Razvoj IoT aplikacije za pametni dom

Jaša Vid Meh Peer

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška cesta 25, Ljubljana
jasa.meh.peer@gmail.com

Abstract. Internet-of-Things (IoT) devices have been on the market for some time now, but they are not yet accessible to everyone, mainly due to price concerns. We wanted to create a proof of concept device and application that is more affordable. We designed and built basic appliances such as smart switch for turning light over the internet and smart blinds control that can control windows shades over the internet. Our simple IoT devices consist of simple electronics and run on microcontrollers that are based on the ESP8266 integrated circuit that can be connected over wireless network. The appliances connect to the network and they communicate using Message Queuing Telemetry Transport (MQTT) protocol over the public internet to the cloud service called Adafruit IO, which runs control application for connected appliances, save their data or inter-connect these appliances to other services like If That Then That (IFTTT) and WebHooks. We also tested our appliances and application to verify their security and reliability making it useful for development of IoT devices in the future.

1 Uvod

Želja po razvoju in uporabi aplikacij Interneta stvari (*ang. Internet-of-Things, IoT*) je zagotovo človeška lenoba in težnja k izboljšanju življenjskega sloga. Prvi koncept o povezanemu sistemu internetnih naprav je bil predstavljen leta 1982 na primeru prodajnega aparata s povezljivostjo na internet, ki je svojemu upravniku lahko sporočal ali je katerega izdelka že zmanjkalo in če so se novi izdelki že ohladili [6]. Od takrat naprej se je uporaba in implementacija različnih IoT aplikacij in naprav razširila na vsa področja, saj z njimi olajšamo tako tehnološke postopke kot vsakodnevna opravila.

Posebno področje IoT aplikacij so pametni domovi, vdelani pametni sistemi kot so KNX, Wireless in Loxone so za gospodinjstva v današnjem časupostali nepogrešljivi. Ker imajo vsi profesionalni sistemi sorazmerno visoko ceno za uporabo v manjšem ali starejšem stanovanju, smo želeli ta problem rešiti z uporabo vdelanih naprav, ki komunicirajo z internetom in nam omogočajo upravljanje izhodnih enot, ki so pravilno prilagojene za različne porabnike in opravila.

Za demonstracijske namene smo pripravili napravo in vezje za avtomatizacijo luči in zaves, ki jih lahko izdelamo sorazmerno enostavno, s pomočjo preprostih enostavno dobavljivih komponent. V drugem poglavju bomo predstavili uporabljene tehnologije, v tretjem strojno in programsko opremo ter komunikacijski protokol. V četrtem poglavju bomo predstavili rezultate testiranja naše aplikacije. Na koncu bomo zaključili z rezultati in razpravo o možnihnadgradnjah sistema.

2 Uporabljene tehnologije

2.1 Telemetrijski transportni protokol za čakalne vrste

Message Queuing Telemetry Transport (MQTT) je odprtokodni International Organization for Standardization (ISO) standard, ki je nizko-zahteven glede strojne opreme, in deluje po vzorcu publish/subscribe.

Protokol se zelo pogosto implementira v nizko nivojnih sistemih in v mobilnih aplikacijah, saj je uporabniku ključnega pomena zanesljivost in nizka poraba energije [7].

2.2 Platforma Adafruit IO

Oblačna storitev Adafruit IO se uporablja za povezovanje naprav in aplikacij preko interneta ter za shranjevanje in pošiljanje podatkov. Ponuja tudi mnogo drugih storitev, kot so prikazovanje stanj povezanih naprav v realnem času, upravljanje naprav in aplikacij preko interneta ter pošiljanje podatkov, kot je vremenska napoved, napravam in aplikacijam. Adafruit IO je pogosto uporabljena storitev, razlog za to je predvsem, ker je večji del storitve brezplačen.

2.3 Mikrokrmilnik ESP8266

ESP8266 je mikrokrmilnik starejše generacije, ki je cenovno ugoden, omogoča pa Wi-Fi povezljivost, in ima implementiran protokolni sklad Transmission-Control-Protocol/Internet-Protocol (TCP/IP). Gre za 32 bitni procesor, ki deluje z logičnimi napetostnimi nivoji na 3,3 V in s frekvenco programske ure 80 MHz ali 160 MHz.

Med izdelovalci je postal mikrokrmilnik zelo priljubljen v letu 2014, ko je na trg stopila prva razvojna platforma, ki ga je uporabljala. Od tistega leta dalje je na osnovi ESP8266 nastalo veliko različnih razvojnih platform, ki so delovanje in uporabo mikrokrmilnika izboljšale in poenostavile. Med najbolj priljubljene spadajo ESP 1.0, NodeMCu, Wemos, Adafruit Huzzah in še mnogo drugih.

3 Razvoj IoT sistema in naprav

S predstavljenim sistemom želimo pokazati, kako iz osnovnih elementov sestaviti napredno IoT aplikacijo. Pripravili smo dve prototipni IoT napravi, ki sta povezani na oblačno platformo preko lokalne brezžične povezave

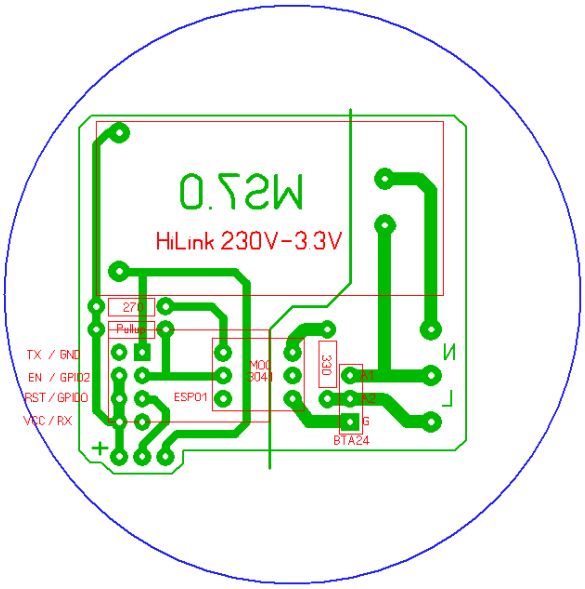
s protokolom MQTT. Glavna komponenta IoT naprav je mikrokrmilnik ESP8266, ki krmili ostale naprave in je povezan s spletom. Za spletno kontrolo smo uporabili brezplačno platformo Adafruit IO, na katero se povežejo vse naprave v našem sistemu.

Uporabo komponent in povezavo vezij je narekovalo več zahtev. Najosnovnejši zahtevi sta bili sposobnost brezžične povezave in avtomatizacija vodenih komponent, kot so stikala in motorji. Kasneje pa smo naprave pripravili v velikostnih okvirih potrebnih za montažo in prilagodili delovanje glede na specifično aplikacijo naprave.

3.1 IoT pametno stikalo

Zahteve pri izdelavi pametnega stikala so bile vezane predvsem na velikost samega vezja, saj je bilo potrebno celotno napravo vgraditi v inštalacijsko dozo kot nadomestilo klasičnemu stikalu. Ohranili smo tudi klasično delovanje stikala torej na pritisk z uporabo kapacitivnega ali navadnega preklopnega stikala.

3.1.1 Strojna oprema



Slika 1. Vezje za pametno stikalo na katerem lahko vidimo vse povezave med mikrokrmilnikom ESP01 in ostalimi uporabljenimi komponentami. Moder krog je za primerjavo vezja z velikostjo inštalacijske doze.

Pametno stikalo sestavljajo 3 W usmernik HiLink, ki iz 230 V izmenične napetosti pretvori na 3,3 V enosmerne napetost, ki napaja mikrokrmilnik. Izbrali smo mikrokrmilnik ESP 1.0, saj je najmanjša razvojna platforma, ki uporablja ESP8266 in ustreza vsem zahtevam.

Ima 2 Večnamenska Vhodno/Izhodna priključka (*ang. General-Purpose-Input/Output (GPIO)*) od katerih je priključek 2 uporabljen za vhodne enote, ki so lahko kapacitivno stikalo na dotik, navadno preklopno stikalo ali gumb, v našem primeru smo izbrali preklopno stikalo. Izhodne enote so povezane na drugi GPIO priključek, priključek 0, na katerega je bil sprva povezan rele, vendar smo zaradi zmanjšanja vezja naprave prešli na vezavo optotriak-triak [1], ki zavzame dosti manj prostora.

Izhodna enota preko glavnega triaka preklaplja izmenično breme v našem primeru je to žarnica, bi pa na takšen način lahko krmilili katerokoli napravo.
Vezje je veliko 41x37mm in ga je, glede na Sliko 1, mogoče vstaviti v inštalacijsko dozo.

3.1.2 Programsko delovanje

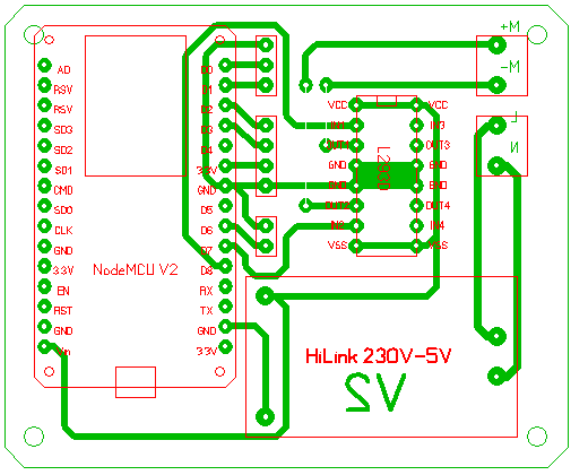
Program, ki se izvaja na mikrokrmilniku, vključuje štiri knjižnice in dve dodatni funkciji [2, 3]. Knjižnice omogočajo vzpostavitev WiFi povezave ter povezovanje preko MQTT protokola do storitve Adafruit IO v oblaku [4, 5, 10].

Ob priklopu na napajanje, mikrokrmilnik začne z izvajanjem programa. Sprva se izvede funkcija za inicializacijo sistema. Vzpostavi se povezava na brezžično omrežje in povezava na Adafruit MQTT posrednik. Določijo se tudi namembnosti priključkov mikrokrmilnika. Nato mikrokrmilnik preide v stanje preverjanja lastnega stanja in odzivanja na zahteve iz strežnika. V tem stanju kličemo dve funkciji za spreminjanje stanja na izhodu. Eno za ročen vklop razsvetljave, ki spremlja stanje na vhodnem priključku, ter drugo, ki spremlja podatke, ki prihajajo iz strežnika, in na podlagi teh ukazov spremeni stanje izhoda.

3.2 IoT pametne zavese

Zahteve pri tej napravi so bile drugačne kot pri IoT stikalu. Velikost naprave ni bila odločilnega pomena, zato smo lahko uporabili različico mikrokrmilnika ESP8266 z več priključki. Za krmiljenje motorja smo potrebovali vsaj 2, več priključkov pa smo potrebovali za generiranje prikaza na štiri znakovnem sedem segmentnem zaslonu.

3.2.1 Strojna oprema



Slika 2. Vezje za pametne zavese na katerem lahko vidimo vse povezave med mikrokrmilnikom NodeMCU 12E in ostalimi uporabljenimi komponentami.

Ker so bile zahteve za to napravo bolj obsežne, smo potrebovali več vhodno-izhodnih enot in posledično tudi

mikrokrmilnik z več GPIO priključki, kar je bil največji razlog, da smo se odločili za NodeMCU 12E.

Vhodni napravi sta ročica za krmiljenje in gumb za izbiro menijev. Ročica potrebuje dva GPIO vhoda, gumb pa je povezan na en GPIO vhod. Na vseh vhodnih enotah smo uporabili že integrirane *pull-up* upore in zato posledično nismo potrebovali dodatnih zunanjih.

Izhodne enote so modul s štirimi sedem segmentnimi zasloni (vsak za prikaz ene števke) za prikazovanje menija, za kar potrebujemo 4 GPIO priključke ter dva GPIO priključka, ki sta uporabljena za krmiljenje vhodov integriranega vezja L293D [9], ki smo ga uporabili za krmiljenje motorja, ki je v našem primeru 6V DC z reduktorjem.

3.2.2 Programsko delovanje

Program pametnih zaves je po veliki večini enak programu za pametno stikalo, sestavljajo ga iste knjižnice in glavne funkcije [2, 3].

Programa se razlikujeta v dveh knjižnicah in treh funkcijah. Program pametnih zaves vsebuje dve dodatni knjižnici in sicer eno za štiri znakovni sedem segmentni zaslon ter drugo, ki omogoča povezavo preko Omrežno časovnega protokola (*ang. Network Time Protocol (NTP)*) na server za podatke o času. Funkcije, *Manual*, *Automation* in *wifiControl*, ki spremljajo vhode in podatke iz interneta ter na podlagi le-teh spreminjajo stanje izhoda preko funkcije Output. Ter funkcija timeSet v kateri je meni, za nastavljanje podatkov za avtomatsko delovanje.

3.3 Komunikacijski protokol

MQTT protokol uporablja model publish/subscribe, ki je alternativa tradicionalnemu Hipertekstovnemu prenosnemu protokolu (*ang. HyperText Transfer Protocol (HTTP)*) modelu klient/server [7].

Najpomembnejša točka v MQTT sistemu je MQTT posrednik [8], katerega naloga je prenašanje podatkov ob spremembah od pošiljateljev do prejemnikov, v našem primeru je posrednik in prejemnik/strežnik ista stvar, Adafruit IO. Sporočila morajo imeti ustrezen naslov, da lahko posrednik pošlje podatek pravemu prejemniku.

3.4 Oddaljeno upravljanje

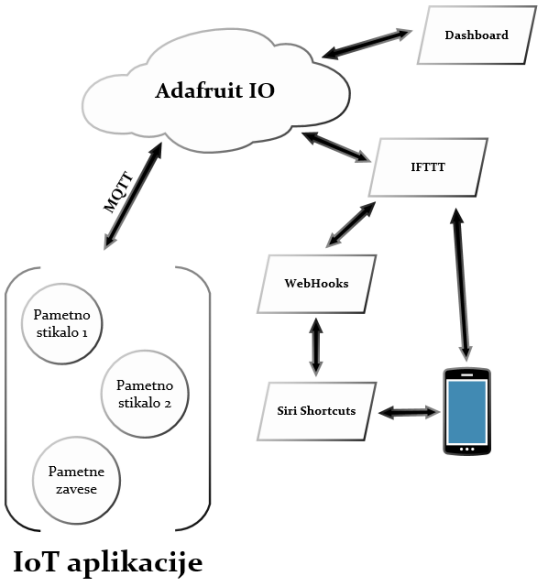
Za oddaljeno upravljanje IoT aplikacije uporabljamo storitev Adafruit IO. Na javno dostopni strežnik se povežemo preko MQTT protokola s pomočjo knjižnice *AdafruitIO*. Ker za uporabo storitve potrebujemo uporabniški račun in edinstven identifikator, smo za potrebe naše aplikacije to ustvarili na spletni platformi.

V platformi Adafruit IO spremljamo stanje izhodov in jih na kontrolni plošči tudi smiselno spreminjamo. Prav tako lahko s pomočjo platforme, naše naprave povežemo z drugimi spletnimi storitvami, kot so If That Then That (IFTTT).

IFTTT nam omogoča povezavo z vrsto ostalih oblačnih storitev, kot so npr.: Amazon Alexa, Instagram, lokacijski podatki, Facebook, Gmail in WebHooks. Mi smo implementirali uporabo IFTTT aplikacije na mobilnem telefonu tako, da smo povezali Adafruit IO in IFTTT ter s pomočjo *button widget*, navideznega gumba na namizju telefona, in WebHooks povezav spreminjali stanja na izhodnih enotah naših aplikacij.

WebHooks so zelo pogosto uporabljeni v aplikacijah, ko želimo, da se določena procedura izvede, ko odpremo ustrezno spletno povezavo. S pomočjo WebHooks in aplikacije za iPhone Siri Shortcuts smo povezali Siri in naše naprave, kar nam je omogočilo upravljanje IoT storitve z glasom.

4 Testiranje sistema



Slika 3. Shema sistema, na kateri lahko vidimo vsak posamezen gradnik sistema in kje v verigi delovanja stoji.

4.1 Splošno delovanje sistema

Delovanje sistema je glede na uporabljene komponente in odprtokodno okolje, skoraj brezhibno. Čas potovanja informacije od uporabnika do IoT naprave je v razponu 2-5 sekund. Do razlik pride zaradi moči brezžičnega signala in načina pošiljanja podatkov, od telefona do Adafruit IO platforme in do IFTTT ali Siri s pomočjo WebHooks (Slika 3).

Normalno uporabo sistema lahko razdelimo na ročno, na dotik in prostoročno. Kot smo že prej omenili lahko stikalo in zavese še vedno upravljamo s fizičnimi stikali torej ročno. S pomočjo IFTTT lahko s pritiskom na zaslon telefona spremenimo stanje izhodne enote, upravljanje na dotik ali pa sistem upravljamo s pomočjo glasovnih komand preko Siri, prostoročno upravljanje, v primeru, ko imamo roke polne ali pa smo zaposleni z čim drugim. Vsi ti različni načini upravljanja sistema nam dajo veliko izbire in s tem nam oziroma uporabniku izboljšajo izkušnjo.

4.2 Varnost sistema

Ko obravnavamo IoT sisteme, je zelo pomembno vprašanje varnosti. MQTT protokol ima zelo visok nivo varnosti, če uporabljamo šifriranje. Kljub temu da uporablja TCP protokol, je možna popolna ali samo delna enkripcija podatkov [7].

V naši storitvi se pojavi težava z varnostjo, ko povezavi med napravami in Adafruit IO dodamo druge

storitve preko IFTTT. Večina storitev, ki jih lahko povežemo z IFTTT deluje brez motenj, vendar v določenih primerih, kot so WebHooks, lahko pride do nezaželenega dogajanja. Ko naredimo spletno povezavo kot sprožilec za določeno dejanje naše IoT aplikacije, vsebuje spletna povezava ime dogodka in naš varnostni ključ. Če nekdo pozna ta dva podatka, nam lahko preko splošnega razporeda podatkov v spletni povezavi sproži naš dogodek. Seveda se lahko zgodi, da določena oseba vzpostavlja svojo spletno povezavo in ima isto ime dogodka ter se pri varnostnem ključu zmoti in vtipka našega nenamerno in nam tako sproži nezaželen vklop porabnika.

4.3 Zanesljivost sistema

Težave pri uporabi se še pojavijo pri zanesljivosti našega sistema, ki je predvsem odvisna od delovanja storitve v oblaku, načina delovanja MQTT protokola in močjo signala brezžičnega omrežja, na katerega so naše IoT aplikacije povezane. Če ena od naštetih ne deluje, pomeni, da podatki ne bodo prišli od IoT naprave na strežnik oz. iz telefona do IoT naprave. Odvisno od napake je ali se bodo podatki izgubili ali pa shranili v čakalno vrsto MQTT posrednika in se kasneje, ko bo povezava ponovno vzpostavljena, poslali k čakajoči napravi. V takem primeru se zgodi nepričakovana reakcija sistema na že pretečeni dogodek.

5 Zaključek

S tem člankom smo pokazali, da je mogoče s pomočjo preprostih komponent in odprtokodnih orodij izdelati preprosto, a vseeno široko uporabno IoT storitev. Med izdelavo aplikacije se je pokazalo, da je načrtovanje in izdelava enoslojnih vezij preprost postopek. Z nekaj različnimi prototipi smo izdelali končni produkt, ki deluje dobro glede na različne pogoje v okolju in z različnimi komponentami, ki jih upravljamo.

Našemu sistemu je težavo predstavljalo predvsem delovanje programa. Da je bilo možno izhode spreminjati na tradicionalen način s stikalom in preko interneta, smo morali preizkusiti kar nekaj različnih variacij kode. Seveda smo na koncu tudi programsko izvajanje optimizirali za najboljše možno delovanje. To sicer ni odpravilo vseh težav, saj zaradi komunikacijskega protokola in slabe brezžične povezave še kdaj pride do motenj v delovanju.

Strojno delovanje sistema in IoT aplikacije je že sedaj brez napak, vendar je rezkanje zgolj prototipni postopek, tako bi lahko v prihodnje naročili profesionalno izdelana večslojna vezja pri kakšnemu priznanemu proizvajalcu ali uporabili tehniko jedkanja.

Izboljšali bomo tudi kontroliranje zaves, saj enosmerni krtačni motorji niso zelo zanesljivi z vidika regulacije. Tako bomo za dviganje in spuščanje zaves rajši uporabili koračni motor in primerno krmilno enoto. S tem povečamo zanesljivost in natančnost ter tako dodamo več stopenj dviganja.

Nasprotno kot pri strojni opremi je programsko delovanje zagotovo potrebno še izboljšati, da bo delovanje aplikacije potekalo popolnoma brez motenj. Izboljšanja delovanja se lahko lotimo tako, da vztrajamo pri načinu, ki smo ga do sedaj uporabljali in popravimo napake, katere ima. Ali poskusimo drugo metodo pristopa in uporabimo mikroračunalnik (npr. Raspberry Pi) za vodenje storitev v oblaku in tako lokalno namestimo MQTT posrednik ter razvijemo lastno storitev podobno Adafruit IO za kontrolo in shranjevanje podatkov. Tako bi našo storitev popolnoma ločili od svetovnega spleta ter povečali varnost ter zanesljivost.

Zahvala

Za pomoč in nasvete se zahvaljujem as. Matevžu Hriberniku.

Literatura

[1] Horowitz Hill, The Art of Electronics. Cambridge University Press, 1980.

[2] Iztok Fajfar, Programiranje mikrokrmilnikov in programiranje 2, zapiski predavanj, UL FE, 2020.

[3] Tomaž Dobravec, abC, Založba FE in FRI, Ljubljana, 2010.

[4] Internet of Things Class, https://www.instructables.com/class/Internet-of-Things-Class/ (30.8.2020).

[5] Esp8266 control, https://www.instructables.com/id/Control-ESP8266-Over-the-Internet-from-Anywhere/ (30.8. 2020).

[6] Internet of things, https://en.wikipedia.org/wiki/Internet_of_things (30.8.2020)

[7] MQTT – vodilni komunikacijski protokol za Industrijo 4.0, https://www.tipteh.si/mqtt/ (20.8.2020).

[8] Matevž Hribernik, Razvoj varnega komunikacijskega posrednika in učinkovite arhitekture za povezovanje interneta stvari in spletnih aplikacij, magistrsko delo, UL FE, 2019.

[9] L293 dokumentacija, https://www.ti.com/lit/ds/symlink/l293.pdf (2.9.2020)

[10] ESP8266 as a WiFi Switch, https://www.instructables.com/id/ESP8266-Wifi-Switch/ (30.8.2020).