

Prototip sustava dojave slanjem poruka na mobilni uređaj

Glavurtić, Borna

Undergraduate thesis / Završni rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **The Polytechnic of Rijeka / Veleučilište u Rijeci**

Permanent link / Trajna poveznica: <https://um.nsk.hr/um:nbn:hr:125:298728>

Rights / Prava: [In copyright](#) / [Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-02-22**



Repository / Repozitorij:

[Polytechnic of Rijeka Digital Repository - DR PolyRi](#)



VELEUČILIŠTE U RIJECI

Borna Glavurčić

Prototip sustava dojave slanjem poruka na mobilni uređaj

(završni rad)

Rijeka, 2021.

VELEUČILIŠTE U RIJECI

Poslovni odjel

Preddiplomski stručni studij Informatika

Prototip sustava dojave slanjem poruka na mobilni uređaj

(završni rad)

MENTOR

izv. prof. dr. sc. Alen Jakupović, prof. v. š.

STUDENT

Borna Glavurčić

MBS: 2422000017/18

Rijeka, lipanj 2021.

VEUČILIŠTE U RIJECI

Poslovni odjel

Rijeka, 15.3.2021.

ZADATAK
za završni rad

Pristupnik Borna Glavurtić, MBS: 2422000017/18.

Studentu preddiplomskog stručnog studija Informatika izdaje se zadatak za završni rad –
tema završnog rada pod nazivom:

PROTOTIP SUSTAVA DOJAVE SLANJEM PORUKA NA
MOBILNI UREĐAJ

Sadržaj zadatka:

Opisati osnovne karakteristike mikroročunala Raspberry Pi te proceduru njegova postavljanja. Prikazati uslugu besplatnog slanja poruka preko Vonage API. Objasniti primjenu Vonage API kroz alat cURL. Implementirati shell skriptu u sklopu koje se poziva cURL naredba primjenom alata Bash. Izraditi automatizirano i periodičko pokretanje skripte primjenom alata Cron. Izraditi prototip sustava dojava upaljenog svjetla slanjem poruke na mobilni uređaj primjenom programskog jezika Python.

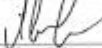
Preporuka:

Rad obraditi sukladno odredbama Pravilnika o završnom radu Veleučilišta u Rijeci.

Zadano: 15.3.2021.

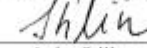
Predati do: 15.9.2021.

Mentor:



(izv.prof.dr.sc. Alen Jakupović, prof.v.š.)

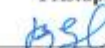
Pročelnik odjela:



(dr.sc. Anita Stilin, v. pred.)

Zadatak primio dana: 15.3.2021.

Pristupnik:



(Borna Glavurtić)

Dostavlja se:

- mentoru
- pristupniku

IZJAVA

Izjavljujem da sam završio rad pod naslovom PROTOTIP SISTEMA DODATNE
PODRUČKA NA MOBILNI UREDAJ izradio samostalno pod
nadzorom i uz stručnu pomoć mentora ALENA JAKUPOVIĆA.

BORNA GLAVURTIĆ
Ime i prezime

B. Gl
(potpis studenta)

SAŽETAK

Ovaj rad predstavlja i opisuje uspješan proces izrade projekta za završni rad, napominje prepreke koje su se pojavile putem i objašnjava kako su riješene.

Cilj projekta je bio ostvariti automatizirani sustav koji očitava trenutno stanje željenih parametara (broj osoba u nekoj prostoriji, jesu li u nekoj prostoriji uključena svjetla, koliko je ljudi u nekom periodu ušlo/izašlo iz prostorije, temperatura ili vlaga u nekoj prostoriji, itd.), sprema očitane podatke i zaposleniku šalje poruku na njihov mobilni uređaj sa informacijama koje su im ključne za kvalitetno odrađivanje njihovog posla, ili tek kao podsjetnik da isključe grijanje u prostorijama u kojima je ostalo uključeno i u istoj poruci im kaže u kojim točno prostorijama je grijanje još uključeno i treba ga ugasiti.

KLJUČNE RIJEČI: Automatizacija, Raspberry Pi, API, cURL, Python

SADRŽAJ

1.	Uvod	1
2.	Raspberry Pi	2
2.1	Konfiguracija pločice.....	3
2.2	Raspbian OS	4
2.3	Prvo pokretanje Raspberry Pi-a	5
2.4	Instalacija softvera i aplikacija	5
2.5	Primjena Raspberry Pi-a u projektu.....	7
3.	Vonage API i cURL	9
3.1	API (Application Programming Interface)	12
3.2	Vonage WhatsApp Sandbox API	12
3.3	cURL, što je i gdje se koristi?.....	14
3.4	Prilagođena cURL skripta za projekt.....	18
4.	Bash (Bourne Again Shell).....	19
5.	Cron	21
5.1	Prvi crontable	22
6.	Python.....	26
6.1	Import i deklaracija.....	27
6.2	Očitavanje senzora.....	28
6.3	Slanje poruka	28
6.4	Povratna informacija.....	29
7.	Zaprimljene poruke.....	30
8.	Zaključak	32
	POPIS LITERATURE.....	33
	POPIS KRATICA	34
	POPIS SLIKA	35

1. Uvod

Svrha ovog projekta je podignuti produktivnost zaposlenika, olakšati im rad i istovremeno postići nižu razinu propusta i grešaka na radnom mjestu. To je postignuto kombiniranjem nekoliko različitih resursa, tehnologija, uređaja i programskih jezika.

U drugom poglavlju rada, „Raspberry Pi“, ukratko se opisuje što je Raspberry Pi, u koje se svrhe koristi, te se поближе opisuje kako je konfiguriran za svrhe ovog projekta.

U trećem poglavlju, „Vonage API i cURL“, četvrtom poglavlju „Bash“, i petom poglavlju „Cron“, opisuje se što je svaki od navedenih pojmova, za što se koriste u današnjem svijetu i kako je svaki od njih ključan za uspjeh sustava za automatiziranu dojavu informacija zaposleniku.

U šestom poglavlju, „Python“, obrađuje se povezivanje svih prije navedenih ključnih elemenata sustava i kako su ujednačeni u jedan kratak, sažet i jednostavan proceduralan Python programski kod.

U sedmom poglavlju, „Zaprimljene poruke od sustava“, pokazuju se primjeri zaprimljenih poruka koje je Raspberry Pi poslao zaposlenicima.

U osmom poglavlju „Zaključak“ navedene su osnovne značajke ovog završnog rada te je ukratko opisano mišljenje autora.

2. Raspberry Pi

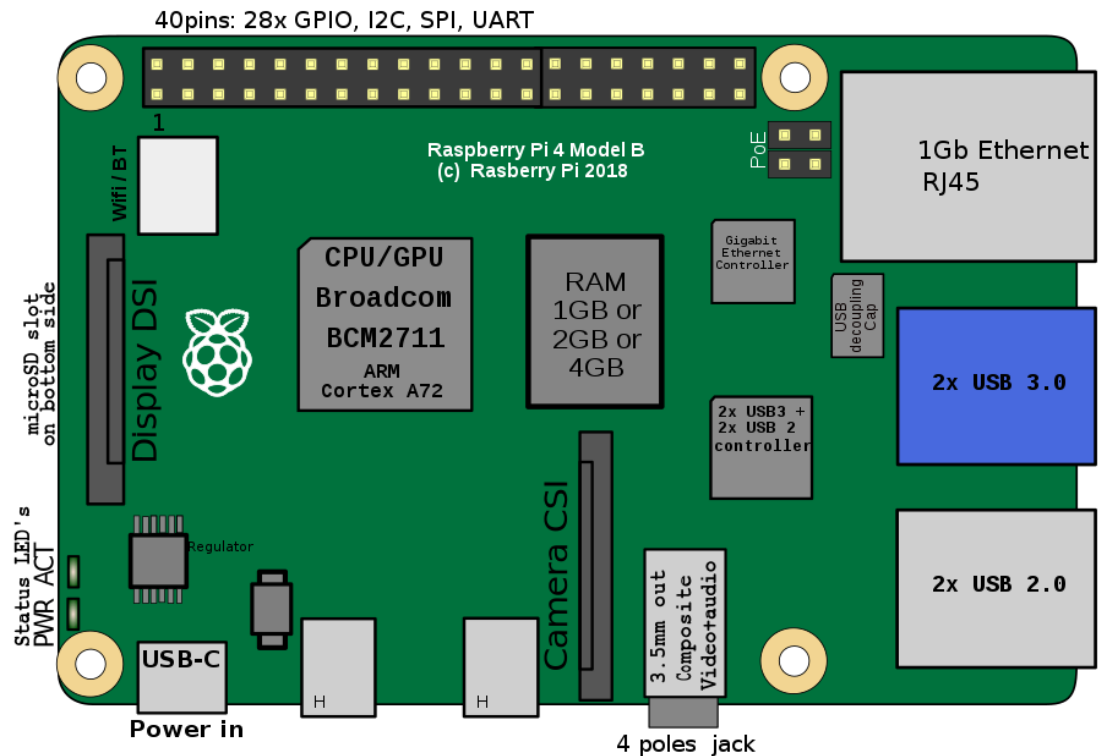
Raspberry Pi je serija malih računala na kojima je svaka komponenta prisutna u modernim računalima, integrirana na jednu pločicu. Idealni su za nezahtjevne poslove i iznimno prilagodljivi za skoro svaku potrebu. Najviše se koriste u robotici i svakakvim područjima gdje se može primijeniti senzore ili automatizaciju procesa. (Raspberry Pi, 2021.)

Prednosti Raspberry Pi-a su u njegovoj niskoj cijeni i dobrom rasponu I/O (Input/Output) uređaja s kojima je kompatibilan. To ga čini najboljim uređajem za one koji izrađuju razne male ili velike projekte u području robotike i elektronike. Do prosinca 2019. godine prodano je 30 milijuna primjeraka raznih modela Raspberry Pi računala. (Upton, 2019.)

Nedostatci Raspberry Pi-a su u ograničenoj snazi i brzini CPU-a (Central Processing Unit) i izloženosti pločice (pločica dolazi bez ikakvog kućišta).

Za izradu ovog projekta korišten je Raspberry Pi 4B, drugi najnoviji model. Sadrži 40 GPIO (General Purpose Input/Output) pinova, 2 USB 2.0 i 2 USB 3.0 konektora, 2 microHDMI konektora, 1 Ethernet konektor, 1 USB-C konektor za napajanje, 1 3.5mm audio jack i 1 microSD slot, ARM Cortex-A72 CPU (1.5GHz 64-bit quad core), 802.11ac Wi-Fi karticu i Bluetooth 5 čip.

Slika 1. Konektori i gdje se nalaze na Raspberry Pi 4B pločici



Izvor:

https://en.wikipedia.org/wiki/Raspberry_Pi#/media/File:RaspberryPi_Model_4B.svg

2.1 Konfiguracija pločice

Nakon utvrđivanja ciljeva i zahtjeva projekta, prvi korak je bio osposobiti Raspberry Pi za rad. To se radi instalacijom OS-a (operacijski sustav) na microSD karticu, koju se potom ubaci u njegov microSD slot. Kod odabira OS-a koji želimo instalirati imamo mnoštvo opcija (Linux, Raspbian, Windows IoT core, OSMC, RISC OS, itd.).

Najbolji od njih, za opću namjenu, je Raspbian OS.

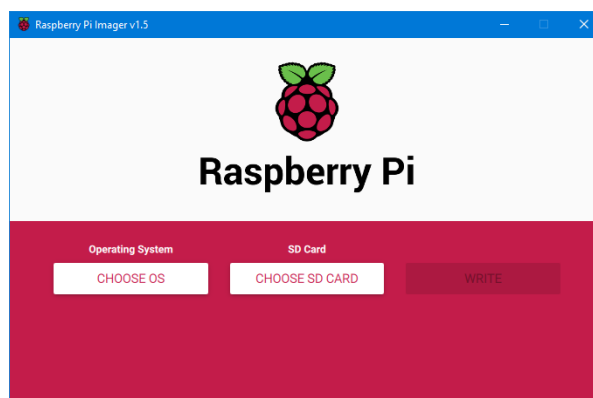
2.2 Raspbian OS

Raspbian OS je posebno izgrađen OS od strane Raspberry Pi inženjera. Baziran je na Linux distribuciji „Debian“ i idealan je i za početnike, a i iskusnije hobiste zahvaljujući velikom broju raznog softvera koji se dobije samom instalacijom OS-a.

Sama instalacija OS-a je također iznimno jednostavna i puno je truda uloženo kako bi se manje iskusnim korisnicima pružila prilika da se što prije mogu početi baviti onime što su namijenili.

Sve što korisnik treba učiniti je otvoriti službenu Raspberry Pi web-stranicu, navigirati do Operating System Images, kliknuti na hipervezu Raspberry Pi Imager koja pokreće download *Imager* aplikacije, nju se instalira na računalo i pokrene. Pokretanjem *Imager-a* otvara se vrlo jednostavan GUI (Graphical User Interface) u kojem korisnik odabire OS koji želi instalirati i na koju umetnutu SD karticu, nakon čega se polje „WRITE“ zabijeli poput druga dva, i korisnik potom klikne na njega.

Slika 2. Raspberry Pi Imager v1.5



Izvor: Izradio autor

Imager zatim pokreće proces pisanja OS-a na SD karticu, cijeli proces općenito traje oko 20 minuta. (Raspberry Pi, 2021.)

2.3 Prvo pokretanje Raspberry Pi-a

Prvim pokretanjem korisnika će dočekati „Welcome Wizard“ koji će provesti korisnika kroz konfiguraciju raznih postavki (korisnička lozinka, lokacija i vrijeme, odabir Internet mreže, itd.) i za kraj procesa nudi korisniku „Check for updates“ opciju, koju korisnik prihvaća ili odbija.

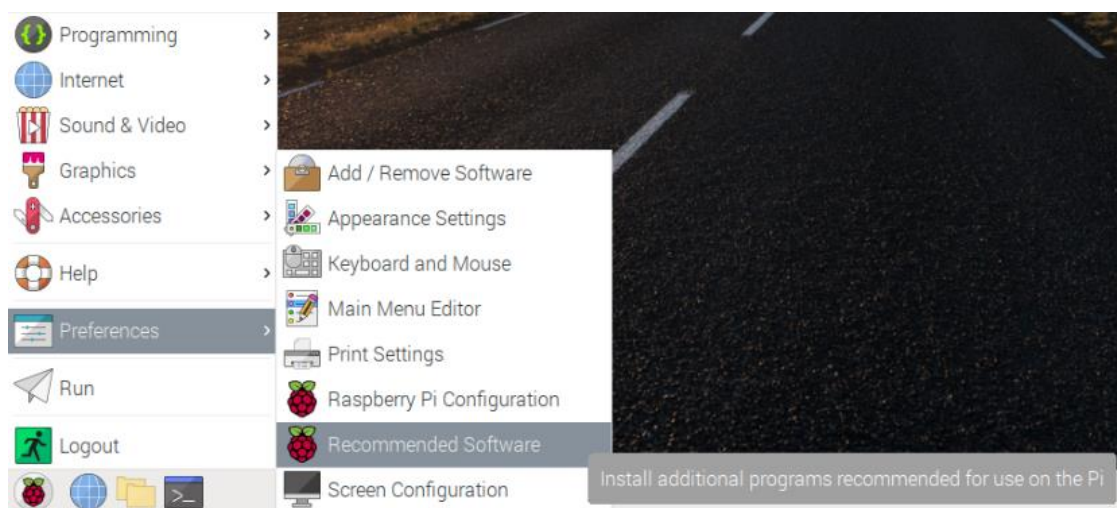
Za svaku primjenu Raspberry Pi-a za koju mu je potreban pristup internetu, preporuča se pokrenuti „Check for updates“ proces.

2.4 Instalacija softvera i aplikacija

Nakon što je Raspberry Pi konfiguriran za rad, korisnik treba instalirati softver i aplikacije koje će mu trebati za rad.

Klikom na Raspberry Pi logo u donjem lijevom kutu otvara se Start menu u kojem korisnik klikom miša odabire Preferences – Recommended Software.

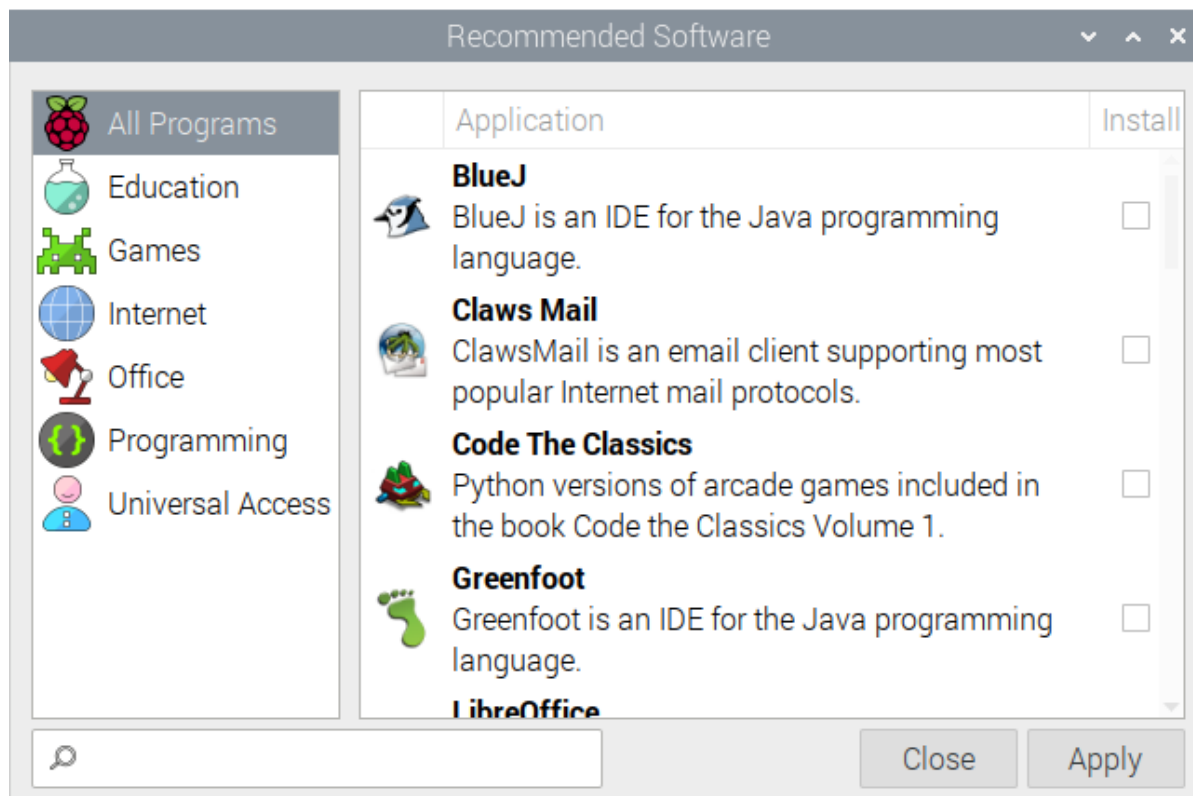
Slika 3. Instalacija softvera i aplikacija



Izvor: Izradio autor

Korisniku se otvara Recommended Software prozor sa popisom raznih aplikacija, podijeljenih u kategorije, od kojih korisnik odabire koje želi instalirati.

Slika 4. Recommended Software prozor



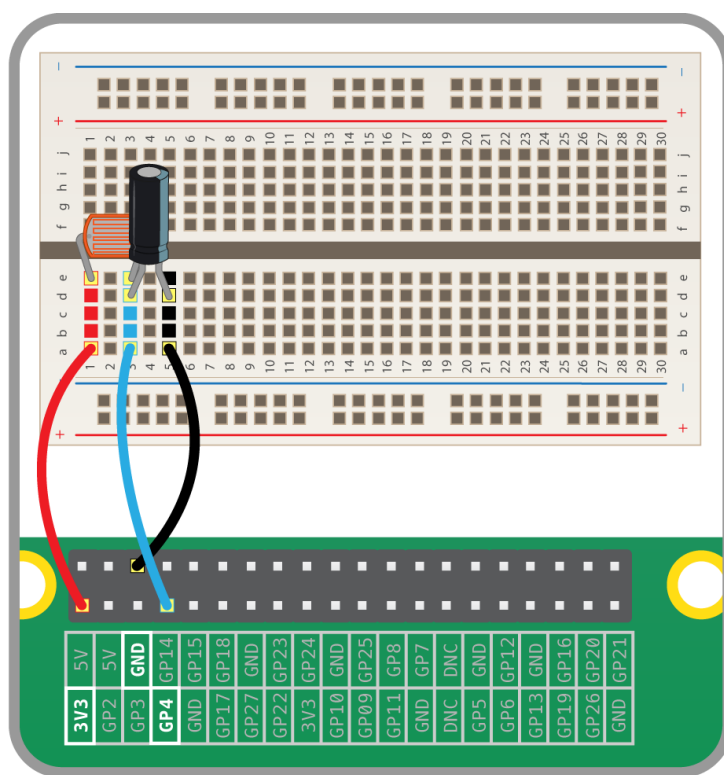
Izvor: Izradio autor

Ovisno o odabranom broju aplikacija, ovaj proces također može potrajati do 20 minuta.

2.5 Primjena Raspberry Pi-a u projektu

U svrhu izrade projekta, Raspberry Pi je konfiguriran na opće postavke, instaliran je Python IDE (Integrated Development Environment) i korištena su 3 LDR (Light Detecting Resistors) fotootpornika, 3 kondenzatora od 10 μF , 9 M-Ž (Muško-Žensko) žica i jedna eksperimentalna pločica na koju su spojeni. (Raspberry Pi, 2021.)

Slika 5. Shema napajanja eksperimentalne pločice sa Raspberry Pi GPIO-a



Izvor: <https://projects.raspberrypi.org/en/projects/laser-tripwire/2>

Na prethodnoj slici se vidi shema spajanja fotootpornika i kondenzatora na Raspberry Pi GPIO pinove. Sva 3 korištena seta LDR-a i kondenzatora su spojeni na isti način, paralelno jedan s drugim.

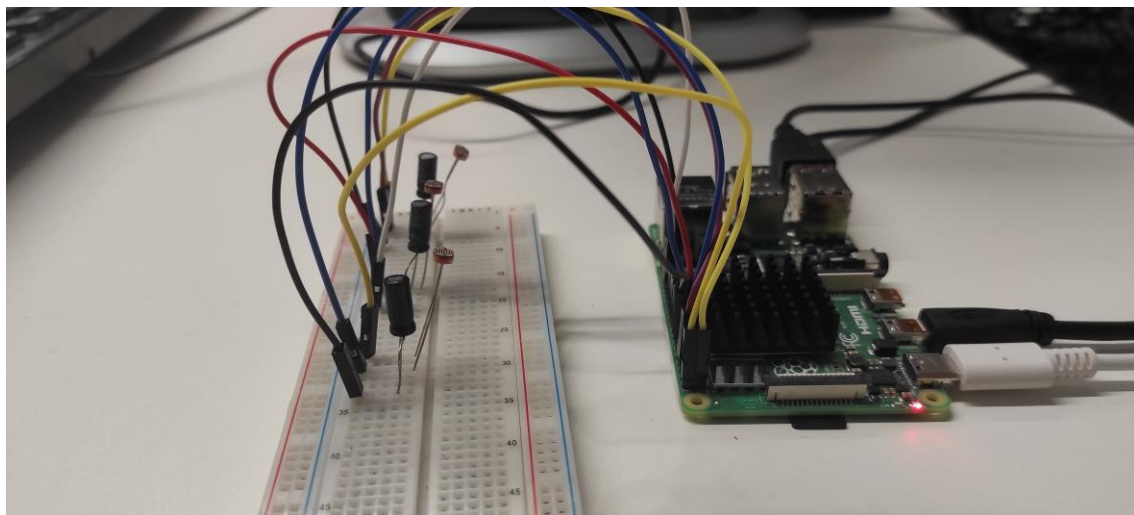
Za izradu projekta korišten je hipotetski scenarij u kojem imamo zgradu fakulteta sa krajem radnog vremena u 20:00 radnim danom, subotom i nedjeljom ne radi. Na fakultetu su zaposlena 2 portira.

U svrhu smanjena troškova poduzeće želi osigurati da nakon radnog dana u nekoj prostoriji nisu ostala upaljena svjetla. Portir svaki dan prije zatvaranja obilazi zgradu i provjerava svaku prostoriju kako bi ugasio svjetla gdje treba, ali to je monoton i dugotrajan posao i još gore, podložan ljudskoj grešci.

Ovaj je problem riješen tako da se u svaku prostoriju postavi senzor svjetla povezan na Raspberry Pi. U svrhu projekta, korištena su 3 senzora od kojih svaki predstavlja jednu prostoriju.

Raspberry Pi pokreće program koji očitava stanje svakog senzora i obavještava portira porukom na njegov WhatsApp sa 100% preciznom i pouzdanom informacijom u kojim prostorijama je svjetlo upaljeno.

Slika 6. Strujni krug sa 3 LDR-a i kondenzatora spojeni na GPIO



Izvor: Izradio autor

3. Vonage API i cURL

Vonage (Vonage Holdings Corp.) je javna korporacija koja pruža usluge poslovne cloud komunikacije sa sjedištem u New Jerseyu. Osnovana je 2001. godine kao poslužitelj telekomunikacijskih i VoIP (Voice over Internet Protocol) usluga.

Sa preko 2 000 zaposlenika, 1 000 000 registriranih korisnika i preko 100 000 poslovnih odnosa, Vonage je vodeća snaga u modernim telekomunikacijskim uslugama.

Kako većina računalnih OS-ova, uključujući Raspbian OS, nema podršku za slanje i primanje SMS poruka i ostalih usluga koje mobilni uređaji nude, kao niti utor za SIM karticu, Vonage-ov besplatan *Sandbox* virtualni mobilni uređaj nam nudi odgovor na problem slanja poruke sa Raspberry Pi-a na mobilni uređaj.

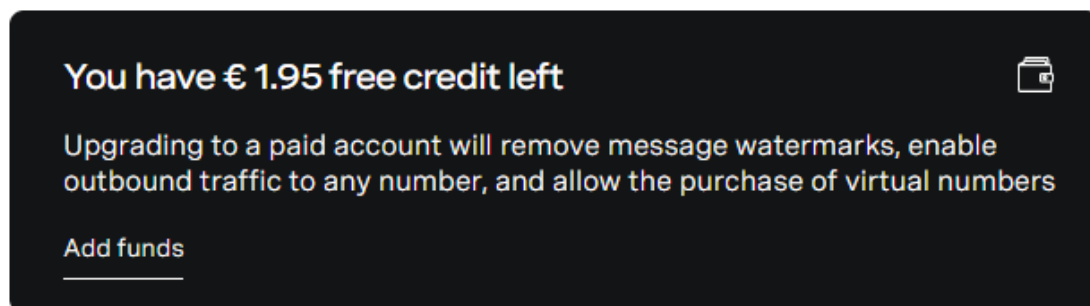
Kako bi se dobilo pristup Vonage-ovom *Sandbox* API-u (Application Programming Interface) treba napraviti korisnički profil na Vonage.com sa valjanom E-mail adresom i *whitelist-ati* broj mobilnog uređaja koji želimo koristiti, nakon čega nam Vonage odmah nudi razne priručnike koji su iznimno kvalitetni i jednostavni za pratiti. Jedan od njih pokazuje korisniku kako kontaktirati *Sandbox* API za slanje SMS poruke na *whitelist-ani* broj.

Mobilni broj se *whitelist-a* slanjem ključne riječi „Join frail easel“ na virtualni broj API-a (+1 415-738-6170).

Izvorno je projekt bio zamišljen tako da šalje SMS poruke, a ne WhatsApp poruke. Međutim to je predstavilo nezgodnu prepreku, cijena. SMS poruke se naplaćuju.

Slika 7. Vonage SMS API Dashboard

Vonage API Dashboard



Izvor: <https://dashboard.nexmo.com/>

Dok Vonage svim besplatno registriranim korisnicima na njihov korisnički račun postavi gratis 2€ (euro), svaki poslani SMS se plaća 5 centi u Hrvatskoj (cijena varira ovisno o državi u kojoj se nalazi broj na koji se šalje SMS). To znači da imamo ograničen broj (40) SMS poruka koje možemo poslati prije nego bi trebalo nadoplatiti račun.

Prednost SMS poruka je što svaka osoba sa mobilnim uređajem može primiti SMS, dok nije svatko korisnik WhatsApp-a. U drugu ruku, velika prednost WhatsApp-a jest što su poruke besplatne za slanje, i za svrhu ovog projekta, to je bio iznimno važan faktor zahvaljujući kojem je odluka prevagnula u korist WhatsApp-a.

Ako zaposlenik ne koristi WhatsApp, isti se program s relativnom lakoćom može primijeniti i za Viber ili Facebook Messenger.

Slika 8. WhatsApp, Viber i Facebook Messenger Sandbox set up

Set up your sandbox

The sandbox currently supports these channels.

The screenshot displays a user interface for setting up a sandbox. On the left, a green panel titled 'WhatsApp' contains a QR code and instructions: 'Scan the QR code, or [click this link](#), and hit send on the pre-filled message.' Below this is a button labeled 'Send invite email' with a dropdown arrow. Further down, it says 'Send a WhatsApp message to +14157386170 with the passphrase [Join frail easel](#)'. At the bottom of the panel, it states 'You have 2 user(s) whitelisted. [Refresh](#)'. To the right of the WhatsApp panel are two white boxes. The first is for 'Viber', with the Viber icon, the text 'Exchange inbound, outbound, and image messages.', and a purple link 'Add to sandbox'. The second is for 'Facebook Messenger', with the Messenger icon, the text 'Exchange inbound, outbound, and rich media messages.', and a purple link 'Add to sandbox'.

Izvor: <https://dashboard.nexmo.com/messages/sandbox>

3.1 API (Application Programming Interface)

API je akronim za Application Programming Interface, koji je softverski posrednik koji omogućuje da dvije aplikacije međusobno razgovaraju. Drugim riječima, API je prijenosnik koji korisnikov zahtjev dostavlja davatelju od kojeg ga traži, a zatim mu vraća odgovor natrag.

API se prema tipu otvorenosti dijeli na 3 tipa:

1. Otvoreni ili javni API
2. Partner API
3. Interni ili privatni API

Javni API-i su dostupni bilo kojem programeru bez obzira na lokaciju ili njihov radni odnos sa vlasnikom API-a, uz minimalne restrikcije.

Partner API-i dozvoljavaju pristup i uporabu samo onim programerima čiji su poslodavci u poslovnom odnosu.

Privatni API-i su skriveni od eksternih korisnika i programera i isključivo dostupni zaposlenicima vlasnika API-a.

3.2 Vonage WhatsApp Sandbox API

API koji se kontaktiralo za izradu ovog projekta je javni Vonage-ov *Sandbox* API za slanje i primanje WhatsApp poruka. Kao što se iz njegovog tipa (javni) vidi, on je javno dostupan i besplatan za korištenje. Vonage je također razvio i naprednije verzije API-a za SMS, MMS, WhatsApp, Viber i Facebook Messenger platforme, ali ti API-i nisu javno dostupni, već su partnerskog tipa i dostupni su za uporabu nakon stupanja u poslovni odnos.

Nakon što je odabran API koji će se koristiti za kontaktiranje korisnikovog WhatsApp-a, treba odgovoriti na sljedeće pitanje: „Kako?“.

Tu je od velike pomoći Vonage-ov *dashboard* koji je opremljen raznim primjerima kodova kako bi korisniku omogućio da što lakše i brže shvati što se koristi i kako.

To se ostvaruje izvođenjem cURL skripte.

Slika 9. Sample kod za kontaktiranje WhatsApp API-a

Send a message

WhatsApp Viber Facebook Messenger

Don't forget to replace `$TO_NUMBER` in the code example with your whitelisted number. Do not use a + or 00, just start with the country code, for example 447700900000.

```
$ curl -X POST https://messages-sandbox.nexmo.com/v0.1/messages \
$ -u '5bbb9f1e:.....2mI' \
$ -H 'Content-Type: application/json' \
$ -H 'Accept: application/json' \
$ -d '{
$   "from": { "type": "whatsapp", "number": "14157386170" },
$   "to": { "type": "whatsapp", "number": "$TO_NUMBER" },
$   "message": {
$     "content": {
$       "type": "text",
$       "text": "This is a WhatsApp Message sent from the Messages API"
$     }
$   }
$ }'
```

Izvor: <https://dashboard.nexmo.com/messages/sandbox>

3.3 cURL, što je i gdje se koristi?

cURL je *command-line* alat za dobivanje ili slanje podataka, uključujući datoteke, koristeći protokole i sintaksu URL-a. Idealan je za prijenos podataka sa i prema serveru, slanje zahtjeva i testiranje API-a. (Stenberg, 2015.)

Autor i vodeći programer je Daniel Stenberg, koji je stvorio cURL jer je želio automatizirati dohvaćanje tečajnih listi za korisnike IRC-a (Internet Relay Chat).

cURL dolazi instaliran na većinu Linux sustava, dok se na Windows treba ručno instalirati. Kako je Raspbian OS izgrađen na bazi Debiana, imamo ga dostupnog na korištenje.

cURL skripte se upisuju u terminal i izvršavaju tipkom Enter. Na slici 8. *WhatsApp, Viber i Facebook Messenger Sandbox set up* uočavamo da se sintaksa razlikuje od ostalih programskih jezika.

cURL skripta se uvijek započinje sa ključnom riječi „curl“. Sve nakon riječi „curl“ pa dok korisnik ne pritisne tipku Enter, računalo smatra dijelom cURL skripte. Slijedeće u našoj cURL skripti je „-X POST“. -X je ključna riječ za slanje zahtjeva na server popraćena ključnom riječi o tipu zahtjeva koji korisnik vrši. To može biti GET, POST, PUT, PATCH i DELETE. U našem primjeru koristimo POST metodu iz jednostavnog razloga što šaljemo podatke API-u. Nakon što smo rekli koji tip zahtjeva vršimo na server, potrebno je reći koji server želimo kontaktirati, a to se jednostavno vrši unošenjem linka na taj server. (Traversy Media, 2017.)

Sljedeće što je Vonage-ovom API-u potrebno su podaci o korisniku, *header* informacije i sami podaci koje šaljemo.

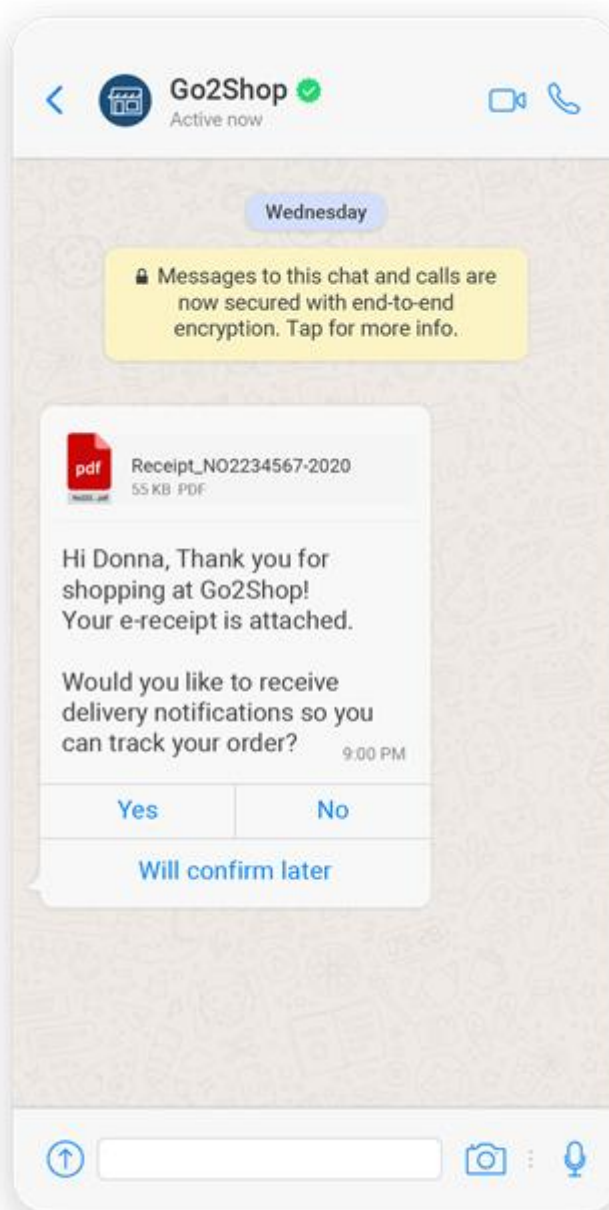
Ključnom riječi „-u“ (user) šaljemo naše korisničke podatke (korisničko ime, ako je postavljena lozinka lozinku, privatni ključ, itd.). Dalje šaljemo informacije o *header-u* korištenjem „-H“. One govore API-u koji format podataka se očekuje, u našem slučaju JSON (JavaScript Object Notation) format.

Sada kada API zna što radi i s kojim tipom podataka, prosljeđujemo mu same podatke koristeći ključnu riječ „-d“ (data). Unutar Stringa koji unosimo u jednostruke navodnike, kao i prijašnje (osim linka na server), unosimo podatke koje API koristi kako bi izvršio svoju funkciju. Tu mu govorimo koji tip platforme (WhatsApp, Viber, Facebook Messenger, itd.) koristi pošiljatelj (linija „from“:), koji tip platforme koristi primatelj (linija „to“:) i što se nalazi u samom tijelu poruke („message“:).

Unutar tijela poruke trebamo definirati koji tip poruke želimo. Poruke u WhatsApp-u prema tipu mogu biti:

1. Tekstualne
2. Video
3. Audio
4. Kontaktne
5. Slikovne
6. Lokacijske
7. Datotečne
8. Quick Reply Button
9. Link Button

Slika 10. Primjer Quick Reply Button poruke u WhatsApp-u

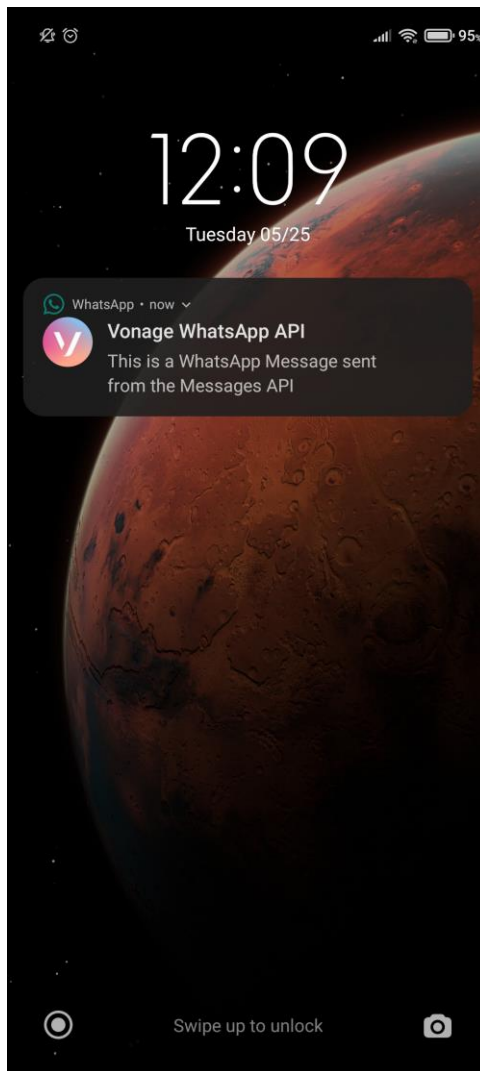


Izvor: <https://www.infobip.com/docs/whatsapp/message-types>

Za naše potrebe koristimo tekstualni tip poruke (slika 8. *WhatsApp, Viber i Facebook Messenger Sandbox set up*), a prosljeđeni tekst je uneseni tekst u liniju „text“:

Izvršavanjem skripte tipkom Enter korisniku dolazi poruka na WhatsApp.

Slika 11. Primjer primljene poruke



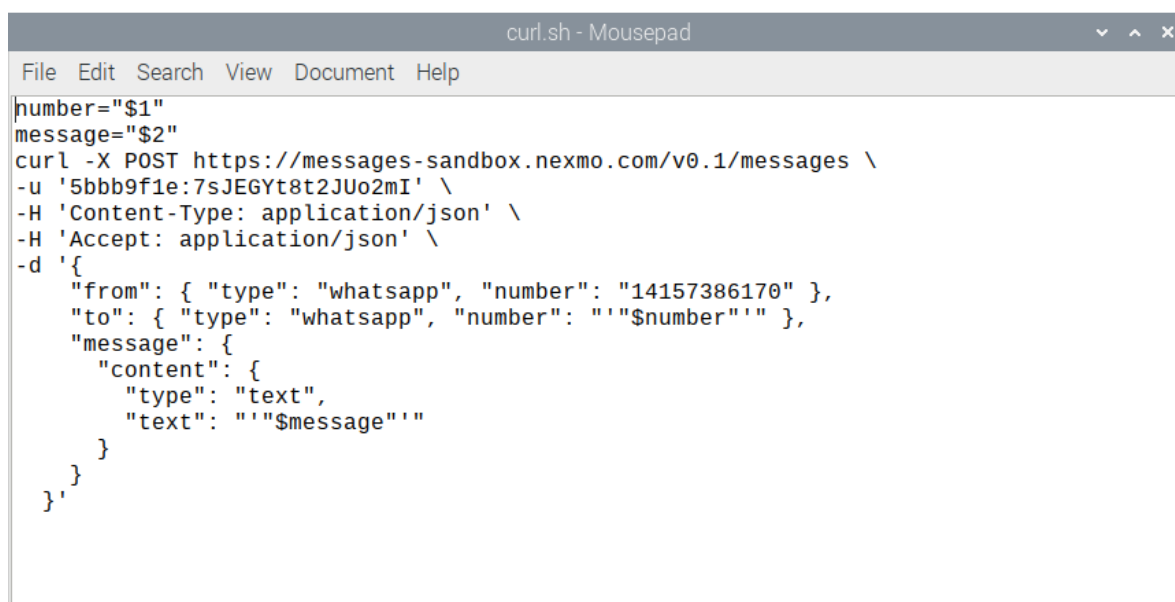
Izvor: Izradio autor

3.4 Prilagođena cURL skripta za projekt

cURL skripta korištena za slanje probne poruke ima neke velike nedostatke za zahtjeve projekta. Prvi je što se broj primatelja izravno upisuje u sam kod, a drugi što se isto radi i za tijelo poruke. To bi značilo da treba imati posebnu skriptu za svakog zaposlenika i za svaki mogući slučaj.

Ta je prepreka riješena na način da se na početku skripte, prije ključne riječi „curl“, deklariraju varijable **number** i **message**.

Slika 12. Prilagođena cURL skripta korištena za projekt

A screenshot of a terminal window titled "curl.sh - Mousepad". The window contains a cURL command and a JSON body. The command uses variables \$1 and \$2 for the number and message respectively. The JSON body is a WhatsApp message object with a "from" field, a "to" field that uses "\$number", and a "message" field with a "text" field that uses "\$message".

```
number="$1"
message="$2"
curl -X POST https://messages-sandbox.nexmo.com/v0.1/messages \
-u '5bbb9f1e:7sJEGYt8t2JUo2mI' \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
-d '{
  "from": { "type": "whatsapp", "number": "14157386170" },
  "to": { "type": "whatsapp", "number": "'"$number"'" },
  "message": {
    "content": {
      "type": "text",
      "text": "'"$message"'"
    }
  }
}'
```

Izvor: Izradio autor

Te se varijable inicijaliziraju na vrijednosti \$1 i \$2, što predstavlja kojim redom se varijable prosljeđene cURL skripti spremaju u varijable **number** i **message**. Varijabla **number** je inicijalizirana na \$1 vrijednost što znači da se **prva** varijabla prosljeđena cURL skripti sprema u nju. Varijabla **message** je inicijalizirana na \$2 vrijednost što znači da se **druga** varijabla prosljeđena cURL skripti sprema u nju.

4. Bash (Bourne Again Shell)

Bash (Bourne Again Shell) je naredbeni jezik za Unix shell koji je napisao Brian Fox za GNU (Gnu Not Unix) projekt kao besplatnu zamjenu softvera za Bourne shell. Prvo izdanje 1989. godine, koristi se kao zadana ljuska za većinu Linux distribucija. (Stevens, 2001.)

Slika 13. Primjer izvršenih komandi u Bash-u

```
mars@marsmain ~ $ pwd
/home/mars
mars@marsmain ~ $ cd /usr/portage/app-shells/bash
mars@marsmain /usr/portage/app-shells/bash $ ls -al
total 130
drwxr-xr-x  3 portage portage 1024 Jul 25 10:06 .
drwxr-xr-x 33 portage portage 1024 Aug  7 22:39 ..
-rw-r--r--  1 root  root   35808 Jul 25 10:06 ChangeLog
-rw-r--r--  1 root  root   27002 Jul 25 10:06 Manifest
-rw-r--r--  1 portage portage  4645 Mar 23 21:37 bash-3.1_p17.ebuild
-rw-r--r--  1 portage portage  5977 Mar 23 21:37 bash-3.2_p39.ebuild
-rw-r--r--  1 portage portage  6151 Apr  5 14:37 bash-3.2_p48-r1.ebuild
-rw-r--r--  1 portage portage  5988 Mar 23 21:37 bash-3.2_p48.ebuild
-rw-r--r--  1 portage portage  5643 Apr  5 14:37 bash-4.0_p10-r1.ebuild
-rw-r--r--  1 portage portage  6230 Apr  5 14:37 bash-4.0_p10.ebuild
-rw-r--r--  1 portage portage  5648 Apr 14 05:52 bash-4.0_p17-r1.ebuild
-rw-r--r--  1 portage portage  5532 Apr  8 10:21 bash-4.0_p17.ebuild
-rw-r--r--  1 portage portage  5660 May 30 03:35 bash-4.0_p24.ebuild
-rw-r--r--  1 root  root    5660 Jul 25 09:43 bash-4.0_p28.ebuild
drwxr-xr-x  2 portage portage  2048 May 30 03:35 files
-rw-r--r--  1 portage portage   468 Feb  9 04:35 metadata.xml
mars@marsmain /usr/portage/app-shells/bash $ cat metadata.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pkgmetadata SYSTEM "http://www.gentoo.org/dtd/metadata.dtd">
<pkgmetadata>
<herd>base-system</herd>
<use>
  <flag name='bashlogger'>Log ALL commands typed into bash; should ONLY be
used in restricted environments such as honeypots</flag>
  <flag name='net'>Enable /dev/tcp/host/port redirection</flag>
  <flag name='plugins'>Add support for loading builtins at runtime via
'enable'</flag>
</use>
</pkgmetadata>
mars@marsmain /usr/portage/app-shells/bash $ sudo /etc/init.d/bluetooth status
Password:
+ status: started
mars@marsmain /usr/portage/app-shells/bash $ ping -q -c1 en.wikipedia.org
PING rr.esams.wikimedia.org (91.198.174.2) 56(84) bytes of data.

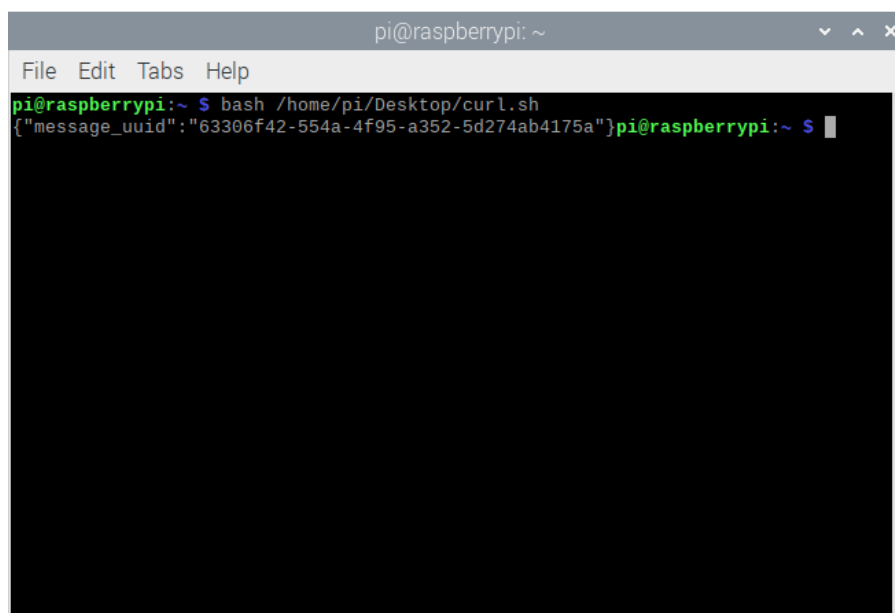
--- rr.esams.wikimedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 2ms
rtt min/avg/max/mdev = 49.820/49.820/49.820/0.000 ms
mars@marsmain /usr/portage/app-shells/bash $ grep -i /dev/sda /etc/fstab | cut --fields=3
/dev/sda1      /boot
/dev/sda2      none
/dev/sda3      /
mars@marsmain /usr/portage/app-shells/bash $ date
Sat Aug  8 02:42:24 MSD 2009
mars@marsmain /usr/portage/app-shells/bash $ lsmod
Module          Size  Used by
rndis_wlan      23424  0
rndis_host      8696   1 rndis_wlan
cdc_ether        5672   1 rndis_host
usbnet          18688   3 rndis_wlan,rndis_host,cdc_ether
parport_pc      38424  0
fglrx           2388128 20
parport         39648   1 parport_pc
iTCO_wdt         12272  0
i2c_i801         9380   0
mars@marsmain /usr/portage/app-shells/bash $
```

Izvor: [https://en.wikipedia.org/wiki/Bash_\(Unix_shell\)#/media/File:Bash_screenshot.png](https://en.wikipedia.org/wiki/Bash_(Unix_shell)#/media/File:Bash_screenshot.png)

Bash je procesor naredbi koji se obično izvodi u tekstualnom prozoru (terminal za Linux, cmd za Windows) gdje korisnik upisuje naredbe koje izazivaju radnje. Bash također može čitati i izvršavati naredbe iz datoteke koja se zove *shell script*. Ovo je ključna informacija koja nam omogućuje da cURL skriptu (slike 8. *WhatsApp, Viber i Facebook Messenger Sandbox set up* i 11. *Primjer primljene poruke*) možemo spremiti u neku datoteku i naknadno ju izvršiti pozivanjem putanje do iste, umjesto da ju se mora svaki put ispisivati u terminal. (Hamilton, 2008.)

cURL skripta „curl“ korištena za projekt (slika 11.) je spremljena na Desktop Raspberry Pi-a u *shell script* formatu, .sh ekstenzija.

Slika 14. Izvršavanje cURL skripte iz Shell-a korištenjem Bash komande



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ bash /home/pi/Desktop/curl.sh  
{"message_uuid": "63306f42-554a-4f95-a352-5d274ab4175a"}pi@raspberrypi:~ $
```

Izvor: Izradio autor

cURL skriptu možemo izvršiti iz terminala komandom „bash <putanja/curl.sh>“. Sljedeća linija sadrži **message_uuid** i predstavlja uspješno poslanu, ali ne nužno i zaprimljenu, WhatsApp poruku. API ne provjerava je li poruka došla do korisnika, npr. ako korisnik nije spojen na internet, ali ona je poslana i do korisnika će doći kada isti bude imao pristup internetu.

5. Cron

Jedna problematika kojom se projekt bavi je kako zaposlenika redovito obavještavati o tome u kojim se prostorijama nije ugasilo svjetla. Dosada smo riješili problem slanja poruke, a sada trebamo tu poruku i slati, redovno. Jedno od rješenja je dati portiru pristup serveru (u našem slučaju Raspberry Pi) i obući ga kako pokrenuti skriptu, međutim to je jako loša praksa i donosi više problema nego rješenja, najveće od kojih su sigurnosni rizici, pogotovo ako se na istom serveru nalaze osjetljivi podaci. Drugi razlog zašto ovo nije dobro rješenje je što je to cijeli proces ponovno ovisi o portiru, stoga ponovno imamo element ljudske greške.

Pravo je rješenje automatizirati izvršenje programa, bez da itko, portir ili IT stručnjak, mora išta igdje ručno pokretati, upisivati ili klikati.

Programski uslužni program „cron“, poznat i kao „cron job“ je time-based job scheduler (vremenski raspoređivač poslova) u računalnim operacijskim sustavima na bazi Unix-a. Korisnici koji postavljaju i održavaju softverska okruženja koriste cron za planiranje poslova (komandne naredbe ili *shell skripte*) za povremeno pokretanje u određeno vrijeme, datume ili vremenske intervale. (Admin's Choice, 2013.)

Postavljanje cron job-a kreira „crontable“ datoteku, koja je tablica svih postavljenih, aktivnih i neaktivnih (komentiranih) cron job-ova. Cron se uređuje u tekstualnom editoru kojem se pristupa iz terminala komandom „crontab -e“, gdje je „-e“ ključna riječ za „edit“ naredbu. Primjer druge naredbe bi bio „-l“ za „list“ koja nam u terminal ispiše cijeli crontable.

5.1 Prvi crontable

Prilikom prve upotrebe „crontab –e“ naredbe na određenom sustavu, sustav će nas pitati koji tekstualni editor želimo koristiti i preporučiti nekoliko najkorištenijih.

Slika 15. Odabir tekstualnog editora za uređivanje crontable-a

```
Select an editor. To change later, run 'select-editor'.
 1. /usr/bin/joe
 2. /usr/bin/jstar
 3. /usr/bin/jpico
 4. /usr/bin/jmags
 5. /usr/bin/jed
 6. /bin/nano          <---- easiest
 7. /usr/bin/vim.basic
 8. /usr/bin/rjoe
 9. /usr/bin/mcedit
10. /usr/bin/vim.tiny
11. /bin/elvis-tiny
12. /usr/bin/emacs25
13. /bin/ed

Choose 1-13 [3]: 6
```

Izvor: Izradio autor

U istom terminalu se otvara tekstualni editor u novom prozoru“.

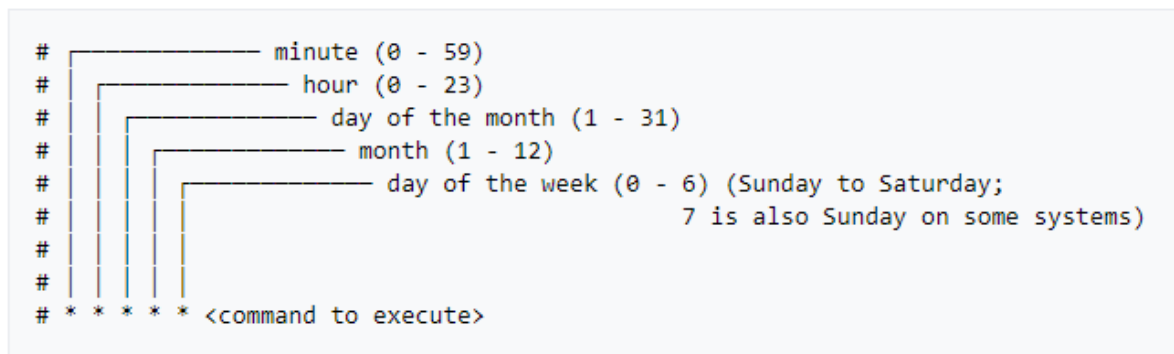
Slika 16. Nano tekstualni editor

```
pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 3.2 /tmp/crontab.TEx5Kg/crontab Modified
# m h dom mon dow  command
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^N Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

Izvor: Izradio autor

Editor je kompletno prazan osim komentirane linije koja nam služi kao podsjetnik za format kojim poslovi zakazuju. (Richard, B. 2021.)

Slika 17. Vizualni prikaz formata za postavljanje cron job-a

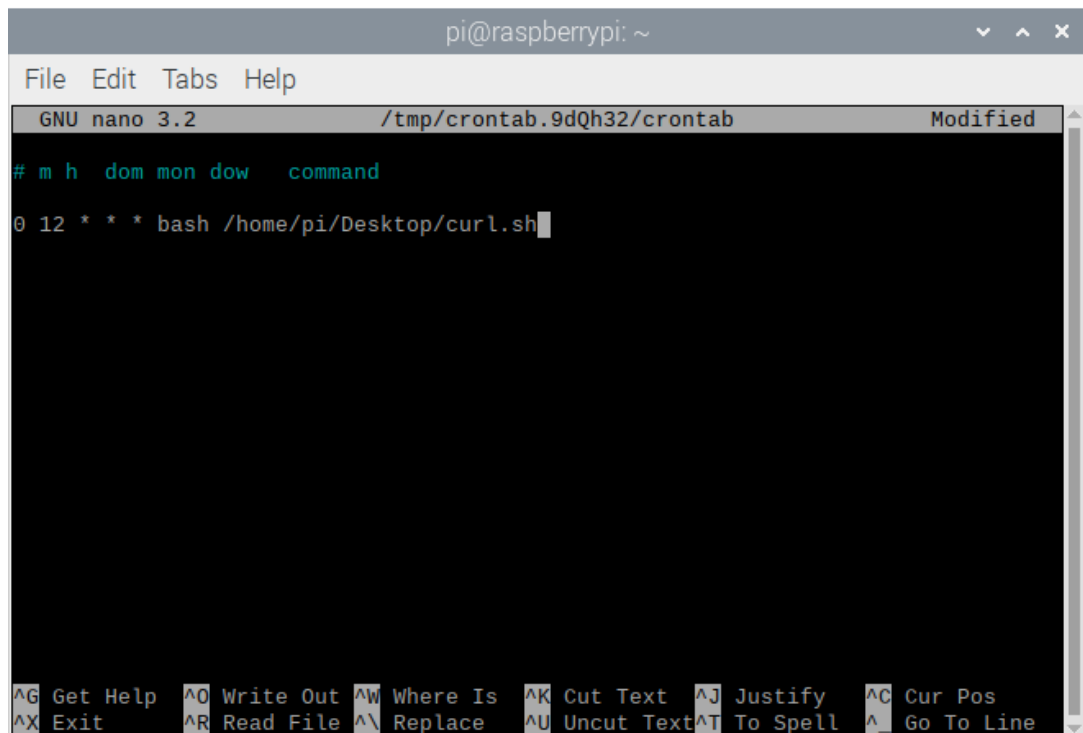


Izvor <https://www.hostinger.com/tutorials/cron-job>

- **Minuta** — minuta u kojoj se komanda izvršava, 0-59.
- **Sat** — sat kojem se komanda izvršava, 0-23.
- **Dan u mjesecu** — datum na koji se komanda izvršava, 1-31.
- **Mjesec** — mjesec u kojem se komanda izvršava, 1-12.
- **Dan u tjednu** — dan u tjednu u kojem se komanda izvršava, 0-7.

Tako možemo odrediti da se naša cURL skripta izvodi svakog dana u 12 sati i 0 minuta. To bi smo napisali ovako:

Slika 18. Primjer cron job-a



```
pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 3.2 /tmp/crontab.9dQh32/crontab Modified
# m h dom mon dow command
0 12 * * * bash /home/pi/Desktop/curl.sh
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^ _ Go To Line
```

Izvor: Izradio autor

Cron je jako moćan alat koji se često koristi za razne poslove poput backup-a podataka i provjera novih dostupnih update-ova i više.

Cron nudi i više opcija od samo definiranja specifičnih vremena i datuma pokretanja naredbe. (Comtronic Blog, 2020.) Neke od drugih mogućnosti su:

- **Zvezdica (*)** — poput SQL-a, * predstavlja „any”, tj „bilo koji”. Ako se na trećem mjestu „cron job-a nalazi * to znači da se izvršava na bilo koji datum, tj. neovisno o datumu.

- **Zarez (,)** — Koristi se za unošenje dvije ili više vrijednosti za isti parameter, ako želimo naredbu izvršiti u 20. i 40. minuti sata, u prvo polje bi smo napisali „20,40”.
- **Hyphen (-)** — Koristi se za unošenje raspona vrijednosti, ako želimo naredbu izvršavati u prvom tjednu nekog mjeseca, u treće polje bi smo upisali „1-7”.
- **Slash (/)** — Koristi se za definiranje vremenskog intervala, ako želimo naredbu izvršavati svakih 5 minuta, u prvo polje bi smo napisali */5.

Slika 19. Primjer cron job-a koji se izvršava svakih 5 minuta

The screenshot shows a terminal window titled 'pi@raspberrypi: ~'. Inside, the GNU nano 3.2 editor is open to the file '/tmp/crontab.RB4TE1/crontab'. The editor's status bar indicates it has been 'Modified'. The content of the crontab file is as follows:

```
# m h dom mon dow  command
*/5 * * * * bash /home/pi/Desktop/curl.sh
```

The bottom of the terminal shows standard nano editor shortcuts: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^C Cur Pos, ^X Exit, ^R Read File, ^N Replace, ^U Uncut Text, ^T To Spell, and ^_ Go To Line.

Izvor: Izradio autor

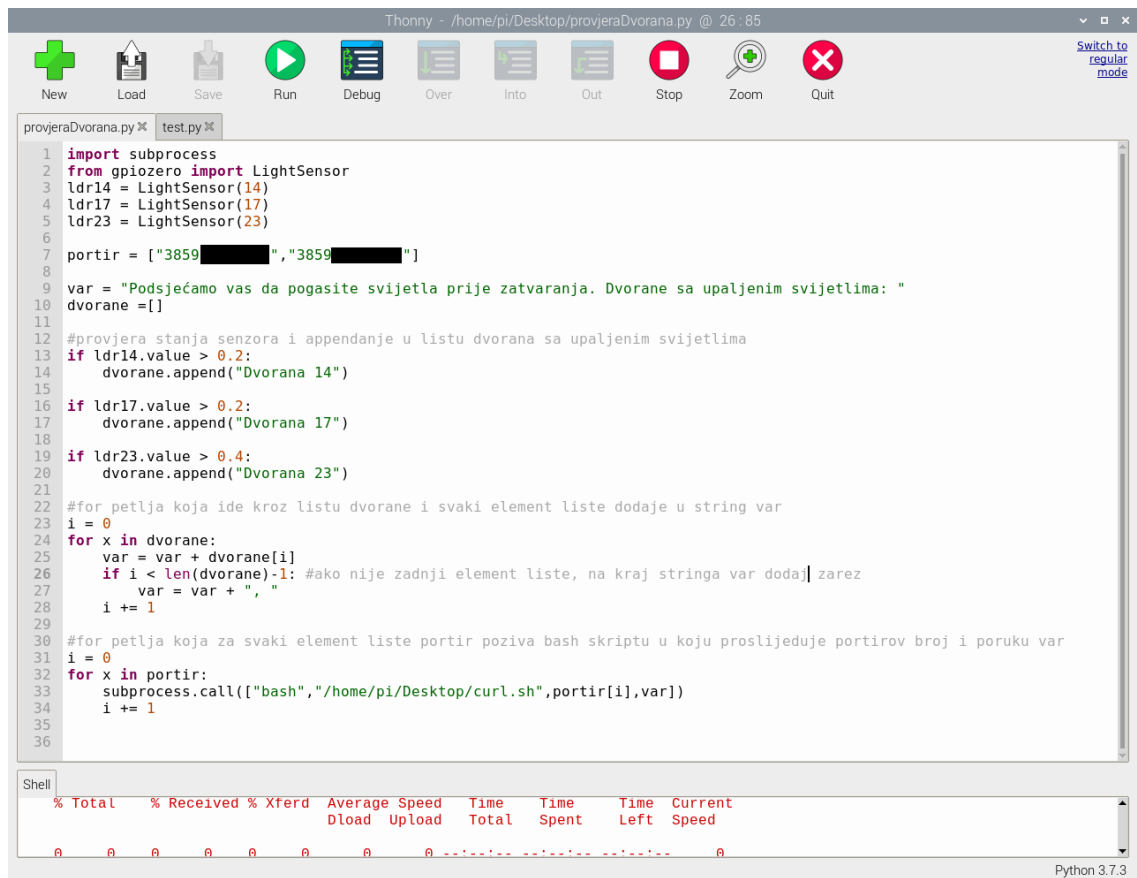
Korištenjem crontab-a vrlo je jednostavno i kvalitetno riješen problem automatiziranog izvršavanja skripte za slanje poruka portirima. Jedini su zahtjevi za uspješno izvršavanje cron job-ova ti da je Raspberry Pi uključen i, u našem slučaju, da ima pristup Internet vezi. Nedostatak ovog pristupa je taj što bi ovaj sustav portirima slao poruke i praznicima i blagdanima, ukoliko padaju na jedan od dana od ponedjeljka do petka jer se radni dan definira sa „1-5“ u petom polju.

6. Python

Python je skriptni programski jezik opće namjene koji se može koristiti za razne svrhe, od malih i velikih lokalnih programa, do cijelih aplikacija i web aplikacija.

Za ovaj projekt Python3 je idealno okruženje za pisanje programa koji će ujediniti sve prijašnje opisane elemente projekt u jedan izvršivi kod.

Slika 20. Programski kod u Python-u



```
1 import subprocess
2 from gpiozero import LightSensor
3 ldr14 = LightSensor(14)
4 ldr17 = LightSensor(17)
5 ldr23 = LightSensor(23)
6
7 portir = ["3859", "3859"]
8
9 var = "Podsjećamo vas da pogasite svijetla prije zatvaranja. Dvorane sa upaljenim svijetlima: "
10 dvorane = []
11
12 #provjera stanja senzora i appendanje u listu dvorane sa upaljenim svijetlima
13 if ldr14.value > 0.2:
14     dvorane.append("Dvorana 14")
15
16 if ldr17.value > 0.2:
17     dvorane.append("Dvorana 17")
18
19 if ldr23.value > 0.4:
20     dvorane.append("Dvorana 23")
21
22 #for petlja koja ide kroz listu dvorane i svaki element liste dodaje u string var
23 i = 0
24 for x in dvorane:
25     var = var + dvorane[i]
26     if i < len(dvorane)-1: #ako nije zadnji element liste, na kraj stringa var dodaj zarez
27         var = var + ", "
28     i += 1
29
30 #for petlja koja za svaki element liste portir poziva bash skriptu u koju prosljeđuje portirov broj i poruku var
31 i = 0
32 for x in portir:
33     subprocess.call(["bash", "/home/pi/Desktop/curl.sh", portir[i], var])
34     i += 1
35
36
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current	
			Dload	Upload	Total	Spent	Left	Speed
0	0	0	0	0	0	0	0	0

Python 3.7.3

Izvor: Izradio autor

Program će se podijeliti u 3 dijela: import i deklaracija, očitavanje senzora i slanje poruka.

6.1 Import i deklaracija

Slika 21. Import i deklaracija

```
1 import subprocess
2 from gpiozero import LightSensor
3 ldr14 = LightSensor(14)
4 ldr17 = LightSensor(17)
5 ldr23 = LightSensor(23)
6
7 portir = ["38595", "38599"]
8
9 var = "Podsjećamo vas da pogasite svjetla prije zatvaranja. Dvorane sa upaljenim svjetlima: "
10 dvorane = []
11
```

Izvor: Izradio autor

Na početku Python programa moramo importati subprocess i iz gpiozero knjižnice LightSensor. U 3 varijable se sprema pin na koji su senzori spojeni na GPIO-u, u našem slučaju 3 senzora su spojena na pinove 14, 17 i 23, stoga predstavljaju Dvorane 14, 17 i 23.

Ispod toga deklariramo String array **portir** i punimo ju *whitelist-anim* mobilnim brojevima naših portira.

Potom se deklarira varijabla **var**. To je varijabla koja će se proslijediti cURL skripti i predstavlja tijelo poruke koja se šalje portiru.

Nakon nje se deklarira prazan String array **dvorane** koji ćemo puniti brojem dvorana za koje odgovarajući senzor detektira dovoljnu količinu svjetla.

6.2 Očitavanje senzora

Slika 22. Kod za očitavanje stanja senzora

```
12 #provjera stanja senzora i appendanje u listu dvorana sa upaljenim svjetlima
13 if ldr14.value > 0.2:
14     dvorane.append("Dvorana 14")
15
16 if ldr17.value > 0.2:
17     dvorane.append("Dvorana 17")
18
19 if ldr23.value > 0.4:
20     dvorane.append("Dvorana 23")
21
```

Izvor: Izradio autor

Stanje senzora se očitava postojećom funkcijom **value**. Korištenjem **if** funkcije odmah se u **dvorane** funkcijom **append** unosi element liste koji glasi „Dvorana X“ ukoliko je stanje na senzoru iznad 0.2.

6.3 Slanje poruka

Slika 23. Kod za izvršenje shell skripte koja sadrži cURL naredbu

```
22 #for petlja koja ide kroz listu dvorane i svaki element liste dodaje u string var
23 i = 0
24 for x in dvorane:
25     var = var + dvorane[i]
26     if i < len(dvorane)-1: #ako nije zadnji element liste, na kraj stringa var dodaj zarez
27         var = var + ", "
28     i += 1
29
30 #for petlja koja za svaki element liste portir poziva bash skriptu u koju prosljeđuje portirov broj i poruku var
31 i = 0
32 for x in portir:
33     subprocess.call(["bash", "/home/pi/Desktop/curl.sh", portir[i], var])
34     i += 1
35
```

Izvor: Izradio autor

U prvoj petlji se ide kroz svaki element u listi **dvorane** i pribraja ga se String varijabli **var**. Potom se **if** funkcijom provjerava je li to bio zadnji element liste i ako **nije**, varijabli **var** se također pribraja zarez i razmak.

Nakon što su provjereni svi senzori, popunjena lista dvorana sa uključenim svjetlima i finalizirana vrijednost varijable **var**, inkrementalna varijabla **i** se ponovo postavlja na nulu i pokreće se druga **for** petlja.

Ona ide kroz listu portira, u našem primjeru 2, i za svaki element liste nad objektom **subprocess** izvršava funkciju **call**. Toj funkciji se prosljeđuju sljedeće vrijednosti:

1. „bash“
2. „/home/pi/Desktop/curl.sh“
3. portir[i]
4. var

Funkcija **call** nad objektom **subprocess** prosljeđene vrijednosti izvršava kao da su upisane u sam terminal. Prva varijabla, String „bash“ govori sustavu kakvu vrstu datoteke da izvrši, druga varijabla „home/pi/Desktop/curl.sh“ prosljeđuje putanju do te datoteke.

Treća i četvrta varijabla, **portir[i]** i **var**, su varijable čije su vrijednosti lokalno spremljene na računalo tokom izvršavanja Python koda i prosljeđuju se cURL skripti koja ih sprema u varijable **number** i **message** (slika 12.).

6.4 Povratna informacija

U *shell* (konzola) nam se ispisuje povratna informacija o poslanim porukama, ili eventualna greška (error).

Slika 24. Povratne informacije

```
>>> %Run provjeraDvorana.py
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   Dload  Upload  Total    Dload  Upload  Total    Spent    Left    Speed

  0     0     0     0     0     0     0     0  --:--:--  --:--:--  --:--:--     0
100   384  100    55   100   329    56   337  --:--:--  --:--:--  --:--:--   393
100   384  100    55   100   329    56   337  --:--:--  --:--:--  --:~:~:~   393
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   Dload  Upload  Total    Dload  Upload  Total    Spent    Left    Speed

  0     0     0     0     0     0     0     0  --:~:~:~  --:~:~:~  --:~:~:~     0
100   384  100    55   100   329    106   636  --:~:~:~  --:~:~:~  --:~:~:~   744
{"message_uuid": "bc3b20ed-8cf0-482e-a3a5-8f7e41a79235"}{"message_uuid": "f2fff502-770d-4819-89c0-f4e020716512"}
```

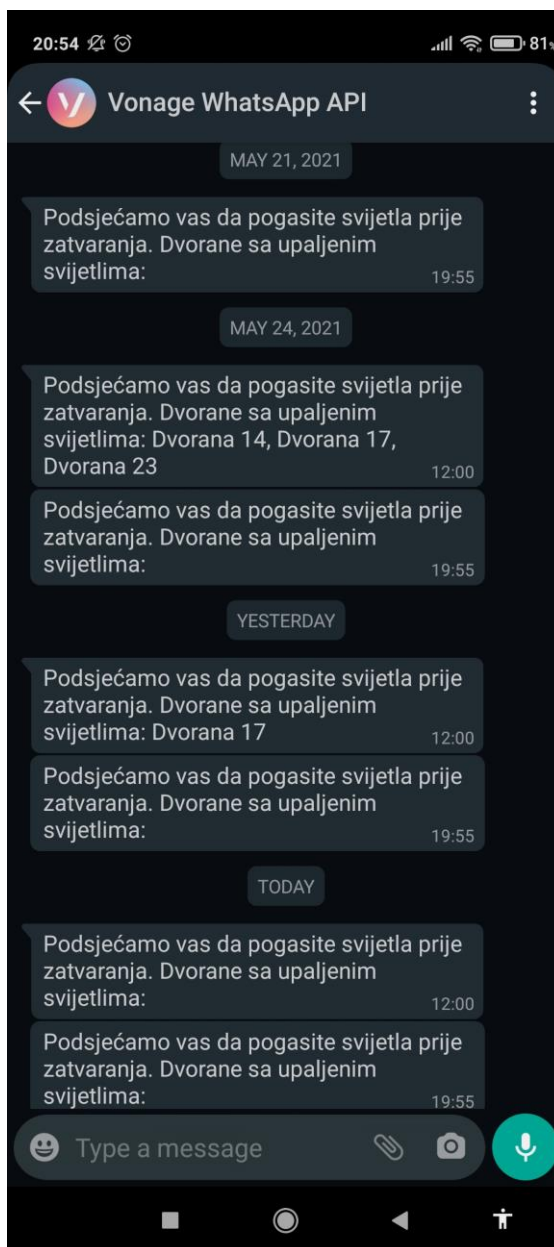
Izvor: Izradio autor

Iz njih možemo iščitati razne podatke poput prosječne brzine downloada i uploada za svaku pojedinu poruku.

7. Zaprimljene poruke

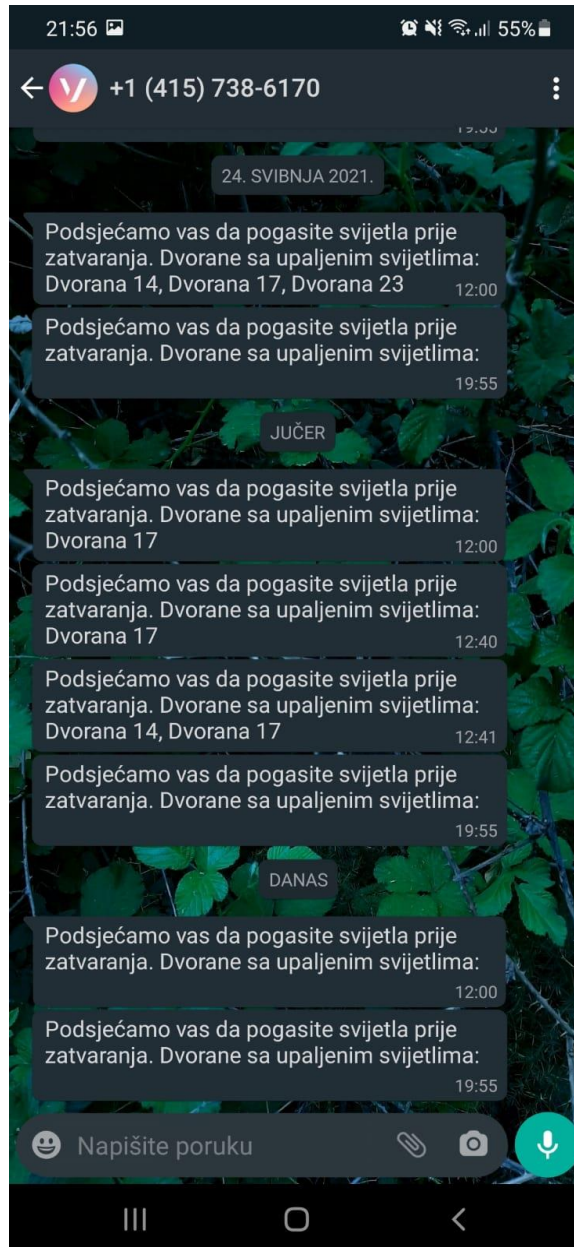
Na sljedećim slikama prikazane su zaprimljene WhatsApp poruke poslane sa Raspberry Pi-a dvojici korisnika automatiziranim izvršavanjem Python programa *provjeraDvorana.py*.

Slika 25. Primljene poruke na prvi broj



Izvor: Izradio autor

Slika 26. Primljene poruke na drugi broj



Izvor: Izradio autor

Na slikama 25. i 26. se vidi kako svaki radni dan dolaze dvije poruke, jedna u 12:00 i druga u 19:55. Na slici 25. se vidi kako poruka nije poslana 22.05. i 23.05., razlog tome je što je to bio vikend. Na slici 26. se vide dvije dodatne poruke, u 12:40 i 12:41, te su poruke poslane ručno u svrhu *debuggiranja* greške u Python kodu.

8. Zaključak

Projekt izrade automatiziranog sustava za dojavu informacija korisniku je bio moj osobni izazov samome sebi, nekakav *limit test* svojih sposobnosti i vještine korištenja stečenog znanja tokom studiranja.

Tokom izrade projekta naišao sam na razne prepreke, neke od kojih su prijeđene, druge zaobiđene. Projekt je bio jako zanimljiv, zabavan i poučan za izradu. Koristio sam razne tehnologije i uređaje s kojima nikada prije nisam imao kontakta (npr. Raspberry Pi), prije definiranja koji će točno zadatak za projekt biti i kako će glasiti, napravio sam par manjih, ali jednako zabavnih projekata, također sa Raspberry Pi-em, ali nijedan opsežan i edukativan poput ovog.

Ovaj me projekt naučio kako koristiti cron, bash, cURL i API, tehnologije korištene svugdje i u masivnim količinama, a za koje nisam ni znao kao pojam, izuzev API-a, prije bavljenja projektom zadatkom.

Projekt je bio uspješan na većoj razini nego sam se nadao, iako još uvijek ograničen u nekim aspektima pošto koristi usluge besplatnog, javnog *sandbox* API-a, za razliku od dalje razvijenog poslovnog, partnerskog API-a koji je rezerviran za poslovne partnere. Partnerski API podržava funkcionalnosti gdje korisnik šalje poruku API-u, i isti mu odgovara. Poput Apple-ovog virtualnog asistenta Siri, kojem možete postaviti pitanje poput „Gdje je najbliži restoran?“ i dobiti odgovor u roku od par sekundi. Na sličan način radi usluga teleoperatera koju bonaši često koriste, „Stanje“. Na određeni broj (npr. 13888) korisnik pošalje poruku „Stanje“ i u roku od par sekundi mu dolazi SMS poruka sa količinom preostalih minuta, poruka i megabajta/gigabajta na raspolaganju.

U svojoj trenutnoj i finalnoj verziji projekt je uspješan, sustav je visoko funkcionalan, pouzdan i precizan, no ostavlja dosta prostora za poboljšanje. Kao i svi drugi IT sustavi, ne postoji verzija kojoj se ne može nešto novo dodati ili postojeće poboljšati.

POPIS LITERATURE

Internetski izvori:

1. Admin's Choice (2013.), <https://www.adminschoice.com/crontab-quick-reference> (Rujan 2013.)
2. Comtronic Blog (2020.), <https://comtronic.com.au/automation-with-cron-job-on-centos-8/> (6. travnja 2020.)
3. Hamilton, Naomi (2008.), https://web.archive.org/web/20110706103704/http://www.computerworld.com.au/article/222764/a-z_programming_languages_bash_bourne-again_shell/?pp=2&fp=16&fpid=1 (30. svibnja 2008.)
4. Raspberry Pi (2021.), Raspberry Pi Foundation - About Us, <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/>
5. Raspberry Pi (2021.), Raspberry Pi Foundation – Laser tripwire, <https://projects.raspberrypi.org/en/projects/laser-tripwire/2>
6. Raspberry Pi (2021.), Raspberry Pi Foundation - Operating System Images, <https://www.raspberrypi.org/software/operating-systems/>
7. Richard, B. (2021.), <https://www.hostinger.com/tutorials/cron-job> (18. svibnja 2021.)
8. Stenberg, Daniel (2015.), <https://daniel.haxx.se/blog/2015/03/20/curl-17-years-old-today/> (20. ožuka 2015.)
9. Stevens, Al (2001.), <https://www.drdobbs.com/i-almost-get-a-linux-editor-and-compiler/184404693> (1. lipnja 2001.)
10. Traversy Media (2017.), <https://www.youtube.com/watch?v=7XUibDYw4mc> (8. svibnja 2017.)
11. Upton, Ebon (2019.). „[Ebon Upton tweet – sales up to 30 million](#)“ (14. prosinca 2019.)

POPIS KRATICA

1. I/O – Input/Output
2. CPU – Central Processing Unit
3. GPIO – General Purpose Input Output
4. USB – Universal Serial Bus
5. microHDMI – micro High-Definition Multimedia Interface
6. OS – Operating System ili Operativni Sustav
7. GUI – Graphical User Interface
8. IDE – Integrated Development Enviroment
9. LDR – Light Detecting Resistor
10. M-Ž – Muško-Žensko
11. API – Application Programming Interface
12. VoIP – Voice over Internet Protocol
13. IRC – Internet Relay Chat
14. Bash – Bourne Again Shell
15. GNU – Gnu Not Unix

POPIS SLIKA

<i>Slika 1. Konektori i gdje se nalaze na Raspberry Pi 4B pločici</i>	3
<i>Slika 2. Raspberry Pi Imager v1.5</i>	4
<i>Slika 3. Instalacija softvera i aplikacija</i>	5
<i>Slika 4. Recommended Software prozor</i>	6
<i>Slika 5. Shema napajanja eksperimentalne pločice sa Raspberry Pi GPIO-a</i>	7
<i>Slika 6. Strujni krug sa 3 LDR-a i kondenzatora spojeni na GPIO</i>	8
<i>Slika 7. Vonage SMS API Dashboard</i>	10
<i>Slika 8. WhatsApp, Viber i Facebook Messenger Sandbox set up</i>	11
<i>Slika 9. Sample kod za kontaktiranje WhatsApp API-a</i>	13
<i>Slika 10. Primjer Quick Reply Button poruke u WhatsApp-u</i>	16
<i>Slika 11. Primjer primljene poruke</i>	17
<i>Slika 12. Prilagođena cURL skripta korištena za projekt</i>	18
<i>Slika 13. Primjer izvršenih komandi u Bash-u</i>	19
<i>Slika 14. Izvršavanje cURL skripte iz Shell-a korištenjem Bash komande</i>	20
<i>Slika 15. Odabir tekstualnog editora za uređivanje crontable-a</i>	22
<i>Slika 16. Nano tekstualni editor</i>	22
<i>Slika 17. Vizualni prikaz formata za postavljanje cron job-a</i>	23
<i>Slika 18. Primjer cron job-a</i>	24
<i>Slika 19. Primjer cron job-a koji se izvršava svakih 5 minuta</i>	25
<i>Slika 20. Programski kod u Python-u</i>	26
<i>Slika 21. Import i deklaracija</i>	27
<i>Slika 22. Kod za očitavanje stanja senzora</i>	28
<i>Slika 23. Kod za izvršenje shell skripte koja sadrži cURL naredbu</i>	28
<i>Slika 24. Povratne informacije</i>	29
<i>Slika 25. Primitljene poruke na prvi broj</i>	30
<i>Slika 26. Primitljene poruke na drugi broj</i>	31