Primljen: 11.09.2023. Stručni rad
Prihvaćen: 22.12.2023. UDK: 004

Izrada aplikacije za web stvari

Development of Web of the Things application

[1]Filip Hrstić, [2]Darko Andročec
[1,2]Fakultet organizacije i informatike, Pavlinska 2, 42000 Varaždin, Hrvatska
e-mail: [1]fhrstic@student.foi.hr, [2]dandrocec@foi.unizg.hr

Sažetak: *Tema rada razvoj je aplikacije za web stvari. U prvom dijelu rada detaljno je razrađena teorijska podloga vezana uz internet stvari. Uključuje povijest interneta stvari, opis osnovnih karakteristika i arhitekture. Navedeni su najpoznatiji uređaji i senzori s naglaskom na one, korištene u praktičnom dijelu rada. Pojam interneta stvari bilo je potrebno pobliže objasniti jer je usko vezan uz pojam weba stvari. Nakon podrobnoga presjeka navedenih pojmova opisan je praktični dio. Sastoji se od programiranja IoT uređaja i ostvarivanja interakcije uređaja s Firebase platformom. Temelj praktičnoga dijela razvoj je aplikacije korištenjem Flutter okvira. Aplikacija je realizirana kao centralna, potpuno modularna korisnička jedinica za daljinsko upravljanje IoT uređajima.*

Ključne riječi: *internet stvari, web stvari, interoperabilnost, aplikacija za web stvari*

Abstract: *The topic of this paper is the development of the Web of Things application. In the first part of the paper, the theoretical background related to the Internet of Things is elaborated in detail. It includes the history of the Internet of Things, a description of basic features and architecture. Some most used devices and sensors are listed, with an emphasis on those used in the practical part of the paper. The notion of the Internet of Things needed to be explained in more detail because it is closely related to the notion of the Web of Things. After a detailed cross-section of the above terms, the practical part is described. It consists of programming IoT devices and interacting devices with the Firebase platform. The basis of the practical part is the development of the application using the Flutter framework. The application is realized as a central, fully modular user unit for remote control of IoT devices.*

Keywords: *internet of things, web of things, interoperability, web of things application*

1. Uvod

Internet stvari relativno je novi koncept čija primjena se razvojem tehnologije eksponencijalno povećava. Koristi se gotovo u svim sferama života i danas je pristupačan kao nikada ranije. Upravo su se zbog nekontroliranoga rasta interneta stvari izrodili problemi interoperabilnosti, kao i nedostatak standardiziranih metoda i protokola. Rješenje toga problema pronađeno je u već postojećim i stabilnim web standardima zbog čega nastaje pojam weba stvari. Sve veća važnost interneta i weba stvari u današnjem vremenu primarna je motivacija za izradu ovoga rada. U nastavku rada objasnit će se sličnosti i razlike tih dvaju pojmova. Koristeći stečena znanja iz teorijskoga dijela rada, implementiran je jednostavan sustav pametnoga doma koristeći pristupačne razvojne ploče i senzore. Poštujući arhitekturu weba stvari, uređajima je omogućena komunikacija s web servisima i postignuto je daljinsko

upravljanje uređajima putem mobilne aplikacije. Flutter je sve popularniji Googleov okvir za razvoj višeplatformskih mobilnih aplikacija visokih performansi. U konačnici, cilj rada shvatiti je pojam weba stvari te pomoću naučenih koncepata razviti pametni dom i mobilnu aplikaciju za upravljanje istim.

2. Internet stvari

2.1. Povijest interneta stvari

Premda je pojam interneta stvari relativno nov, njegova popularnost i rastuća primjena učinile su da pojam vrlo brzo postane općepoznat. Stručnjaci tvrde da se pojam internet stvari pojavio krajem prvoga desetljeća 21. stoljeća, između 2008. godine i 2009. godine. Procjenjuje se da je u tom razdoblju broj povezanih uređaja na internet premašio broj ljudi povezanih na internet, čime se simbolično obilježava početak interneta stvari.

Unatoč tome vrijedi istaknuti najraniji oblik pametnoga uređaja koji je razvijen čak 1982. godine na sveučilištu Carnegie Mellon u Sjedinjenim Američkim Državama. Riječ je o automatu Coca-Cole koji je putem ARPANET-a bio sposoban izvještavati o količini proizvoda u automatu kao i informirati o tome je li piće hladno ili nije (Wikipedia contributors, 2021a). No, prvo spominjanje pojma i njegova konceptualizacija pripisuje se Peteru Lewisu u jednom od njegovih javnih govora u rujnu 1985. godine. On je pojam interneta stvari definirao kao integraciju ljudi, procesa i tehnologije sa spojivim uređajima i senzorima čime se omogućuje daljinsko upravljanje i nadzor takvih uređaja (“Correcting the IoT History,” 2017).

Ideje i vizije međusobne integracije zasebnih uređaja i njihovo udaljeno upravljanje postoje odavno, a razvoj tehnologije i popularizacija interneta to su konačno i omogućile. Početkom 21. stoljeća pojam interneta stvari se počinje sve češće koristiti, a već tijekom nekoliko godina se ubrzano razvijaju čitavi sustavi, sastavljeni od različitih tehnologija, koji ostvaruju sve potencijale interneta stvari.

2.2. Osnovne karakteristike

Kao što je već spomenuto, pojam internet stvari definira se kao mreža povezanih fizičkih objekata koja zajedno sa softverom čini svrsishodan sustav. Stvari tako mogu međusobno komunicirati putem raznih tehnologija, a jedna od njih je i internet. O njemu će biti veći fokus u odnosu na izravnu komunikaciju između dva uređaja, takozvanu M2M komunikaciju (engl. *Machine to Machine*). Ovakvi sustavi mogu biti iznimno veliki, sačinjeni od velikoga broja integriranih elemenata, koji se upotrebljavaju u različite svrhe. Osnovna prednost ovakvoga sustava je što "stvar" prima vrlo precizne informacije iz okoline, samostalno ih obrađuje i šalje gdjegod je potrebno raznim protokolima. Sve se odvija bez ljudske interakcije, bez ljudskoga faktora koji može donositi pogreške, biti manje precizni i vremenski ograničeni. Ta "stvar" može biti senzor tlaka u automobilskoj gumi koji će upozoriti vozača u slučaju niskoga tlaka u gumama, može biti detektor dima i smrtonosnih plinova, životinja koja ima GPS lokator u svojoj ogrlici ili osoba s dijabetesom koja na ruci ima uređaj za mjerenje razine glukoze u krvi. Dakle, to može biti svaki objekt kojem se može dodijeliti IP adresa, odnosno svaki objekt koji ima sposobnost spojiti se na mrežu i slati podatke. Objektima se može dodijeliti i jedinstveni indentifikator ili oznaka (engl. *tag*) koja se može pročitati putem RFID (engl. *Radio Frequency Identification*) tehnologije. Postavljanje oznaka na razne predmete omogućuje korisniku da lako identificira i kontrolira označene stvari. RFID funkcionira bežično, pomoću elektromagnetskih polja, prilikom čega RFID čitač odašilje radiovalove koji napajaju pasivne RFID oznake. Nakon što je oznaka aktivirana tim impulsom, ona odašilje digitalne podatke koji dalje služe za identifikaciju (Wikipedia contributors, 2021b).

Mnogo je čimbenika koji utječu na razvoj interneta stvari; veliku ulogu ima bežična tehnologija, mikroelektromehanički sustavi (MEMS, engl. *Microelectromechanical systems*) te razvoj samoga interneta (Bhabad and Bagade, 2015). Mikroelektromehanički sustavi sastavljeni su od mikroskopskih uređaja, naročito od uređaja s pokretnim dijelovima. Njihova veličina može biti u nanometarskim rasponima, a često se nazivaju i mikrostrojevi ili mikrouređaji (Wikipedia contributors, 2021c). Što se tiče razvoja interneta najvažniji je internetski protokol IPv6 (engl. *Internet Protocol version 6*) jer ima jako veliku moć adresiranja i transferiranja velikoga broja podataka. IPv6 je relativno novi protokol koji je 2017. godine postao i internetski standard, zamijenivši svog prethodnika IPv4 (Wikipedia contributors, 2021d, p. 5). Zamjena se dogodila zbog rapidnoga rasta broja uređaja spojenih na Internet koji trebaju jedinstvenu IP adresu. Kapaciteti protokola IPv4 vrlo brzo postali su nedostatni pa je uvođenjem protokola IPv6 taj problem riješen.

2.3. Arhitektura

Arhitekturu interneta stvari možemo razdvojiti u tri sloja: percepcijski sloj, mrežni sloj i aplikacijski sloj. Prvi sloj podrazumijeva povezane fizičke uređaje, senzore, RFID oznake, aktuatore (dijelovi stroja koji ostvaruju fizičke pokrete). U ovom najnižem sloju odvija se prikupljanje informacija izvana od strane senzora i kontrola uređaja, dijelova stroja, aktuatora i slično. U nekim slučajevima može se izostaviti proces slanja podataka na oblak ili na poslužitelja ako je potrebna što brža reakcija uređaja na očitanje senzora. U tom slučaju se komunikacija odvija izravno između senzora i uređaja koji će obaviti neku radnju, a podatci se sukladno tome obrađuju na licu mjesta (engl. *edge processing*).

U mrežnom sloju odvija se slanje prikupljenih podataka na internet uz pomoć računala, bežičnoga ili žičnoga pristupa na mrežu i drugih načina. Prikupljeni podatci se prije slanja pretvaraju iz analognoga u digitalni oblik, agregiraju i formatiraju, a ovaj sustav prikupljanja i obrade podataka prije slanja se označava kraticom DAS (engl. *Data Aquisition Systems*). Ovaj sloj se u nekim slučajeima može i proširiti ako je potrebno pretprocesiranje podatka na licu mjesta (engl. *edge pre-processing*). Količine prikupljenih podataka u ovom stadiju mogu biti izrazito velike, pogotovo ako je riječ o industrijskoj proizvodnji sa stotinama senzora koji prikupljaju i simultano šalju gomile podataka. Postupkom pretprocesiranja bi se količina podataka filtrirala i komprimirala što olakšava kasnije slanje na server, oblak ili nešto slično. Ovdje do izražaja dolazi strojno učenje (engl. *machine learning*) i umjetna inteligencija. Zbog toga se u stvarnom vremenu može poslati povratna informacija stroju izravno te instantno poboljšati proces bez da se čekaju instrukcije iz nekoga udaljenoga centra ili oblaka. Veliku ulogu u mrežnom sloju ima već spomenuti protokol IPv6 koji može podržati veliku skalabilnost interneta stvari. Također, jedan od najčešćih protokola koji se koriste je i MQTT protokol (engl. *Message Queuing Telemetry Transport*). Služi za komunikaciju između pametnih uređaja, pripada kategoriji laganih protokola, a najčešće koristi TCP/IP model za funkcioniranje.

Posljednji sloj je aplikacijski sloj pri čemu se odvija analiza dobivenih podataka, na temelju rezultata obrade i analize se donose odluke i kontroliraju uređaji, a podatci se skladište. Uz ovaj sloj vežu se inteligentni sustavi s podatkovnim centrima ili oblacima. Time se dobiveni podatci s različitih lokacija mogu integrirati u neku širu sliku prilikom donošenja potrebnih akcija ili poslovnih odluka. Dakle, u ovom sloju se događa analitika podataka, menadžment i arhiviranje. Sustav može biti i u takozvanom oblaku, stoga se oblak može zamisliti kao mozak, sve uređaje i senzore kao udove, a mrežne protokole kao kralježnicu ili okosnicu tijela interneta stvari. Sustavi bazirani na oblaku dizajnirani su da skladište, procesiraju i analiziraju ogromne količine podataka te da koriste strojno učenje čime pružaju mnogo bolji uvid u stanje stvari. Nešto što se nikada ne bi moglo ostvariti spomenutim procesiranjem na licu mjesta. Računarstvo u oblacima (engl. *cloud computing*) sve više se koristi u industrijskom internetu stvari zbog

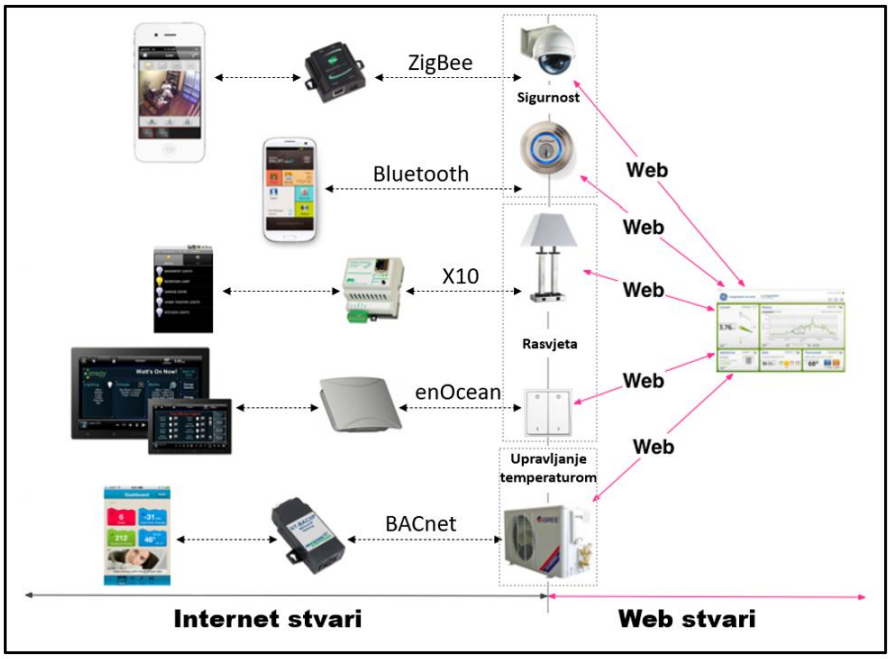
povećanja produktivnosti, smanjenja zastoja i vremena čekanja, a pogotovo zbog uštede energije.

3. Web stvari

Pojmovi internet stvari i web stvari mogu zvučati poput sinonima, ali postoji bitna razlika među njima. Pojam internet stvari je obrađen u prethodnom poglavlju iz kojeg se može zaključiti da IoT nastoji povezati veliki broj različitih uređaja na različite načine. Zbog velikoga broja proizvođača i proizvoda ponekad je potrebno koristiti i isto toliko različitih platformi i protokola. Primjerice, moguće je imati aplikaciju povezanu s nadzornom kamerom putem Zigbee protokola, posebnu aplikaciju za upravljanje pametnim žaruljama putem WiFi-ja, nekakvu drugu platformu za upravljanje potrošnjom struje itd. Ako korisnik, primjerice, želi pametni dom, najbolja opcija bi bila kupiti sve uređaje od istoga proizvođača. Ta preopterećenost tržišta i manjak interoperabilnosti (Androcec and Vrcek, 2016) zahtijevaju rješenje. Vrlo je kompleksno, gotovo nemoguće, napraviti jedinstveni komunikacijski protokol za efikasnu komunikaciju svih pametnih stvari. Vremenom su se razvijali takvi protokoli, čak i novi jezici, međutim vrlo brzo je postalo jasno da takav način zahtijeva previše vremena i resursa za standardizaciju. Pod tim okolnostima stvorio se koncept weba stvari, odnosno, korištenje već postojećega, afirmiranoga sustava kao što je web. Zato se web stvari definira kao mrežni standard koji omogućuje komunikaciju između pametnih stvari i web aplikacija.

Web stvari je premostio problem interoperabilnosti i potrebu da svaki uređaj ima drugačiji protokol i vlastitu aplikaciju, kao što imaju uređaji na lijevoj strani slike 1. Oslanja se isključivo na protokole i alate aplikacijske razine, a gotovo svaki protokol ili standard interneta stvari može biti povezan na web zahvaljujući "mostovima" (engl. *gateways*). Obzirom da je WoT fokusiran na aplikacijski sloj koji pruža visoku razinu apstraktnosti moguće je povezati podatke i servise s mnogo uređaja koji zapravo imaju drugačije transportne protokole. Prema tome, WoT pristupom je lakše programirati uređaje, podatci i servisi se mogu brže povezati, jednostavnije je razviti prototipe, rješenja i, u konačnici, održavati velike sustave (Guinard and Trifa, 2016). S druge strane, IoT pristup ima fokus na niže razine, optimiziraniji je za fiksne, ugrađene uređaje i kontroliranje potrošnje baterije, memorijskih i mrežnih performansi i sl.

Slika 1. *Razlika između IoT i WoT*



Izvor: preuzeto i prilagođeno s (Guinard and Trifa, 2016)

Uređaji niske razine se pomoću koncepta WoT-a apstrahiraju na višu razinu. Kao što je web globalna platforma za distribuciju aplikacija putem interneta, tako je isti princip iskorišten za pretvaranje interneta stvari u web stvari. U webu stvari su kompleksnosti i razlike među protokolima konačno zanemarene, a pažnja je usredotočena na aplikacijsku logiku. Mnogi benefiti ostvaruju se spomenutim pristupom. Moguće je koristiti moderne web standarde na uređajima, integrirati ih na web zajedno sa servisima na isti način na koji bi se inače razvijala web stranica. Sukladno tome, sve aplikacije komunicirat će s uređajima kao kad bi komunicirale s web servisom koji koristi web API (engl. *Application Programming Interface*). Time se pruža mogućnost za razvoj novih vrsta interaktivnih aplikacija. Koristeći standarde poput HTML-a, CSS-a i JavaScript-a se na web stranicu lako mogu prikazati prethodno prikupljeni i obrađeni podatci.

Web standardi (HTTP, HTML, CSS, URL, JavaScript...) prisutni su već dugo vremena, otvoreni su i besplatni te garantiraju stabilnost. Omogućuju brzo i pouzdano distribuiranje podataka diljem mreže. Web sigurno neće najednom prestati raditi ili zahtijevati ažuriranje, dok s gomilom IoT protokola dolazi taj rizik. Značajna prednost WoT-a je velika neovisnost među pametnim stvarima koje se mogu individualno mijenjati i razvijati. Stari uređaji mogu komunicirati s novim uređajima bez potrebe za ažuriranjem, isto kako se danas još uvijek mogu posjetiti stare web stranice bez ikakvih problema.

3.1. Arhitektura weba stvari

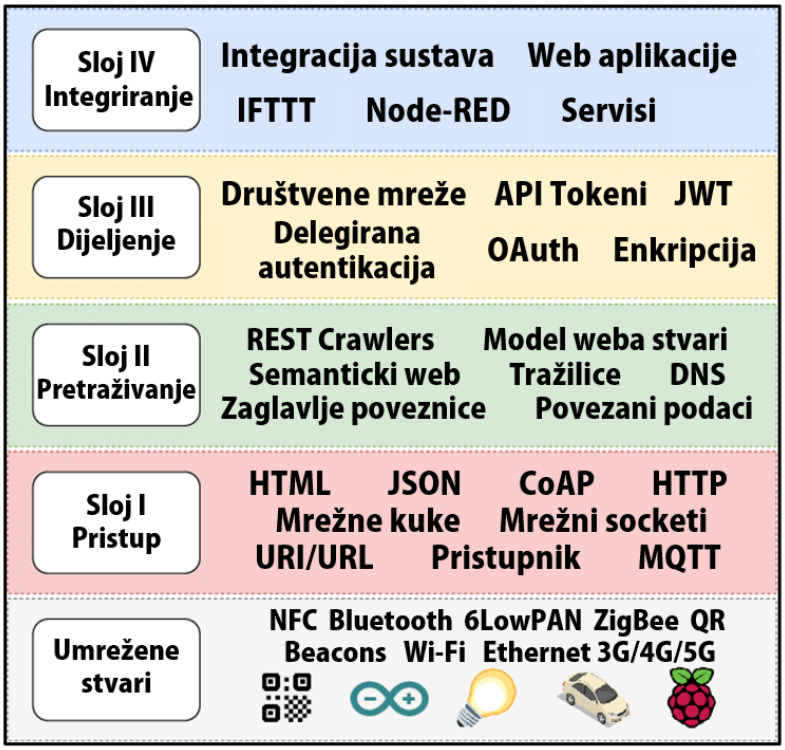
Glavni elementi arhitekture weba stvari su korisnici, stvari i posrednici. Stvar je naziv za fizički ili virtualni entitet standardiziranih metapodataka koji ga opisuju (W3C contributors, 2020). Stvari moraju biti povezane na mrežni sustav kako bi mogle interaktirati putem mrežnoga sučelja (engl. *interface*). Primjerice, na uređaju se može nalaziti HTTP server dok u pozadini senzori i ostali elementi mogu normalno komunicirati s fizičkim uređajem. Posrednici se uvode ako lokalnu mrežu nije moguće dohvatiti preko interneta. Zahvaljujući posredničkom

WoT sučelju moguće je ostvariti komunikaciju između korisnika i stvari. Principi weba stvari mogu se upotrijebiti u svim razinama i načinima povezivanja elemenata kao što je veza između više stvari, veza između stvari i pristupnika (engl. *gateway*), veza između stvari i oblaka te veza između pristupnika i oblaka.

Svaka web stvar ima četiri arhitekturalna apsekta: ponašanje, interakcijske mogućnosti, sigurnosne postavke i povezivanje protokola (W3C contributors, 2020). Interakcijske mogućnosti pružaju model prema kojem se korisnik može povezati sa stvarima, a povezivanje protokola daje dodatne informacije za ispravno ostvarivanje komunikacije. Sigurnosnim aspektima kontrolira se pristup interakcijskim mogućnostima i samim podatcima.

Arhitektura weba stvari je slična internetu stvari, a može se podijeliti u pet razina. Na najnižoj razini nalaze se fizički ili virtualni uređaji, uključujući Arduino, Raspberry Pi, pripadajuće senzorske sustave i ostalo. Druga razina je pristupna razina preko koje se ostvaruje komunikacija s uređajima. To su protokoli poput HTTP (engl. *Hypertext Transfer Protocol*), CoAP (engl. *Constrained Application Protocol*), MQTT (engl. *Message Queuing Telemetry Transport*) i drugi. Korištenjem ostalih web tehnlogija poput JSON (engl. *JavaScript Object Notation*), URL (engl. *Uniform Resource Locator*) te URI (engl. *Uniform Resource Identifier*) ostvaruje se bolja integracija cijeloga sustava. Treća razina WoT arhitekture služi za pretraživanje i pronalazak umreženih objekata, a sastoji se od raznih algoritama i tehnologija poput semantičkoga weba, pretraživača (engl. *search engines*), DNS-a (engl. *Domain Name System*) itd. Predzadnji sloj zadužen je za dijeljenje resursa, a primjeri takvih tehnologija su RESTful API mehanizmi i sigurnosni alati poput enkripcije, OAuth standard za omogućavanje pristupa i API tokeni. U najvišem sloju su sustavi za pojednostavljen razvoj aplikacija, a najpoznatiji su IFTTT (engl. *If This Then That*), Node-RED i drugi servisi web aplikacija (Barros et al., 2019).

Slika 2. *Arhitektura weba stvari*



Izvor: preuzeto i prilagođeno s (Barros et al., 2019)

3.2. Korištenje weba stvari

Za primjenu weba stvari u stvarnom životu podrazumijeva se prethodno napisano poglavlje o primjenama interneta stvari. Jedina je razlika što web stvari omogućava udaljeno kontroliranje svim uređajima, pristup uređajima putem web preglednika ili aplikacije. Stoga će u ovom poglavlju biti opisani načini na koji se uređaji mogu povezati međusobno, s kontrolerima, poslužiteljima i slično.

Najjednostavniji slučaj korištenja lokalni je uređaj kojim se upravlja daljinskim upravljačem (npr. klima uređaj). U ovom slučaju, daljinski upravljač bi se mogao implementirati kao gumb u aplikaciji ili web stranici, a komunikacija se odvija na razini lokalne mreže. Komunikacija između dvaju uređaja najčešće se postiže server-klijent vezom pri čemu jedan uređaj može slati poruku drugom i time izazvati neku akciju. Primjerice, klima uređaj povezan je sa senzorom temperature. Nakon što temperatura padne ispod određene granice, senzor to detektira, šalje poruku klima uređaju koji se zatim upali i radi dok se temperatura ne spusti.

Slučaj korištenja sličan prvom je slučaj udaljenoga pristupa. Tu korisnik može upravljati uređajem daljinskim upravljačem dok je unutar lokalne mreže, ali i izvan nje, samo nekim drugim protokolom. Kontroler se može nalaziti na pametnom mobitelu te može prelaziti s mobilne mreže na kućnu i obrnuto.

Između interneta i kućne mreže često se postavlja posrednik, takozvani *gateway*, primjerice, u pametnim domovima. *Gateway* upravlja uređajima unutar kućne mreže i može primati zapovijedi daljinskoga upravljača (npr. pametnoga mobitela kao u prethodnom slučaju). Pomoću *gatewaya* može se dobiti i virtualna reprezentacija uređaja kao i dobrobiti koje inače pruža vatrozid (engl. *firewall*). U ovom slučaju, *gateway* je samo posrednik između lokalne mreže i interneta. U nekim slučajevima *gateway* ima sposobnost obavljanja određenih funkcija pa se naziva i *edge gateway*. Najčešće se upotrebljava u industrijskim rješenjima gdje može obaviti pretprocesiranje, filtriranje i agregiranje velikih količina podataka.

Zahvaljujući oblaku ili poslužitelju moguće je napraviti digitalne blizance stvarnih fizičkih uređaja te komunicirati s njima. To pruža prednosti kod uređaja koji nisu konstantno online, omogućuje provođenje simulacija i testiranje, a tek onda komunikaciju sa stvarnim uređajem.

3.3. Sigurnost i privatnost

Sigurnost i privatnost važni su elementi koje se mora uzeti u obzir u svim implementacijama weba stvari. Cilj je osigurati sustav čijim podatcima mogu pristupiti samo autorizirani korisnici. Bitno je reći kako web stvari ne mogu pretvoriti nesiguran sustav u siguran, apsolutna sigurnost i privatnost ne može se garantirati, ali se arhitekturno mogu podržati sigurnosni mehanizmi. I u ovom pogledu je web stvari naprednij od interneta stvari zbog korištenja stabilnih i pouzdanih web protokola, ali još uvijek nijedan sustav nije neprobojan.

Uređaji mogu sadržavati osjetljive podatke zapisane u svojim TD-ovima u obliku metapodataka. Zato se nastoji postići strogo razdvajanje javnih i privatnih metapodataka. TD bi trebao posjedovati samo javne metapodatke kojima korisnici mogu pristupiti samo ako su autorizirani. Uređaji mogu posjedovati i informacije o identitetu osobe ili korisnika koji se mogu zlouporabiti. Takvi podatci trebaju biti minimizirani, a ako je moguće i sasvim izostavljeni. Generalno gledajući, TD-om bi se uvijek trebalo rukovati kao da posjeduju osjetljive identifikacijske podatke. Trebali bi se spremati i prenositi na što sigurniji način, pristup bi trebali imati samo autorizirani korisnici, trebali bi se brisati nakon određenoga perioda. Također, WoT načini povezivanja moraju ispravno podržavati sigurnosne mehanizme koji se već nalaze na IoT platformama ili uređajima. Ove mjere mogu znatno smanjiti sigurnosne rizike.

U WoT okruženju potrebno je izolirati skripte koje se na uređaju izvode, a sadržavaju osjetljive osobne podatke. Ako se takve skripte odvoje smanjuje se mogućnost curenja podataka. Fizički uređaj također može biti ugrožen ako skripta postane neispravna ili kompromitirana, pogotovo ako sučelju uređaja nedostaju sigurnosne provjere. U takvom slučaju potrebno je minimizirati broj sučelja koja su izložena opasnosti. Kao rješenje toga problema koriste se dodatni hardverski mehanizmi koji mogu odbiti izvršavanje određenih naredbi koje bi mogle naštetiti uređaju.

Neki uređaji podržavaju ažuriranje sebe, svojih skripti i ostalih podataka uključujući sigurnosne vjerodajnice. To može biti slaba točka i prilika za napade. Kako bi se rizik od napada smanjio, potrebno je poštivati sigurnosne preporuke prilikom ažuriranja uređaja. Osim toga, veliki rizik je i spremanje vjerodajnica na uređaj. Kada bi napadač došao do njih mogao bi pristupiti povjerljivim podatcima i izvesti hakerske napade poput DoS napada (engl. *Denial of Service*). WoT mora osigurati sigurno spremanje vjerodajnica, njihovu integriranost i povjerljivost.

4. Slučajevi korištenja

4.1. Korišteni alati i tehnologije

Za izradu praktičnoga dijela ovoga rada korišteni su sljedeći alati i tehnologije: Firebase, Fritzing, Android Studio, ArduinoIDE i Flutter. ArduinoIDE koristi se za programiranje mikrokontrolera, Flutter za izradu mobilne aplikacije, a Firebase je "most" između njih. Fritzing je alat za izradu vizualno pristupačnih dijagrama kako bi se jednostavnije prikazao svaki zaseban projekt u ovom radu. Android Studio koristi se prilikom razvoja mobilne aplikacije jer omogućava testiranje na virtualnim Arduino uređajima.

Firebase je platforma za razvoj mobilnih i web aplikacija, osnovana 2012. godine, a od 2014. godine je u vlasništvu tvrtke Google (Wikipedia contributors, 2021e). Firebase je baza podataka u stvarnom vremenu, ima ulogu API-ja koji sinkronizira podatke u Flutter aplikaciji s podatcima na Firebase-ovom oblaku. Izvrsno je rješenje softverskim programerima za razvoj aplikacija koje rade u stvarnom vremenu. Postupak pripreme Firebase-a za korištenje započinje kreiranjem besplatnoga korisničkoga računa na službenoj Firebase stranici firebase.google.com. Nakon kreiranja računa, pokraj korisničkoga avatara će se pojaviti gumb "Go to console". Klikom na gumb pristupa se konzoli sa svim korisnikovim projektima. Prilikom prvoga pokretanja potrebno je napraviti novi projekt opcijom "Add project".

Fritzing je program korišten za izradu shema, tj. strujnih krugova, no na vizualno pristupačniji način. Primarna svrha softvera je upravo dokumentiranje Arduino prototipova te izrada PCB dizajna za proizvodnju. Program ne koriste isključivo inženjeri, nego dizajneri, umjetnici, istraživači i hobisti prilikom razvoja projekata. Sam program je otvorenoga tipa, napisan u C++ jeziku koji se može pronaći na GitHub-u.

Kao pomoć pri programiranju mobilne aplikacije koristio se Android Studio, razvojno okruženje (engl. *Integrated Development Environment*) specijalizirano za razvoj Android mobilnih aplikacija. Razvijen od strane tvrtki Google i JetBrains, Android Studio kompatibilan je s macOS, Windows i Linux operacijskim sustavima. Iako se mobilna aplikacija u potpunosti mogla razviti u Android Studiju, odlučeno je koristiti samo jednu njegovu funkcionalnost, a ostatak je razvijen u Dartu/Flutteru. Ta funkcionalnost je AVD Manager (engl. *Android Virtual Device*). Zahvaljujući AVD-u moguće je kreirati, duplicirati, prilagođavati i pokretati Android virtualne uređaje. To znači da za razvoj i testiranje Android mobilne aplikacije nije neophodan fizički uređaj, samo virtualne instance koje pruža AVD.

Glavni element ovoga rada su mikrokontroleri za čije programiranje je korišteno razvojno okruženje *ArduinoIDE*. Programski jezik koji se pritom koristi je C++ uz nekoliko metoda i

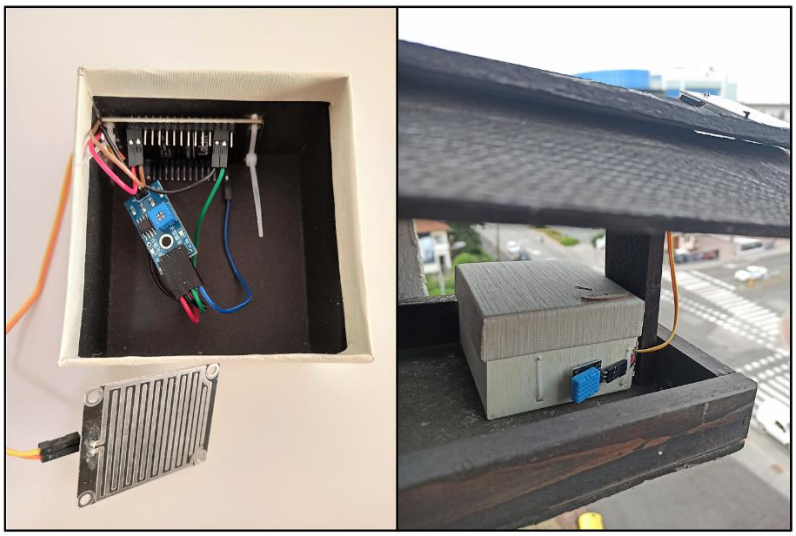
funkcija specifičnih za ArduinoIDE i ovakav način programiranja. Za svrhe praktničnoga dijela ovoga rada izrađeno je pet različitih ".ino" skripti, po jedna za svaku NodeMCU razvojnu ploču. Skripte imaju neke zajedničke dijelove koda, poput spajanja na WiFi i povezivanja s Firebase-om, dok ostatak koda ovisi o namjeni projekta.

Visual Studio Code pripada kategoriji uređivača izvornoga koda (engl. *source code editor*), za razliku od Visual Studija ili Android Studija koji su razvojna okruženja. Postavljanje Flutter okvira na Visual Studio Code jednostavno je, potrebno je samo odabrati opciju "Extensions", pronaći Flutter i instalirati. Instaliranjem Flutter pristupa se svim funkcionalnostima koje nudi, a pripadnim SDK-om (engl. *Software Development Kit*) moguće je, u konačnici, kompilirati napisani kod. Uz Flutter će se automatski instalirati i Dart ekstenzija. Dart je programski jezik čiji glavni fokus je na razvoju *front-end* korisničkih sučelja i aplikacija. Neovisan je o Flutteru te se njime mogu razvijati i web aplikacije. U ovom projektu naglasak nije na samom Dartu nego na tome kako ga Flutter koristi za razvoj mobilnih aplikacija. Flutter je okvir za Dart jezik, odnosno kolekcija alata, funkcija i *widgeta* koji se implementiraju pomoću Dart jezika. Cilj Flutter frameworka je učiniti razvoj mobilnih višeplatformskih aplikacija što jednostavnijim. Ključna riječ je višeplatformski (engl. *cross-platform*), što označava mogućnost generiranja aplikacija za različite platforme na temelju samo kodne baze. Kod napisan u Dartu/Flutteru se pomoću Flutter SDK kompilira u izvorni kod (engl. *native code*) posebno za Android i za iOS. Rezultat toga su optimizirane aplikacije visokih performansi.

4.2. Izrađeni IoT prototipovi

U svrhu kreiranja aplikacije za web stvari i integracije različitih IoT uređaja napravljeno je pet IoT prototipova. Prvi je vremenska stanica (slika 3). Vremenska stanica sastavljena je od NodeMCU V3 razvojne ploče na koju su povezani senzori za temperaturu i vlagu (DHT11) i senzor za kišu, DHT11 modul ima tri pina koja su povezana spomenutim *jumper* žicama i to tako da se odgovarajuće pinove spoji s "GND" pinom za uzemljenje, "3v" pinom za napajanje DHT11 modula i "D3" pinom za slanje digitalnoga signala. Senzor za kišu također je spojen na "GND" i "3V" pinove iz istih razloga te na "A0" analogni pin za slanje informacija o jačini padalina. Stanica s navedenim senzorima postavi se na otvorenom i redovno šalje podatke o temperaturi, vlažnosti zraka i informacije o tome pada li kiša i koliko jako.

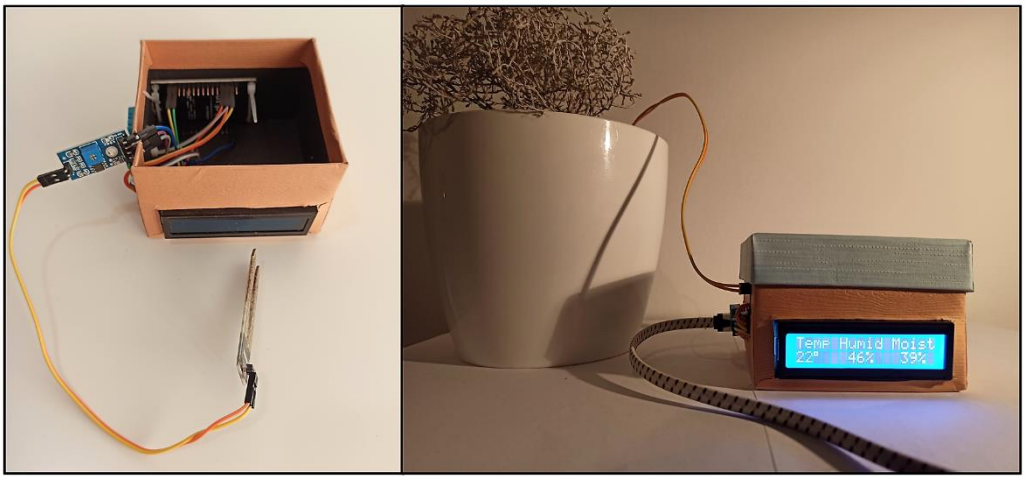
Slika 3. *Vremenska stanica*



Izvor: autorski rad

Stanica za biljke (slika 4.) također je koncipirana tako da očitava i obrađuje podatke iz okoline. Budući da je prikladnija za sobnu upotrebu, ima spojen i LCD ekran za ispis podataka: temperature, vlage i vlažnosti tla. Za temperaturu i vlagu se i ovdje koristi DHT11 senzor, a za vlažnost tla se koristi *Soil Moisture Sensor* koji se zabode u zemlju.

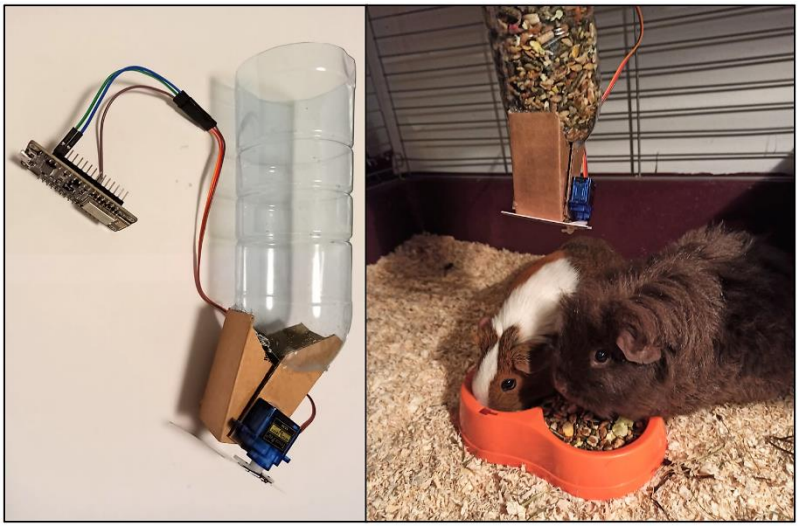
Slika 4. *Stanica za biljke*



Izvor: autorski rad

Projekt hranilice za ljubimce (slika 5.) sastoji se samo od jedne dodatne komponente, a to je mali servo motor. On će se upaliti kada skripta iz baze dohvati određen podatak, točnije kada dohvati logičku vrijednost postavljenu na "*true*". Ovo je prvi primjer u kojem je potrebno dohvaćati podatke s Firebasea, a ne samo spremati ih. Na rotacijski dio servo motora pričvršćena je pločica koja ima ulogu vrata. Kada se motor aktivira, vrata se otvaraju, a kroz vrata cure sjemenke za zamorce.

Slika 5. *Hranilica za ljubimce*

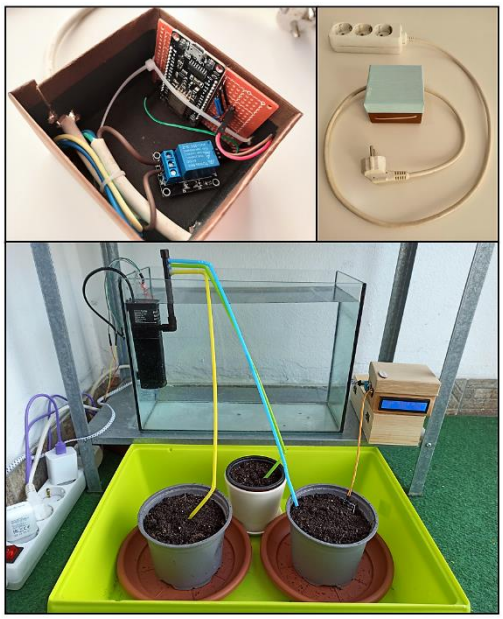


Izvor: autorski rad

Relej sklopka (engl. *Relay Switch*), funkcionira na principu običnoga prekidača za svjetlo. Njegova namjena je otvaranje ili zatvaranje strujnoga kruga što se postiže slanjem signala s mikrokontrolera. Pomoću NodeMCU razvojne ploče to je moguće činiti i na daljinu. U

stvarnom projektu prekidač je ugrađen u produžni kabel pa se tako bilo koji kućanski aparat i uređaj može paliti i gasiti preko mobilne aplikacije. Osim toga, dodan je i senzor za očitavanje razine vode postavljen u akvarij s vodom. Naime, u produžni kabel uštekan je obični akvarijski filter s cijevi na vrhu iz koje vraća vodu u akvarij. On je prenamijenjen tako da voda ne curi natrag u akvarij nego u lončanicu sa zemljom. Time se dobio jednostavan sustav za navodnjavanje koji u isto vrijeme može pokretati i zaustavljati navodnjavanje zemlje te pratiti trenutačnu razinu vode u akvariju.

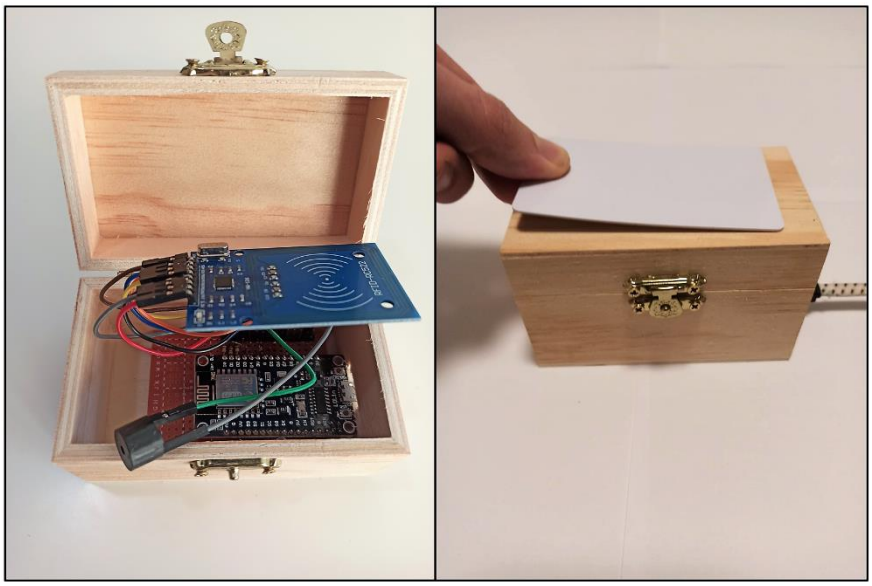
Slika 6. *Relej sklopka*



Izvor: autorski rad

Projekt RFID čitača sastoji se od NodeMCU razvojne ploče, RFID čitača i maloga zvučnika (*Mini Piezo Buzzer*). Za aktiviranje čitača koristi se RFID kartica i RFID privjesak za ključeve. Oba predmeta imaju jedinstveni "otisak" koji se pomoću skripte dohvati i ispiše na serijski monitor. Budući da je otisak predmeta stalan i jedinstven, svi otisci se kopiraju sa serijskoga monitora i spreme za kasnije korištenje. Zvučnik je dodan zbog praktičnosti, ali i zbog efikasnijega testiranja.

Slika 7. *RFID čitač*



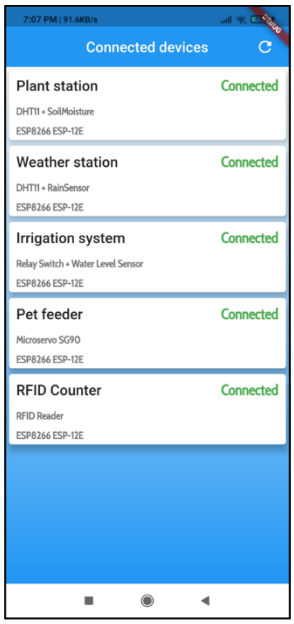
Izvor: autorski rad

4.3. Mobilna aplikacija za web stvari

Programiranje mobilnih aplikacija u Flutteru ne podrazumijeva korištenje vizualnoga editora s *drag-and-drop* elementima. Korisničko sučelje se gradi isključivo kodiranjem i konstruiranjem tzv. stabla widgeta. Widget je osnovni građevni element u Flutteru. Cijela aplikacija zapravo jedan kompleksni widget (najčešće "MaterialApp") koji za argumente prima druge widgete i tako dalje. Flutter dolazi s gomilom postojećih widgeta, ali moguće je kreirati i vlastite ako je to potrebno. Svaki zaslon kojega korisnik vidi će u suštini biti klasa koja proširuje jedan od dva *state widgeta*: "StatefulWidget" ili "StatelessWidget". "StatelessWidget" uobičajen je widget koji služi za jednostavan prikaz podataka i statičnoga korisničkoga sučelja. Međutim, ako se podatci moraju osvježavati i mijenjati u stvarnom vremenu, potrebno je koristiti "StatefulWidget" koji ima mogućnost ažuriranja korisničkoga sučelja.

Klik na svaku karticu vodi do *dashboard* prikaza odabranoga projekta. Programski kod zadužen za *dashboard* je u suštini isti za svaki projekt, dok konkretan prikaz ovisi o podatcima iz baze. Na dnu ekrana prikazane su informacije o projektu, odnosno svi podatci koji se nalaze u "BasicInfo" segmentu u bazi podataka. Projekt vremenske stanice sa slike 3 se sastoji od dva tipa podataka: brojčanoga i tekstualnoga. Brojčani podatci odnose se na temperaturu i vlažnost zraka te su prikazani u lijevom stupcu. Tekstualni podatak vezan za intenzitet kiše prikazuje se u desnom stupcu.

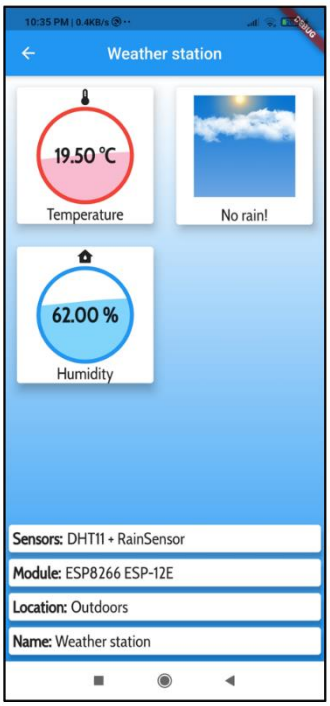
Slika 8. *Početni zaslon mobilne aplikacije za web stvari*



Izvor: autorski rad

Glavni dio *dashboarda* isprogramiran je u svrhu ostvarivanja potpune modularnosti. To znači da nije bitno koliko će brojčanih, numeričkih ili nekih drugih podataka biti spremljeno u bazu ili kako će ti podatci biti imenovani. Korisnik može imati sve tekstualne elemente napisane na nekom drugom jeziku, može u svom projektu imati desetke povezanih senzora ili nijedan. Svi će podatci biti prikazani u vlastitim karticama na *dashboardu* i raspoređeni u stupce ovisno o tipu. Stoga je jedina obaveza korisnika pratiti nekoliko jednostavnih pravila prilikom kreiranja skripte za mikrokontroler. Ako korisnik ispravno unese sve podatke u skriptu i uspješno ih pošalje na Firebase, oni će se instantno prikazati u aplikaciji.

Slika 9. *Prikaz vremenske stanice u aplikaciji*



Izvor: autorski rad

5. Zaključak

U ovom radu opisani su najbitniji koncepti i karakteristike koji se vežu uz pojmove interneta i weba stvari. Navedene su najvažnije razlike između pojmova i razlozi zašto je web stvari bolja verzija interneta stvari. Bitno je napomenuti kako je broj uređaja povezanih na internet prestigao ukupni broj ljudske populacije, što dovoljno govori o kakvoj rastućoj industriji je riječ. Internet stvari prati nove tehnološke inovacije i dosege te se sukladno tome neprestano razvija.

Praktični dio rada prati razvoj jednostavnoga pametnoga doma koji se sastoji od pet neovisnih projekata. Unatoč tome što su neovisni, zahvaljujući Firebase platformi i web protokolima, centralizirani su u jednoj mobilnoj aplikaciji. Aplikacija je razvijena kombinacijom Dart jezika i Flutter okvira, a glavne značajke su modularnost i fleksibilnost. Ovaj rad može se zamisliti kao kostur ili temelj većega sustava na koji je moguće dodati mnoge druge funkcionalnosti. Uz relativno malo programskoga koda omogućen je prilagodljiv prikaz različitih podataka iz baze. Gomila funkcionalnosti koje Flutter pruža, brza krivulja učenja i interesantna tema učinile su izradu ovoga rada vrlo poučnim i korisnim iskustvom.

Literatura

1. Androcec, D., Vrcek, N., 2016. Thing as a Service Interoperability: Review and Framework Proposal. IEEE, pp. 309–316. https://doi.org/10.1109/FiCloud.2016.51
2. Barros, V., Junior, S., Bruschi, S., Monaco, F., Estrella, J., 2019. An IoT Multi-Protocol Strategy for the Interoperability of Distinct Communication Protocols applied to Web of Things. pp. 81–88. https://doi.org/10.1145/3323503.3349546
3. Bhabad, M.A., Bagade, S.T., 2015. Internet of things: architecture, security issues and countermeasures. Int. J. Comput. Appl. 125.
4. Correcting the IoT History, 2017.

5. Guinard, D.D., Trifa, V.M., 2016. Building the Web of Things: With examples in Node.js and Raspberry Pi. Manning Publications.
6. W3C contributors, 2020. Web of Things (WoT) Architecture.
7. Wikipedia contributors, 2021a. Internet of things — Wikipedia, The Free Encyclopedia.
8. Wikipedia contributors, 2021b. Radio-frequency identification — Wikipedia, The Free Encyclopedia.
9. Wikipedia contributors, 2021c. Microelectromechanical systems — Wikipedia, The Free Encyclopedia.
10. Wikipedia contributors, 2021d. IPv6 — Wikipedia, The Free Encyclopedia.
11. Wikipedia contributors, 2021e. Firebase — Wikipedia, The Free Encyclopedia.