Факултет инжењерских наука Универзитета у Крагујевцу

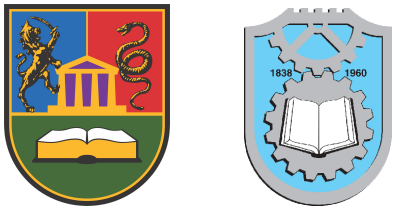
Софија М. Анђелковић

Биолошки инкубатор са мобилном апликацијом за мониторинг параметра

завршни рад

Крагујевац, септембар 2019.

Факултет инжењерских наука Универзитета у Крагујевцу



Назив студијског програма: Рачунарска техника и софтверско инжењерство
Ниво студија: Основне академске студије
Предмет: Програмирање мобилних апликација
Број индекса: 565/2015

Софија М. Анђелковић

Биолошки инкубатор са мобилном апликацијом за мониторинг параметра

завршни рад

Комисија за преглед и одбрану:

1. **Проф. Др Ненад Грујовић - ментор**
2. _____
3. _____

Датум одбране: _____

Оцена: _____

У оквиру овог завршног рада кандидат треба да:

- објасни принцип рада и примену биолошких инкубатора
- објасни и реализује подсистем биолошког инкубатора за праћење одговарајућих параметра (температура, влажност, CO2)
- изради мобилну апликацију за Андроид уређаје која омогућава мониторинг параметра реалног биолошког инкубатора (температура, влажност, CO2)

Препоручена литетартура:

[1] James Steele, Nelson To: Андроид: Израда апликација помоћу пакета Андроид SDK, Микро књига, Београд, 2011.

[2] James Talbot, Justin McLean: Програмирање Андроид апликација; CET, 2014.

[3] Prof. Dr Dogan Ibrahim i Ahmet Ibrahim: ESP8266 i MicroPython; EHO, 2017.

[4] Lab Manager, CO2 Incubators: Mayinstays of Cell or Tissue Culture

Крагујевац, 2019. ментор:

др Ненад Грујовић, редовни професор

Резиме рада:

У овом раду је представљен система за мерење параметра чији се ниво контролише у биолошким инкубаторима. Систем за мерење комуницира са мобилном апликацијом која омогућава мониторинг параметра биолошког инкубатора (температура, релативна влажност, угљен диоксид). Хардверски део пројекта чине сензори и развојна плоча са интегрисаним WiFi модулом, чиме је омогућена бежична комуникација у оквиру апликације. Читање вредности са сензора реализовано је помоћу Arduino развојног окружења који податке преко WiFi мреже шаље на сервер. Серверски део је реализован помоћу WAMP софтверског пакета и PHP програмског језика. Улога мобилне апликације је да учитава податке са сервера и у реалном времену приказује вредности измерене од стране сензора.

Кључне речи:

CO2 инкубатор, угљен диоксид, температура, релативна влажност, Arduino, Android, мобилна апликација, WeMos D1 mini, ESP8266, DHT22 сензор, MH-Z19b модул

Summary of work:

In this thesis system for monitoring CO2 incubator parameters is presented. The measuring system is connected to the mobile application which will monitor those parameters. The hardware is composed of sensors and a development board with an integrated WiFi module which establishes the wireless connection. The reading of the parameters from the sensors occurs with the help of Arduino development surrounding which sends the data to the server via WiFi. The server is comprised of the WAMP software pack and the PHP programming language. The role of the mobile application is to display the data from the server and to show the values measured by the sensors in real time.

Key words:

CO2 incubator, carbon dioxide, temperature, relative humidity, Arduino, Android, mobile application, WeMos D1 mini, ESP8266, DHT22 sensor, MH-Z19b module

Садржај

1. УВОД

Сведоци смо великог развоја на пољу науке, посебно кад је у питању технологија и инжењерство. Применом инжењерских принципа у биологији настала је нова област - синтетичка биологија. Нова област настала је спајањем више дисциплина из области биологије, хемије, рачунарства и инжењерства. Ова интердисциплинарна област се може замислити као "алат" заснован на биологији који користи апстракцију, стандардизацију и аутоматску конструкцију како би променио начин на који израђују биолошке систем и проширио спектар могућих производа. Синтетичка биологија фокусирана је на стварање технологија за пројектовање и изградњу организама или активних биолошких структура. З Група стручњака из више области фокусирана је на проучавање функционисања генетских делова, а затим њиховим комбиновањем произвела корисне апликације. Напретком синтетичке биологије и разумевањем како различити делови генома функционишу, омогућена је производња низа потрошачких производа, од горива до козметике.

У медицини, синтетичка биологија помера границе у лечењу рака, дизајнирајући микробе који ће тражити и уништити туморе у телу пре него што дође до њиховог деградирања. Синтетичка биологија такође помаже у разумевању особина грипа и производњи вакцина. Синтетичка биологија узима учешће и у производњи хране. Синтетичка биологија узима учешће и у производњи хране. На МИТ универзитету истраживачи активно раде на процесу који ће омогућити биљкама везивање азота, што би омогућило смањивање коришћења вештачких ђубрива у пољопривреди.

Могућности синтетичке биологије су велике и њена примена ће у будућности бити све већа. Проблем настаје јер се цела област ослања на компликоване и веома скупе инструменте и машине. То практично онемогућава свима изван лабораторије универзитета или компанија да имају приступ овој технологији. Због тога је настао друштвени биотехнолошки покрет "**DIYbio**" (*Do It Yourself biolog*y - Уради сам биологија) у којем појединци, заједница и мале организације проучавају биологију и науке о животној средини користећи исте методе као и истраживачке институције. Циљ покрета је да алати и ресурси за обављање биолошког инжењеринга буду доступни свима, а не само професионалцима, чиме би се убрзао напредак у оквиру ових области. Део самог покрета је и синтетичка биологија, која постаје све више заступљена.

Као што је поменуто раније, највећи проблем је неприступачност инструмената потребних за рад људима који би вршили истраживања и реализацију биотехнолошких пројеката из хобија. У оквиру овог рада је реч о једном таквом инструменту - биолошком инкубатору. Основна идеја је уређај сличних перформанси као у професионалним лабораторијама, уз редукцију трошкова производње.

Биолошки инкубатори се генерално користе за узгој ћелија у пажљиво контролисаним условима, као што су одговарајућа температура, релативна влажност и концентрација угљен диоксида. Цео систем се може поделити на два подсистема. Један подсистем чини група сензора за мерење одређених параметра битних за правилан раст ћелија и њихов мониторинг, док други подсистем чини аутоматизован систем за одржавање идеалних услова. У оквиру овог рада пројектован је и описан подсистем за мерење параметра и њихов мониторинг.

Циљ пројекта је израда подсистема који мери температуру, релативну влажност и концентрацију угљен диоксида који би се могао касније имплементирати у целокупни биолошки инкубатор. Уз то, циљ је направити и мобилну апликацију за мониторинг тих параметра. Комуникација између сензора и апликације је бежична и то WiFi, чиме се омогућава мониторинг параметра и удаљеним приступом. Инсталирање апликације омогућава кориснику праћење параметара унутар инкубатора у реалном времену.

У оквиру рада ће бити детаљно описан развој и имплементација андроид апликације за мониторинг параметра у биолошком инкубатору. Осим тога, на самом почетку биће реч и о томе шта су биолошки инкубатори и шта је њихова сврха.

За мерење температуре и релативне влажности коришћен је **DHT22** модул, док је за мерење концентрације угљен диоксида коришћен **MH-Z19b NDIR** модул. За развојну плочу изабрана је **WeMos D1 mini** која је базирана на **ESP8266EX WiFi** модулу помоћу које је реализована бежична комуникација.

2. БИОЛОШКИ ИНКУБАТОР

Инкубатори представљају изолована окружења у којима се температура, влажности и други услови у окружењу могу одржавати на одређеном оптималном нивоу сходно намени. Инкубатори су пронађени још пре пар хиљада година, постојали су још у древном Египту и Кини. У међувремену технологија је напредовала, али њихова главна улога је остала непромењена: створити стабилно, контролисано окружење погодно за истраживање, проучавање и узгој. Постоје три основна типа инкубатора, а то су: инкубатори за живину, инкубатори за бебе и биолошки инкубатори. Проучавање и пројектовање у оквиру овог рада односи се на биолошке инкубаторе [1].

Биолошки инкубатор (у даљем тексту инкубатор) обезбеђује контролисано окружење како би се поспешио раст микроорганизама и ћелијских култура. У медицини се овај тип инкубатора користи за идентификацију микроорганизама који узрокују болест како би се тачно утврдио узрок болести пацијента. Узорак се поставља у медијум за раст унутар инкубатора како би се умножио, јер се такав умножени узорак може лакше и са већом сигурношћу идентификовати. Инкубатори су неопходни за извођење експеримената у областима биологије, микробиологије, биохемије итд. Осим тога, користе се и у млечној и прехрамбеној индустрији, као и у постројењима за пречишћавање воде и канализације [1].

Инкубатор је основа за медицински напредак и експериментални рад у ћелијској и молекуларној биологији. Најједноставнији инкубатори омогућавају одржавање константне температуре, док неки сложенији инкубатори омогућавају контролисање и одржавање и других параметра, попут влаге, концентрације угљен диоксида, па чак и концентрације кисеоника и азота. Температура је посебно битан фактор јер живи организми су врло осетљиви на температурне промене, па је приликом испитивања битно да се одржи температурна стабилност и хомогеност, како би резултати били валидни [2].

Инкубатори у оквиру којих се контролише само температура су више намењени за микробиологију и организме за чији раст и правилан развој није битна pH вредност средине. Када је реч о узгајању ћелија и ткива, контрола влажности и киселости средине је од великог значаја. Како би се осигурала одређена киселост средине, потребно је контролисати концентрацију угељен диоксида (CO_2). У том случају се користи **CO_2 инкубатор** чији су принципи описани и имплементирани у оквиру овог рада.

2.1. CO2 инкубатор

Улога CO2 инкубатора је да одржава оптимално окружење за раст ћелија уз одређену концентрацију угљен диоксида и то у влажној атмосфери са константном температуром. Најчешће се користе у медицинским истраживањима и фармацеутској индустрији. CO2 инкубатор обезбеђује стабилно окружење дизјнирано да опонаша природно окружење ћелије: pH вредност од 7.2 до 7.4, температура од 37℃ и релативна влажност од око 95%. Концентрација CO2, око 5%, контролише се тако да одговара физиолошким условима и одржава се стални pH [3].

Слика 1 - *Професионални CO2 инкубатор* [5]

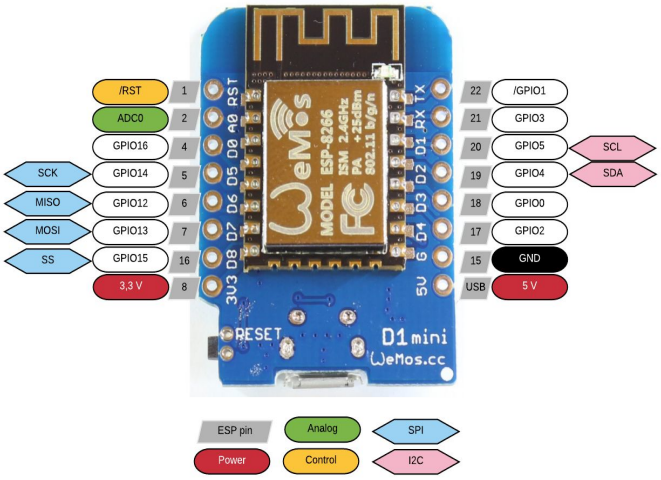
CO2 инкубатор има унутрашњност која је потпуно запечаћена и изолована од околине, како би се осигурало да на атмосферу унутар инкубатора ни на који начин не утичу спољни фактори. Изазов у пројектовању лежи у стварању хомогености кроз целу унутрашњост, како би се осигурало равномерно снабдевање угљен диоксид свих узорака у условима константне температуре и влаге [4].

3. КОМПОНЕНТЕ И СОФТВЕР КОРИШЋЕНИ У РАДУ

3.1. WeMos D1 mini WiFi микроконтролер

WeMos D1 mini део је породице WeMos D1 развојних плоча на којима се налази популарни **ESP8266 WiFi** чип. Овај уређеај је веома компактан за израду малих паметних уређаја повезаних на светску мрежу (енгл. World Wide Web), захваљујући Espressif ESP8266 WiFi функционалностима. D1 mini има веома једноставно подешавање и користи развојна окружења која су позната свим Арудино корисницима. Једноставнији је од осталих ESP модула, јер не захтева никаво флешовање ни програмирање пре самог коришћења. На плочи је интегрисан USB-UART, па је за сам почетак његовог програмирања довољан USB кабал.

Слика 2 - *WeMos D1 Mini*



Слика 3 - *WeMos D1 Mini Pins*

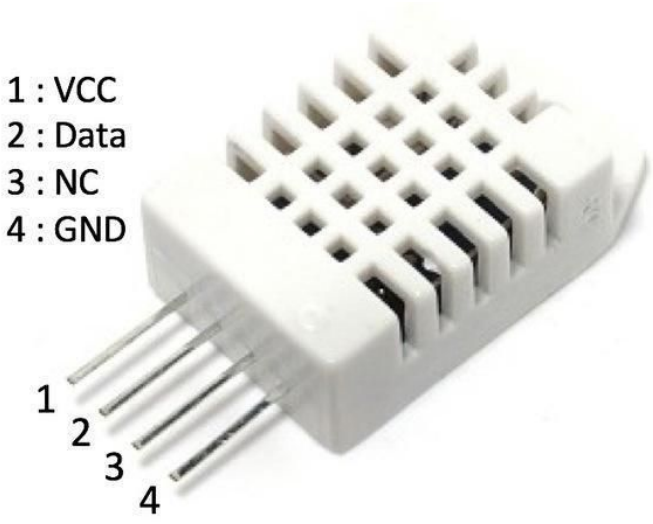
WeMos D1 Mini је базиран на 32-битном RISC контролеру који се напаја са 3.3V (евентуално 5 V уколико је напајање са рачунара). Сви пинови раде на логичком нивоу од 3.3V, осим једног пина који може да произведе 5V. Ради на 80MHz и у питању је пуни WiFi примопредајник. Садржи 4MB flash меморије и управо је то велика предност ове развојне плоче. Компатибилан је са Ардуином, MicroPython-ом и NodeMCU-ом [6].

Техничка спецификација:

- Микроконтролер: ESP8266EX
- Процесор: RISC (32-битни)
- Радни напон: 3.3V
- Број дигиталних И/У пинова: 11 (са подршком за I2C, interrupt, pwm)
- Број аналогних улазних пинова: 1 (максималан улаз: 3.2V)
- Flash меморија: 4MB
- Clock Speed: 80MHz/160MHz
- Микро USB конекција
- Димензије: 34mm x 25mm

3.2. Температурни сензор DHT22

DHT22 је веома заступљен сензор за мерење температуре и релативне влажности. Користи капацитативни сензор влажности за мерење релативне влажности и термистор негативног температурног коефицијента за мерење температуре. У оквиру сензора се извршава аналогно-дигитална конверзија и на излаз сензора се шаље дигитални сигнал. Сензор је фабрички калабрисан, тако да није потребна никаква припрема за коришћење. Довољно је повезати га са микроконтролером и читати вредности. Читање вредности се врши на сваке две секунде, па самим тим вредности могу да буду старе до две секунде [8].



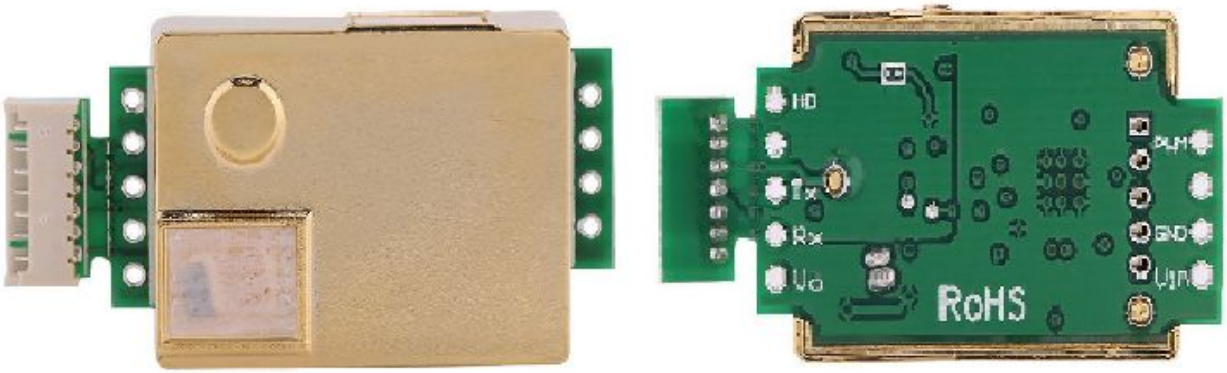
Слика 4 - *DHT22 сензор*

Техничка спецификација:

- Напајање: 3.3V - 6.0V
- Излаз: дигитални сигнал
- Опсег мерења:
    - Температура: -40-80℃
    - Релативна влажност: 0-100% RH
- Тачност мерења:
    - Температура: < ± 0.5℃
    - Релативна влажност: ±2%
- Потрошња струје за време мерења: 1 - 1.5 mA
- Потрошња струје за време мировања: 40 - 50μA
- Димензије: 14mm x 18mm x 5.5mm

3.3. Модул за мерење концентрације угљен диоксида MH-Z19b

MH-Z19b NDIR је инфрацрвени модул за гас који користи недисперзивне инфрацрвене принципе (NDIR) за детекцију количине угљен-диоксида у окружењу. Одликује га добра стабилност, дуг живот, као и то да не зависи од кисеоника. Комора модула је позлаћена, водоотпорна и антикорозивна, па самим тим може да ради у условима високе влаге без икаквих проблема. Добро компензује температуру и даје одличан линеарни излаз. Користи се за праћење квалитета ваздуха у просторијама, за паметне куће, вентилацијске ситеме, као и за уређаје за пречишћавање ваздуха [10].



Слика 5 - *MH-Z19b модул*

Недисперзивни инфрацрвени сензор је релативно једноставан спектроскопски сензор који се често користи за детекцију гаса. Није дисперзиван у смислу оптичке дисперзије, јер је инфрацрвеној енергији дозвољено да прође кроз комору атмосферских узорака без деформација. Главна компонента сензора је инфрацрвени извор, у овом случају лампица, комора за узорке или светлосна цев, светлосни филтер и инфрацрвени детектор. Инфрацрвена светлост је усмерена кроз комору за узорке ка детектору. Паралелно постоји још једна комора са референтним гасом, обично азотом. Гас у комори за узорке врши апсорпцију специфичних таласних дужина, а пригушење ових таласних дужина мери се детектором за одређивање концентрације гаса. Детектор има оптички филтер који елиминише сву светлост осим таласне дужине коју одабрани молекули гаса могу да апсорбују [9].

Пре почетка коришћења MH-Z19b модула, потребно је извршити калибрацију нулте тачке. Нулта тачка је 400 ppm (енгл. *parts-per million*), што је 0.04%. Постоје три начина да се то учини. Први начин је хардверски и то повезивање HD пина на низак логички ниво (0V) у трајању од најмање седам секунди. Други начин је софтверски, слањем команде. Трећи начин је самостална аутоматска калибрација. Уколико је модул активан 24 часа, врши се аутоматска калибрација нулте тачке узимањем најниже измерене вредности као нулте тачке. Тај процес се понавља на свака 24 сата. Аутоматска калибрација је погодна за канцеларисјко и кућно окружење, али не и за стакленике, фарме, фрижидере итд., па је у том случају потребно ручно искључити ову функцију [10].

Техничка спецификација:

- Радни напон: 4.5V - 5.5V
- Просечна струја: < 20mA
- Максимална струја: 150mA
- Напонски ниво интерфејса: 3.3V (компатибилан са 5V)
- Излазни сигнал: UART (серијски), PWM
- Време загревања: 3 минута
- Радна температура: 0 - 50℃
- Радна влажност: 0 - 95% RH
- Маса: 5g
- Животни век: више од 5 година

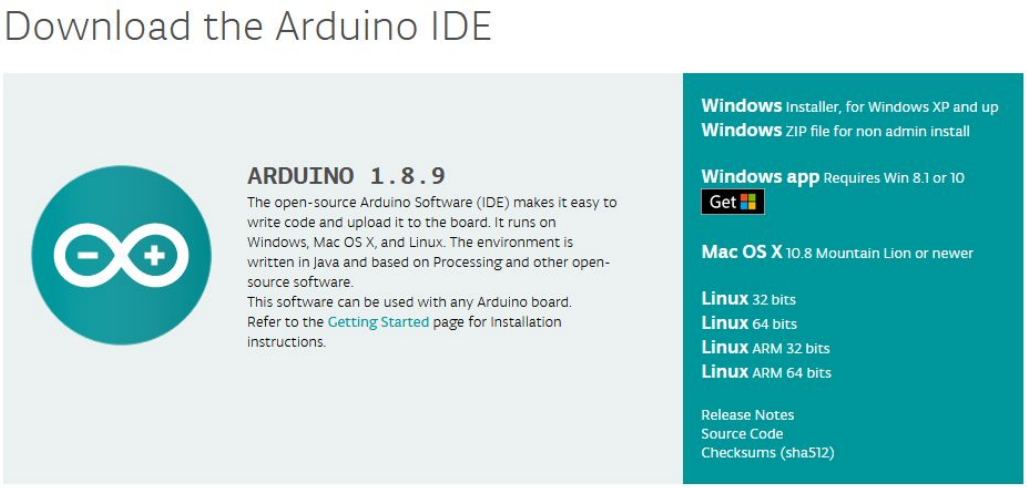
MH-Z19b модул има два различита излазна режима. Један је PWM (*Pulse Width Modulation*), док је други UART. Опсег мерења код PWM режима је од 0 до 2000 ppm, док је код UART режима од 0 до 5000 ppm, уз тачност од око ±50 ppm. Мерење извршава на сваких пет секунди [10].

3.4. Arduino IDE развојно окружење

Ардуино интегрисано развојно окружење (енгл. **Arduino IDE**) је вишеплатформска апликација која се користи за писање, компајлирање и слање кода на Ардуино уређаје, као и друге уређаје компатибилне са Ардуином. Развој Ардуино развојног окружења је везан за програмски алат под називом Processing, од којег је преузео мноштво ствари. Processing је платформа која омогућава писање апликација на поједностављеној верзији програмског језика Java**.** Основни програмски језик који се користи на платформи Ардуино је поједностављена варијанта језика C. Из језика су склоњене неке комплексније ствари и уведена је C/C++ библиотека звана Wiring**,** која је преузета из поменуте платформе Processing. Предности Ардуино софтвера се огледају у томе што представња отворену платформу (енгл. open source) и што програмирање хардверских уређаја чини доста лакшим [11].

3.4.1. Инсталација

Инсталирање Ардуино развојног окружења је једноставно. Последња верзија софтвера се може наћи на званичном сајту Ардуина у делу за преузимање софтвера. На страници су приказане све платформе на којима се може инсталирати Ардуино софтвер, треба изабрати одговарајућу и у пар корака обавити инсталацију. У случају коришћења платформи Windows 8 или Windows 10, инсталација се може наћи у оквиру Microsoft Store-a.



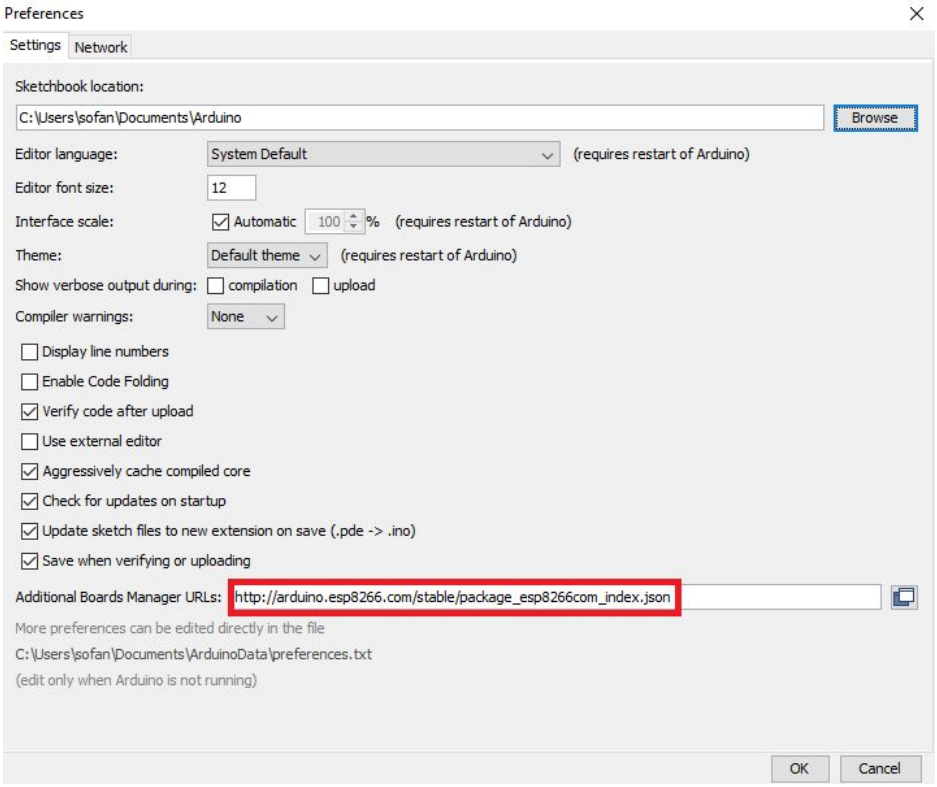
Слика 6 - *Инсталација Arduino IDE софтвера*

Након инсталирања, потребно је обавити подешавања радног окружења у складу са хардвером који се програмира.

3.4.2. Подешавање радног окружења

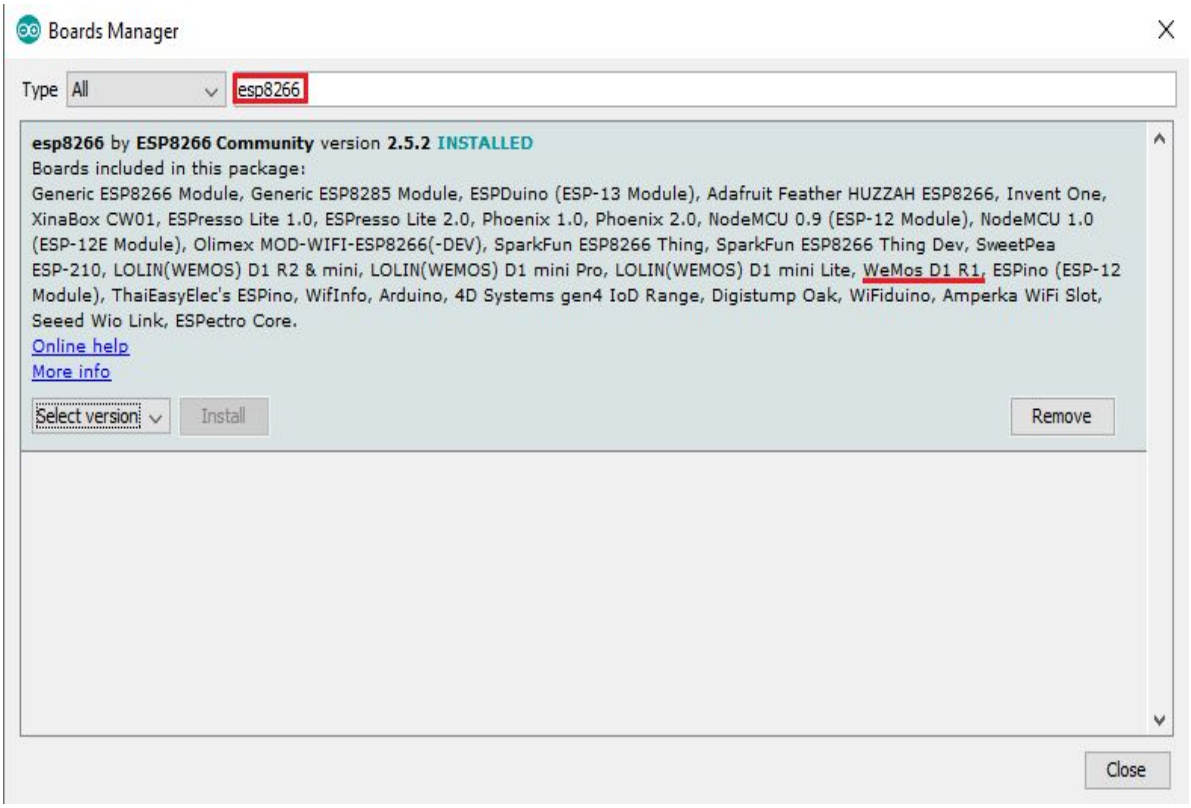
WeMos D1 mini развојна плоча је компатибилна са Ардуино развојним окружењем, али су за њено програмирање потребна одређена подешавања. За почетак је потребно покренути Ардуино софтвер и направити нови пројекат. Након тога је потребно направити нека подешавања у оквиру **Preferences** прозор и то одељак **Settings**. Прозор се отвара кликом на **File->Preferences**. Када се отвори прозор потребно је у део **Additional Boards Manager URL*s*** копирати следећи линк [7]:

http://arduino.esp8266.com/stable/package_esp8266com_index.json



Слика 7 - *Preferences прозор*

Када је линк додат потребно је инсталирати пакете за одговарајућу развојну плочу. У овом случају је то **WeMos D1 mini.** То се може учинити тако што се у **Boards Maganer**-u пронађе и инсталира адекватан пакет. Налази се помоћу кључне речи **esp8266** што представља назив породице чипова из које је чип на горе поменутој развојној плочи. На слици 8 приказан је поступак претраге у оквиру породице чипова коришћењем кључне речи **esp8266** [7].

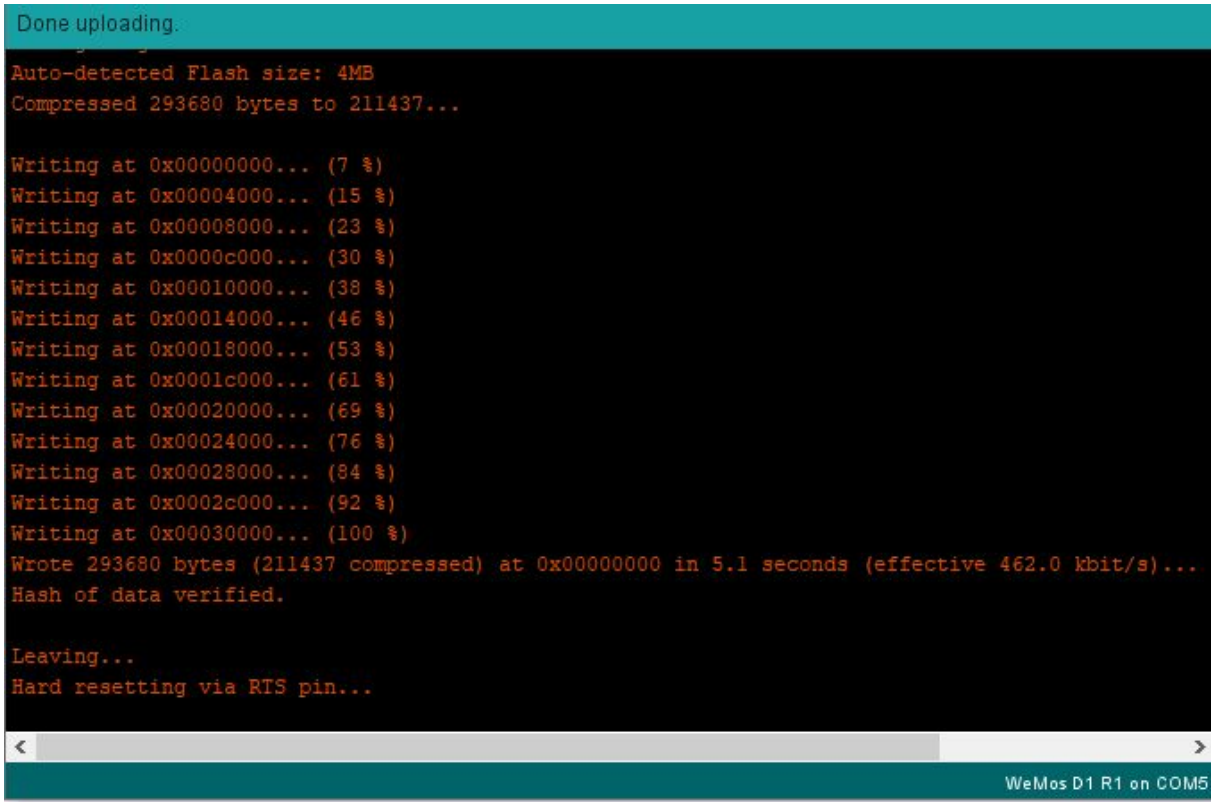


Слика 8 - *Инсталација пакета за WeMos D1 R1 развојну плочу*

Инсталирањем esp8266 пакета, инсталиране су и све потребне библиотеке које се односе на само коришћење WeMos D1 mini развојне плоче као што су ***ESP8266WiFi.h*** и ***ESP8266HTTPClient.h*** које ће бити коришћене касније у овом раду. Осим библиотека, уз пакет иду и основни примери директно везани за примену поменуте WiFi развојне плоче.

Када је пакет инсталиран, потребно је у опцијама **Tools->Board** из падајућег менија изабрати **WeMos D1 R1.** Након тога треба повезати WeMos D1 mini са рачунаром преко USB кабла. Уколико рачунар сам не региструје нови уређај, потребно је инсталирати одговарајуће драјвере, али то се дешава у ретким случајевима. Када рачунар региструје уређај, потребно је у оквиру Ардуино софтвера селектовати одговарајући порт, одабиром у **Tools->Port** и селектовањем порта на који је повезана развојна плоча.

Након тога, кликом на стрелицу **upload** програмирамо чип. Тада је потребно снимити пројекат, уколико то није урађено раније. У овом случају је то празан пројекат који само служи да се провери да ли је радно окружење успешно подешено. Уместо празног пројекта, може се провера извршити и са неким од основних примера која иду уз пакет. Примери се могу наћи у **File-Examples**. Уколико је код успешно отпремљен, значи да је радно окружење успешно подешено. Изглед успешно отпремљеног кода приказан је на слици



Слика 9 - *Успешно отпремљен код на WeMos D1 mini*

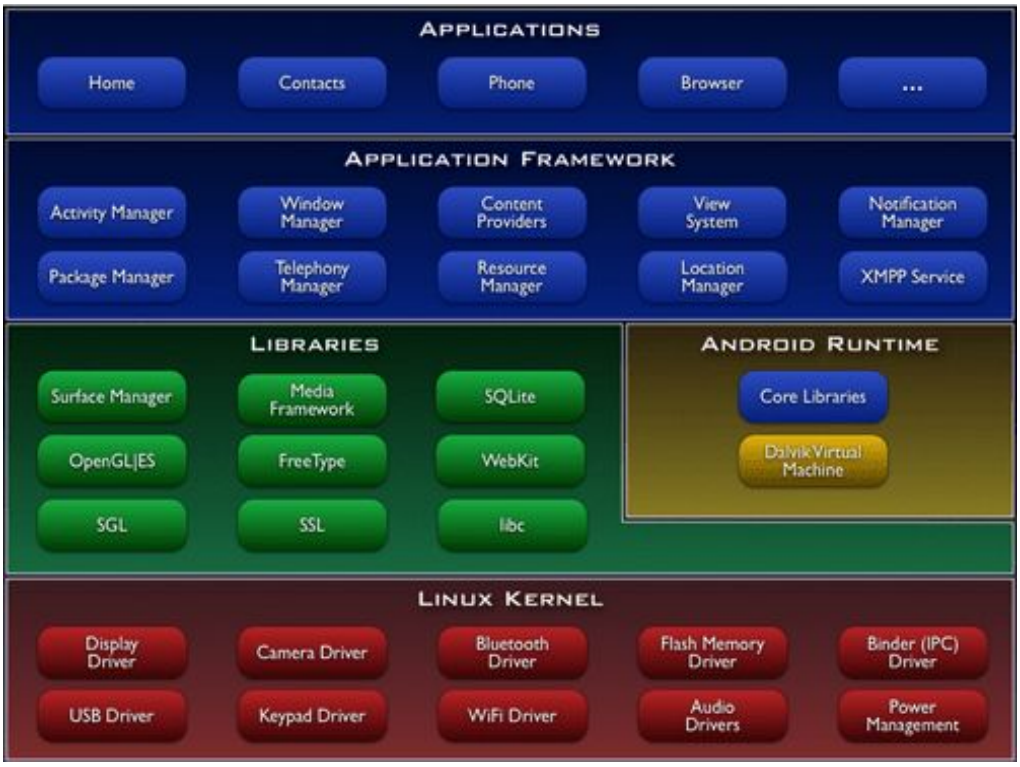
3.5. ANDROID

Андроид је мобилни оперативни систем развијен од стране компаније **Google**. Базиран је на модификованој верзији Линуксовог кернела. Првенствено је дизајниран за паметне уређаје осетљиве на додир као што су мобилни телефони и таблети. У основи, Андроид је развила компанија **Android, Inc**. коју је **Google** купио 2005.године. Први Андроид уређај на тржиште је изашао 2008.године и од тад се Андроид платформе убрзано развијају. По последњим статистикама, 75% паметних уређаја раде на Андроид оперативном систему.

Слика 10 – *Лого Андроида*

3.5.1. Архитектура Андроид оперативног система

Андроид оперативни системи засновани су на на верзији 2.6.*, а новији на верзији 3.1.* Linux оперативног система, тако да се за Ардуино оперативни систем може рећи да је Linux развијен за **ARM** и **x86** архитектуру. Андроид ОС је подељен на слојеве као што је приказано на слици 9.



Слика 11 - *Слојеви Андроид оперативног система* [12]

Linux језгро представља основу оперативног система и у њему су садржани покретачки програми свих хардверских компонената интегрисаних у мобилни уређај. У другом слоју су смештене библиотеке потребне за функционисање оперативног система. Слој **Android Runtime** је истог нивоа као и слој где се налазе библиотеке и у њему се налазе библиотеке потребне за програмирање **Java** Андроид апликација, као и **Dalvik** виртуална машина. **Dalvik** виртуална машина је софтвер одговоран за покретање апликација на Андроид уређајима. Ниво **Application Framework** директно комуницира са апликацијама и служи за управљање ресурисима, локацијама, активностима, нотификацијама итд. Апликативни слој чине апликације видљиве кориснику [12].

3.5.2. Програмирање Андроид апликација

Андроид је платформа отвореног кода. Изворни код чини комбинција више програмских језика, као што су Java, C, C++ и други. По први пут је то апсолутно отворена платформа која раздваја хардвер од софтвера који на њему ради. То омогућава много већем броју уређаја да покрећу исту апликацију. Осим тога, Андроид платформа је свеобухватна, што значи да представља комплетни софтверски пакет за мобилни уређај. За програмере, Андроид пружа све алате и оквире за брзо и лако развијање мобилних апликација.

Апликације које проширују функционалност уређаја, пишу се помоћу Андроид софтверског развојног комплета (енгл. *Software Development KIt - SDK*) и најчешће у **Java** програмском језику. **SDK** обухвата скуп развојних алата, као што су алати за дебагирање, софтверске библиотеке, емулатор, потребне документације, као и основне примере и туторијале. Осим **Jave** све већу популарност стиче и програмски језик **Kotlin**, а уз то могуће је писати апликације и у **C/C++** програмском језику, уз одговарајућа проширења[13].

Некада је Ардуиново развојно окружење (енгл. IDE) био је **Eclipse** уз коришћење Андроид развојног алата (енгл. *Android Development Tools - ADT*). Децембра 2014.године, **Google** је развио **Android Studio**, као примарно окружење за развој мобилних апликација.

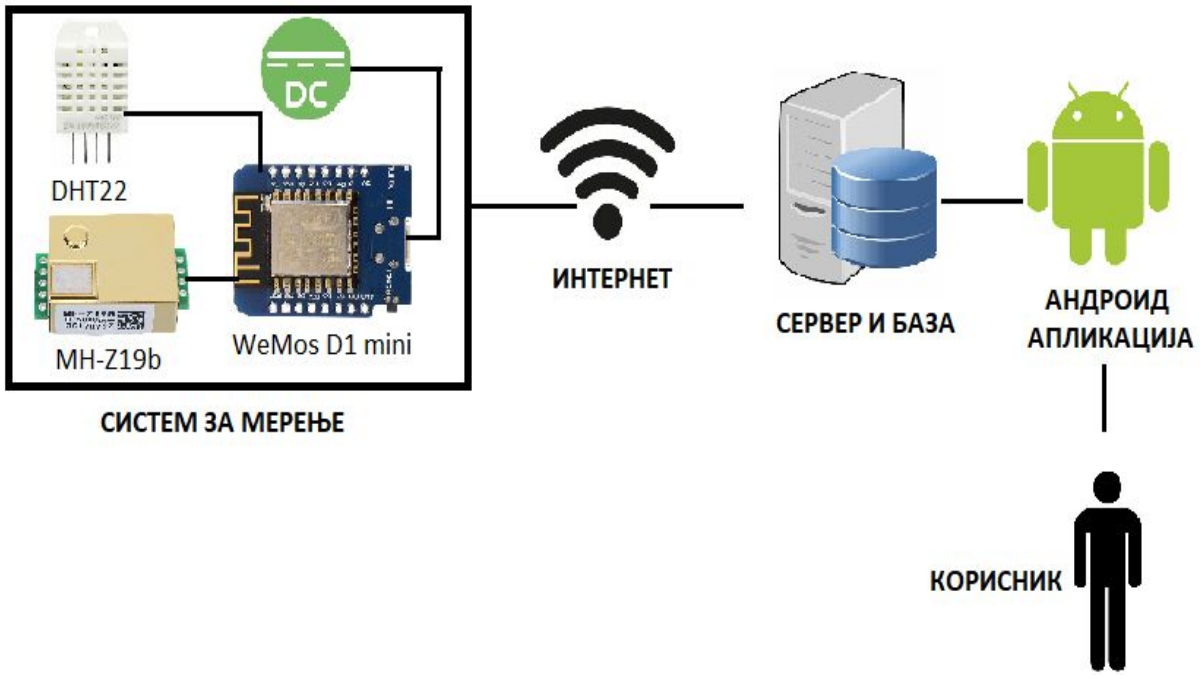
4. РЕАЛИЗАЦИЈА

Идеја је направити систем за мерење који прати три параметра и у одређеном временском интервалу снима измерене вредности у бази података. Параметри који се прате су температура, релативна влажност и концентрација угљен диоксида. Систем чине три компоненте:

- **WeMos D1 mini** - развојна плоча са esp8266 WiFi чипом
- **DHT22** - модул за мерење температуре и влажности
- **MH-Z19b** - модул за мерење концетрације угљен диоксида

Измерене вредности се помоћу WiFi конекције реализоване помоћу WiFi модула шаљу на сервер и складиште се у базу података.

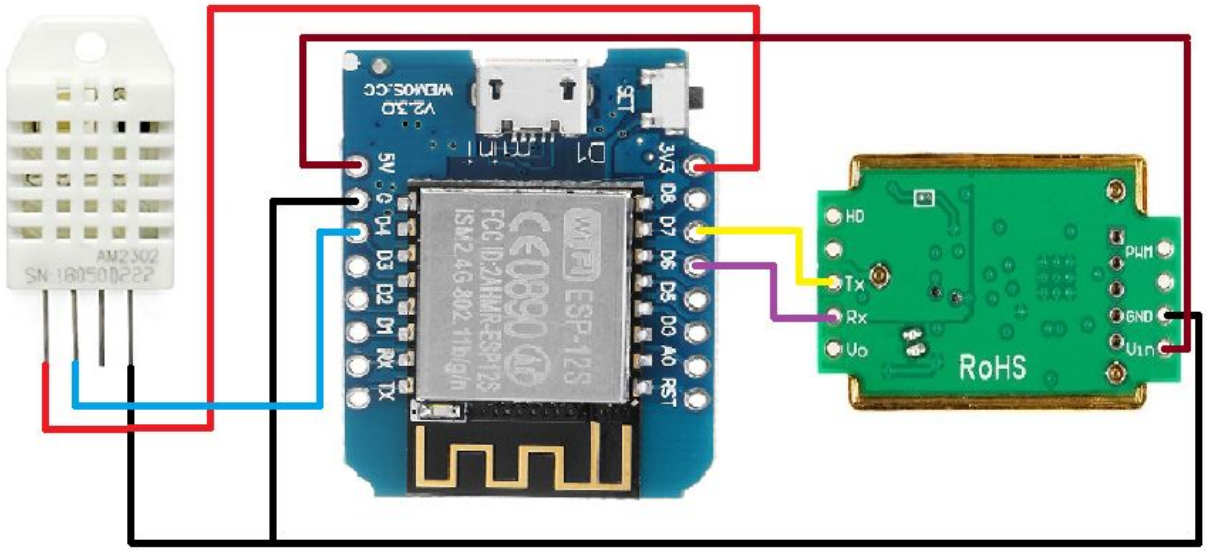
Други део система чини мобилна апликација за мониторинг измерених параметра. Коришћењем мобилне апликације корисник је у могућности да у реалном времену прати кретање параметра, као и да види графички приказ претходних мерења.



Слика 12 - *Шематски приказ пројекта*

4.1. Повезивање система

Први корак у реализацији мерења је физичко повезивање система, тј.повезивање модула за мерење са развојном плочом. Реализовано је тако што се пин VCC модула **DHT22** повеже на напон од 3.3V, GND на GND, а пин за пренос података са пином D4 развојне плоче. Кад је у питању повезивање **MH-Z19b** модула, потребно је Rx и TX пинове повезати са пиновима D6 и D7 развојне плоче, респективно. Пин Vin потребно је повезати на напон од 5V, а пин GND на GND. На слици 13 је приказано физичко повезивање модула са развојном плочом.

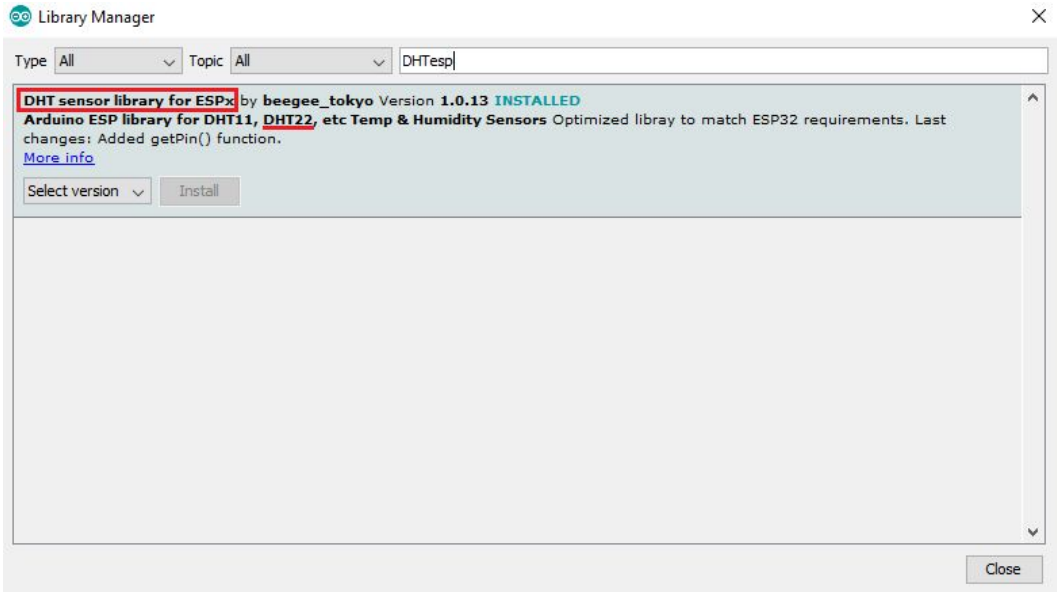


Слика 13 - *Повезивање модула са развојном плочом*

Наредни корак у реализацији је инсталирање потребних библиотека и програмирање чипа на плочи у Ардуино развојном окружењу.

4.2. Инсталирање потребних библиотека

Библиотеке потребне за рад са WeMos D1 mini развојном плочом су аутоматски инсталиране приликом инсталирања пакета за поменуту плочу. Осим тих библиотека, потребне су и додатне библиотеке како би се рализовало читање вредности са модула за мерење. Када је у питању сензор за температуру и влажност потребно је инсталирати ***DHTesp.h*** библиотеку. Инсталирање се врши преко опције **Sketch->Include Library->Manage Libraries** где помоћу поља за претрагу и кључне речи DHTesp проналазимо потребну библиотеку. Библиотеку инсталирамо кликом на **install**. Инсталирана библиотека је специјално дизајнирана за ESP чипове. Доступна је и обична DHT.h библиотека, која може да се користи и са ESP чиповима, али није потпуно стабилна. Проблем са стабилношћу настаје зато што ESP8266 ради много брже него Ардуино.



Слика 14 - *Инсталирање библиотеке за модул DHT22*

Осим *DHTesp.h* библиотеке, како би **DHT22** модул могао да ради потребна је инсталирати и **Adafruit Unified Sensor** библиотеку. Инсталирање те библиотеке се врши на исти начин.

4.3. Програмирање плоче

Наредни корак је писање кода који треба да се отпреми на развојну плочу. На почетку је потребно учитати библиотеке које су потребне за програмирање функционалности система, као и дефинисати пинове и серијску комуникацију између развојне плоче и модула за мерење угљен-диоксида, као и прављење инстанце потребне за слање података са температурног сензора на развојну плочу.

```
#include <SoftwareSerial.h>
#include "ESP8266WiFi.h"
#include "ESP8266HTTPClient.h"
#include "DHTesp.h"

#define MH_Z19_RX D7
#define MH_Z19_TX D6

SoftwareSerial co2Serial(MH_Z19_RX, MH_Z19_TX);

DHTesp dht;
```

Код 1 - *Учитавање библиотека и прављење инстанци за комуникацију*

Након тога је потребно подесити окружење тако да може да се конектује на WiFi мрежу, како би се остварило слање података на сервер. За WiFi конекцију потребан је **ssid** и **password** WiFi мреже. Користећи функције из библиотеке ***ESP8266WiFi.h*** реализује се конекција на мрежу. Код за конекцију на WiFi, као и сетовање комуникације између модула и развојне плоче приказан је на слици 13.

```
const char* ssid = "[redacted]";
const char* password = "[redacted]";

void setup() {
  Serial.begin(115200); // Init console
  pinMode(2, OUTPUT);
  Serial.println("Setup started");
  Serial.println("Starting...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println(WiFi.localIP());

  co2Serial.begin(9600);
  dht.setup(D4, DHTesp::DHT22);
}
```

Код 2 - *Конекција на WiFi мрежу*

Веома битан део кода је функција за читање концентрације угљен диоксида. Након што је остварена серијска комуникација између MH-Z19b модула и развојне плоче, шаље се команда која представља захтев за читање концентрације угљен диоксида. Помоћу функције ***memset()*** се преузима одговор, на основу ког се рачуна концентрација. Одгвор се састоји од девет хексадецималних бројева. Други број представља високу концентрацију (HIGH), док трећи представља ниску концентрацију. Потребно је превести обе концентрације у природне бројеве и након тога одговарајућом формулом израчунати концентрацију угљен диоксида. Осим тога, како би били сигурни да је одговор тачан, потребно је проверити да ли је формат одговора одговарајући, као и да ли је контролни збир, која је део одговора на захтев, тачан.

```
int readCO2()
{
  byte cmd[9] = {0xFF, 0x01, 0x86, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79};
  byte response[9];
  co2Serial.write(cmd, 9);
  while (co2Serial.available() > 0 && (unsigned char)co2Serial.peek() != 0xFF) {
    co2Serial.read();}
  memset(response, 0, 9);
  co2Serial.readBytes(response, 9);
  if (response[1] != 0x86){
    Serial.println("Invalid response from co2 sensor!");
    return -1;}
  byte crc = 0;
  for (int i = 1; i < 8; i++) {
    crc += response[i];
  }
  crc = 255 - crc + 1;
  if (response[8] == crc) {
    int responseHigh = (int) response[2];
    int responseLow = (int) response[3];
    int ppm = (256 * responseHigh) + responseLow;
    return ppm;
  } else {
    Serial.println("CRC error!");
    return -1;
  }
}
```

Код 3 - *Функције за читање концентрације угљен диоксида*

За читање температуре и релативне влажности користе се функције из библиотеке ***DHTesp.h*** и то **getTemperature()** и **getHumidity()**, респективно.

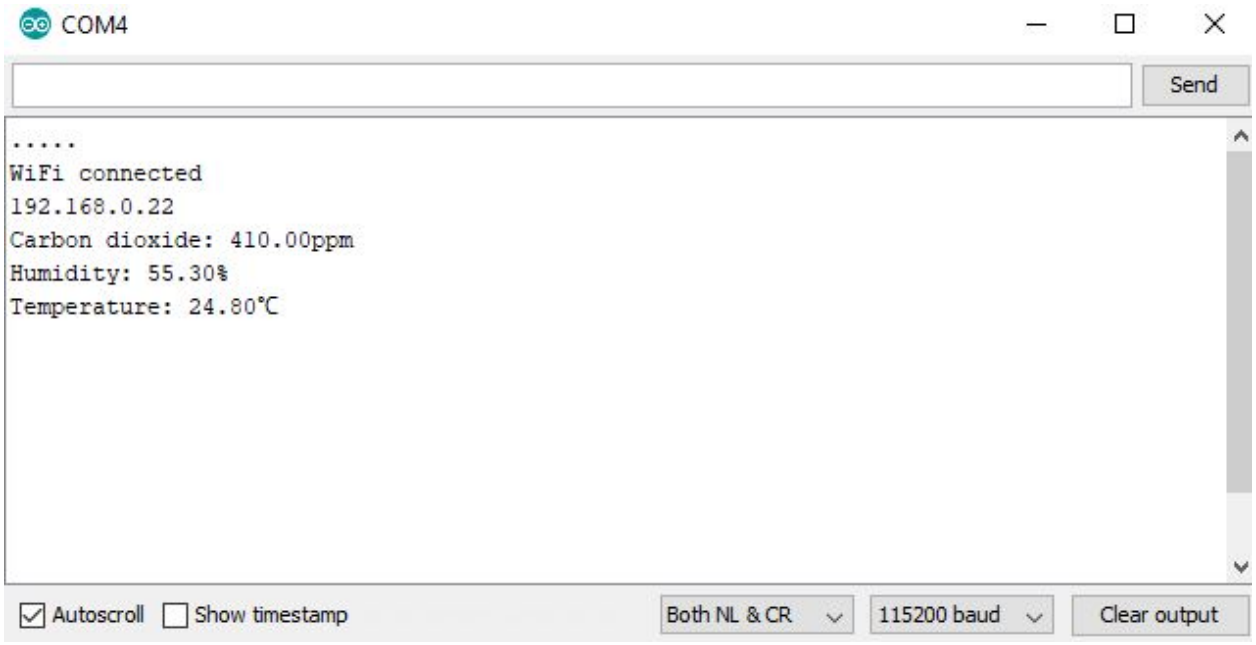
Након читања података, потребно је податке послати на сервер. Слање података је реализовано помоћу функције ***sendData()*** која sa аргументе прима назив сензора, парематар који мери и измерену вредност. У оквиру функције се прво проверава да ли је плоча повезана на мрежу и уколико јесте, помоћу HTTP GET захтева шаље податке серверу. HTTP GET захтев је реализован помоћу ***ESP8266HTTPClient.h*** библиотеке.

```
void sendData(String sensor, String attribute, float val)
{
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String getData, link, value;
    value = String(val, 2);
    getData = "?sensor=" + sensor + "&attribute=" + attribute + "&val=" + value;
    link = "http://192.168.0.16/sensorserver/addData.php" + getData;
    http.begin(link);

    int httpCode = http.GET();
    String payload = http.getString();
    http.end();
  }
  else
  {
    Serial.println("Error in WiFi connection");
  }
}
```

Код 4 - *Функција sendData()*

Након имплементације кода, потребно га је отпремити на развојну плочу помоћу опције **upload.** Када је све завршено, резлтати се могу видети на серијском монитору Ардуино развојног окружења.



Слика 15 - *Приказ серијског монитора*

4.4. База података

Базу података чини једна табела у којој су смештени сви подаци. Како изгледа табела приказано је на слици 16.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**id**	int(11)			No	*None*		AUTO_INCREMENT
2	**sensor**	varchar(20)	latin1_swedish_ci		No	*None*		
3	**attribute**	varchar(20)	latin1_swedish_ci		No	*None*		
4	**val**	float			No	*None*		
5	**time**	datetime			No	CURRENT_TIMESTAMP		

Слика 16 - *Приказ структуре табеле у PhpMyAdmin-u*

Табела се састоји од пет колона (атрибута):

- **id** - примарни кључ
- **sensor** - назив сензора
- **attribute** - параметар који сензор мери (температура, влажност, угљен-диоксид)
- **val** - измерена вредност
- **time** - датум и време кад је измерена вредност

Како би се бележило датум и време када је измерена одређена вредност, постављено је да атрибут **time** има подразумевану вредност тренутни датум и време (енгл. *current timestamp*). У табели су садржане све потребне информације. Додавањем додатних података, као што су информације о самим сензорима, учинило би базу компликованијем, без видљивих бенефита.

	id	sensor	attribute	val	time
Edit Copy Delete	26	DHT22	humidity	58.7	2019-09-09 18:10:35
Edit Copy Delete	27	DHT22	temperature	25.1	2019-09-09 18:10:35
Edit Copy Delete	28	MH-Z19b	CO2	799	2019-09-09 18:10:35
Edit Copy Delete	29	DHT22	humidity	58.5	2019-09-09 18:10:40
Edit Copy Delete	30	DHT22	temperature	25.1	2019-09-09 18:10:40
Edit Copy Delete	31	MH-Z19b	CO2	798	2019-09-09 18:10:40
Edit Copy Delete	32	DHT22	humidity	58.5	2019-09-09 18:10:45
Edit Copy Delete	33	DHT22	temperature	25.1	2019-09-09 18:10:45
Edit Copy Delete	34	MH-Z19b	CO2	795	2019-09-09 18:10:45

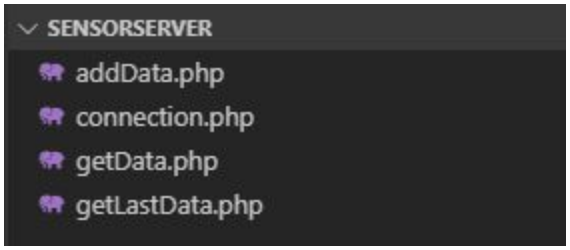
Слика 17 - *Приказ табеле измерених вредности*

4.5. Сервер

Сервер има улогу да прихвата све потребне податке и да их смешта у базу података. Осим тога, има улогу и да преузима већ складиштене податке из базе како би могли бити приказани у мобилној апликацији. Не постоји кориснички интерфејс, корисник апликације није свестан постојања овог дела апликације, већ се све обавља помоћу HTTP захтева. Серверски део апликације је реализован помоћу **WAMP** софтверског пакета, кога чини **Apache HTTP** веб сервер и **MySQL** систем за управљање базом података уз подршку **PHP** програмског језика.

Како би се реализовали HTTP захтеви, потребан је сервер на ком може да се покрене PHP код. WAMP користи **localhost**, тј. сам рачунар на ком се покреће програм представља виртуални сервер. Када је рачунар на мрежи, његов *localhost* је његова **IP** мрежна адреса. Извршавањем команде ***ipconfig*** у оквиру Command Prompt-a може се сазнати IP адреса рачунара. Како би био дозвољен приступ и другим уређајима потребно је користити IP адресу, а не резервисану реч *localhost.* Осим тога, потребно је искључити *windows firewall* на **порту 80**, што представља резервисани порт за *localhost*.

За развој PHP дела апликације коришћен је **Visual Studio Code**. Структура се састоји од четири фајла приказаних на слици 18.



Слика 18 - *Структура PHP дела апликације*

У фајлу **connection.php** је имплементирана функција за конектовање са базом података. Она садржи све потрбне параметре за конекцију, као што су име сервера, корисничко име и лозинка за PhpMyAdmin који је коришћен за администрацију MySQL система за управљање базом података, као и само име базе података. У оквиру функције је имплементирана и провера да ли је конекција успешна. Уколико је конекција неуспешна, функција враћа грешку.

```php
<?php
    function connection()
    {
        $servername = 'localhost';
        $username = 'root';
        $password = '';
        $dbname = 'sensorsdatadb';

        $conn = new mysqli($servername, $username, $password, $dbname);
        // Check connection
        if ($conn->connect_error) {
            die("Database Connection failed: " . $conn->connect_error);
        }
        return $conn;
    }
?>
```

Код 5 - *Функција за конекцију са базом података*

Фајл **addData.php** је намењен клијенту, задужен је за уписивање података у базу и то помоћу GET методе. На самом почетку се укључује connection.php фајл који је потребан за конекцију са базом. Након тога се преузимају подаци које треба уписати. Потребни подаци су име сензора, атрибут који сензор мери и измерена вредност. Датум и време кад је извршено мерење, као и ид се аутоматски генеришу и то је реализовано на нивоу саме базе. Извршавањем упита подаци се уписују у базу. Ако се деси да упит за уписивање у базу није успешно извршен, приказује се грешка.

```php
<?php
    include 'connection.php';
    $conn = connection();

    if(!empty($_GET['sensor']) && !empty($_GET['attribute']) && !empty($_GET['val']))
    {
        $sensor = $_GET['sensor'];
        $attribute = $_GET['attribute'];
        $val = $_GET['val'];

        $qry = "INSERT INTO sensorsdata(sensor, attribute, val) VALUES ('$sensor', '$attribute', $val)";

        if ($conn->query($qry) === TRUE) {
            echo "OK";
        } else {
            echo "Error: " . $qry . "<br>" . $conn->error;
        }
    }
    $conn->close();
?>
```

Код 6 - *Имплементација уписивање података у базу*

Други део PHP дела апликације је задужен за читање података из базе потребних за мобилну апликацију. Фајлови задужени за реализацију тог дела су **getData.php** и **getLastData.php.** Фајла getData.php чини функција getData() која за параметар прима број најновијих података по сензору које треба да прикаже. У оквиру функције прво се врши конекција са базом, након чега се извршавају упити за селектовање података из базе. Сви селектовани подаци се смештају у један низ. Због каснијег приказивања података на дијаграмима, битно је да селектовани подаци буду поређани у растућем поретку у односу на id, тј. да на крај низа иду најновије измерене вредности. То је реализовано у оквиру упита. Функција враћа исти тај низ у формату **JSON низа**.

```php
<?php
include 'connection.php';
function getData($num)
{
    $conn = connection();
    $qry1 = "SELECT * FROM (SELECT * FROM `sensorsdata` WHERE `attribute`='temperature' ORDER BY `id` DESC LIMIT $num)
                `sensorsdata` ORDER BY `id`";
    $qry2 = "SELECT * FROM (SELECT * FROM `sensorsdata` WHERE `attribute`='humidity' ORDER BY `id` DESC LIMIT $num)
                `sensorsdata` ORDER BY `id` ";
    $qry3 = "SELECT * FROM (SELECT * FROM `sensorsdata` WHERE `attribute`='CO2' ORDER BY `id` DESC LIMIT $num)
                `sensorsdata` ORDER BY `id`";
    $dbdata = array();
    $result1 = $conn->query($qry1);
    while ($row1 = $result1->fetch_assoc()){
        array_push($dbdata, $row1);
    }
    $result2 = $conn->query($qry2);
    while ($row2 = $result2->fetch_assoc()){
        array_push($dbdata, $row2);
    }
    $result3 = $conn->query($qry3);
    while ($row3 = $result3->fetch_assoc()){
        array_push($dbdata, $row3);
    }
    $conn->close();
    return json_encode($dbdata);
}
?>
```

Код 7 - *Фунцкција getData()*

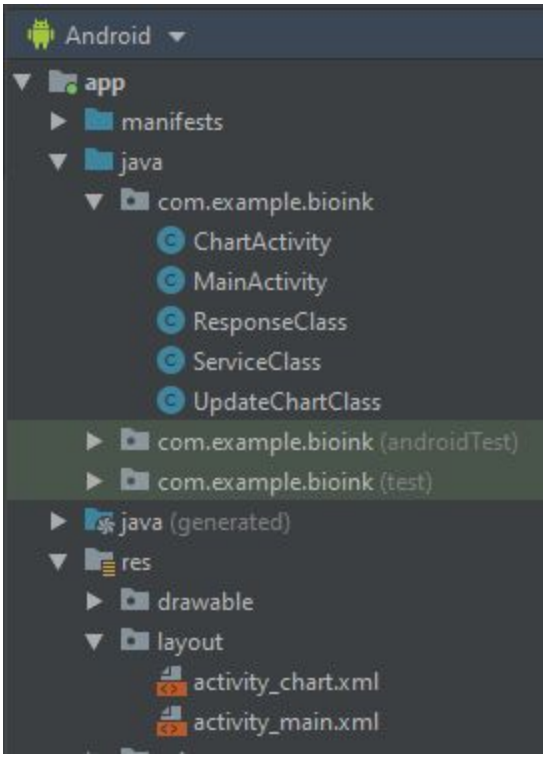
Фајл **getLastData.php**, као и addData.php намењен је коришћењу од стране клијента. У оквиру url-a је потребно назначити вредност параметра **lastData**, што представља број података по сензору које треба прочитати из базе и уз помоћ GET захтева се ивршава читање.

```php
<?php
    include 'getData.php';
    $numOfData = $_GET['lastData'];
    echo getData($numOfData);
?>
```

Код 8 - *Читање података из базе*

4.6. Корисничка апликација

Након реализације читања података са сензора и смештања тих истих података у базу развијена је апликација која служи за мониторинг тих вредности. Инсталирањем апликације на мобилном уређају, корисник ће бити у могућности да у реалном времену прати вредности које сензори измере. Апликација је рађена у **Android Studio** развојном окружењу уз коришћење **Java** програмског језика. Структура корисничке апликације приказана је на слици 19.



Слика 19 - *Структура апликације*

Апликација се састоји од два главна екрана, **MainActivity** и **ChartActivity**. MainActivity је главни екран и на њему се приказују последње измерене вредности температуре, релативне влажности и угљен диоксида. Кликом на слику графа, прелази се на ChartActivity на ком је графички приказано кретање измерених вредности у протеклом периоду појединачно за сваку величину.

У оквиру класе **ResponseClass** имплементирана је метода за учитавање последњих података са сервера. Учитавање података је реализовано помоћу библиотеке **Volley**. Креирањем објекта класе JsonArrayRequest, шаље се захтев за добијање података са сервера у облику JSON низа. У оквиру параметра приликом креирања објекта назначава се који захтев се шаље, у овом случају је то GET захтев. Осим тога, објекат за параметар прима и Response.Listener у оквиру ког се дефинише шта да се ради са одговором са сервера.

```
public synchronized void getCurrentData(String url, final Context context) {
    RequestQueue requestQueue = Volley.newRequestQueue(context);
    JsonArrayRequest jsonArrayRequest = new JsonArrayRequest(Request.Method.GET, url,
            jsonRequest: null, new Response.Listener<JSONArray>() {
        @Override
        public void onResponse(JSONArray response) {
            sendData(context,  key: "currentData", response.toString());
            Log.d( tag: "RESPONSE", response.toString());
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Log.e( tag: "errorResponse", error.toString());
        }
    });
    requestQueue.add(jsonArrayRequest);
}
```

Код 9 - *Функција за учитавање последњих података са сервера*

Као што је већ напоменуто, одговор је у облику JSON низа и помоћу функције sendData() одговор се шаље Activity-у за приказ. Функција sendData() за параметар прима одговор у облику стринга, зато је потребно JSON низ пребацити у стринг помоћу методе toString(). Функција sendData() креира интент са задатим кључем, убацује у њега податке које треба да проследи, у овом случају је то одговор са сервера и помоћу **LocalBroadcastManagera** их шаље.

```
public void sendData(Context context, String key, String data){
    Intent intent = new Intent(key);
    intent.putExtra( name: "data", data);
    LocalBroadcastManager.getInstance(context).sendBroadcast(intent);
}
```

Код 10 - *Функција sendData()*

Ради комуникације са сервером апликацији је потребан приступ интернету, а за то су јој потребна права приступа. Како би омогућили апликацији приступ интернету, потребно је у фајл **AndroidManifest.xml** додати следећу линију кода:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Код 11 - *Права приступа интернету*

Потребно је да апликација ради у реалном времену, да се стално освежава, како би увек биле приказане најновије измерене вредности. Како би се то реализовало, направљен је сервис **ServiceClass**, који периодично учитава податке са сервера. У оквиру сервиса је креиран процесни блок Runnable који имплементира методу *run()*. Унутар методе run() је стварни задатак који треба да се изврши у току текуће нити. У овом случају у оквиру run() методе се позива функција за учитавање података са сервера. Осим тога, у оквиру run() методе позива се и метода postDelayed(), која омогућава освежавање података на задати временски интервал. При покретању сервиса прво се позива метода onStartCommand(). У оквиру ње је имплементирано додавање runnable процеса као нове нити. Метода враћа **START_STICKY** и на тај начин је обезбеђено да уколико из било ког разлога сервис престане да ради да се опет покрене метода onStartCommand() и да сервис настави да ради.

```
public class ServiceClass extends Service {
    Handler handler;
    ResponseClass responseClass = new ResponseClass();

    public ServiceClass(){}

    @Nullable
    @Override
    public IBinder onBind(Intent intent) { return null; }
    private Runnable runnableService = new Runnable() {
        @Override
        public void run() {
            responseClass.getCurrentData( url: "http://192.168.0.16/sensorserver/getLastData.php?lastData=1",
                    context: ServiceClass.this);
            handler.postDelayed(runnableService, delayMillis: 5000);
        }
    };
    public int onStartCommand(Intent intent, int flags, int startI){
        handler = new Handler();
        handler.post(runnableService);

        return START_STICKY;
    }
}
```

Код 12 - *Имплементација класе ServiceClass*

Како би сервис могао бити покренут и како би систем могао да га види потребно га је декларисати у оквиру фајла **androidManifest.xml**. У овом случају то је сервис ServiceClass и декларација се реализује додавањем у манифест следеће линије кода:

```
<service android:name=".ServiceClass" />
```

Код 13 - *Декларација сервиса*

Након што је реализовано периодично читање података са сервера, као и декларисање сервиса, потребно је започети сервис и приказати последња мерења. Последња мерења су приказана у оквиру главног прозора, тако да је тај део кода имплементиран у класи MainActivity. У оквиру onCreate() методе региструје се **lastDataReceiver**, који је задужен за прихватање података са сервера. lastDataReceiver је објекат класе **BroadcastReceiver**. Он прихвата податке у облику стринга, који су послати преко раније поменуте sendData() методе и од њих поново прави JSON низ. Помоћу for петље врши се читање чланова низа и креирање одговарајућих JSON објекта.

```
private BroadcastReceiver lastDataReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String jsonArray = intent.getStringExtra( name: "data");
        try {
            JSONArray response = new JSONArray(jsonArray);
            for(int i = 0;i < response.length();i++){
                JSONObject jsonObject = response.getJSONObject(i);
                String value = jsonObject.getString( name: "val");
                String attribute = jsonObject.getString( name: "attribute");
                String sensor = jsonObject.getString( name: "sensor");

                if (attribute.equalsIgnoreCase( anotherString: "temperature")){
                    temperature = value;
                    temp.setText(temperature + "°C");
                    sensor1.setText(sensor);
                }
                else if (attribute.equalsIgnoreCase( anotherString: "humidity")){
                    humidity = value;
                    hum.setText(humidity + "%");
                    sensor2.setText(sensor);
                }
                else if(attribute.equalsIgnoreCase( anotherString: "CO2")){
                    CO2 = Double.parseDouble(value);
                    double fCO2 = CO2/10000;
                    co2.setText(fCO2 + "%");
                    sensor3.setText(sensor);
                }
            }
```

Код 14 - *Имплементација објекта* ***lastDataReceiver***

Осим регистровања објекта класе BroadcastReceiver у оквиру методе onCreate() референцирана су поља за приказ података са сервера. Уз то, имплементиран је и **ImageButton**. Кликом на дугме, прелази се на екран за приказ дијаграма. На крају саме методе покреће се сервис за периодично учитавање најновијих података из базе.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    LocalBroadcastManager.getInstance(this).registerReceiver(lastDataReceiver,
            new IntentFilter( action: "currentData"));

    temp = (TextView) findViewById(R.id.temperature);
    hum = (TextView) findViewById(R.id.humidity);
    co2 = (TextView) findViewById(R.id.co2);
    sensor1 = (TextView) findViewById(R.id.sensor1);
    sensor2 = (TextView) findViewById(R.id.sensor2);
    sensor3 = (TextView) findViewById(R.id.sensor3);

    graphs = (ImageButton) findViewById(R.id.imageButton3);
    graphs.setOnClickListener((view) -> {
            Intent i = new Intent( packageContext: MainActivity.this, ChartActivity.class);
            startActivity(i);
    });
    startService(new Intent(getApplicationContext(), ServiceClass.class));
}
```

Код 15 - *onCreate() метода / MainActivity*

Као што је већ поменуто, кликом на слику графика, прелази се на други Activity и то **ChartActivity**. Екран се састоји од дијаграма, који приказују историју мерења за сваку од мерних величина, појединачно, и то последњих 100 мерења.

Учитавање података са сервера, периодично освежавање података, као и све везано за сервисе врши се истоветно као и код најновијих мерења.

За креирање дијаграма користи се **GraphView**. Они су референцирани у оквиру onCreate() методе. Унутар onCreate() методе регистрован је и објекат класе BroadcastReceiver, помоћу ког се као што је већ напоменуто учитавају подаци са сервера. На крају саме методе се покреће сервис позивом методе startService().

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_chart);


    LocalBroadcastManager.getInstance(this).registerReceiver(lastData,
            new IntentFilter( action: "lastData"));

    graphTemp = (GraphView) findViewById(R.id.tempgraph);
    graphHum = (GraphView) findViewById(R.id.humgraph);
    graphCO2 = (GraphView) findViewById(R.id.co2graph);

    startService(new Intent(getApplicationContext(), UpdateChartClass.class));
}
```

Код 16 - *onCreate() метода / ChartActivity*

Потребно је дефинисати низ обејката **DataPoint**, где сваки DataPoint обејкат представља једну тачку дијаграма. Низ објеката чини серију типа **LineGraphSeries** која се додаје графу помоћу методе **addSeries()**. У овом случају серија се састоји од 100 тачака, што представља последњих 100 измерених вредности за сваку од посматраних величина.

```
public void onReceive(Context context, Intent intent) {
    String jsonArray = intent.getStringExtra( name: "data");
    try
    {
        int numOfValues = 100;
        DataPoint[] tempData = new DataPoint[numOfValues];
        DataPoint[] humData = new DataPoint[numOfValues];
        DataPoint[] co2Data = new DataPoint[numOfValues];
        int tempX = 0, humX = 0, co2X = 0;
        JSONArray response = new JSONArray(jsonArray);
        for (int i=0;i<response.length();i++){
            JSONObject jsonObject = response.getJSONObject(i);
            String attribute = jsonObject.getString( name: "attribute");
            Double val = jsonObject.getDouble( name: "val");
            String datetime = jsonObject.getString( name: "time");
            String timeStr = datetime.substring(11, 16);
            if (attribute.equalsIgnoreCase( anotherString: "temperature")){
                tempData[tempX] = new DataPoint(tempX, val);
                tempX = tempX + 1;
            }
            else if (attribute.equalsIgnoreCase( anotherString: "humidity")){
                humData[humX] = new DataPoint(humX, val);
                humX = humX + 1;
            }
            else if (attribute.equalsIgnoreCase( anotherString: "CO2")){
                double fval = val/10000;
                co2Data[co2X] = new DataPoint(co2X, fval);
                co2X = co2X + 1;
            }
        }
```

Kod 17 - *Имплементација DataPont серија*

GraphView захтева да вредности буду сортиране од најмањег ка највећем времену. Низ који се добија учитавањем података са сервера је већ сортиран у том поретку, што се реализује на нивоу упита. У реалном времену се добијају најновије вредности, али је низ увек сортиран тако да су најновије вредности на крају. Због динамичног мењања вредности, потребно је пре додавања ажуриране серије уклањање претходне. То се може учинити помоћу функције removeSeries().

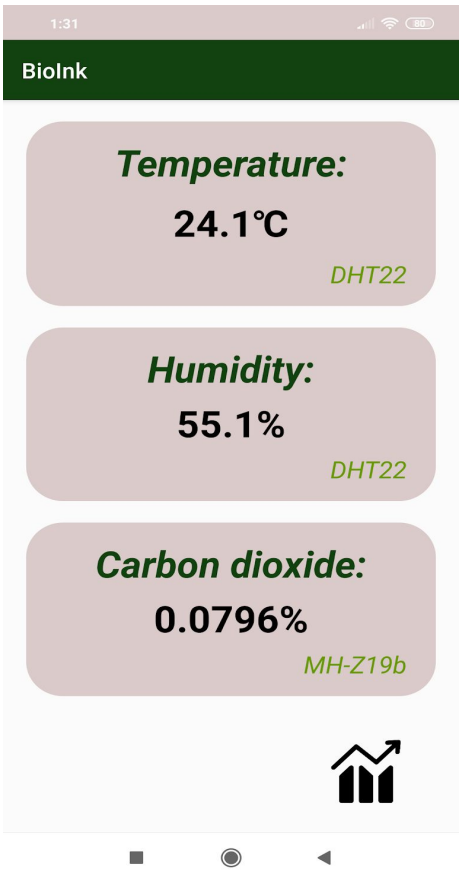
Постављање опсега оса врши се аутоматски, али је омогућено и ручно подешавање. Опсег на вертикалној оси се поставља аутоматски. Вредности посматраних параметра се не мењају брзо у току времена. Ради бољег приказа, опсег се скалира на основу тренутних вредности. На дијаграму су приказни последњих 100 мерења, па је на хоризонталној оси постављен опсег од 0 до 100. Сваки од три графичка приказа претходних података, за температуру, влажност и угљен диоксид су на исти начин имплементирани.

```
graphTemp.removeSeries(seriesTemp);
seriesTemp = new LineGraphSeries<>(tempData);
seriesTemp.setColor(Color.argb( alpha: 255,  red: 18,  green: 66,  blue: 0));
seriesTemp.setThickness(8);
graphTemp.addSeries(seriesTemp);
graphTemp.getViewport().setXAxisBoundsManual(true);
graphTemp.getViewport().setMinX(0);
graphTemp.getViewport().setMaxX(numOfValues);
```

Код 18 - *Имплементација дијаграма кретања температуре*

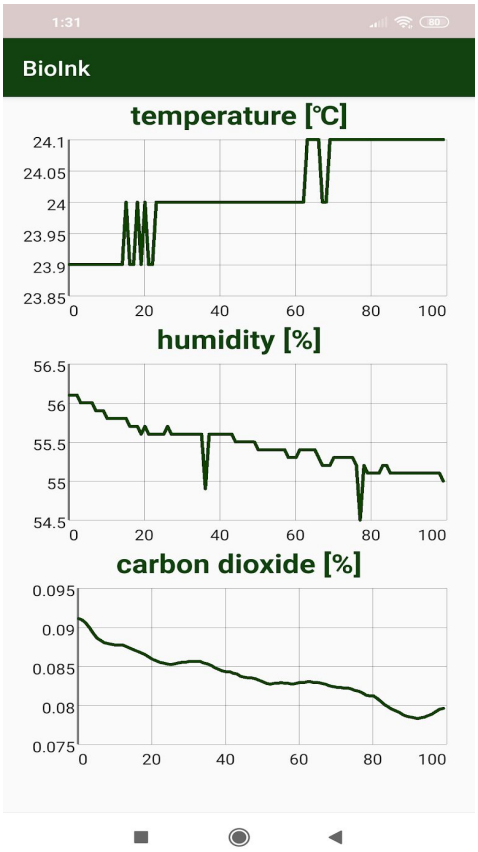
4.7. Коришћење апликације

Апликација је названа **BioInc**. Састоји се из два екрана. При покретању апликације се приказује главни екран, на ком су приказане тренутне вредности температуре, релативне влажности и концентрације угљен диоксида. Температура је приказана у целзијусима (℃), док су релативна влажност и концентрација угљен диоксида приказани у процентима (%). С обзиром да се учитавање из базе врши периодично, на сваких пет секунди учитавају се нове вредности. Уколико је дошло до промене неких од вредности, моћи ће се примети промена. Како изгледа почетни екран приказан је на слици 20.



Слика 20 - *Приказ почетног екрана апликације*

Осим самих вредности параметра које сензори мере, приказан је и назив сензора који је измерио вредност. У доњем десном углу се налази слика графика. Кликом на ту слику прелази се на други екран, на ком се налазе графички прикази кретања вредности за сваку величину појединачно. На дијаграмима су приказани последњих 100 измерених вредности. Како то изгледа приказано је на слици 21.



Слика 21 - *Приказ екрана са дијаграмима*

На сваких пет секунди се врши освежавање дијаграма, додају се нове измерене вредности, што се примећује на самим дијаграмима током коришћења апликације.

Закључак

Комерцијални CO2 инкубатори су доста скупи, па самим тим су неприступачни људима ван професионалих лабораторија који би се бавили неким мањим истраживањима. У оквиру овог рада је показано да уз мало стрпљења и импровизације могуће је направити систем за мерење истог типа као код професионалних инкубатора који је доста једноставнији, али релативно прецизан. У оквиру система за мерење пројектованог у овом раду коришћен је сензор за мерење концентрације угљен диоксида са мањим опсегом мерења него што то захтевају CO2 инкубатори. Код јефтиних модула опсег мерења концентрације угљен диоксида иде до 1%, а за CO2 инкубаторе је битно да може да мери бар до 5%, јер толики проценат обезбеђује потребну киселост средине. Како би се обезбедили оптимални услови које захтева CO2 инкубатор потребно је заменити тај модул за неки скупљи, за већим опсегом мерења, али у сваком случају принципи остају непромењени.

Мобилна апликација је дизајнирана тако да свако може да је инсталира на свом мобилном уређају и да у реалном времену прати како се крећу параметри. Комуникација између система за мерење и мобилне апликације је бежична, па самим тим није неопходно да се буде у близини инкубатора, како би се вршио мониторинг. Корисник апликације у сваком тренутку може да види које су тренутне вредности параметра, без обзира где се налази, потребно је само да има интернет.

У оквиру рада је пројектован само систем за мерење, а не и систем за аутоматско управљање параметрима. Како би CO2 инкубатор био комплетан, потребно је реализовати и систем за аутоматско одржавање оптималинх услова који треба сместити унутар неког изолованог кућишта.

Литература

[1] Encyclopedia Britannica, Incubator, insulated enclosure https://www.britannica.com/technology/incubator [последњни приступ: 19. септембар 2019.]

[2] AtmoSAFE.net, Incubator https://www.atmosafe.net/en/glossary/incubator.html [последњи приступ: 19. септембар 2019.]

[3] Lab Manager, CO2 Incubators: Mayinstays of Cell or Tissue Culture https://www.labmanager.com/lab-product/2011/11/co-sub-2-sub-incubators-.XYJcDWaxXIU [последњи приступ: 19. септембар 2019.]

[4] Binder, Everything worth knowing about CO2 Incubators https://pages.binder-world.com/en/co2-incubator-function [последњи приступ: 19. септембар 2019.]

[5] Професионални CO2 инкубатор https://www.litnz.co.nz/wp-content/uploads/2018/06/CO2-Incubator.jpg [датум преузимања: 19. септембар 2019.]

[6] WEMOS Electronics, D1 mini https://wiki.wemos.cc/products:d1:d1_mini [последњи приступ: 19. Септембар 2019.]

[7] Getting Started with WeMos D1 Mini https://www.teachmemicro.com/getting-started-wemos-d1-mini/ [последњи приступ: 19. септембар 2019.]

[8] Digital-output relative humidity & temperature module DHT22 https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf [последњи приступ: 19. септембар 2019.]

[9] Testing the MH-Z19 Infrared CO2 Sensor Module https://www.circuits.dk/testing-mh-z19-ndir-co2-sensor-module/ [последњи приступ: 19. септембар 2019.]

[10] Intelligent Infrared CO2 Module (Model: MH-Z19b) https://www.winsen-sensor.com/d/files/infrared-gas-sensor/mh-z19b-co2-ver1_0.pdf [последњи приступ: 19. септембар 2019.]

[11] SVET KOMPJUTERA, Programiranje na platformi Arduino https://www.sk.rs/2016/11/skp05.html [последњи приступ: 19. септембар 2019.]

[12] The Beginner's Guide to Android: Android Architectura https://www.edureka.co/blog/beginners-guide-android-architecture/ [последњи приступ: 19. септембар 2019.]

[13] Uvod u Android programiranje https://www.knjizara.com/pdf/142076.pdf [последњи приступ: 19. септембар 2019.]