали.

1. бихме заменили Arduinoto с ESP.
2. бихме добавили на сензор за топлина. Така бихме могли да следим по-добре климатичните обстоятелства които влияят на растенията.
3. Бързо и лесно добавяне на нови растения и възможността да има идивидуален профил за всяко растение

# Източници

Препоръчително е да има референция към източника в текста.

Фиг 1 - [www.tinkercad.com](http://www.tinkercad.com)

<https://stackoverflow.com/questions/33633243/connecting-to-heroku-postgres-from-spring-boot>

<https://devcenter.heroku.com/articles/deploying-spring-boot-apps-to-heroku>

(2) <https://www.aliexpress.com/item/32886868425.html?mb=nOyGgrVeDPOw4lq&srcSns=Copy&tid=white_backgroup_101&tt=sns_Copy&image=H4990113857f9444b93882d7422bb2f1by.jpg&aff_request_id=cd81ea6052a34028a740a2f24488bf80-1592253598059-07217-_d7bniOm&spreadType=socialShare&aff_platform=default&sk=_d7bniOm&aff_trace_key=cd81ea6052a34028a740a2f24488bf80-1592253598059-07217-_d7bniOm&businessType=ProductDetail&templateId=white_backgroup_101&title=BGN+1.12++16%25OFF+%7C+Analog+Capacitive+Soil+Moisture+Sensor+V1.2+3.3%7E5.5V+Corrosion+Resistant+With+Gravity+3-Pin+Interface+for+Arduino+Raspberry&platform=AE&terminal_id=5a3a18695d3b4d6aa064e1b8f3a38ce6>

Консултация с агрономи.

# Приложение

Само и единствено за отглеждане на растения. Както от любители, така и от градинари.