Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tim Knez

Domači alarmni sistem

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Branko Šter

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Na razvojni plošči STM32F4 Discovery, z jedrom ARM Cortex-M4, načrtujte pametni, domači alarmni sistem. Sistemu dodajte infrardeči senzor za zaznavo gibanja, magnetni senzor, senzor za zaznavo plinov in GSM modul za pošiljanje SMS sporočil. Sistem povežite z oddaljenim spletnim strežnikom implementiranim na mikroračunalniku Raspberry PI 3. Na spletnem strežniku naj teče operacijski sistem Raspbian, ki naj služi vpogledu stanja in spreminjanju nastavitev celotnega sistema.

Predvsem bi se rad zahvalil prof. dr. Branku Šteru, za strokovno pomoč in vodenje pri izdelavi diplomske naloge. Prav tako bi se zahvalil družini, prijateljem ter vsem, ki ste mi kakorkoli pomagali tekom študija in mi stali ob strani.

Kazalo

Kazalo slik

Seznam uporabljenih kratic

kratica	angleško	slovensko
ARM	Acorn RISC Machine	Tip RISC procesorjev
FPU	Floating Point Unit	Enota za plavajočo vejico
RAM	Random Access Memory	Bralno-pisalni pomnilnik
SWIM	Single Wire Interface Module	Modul za razhroščevanje
JTAG	Joint Test Action Group	Vmesnik namenjen razhroščevanju in testiranju elektronskih vezij
SWD	Serial Wire Debugging	Razhroščevanje preko serijske povezave
USB	Universal Serial Bus	Univerzalno serijsko vodilo
COM	Communication port	Priključek za serijsko komunikacijo
I^2C	Inter-Integrated Circuit	Serijska povezava v integriranih vezjih
SPI	Serial Peripheral Interface bus	Tip serijske povezave
DAC	Digital-to-Analog Converter	Digitalno/analogni pretvornik
OTG	On-The-Go	Specifikacija, ki omogoča USB napravam igrati vlogo gostitelja
LED	Light-Emitting Diode	Svetleča dioda
PIR	Passive Infrared Sensor	Pasivni infrardeči senzor
LPG	Liquefied Petroleum Gas	Utekočinjeni naftni plin
GSM	Global System for Mobile Communications	Globalni sistem za mobilno komunikacijo
GPRS	General Packet Radio Service	Splošna paketna radijska postaja
SMS	Short Message Service	Sistem kratkih sporočil
UART	Universal Asynchronous Receiver-Transmitter	Vezje za serijsko komunikacijo

Tx	transmit	pošiljanje
Rx	receive	sprejemanje
SD	Secure Digital	Tip zunanjega pomnilnika
GPIO	General Purpose Input Output	Splošnonamenski vhod/izhod
IDE	Integrated Development Environment	Integrirano razvojno okolje
SQL	Structured Query Language	Strukturiran povpraševalni jezik

Povzetek

Naslov: Domači alarmni sistem

Avtor: Tim Knez

V okviru diplomske naloge sem izdelal domači alarmni sistem. Alarmni sistem je skupek komponent, ki skrbijo za nadzor varovanega objekta in pristojne obveščajo o morebitnih nenavadnih dogodkih. Sestavljen je iz nadzorne enote, kopice senzorjev in oddaljenega strežnika. Nadzorna enota s pomočjo senzorjev, ki konstantno pridobivajo podatke iz okolja, spremlja dogajanje v prostoru. V primeru odstopanja podatkov od optimalnih vrednosti to sporoči tudi pristojnim osebam. Na oddaljenem strežniku teče spletna aplikacija, preko katere je možno spreminjati nastavitve sistema.

Diplomska naloga opisuje zgradbo celotnega sistema, torej nadzorno enoto sistema, infrardeči senzor gibanja, magnetni senzor, senzor plinov, oddaljeni strežnik in GSM modul. Nato sledi inicializacija senzorjev, njihovi prekinitveni programi ter serijska komunikacija med mikrokrmilnikoma. Na koncu pa si pogledamo še operacijski sistem Raspbian in spletno aplikacijo, preko katere je moč nadzorovati stanje sistema in spreminjati njegove nastavitve. Končni izdelek je povsem delujoč, dovolj učinkovit in cenovno ugoden alarmni sistem za nadzor doma.

Ključne besede: STM32F4, Raspberry PI 3, senzorji, domači alarmni sistem, spletni strežnik.

Abstract

Title: Home alarm system

Author: Tim Knez

The aim of my thesis was to build a home alarm system. An alarm system is a set of components that controls a specific protected object and informs competent authorities about unusual events. It consists of a control unit, a stack of sensors and a remote server. The control unit uses the sensors that constantly receive data from the environment, to monitor the assigned space. Should the data deviate from optimal values, a signal is sent to authorities. The remote server runs a web application, where the settings of the system can be monitored.

The thesis describes the structure of the whole system – from the control unit of the system to infrared motion sensor, magnetic sensor, gas sensor, remote server and GSM module. After this it deals with the initialisation of the sensors, their interrupt routines and serial communication with microcontrollers. Finally, it discusses the Raspbian operating system and a web application, where the state of the system can be monitored and its settings adjusted. The final product is a fully functioning, efficient and affordable alarm system for home control.

Keywords: STM32F4, Raspberry PI 3, sensors, home alarm system, web server.

Poglavje 1

Uvod

V današnjem času so vlomi, tatvine in ropi vse bolj pogosti, v nekaterih mestih celo vsakdanji. Časi so se spremenili in ni več tako kot včasih, ko so se ljudje, predvsem na vasi, med seboj dobro poznali in svojih hiš praktično niso zaklepali. Danes si tega žal ne moremo privoščiti. Eden izmed razlogov za skrb je prav gotovo tudi vse večje število turistov po celem svetu. Da bi ljudje obvarovali svojo lastnino, se pred nepridipravi zavarujejo na različne načine. Kupijo si močna, precej draga, protivlomna vrata ter okna, namestijo kamere, imajo pse čuvaje ali pa si enostavno kupijo alarme oz alarmne sisteme. Taki sistemi, sploh kvalitetni, znajo biti precej dragi in nerodni za vgradnjo. Prav zato bi bilo dobro imeti alarmni sistem, ki bi bil dovolj učinkovit za uporabo, obenem pa cenovno ugoden, da bi si ga lahko privoščil vsakdo.

Alarmni sistemi so (pametni) sistemi za nadzor objektov oziroma prostorov, ki zaznajo kakršnokoli nenavadno stanje ali dogajanje in o tem takoj obvestijo eno ali več oseb. Nekateri tudi oddajajo izredno glasne opozorilne signale in opozorijo varnostne centre oziroma pristojne osebe.

Ponavadi so sestavljeni iz nadzorne enote, oddaljenega strežnika in večjega števila senzorjev, ki zaznavajo spremembe v okolju. Nadzorna enota skrbi za nadzor vseh senzorjev in ob morebitnem prevelikem odstopanju od normalnih vrednosti senzorja sproži alarm. Oddaljeni strežnik je zadolžen za shrambo podatkov, preverja stanje nadzorne enote in senzorjev ter omogoča vpogled

oziroma spreminjanje nastavitev celotnega sistema. Priporočljivo je, da je strežnik dostopen preko manjšega lokalnega omrežja, saj je s tem zmanjšano tveganje zlorab in vdorov v sistem.

Diplomska naloga v celoti opisuje strukturo ter izdelavo domačega alarmnega sistema na razvojni plošči STM32F4 Discovery in mikroračunalniku Raspberry PI 3. Cilj je bil ustvariti cenovno ugoden, a dovolj učinkovit sistem, ki bi brez težav nadzoroval stanje objekta. Sistem zna detektirati gibanje v prostoru, povečano količino strupenih plinov, odprta vrata in okna. Preko spletne aplikacije je omogočeno spreminjati nastavitve celotnega sistema. O morebitnem nedovoljenem vdoru v prostor, oziroma uhajanju nezaželenega plina, sistem preko SMS sporočila to takoj sporoči izbranemu številu oseb. Omogoča ugašanje ter prižiganje samega sistema in podpira večje število uporabnikov. Stanje celotnega sistema lahko spremljamo preko računalnika ali prenosne naprave, ki je povezana v lokano omrežje.

Najprej si bomo ogledali strojno opremo sistema in njene pomembnejše specifikacije. Torej samo ploščo STM32F4 Discovery, ki služi kot nadzorna enota sistema. Nato sledijo infrardeči senzor za zaznavo gibanja HC-SR501, magnetni senzor MC-38, senzor za zaznavo plinov ZYMQ-2 in GPRS/GSM modul A6 za pošiljanje SMS sporočil.

V drugem delu naloge si bomo pogledali inicializacijo GPIO naprav, implementacijo senzorjev in njihove prekinitvene funcije.

Za konec nam ostane še oddaljeni strežnik, realiziran na mikroračunalniku Raspberry PI 3. Pogledali si bomo, kako dostopamo do SQL podatkovne baze, serijsko komunikacijo med obema mikrokrmilnikoma ter spletno aplikacijo, napisano v jeziku C. Preko omenjene aplikacije je možno spremljati stanje sistema ter spreminjati nastavitve.

Poglavje 2

Strojna oprema

2.1 Plošča STM32F4 Discovery

Za nadzorno enoto alarmnega sistema sem si izbral razvojno ploščo STM32F4 Discovery (Slika 2.1) proizvajalca mikrokrmilnikov STMicroelectronics. Poganja jo visoko zmogljiv 32-bitni ARM (ang. *Acorn RISC Machine* oz. *Advanced RISC Machine*) procesor Cortex-M4 s FPU (ang. *Floating Point Unit*) enoto, 1 MB bliskovnega (Flash) pomnilnika ter 192 KB RAM-a (ang. *Random Access Memory*). Maksimalna frekvenca delovanja znaša 168 MHz.

Za lažji razvoj uporabniških aplikacij in razhroščevanje programske kode plošča vsebuje vgrajen razhroščevalnik ter programer ST-LINK/V2. Komunikacija med razhroščevalnikom in razvojnim okoljem poteka preko modula SWIM (ang. *Single Wire Interface Module*) ter JTAG/SWD (ang. *Joint Test Action Group/Serial Wire Debugging*) vmesnika. Ploščo na računalnik fizično priključimo preko USB 2.0 (ang. *Universal Serial Bus*) vmesnika z USB kablom. Kabel ima standardni A priključek na eni strani ter mini-B priključek na drugi strani. Vmesnik USB ST-LINK lahko poleg razhroščevalnih vrat uporabljamo še kot virtualna COM (ang. *Communication port*) vrata ali vrata za masovno shrambo. Ploščo je možno napajati preko vodila USB ali preko zunanjega 5 voltnega napajanja [16].

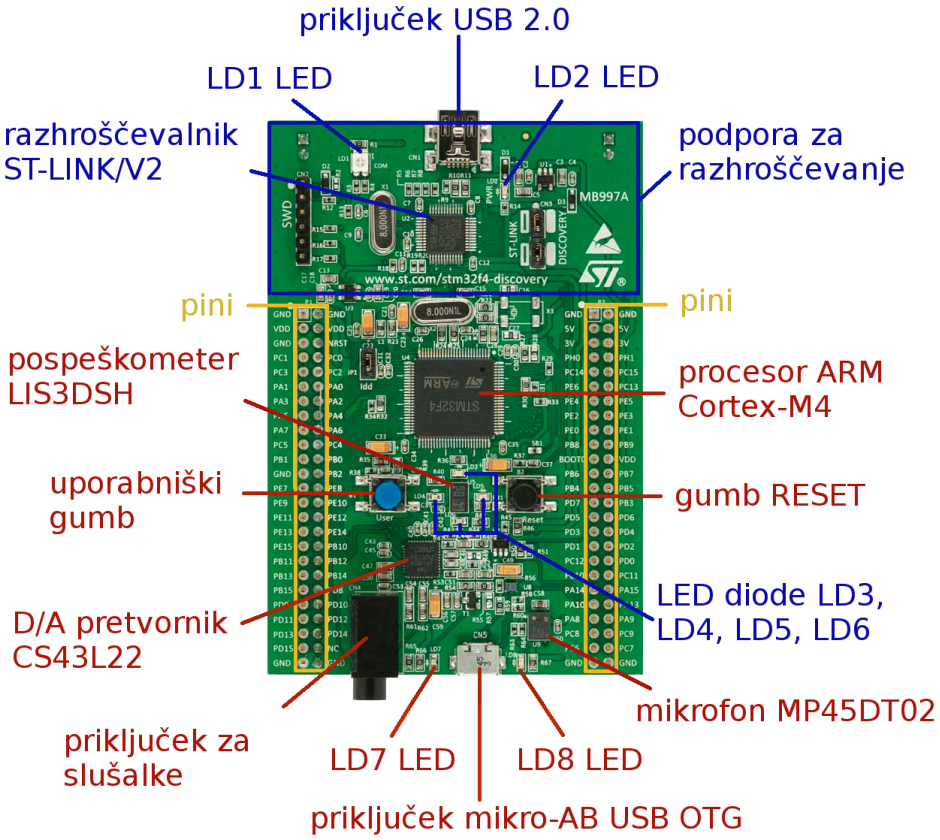
Za merjenje nagiba, premika in pospeška lahko uporabimo vgrajeni 3 osni

linearni merilnik pospeška LIS302DL. LIS302DL poleg merilnega elementa, ki skrbi za zajem podatkov iz okolja, sestavlja še IC vmesnik. Vmesnik preko I^2C (ang. *Inter-Integrated Circuit*) ali serijskega vmesnika SPI (ang. *Serial Peripheral Interface bus*) omogoča komunikacijo z zunanjim svetom. Merilnik je sposoben podatke navzven pošiljati z izhodno frekvenco 100 ali 400 Hz [11].

Z vgrajenim digitalnim mikrofonom MP45DT02 ST-MEMS je moč snemati zvočne posnetke [1]. Preko analognega priključka za slušalke pa jih lahko predvajamo zunanjemu svetu. Za pretvorbo digitalnega signala v analognega poskrbi avdio digitalno-analogni (D/A) pretvornik DAC (ang. *Digital-to-Analog Converter*) CS43L22 [2] z vključenim gonilnikom razreda D. Prav tako kot pospeškometer, tudi mikrofon poleg merilnega elementa sestavlja še stereo IC vmesnik. Zraven priključka za avdio izhod najdemo tudi priključek mikro-AB USB OTG (ang. *On-The-Go*). Nanj lahko priključimo zunanje naprave, na primer računalniške miške, tipkovnice, digitalne kamere ali zunanje pomnilnike.

Plošča vsebuje tudi 2 gumba. Moder gumb je namenjen uporabniku, ki ga svojim potrebam ustrezno sprogramira. S pritiskom na črn gumb pa ponovno zaženemo celotno ploščo.

Na plošči se nahaja 8 LED (ang. *Light-Emitting Diode*) diodic za različne namene. Rdečo-zelena dioda LD1, zraven USB A priključka, signalizira USB komunikacijo z računalnikom. Na drugi strani istega priključka pa je dioda LD2 rdeče barve. Ta nam sporoča, ali je plošča priključena v električno omrežje in s tem tudi prižgana. Povsem na drugem delu plošče, poleg mikro USB priključka, sta nameščeni še dve sistemski diodi. Zelena LD7 signalizira stanje vodila VBUS, rdeča LD8 pa sveti, kadar skozi vmesnik teče odvečni tok (ang. *overcurrent*). Med obema gumboma so postavljene 4 uporabniške diodice. LD3 je oranžne, LD4 zelene, LD5 rdeče in LD6 modre barve. Podobno kot gumb, tudi te ledice uporabnik lahko sprogramira po svojih potrebah in željah [17].



Slika 2.1: Plošča SMT32F4 Discovery, vir: [17]

2.2 PIR senzor za zaznavo gibanja HC-SR501

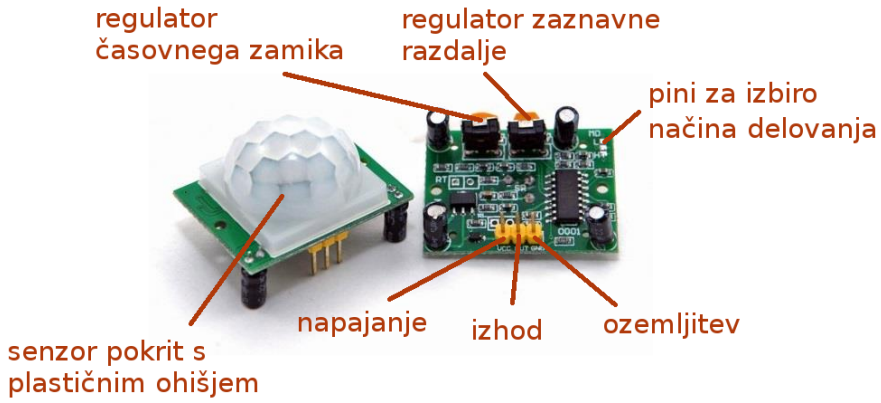
Vsa telesa s temperaturo nad absolutno ničlo oddajajo toploto v obliki sevanja. Omenjena toplota ni vidna človeškemu očesu. Jo pa precej dobro zaznavajo elektronski infrardeči PIR (ang. *Passive Infrared Sensor*) senzorji kot je HC-SR501 (Slika 2.2) [6]. Senzor torej v svojem vidnem polju s pomočjo infrardeče svetlobe zaznava gibanje. Vidno polje senzorja je stožčaste oblike, s kotom 100 stopinj. Glavna procesna enota senzorja je BISS0001.

Za delovanje potrebuje električno energijo ter delovno napetost v intervalu od 4,5 do 20 V. Temperatura, pri kateri senzor še optimalno deluje, je med

-15 ter +70 °C.

Senzor je standarne velikosti 32 x 24 mm, obdaja pa ga plastično ohišje. Z regulatorjem časovnega zamika je možno nadzorovati odzivni čas, z regulatorjem zaznavne razdalje pa razdaljo, do katere senzor še zaznava gibanje. Odzivni čas lahko nastavimo od 5 do 200 ms, razdalja zaznave pa se giblje na intervalu od 3 do 7 m [18].

Način delovanja je binarni, torej oddaja samo dve logični vrednosti. Kadar zaznava gibanje, oddaja napetost 3,3 V, torej visoko logično vrednost ali "1", v nasprotnem primeru pa 0 V, torej nizko logično vrednot ali "0". Ob zaznavi gibanja je senzor približno 2,5 s nedelujoč. Temu času pravimo blokirni čas (ang. *blocking time*).



Slika 2.2: PIR senzor za zaznavo gibanja HC-SR501, vir: [23]

Senzor podpira dva načina delovanja, ki ju je možno nastaviti s pomočjo mostičev. Ponovljivi način (ang. *repeatable trigger*) uporabimo takrat, kadar želimo, da vrednost ostane visoka, če senzor zaznava gibanje dalj časa. Neponovljivi način (ang. *unrepeatable trigger*) pa uporabimo takrat, kadar želimo, da se vrednost ob prav tako dalj časa trajajočem gibanju ponastavi (resetira).

Na spodnjem delu elementa najdemo tri pine, preko katerih senzor pri-

ključimo na zunanjo komponento. S prvim pinom senzor ozemljimo, srednji pin služi kot izhod, preko katerega signaliziramo aktivnost, tretji pin pa je namenjen napajanju.

2.3 Magnetni senzor MC-38

Magnetni senzor je sestavljen iz dveh ločenih delov, od katerih je eden povezan z mikrokrmilnikom. Ponavadi ga uporabljamo, da izvemo, ali so okna (oziroma vrata) odprta ali zaprta. En del senzorja z vijaki pritrdimo na okvir okna (oziroma vrat), drugi del pa na okensko (oz. vratno) krilo. Pritrditi ju moramo tako, da ko je okno zaprto, sta dela senzorja sklenjena. Sklenjen senzor ustvarja magnetno polje. Ko senzor ločimo, magnetno polje izgine, senzor pa signalizira spremembo [7].

Slika 2.3: Magnetni senzor MC-38, vir: [22]

Magnetni senzor MC-38 (Slika 2.3) je bele barve oblečen v plastično ohišje. Velik je 27 x 14 x 7,6 mm. Da senzor lahko ustvarja magnetno polje, mora biti razdalja med deloma senzorja manjša od 18 mm [8].

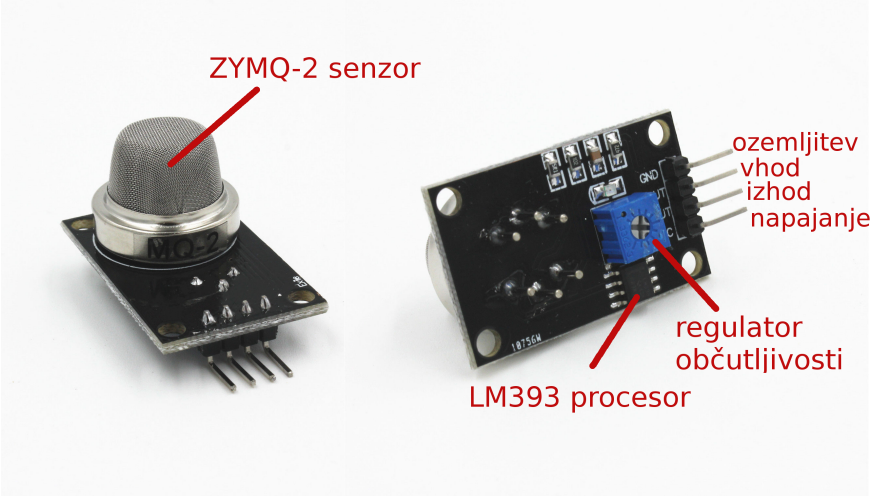
Maksimalna napetost, ki jo senzor lahko prejema, je 12 V. Za delovanje senzor potrebuje 300 mA električnega toka. Na mikrokrmilnik je priključen preko dveh žic.

Življenska doba senzorja je približno 1 milijon sklenitev. Namestitev je izredno enostavna.

2.4 Senzor za zaznavo plinov ZYMQ-2

Senzor plina je naprava, ki zaznava prisotnost dražljivih, vnetljivih ali strupenih plinov v prostoru. Senzor plina je idealna izbira za domači alarmni sistem, saj ljudem in živalim škodljive pline najdemo tudi v vsakdanjih predmetih in napravah v našem domu [19].

Senzor za zaznavo plinov ZYMQ-2 (Slika 2.4) poganja procesna enota LM393. Velik je 32 x 22 x 27 mm, sam senzor pa je obdan s kovinskim mrežičastim ohišjem. Deluje z delovno napetosjo 5 V, vsebuje pa tudi regulator občutljivosti. Na mikrokrmilnik ga priključimo preko treh pinov, ki služijo kot ozemljitev, napajanje ter izhodna vrednost [20].



Slika 2.4: Senzor za zaznavo plinov ZYMQ-2, vir: [24]

Senzor je občutljiv na različne vrste plinov, kot so LPG (ang. *Liquefied*

Petroleum Gas oz *Liquid Petroleum Gas*), butan, metan, več vrst alkoholov, propan, vodik, dim ter nekateri drugi.

2.5 GPRS/GSM moodul A6

Za komunikacijo z modulom A6 (Slika 2.5) preko GSM/GPRS (ang. *Global System for Mobile Communications* originalno *Groupe Spécial Mobile/General Packet Radio Service*) omrežja lahko izbiramo med 4 frekvenčnimi pasovi po celem svetu. Podpira 850, 900, 1800 ali 1900 MHz frekvenčni pas [5].

Napaja se z delovno napetostjo od 3,3 do 4,3 V, za stanje v pripravljenosti pa potrebuje približno 3 mA električnega toka. Velik je 22,8 x 16,8 x 2,5 mm. Optimalna temperatura za delovanje senzorja mora biti v intervalu med -30 in +80 °C.

Slika 2.5: GSM/GPRS modul A6, vir: [21]

Modul podpira glasovne klice, SMS (ang. *Short Message Service*) sporočila in prenos GPRS podatkov. Maksimalna hitrost prenosa znaša 85.6 Kb/s navzdol ter 42.8 Kb/s navzgor. Vgrajeno ima tudi podporo za GSM07.07, 07.05

AT ukaze, razširjene Ai-Thinker ukaze, TCP/IP ukazni vmesnik ter digitalno in analogno HR, FR, EFR, AMR kodiranje.

Na plošči lahko najdemo še priključek mikro USB, gumb za vklop, anteno in 24 pinov. Senzor na zunanji mikrokrmilnik priključimo preko UART (ang. *Universal Asynchronous Receiver-Transmitter*) pinov Tx (ang. *transmit*) in Rx (ang. *receive*) in ga ozemljimo. Napaja se preko vmesnika mikro USB.

2.6 Raspberry PI 3

Raspberry PI 3 (Slika 2.6) je izredno zmogljiv mikroračunalnik v velikosti kreditne kartice [13]. Poganja ga 1.2 GHz procesor ARM-Cortex-A53. Pohvali se lahko z 1GB LPDDR2 RAM-a ter grafično procesno enoto Broadcom VideoCore IV. Podpira klasičen Bluetooth 4.1, 10/100 žično Ethernet povezavo ter 2.4 GHz 802.11n brezžično povezljivost. Podatke lahko shranjujemo na micro SD (ang. *Secure Digital*) zunanjo kartico. Na plošči lahko najdemo tudi 4 USB 2.0 konektorje, 3.5 milimeterski analogni avdio-video konektor, HDMI konektor, SCI in DSI priključka in 40 pinov za GPIO (ang. *General-Purpose Input/Output*) naprave [9].

Razvija ter proizvaja ga dobrodelna fundacija Raspberry PI iz Velike Britanije. Prvotni namen serije mikroračunalnikov Raspberry PI je bil poučevanje osnovnega programiranja v šolah manj razvitih držav. Zaradi dobrih specifikacij, majhnega dizajna, dokaj nizke cene ter zelo velike skupnosti neodvisnih razvijalcev je njegova popularnost zelo visoka, in še narašča [14].

Slika 2.6: Mikroračunalnik Raspberry PI 3, vir: [9]

Poglavje 3

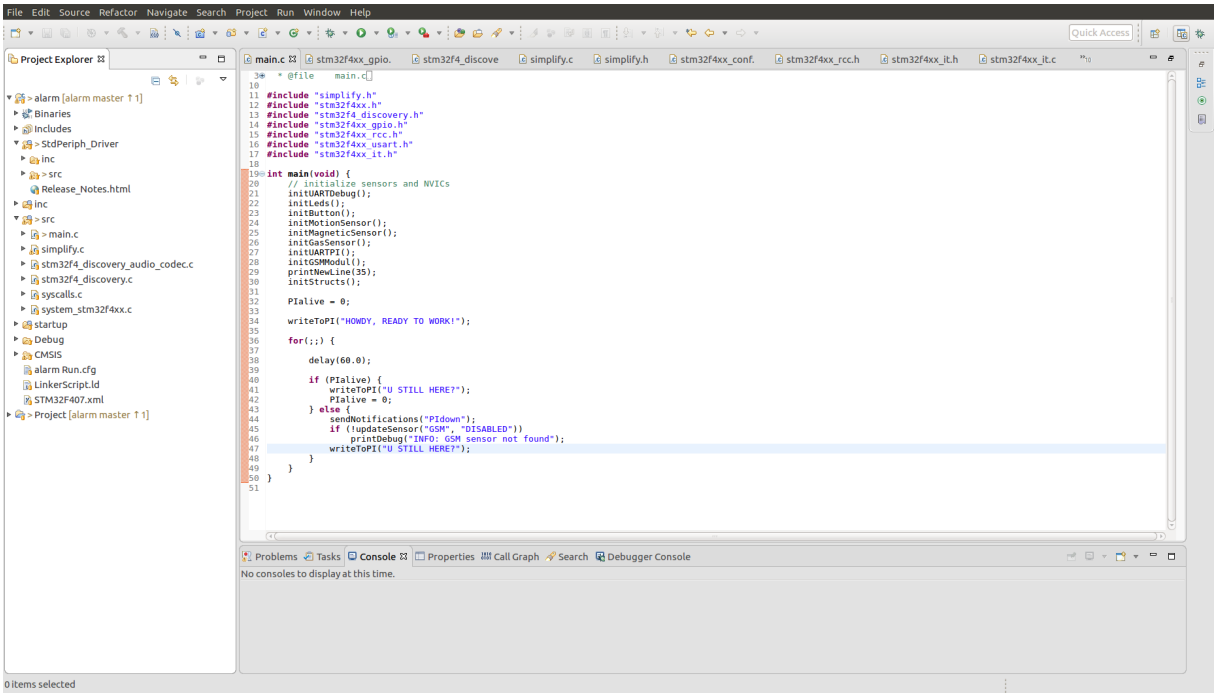
Programska oprema

3.1 System Workbench for STM32

Za programiranje plošče STM32F4 Discovery sem uporabil orodje System Workbench for STM32 v razvojnem okolju Eclipse IDE (ang. *Integrated Development Environment*) (Slika 3.1). Orodje System Workbench ima bogato knjižnico za delo z STM32 ploščami v programskem jeziku C/C++. Med drugimi ima predpripravljene funkcije za delo z GPIO napravami, USART vmesniki, sistemsko uro, zunanjimi prekinitvami ter prekinitvenimi vektorji. Prav tako omogoča razhroščevanje kode v realnem času. Gcc prevajalnik poskrbi za prevajanje programske kode. Eclipse in System Workbench for STM32 sta obe prosto dostopni, odprtokodni, programski orodji dobavljivi na svetovnem spletu.

3.2 Inicializacija sistema

Preden sistem lahko začne z opravljanjem svoje funkcije, je potrebno inicializirati vse strukture in zunanje naprave, ki jih bo uporabljal. Za vsako napravo posebej vklopimo uro (Slika 3.2), skonfiguriramo ter inicializiramo GPIO napravo (Slika 3.3). Skonfiguriramo tudi zunanjo prekinitev (Slika 3.4), jo povežemo z GPIO napravo in inicializiramo, ter skonfiguriramo in



Slika 3.1: Razvojno okolje Eclipse in orodje System Workbench

inicializiramo še njen prekinitveni krmilnik (Slika 3.5).

```
// vklopimo uro GPIOC naprave
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC,ENABLE);
```

Slika 3.2: Vklop ure GPIO naprave

Če bomo napravo uporabljali kot vmesnik UART, moramo pina GPIO inicializirati za alternativne funkcije (Slika 3.6) in prekinitvenemu krmilniku sporočiti, da bo prekinitev prožil signal UART Tx. Seveda je v tem primeru potrebno še skonfigurirati in inicializirati vmesnik UART (Slika 3.7). Zunanje prekinitve tukaj ne potrebujemo.

Za projekt sem potreboval 7 GPIO naprav, ki so opravljale različne funkcije. Trem sem omogočil še zunanje prekinitve, tri pa sem uporabil kot vmesnike UART. Vsaki GPIO napravi sem omogočil prekinitev ter dodelil

```
// skonfiguriramo GPIO napravo
GPIO_InitTypeDef GPIO_InitStructure;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;

// inicializiramo GPIOC napravo
GPIO_Init(GPIOC, &GPIO_InitStructure);
```

Slika 3.3: Konfiguracija in inicializacija GPIO naprave

```
// skonfiguriramo GPIOC napravo za zunanjo prekinitev
SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOC, EXTI_PinSource1);

// skonfiguriramo EXTI zunanjo prekinitev
EXTI_InitTypeDef EXTI_InitStructute;
EXTI_InitStructute.EXTI_Line = EXTI_Line1;
EXTI_InitStructute.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_InitStructute.EXTI_Trigger = EXTI_Trigger_Falling;
EXTI_InitStructute.EXTI_LineCmd = ENABLE;

// skonfiguriramo EXTI zunanjo prekinitev
EXTI_Init(&EXTI_InitStructute);
```

Slika 3.4: Konfiguracija in inicializacija zunanje prekinitve

```
// skonfiguriramo NVIC krmilnik
NVIC_InitTypeDef NVIC_InitStructure;
NVIC_InitStructure.NVIC_IRQChannel = EXTI1_IRQn;
NVIC_InitStructure.NVIC_IRQChannelCmd = 0x01;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;

// inicializiramo NVIC krmilnik
NVIC_Init(&NVIC_InitStructure);
```

Slika 3.5: Konfiguracija in inicializacija prekinitvenega krmilnika

prekinitveni vektor, ki se proži ob spremembi vrednosti.

Uporabniške LED diode so edina naprava GPIO, ki nima svojega prekinitvenega vektorja. Krmilim jih ročno. Zvezane so na pine 3, 4, 5 in 6 na napravi GPIOD.

Senzor za zaznavo gibanja za sporočanje vrednosti uporablja pin 1 na-

```
// inicializiramo pine za alternativno funkcijo
GPIO_PinAFConfig(GPIOB, GPIO_PinSource6, GPIO_AF_USART1);
GPIO_PinAFConfig(GPIOB, GPIO_PinSource7, GPIO_AF_USART1);
```

Slika 3.6: Konfiguracija pinov za alternativno funkcijo

```
// skonfiguriramo UART
USART_InitTypeDef USART_InitStructure;
USART_InitStructure.USART_BaudRate = 9600;
USART_InitStructure.USART_WordLength = USART_WordLength_8b;
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Parity = USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;

// inicializiramo UART
USART_Init(USART1, &USART_InitStructure);

// vklopimo UART prekinitev
USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);

// skonfiguriramo UART NVIC krmilnik
NVIC_InitTypeDef NVIC_InitStructure;
NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
NVIC_InitStructure.NVIC_IRQChannelCmd = 0x01;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;

// inicializiramo UART NVIC krmilnik
NVIC_Init(&NVIC_InitStructure);

// vklopimo UART
USART_Cmd(USART1, ENABLE);
```

Slika 3.7: Konfiguracija in inicializacija vmesnika UART in njegovega prekinitvenega krmilnika

prave GPIOC. Vezan je na zunanjo prekinitev na liniji 1 (EXTI_Line1), ki proži prekinitveni vektor na kanalu 1 (EXTI1_IRQn).

Magnetni senzor svoje stanje sporoča preko pina 11 naprave GPIOB. Njegov prekinitveni vektor na kanalu od 10 do 15 (EXTI15_10_IRQn) pa proži zunanja prekinitev na liniji 11 (EXTI_Line11).

Koda prekinitvenega vektorja senzorja plinov se nahaja na kanalu od 5 do 9 (EXTI9_5_IRQn). Proži ga zunanja prekinitev na liniji 7 (EXTI_Line7).

Senzor z STM-om komunicira preko 7 pina naprave GPIOC.

Pina 6 in 7 naprave GPIOD sem uporabil kot pina Tx in Rx vmesnika UART1 za komuniciranje s serijsko konzolo na računalniku. Torej preko UART1 sem si na računalnik pošiljal izpise med izvajanjem programa. S tem sem si olajšal razhroščevanje programske kode in sporočal, v kakšnem stanju so senzorji ter razvojna plošča. Prekinitveni vektor se nahaja na kanalu UART1 (USART1_IRQn).

Napravo GPIOD ter pina 5 in 6 sem uporabil za serijsko komunikacijo z oddaljenim strežnikom. Pina služita kot pina Tx in Rx vmesnika UART2. Njegov prekinitveni vektor se nahaja na kanalu UART2 (USART2_IRQn).

Da lahko kominuciram z GSM modulom, sem uporabil še vmesnik UART3 in njegov prekinitveni vektor na kanalu USART3 (USART3_IRQn). Modul z STM ploščo sem povezal preko pina 8 in 9 naprave GPIOD.

3.3 Prekinitve GPIO naprav

Prekinitev je signal, ki je poslan centralni procesni enoti, signalizira pa dogodek, ki potrebuje takojšno pozornost. Prekinitev lahko proži tako strojna oprema, torej senzorji, moduli, periferne naprave, itd., kot tudi programska oprema. Procesna enota se na prekinitev odzove tako, da takoj preneha izvajati trenutne procese, shrani trenutno stanje na sklad in začne z izvajanjem prekinitveno servisnega podprograma oz. prekinitvene funkcije. Ko konča z izvajanjem prekinitvene funkcije, obnovi stanje s sklada ter nadaljuje z izvajanjem istega procesa, ki je bil prekinjen. Prekinitveni vektor je naslov na katerem je napisana koda, ki jo bo izvajal prekinitveni krmilnik [12].

Kot sem že omenil, sem vsaki GPIO napravi, razen uporabniškim LED diodam, vklopil prekinitve. Vse prekinitve imajo enako prioriteto, zato se v vrsto za izvajane dodajajo tako kot so se prožile.

Prekinitvene funkcije senzorja za zaznavo premikov (Slika 3.8), magnetnega senzorja (Slika 3.9) in senzorja za zaznavo plinov (Slika 3.10) so si med seboj zelo podobne. Prekinitev se sproži, ko senzor zazna odstopanje

od optimalne vrednosti. Prekinitvena funcija najprej preveri, ali je res ta GPIO naprava zahtevala prekinitev (Slika 3.11). Nato preveri, ali je senzor sploh omogočen, vrednost, ki jo ob inicializaciji sporoči oddaljeni strežnik, je shranjena v strukturi senzorja. V primeru, da sta oba pogoja izpolnjena, prekinitvena funkcija preko SMS sporočila takoj obvesti, da je senzor zaznal neobičajno stanje.

```
if (checkSensor("Motion")) {

    printDebug("");
    printDebug("MOTION SENSOR INTERRUPT!.....................DETECTED MOVEMENT!!");

    // utripne modro ledico
    blinkLed3Sec(5.0, "blue");

    // pogledamo ali je GSM modul omogočen
    // checkSensor() - preveri vrednost shranjeno v strukturi
    if (checkSensor("GSM"))
        sendNotifications("Motion");

    printDebug("INFO: Motion Sensor Available Again");
}
```

Slika 3.8: Prekinitvena funkcija senzorja za zaznavo gibanja

```
if (checkSensor("Gas")) {

    printDebug("");
    printDebug("GAS SENSOR INTERRUPT!.....................DETECTED!!");

    // utripne oranžno ledico
    blinkLed3Sec(0.5, "orange");

    // pogledamo ali je GSM modul omogočen
    // checkSensor() - preveri vrednost shranjeno v strukturi
    if (checkSensor("GSM"))

        // pošljemo opozorilo
        sendNotifications("Gas");
}
```

Slika 3.9: Prekinitvena funkcija senzorja za zaznavo plinov

Prekinitvena funkcija vmesnika UART2 je zadolžena za branje ukazov, ki jih pošilja oddaljeni strežnik. Znake na vodilu bere toliko časa, dokler ne

```
if (checkSensor("Reed")) {

    magneticSensor1 = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_11);
    if (!magneticSensor1) {
        printDebug("MAGNETIC SENSOR INTERRUPT!.....................reattached!!!!");

        // utripne zeleno ledico
        blinkLed3Sec(0.5, "green");
    } else {
        printDebug("");
        printDebug("MAGNETIC SENSOR INTERRUPT!.....................deattached!!!!");

        // pogledamo ali je GSM modul omogočen
        // checkSensor() - preveri vrednost shranjeno v strukturi
        if (checkSensor("GSM"))
            sendNotifications("Reed");
    }
}
```

Slika 3.10: Prekinitvena funkcija magnetnega senzorja

```
// pregledamo ali je res senzor plinov zahteval prekinitev
if (EXTI_GetITStatus(EXTI_Line7)) {

    // pogledamo ali je senzor plinov omogočen
    // checkSensor() - preveri vrednost shranjeno v strukturi
    if (checkSensor("Gas")) {
```

Slika 3.11: Preverjanje vira prekinitve ter vrednosti senzorja

prebere v naprej sprogramiranega znaka za konec branja. Nato celoten niz znakov sestavi v ukaz. Glede na prebrani ukaz, izvede zaporedje funkcij ter pobriše prekinitveno zastavico.

Prepoznati zna naslednje ukaze:

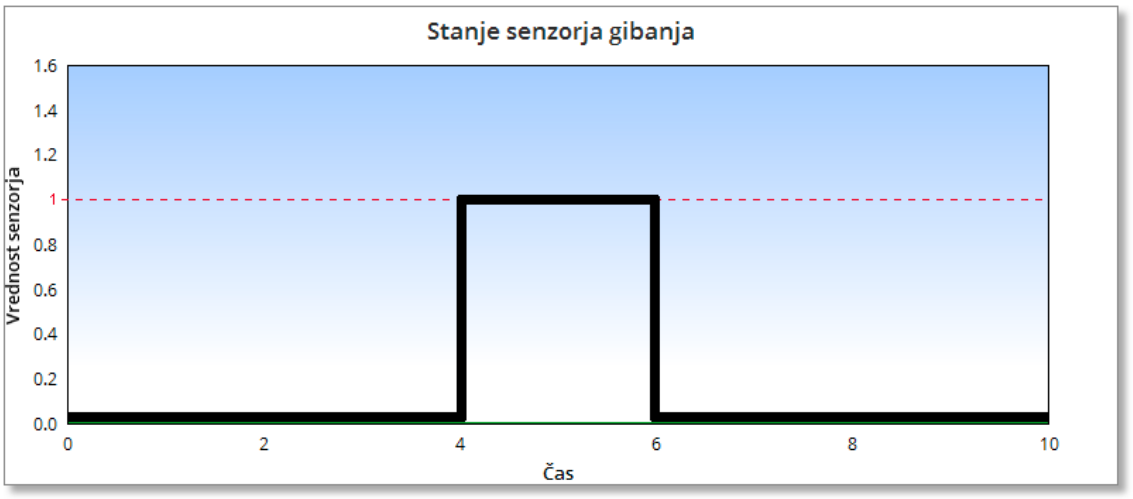
- **ping** - oddaljeni strežnik preverja, ali sistem še deluje
- **addalarm** - napolni strukturo alarma
- **adduser** - napolni strukturo uporabnikov
- **addsensor** - napolni strukturo senzorjev
- **updateuser** - posodobi strukturo uporabnikov
- **updatesensor** - posodobi strukturo senzorjev

- **Hello, I'm alive!** - oddaljeni strežnik sporoča da deluje
- **YEAH STILL HERE** - oddaljeni strežnik odgovarja da še vedno deluje

3.4 Delovanje sistema

Senzorji neprestano oddajajo eno izmed dveh razpoložljivih logičnih stanj. Ko zaznajo nenavadno dogajanje v prostoru, spremenijo svojo logično stanje. S tem prožijo prekinitev. Senzor gibanja in senzor plina oddajata nizko logično stanje, kadar ne zaznavata nič. V nasprotnem primeru oddajata visoko logično stanje. Magnetni senzor pa oddaja nizko logično stanje, kadar dela senzorja nista sklenjena. Torej, kadar so vrata odprta.

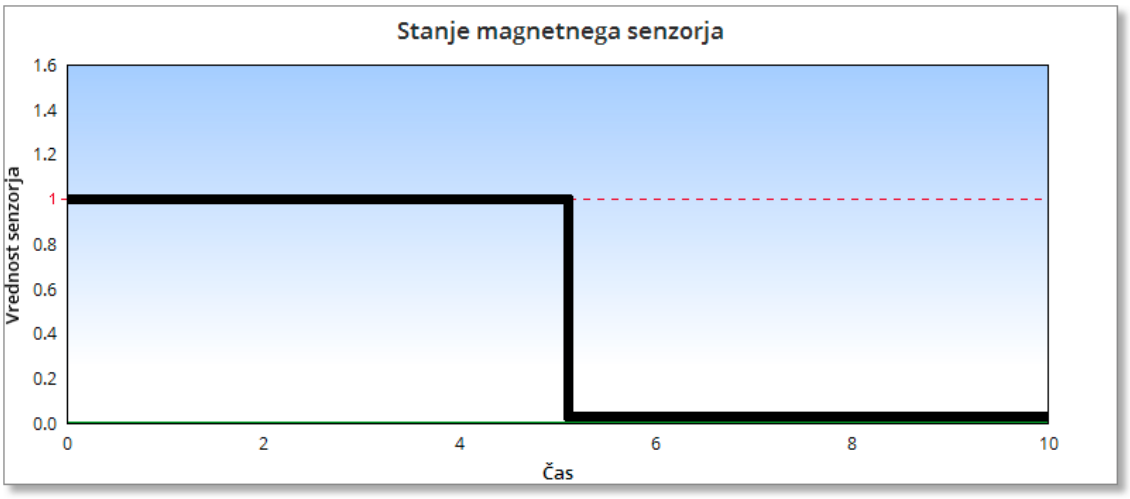
Slika 3.12 prikazuje stanje senzorja gibanja v časovnem intervalu 10 s. Opazimo lahko, da v 4. s senzor zazna premikanje.



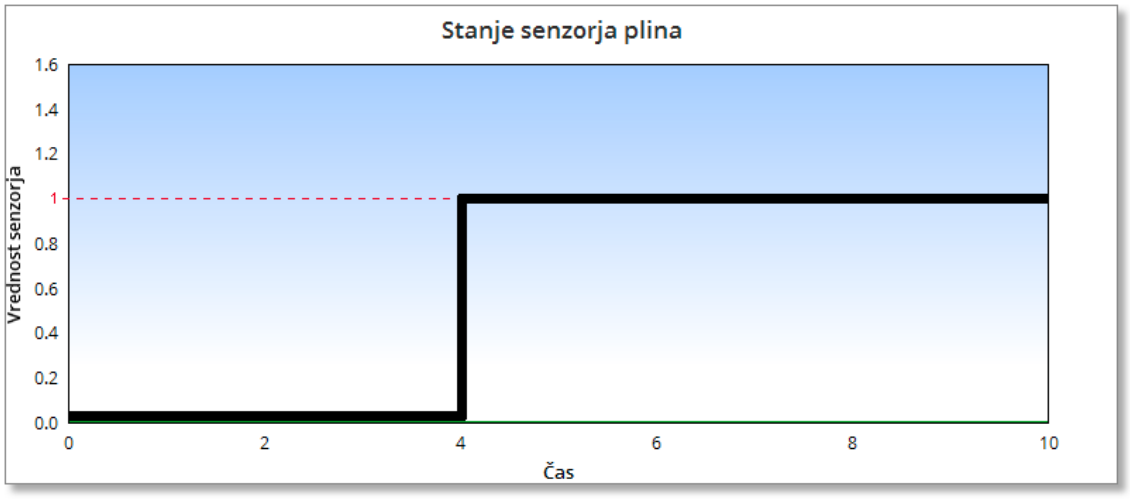
Slika 3.12: Stanje senzorja gibanja

Na Sliki 3.13 si lahko pogledamo stanje magnetnega senzorja v časovnem intervalu 10 s. Opazimo, da so bila v 5. s vrata odprta.

Slika 3.14 pa prikazuje stanje senzorja plina. V 4. s senzor zazna prisotnost neželjenega plina.



Slika 3.13: Stanje magnetnega senzorja



Slika 3.14: Stanje senzorja plina

Poglavje 4

Spletna aplikacija

Za oddaljeni strežnik sem si izbral mikroračunalnik Raspberry Pi 3. Nanj sem namestil na Debianu baziran operacijski sistem Raspbian. Raspbian ima poleg visoke stopnje optimizacije pripravljenih več kot 35000 paketov vnaprej prevedene programske kode za izredno enostavno namestitev [3]. Prav tako ima že precej vnaprej nameščene programske opreme za izobraževalno, programersko ter splošno namensko uporabo. Python, Scratch, Sonic Pi, Java, Mathematica, Minecraft Pi in Chromium so le del, tega kar ponuja Raspbian [15]. Od leta 2015 je uradno priznan ter uporabljen operacijski sistem dobrodelne fundacije Raspberry Pi [4]. Raspbian OS je odprtokoden prostodostopen operacijski sistem s konstantnimi posodobitvami in nenehnim razvojem. Brezplačno ga je možno prenesti iz uradne spletne strani Raspbian. Na Raspberry Pi 3 sem ga namestil s pomočjo mikro SD kartice, na katero sem zapekel sliko operacijskega sistema.

Spletna aplikacija, v katero se je potrebno prijaviti z uporabniškim imenom ter geslom, omogoča spreminjanje nastavitev celotnega sistema. Možno je pregledovati stanje senzorjev, jih vklapljati in izklapljati, nastaviti uro, kdaj naj bo sistem vklopljen, spreminjati nastavitve uporabnika ter spremeniti uporabniško geslo. Zaradi varnosti po izteku veljavnosti seje spletna aplikacija vsakega uporabnika avtomastko izpiše iz računa. Celotna aplikacija je napisana v programskem jeziku C. Za različne dele aplikacije sem

uporabil prostodostopne C knjižnice, ki sem jih našel na spletu.

Za shranjevanje podatkov o sistemu, uporabnikih in senzorjih sem uporabil podatkovno bazo SQL. Shranjevanje v in pridobivanje iz podatkovne baze sem si olajšal s knjižnico **libmysqlclient-dev**. S pomočjo knjižnice sem spisal funkcije za pridobivanje, posodabljanje in shranjevanje podatkov iz tabel, katere sem uporabil v glavnem programu. Za dostop do podatkov je najprej potrebno odpreti povezavo do baze. Nato sestavimo SQL (ang. *Structured Query Language*) stavek z vsemi potrebnimi parametri in izvedemo poizvedbo (Slika 4.1). V primeru, da smo podatke zahtevali, torej smo izvedli stavek SELECT, in je bila poizvedba uspešna, podatke razčlenimo, zapremo povezavo do baze ter vrnemo rezultate programu, ki je klical funkcijo (Slika 4.2).

```
int editSensor(int sensId, char *enabled)
{
    int ret = 1;
    char query[255];
    MYSQL *con;

    if ((con = mysql_init(NULL)) == NULL)
    {
        fprintf(stderr, "%s\n", mysql_error(con));
        printf("editSensor: mysql_init error\n");
        return 0;
    }

    if ((ret = openConnection(con)) == 0)
    {
        printf("editSensor: openConnection error\n");
        return ret;
    }

    sprintf(query, "UPDATE sensor SET enabled='%s' WHERE sensor_id='%d'", enabled, sensId);

    if (mysql_query(con, query)) {
        finish_with_error(con);
        printf("editSensor: mysql_query error\n");
    }

    mysql_close(con);

    return ret;
}
```

Slika 4.1: Primer posodobitve podatkov v tabeli sensor

Komunikacijo med ploščo STM32F4 ter Raspberry Pi 3 sem realiziral preko UART vmesnikov na obeh straneh. Na Raspberryu sem uporabil vme-

snik **serial0**, ki je privzeto namenjen Linuxovi konzoli. Zato sem moral v datoteki cmdline.txt odstraniti nastavitve privzete serijske konzole. Tudi tukaj sem uporabil prosto dostopno knjižnico **libcssl**. Za nemoteno komunikacijo je najprej potrebno skonfigurirati ter inicializirati vmesnik ter odpreti povezavo do plošče. Nato lahko aplikacija poljubno pošilja ter sprejema nize znakov, na podlagi katerih izvaja zaporedje funkcij. Glavna zanka konstano pregleduje, ali je STM plošča še aktivna, preverja, ali je potrebno sistem vklopiti/izklopiti, posodablja podatke uporabnikov ter senzorjev (Slika 4.3).

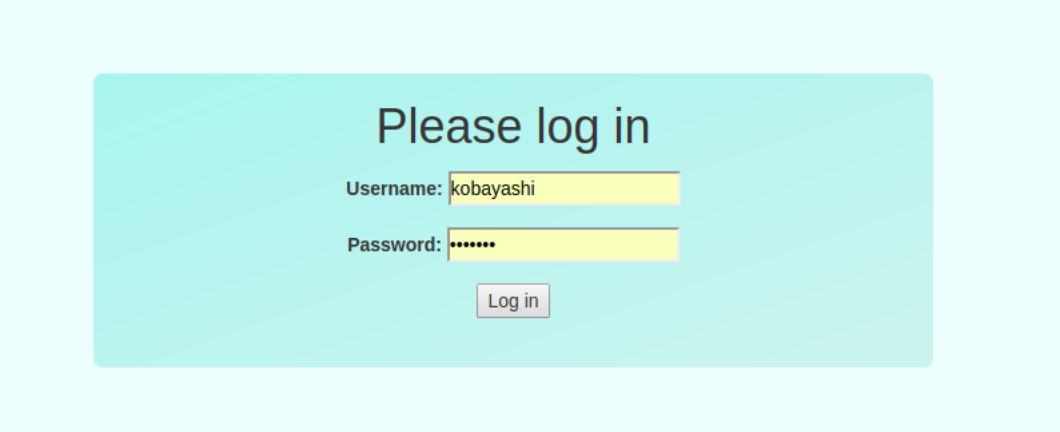
Za serviranje spletnih strani sem potreboval spletni strežnik. Izbral sem si prostodostopno knjižnico za vgrajene sisteme **mongoose**. Narejena je bila z namenom, da olajša povezovanje različnih naprav med seboj in jim omogoči enostavno povezljivost s spletom. Na trgu je že vse od leta 2004, najdemo pa jo lahko tako v odprtokodnih, kot tudi komercialnih produktih, bojda celo na mednarodni vesoljski postaji [10]. Za temelj moje aplikacije sem uporabil primer za vpis uporabnika v portal, ki ga ponuja že sama knjižnica. Seveda sem ga priredil in nadgradil, da ustreza mojim potrebam. Dodal sem še svoje html strani, katere sem izoblikoval s css stili. Aplikacija torej vsebuje 4 html spletne strani. Ob prvem zahtevku nam servira stran za vpis uporabnika (Slika 4.4). Če vpis spodleti, nas preusmeri na opozorilno stran, da je nekaj šlo narobe (Slika 4.5). Ob uspešnem vpisu pa nas preusmeri na glavno stran, kjer lahko nadzorujemo stanje sistema (Slika 4.6). Ob kliku na gumb "uredi alarm" (ang. *Edit Alarm*), se odpre novo okno, kjer lahko spreminjamo stanja senzorjev in nastavljamo čas, ko je sistem vklopljen (Slika 4.7 in 4.8). Gumb "uredi uporabnika" (ang. *Edit User*) nam odpre novo okno za spreminjanje uporabniških nastavitev ter spreminjanje gesla (Slika 4.9 in 4.10). Če želimo nastavitev spremeniti, moramo v obeh primerih klikniti na gumb "shrani" (ang. *Save*).

```c
int getUser(char *username, char *email, char *phone, char *fname, char *lname, char *utype, char *sms)
{
    int ret = 1;
    char query[255];
    MYSQL_ROW row;
    MYSQL *con;
    MYSQL_RES *result;

    // open connection
    if ((con = mysql_init(NULL)) == NULL)
    {
        fprintf(stderr, "%s\n", mysql_error(con));
        printf("getUser: mysql_init error\n");
        return 0;
    }

    if ((ret = openConnection(con)) == 0)
    {
        printf("getUser: openConnection error\n");
        return ret;
    }

    // make query
    sprintf(query, "SELECT email, phone_number, first_name, last_name, user_type,\
    sms_enabled FROM user WHERE username='%s' or email='%s';", username, username);

    if (mysql_query(con, query))
    {
        printf("getUser: mysql_query error\n");
        finish_with_error(con);
        mysql_close(con);
        return 0;
    }

    // get result - gets whole result set from DB (mysql_use_result() gets only one element!)
    if ((result = mysql_store_result(con)) == NULL)
    {
        printf("getUser: mysql_store_result error\n");
        finish_with_error(con);
        mysql_close(con);
        return 0;
    }

    if ((row = mysql_fetch_row(result)))
    {
        sprintf(email, "%s", row[0]);
        sprintf(phone, "%s", row[1]);
        sprintf(fname, "%s", row[2]);
        sprintf(lname, "%s", row[3]);
        sprintf(utype, "%s", row[4]);
        sprintf(sms, "%s", row[5]);

        ret = 1;
    }
    else {
        printf("getUser: mysql_fetch_row error\n");
        ret = 0;
    }

    mysql_free_result(result);
    mysql_close(con);

    return ret;
}
```

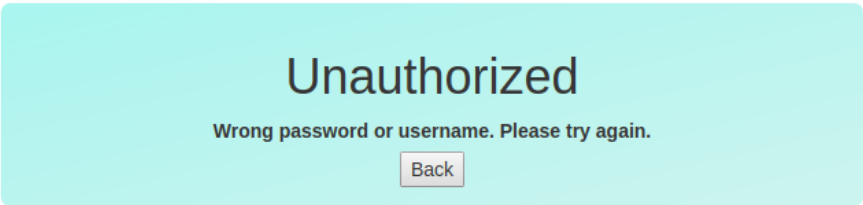
Slika 4.2: Primer pridobivanja podatkov iz tabele user

```
// send users info to STM
void sendUsers(char *action) {
    int i = 1;
    char cmd[128], username[64], phone[64], sms[64];

    printf("Sending users info to STM...\n");

    for (i = 1; i <= 10; i++) {
        if (getUserById(i, username, phone, sms)) {
            if (!strcmp(action, "add")) {
                sprintf(cmd, "adduser[%s;%s;%s]", username, phone, sms);
                sendString(cmd);
                usleep(150000);
            } else if (!strcmp(updateUser, username)) {
                sprintf(cmd, "updateuser[%s;%s;%s]", username, phone, sms);
                sendString(cmd);
                break;
            }

        }
    }
    printf("Sending users info.............SUCCESFUL\n\n");
}

// send alarm info to STM
void sendAlarm(char *action) {
    char cmd[128];

    printf("\nSending alarm info to STM...\n");

    // get info about alarm from db
    alarm.id = 1;
    if (getAlarm(alarm.id, alarm.name, alarm.enabled, &alarm.turnOn, &alarm.turnOff)) {
        sprintf(cmd, "%salarm[%s]", action, alarm.enabled);
        sendString(cmd);
        usleep(100000);
    }
    printf("Sending alarm info.............SUCCESFUL\n");
}

// send sensors info to STM
void sendSensors(char *action) {
    int i = 1;
    char cmd[128];

    printf("Sending sensors info to STM...\n");

    for (i = 0; i < 10; i++) {
        if (getSensor(alarm.id, i, sens[i].name, sens[i].enabled)) {
            sprintf(cmd, "%ssensor[%s;%s]", action, sens[i].name, sens[i].enabled);
            sendString(cmd);
            if (!strcmp(action, "add"))
                usleep(100000);
            else
                usleep(150000);
        } else
            strcpy(sens[i].enabled, "UNCONFIGURED");
    }
    printf("Sending sensors info..........SUCCESFUL\n");
}
```

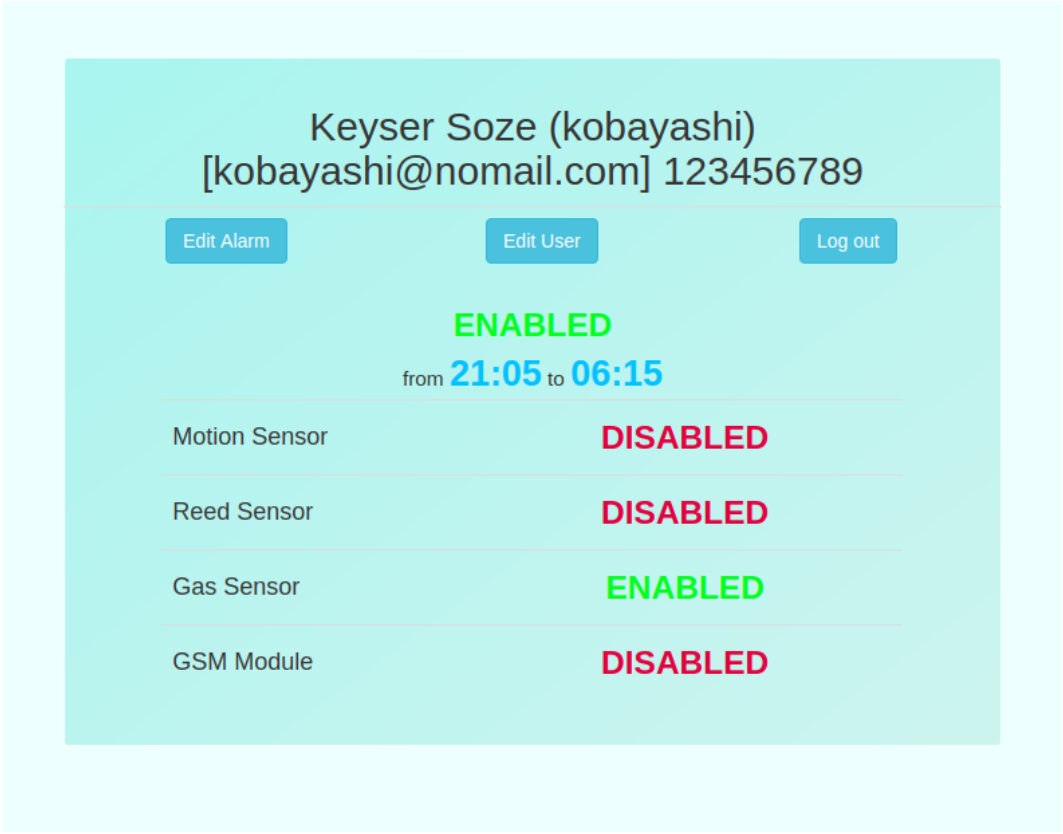
Slika 4.3: Sporočanje podatkov nadzorni enoti



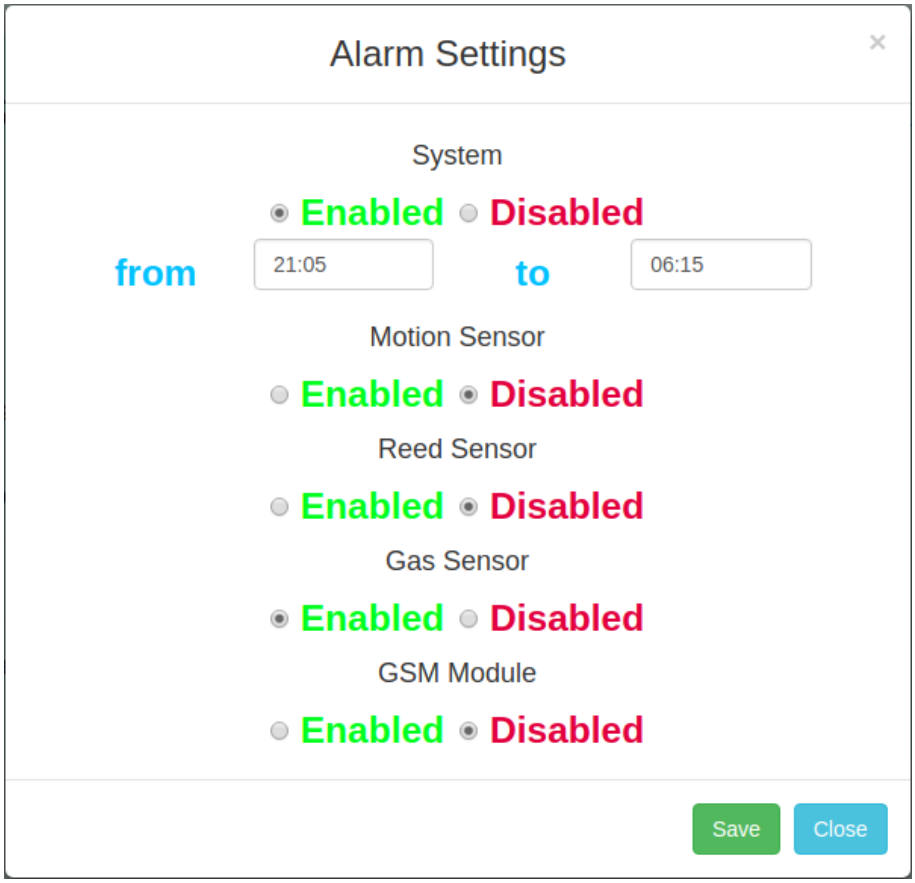
Slika 4.4: Vpisna stran



Slika 4.5: Opozorilo, napačno geslo



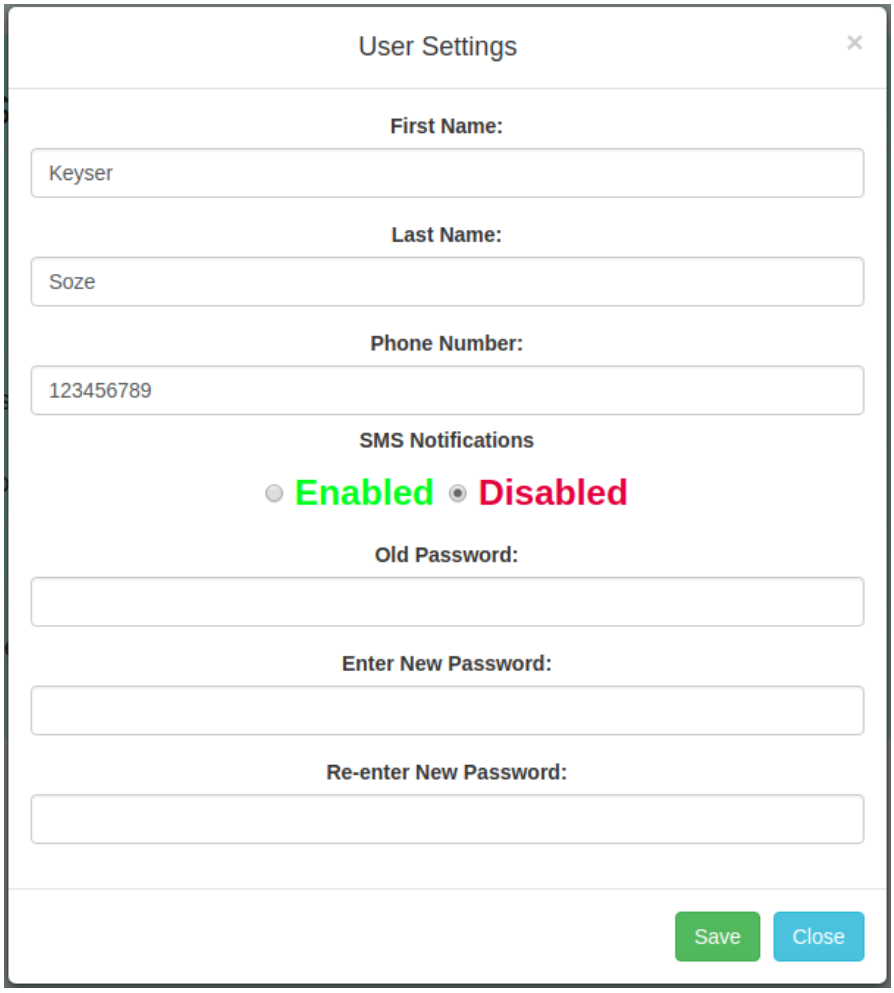
Slika 4.6: Nadzorna stran



Slika 4.7: Nastavitve sistema

```
static void handle_alarmSave(struct mg_connection *nc, struct http_message *hm) {
    char tOn[24], tOff[24], modulType[64], strId[64];
    int i = 0;

    // Get form variables and store settings values
    mg_get_http_var(&hm->body, "AlarmSystem", livingroom.enabled, sizeof(livingroom.enabled));
    mg_get_http_var(&hm->body, "turnon", tOn, sizeof(tOn));
    mg_get_http_var(&hm->body, "turnoff", tOff, sizeof(tOff));
    strptime(tOn, "%H:%M", &livingroom.turnOn);
    strptime(tOn, "%H:%M", &livingroom.turnOff);

    if (!editAlarm(1, "livingroom", livingroom.enabled, tOn, tOff))
        printf("ERROR: editAlarm");

    strcpy(modulType, "Sensor");
    for (i = 0; i < 10; i++) {

        if (strcmp(sens[i].enabled, "UNCONFIGURED")) {
            if (!strcmp(sens[i].name, "GSM"))
                    strcpy(modulType, "Module");

            sprintf(strId, "id%s%s", sens[i].name, modulType);
            mg_get_http_var(&hm->body, strId, sens[i].enabled, sizeof(sens[i].enabled));

            if (!editSensor(sens[i].sensor_id, sens[i].enabled))
                    printf("ERROR: editSensor");
        }
    }

    updateAlarm = 1;

    // Send response
    mg_http_send_redirect(nc, 302, mg_mk_str("/"), mg_mk_str(NULL));
}
```

Slika 4.8: Funkcija zadolžena za posodobitev podatkov sistema



Slika 4.9: Nastavitve uporabnika

```
static void handle_userSave(struct mg_connection *nc, struct http_message *hm) {
    char oldpass[64], newpass1[64], newpass2[64];
    // Get form variables and store settings values
    mg_get_http_var(&hm->body, "fname", usr.fname, sizeof(usr.fname));
    mg_get_http_var(&hm->body, "lname", usr.lname, sizeof(usr.lname));
    mg_get_http_var(&hm->body, "phone", usr.phone, sizeof(usr.phone));
    mg_get_http_var(&hm->body, "smsnotification", usr.sms, sizeof(usr.sms));
    mg_get_http_var(&hm->body, "oldpass", oldpass, sizeof(oldpass));
    mg_get_http_var(&hm->body, "newpass1", newpass1, sizeof(newpass1));
    mg_get_http_var(&hm->body, "newpass2", newpass2, sizeof(newpass2));

    // passwords are not empty -> editPassword
    if (strcmp(oldpass, "")) {
        if (!getUserByUsernameOrEmail(usr.username, oldpass))
            printf("DEBUG: handle_userSave, old pass DO NOT match\n");
        else {
            printf("DEBUG: handle_userSave, old pass DO match\n");

            if (!strcmp(newpass1, newpass2)) {
                if (!editPassword(usr.username, newpass1))
                    printf("ERROR: handle_userSave, editPassword\n");
            }
            else
                printf("DEBUG: handle_userSave, new passwords DO NOT match\n");
        }
    }

    if (!editUser(usr.username, usr.phone, usr.fname, usr.lname, usr.utype, usr.sms))
        printf("ERROR: editUser");

    strcpy(updateUser, usr.username);

    // Send response
    mg_http_send_redirect(nc, 302, mg_mk_str("/"), mg_mk_str(NULL));
}
```

Slika 4.10: Funkcija zadolžena za posodobitev podatkov uporabnika

Poglavje 5

Zaključek

Uspešno sem načrtoval ter izdelal delujoč domači alarmni sistem. Sistem je dovolj učinkovit za preprosto domačo uporabo, z možnostjo nadgradje in nadalnjega razvoja. Zaznati zna premikanje v prostoru, odpiranje oken ter vrat in uhajanje nezaželenih plinov. Preko GSM omrežja zna to sporočiti večjemu številu oseb. Omogoča tudi spreminjanje nastavitev.

Seveda pa sam proces izdelave ni minil brez težav. Vse zunanje komponente so nizkega cenovnega razreda, kupljene preko spletne trgovine z azijskega trga. Nekaj produktov sem dobil pokvarjenih, nekatere pa sem čakal dalj časa kot je bilo obljubljeno. Prvi GSM modul je bil nedelujoč, zato sem ga moral naročiti ponovno. Senzor za zaznavanje loma stekla pa je prav tako prišel pokvarjen, vendar pa časa za ponovni nakup ni bilo. Tako sem bil primoran omenjeni senzor opustiti in zmanjšati število senzorjev.

Mislim, da je majhno število senzorjev ena od slabosti sistema, ki bi se jo prav gotovo dalo izboljšati. Tudi njihova kvaliteta ni na najvišji ravni. Sistem bi bil tudi bolj enostaven za uporabo, če bi ga z oddaljenim strežnikom povezal preko brezžične povezave. Prednosti sistema so neomejeno število uporabnikov, cenovno ugodna izdelava in dostopnoost s katerekoli mobilne naprave. Ena od prednosti je tudi, da se je v sistem možno prijaviti le v lokalnem omrežju. S tem se zmanjša možnost zlorab in vdorov.

Mislim, da je bil cilj diplomske naloge uspešno dosežen.

Literatura

[1] Digitalni mikrofon MP45DT02 MEMS. Dostopno na:. "https://eu.mouser.com/new/stmicroelectronics/stm-mp45dt02-mems-microphone/"(2017).

[2] Digitalno-analogni pretvornik signala CS43L22. Dostopno na:. "https://www.cirrus.com/products/cs43l22/"(2017).

[3] Dobrodošli na Raspbian. Dostopno na:. "https://www.raspberrypi.org/downloads/raspbian/"(2017).

[4] Fundacija Raspberry PI. Dostopno na:. "https://en.wikipedia.org/wiki/Raspberry_Pi_Foundation"(2017).

[5] Gsm modul A6. Dostopno na:. "https://www.ebay.com/itm/A6-A7-Proto-Shield-GPRS-GSM-Module-Adapter-Quad-band-Antenna-900-1800-1900MHZ/172380452320?ssPageName=STRK%3AMEBIDX%3AIT&_trksid=p2057872.m2749.l2649"(2017).

[6] Infrardeči senzor za zaznavo gibanja. Dostopno na:. "https://en.wikipedia.org/wiki/Passive_infrared_sensor"(2017).

[7] Kako delujejo magnetni senzorji za vrata ter okna. Dostopno na:. "https://www.protectamerica.com/home-security-blog/tech-tips/door-window-sensors-work_8461"(2017).

[8] Magnetni senzor MC-38. Dostopno na:. "https://www.ebay.com/itm/1-2-5X-Wired-Door-Window-Sensor-Magnetic-Switch-Mc-

38-Home-Alarm-System-Detector-/322691900762?var=&hash=item0"(2017).

[9] Mikroračunalnik Raspberry PI 3. Dostopno na:. "https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/"(2017).

[10] Mongoose - vgrajen spletni strežnik / vgrajena omrežna knjižnica. Dostopno na:. "https://github.com/cesanta/mongoose"(2017).

[11] Pospeškometer LIS302DL. Dostopno na:. "http://www.st.com/en/mems-and-sensors/lis302dl.html"(2017).

[12] Prekinitev. Dostopno na:. "https://en.wikipedia.org/wiki/Interrupt"(2017).

[13] Raspberry PI 3 Model B. Dostopno na:. "http://wiki.seeed.cc/Raspberry_Pi_3_Model_B/"(2017).

[14] Raspberry PI. Dostopno na:. "https://en.wikipedia.org/wiki/Raspberry_Pi"(2017).

[15] Raspbian. Dostopno na:. "https://www.raspbian.org/"(2017).

[16] Razhroščevalnik ST-LINK/V2. Dostopno na:. "http://www.st.com/en/development-tools/st-link-v2.html"(2017).

[17] Razvojna plošča STM32F4 Discovery. Dostopno na:. "http://www.st.com/en/evaluation-tools/stm32f4discovery.html"(2017).

[18] Senzor za zaznavo gibanja HC-SR501. Dostopno na:. "https://www.ebay.com/itm/New-HC-SR501-Infrared-PIR-Motion-Sensor-Module-for-Arduino-Raspberry-pi-/141721687816?hash=item20ff455708"(2017).

[19] Senzor za zaznavo plinov. Dostopno na:. "https://en.wikipedia.org/wiki/Gas_detector"(2017).

[20] Senzor za zaznavo plinov MQ-2. Dostopno na:. "https://www.ebay.com/itm/MQ-2-MQ2-Smoke-Gas-LPG-Butane-Hydrogen-Gas-Sensor-Detector-Module-For-Arduino-/400368453822?hash=item5d37d1c8be"(2017).

[21] Slika GSM modula. Dostopno na:. "https://s3-ap-southeast-1.amazonaws.com/a2.datacaciques.com/16/10/07/08739op6miy64334/aa9ea5cdb7115aa9.jpg"(2017).

[22] Slika magnetnega senzorja. Dostopno na:. "https://www.google.si/search?biw=1855&bih=965&tbm=isch&sa=1&ei=n3pnWr6vPMiwsAf7v6nwBQ&q=MC-38+sensor&oq=MC-38+sensor&gs_l=psy-ab.3...1583.6392.0.6760.9.9.0.0.0.0.137.947.3j6.9.0....0...1c.1.64.psy-ab..0.3.306...0i19k1j0i13i30i19k1j0i8i30i19k1j0i10i30i19k1.0.ovBtS9cNm1c#imgrc=egImRlvzYqVkEM:"(2017).

[23] Slika PIR senzorja. Dostopno na:. "https://www.google.si/search?q=HC-SR501&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjgwsSYqefYAhUDaFAKHe6iAOUQ_AUICigB&biw=1855&bih=965#imgrc=U8o7tXUxqLS0-M:"(2017).

[24] Slika senzorja za zaznavo plinov. Dostopno na:. "https://ssli.ebayimg.com/images/g/2qQAAOSwQJhUk4Cn/s-l64.jpg"(2017).