Sustav Interneta stvari za mjerenje i prikaz vrijednosti senzora koristeći Waspmote i Pycom
Tehnička dokumentacija
Verzija <1.0>

Studentski tim: Laura Abramović
Filip Đuran
Benjamin Horvat
Domagoj Kolega
Luka Lacković
Josip Lukačević
Borna Majstorović
Ana Mrkonjić
Mihael Rodek

Nastavnik: prof. dr. sc. Mario Kušek

Sadržaj

1. Opis razvijenog proizvoda 4

2. Arhitektura sustava 4

LoRaWAN 7

3. Narrowband IoT 7

4. Senzori 7

5. Poslužitelj 7

6. Android aplikacija 7

7. iOS aplikacija 7

Primjeri modela. 7

8. Literatura 11

Tehnička dokumentacija

Na koji način koristiti predložak?

Dokument se po potrebi može prilagoditi potrebama pojedinog projekta promjenom predloženih naslova predloženih poglavlja, kao i eventualnim dodavanjem novih poglavlja i potpoglavlja.

Cilj dokumenta je opisati rezultat rada studentskog tima, problem koji je riješen u okviru projekta, korištenu tehnologiju, mogućnosti i značajke dobivenog proizvoda i sl. Razinu detalja opisanu u ovom dokumentu studentski tim treba dogovoriti s nastavnikom.

Literatura:
U tekstu rada treba biti navedena literatura svugdje gdje je tekst, slika ili grafički prikaz preuzet ili se temelji na nekom pisanom predlošku. Literatura se navodi iza zaključka. U tekstu se literatura navodi unutar zagrada s navođenjem prvog autora i godine izdanja, npr. (Martinis, 1998).

Primjer citiranja knjige:
Prezime, inicijal(i) imena autora. Naslov: podnaslov. Podatak o izdanju. Mjesto izdavanja: Nakladnik, godina izdavanja.

Primjer citiranja članka u časopisu:
Prezime, inicijal(i) imena autora. Naslov članka: podnaslov. Naziv časopisa. Oznaka sveska/godišta, broj(godina), str. početna-završna.

Primjer citiranja rada sa konferencije:
Prezime, inicijal(i) imena autora. Naslov rada: podnaslov. Naslov zbornika, mjesto održavanja konferencije, (godina), str. početna-završna.

Primjer citiranja doktorskog, magistarskog ili diplomskog rada:
Prezime, inicijal(i) imena autora. Naslov. Vrsta rada. Ustanova na kojoj je rad obranjen, godina.

Primjer citiranja www izvora:
Ime(na) autora (ako je/su poznata), naslov dokumenta, datum nastanka (ako se razlikuje od datuma pristupa izvoru), naslov potpunog djela (italic), potpuna http adresa, datum pristupa dokumentu.

Ostale upute
U svim dokumentima obvezno primjenjivati SI jedinice. Slike, formule i tablice potrebno je numerirati. Opis tablice stavlja se iznad, a opis slike ispod nje. U opisu slike ili tablice pišu se samo podaci neophodni za njeno razumijevanje (npr. Slika 6. Pojačalo s promjenljivim pojačanjem). Dodatna objašnjenja daju se u tekstu uz povezivanje sa slikom ili tablicom. Osi i parametri na slikama i grafičkim prikazima trebaju biti obilježeni. Daljnji opis tog grafičkog prikaza treba se nalaziti u tekstu rada. Formule se obilježavaju brojevima u zagradi, uz desni rub stranice, a u tekstu se poziva na broj formule.

1. Opis razvijenog proizvoda

- opis projektnog zadatka
- funkcionalnosti razvijenih aplikacija
- korištene tehnologije i komunikacijski protokoli

2. Arhitektura sustava

Arhitektura sustava može se podijeliti u nekoliko podsustava:

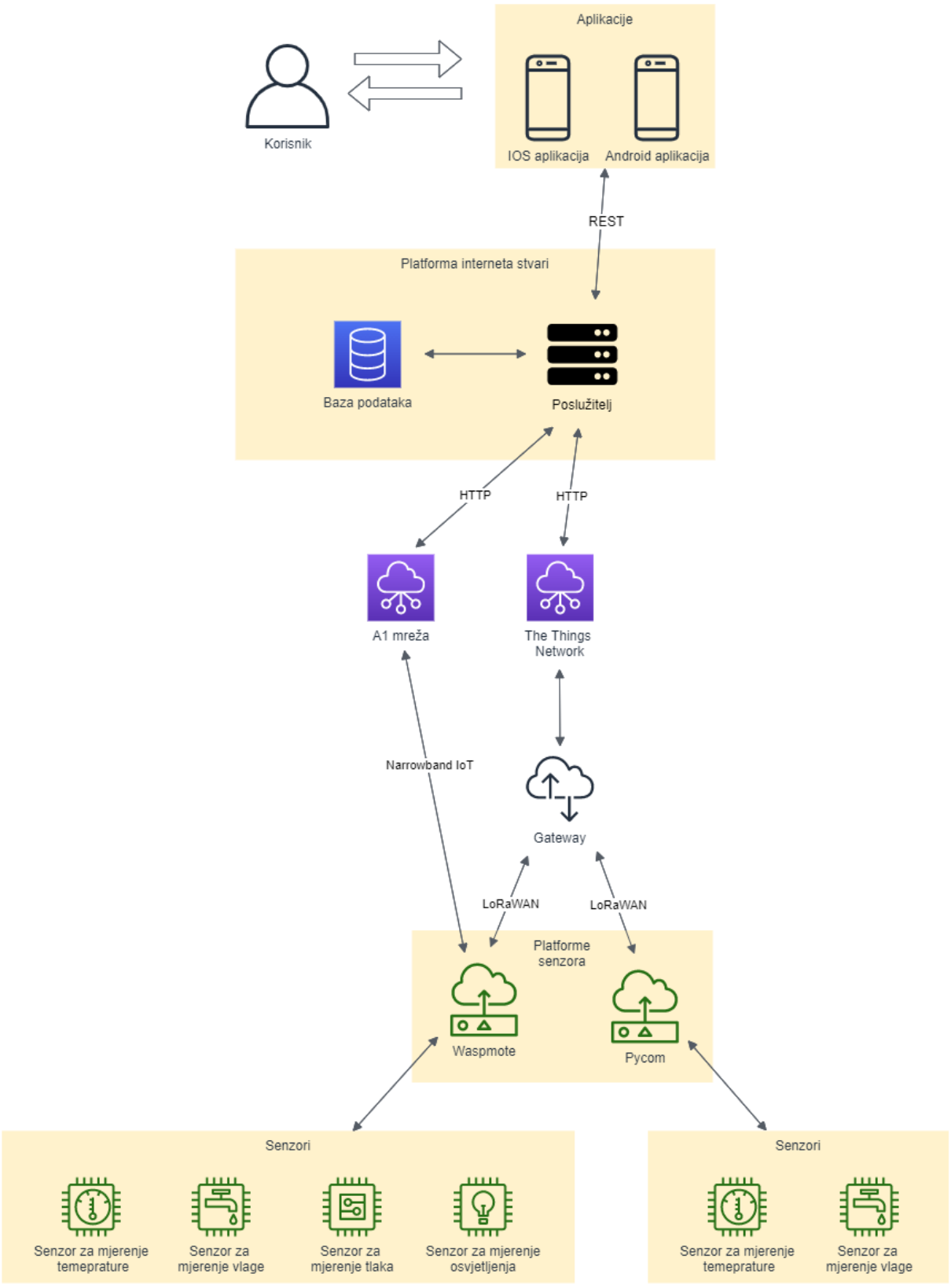
- Mobilna aplikacije
- Platforma interneta stvari
- Pametna okolina

Mobilna aplikacija (*eng. mobile application*) programska je potpora za mobilne uređaje koja korisniku omogućuje pregled sadržaja za koji je ta aplikacija namijenjena. Mobline aplikacije razvijene su okruženju Android Studio za uređaje bazirane na operacijskom sustavu Android, te u okruženju Xcode za uređaje bazirane na operacijskom sustavu iOS.

Platforma interneta stvari (*eng. IoT platform*) sadrži poslužitelj (*eng. server*) koji je osnova rada aplikacije. Poslužiteljski dio razvijen je u Java Springu. Klijent koristi aplikaciju za obrađivanje željenih zahtjeva. Aplikacija obrađuje zahtjev te ovisno o njemu pristupa bazi podataka nakon čega preko poslužitelja vraća odgovor klijentu. Komunikacija između klijenta i poslužitelja temelji se na REST arhitekturskom stilu i koristi se HTTP protokol.

Pametna okolina sastoji se od senzora, sklopovskih platformi i poveznika. Poveznik (engl. gateway) povezuje uređaje i poslužiteljski dio aplikacije. Senzori su uređaji koji mjere neku fizikalnu veličinu te ju pretvaraju u signal pogodan za daljnju obradu. Senzori se spajaju na sklopovske platforme koje su u našem slučaju Waspmote i PyCom platforme. Sklopovske platforme komuniciraju s poveznikom putem nekoliko komunikacijskih protokola, a to su: LoRaWaN, NB-IoT i LTE-M. Podaci koje senzori mjere u ovom projektu su temperatura i vlaga tla i zraka.

Arhitektura sustava prikazana je na slici 1.


Aplikacije
Korisnik
IOS aplikacija
Android aplikacija
REST
Platforma interneta stvari
Baza podataka
Poslužitelj
HTTP
HTTP
A1 mreža
The Things Network
Narrowband IoT
Gateway
LoRaWAN
LoRaWAN
Platforme senzora
Waspmote
Pycom
Senzori
Senzor za mjerenje temeprature
Senzor za mjerenje vlage
Senzor za mjerenje tlaka
Senzor za mjerenje osvjetljenja
Senzori
Senzor za mjerenje temeprature
Senzor za mjerenje vlage

Slika 1 Arhitektura sustava

LoRaWAN

- detaljnije objasniti protokol i način komunikacije s poslužiteljom
- koristeći primjere iz koda i The Things Network konzole, objasniti način implementacije protokola
- sekvencijski dijagram komunikacije uređaja s poslužiteljem

3. Narrowband IoT

- detaljnije objasniti protokol i način komunikacije s poslužiteljom
- koristeći primjere iz koda objasniti način implementacije protokola

4. Senzori

- navesti korištene senzore za pojedini uređaj
- za svaki uređaj pokazati način korištenja senzora u kodu

5. Poslužitelj

- opis REST API-ja
- dijagrami klasa
- opis baze podataka

6. Android aplikacija

- izbor i opis odabrane arhitekture
- izgled modela i servisa
- pojašnjenje važnijih dijelova koda

7. iOS aplikacija

iOS aplikacija rađena je za iOS 13.0, većina stvari rađena je sa native apple frameworkcima.
Za arhitekturu je korišten MVVM, gdje su servisi odvojeni od viewControllera te im se pristupa putem viewModela.
Modeli se dekodiraju u JSON pomoću protokola Decodable, dok se enkodiranje JSON-a za POST pozive radi direktno u pozivu sa JSONSeriliazation klasom.

```
struct MeasurementsModel: Decodable {
    let id: Int?
    let device: DeviceModel?
    let time: String?
    let airHumidity: Double?
    let soilHumidity: Double?
    let airTemperature: Double?
    let soilTemperature: Double?
    let pressure: Double?
}
```

```
class CulturesModel: Decodable {
    let cultureId: Int
    let title: String
    var devices: [DeviceModel]
    let description: String
}
```

Primjeri modela.

Primjer viewModela i servisa.

```
class CulturesViewModel {
    var cultures: [CulturesModel] = []
    var selectedCulture: CulturesModel?
    var selectedIndex: Int?

    func fetchCultures(completion: @escaping ((Result<Void, Error>)->Void)){
        let service = MeasurmentsService()

        service.fetchCultures { (result) in
            switch result {
            case .success(let cultures):
                self.cultures = cultures
                completion(.success(()))
            case .failure(let error):
                self.cultures = []
                completion(.failure(error))
            }
        }
    }

    func culturesForTVC(forIndexPath indexpath: IndexPath) -> CultureCellModel? {
        return CultureCellModel(culture: cultures[indexpath.row])
    }

    func deleteCulture(cultureID: Int){
        let service = MeasurmentsService()
        service.deleteCulture(cultureID: cultureID)
    }
}
```

```
class MeasurmentsService {

    let baseUrlString = Constants.baseUrl

    func fetchMeasurmentData(completion: @escaping ((Result<[MeasurementsModel], Error>)->Void)){

        let urlString = baseUrlString + "/api/measurement/all"
        guard let url = URL(string: urlString) else {return}
        guard let token = UserCredentials.shared.getToken() else {
            return
        }

        var request = URLRequest(url: url)
        request.httpMethod = "GET"
        request.addValue("Bearer \(token)", forHTTPHeaderField: "Authorization")

        URLSession.shared.dataTask(with: request) { (data,response,error) in
            if let data = data {
                do {
                    let model = try JSONDecoder().decode([MeasurementsModel].self, from: data)
                    completion(.success(model))
                } catch let decodeError {
                    completion(.failure(decodeError))
                    return
                }
            } else if let error = error {
                completion(.failure(error))
            }

        }.resume()
    }
```

Za networking je većinom korišten framework URLSession, samo za register i login je korišten Alamofire. Ispod je primjer POST (alamofire), POST (URLSession) i DELETE (URLSession) poziva.

```
extension LoginViewController {
    func loginUserWith(username: String, password: String) {

        let parameters: [String: String] = [
            "username": username,
            "password": password
        ]

        let urlStr = Constants.baseUrl + Constants.loginEndPoint

        AF.request(urlStr,method: .post, parameters: parameters, encoding: JSONEncoding.default)
            .validate()
            .responseDecodable(of: UserModel.self) { response in
                switch response.result {
                case .success(let user):
                    UserCredentials.shared.setToken(with: user.token)
                    self.userModel = user
                    self.navigateToHome()
                case .failure(let err):
                    print(err)
                }
            }
    }
}
```

```
 func deleteCulture(cultureID: Int) {
     let urlString = baseUrlString + "/api/culture/delete/\(cultureID)"
     guard let url = URL(string: urlString) else {return}

     guard let token = UserCredentials.shared.getToken() else {return}

     var request = URLRequest(url: url)
     request.httpMethod = "DELETE"
     request.addValue("Bearer \(token)", forHTTPHeaderField: "Authorization")

     URLSession.shared.dataTask(with: request) { data, response, error in
         if let data = data {
             print(data)
         } else if let err = error {
             print(err.localizedDescription)
         } else if let response = response {
             print(response)
         }
     }.resume()
 }
```

```swift
func addCulture(cultureID: String, title :String, deviceID: String, deviceDevID: String, description: String) {

    let urlString = baseUrlString + "/api/culture/add"
    guard let url = URL(string: urlString) else {return}

    guard let token = UserCredentials.shared.getToken() else {return}

    let data = [
        "cultureId": cultureID,
        "title": title,
        "devices": [
            ["id": deviceID,
            "devId": deviceDevID]
            ],
        "description": description
    ] as [String : Any]

    guard let jsonData = try? JSONSerialization.data(withJSONObject: data, options: []) else { return }

    var request = URLRequest(url: url)
    request.httpMethod = "POST"
    request.addValue("Bearer \(token)", forHTTPHeaderField: "Authorization")
    request.addValue("application/json", forHTTPHeaderField: "Content-Type")
    request.httpBody = jsonData

    URLSession.shared.dataTask(with: request) { data, response, error in
        if let data = data {
            print(data)
        } else if let err = error {
            print(err.localizedDescription)
        }
    }.resume()
}
```

Za crtanje grafova korišten je pod Charts, u metodu setupCharts se proslijedi polje Double-ova koje predstavlja različita mjerenja za određenu kulturu i senzor.
UserCredentials je singleton čija je jedina dužnost pratiti jedinstveni token korisnika.

```swift
func setUpCharth(array: [Double]){
    chartView.delegate = self

    chartView.frame = CGRect(x: 0, y: 0, width: 350, height: 400)
    chartView.center = view.center
    view.addSubview(chartView)

    let xAxisLimit = ChartLimitLine(limit: 10)
    xAxisLimit.lineWidth = 2

    let leftAxis = chartView.leftAxis
    leftAxis.removeAllLimitLines()
    leftAxis.axisMinimum = 0
    leftAxis.axisMaximum = 60
    chartView.rightAxis.enabled = false

    chartView.legend.form = .line

    var entries = [ChartDataEntry]()

    for x in 0 ..< array.count {
        entries.append(ChartDataEntry(x: Double(x), y: Double(array[x])))
    }

    let set = LineChartDataSet(entries: entries, label: "Temperature")
    set.colors = ChartColorTemplates.pastel()

    set.fillColor =
    set.drawFilledEnabled = true

    let data = LineChartData(dataSet: set)
    chartView.data = data
}
```

```swift
final class UserCredentials {
    private var token: String?

    private init(){}
    static let shared = UserCredentials()

    func setToken(with token: String){
        self.token = token
    }

    func getToken() -> String? {
        return token
    }
}
```

8. Upute za korištenje

- inicijalizacija uređaja sa senzorima
- pokretanje poslužitelja
- instalacija i korištenje aplikacija

8. Literatura