Platforma s IOS aplikacijom za upravljanje Arduinom preko interneta

Došlić, Mateo

Master's thesis / Diplomski rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:091425

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-20**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I

INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

PLATFORMA S IOS APLIKACIJOM ZA UPRAVLJANJE ARDUINOM PREKO INTERNETA

Diplomski rad

Mateo Došlić

Osijek, 2019.

SADRŽAJ

1. UVOD

Ovaj diplomski rad ima za cilj kontrolirati Arduino uređaj pomoću udaljenog pametnog telefona. U pametni telefon (engl. *smartphone)* (nadalje u tekstu iOS uređaj) implementira se program koji će kontrolirati paljenje i gašenje LED (eng. *Light Emitting Diode*) rasvjete (nadalje u tekstu: LED) i brzine ventilatora. Prima podatke sa Arduino senzora za temperaturu i vlagu u zraku te iste prikazuje na ekranu iOS uređaja. Komunikacijom preko internet veze pomoću MQTT (eng. *Message Queuing Telemetry Transport*) servera šalju se potrebne informacije sa iOS uređaja na Arduino mikroprocesor i obrnuto. Dobivene informacije se očitaju i pretvaraju u tip podataka koji Arduino ili iOS uređaj „razumiju“ npr. brzina ventilatora ili podatci za temperaturu zraka. Za programiranje Arduina koristi se Arduino *compiler* koji koristi programski jezik C/C++. Dok iOS uređaj (iPhone) koristi Xcode u Swift programskom jeziku.

Glavni zadatak diplomskog rada je realizacija platforme s iOS aplikacijom koja upravlja Arduinom preko interneta. U drugom poglavlju teoretski se obrađuje tehnologija korištenja uređaja i alata. Upoznaje se sa Arduino mikroupravljačkim okruženjem, Croduino razvojnom maketom, MQTT tehnologijom, Xcode-om, iOS operacijskim sustavom i bibliotekama (u daljnjem tekstu: *library*). U trećem poglavlju opisana je realizacija sustava. Od konceptualnog dijagrama i montažne sheme, pa do programiranja i testiranja sustava.

1.1. Zadatak diplomskog rada

U ovom diplomskom radu potrebno je izraditi platformu s iOS aplikacijom za upravljanje Arduinom preko interneta. Potrebno je omogućiti slanje i primanje podataka na iOS uređaj kao i na Arduino uređajima. Ti podatci će omogućavati upravljanje LED-om i brzinu ventilatora te slati podatke iOS uređaju koji sadrže trenutnu temperaturu i vlagu u zraku.

2. KORIŠTENE TEHNOLOGIJE I ALATI

2.1. Server okruženje

2.1.1. MQTT tehnologija

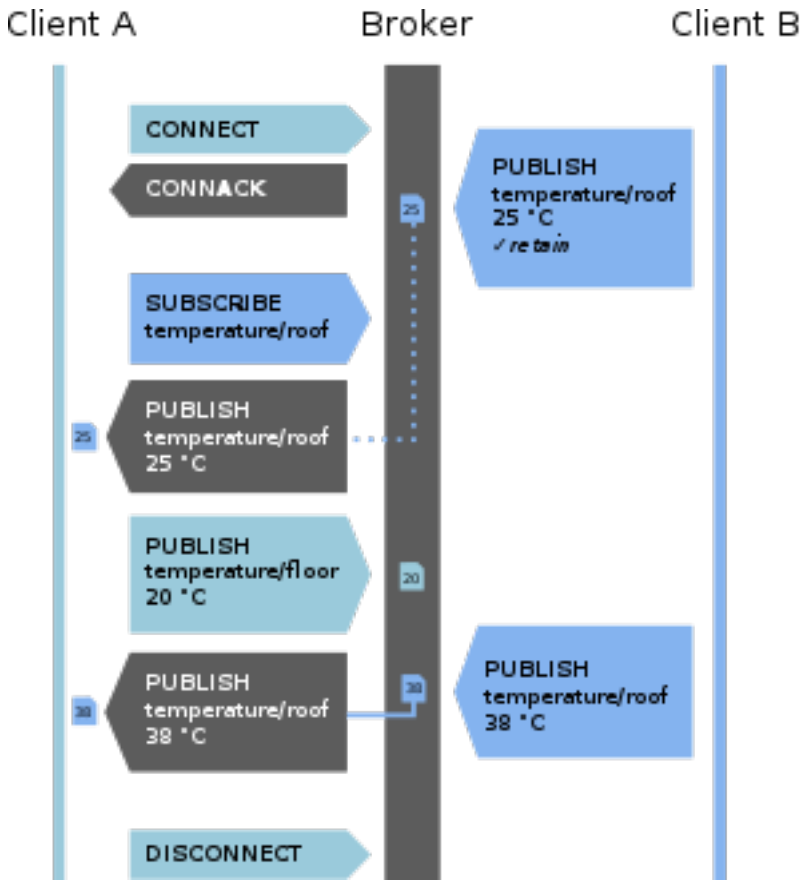
MQTT (eng. *Message Queuing Telemetry Transport*) (u daljnjem tekstu MQTT) je ISO standard (ISO / IEC PRF 20922) *publish-subscribe* protokol za razmjenu poruka koji radi na principu TCP/IP protokola. Dizajniran je za povezivanje s udaljenim lokacijama gdje je potreban *small code footprint* ili je ograničena propusnost mreže (eng. *limited network bandwidth*). *Publish-subscribe messaging* obrazac zahtjeva posrednika poruka (eng. *message broker*). Andy Stanford-Clark iz IBM-a i Arlen Nipper iz Cirrus Linka autori su prve verzije protokola 1999. godine.

MQTT sustav sastoji se od klijenata koji komuniciraju s poslužiteljem (u daljnjem tekstu *broker*). Klijent može biti izdavač (u daljnjem tekstu *publisher*) informacija ili pretplatnik (u daljnjem tekstu *subscriber*). Svaki klijent može se povezati s *broker*-om. Informacije su organizirane u hijerarhiji tema (u daljnjem tekstu *topic*). Kada publisher ima novu poruku za distribuciju, šalje kontrolnu poruku s podacima povezanom *broker*-u. Broker distribuira informacije svim *subscriber*-ima koji su pretplačeni na taj *topic*. Publisher ne mora imati nikakve podatke o broju ili lokacijama *subscriber*-a , a *subscriber*-i ne moraju znati podatke o *publisher*-u. Ako *broker* primi *topic* za koji nema trenutačnih *subscriber*-a, odbacit će *topic* osim ako *publisher* ne navede da se *topic* zadrži. To novim *subscriber*-ima omogućuje primanje najnovijih vrijednosti umjesto da čekaju sljedeće ažuriranje *broker*-a.

Kada se *publisher* prvi puta poveže na *broker*, može postaviti zadanu (eng. *default*) poruku koja će se slati svim *subscriber*-ima ukoliko taj *publish* klijent neočekivano prekine komunikaciju sa *broker*-om.

Najmanja MQTT kontrolna poruka (poruka za distribuciju) može biti veličine dva bajta. Ukoliko je potrebno, može nositi gotovo 256 megabajta podataka.

MQTT se oslanja na TCP protokol za prijenos podataka. Šalje vjerodajnice veze (eng. *connection credentials*) u obliku običnog teksta koji ne uključuje nikakve mjere sigurnosti ili provjeru autentičnosti. Zaštitu integriteta prenesenih informacija od presretanja i dupliciranja pruža temeljni TCP transport. Na slici 2.1. prikazan je primjer MQTT veze između *publisher*a kao i *subscribea*, *publish* poruka između njih preko *broker*-a.

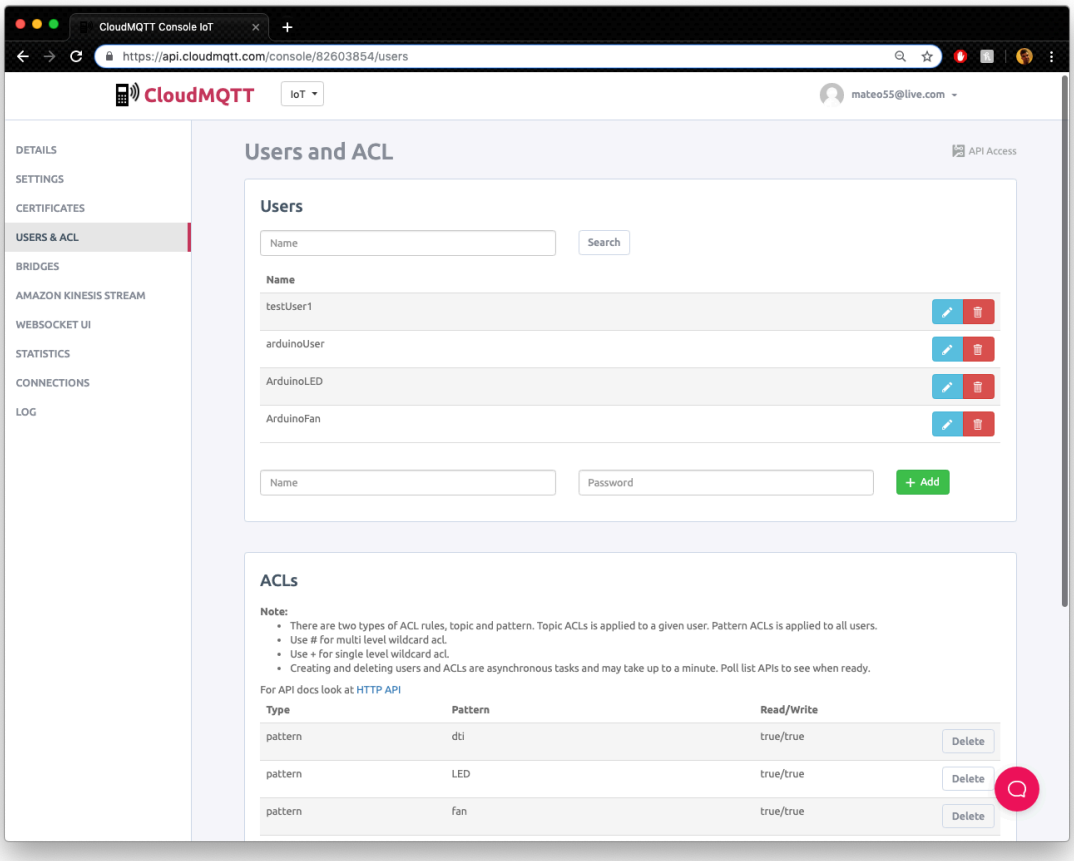


Slika 2.1. Primjer MQTT veze između dva klijenta.

Za ovaj rad koristi se već postojeći online servis MQTT *broker*-a. Taj servis nudi direktno prosljeđivanje poruka svim *subscriber*-ima koji su pretplačeni na taj *topic*.

Ovaj servis ne šalje zadane poruke nego samo prosljeđuje, čim *publisher* objavi poruku, na željeni *topic*.

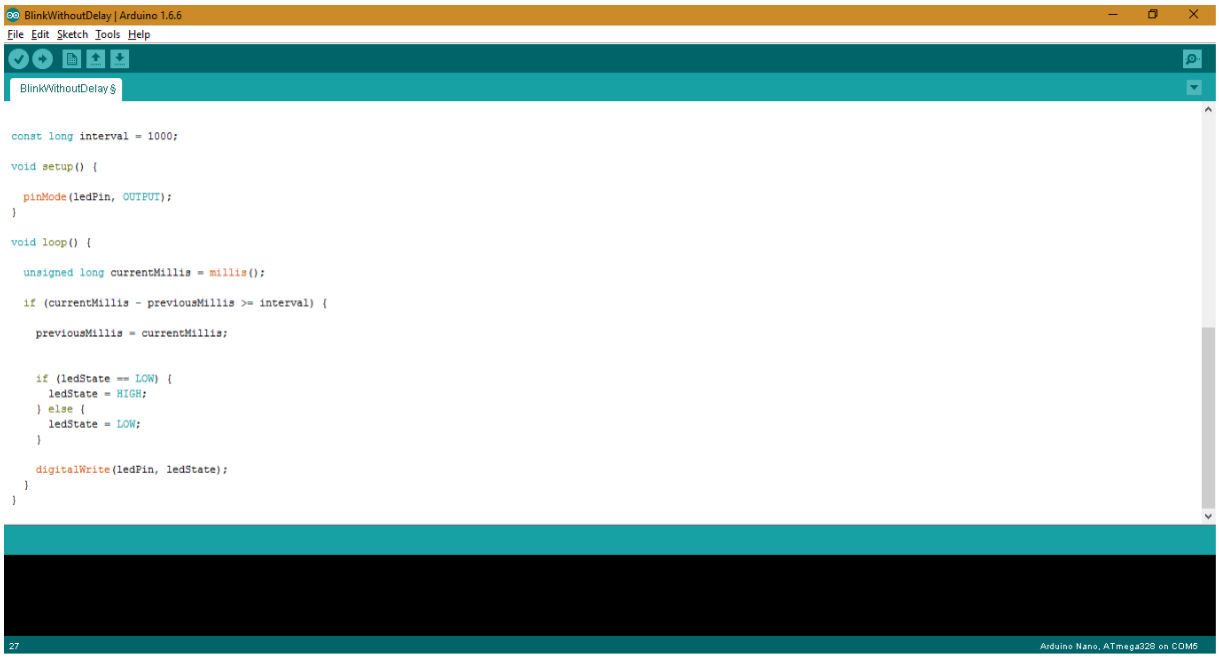
Na slici 2.2. prikazana je stranica online servisa koja prikazuje klijente i *topic*-e koji se koriste za izradu ovoga rada.



Slika 2.2. CloudMQTT online servis koji prikazuje klijente i topic-e.

2.2. Arduino mikroupravljačko okruženje

Arduino je najpoznatije ime za otvorenu računarsku i softversku platformu koja omogućava dizajnerima i konstruktorima stvaranje uređaja i naprava koji omogućuju spajanje računala s fizičkim svijetom. Arduino je stvorila talijanska tvrtka *SmartProjects* 2005.g. rabeći 8-bitne mikrokontrolere Amtel AVR, da bi stvorili jednostavnu malu i jeftinu platformu s kojom bi mogli lakše povezivati računala s fizičkim svijetom. Dizajneri su izabrali ime Arduino po imenu kafića u kojem su se dizajneri sastajali kada su stvarali projekt. Arduino je *open-source* prototipna platforma koja je jednostavna za korištenje hardvera i softvera. Arduino ploče su u stanju pročitati informacije na ulazu kao što je: svjetlost na senzoru, pritisak gumba, SMS poruke i sl. i pretvoriti ga u izlaz (aktiviranje motora, uključivanje LED svjetla, slanje podataka *online,* itd.). Da bi se to realiziralo, koristi se Arduino programski jezik koji je osnovan na programskom jeziku C/C++.



Slika 2.3. Primjer programskog sučelja za programiranje Arduino mikrokontrolera.

2.2.1. Arduino library

Arduino okruženje može se proširiti korištenjem *library*-a, kao i većina programskih platformi. *Library* pruža dodatnu funkcionalnost za upotrebu u skicama (eng. *sketch*), npr. rad s hardverom ili manipuliranje podacima. Da bi se koristio *library* u skici, odabere se *Sketch* > *Include Library*. Uz IDE (eng. *Integrated development environment*) se instalira određeni broj *library*-a, koji se mogu preuzeti sa interneta ili se mogu izraditi vlastite.

U nstavku se navode neki od osnovnih *library*-a:

- EEPROM - čitanje i pisanje u "trajnom" spremištu
- Ethernet - za povezivanje s internetom pomoću Arduino Ethernet štita (eng. *shield*), Arduino Ethernet štit 2 i Arduino Leonardo ETH
- Firmata - za komunikaciju s aplikacijama na računalu pomoću standardnog serijskog protokola.
- GSM - za spajanje na GSM/GRPS mrežu s GSM štitom.
- LiquidCrystal - za kontrolu zaslona s tekućim kristalima (LCD) …

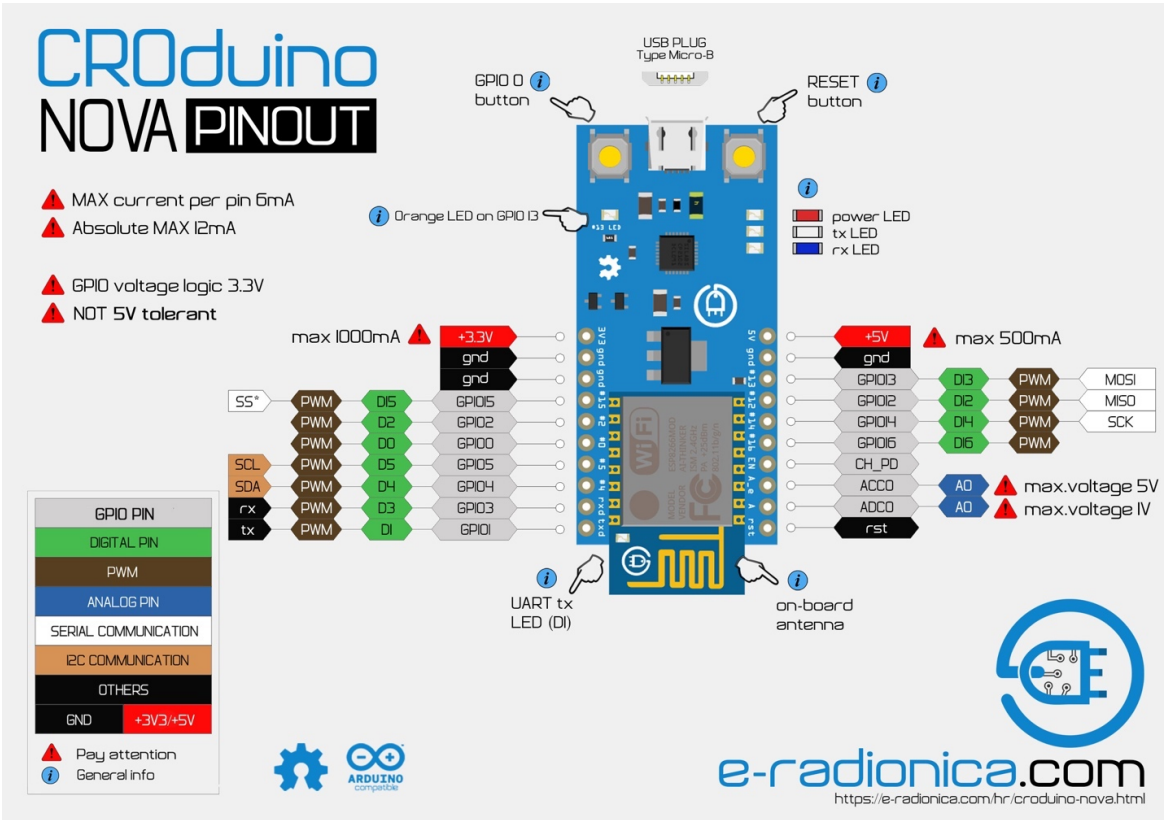
Library-i koji će se koristiti u ovom radu su: MQTT i ESP8266WiFi *library* za spajanje na Internet i povezivanje na MQTT *broker*.

2.3. Croduino razvojna maketa

Prvu Arduino kompatibilnu pločicu – Croduino proizvela je tvrtka TAVU d.o.o. (e-radionica) iz Osijeka sa željom da bude dostupnija našemu tržištu. U dvije godine postojanja, doživjela je proboj na tržište u brojne ustanove, škole i fakultete. Primarni cilj je edukacija u području programiranja i elektronike, a koristi se i za automatizacije. Uz osnovnu verziju nazvanu Croduino Basic, postoje još Pico, Damba, Proto, Power, Nova, Net & Basic 2, itd…

Danas uz pločice proizvode i mnoštvo senzora i aktuatora za njih. Uz široku ponudu proizvoda, nude i mnoštvo tutorijala, uputa i detalja o projektima te kako se koriste. Svaki od njih sa slikama, primjerima koda, neki i s videima. Također u svojoj ponudi nude primjere koje korisnici mogu iskoristiti za svoje projekte kao i korisničku i tehničku podršku za sve proizvode.

U ovom radu se koristi kompaktibilna pločica Croduino Nova prikazana na slici 2.4..



Slika. 2.4. Croduino Nova izgled i raspored pinova.

2.4. iOS

iOS (ranije iPhone OS) je mobilni operacijski sustav koji je Apple Inc. kreirao i razvio isključivo za svoj hardver. To je operacijski sustav koji pokreće mobilne uređaje, uključujući iPhone, iPad i iPod Touch. Nakon Androida drugi najpopularniji mobilni operativni sustav.

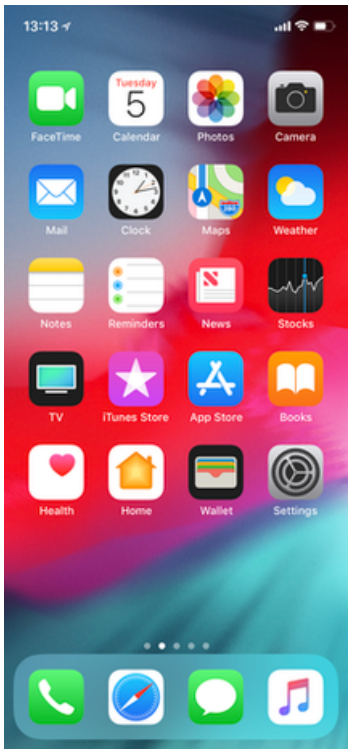
Predstavljen 2007. za iPhone, a proširen kako bi podržao druge Apple uređaje kao što su iPod Touch (rujan 2007.) i iPad (siječanj 2010).

Korisničko sučelje iOS-a temelji se na izravnoj manipulaciji, koristeći *multi-touch* geste. Kontrolni elementi sučelja sastoje se od klizača, prekidača i gumba. Interakcija s operacijskim sustavom uključuje geste kao što su povlačenje prstima, dodirivanje, štipanje (eng. *pinch*) i obrnuto štipanje (eng. *reverse pinch*), od kojih svi imaju određene definicije u kontekstu operacijskog sustava i njegovog *multi-touch* sučelja. Unutarnji akcelerometri koriste neke aplikacije kako bi odgovorili na potresanje uređaja (npr. naredba poništavanja unesenog teksta) ili rotiranje u tri dimenzije (npr. prebacivanje između portretnog i pejzažnog načina). Apple je pohvaljen zbog ugradnje temeljnih funkcija pristupačnosti u iOS, što korisnicima s oštećenjem vida i sluha omogućuje pravilno korištenje njihovih proizvoda.

Nove verzije iOS-a objavljuju se svake godine. Trenutna verzija, iOS 12, objavljena 17. rujna 2018. godine. Dostupna je za sve iOS uređaje s 64-bitnim procesorima. iPhone 5S i kasniji iPhone modeli; iPad (2017.), iPad Air i kasniji iPad Air modeli; sve iPad Pro modele; iPad Mini 2 i kasniji iPad Mini modeli; te iPod Touch 6. generacije. Na svim nedavnim iOS uređajima iOS redovito provjerava dostupnost ažuriranja, a ako je dostupan, od korisnika će zatražiti dozvolu za automatsku instalaciju.

iOS 13 bit će predstavljen javnosti u jesen 2019.g. Uvodi manje prilagodbe korisničkog sučelja zajedno s mnogim novim značajkama kao što su: redizajnirana aplikacija *Reminders*, *swipe* tipkovnica, poboljšana aplikacija *Photos* i novi *Dark Mode*. Lansiranjem na tržište završiti će s podrškom za nekoliko iOS uređaja, uključujući iPhone 5s, iPod Touch (6. generacija), iPhone 6 i iPhone 6 Plus, koji još uvijek čine više od 10% svih iOS uređaja.

Na slici 2.5. prikazan je početni zaslon iOS uređaja.



Slika 2.5. iOS 12 Početni zaslon na iPhoneX uređaju.

2.5. Xcode

Xcode je integrirano razvojno okruženje (IDE) za MacOS koji sadrži paket alata za razvoj softvera. Apple je razvio Xcode za razvoj softvera za MacOS, iOS, iPadOS, watchOS i tvOS. Prva stabilna verzija izdana je 2003. godine i od tada dostupna preko Mac App Store-a besplatno za korisnike MacOS-a. Registrirani developeri mogu preuzeti preliminarna izdanja i prethodne verzije Xcode-a putem Apple Developer web stranice.

Xcode podržava izvorni kod programskih jezika C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit (Rez) i Swift, s različitim programskim modelima, uključujući, ali ne ograničavajući se na, Cocoa, Carbon i Java. Treće strane su dodale podršku za GNU Pascal, Free Pascal, Ada, C #, Perl i D.

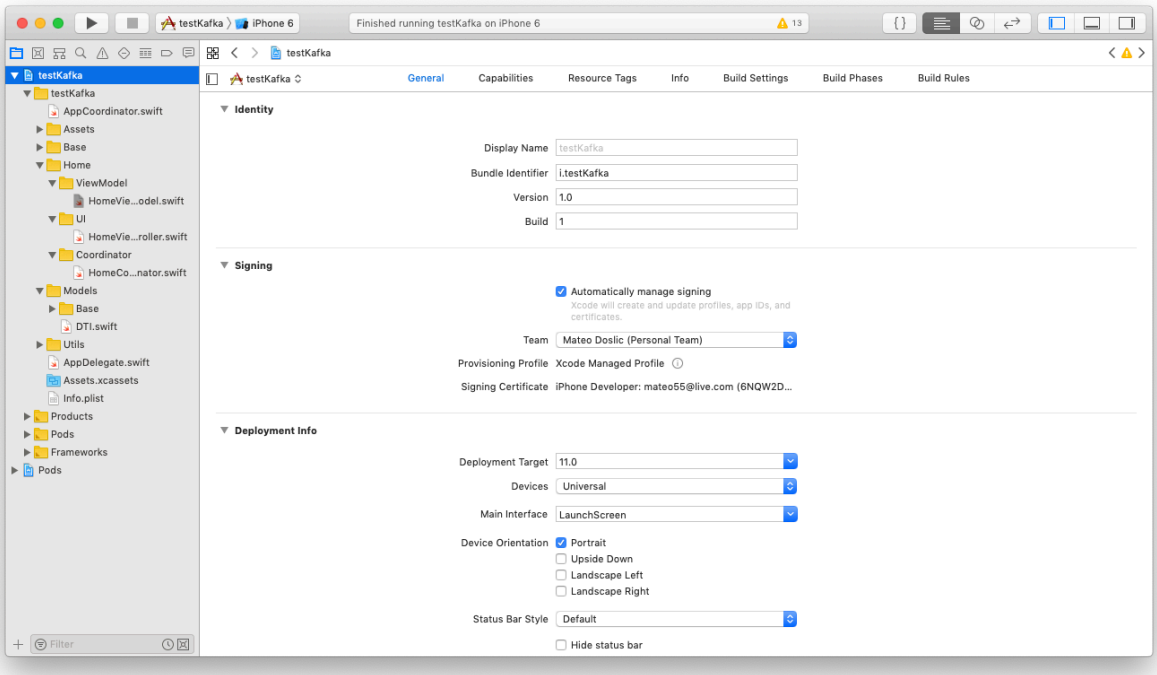
Xcode može izgraditi *fat binary* datoteke koje sadrže kod za više arhitektura s Mach-O izvršnim formatom. Te se datoteke nazivaju *univarsal binary* i omogućuju pokretanje softvera na PowerPC i Intel-baziranim platformama (x86), a mogu uključivati i 32-bitni i 64-bitni kod za obje arhitekture. Koristeći iOS SDK (eng. *Software Development Kit*), Xcode se također može koristiti

za kompajliranje i otklanjanje pogrešaka aplikacija za iOS koje rade na ARM (eng. *Advanced RISC Machine*) procesorima arhitekture.

Xcode uključuje GUI (eng. *Grafical User Interface*) alate koji rade na vrhu dinamičkog okvira za praćenje (eng. *Dynamic Tracing Network*), DTrace, kojeg je Sun Microsystems kreirao i objavio kao dio OpenSolarisa.

Glavna primjena paketa je integrirano razvojno okruženje (IDE), također nazvano Xcode. Paket Xcode uključuje većinu Appleove developerske dokumentacije i ugrađenu aplikaciju koja se koristi za izradu grafičkih korisničkih sučelja (eng. *Interface Builder*).

Na slici 2.6. prikazano je korisničko sučelje Xcode aplikacije.



Slika 2.6. Izgled Xcode korisničkog sučelja.

2.5.1. Swift

Swift je multi-paradigmin, kompilirani programski jezik koji je razvila tvrtka Apple za iOS, macOS, watchOS, tvOS, Linux i z/OS. Dizajniran je za rad s Appleovim Cocoa i Cocoa Touch okvirima i velikim dijelom postojećeg Objective-C koda pisanog za Apple proizvode. Izrađen je *open-source* LLVM *compiler framework*-om i uključen je u Xcode od verzije 6 pa nadalje.

Apple je namijenio Swiftu podršku za mnoge temeljne koncepte povezane s Objective-C, osobito *dynamic dispatch*, *late binding*, *extensive programming* i slične značajke, ali na "sigurniji" način, olakšavajući hvatanje softverskih grešaka.

Ima značajke rješavanja nekih uobičajenih programskih pogrešaka kao što je *null pointer dereferencing* kako bi se izbjegla piramida propasti (eng. *Pyramid of doom*). Podržava koncept proširivosti protokola. Sustav proširivosti koji se može primijeniti na tipove, strukture i klase, koje Apple promiče kao stvarnu promjenu u paradigmama programiranja pod nazivom "protokolno orijentirano programiranje".

Swift je predstavljen na Appleovoj WWDC (eng. *Worldwide Developers Conference*) 2014g..

Sintaksa verzije 3.0 Swifta prošla je kroz značajnu evoluciju, pri čemu je osnovni tim nadogradio stabilnost izvora. U prvoj četvrtini 2018. Swift je nadmašio Objective-C u mjerenju popularnosti.

2017.g. objevljen je Swift 4.0 koji je uveo nekoliko promjena u neke ugrađene klase i strukture. Kod koji je napisan s prethodnim verzijama Swift-a može se ažurirati pomoću značajke migracije ugrađene u Xcode.

U ožujku 2019.g objavljen je Swift 5. Uveo je stabilno binarno sučelje na Appleovim platformama, omogućavajući ugradnju Swift *runtime* u Apple operacijske sustave. Swift 5 izvor kompatibilan je sa Swift 4.

2.5.2. Cocoapods

CocoaPods je *application level dependency manager* za Objective-C, Swift i za bilo koji drugi programski jezik koji se izvodi na Objective-C *runtime* (npr. RubyMotion, koji pruža standardni format za upravljanje vanjskim *library*-ima).

CocoaPods se usredotočuje na distribuciju koda treće strane i automatsku integraciju u Xcode projekte.

CocoaPods se pokreće iz *command line* i također je integriran u JetBrains 'AppCode IDE. Instalira ovisnosti (npr. *Library*) za aplikaciju navođenjem ovisnosti, a ne ručno kopiranje izvornih datoteka. Osim instaliranja „*master*“ repositoriji sadrže meta podatke za sve *open-source library*-e koji se održavaju na git repozitoriju i hostiraju na GitHub-u.

CocoaPods sustav ovisnosti (eng. *dependency resolution system*) pokreće Molinillo koji se također koristi u drugim velikim projektima kao što su Bundler, RubyGems i Berkshelf.

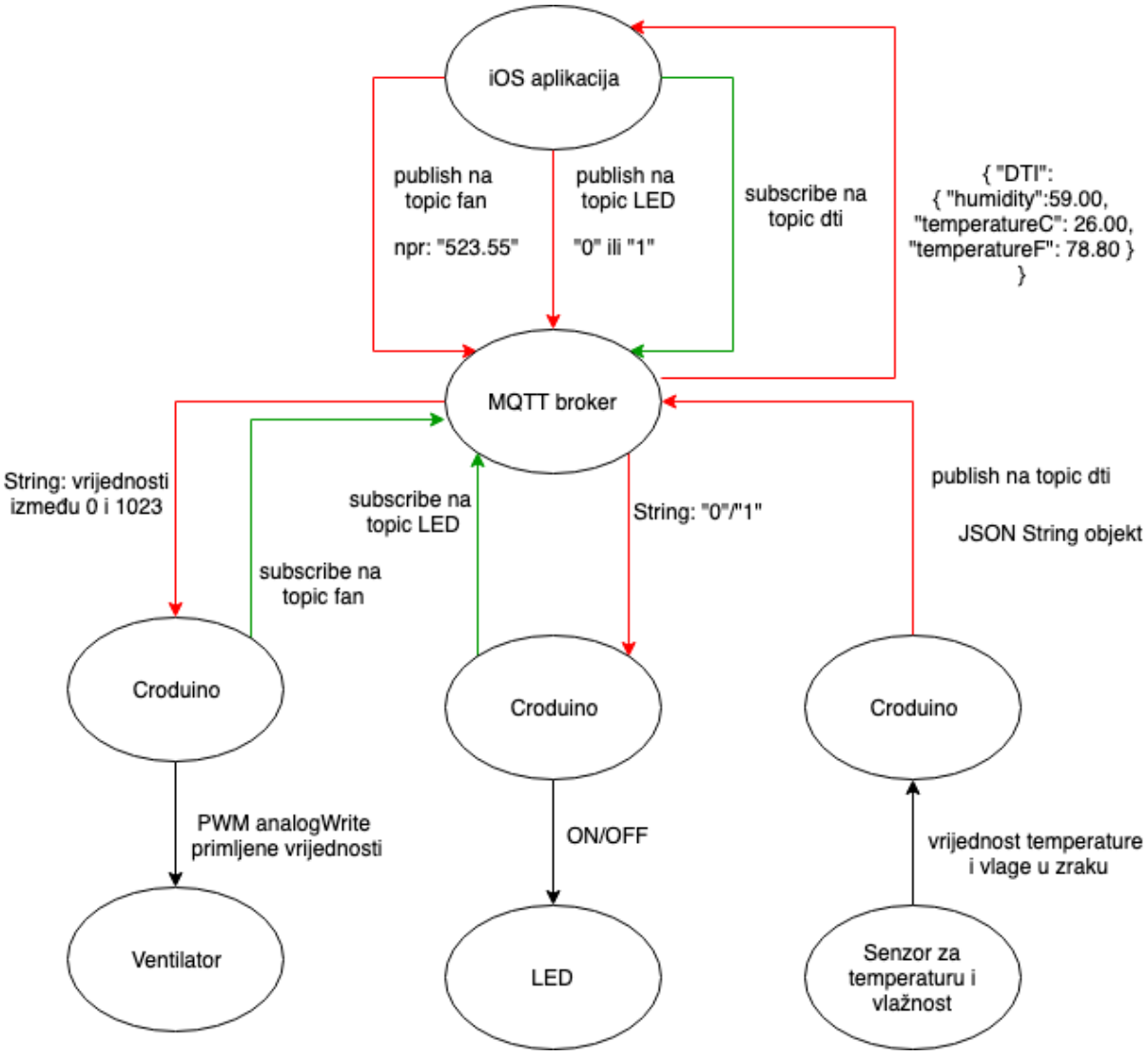
Cocoapod koji se koristi u ovom radu zove se Moscapsule.

Moscapsule je MQTT klijent za iOS pisan u Swiftu. Ovaj *framework* je implementiran kao *wrapper* Mosquitto *library*-a i pokriva gotovo sve značajke Mosquitto-a, te koristi Mosquitto verziju 1.4.8. Mosquitto je *open source message broker* koji implementira MQTT protokol verzije 3.1 i 3.1.1. MQTT pruža lagani način obavljanja poruka pomoću modela za *publish*/*subscribe*. Mosquitto je napisan u C jeziku.

3. REALIZACIJA SUSTAVA

3.1. Block dijagram sustava

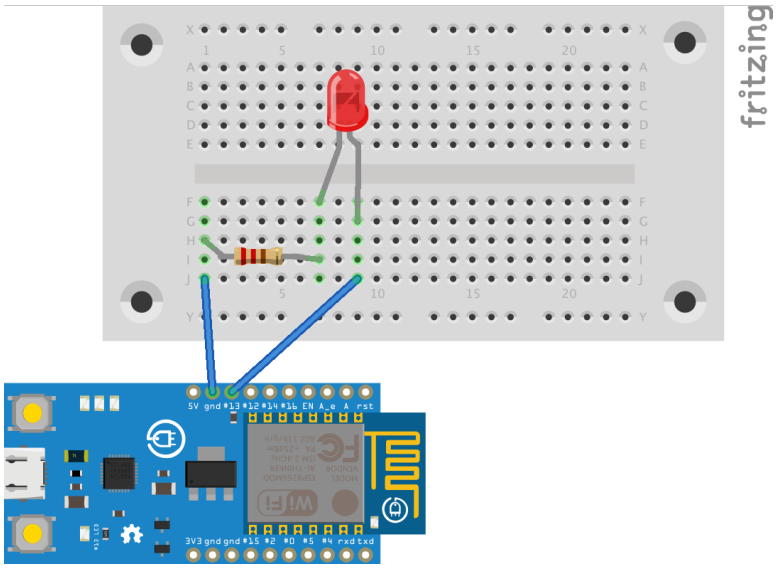
Na slici 3.1. prikazan je blok dijagram sustava platforme s iOS aplikacijom za upravljanje Arduinom putem interneta.



Slika 3.1. Blok dijagram sustava.

U blok dijagramu je vidljivo da glavnu ulogu ima MQTT *broker* koji služi za razmjenu informacija između Croduino-a i iOS uređaja. IOS aplikacija „radi" *subscribe* na *topic* dti u MQTT *broker* serveru gdje prima podatke o temperaturi i vlažnosti sa Croduino uređaja. Sve se informacije „šalju" u String formatu. U ovom radu se „šalje" (*publish*) više tipova podataka koji se kasnije rasčlanjuju (u daljnjem tekstu *parse*) u valjani tip podataka na *subscriber* strani. IOS aplikacija na *topic* LED „šalje" String podatke koji mogu biti „0" ili „1". Na Croduino strani taj String podatak pretvara se u Integer podatak i na osnovu vrijednosti odlučuje da li treba upaliti LED ili ugasiti (0 za gašenje, 1 za paljenje). Aplikacija na *topic* fan šalje String podatak u obliku decimalnog broja (npr. „512,23") koji može biti u rasponu od 0 do 1023. Na Croduino strani taj podatak se pretvara u Double (Float) tip podatka, a pomoću PWM-a (eng. *Pulse Width Modulation*) kontrolira se brzina okretaja ventilatora. iOS aplikacija dobiva podatke o temperaturi i vlazi u zraku u obliku JSON String objekta koji se *parse*-ju u Swift objekt, te prikazuju na korisničkom sučelju (eng. *display*). Croduino uređaji se *subscribe*-aju na *topic*-e LED, dok iOS uređaj se *subscribe*-a na *topic* dti.

3.2. Montažna shema arduino podsustava

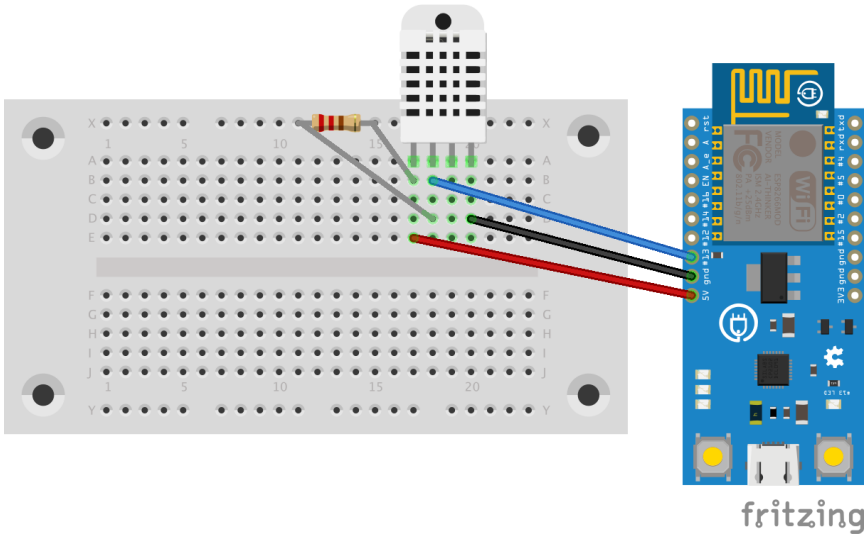


Shema 3.1. Montažna shema Croduino uređaja za LED.

U shemi 3.1. prikazana je montažna shema Croduino uređaja koji preko interneta prima naredbe i na osnovu naredbi pali ili gasi LED.

Shema prikazuje sljedeće elemente:

- Croduino Nova
- Otpornik od 220 Ω
- LED svjetlo

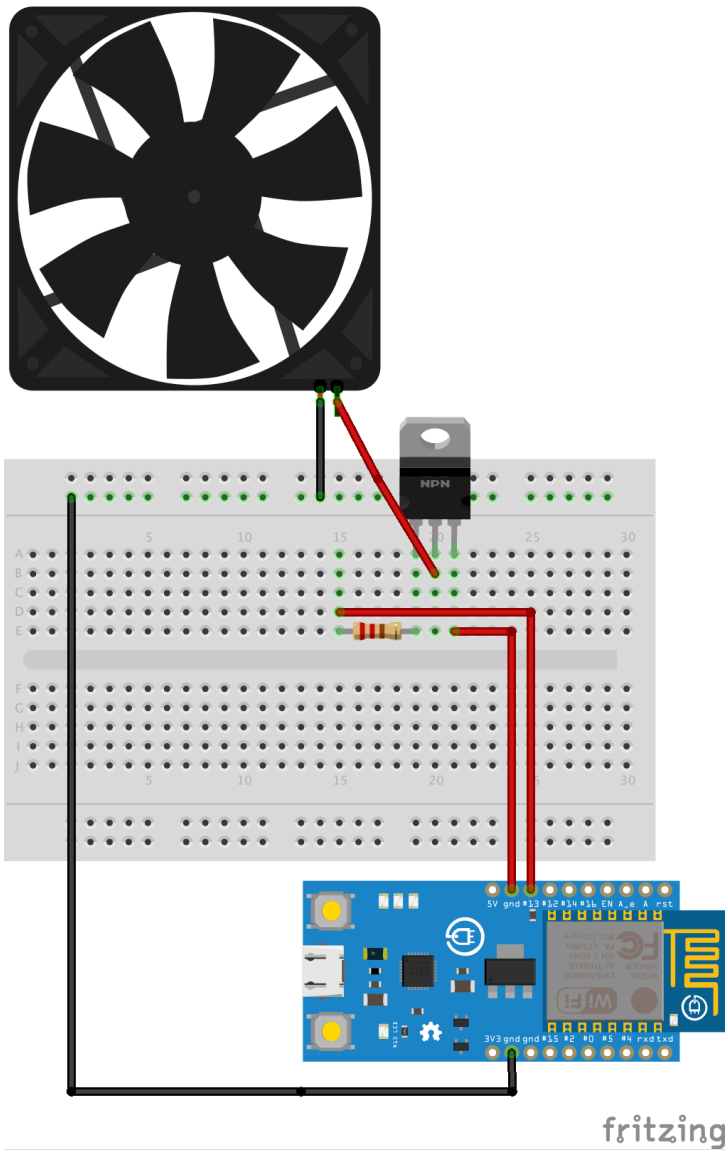


Shema 3.2. Montažna shema Croduino uređaja za mjerenje temperature i vlage u zraku.

U shemi 3.2. prikazana je montažna shema Croduino uređaja koji prikuplja podatke o temperaturi i postotku vlage u zraku, te šalje preko interneta te podatke prema uređajima koji su *subscribe*-ani na taj *topic*.

U toj shemi, vidljivi su sljedeći elementi:

- Croduino Nova
- Otpornik od 10 kΩ
- DHT11 modul



Shema 3.3. Montažna shema Croduino uređaja za mjerenje temperature i vlage u zraku.

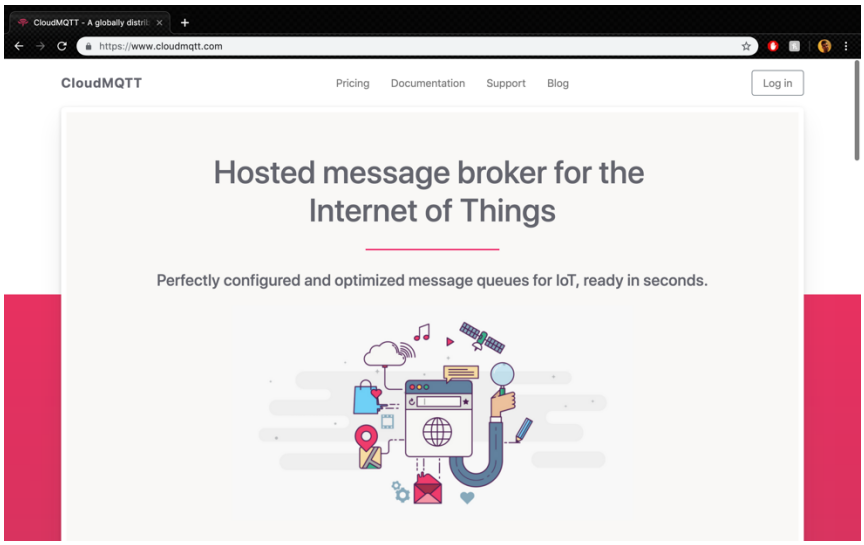
U shemi 3.3. prikazana je montažna shema Croduino uređaja koji preko interneta prima naredbe i na osnovu naredbi postavlja brzinu ventilatora.

Shema prikazuje sljedeće elemente:

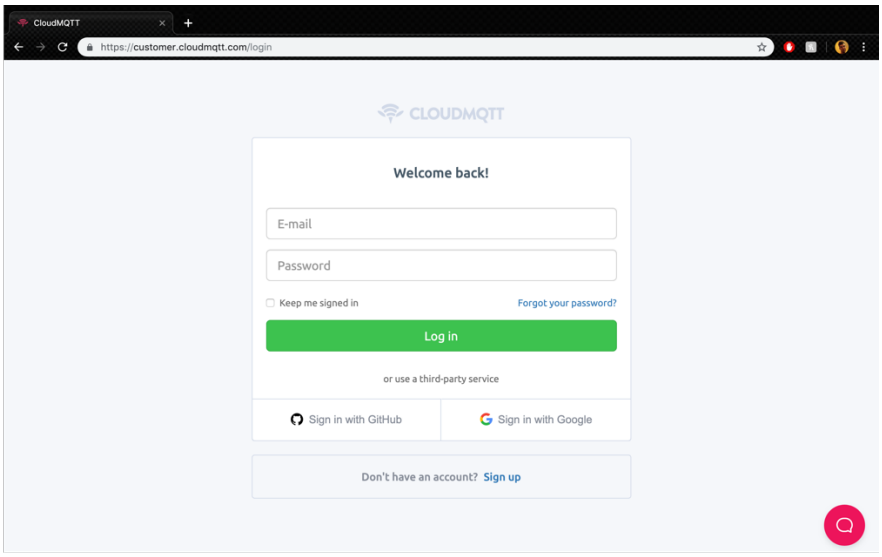
- Croduino Nova
- Otpornik od 10 kΩ
- 5 V ventilator
- NPN tranzistor

3.3. Postavljanje MQTT udaljenog servera

U ovome poglavlju slijede detaljne upute za otvaranje računa na MQTT udaljenom serveru. Da bi se registrirali potrebno je slijediti korake na zaslonu i to redom kako slijedi. Za registraciju je potrebno posjetiti stranicu u literaturi pod rednim brojem [5] . Izgled početne stranice prikazan je na slici 3.2.. Pritiskom na gumb Log in otvara se stranica prikazana na slici 3.3..

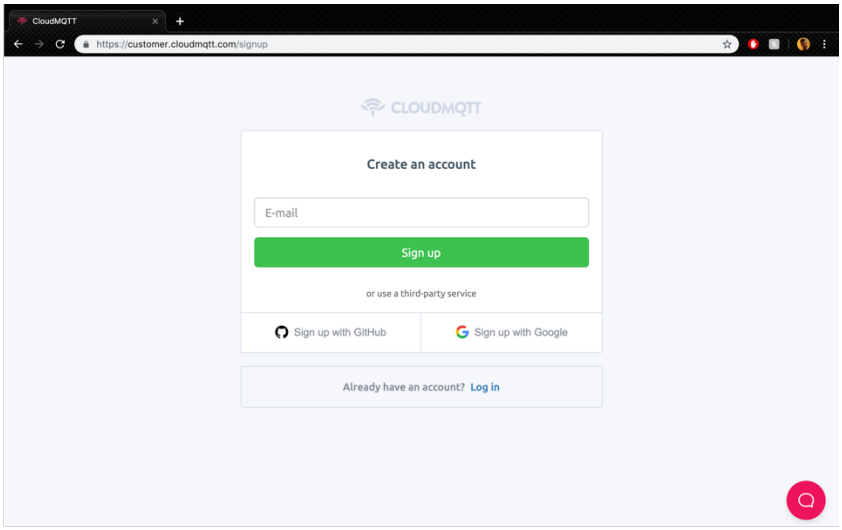


Slika 3.2. Početna stranica CouldMQTT-a.

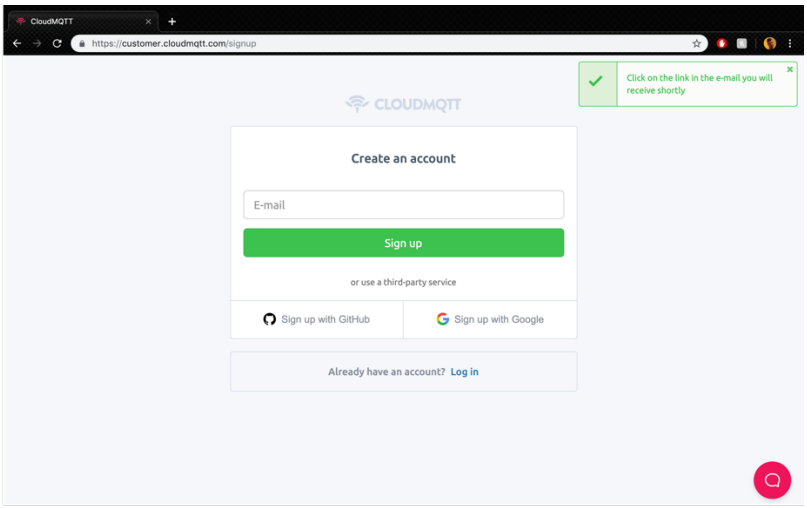


Slika 3.3. Log in stranica CloudMQTT.

U podnožju stranice stisnite Sign up dugme. Otvara se stranica prikazana na slici 3.4.. Unesite svoj email (služi za prijavu) i ponovno pritisnete Sign up dugme. U gornjem desnom kutu stranice pojavljuje se obavijest - provjerite svoj email. Kliknite na link koji se nalazi u mailu kao što je to prikazano na slici 3.5..

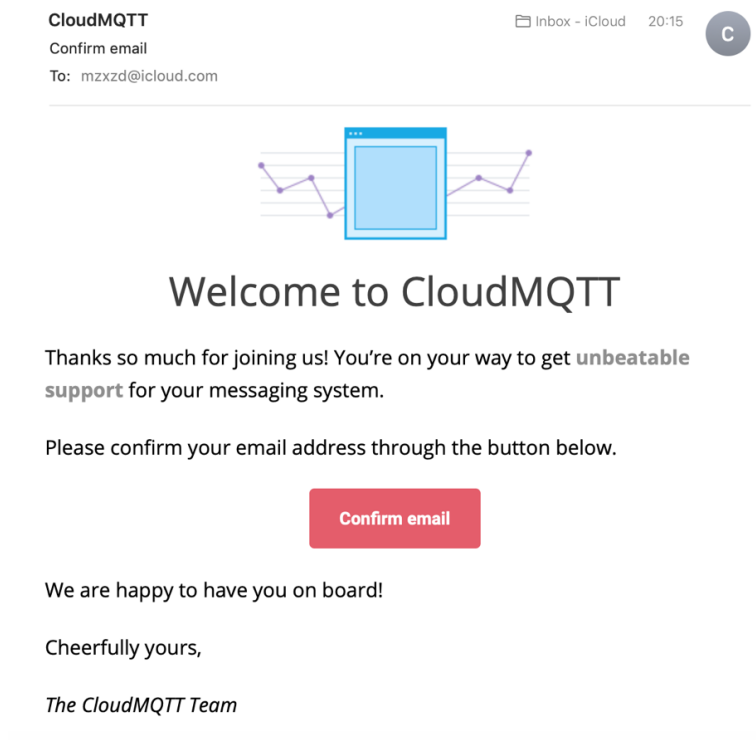


Slika 3.4. Sign up stranica CloudMQTT.

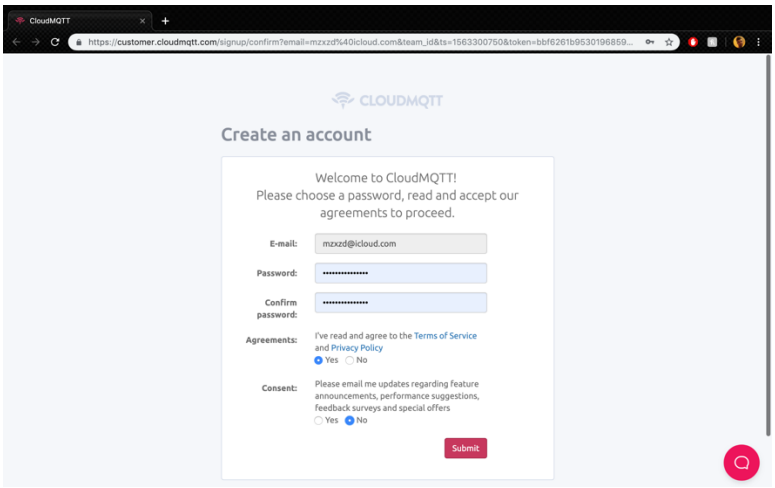


Slika 3.5. Sign up stranica CloudMQTT-a sa email obavijesti.

U pretincu email-a nalazi se poruka dobrodošlice. Ujedno se traži potvrda email-a unesenog kod prijave kao što prikazuje slika 3.6.. Pritiskom na dugme *Confirm email* otvara se stranica prikazana na slici 3.7.. Potrebno je unijeti lozinku, te potvrditi Izjavu o pravu prikupljanja, obrađivanja i korištenja Vaših osobnih podataka i zaštite privatnosti. Klikom na *Submit* izrađen je korisnički račun.

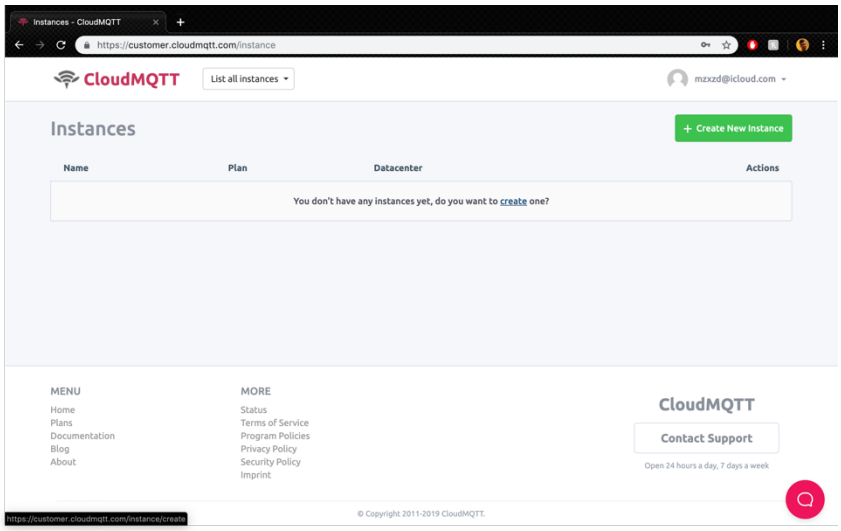


Slika 3.6 Email od CloudMQTT-a.

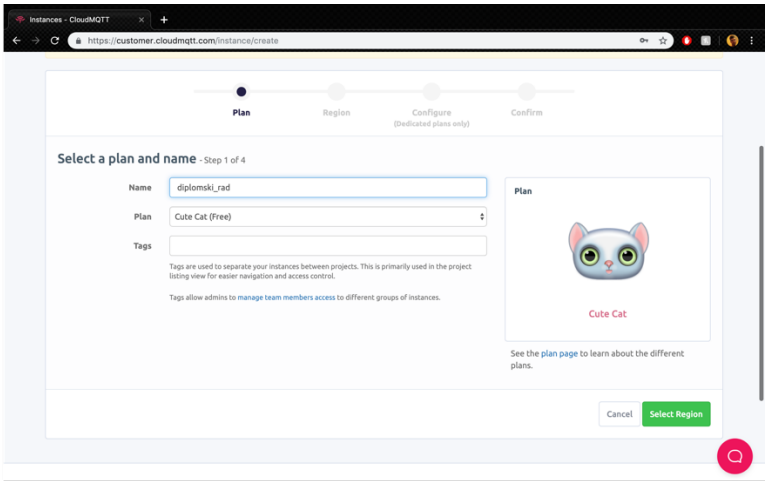


Slika 3.7. Sign up stranica CloudMQTT-a sa unosom lozinke.

Nakon kreiranja korisničkog računa, potrebno je kreirati instancu servera koji će se koristiti u ovom radu. Slika 3.8. prikazuje početnu stranicu nakon prijave. Klikom na Create New Instance, otvara se stranica kao na slici 3.9.. Na toj stranici potrebno je napisati ime instance, te izabrati plan. Odabirom dugmeta *Select region* otvara se stranica na slici 3.10..

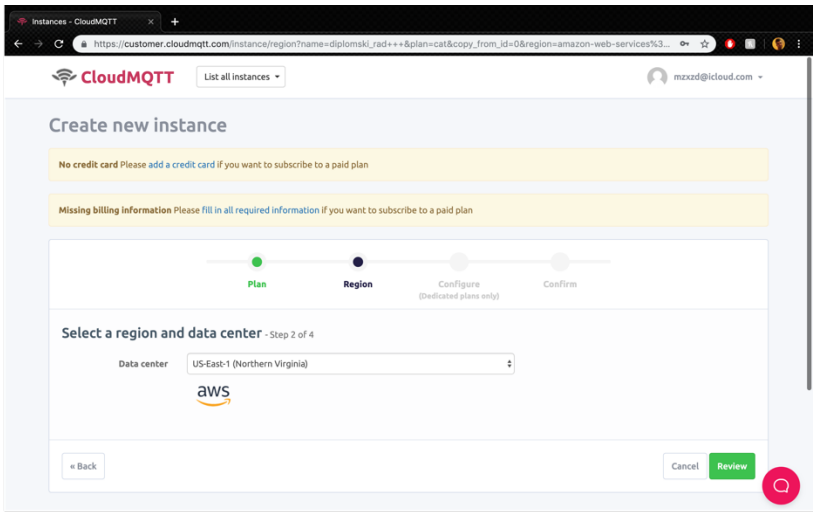


Slika 3.8. Početna stranica CloudMQTT-a.

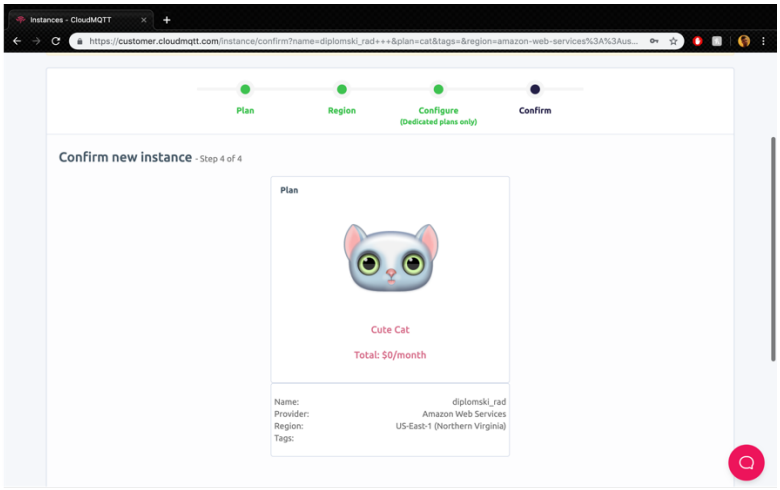


Slika 3.9. Odabir plana i imena.

U odabiru regije servera potrebno odabrati bilo koji server koji je dostupan, te nakon toga stisnuti dugme *Rewiew* koje otvara stranicu prikazanu na slici 3.11.. Na stranici je prikazana informacija o novo kreiranoj instanci. Nakon toga stranica se vraća na početnu stranicu sa pristupom na tu instancu.

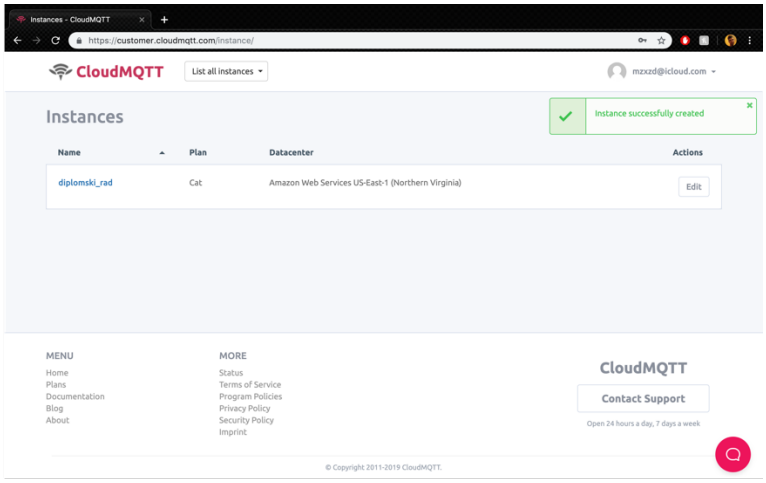


Slika 3.10. Odabir regije servera.

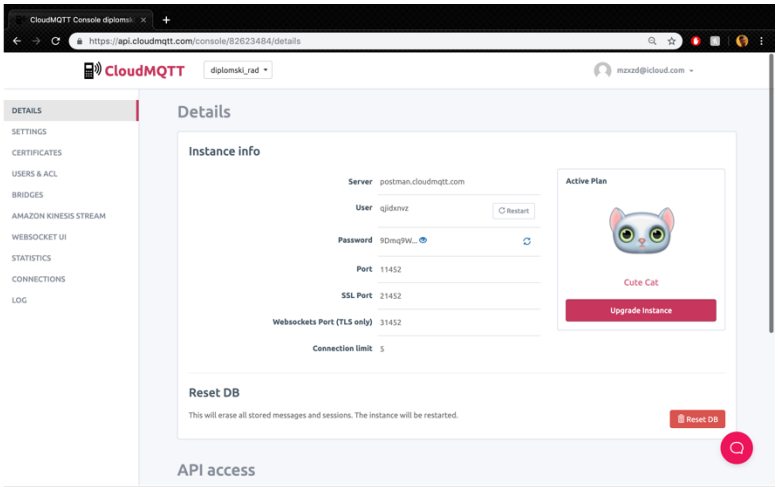


Slika 3.11. Odabir plana korištenja online usluge.

Na početnoj stranici kreirana je instanca što je prikazano na slici 3.12. Pritiskom na ime instance stranica otvara detalje te instance kao što je prikazano na slici 3.13. Stranica sadrži nekoliko bitnih podataka koji se koriste u kodu za povezivanje na tu instancu (npr. Server, User, password, i port). Potrebno je još kreirati *topic*-e i klijente, pa klikom na *Users and ACL* dugme otvara se prozor prikazan na slici 3.14..

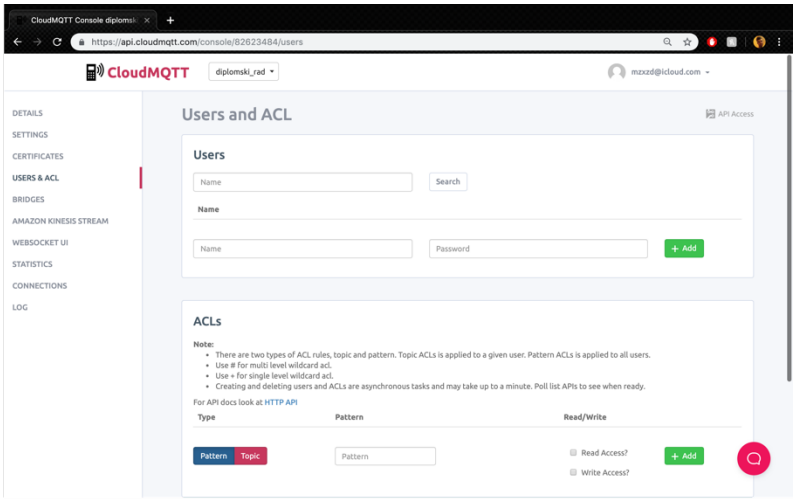


Slika 3.12. Početna stranica CloudMQTT sa novo dodanom instancom.

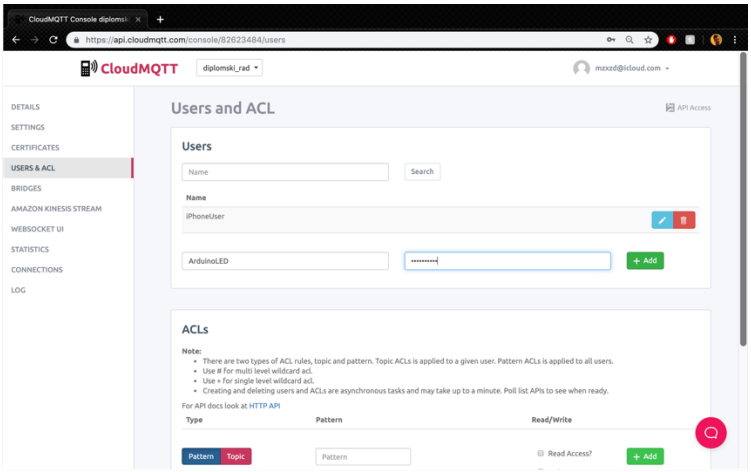


Slika 3.13. Detalji odabrane instance.

U ovom prozoru dodaju se klijenti i *topic*-i. Slika 3.15. prikazuje dodavanje klijenta za Arduino koji će biti odgovoran za LED. Pritiskom na dugme *Add*, taj klijent je dodan u bazu podataka.

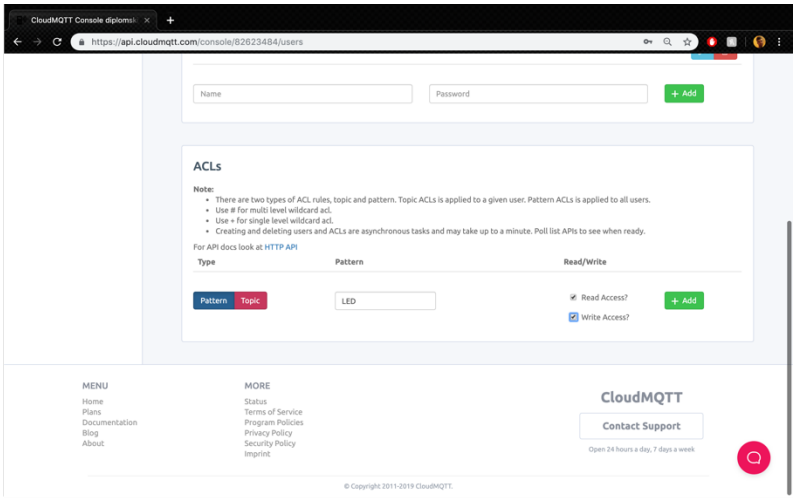


Slika 3.14. *Users and ACL* prozor.

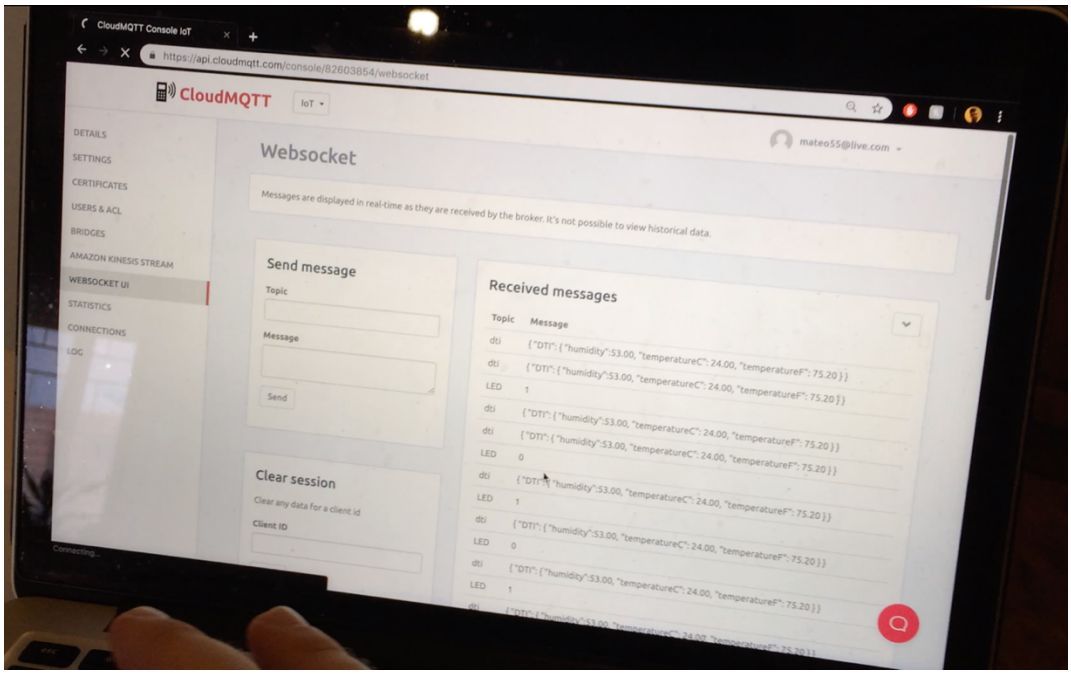


Slika 3.15. Dodavanje novog klijenta.

ACL dio prozora je odvojen za dodavanje *topic*-a. Slika 3.16. prikazuje dodavanje naziva teme (Pattern), a samo pritiskom na *Add* dodaje se novi *topic* za komunikaciju. Primjer komunikacije između *topic*-a se nalazi na slici 3.17..



Slika 3.16. Dodavanje novog topic-a.



Slika 3.17. Primjer ispisa razmijenjenih poruka.

3.4. Programski dio sustava

3.4.1. iOS podsustav

U iOS sustavu potrebno je kreirati *view* koji klijentu omogućuje kontrolu tih uređaja. Kako bi se to realiziralo, kreiran je Coordinator protokol koji služi za navigaciju aplikacije na *view*. Kod 3.1. prikazuje Coordinator protokol, a kod 3.2. implementaciju tog protokola u klasi koja se koristi za pokretanje početnog *view*-a.

```
public protocol Coordinator: class{
  var childCoordinators: [Coordinator] { get set}
  var presenter: UINavigationController { get}
  func start()
}

public extension Coordinator {

  /// Add a child coordinator to the parent
  func addChildCoordinator(childCoordinator: Coordinator) {
    self.childCoordinators.append(childCoordinator)
  }

  /// Remove a child coordinator from the parent
  func removeChildCoordinator(childCoordinator: Coordinator) {
    self.childCoordinators = self.childCoordinators.filter { $0 !== childCoordinator }
  }

}
```

Kod 3.1. Coordinator protokol.

```
class HomeCoordinator: Coordinator{
  var childCoordinators: [Coordinator] = []
  var presenter: UINavigationController
  let controller: HomeViewController
  init (presenter: UINavigationController){
    self.presenter = presenter
    let controller = HomeViewController()
    let homeViewModel = HomeViewModel()
    controller.viewModel = homeViewModel
    self.controller = controller

  }
  func start() {
    presenter.pushViewController(controller, animated: true)
  }

}
```

Kod 3.2. HomeCoordinator koji nasljeđuje Coordinator protokol.

Nakon implementacije Coordinatora, dodaju se elementi koji prikazuju i omogućuju korisniku manipulaciju vrijednostima. Ti elementi su dugme (eng. *button*), klizač (eng. *slider*), *ImageView* i *Label*. Kod 3.3. prikazuje dio inicializacije tih elemenata.

```
  var celsiusLabel: UILabel = {
    let label = UILabel()
    label.translatesAutoresizingMaskIntoConstraints = false
    label.text = "Temperature in °C:"
    return label
  }()


  var temperatureImageView : UIImageView = {
    let view = UIImageView()
    view.image = imageLiteral(resourceName: "icons8-temperature-50")
    view.translatesAutoresizingMaskIntoConstraints = false
    return view

  }()
```

Kod 3.3. Inicializacija UI elemenata.

Nakon postavljanih elemenata, uspostavlja se veza prema MQTT *broker*-u koji se nalazi na internetu. U kodu 3.4. prikazana je inicializacija, kao i povezivanje na server.

```
import Moscapsule


  let mqttConfig = MQTTConfig(clientId: "ClientID", host:"m24.cloudmqtt.com", port: 15785,
keepAlive: 60)

    mqttConfig.mqttAuthOpts = MQTTAuthOpts(username: "username", password: "password")

    mqttConfig.onPublishCallback = { messageId in
      NSLog("published (mid=\(messageId))")
    }
    mqttConfig.onMessageCallback = { mqttMessage in
      print(mqttMessage.topic)
          // ovdje dolazi implementacija fukcije koja parse-a podatke sa Croduina kako bi se moglo
      prikazati na ios uređaju
                self.serialize(message: mqttMessage)

    }

    let mqttClient = MQTT.newConnection(mqttConfig)

        mqttClient.subscribe("dti", qos: 1)
```

Kod 3.4. Implementacija MQTT koda za povezivanje na MQTT server.

Kod 3.5. prikazuje primjer koda koji šalje (*publish*-a) podatke na server na željeni *topic*. U ovom primjeru prikazana je (*publish)* vrijednost za ventilator kao i za LED svjetlo.

```
  func triggerFanSpeed(speed: Float) {
    let scaledFanValue: Int = Int((1023 - 0) * (speed - 0) / (100 - 0))
    self.mqttClient.publish(string: String(scaledFanValue), topic: "fan", qos: 1,
retain: false)
  }



  func triggerLED() {
    if LED {
      self.mqttClient.publish(string: "0", topic: "LED", qos: 1, retain: false)
      self.LED = !LED
    }
    else {
      self.mqttClient.publish(string: "1", topic: "LED", qos: 1, retain: false)
      self.LED = !LED
    }

  }

}
```

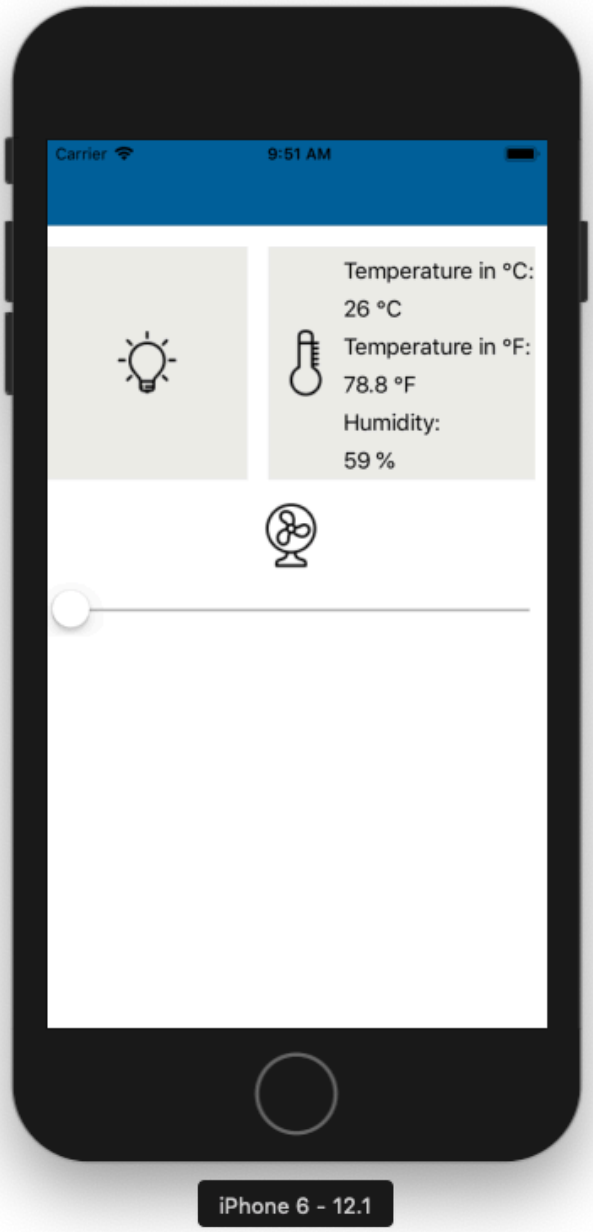
Kod 3.5. Implementacija *publish* funckije.

Kada aplikacija primi podatke o temperaturi i vlazi u zraku, JSON String objekt se *parse*-a. Kod 3.6. prikazuje implemetiranje *parsa*-nja JSON String objekta u SWIFT objekt. Vrijednosti se odmah spremaju u UI *Label* kako bi klijent vidio vrijednosti koje dobiva sa senzora.

```
func serialize(message: MQTTMessage) {

    switch message.topic {
     case mqttTopics.Led.rawValue:
       break
     case mqttTopics.Dti.rawValue:
       let jsonDecoder = JSONDecoder()
       if let dataToDecode = message.payloadString?.data(using: .utf8) {
         do {
           let data = try jsonDecoder.decode(DTIObject.self, from: dataToDecode)
              self.humidity.value = String(data.dti.humidity) + " %"
              self.temperatureC.value = String(data.dti.temperatureC) + " °C"
              self.temperatureF.value = String(data.dti.temperatureF) + " °F"

           } catch let error {
            print(error.localizedDescription)
           }
         }
     case mqttTopics.Fan.rawValue:
       break
      default:
       print(message.topic)
   }

}
```

Kod 3.6. Implementacija metode parsiranja za temperaturu i vlagu u zraku.

Na kraju, kada je sve implementirano, aplikacije na iOS uređaju izgleda kao što je prikazano na slici 3.3.



Slika 3.3. Prikaz aplikacije prilikom rada.

3.4.2. Arduino podsustav

Da bi se implementirao Corduino potrebno ga je povezati na Wifi a za tim na MQTT server. Kod 3.7. prikazuje kod za implementaciju navedenoga.

```
#include <MQTT.h>
#include <ESP8266WiFi.h>

#define BROKER_IP  "m24.cloudmqtt.com"
#define DEV_NAME   "dev_name"
#define MQTT_USER  "username"
#define MQTT_PW   "password"

const char ssid[] = "ssid";
const char pass[] = "password";

WiFiClient net;
MQTTClient client;
unsigned long lastMillis = 0;

void connect() {
 Serial.print("checking wifi...");
 while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");
  delay(1000);
 }

 Serial.print("\nconnecting...");
 while (!client.connect(DEV_NAME, MQTT_USER, MQTT_PW)) {
  Serial.print(".");
  delay(1000);
 }
 Serial.println("\nconnected!");
 client.subscribe("topic"); //SUBSCRIBE TO TOPIC /topic
}
```

Kod 3.7. Arduino implementacija povezivanja na Wifi i na MQTT server.

Nakon implementacije, dodaje se fukcija koja rukuje (eng. *handle*) sa dolaznim podatcima sa MQTT *broker*-a. Kod 3.8. prikazuje primjer implementacije funkcije za upravljanje brzine ventilatora. Poziva se nova funkcija unutar funkcije koja rukuje sa dolazećim podatcima.

```
void messageReceived(String &topic, String &payload) {
 Serial.println("incoming: " + topic + " - " + payload);
  message(payload);
}


void message(String payload) {
 Serial.println(payload);
 analogWrite(13, payload.toInt());

}
```

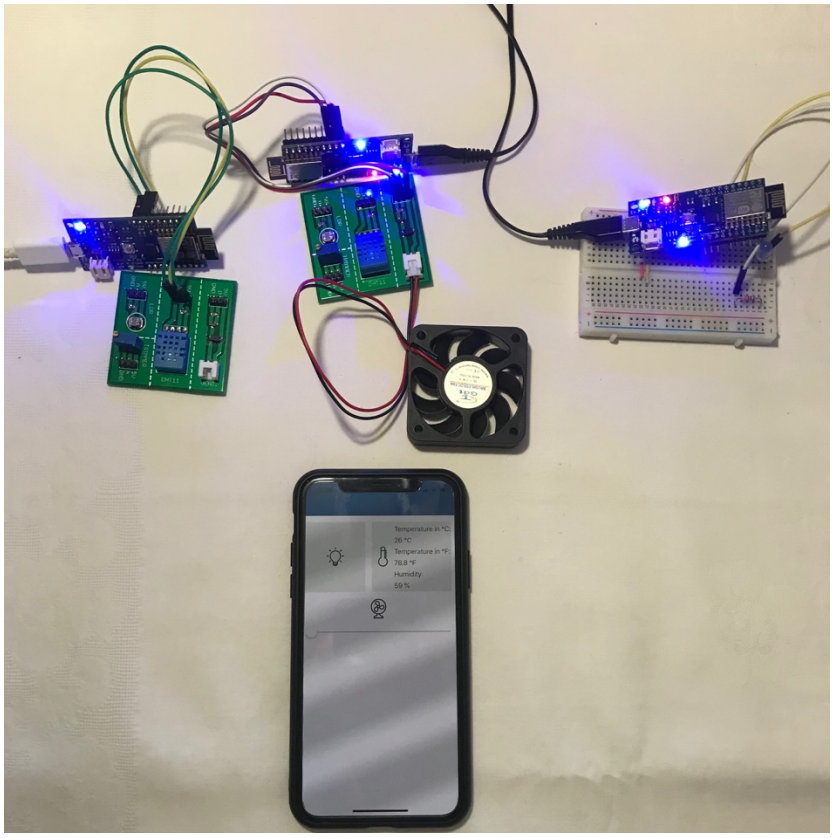
Kod 3.8. Implementacija funkcije za rukovanje dolaznim podatcima.

Nakon implementacije funkcije za rukovanje dolaznim podacima, u *setup* funkciju dodaju se pokretači Wifi objekta kao i MQTT klijent objekta kako bi pokrenuli povezivanje. Loop funkcija provjerava da li je Croduino spojen ili ne. To je prikazano u kodu 3.9.

```
void setup() {
 Serial.begin(115200);
 WiFi.begin(ssid, pass);
 pinMode(13, OUTPUT);
 client.begin(BROKER_IP, 15785, net);
 client.onMessage(messageReceived);
 connect();
}

void loop() {
 client.loop();
 if (!client.connected()) {
  connect();
 }

}
```

Kod 3.9. Pokretanje Wifi i MQTT servisa u *setup* funkciji i *loop* funkciji.

Na kraju izrade programske podrške, testirani su Croduino uređaji kao i iOS aplikacija kako bi se utvrdila realizacija koncepta upravljanja preko interneta. Kako se vidi na slici 3.4., udaljeno upravljanje uspješno je realizirano, a video upravljanja Croduino uređaja preko iOS aplikacije nalazi se u prilogu.



Slika 3.4. Fotografija testiranja aplikacije i Croduino uređaja.

4. ZAKLJUČAK

U ovom radu izrađena je platforma s iOS aplikacijom za upravljanje Arduinom preko interneta. Realizacija rada je uspješna, od kontrole LED svjetla, brzine ventilatora pa do primanja podataka od senzora za temperaturu i vlagu u zraku. Ostvarena je internetska komunikacija između iOS uređaja i Arduino razvojnog sustava preko MQTT servera. Dobivene informacije, poslane sa iOS uređaja uspješno prenose podatke na Arduino (i obrnuto) preko Wifi modula koji je dio Croduino Nova uređaja. Podaci se precizno očitavaju i pretvaraju u signale za obradu i prikaz (primjer temperatura i vlaga u zraku). Koncept platforme s iOS aplikacijom za upravljanje Arduinom preko interneta je uspješno realiziran. Za izradu te platforme korišteno je nekoliko tehnologija (MQTT, Moscapsule, Swift, Cocoapods, itd…). Kreirana je iOS aplikacija u Swift programskom jeziku koja komunicira sa MQTT serverom na globalnoj domeni. Programiran je Arduino (Croduino Nova) sa kodom koji omogućuje komunikaciju sa MQTT serverom te na osnovu dobivene poruke izvršava zadane radnje. Svi navedeni kodovi nalaze se u prilogu. Završna testiranja sa uređajima prošla su bez poteškoća. Platrofma s iOS aplikacijom za upravljanje Arduinom preko interneta je uspješno realizirana, što se može vidjeti u prilogu.

LITERATURA

[1] M Došlić, Udaljeno upravljana noćna lampa, 2015-2016

[2] Z. Webber, Arduino: The Ultimate Beginner's Guide to Learn and Understand Arduino Programming Effectively, 2018

[3] T. Pulver, Hands-On Internet of Things with MQTT, 2019

[4] M. Neuburg, iOS 12 Programming Fundamentals with Swift: Swift, Xcode, and Cocoa Basics, 2018

[5] 84codes AB, Sveavägen 98 11350 Stockholm Sweden, dostupno na: https://www.cloudmqtt.com/ [20.06.2019.g]

SAŽETAK

Platforma s IOS aplikacijom za upravljanje Arduinom preko interneta

Glavni zadatak diplomskog rada je realizacija platforme s iOS aplikacijom za upravljanje Arduinom preko interneta. Tehnologije korištene u ovo radu su: Xcode (IDE za razvoj aplikacije), MQTT server (server koji koristi MQTT protokol za komunikaciju između uređaja), Croduino (Hrvatska inačica Arduina), Cocoapods (Xcode biblioteka (eng. *library*) za proširenje programskog sustava) i Swift (programski jezik korišten za programiranje iOS uređaja). Realizirano je slanje i primanje podataka na iOS uređaj kao i na Arduino uređajima preko MQTT servera. Ti podatci omogućuju upravljanje LED-om i brzinu okretaja ventilatora, te šalju podatke iOS uređaju o trenutnoj temperaturi i vlazi u zraku.

Ključne riječi: Arduino, iOS, Cocoapods, Swift, udaljeno upravljanje, MQTT, Apple

ABSTRACT

Title: Platform with IOS app for Arduino management over the Internet

The main task of the graduate thesis is the implementation of iOS applications platform to manage Arduino over the Internet. Technologies used in this work environment are: Xcode (application development IDE), MQTT server (server using MQTT communication protocol between devices), Croduino (Croatian version of Arduino), Cocoapods (Xcode library for expanding program system) and Swift (programming language for iOS device). Sending and receiving data on the iOS device as well as on Arduino devices via the MQTT server is accomplished. These data allow for LED control and fan speed, as well as data are storing current temperature and humidity in the air.

Keywords: Arduino, iOS, Cocoapods, Swift, remote management, MQTT, Apple

ŽIVOTOPIS

Mateo Došlić rođen je 10.09.1994. g. u Frutigenu u Švicarskoj. Prve 4 godine života živi u Kansterstegu, u Švicarskoj. U Požegi stječe osnovnoškolsko obrazovanje u OŠ Antuna Kanižlića. Srednju tehničku školu, smjer Tehničar za telekomunikacije, završava u Požegi. U srednjoj školi je sudjelovao u međunarodnom natjecanju iz programiranja ASCL. Osvojio je prvo mjesto na državnom natjecanju u Zagrebu 2012. g. s nazivom Opisujemo sustave. Od 2013 studira na FERIT-u. Od početka školovanja pa do danas, sve je godine prošao s vrlo dobrim ili odličnim uspjehom.

Mateo Došlić, mag.ing.comp.

PRILOZI

Video testiranja, iOS kod, Arduino kod nalaze se na CD-u, kao i slike rada.