ŠOLSKI CENTER CELJE

Srednja šola za kemijo, elektrotehniko in računalništvo

ZAKLEP Z NFC IN PODATKOVNO BAZO

Mentor: Borut Slemenšek, univ. dipl. inž.

Avtorja: Amadej Šuperger, R4B

Lars Kolar, R4B

Celje, marec 2018

IZJAVA*

Mentor (-ica) , Borut Slemenšek, v skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi naslovom
Zaklep z NFC in podatkovno bazo,
katere avtorji (-ice) so Lars Kolar, Amadej Supergan, ___, ___:

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo (-ičino) dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje

Celje, 12. 3. 2018

žig šole

ŠOLSKI CENTER CELJE
SREDNJA ŠOLA ZA KEMIJO, ELEKTROTEHNIKO IN RAČUNALNIŠTVO
CELJE, Pot na Lavo 22
1

Podpis mentorja(-ice)

Podpis odgovorne osebe

*
POJASNILO

V skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja(-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja(-ice) fotografskega gradiva, katerega ni avtor(-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.

Vsebina

Povzetek in ključne besede

Povzetek

Raziskovala sva NFC tehnologijo, podatkovno bazo Firebase v povezavi s telefoni z operacijskim sistemom Android. Ključno vprašanje te naveze je bilo, kako bi jih bilo mogoče uporabiti pri avtomatizaciji doma, predvsem, kako bi lahko odklepali ter zaklepali katerakoli vrata tudi če nismo doma. K reševanju tega problema so naju spodbudili številni videi na portalu YouTube. Znanje, ki sva ga za izpeljavo projekta rabila, sva nadgrajevala z metodami analize dokumentov in metod eksperimentiranja pri izdelavi programske opreme. Ugotovila sva, da imajo takšni sistemi zelo enostavno idejo, da pa s svojo funkcionalnostjo hitro postanejo zelo kompleksni. Rezultat najine raziskave in dela je dobro delujoč ter funkcionalen izdelek, ki omogoča delovanje osnovnih funkcij in bo nedvomno služil kot dober temelj za še nadaljnji razvoj in širitev projekta.

Ključne besede

NFC - Near Field Communication

RASPBERRY PI 3 - mikroračunalnik

FIREBASE – podatkovna baza

ANDROID – operacijski sistem mobilnih telefonov

mikro SD – pomnilniška kartica

OS – operacijski sistem

WIFI – brezžično omrežje

Uvod

Opis/predstavitev raziskovalnega problema

Dandanes je tehnologija in njena vsakodnevna uporaba sestavni del našega življenja. Ena izmed takih novejših tehnologij je tudi NFC, kar je kratica za Near Field Communication. Uporaba te tehnologije omogoča, da se sproži medsebojno sporazumevanje med dvema napravama, ko se ena izmed njih približa drugi. S telefoni bi tako lahko odklepali in zaklepali hiše, zaganjali avtomobile ali se prijavili na delovnem mestu oziroma šoli. Z njim bodo znale komunicirati različne naprave, na primer tudi tiskalniki ali televizorji, in prek vzpostavljenega kanala jim bo mogoče pošiljati različne podatke.

Najinega raziskovalnega dela sva se lotila z željo naučiti se in izdelati sistem za zaklepanje in odklepanje vrat. Pri tem pa se je takoj postavilo kar nekaj vprašanj. Predvsem je to najino znanje, nato je tu cena izdelka, ki naj bo relativno poceni, v primerjavi z drugimi podobnimi sistemi, ali lahko vključiva mobilni telefon, da z njim posredno odpiramo vrata… Imava tudi željo po spoznavanju delovanja in uporabe NFC tehnologije ter računalnika Raspberry PI, ki je zelo uporaben tehnološki pripomoček za »mala hišna opravila«. Spodbuda za to so bili številni videi po internetu o avtomatizaciji hise in podobno.

Hipoteze

Glede na najine želje pred pristopom k raziskovalnemu delu sva postavila sledeče hipoteze:

1) Stroški projekta ne bodo višji od 100 EUR
2) Z aplikacijo na telefonu bo mogoče odklepati/zaklepati hišna vrata
3) Za realizacijo projekta lahko uporabimo računalnik Raspberry PI 3
4) Aplikacijo na telefonu bodo znali uporabljati »navadni« uporabniki
5) Z aplikacijo na telefonu bo mogoče odklepati/zaklepati več vrat

Opis raziskovalnih metod

Zaradi kompleksnosti predmeta sva uporabila kombinacijo različnih metod, saj sva tako lažje prišla do rešitve težave, ki sva jo morala rešiti za dosego želenega cilja.

Pri delu sva sprva uporabila predvsem raziskovalno metodo analize dokumentov. Preden sva se lotila praktičnega dela, sva se morala naučiti rokovati in upravljati z računalnikom Raspberry PI ter NFC. Prebrala in preučila sva veliko dokumentacije, številne objave na forumih ter veliko video posnetkov na portalu YouTube.

Pri programiranju je bila večinoma uporabljena raziskovalna metoda eksperimenta, saj sva s poskusi različnih nastavitev, pri tem pa tudi marsikatere napake, poskušala priti do najbolj optimalne rešitve. Pri tem je bilo seveda potrebno napake odpravljati in ponovno testirati delovanje delovanje sistema.

OSREDNJI DEL NALOGE

V osrednjem delu naloge bova predstavila: nakup strojne opreme, ki je potrebna za ta izdelavo projekta, pripravo aplikacije za mobilni telefon z operacijskim sistemom Android, ki bo uporabljena kot »ključ« za vrata, pripravo podatkovne baze Firebase, preko katere bo android aplikacija komunicirala z računalnikom Raspberry PI, pripravo računalnika Raspberry PI, ki bo nadzoroval elektroniko za krmiljenje ključavnice, in kot zadnje še povezavo celotne strojne opreme v delujoč sistem za upravljanje ključavnice.

Za zaključek naloge bova predstavila še težave, ki so nama vzele največ časa in povzročile največ dela pri nastajanju celotnega projekta.

Nakup strojne opreme

Za ta projekt potrebujemo:

- Mobilni telefon z OS Android, ki podpira NFC tehnologijo in ima dostop do interneta. Takšne telefone lahko dobimo v specializiranih trgovinah za telefone po večini mest v Sloveniji. Ti telefoni imajo več varnostnih funkcij, ki so zelo koristne v primeru kraje telefona, saj bi lahko ropar brez njih z lahkoto prišel v hišo.

Slika 1 - Android telefon

- NFC Mifare Classic 1k kartico, za katero je značilno, da ima 1024B prostora, razdeljenega v 16 sektorjev. To kartico sva izbrala, ker popolnoma zadošča potrebam v tem projektu. Take kartice je mogoče kupiti na naslednji povezavi: https://www.amazon.de

Slika 2 - Mifare Classic 1k card

- Računalnik Raspberry PI 3, za katerega sva se odločila zaradi povezave v WIFI omrežje, saj že v osnovi vsebuje potrebno strojno opremo. Kupila sva ga v Celju v trgovini HTE, nasproti Lidla.

Slika 3- Raspberry pi 3

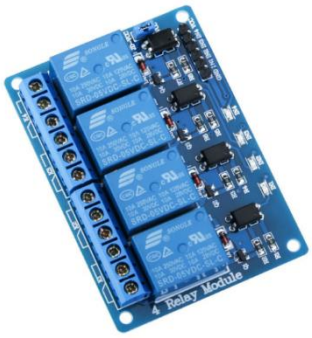
- Povezovalni kabli, ki jih je mogoče dobiti na naslednji povezavi: https://www.amazon.de

Slika 4- Povezovalni kabli

- Elektronska ključavnica. Za izbrani model je značilno, da zdrži približno 3000N sile preden popusti. Ob izpadu napajanja je avtomatsko zaprta (zaklenjena). Kupiti jo je mogoče na naslednji povezavi: https://www.z-home.si/

Slika 5 - Električna ključavnica

- 4-kanalni rele, s katerim nadzorujemo napajanje ključavnice. Naročiti ga je mogoče na naslednji povezavi: https://www.amazon.de/

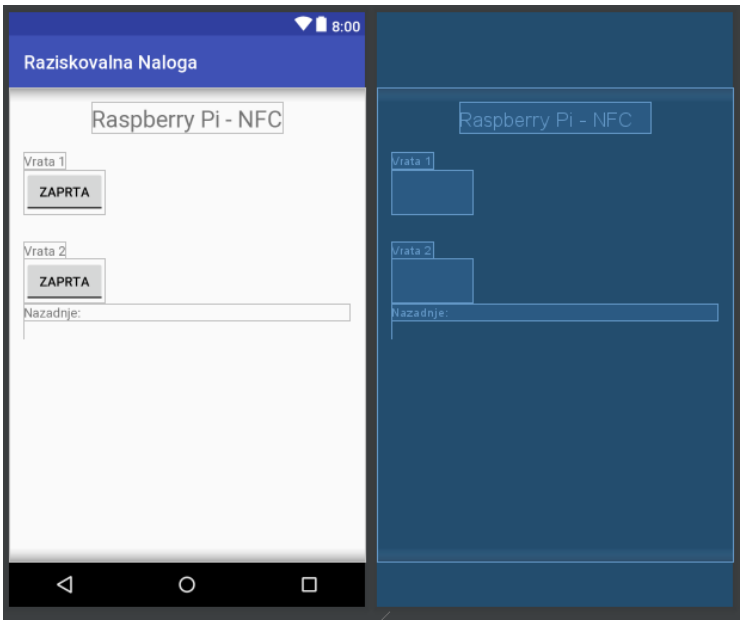


Slika 6 - 4 kanalni rele

Skupna vrednost vse predstavljene strojne opreme vključno s poštnino je malce nad 100 EUR.

Priprava android aplikacije

Aplikacijo za Androidni telefon sva pripravila s programom »Android Studio«. Pred programiranjem sva si zamislila izgled aplikacije – uporabniški vmesnik, ki mora biti za uporabnika preprost in hkrati dovolj pregleden za uporabo. Ker ima aplikacija za začetek le osnovne funkcije, je lahko tudi vmesnik dokaj preprost. Vstavila sva dva gumba za zaklepanje in odklepanje vrat in zraven še nekaj besedili za opise. Slika na desni prikazuje vmesnik, ki se prikaže na telefonu po zagonu aplikacije.



Slika 7 - Izgled aplikacije

Vsak par (besedilo in gumb pod njim) predstavlja vrata, ki jih lahko uporabnik z gumbom upravlja (zaklepa/odklepa). Na samem gumbu se izpisuje trenutno stanje vrat. Vrata se lahko nahajajo le v dveh stanjih in sicer zaprta ali odprta.

Ker je aplikacija namenjena krmiljenju zaklepanja in odklepanja vrat, so zato gumbi za nadzor vrat najpomembnejši in zato tudi na vrhu aplikacije. Za uporabnika pa bi bilo najverjetneje tudi zanimivo zvedeti, kdo je pred našim dostopom do vrat slednja odklepal ali zaklepal. Zato sva dodala še eno besedilo z napisom »Nazadnje:«, v katerega aplikacija zapise, kdo je nazadnje upravljal z vrati.

Celotna koda aplikacije, ki je zapisana pod MainActivity, je v prilogi.

strings.xml:

Poleg glavne kode je potrebno nastaviti še Strings.xml, v katerih so opisi spremenljivk za prikazana besedila. Načeloma bi lahko besedila določili tudi direktno v aplikaciji pod lastnostmi gradnikov, a je ta način primernejši, saj nam omogoča enostavno spreminjanje besedil tudi v primeru, če bi želeli vstaviti besedila za druge aktuatorje ali celo drug jezik.

```
<resources>
    <string name="app_name">Raziskovalna Naloga </string>
    <string name="title">Raspberry Pi - NFC </string>
    <string name="textA">Vrata 1</string>
    <string name="textB">Vrata 2</string>
</resources>
```

AndroidManifest.xml:

Nastaviti je tudi potrebno AndroidManifest.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.user.minfc" >

   <uses-sdk android:minSdkVersion="10" />

   <uses-permission android:name="android.permission.NFC" />

   <uses-feature
      android:name="android.hardware.nfc"
      android:required="true" />

   <uses-permission android:name="android.permission.INTERNET" />

   <android:uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   <android:uses-permission android:name="android.permission.READ_PHONE_STATE" />
   <android:uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/AppTheme" >
      <activity android:name=".MainActivity" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
```

```
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
        <intent-filter>
            <action android:name="android.nfc.action.NDEF_DISCOVERED" />

            <category android:name="android.intent.category.DEFAULT" />

            <data android:mimeType="text/plain" />
        </intent-filter>
        <intent-filter>
            <action android:name="android.nfc.action.TAG_DISCOVERED" />
        </intent-filter>
        <intent-filter>
            <action android:name="android.nfc.action.TECH_DISCOVERED" />
        </intent-filter>

        <meta-data
            android:name="android.nfc.action.TECH_DISCOVERED"
            android:resource="@xml/nfc_tech_filter" />
    </activity>
</application>
```

Priprava podatkovne baze Firebase

Podatkovno bazo Firebase sva pripravila tako, da sva ustvarila nov projekt z imenom »raziskovalna«.

Za tem sva na zavihku na levi strani pripravila strukturo baze. Ustvarila sva »child« izhod (»outputs«), ki ima 2 spremenljivki (A in B). A in B predstavljata v najini aplikaciji vrata, če pa bi želela upravljati druge naprave (luči…), bi lahko zelo enostavno spremenila program in ga priredila za drugo uporabo.

»child« izhod »log« ima zapisane datume in imena telefonov. Pripisan je tudi podatek o načinu delovanja (zaklepanje/odklepanje). Ta opis se prikaže v spodnjem delu uporabniškega vmesnika aplikacije na telefonu.

V bazi sva pripravila še en izhodni podatek: »child« logIndex – v njem je zapisan podatek o uporabniku, ki je z aplikacijo zadnji dostopal do vrat.

Slika 8 - Struktura podatkovne baze

Za varno delovanje aplikacije je potrebno nastaviti tudi varnostna pravila. V najinem primeru so nastavljena na javna, saj je bila aplikacija v toku razvoja le eksperiment in nisva upravljala z dejanskimi vrati v nekem objektu.

Na tako postavljeno podatkovno bazo se ob času pisanja lahko poveže vsak.

Spodaj je slika, ki prikazuje varnostna pravila.

```
{
  "rules": {
      ".read": true,
      ".write": true
  }
}
```

Slika 9 - Varnostna pravila

Priprava računalnika Raspberry PI

Za začetek je potrebno naložiti najnovejšo verzijo Raspberry OS na mikro SD kartico. V najinem primeru je to bil Raspberry stretch November 2017. OS sva naložila na mikro SD kartico s programom Etcher.

Nato na Raspberry priklopimo ekran, miško in tipkovnico. Povežemo ga z internetom in v terminal napišemo ukaz »sudo apt-get install openssh-server«. Ko se aplikacija namesti, lahko upravljamo z računalnikom Raspberry PI preko namiznega računalnika. Ta korak za delovanje celotnega sistema sicer ni potreben, ker pa sva morala celotno aplikacijo še razviti, je bilo delo na tak način veliko lažje.

Nato na namizni računalnik namestimo program Putty. Vanj vpišemo IP naslov računalnika Raspberry PI ter uporabniško ime in geslo, s katerim se predstavimo kot uporabnik z dovoljenji za delo z računalnikom Raspberry PI. Uporabila sva uporabnika »super user«, ki ima v sistemu najvišje pravice, saj sva jih potrebovala za njegovo programiranje. Ta uporabnik uporablja uporabniško ime »pi«, pristopno geslo pa je »raspberry«.

Naslednji korak je priprava programske kode v jeziku Python, ki jo bo izvrševal računalnik Raspberry PI za nadzor nad zaklepanjem vrat. Najprej vpišemo ukaz v terminal »sudo nano nfc.py«. S tem ustvarimo novo datoteko nfc.py, v kateri bo zapisan program. V to datoteko vstavimo naslednjo Python kodo:

```
import RPi.GPIO as GPIO

import time

from firebase import firebase

import threading


class Conection(threading.Thread):


    def __init__(self, re):

        threading.Thread.__init__(self)

        self.re = re

        self.fire = firebase.FirebaseApplication('https://raziskovalna-ea3bd.firebaseio.com/', None)

        self.lastStateA = self.fire.get('/outputs/A', None)

        self.lastStateB= self.fire.get('/outputs/B', None)

        self.re(self.lastStateA, self.lastStateB)


    def run(self):

        A1 = self.lastStateA
```

```
            B1 = self.lastStateB

            while True:
                stateActA = self.fire.get('/outputs/A', None)
                stateActB = self.fire.get('/outputs/B', None)
                A2 = stateActA
                B2 = stateActB

                if A1 != A2:
                    self.re(A2, B2)
                    A1 = A2

                if B1 != B2:
                    self.re(A2, B2)
                    B1 = B2
                time.sleep(0.3)

GPIO.setmode(GPIO.BCM)

pinList = [17, 23]

for i in pinList:
    GPIO.setup(i, GPIO.OUT)
    GPIO.output(i, GPIO.HIGH)

def proces(value_A, value_B):
    if value_A:
        GPIO.output(17, GPIO.LOW)
        print"A on"
    else:
        GPIO.output(17, GPIO.HIGH)
```

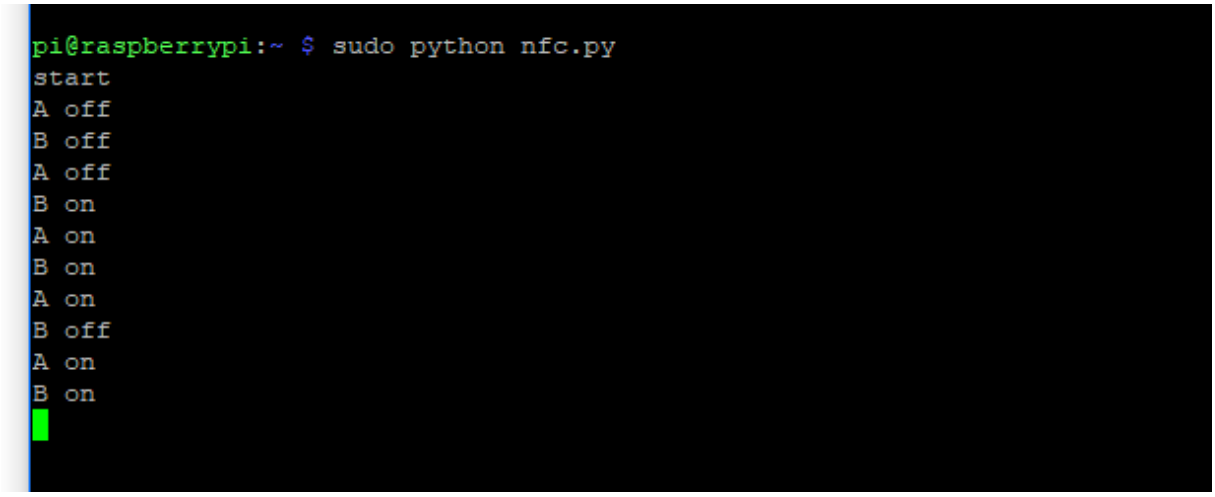
```
            print"A off"


        if value_B:
            GPIO.output(23, GPIO.LOW)
            print"B on"
        else:
            GPIO.output(23, GPIO.HIGH)
            print"B off"


try:
    print"start"
    a = Conection(proces)
    a.deamon = True
    a.start()


except KeyboardInterrupt:
    print"Quit"
    GPIO.cleanup()
```

Spodnja slika prikazuje izpis na zaslonu, ki ga ustvari delujoč program napisan v Pythonu.



Slika 10 - Delujoč Python program

Program ob zagonu izpiše »start«. Nato preverja stanje spremenljivk »outputs« v podatkovni bazi in čaka, da se stanje katerekoli izmed njiju spremeni (bodisi A, B ali obe). A v tem primeru predstavlja vrata 1, B pa vrata 2. Ko program zazna kakršnokoli spremembo, bo izpisal novo stanje obeh spremenljivk. Stanje »off« pri tem pomeni, da so vrata zaklenjena, stanje »on« pa da so vrata odklenjena.

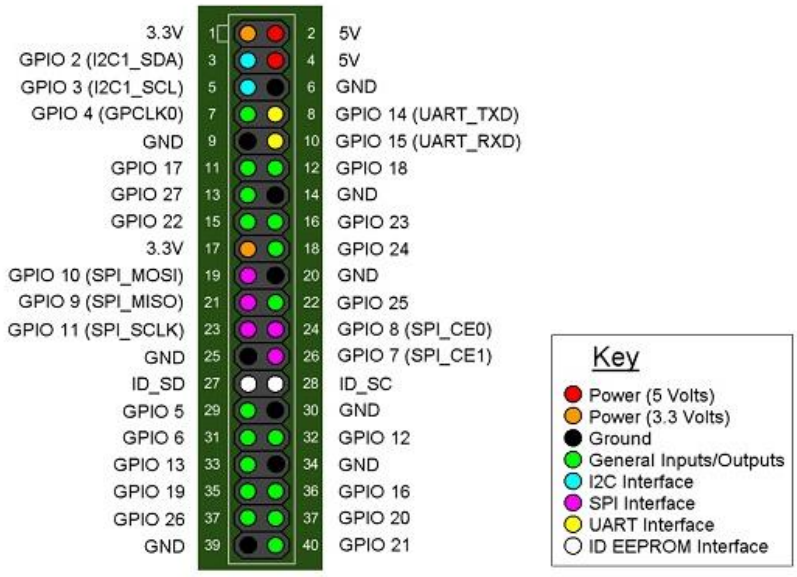
Povezava računalnika Raspberry PI z relejem in ključavnico

V najinem primeru sva za izhod za signal izbrala GPIO priključek 17 za izvajanje aktivnosti spremenljivke A – vrata 1 in priključek 23 za izvajanje aktivnosti spremenljivke B – vrata 2. Izhodi računalnika Raspberry PI pa niso dovolj močni, da bi lahko direktno upravljali s ključavnico oziroma zaklepom, zato moramo uporabiti dodatni vmesni člen – rele.

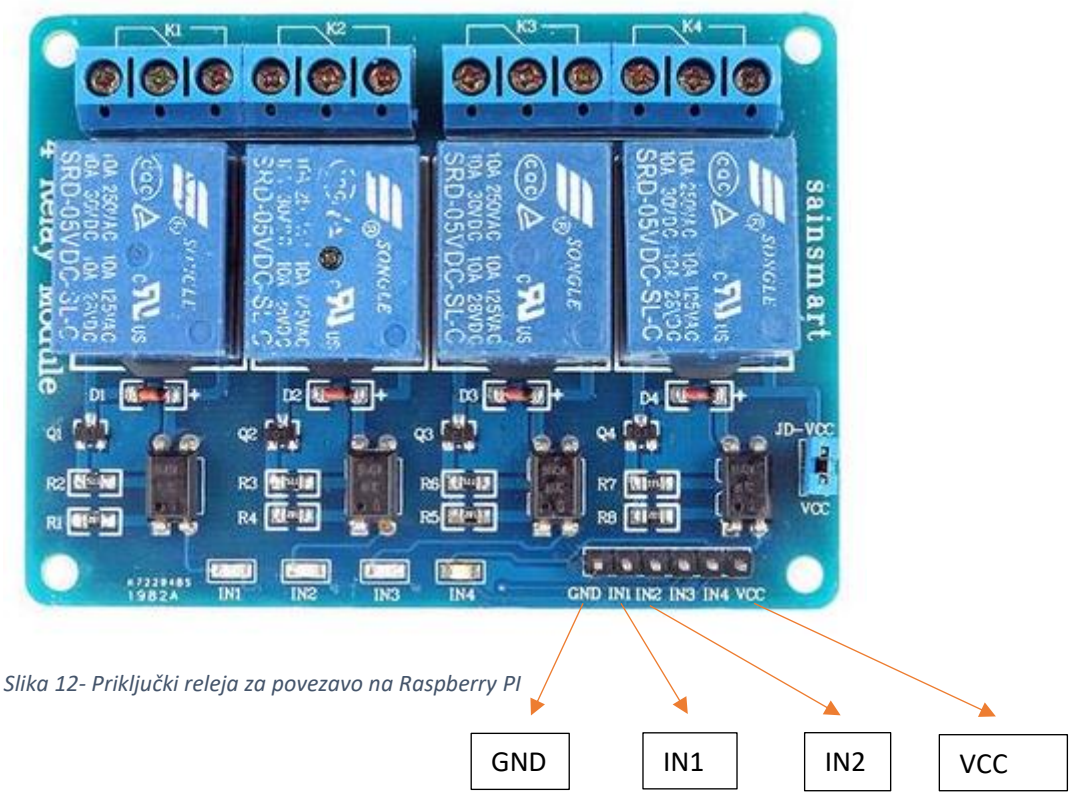
Povezovanje med računalnikom Raspberry PI in releji izvedemo s povezovalnimi kabli. Najprej priključimo napajanje za delovanje elektronike pri krmilju relejev. V ta namen povežemo priključka GND na Raspberry PI in GND na tiskanini releja ter priključka 5V na Raspberry PI in VCC na tiskanini releja. Za krmiljenje relejev sva izbrala priključka 17 in 23, ki sta označena z GPIO17 in GPIO23. Povežemo ju s priključkoma IN1 in IN2 na tiskanini releja. S tem sva zagotovila, da bodo krmilni signali vklapljali in izklapljali posamezen rele glede na zahtevo uporabnika.

VCC in GND priključka na tiskanini releja lahko povežemo na katerakoli priključka 5V in GND na računalniku Raspberry PI.

Krmiljenje relejev preko priključkov GPIO17 in GPIO 23 ni obvezno. Ta dva sva si izbrala sama. Namesto njiju lahko uporabimo katerikoli priključek računalnika Raspberry PI, ki predstavlja splošen izhod/vhod. Vtem primeru ne smemo pozabiti spremeniti tudi kode napisane v programu v Pythonu. Slednjo spremenimo enostavno, saj je potrebno zamenjati le številki priključkov 17 in 23 z drugimi izbranimi številkami izhodov.

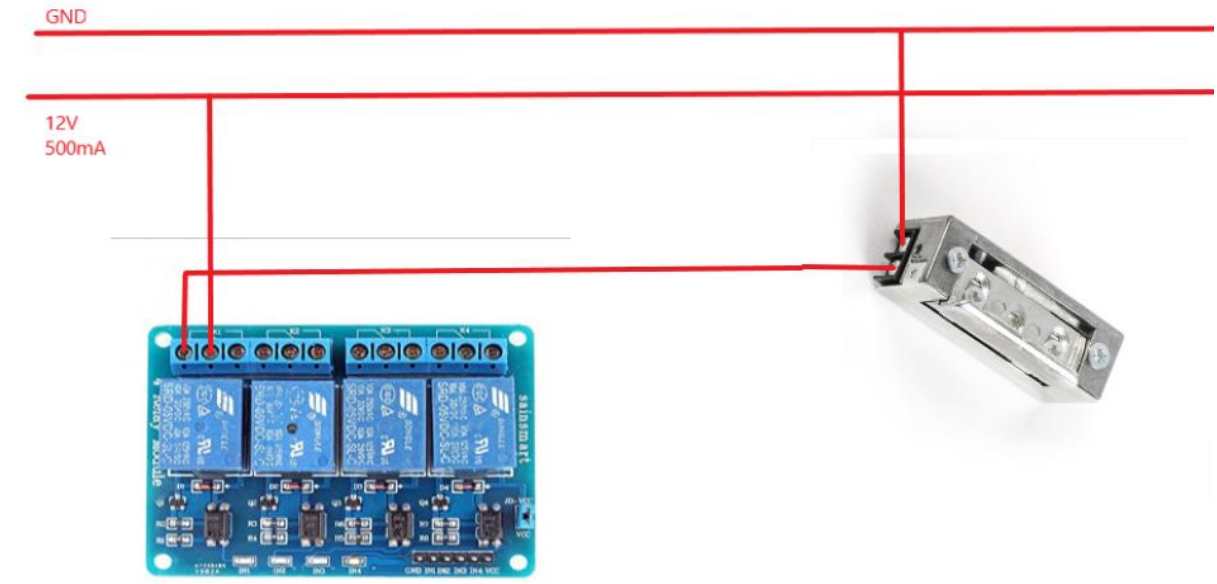


Slika 11 - Raspberry pi GPIO priključki



Slika 12- Priključki releja za povezavo na Raspberry PI

Spodnja slika prikazuje še zadnje povezave – priključitev zaklepa na rele. Napajanje 12V 500mA za zaklep dobimo iz posebnega napajalnika, ki je ločen od računalnika Raspberry PI. Zaklep je povezan na levi priključek releja, ki je ob izpadu napajanja normalno izključen. To pomeni, da je zaklep ob izpadu napajanja zaprt (vrat ni mogoče odpreti).



Slika 13 - Povezava Releja na ključavnico

Težave med delom

Raziskovalno delo, ki sva ga opravila, je slonelo predvsem na učenju o novega programskega jezika Python ter spoznavanju računalnika Raspberry PI in tehnologije NFC. O znanjih z omenjenih področij se v šoli nismo učili in jih zato nisva poznala, zato sva morala spoznati in se naučiti veliko novosti z prej omenjenih področij, da sva lahko izdelala in dokončala najino nalogo.

Med izdelavo Android aplikacije sva si pomagala z že obstoječimi zapisi kode, ki sva jih našla na spletu. Primer take kode je bil, kako povezati Android napravo z Mifare Classic 1k kartico in kako prebrati NDEF. Takih uporabnih kod je bilo na spletu relativno malo, zato sva morala večino kode spisati sama in pri tem veliko eksperimentirati, da sva ugotovila pravilne načine delovanja in nastavitve.

V zadnji fazi programiranja Android aplikacije se je pojavila težava z branjem NDEF iz kartice. Po večih urah testiranja in spreminjanja programa, sva ugotovila, da NDEF sploh ni bil pravilno zapisan. To sva ugotovila šele potem, ko sva zapisala NDEF z drugo aplikacijo (»NFC tools« iz trgovine Google).

Upravljanje z Raspberry PI je bilo sprva težavno, saj na začetku nisva vedela za Windows program Putty, s katerim se lahko povežeš z Raspberry PI preko omrežja in uporabljaš terminalski dostop. Pred tem sva uporabljala konzolo (monitor, miško in tipkovnico računalnika), ki sva jo preklapljala med namiznim računalnikom in računalnikom Raspberry PI.

Posebne težave so se pojavile pri povezovanju strojne opreme. Ker sva oba računalničarja, nisva imela znanja o povezovanju elektronskih komponent na fizičnem nivoju. Prvi problem je bil priklop releja, nato pa tudi zaklepa in njegovega napajanja, saj to ni najino področje.

Razprava

Najine hipoteze so bile:

1) Stroški projekta ne bodo višji od 100 EUR
2) Z aplikacijo na telefonu bo mogoče odklepati/zaklepati hišna vrata
3) Za realizacijo projekta lahko uporabimo računalnik Raspberry PI 3
4) Aplikacijo na telefonu bodo znali uporabljati »navadni« uporabniki
5) Z aplikacijo na telefonu bo mogoče odklepati/zaklepati več vrat

Stroški projekta ne bodo višji od 100 EUR: OVRŽENA

- Stroški projektne naloge so presegali 100eur.

Z aplikacijo na telefonu bo mogoče odklepati/zaklepati hišna vrata: POTRJENA

- Sistem je narejen, tako da preko telefona nadzorujemo zaklep vrat.

Za realizacijo projekta lahko uporabimo računalnik Raspberry PI 3: POTRJENA

- Zaklep je nadzorovan preko Raspberry PI 3

Aplikacijo na telefonu bodo znali uporabljati »navadni« uporabniki: POTRJENA

- Aplikacijo sva dala preskusiti staršem, ki so jo znali uporabiti.

Z aplikacijo na telefonu bo mogoče odklepati/zaklepati več vrat: POTRJENA

- Sistem je trenutno konfiguriran tako, da lahko odpiramo dvoje vrat

Izdelava raziskovalne naloge je zahtevala prebiranje velike količine dokumentacije in veliko ogledov različnih primerov programske kode na forumih. Sem spadajo tudi najrazličnejši vodiči, ki jih je moč najti na YouTube. Če bi imela več časa in več denarja, bi lahko sistem še veliko bolj razširila. Dodala bi lahko na primer nadzor nad lučmi, nadzor nad električnimi zavesami, lahko bi kupila tudi senzorje gibanja, ki bi lahko bili nameščeni v vsaki sobi ter bili povezani z računalnikom Raspberry PI. S temi dodatki bi lahko zaznali premikanje oseb v sobi, lahko bi prižgali luči, spustili žaluzija…

Zaključek

Tehnologija NFC se spreminja iz dneva v dan. Najpogosteje se omenja v povezavi s plačevanjem z mobilnimi napravami v trgovinah in različnih terminalih. Tudi pri nas uporaba te tehnologije ni novost, saj lahko Moneto že dolgo uporabljamo za plačevanje na različnih mestih. Uporaba računalnika Raspberry PI, podatkovne baze Firebase in aplikacije Android Studio nama je prikazala številne možnosti uporabe le-teh na različne načine, ki jih lahko uporabimo pri hišni avtomatizaciji kot npr.: nadzor luči s telefonom... Med iskanjem dokumentacije o priključkih na računalniku Raspberry PI sva našla tudi primere za priključitev senzorjev gibanja in prikaz podatkov na LED prikazovalniku, ki prikaže v kateri sobi v hiši senzor zaznava gibaje.

No, NFC obljublja, da bo celoten postopek plačevanja še bolj preprost, kar pomeni, da bo uporabnost NFC kartic postala še bolj enostavna in priljubljena.

Viri in literatura

Slike:

SAIN, Smart. 4 kanalni rele. [Citirano 4.3.2018]. Dostopno na spletnem naslovu: https://www.amazon.com/SainSmart-101-70-101-4-Channel-Relay-Module/dp/B0057OC5O8

Z-home. Električna ključavnica. [Citirano 4.3.2018]. Dostopno na spletnem naslovu: https://www.z-home.si/varnost/el.kljucavnice/el.kljucavnica-r4-ld

Kuman. Povezovalni kabli. [citirano 4.3.2018]. Dostopno na spletnem naslovu: https://www.amazon.de/dp/B01BV3Z342/ref=pe_3044161_185740101_TE_item

Mifare. Classic 1k kartice. [citirano 4.3.2018]. Dostopno na spletnem naslovu: https://www.mifare.net

Raspberry Pi. Računalnik Raspberry pi. [citirano 4.3.2018]. Dostopno na spletnem naslovu: < https://www.raspberrypi.org

Oneplus. Android telefon. [citirano 4.3.2018]. Dostopno na spletnem naslovu: < http://www.zdnet.com/product/oneplus-5t/

Dokumentacija:

Android, Dev. [s. d.]. NFC osnove. [citirano 4.3.2018]. Dostopno na spletnem naslovu: https://developer.android.com/guide/topics/connectivity/nfc/nfc.html

Android, Dev. [s. d.]. Near Field Communication. [citirano 4.3.2018]. Dostopno na spletnem naslovu: https://developer.android.com/guide/topics/connectivity/nfc/index.html

Android, Dev. [s. d.]. android.nfc. [citirano 4.3.2018]. Dostopno na spletnem naslovu:

https://developer.android.com/reference/android/nfc/package-summary.html

Vedat Coskun, Kerem Ok, Busra Ozdenizci. 3.4.2013. NFC Application Development for Android. John Wiley & Sons. [citirano 4.3.2018]. Dostopno na spletnem naslovu: https://books.google.si/books?id=c4QRU17e494C&pg=SA2-PA42&lpg=SA2-PA42&dq=nfc+documentation+android&source=bl&ots=VazU_O1S5J&sig=1J3qNZ9sttgMxG1OePSHtV1TzNQ&hl=sl&sa=X&ved=0ahUKEwjprsmMqtPZAhWHaFAKHQQ-BSAQ6AEIZzAH#v=onepage&q=nfc%20documentation%20android&f=false

Wondratschek, Ralf. 16.5.2013. Reading NFC Tags With Android. [citirano 4.3.2018]. Dostopno na spletnem naslovu: https://code.tutsplus.com/tutorials/reading-nfc-tags-with-android--mobile-17278

Clem57. 9.1.2015. Python code. [citirano 4.3.2018]. Dostopno na spletnem naslovu: https://www.element14.com/community/thread/40299/l/i-need-simple-python-code-to-run-my-pi-gpio-with-channgel-relay?displayFullThread=true

Freakish. 7.11.2013. Class threading. [citirano 4.3.2018]. Dostopne na spletnem naslovu: https://stackoverflow.com/questions/19846332/python-threading-inside-a-class

Ozgur Vatansever. [s. d.]. Python Firebase. [citirano 4.3.2018]. dostopno na spletnem naslovu: https://pypi.python.org/pypi/python-firebase/1.2

Jared Rummler. 14.12.2015. Device name and model. [citirano 9.3.2018]. Dostopno na spletnem naslovu: https://stackoverflow.com/questions/7071281/get-android-device-name

Alper. 22.5.2018. Substring from string. [citirano 9.3.2018]. Dostopno na spletnem naslovu: https://stackoverflow.com/questions/16702357/how-to-replace-a-substring-of-a-string

Zahvala

Za pomoč pri izvedbi raziskovalne naloge bi se zahvalila profesorju Borutu Slemenšku, ki naju je usmerjal pri izvedbi zaklepa z NFC, ter za vse ostale napotke glede raziskovalna naloge.

Seveda pa se morava zahvaliti tudi najinim staršem, ki so naju spodbujala in nama dvigala moralo pri izdelavi raziskovalne naloge. Ob njenem zaključku so si vzeli tudi čas in najin izdelek preizkusili. Hvala.

Priloge

Uporabniška aplikacija za Androidni telefon

```
package com.example.user.minfc;

import android.app.PendingIntent;
import android.content.Intent;
import android.content.IntentFilter;
import android.nfc.NdefMessage;
import android.nfc.NdefRecord;
import android.nfc.NfcAdapter;
import android.nfc.Tag;
import android.nfc.tech.Ndef;
import android.os.AsyncTask;
import android.os.Build;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

import com.firebase.client.ChildEventListener;
import com.firebase.client.DataSnapshot;
import com.firebase.client.Firebase;
import com.firebase.client.FirebaseError;
import com.firebase.client.ValueEventListener;

import java.io.UnsupportedEncodingException;
import java.text.SimpleDateFormat;
```

```
import java.util.Arrays;
import java.util.Date;
import java.util.Locale;

public class MainActivity extends AppCompatActivity {

   public static final String MIME_TEXT_PLAIN = "text/plain";
   public static final String TAG = "NfcDemo";
   Firebase fire_A, fire_B, fire_log, fire_i;

   private TextView textLog = null;
   private NfcAdapter mNfcAdapter = null;
   private Boolean lastStateA, lastStateB;

   private ToggleButton toggleButtonA, toggleButtonB;

   int logIndex = 0;

   String currentDate = new SimpleDateFormat("yyyy-MM-dd", Locale.getDefault()).format(new
Date());


   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      fire_B.setAndroidContext(this);
      fire_A.setAndroidContext(this);

      fire_A = new Firebase("https://raziskovalna-ea3bd.firebaseio.com/outputs/A");
      fire_B = new Firebase("https://raziskovalna-ea3bd.firebaseio.com/outputs/B");
```

```
fire_log = new Firebase("https://raziskovalna-ea3bd.firebaseio.com/log/");
fire_i = new Firebase("https://raziskovalna-ea3bd.firebaseio.com/logIndex");
fire_B.setValue(false);
fire_A.setValue(false);


textLog = findViewById(R.id.msgLog);

toggleButtonA =  findViewById(R.id.TB_A);
toggleButtonB = findViewById(R.id.TB_B);

mNfcAdapter = NfcAdapter.getDefaultAdapter(this);

toggleButtonA.setEnabled(false);
toggleButtonA.setTextOff("Zaprta");
toggleButtonA.setTextOn("Odprta");

toggleButtonB.setEnabled(false);
toggleButtonB.setTextOff("Zaprta");
toggleButtonB.setTextOn("Odprta");


fire_i.addValueEventListener(new ValueEventListener() {
   @Override
   public void onDataChange(DataSnapshot dataSnapshot) {
      logIndex = Integer.parseInt(dataSnapshot.getValue().toString());
   }

   @Override
   public void onCancelled(FirebaseError firebaseError) {
```

```
    }
});

fire_log.child(currentDate).addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        String asd = dataSnapshot.getValue().toString();
        String asd2 = asd.replaceAll("=true", ": odprl vrata");
        String asd3 = asd2.replaceAll("=false", ": zaprl vrata");
        String asd4 = asd3.replace("{","");
        String asd5 = asd4.replace("}", "");
        textLog.setText(asd5);
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {
        String asd = dataSnapshot.getValue().toString();
        String asd2 = asd.replaceAll("=true", ": odprl vrata");
        String asd3 = asd2.replaceAll("=false", ": zaprl vrata");
        String asd4 = asd3.replace("{","");
        String asd5 = asd4.replace("}", "");
        textLog.setText(asd5);
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {

    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {
```

```
        }

        @Override
        public void onCancelled(FirebaseError firebaseError) {

        }
    });

    fire_A.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            lastStateA = (Boolean) dataSnapshot.getValue();
            toggleButtonA.setChecked(lastStateA);
            //Toast.makeText(MainActivity.this, dataSnapshot.getValue().toString() + " A",
Toast.LENGTH_LONG).show();
        }

        @Override
        public void onCancelled(FirebaseError firebaseError) {

        }
    });

    fire_B.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            lastStateB = (Boolean) dataSnapshot.getValue();
            toggleButtonB.setChecked(lastStateB);
            //Toast.makeText(MainActivity.this, dataSnapshot.getValue().toString() + " B",
Toast.LENGTH_LONG).show();
        }
```

```
        @Override
        public void onCancelled(FirebaseError firebaseError) {

        }
    });

    if (mNfcAdapter == null) {
        Toast.makeText(this, "telefon ne podpira NFC.", Toast.LENGTH_LONG).show();
        finish();
        return;
    }

    if (!mNfcAdapter.isEnabled()) {
        textLog.setText("NFC deactivated");
    } else {
        textLog.setText("NFC activated");
    }

    handleIntent(getIntent());
}


@Override
protected void onResume() {
    super.onResume();

    setupForegroundDispatch(this, mNfcAdapter);
}

@Override
```

```
protected void onPause() {

    stopForegroundDispatch(this, mNfcAdapter);

    super.onPause();
}



@Override
protected void onNewIntent(Intent intent) {

    handleIntent(intent);
}

private void handleIntent(Intent intent) {
    String action = intent.getAction();
    if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)) {

        String type = intent.getType();
        if (MIME_TEXT_PLAIN.equals(type)) {

            Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
            new NdefReaderTask().execute(tag);

        } else {
            Log.d(TAG, "Wrong mime type: " + type);
        }
    } else if (NfcAdapter.ACTION_TECH_DISCOVERED.equals(action)) {

        Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
        String[] techList = tag.getTechList();
```

```
        String searchedTech = Ndef.class.getName();

        for (String tech : techList) {
            if (searchedTech.equals(tech)) {
                new NdefReaderTask().execute(tag);
                break;
            }
        }
    }

}

public static void setupForegroundDispatch(final MainActivity activity, NfcAdapter adapter) {
    final Intent intent = new Intent(activity.getApplicationContext(), activity.getClass());
    intent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);

    final PendingIntent pendingIntent = PendingIntent.getActivity(activity.getApplicationContext(),
0, intent, 0);

    IntentFilter[] filters = new IntentFilter[1];
    String[][] techList = new String[][]{};

    filters[0] = new IntentFilter();
    filters[0].addAction(NfcAdapter.ACTION_NDEF_DISCOVERED);
    filters[0].addCategory(Intent.CATEGORY_DEFAULT);
    try {
        filters[0].addDataType(MIME_TEXT_PLAIN);
    } catch (IntentFilter.MalformedMimeTypeException e) {
        throw new RuntimeException("Check your mime type.");
    }
```

```
        adapter.enableForegroundDispatch(activity, pendingIntent, filters, techList);
    }

    public static void stopForegroundDispatch(final MainActivity activity, NfcAdapter adapter) {
        adapter.disableForegroundDispatch(activity);
    }

    private class NdefReaderTask extends AsyncTask<Tag, Void, String> {

        @Override
        protected String doInBackground(Tag... params) {
            Tag tag = params[0];

            Ndef ndef = Ndef.get(tag);
            if (ndef == null) {
                // NDEF ne podpira tega Taga.
                return null;
            }

            NdefMessage ndefMessage = ndef.getCachedNdefMessage();

            NdefRecord[] records = ndefMessage.getRecords();
            for (NdefRecord ndefRecord : records) {
                if (ndefRecord.getTnf() == NdefRecord.TNF_WELL_KNOWN &&
Arrays.equals(ndefRecord.getType(), NdefRecord.RTD_TEXT)) {
                    try {
                        return readText(ndefRecord);
                    } catch (UnsupportedEncodingException e) {
                        Log.e(TAG, "Unsupported Encoding", e);
                    }
                }
```

```
        }

        return null;
    }

    private String readText(NdefRecord record) throws UnsupportedEncodingException {

        byte[] payload = record.getPayload();

        String textEncoding = ((payload[0] & 128) == 0) ? "UTF-8" : "UTF-16";

        int languageCodeLength = payload[0] & 0063;

        return new String(payload, languageCodeLength + 1, payload.length - languageCodeLength - 1,
textEncoding);
    }

    @Override
    protected void onPostExecute(String result) {
        if (result != null) {
            //textLog.setText("Message NDEF: " + result);
            logIndex ++;

            if (result.equals("A")){
                Log.d("A: ", lastStateA.toString());
                fire_A.setValue(!lastStateA);
```

```
fire_log.child(currentDate).child(Integer.toString(logIndex)).child(getDeviceName()).setValue(!lastStateA);
                Toast.makeText(MainActivity.this, "Vrata 1", Toast.LENGTH_LONG).show();
            }

            if(result.equals("B")){
                Log.d("B: ", lastStateB.toString());
                fire_B.setValue(!lastStateB);
                Toast.makeText(MainActivity.this, "Vrata 2", Toast.LENGTH_LONG).show();

fire_log.child(currentDate).child(Integer.toString(logIndex)).child(getDeviceName()).setValue(!lastStateB);
            }
            fire_i.setValue(logIndex);

            }
        }


    }

    public String getDeviceName() {
        String manufacturer = Build.MANUFACTURER;
        String model = Build.MODEL;
        if (model.startsWith(manufacturer)) {
            return capitalize(model);
        } else {
            return capitalize(manufacturer) + " " + model;
        }
    }
```

```
    private String capitalize(String s) {
        if (s == null || s.length() == 0) {
            return "";
        }
        char first = s.charAt(0);
        if (Character.isUpperCase(first)) {
            return s;
        } else {
            return Character.toUpperCase(first) + s.substring(1);
        }
    }
}
```