UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Marko Krečič

# Vgrajen sistem za kontrolo dostopa

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Patricio Bulić

Ljubljana, 2012

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

*Tržaška 25*
*1001 Ljubljana, Slovenija*
*telefon: 01 476 84 11*
*01 476 83 87*
*faks: 01 426 46 47*
*01 476 87 11*
*www.fri.uni-lj.si*
*e-mail: dekanat@fri.uni-lj.si*

Št. naloge: 01874/2012
Datum: 03.10.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MARKO KREČIČ**

Naslov: **VGRAJEN SISTEM ZA KONTROLO DOSTOPA**
**AN EMBEDDED SYSTEM FOR ACCESS CONTROL**

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Načrtujte in izdelajte vgrajen sistem za kontrolo dostopa. Kontrola dostopa naj temelji na dvostopenjski identifikaciji: z uporabnikovo značko RFID ter kodo PIN, ki jo uporabnik vnese preko zaslona na dotik. Vgrajen sistem naj bo zasnovan na mikrokrmilniku z jedrom ARM Cortex-M3. Za implementacijo sistema uporabite ceneni čitalec pasivnih značk RFID, ki delujejo s frekvenco 125 kHz, grafični zaslon LCD na dotik ter razvojni kit STMicroelectronics STM32F103RB. Preverjanje identitete naj opravlja aplikacija na osebnem računalniku, ki z vgrajenim sistemom komunicira preko omrežene povezave. Aplikacija na osebnem računalniku naj vodi dnevnik prijav in dostopov.

Mentor:
prof. dr. Patricio Bulić

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

# IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Marko Krečič,

z vpisno številko **63060143**,

sem avtor diplomskega dela z naslovom:

*Vgrajen sistem za kontrolo dostopa*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Patricia Bulića,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 11. december 2012 Podpis avtorja:

*Rad bi se zahvalil mentorju prof. dr. Patriciu Buliću za mentorstvo in pomoč pri izdelavi diplomske naloge.*

*Zahvaljujem se družini, predvsem staršem, za podporo in razumevanje v času študija.*

# Kazalo


Povzetek....1
Abstract....3
1 Uvod....5
2 Tehnologija RFID....7
3 Zaslon na dotik....9
3.1 Rezistivni zasloni na dotik....9
4 Sistem za kontrolo dostopa....13
4.1 Strojna oprema....14
4.1.1 Razvojna plošča, mikrokrmilnik STM32F103RB....15
4.1.2 Čitalec RFID....17
4.1.3 Zaslon na dotik....18
4.1.3.1 Zaslon LCD in krmilnik za LCD SSD1289....19
4.1.3.2 Krmilnik za zaznavo dotikov ADS7843....21
4.1.4 Ethernet modul ENC28J60....22
4.2 Programska rešitev....23
4.2.1 Programiranje mikrokrmilnika....24
4.2.2 Konfiguracija krmilnika za LCD SSD1289....24
4.2.3 Konfiguracija krmilnika za zaznavanje dotika ADS7843....28
4.2.4 Kalibracija zaslona....32
4.2.5 Konfiguracija čitalca RFID....36
4.2.6 Konfiguracija krmilnika za Ethernet ENC28J60....43
4.2.7 Sklad uIP TCP/IP....44
4.2.8 Končni avtomat....52
4.2.9 Strežniški program....57
5 Sklepne ugotovitve....61
A Inicializacija krmilnika LCD SSD1289....63
B Shema povezav....65
Kazalo slik....67
Literatura....69

# Seznam uporabljenih kratic in simbolov

| | |
|---|---|
| RFID | Radio Frequency Identification |
| SPI | Serial Peripheral Interface |
| CRC | Cyclic Redundancy Check |
| MAC | Media Access Control |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| IP | Internet Protocol |
| ICMP | Internet Control Message Protocol |
| UDP | User Datagram Protocol |
| TCP | Transmission Control Protocol |
| APB | Advanced Peripheral Bus |
| AHB | Advanced High-performance Bus |
| USART | Universal synchronous asynchronous receiver transmitter |
| RAM | Random-Access Memory |
| LCD | Liquid Crystal Display |
| A/D | analogno/digitalni |
| TTL | Transistor-Transistor Logic |
| PIN | Personal Identification Number |
| DMA | Direct Memory Access |
| NVIC | Nested Vectored Interrupt Controller |
| TLS | Transport Layer Security |
| SSL | Secure Sockets Layer |

# Povzetek

Kljub temu da za omejitev vstopa v prostor danes še vedno prevladuje uporaba mehanskih ključavnic, se marsikje pojavlja potreba po bolj prilagodljivi in nadzorovani metodi zaklepanja in odklepanja vrat, ki jo predstavlja uporaba sistema za kontrolo dostopa.

V okviru diplomske naloge smo izdelali vgrajen sistem za kontrolo dostopa, ki uporablja tehnologijo RFID in dvostopenjsko preverjanje identitete uporabnika, kontrolo in nadzor pa izvaja strežniški program na računalniku. Sistem sestavljajo mikrokrmilnik z arhitekturo ARMv7 (jedro ARM Cortex-M), modul za Ethernet, zaslon na dotik in čitalec kartic RFID. Sistem za kontrolo dostopa s čitalcem kartic RFID prebere številko kartice, nato pa izriše številčnico na zaslon LCD. Ko uporabnik z dotikom gumbov na zaslonu vnese kodo PIN, se številka kartice in koda PIN skupaj z identifikacijsko številko naprave pošlje v preverjanje strežniku, ki zahtevo za vstop potrdi oziroma zavrne. Strežnik vsako pošiljanje prijavnih podatkov zabeleži.

V nalogi smo predstavili tehnologijo RFID in delovanje rezistivnih zaslonov na dotik. Sledila je predstavitev sestavnih delov sistema in njihovo delovanje. V programskem delu naloge smo opisali nastavitve modulov in prikazali uporabo naprave DMA. Predstavili smo problem mehanske neporavnanosti zaslona na dotik in kalibracijski algoritem. Ker se povezujemo z napravo na strežnik preko omrežja Ethernet, smo morali uporabiti protokola TCP/IP. Odločili smo se za sklad uIP TCP/IP, katerega uporabo smo predstavili na primeru izdelanega sistema za kontrolo dostopa. V sklopu programskega dela smo predstavili še končni avtomat sistema in opisali strežniški program.

**Ključne besede:**

tehnologija RFID, zaslon na dotik, sklad uIP TCP/IP, kalibracija zaslona, kontrola pristopa

# Abstract

Although the use of mechanical locks to limit entry into space is still predominating today, there is a need for a more flexible and controllable method of locking and unlocking the door. That method can be represented by the use of an embedded access control system.

In the thesis we have developed an embedded system for access control that uses a RFID technology and two-factor authentication, while a server program on a personal computer executes control and surveillance. The system is composed of a microcontroller with ARMv7 architecture (ARM Cortex-M core), Ethernet module, touch screen LCD and RFID card reader. Access control system reads the card number with a RFID card reader and then shows the keypad on the LCD touch screen. When a user enters the PIN code by touching buttons on LCD, the PIN code along with the RFID card identification number are sent to the server that approves or refuses the entry. The server logs user authentication.

In this work we have presented the RFID technology and the operation of resistive touch screens. This has been followed by the presentation of the system components and their operation. In the programming part we have described the configuration of modules and the use of DMA devices. We have presented the problem of the mechanical misalignment of the touch screen and calibration algorithm. Because of the fact we connect the device to a server via Ethernet, we need to use TCP/IP protocols. We have chosen uIP TCP/IP stack and described the application of it based on embedded system for access control that we have made. We have also presented the finite state automaton of the system and described the server program.



**Key words:**

RFID technology, touch screen, uIP TCP/IP stack, touch screen calibration, access control

# 1 Uvod

Že od nekdaj si človek prizadeva zavarovati svoje ozemlje pred nezaželenimi obiskovalci. Čeprav še vedno prevladuje zaklepanje z mehansko ključavnico, pa se pojavlja potreba po bolj prilagodljivi in nadzorovani metodi zaklepanja in odklepanja vrat, ki jo predstavlja uporaba sistema za kontrolo dostopa. Vzemimo za primer podjetje, ki ima veliko zaposlenih in večje število prostorov, kjer imajo omejen dostop. Pri uporabi ključev se že na začetku pojavijo problemi, saj je treba narediti veliko kopij ključev za vsako ključavnico posebej ter voditi evidenco, komu se je dalo kateri ključ. Nevarnost uporabe ključev za omejevanje dostopa v tem primeru predstavlja tudi možnost lahkega kopiranja ključa, ki lahko pride v napačne roke. Problem nadzora vstopov v prostore se s številom prostorov dodatno povečuje. Ena od rešitev predstavljenih problemov bi bila uporaba sistema za kontrolo pristopa.

Sistem za kontrolo dostopa je sistem, ki omejuje dostop do posameznih prostorov. Sistem sestavljata kontrolna enota in električna ključavnica.

Kontrolno enoto sistema kontrole pristopa sestavljata strojna in programska oprema, ki omogočata, da posameznikom dovolimo ali onemogočimo vstop (pristop) do varovanih prostorov. Programski del preveri uporabniške podatke (razne kode), strojni del pa služi kot vmesnik med uporabnikom in programskim delom.

Poznamo več sistemov kontrole pristopa, ki jih med seboj ločimo glede na način delovanja in na način identifikacije. Glede na način delovanja ločimo samostojne sisteme in sisteme, ki jih je treba priključiti na napravo, ki hrani prijavne podatke. Za identifikacijo trenutno obstaja več načinov: brezkontaktne kartice, magnetne kartice, kode PIN (ang. *Personal Identification Number*), identifikacija z branjem prstnih odtisov in drugih biometričnih značilnosti. Sisteme ločimo tudi glede na načine identifikacije, ki jih lahko med seboj kombiniramo.

Današnje komercialne izvedenke sistemov pogosto delujejo v povezavi s sistemi za registracijo delovnega časa. Pri kontroli dostopa se identifikacijski podatki pošljejo v strežnik, ki beleži prihode in odhode. Tako imamo vedno podatke kje, kdo in kdaj je uporabil sistem, kar nam omogoča lažji nadzor nad zaposlenimi in nam olajša beleženje delovnega časa.

Sistemi za kontrolo pristopa lahko nadzirajo vrata, ograje, dvigala, rampe – vse, kar lahko krmilimo elektronsko.

Cilj diplomske naloge je izdelati sistem za kontrolo dostopa, ki ni omejen s številom uporabnikov in se poslužuje uporabe tehnologije RFID (ang. *Radio Frequency Identification*). Imeti mora možnost dodajanja in brisanja uporabnikov, identiteta uporabnikov se preverja dvostopenjsko. Za beleženje uporabe mora biti sistem povezan na računalnik.

V diplomski nalogi si bomo najprej ogledali delovanje tehnologije RFID. Na kratko bomo predstavili zgodovino tehnologije, vrste elektronskih oznak (značk) in branje brezkontaktne kartice. Sledila bo predstavitev zaslonov na dotik, podrobneje si bomo pogledali delovanje rezistivnih zaslonov na dotik.

Drugi del obsega predstavitev izdelave sistema za kontrolo dostopa. Predstavljena bo strojna oprema, kjer bomo opisali komponente, ki sestavljajo sistem. Sledila bo programska rešitev, kjer bomo predstavili konfiguracijo različnih komponent sistema, ogledali si bomo branje

dotikov iz krmilnika za zaznavanje dotika. Predstavljena bo komunikacija mikrokrmilnika z zaslonom LCD (ang. *Liquid Crystal Display*), nastavitev krmilnika za Ethernet in delovanje sklada uIP TCP/IP (ang. *Transmission Control Protocol/Internet Protocol*), ki smo ga uporabili. Ogledali si bomo tudi postopek kalibracije zaslona, predstavili bomo končni avtomat, ki upravlja sistem. Predstavitev programske rešitve bomo zaključili z opisom strežnika, kateremu naprava posreduje prijavne podatke in katerega čaka na odgovor o veljavnosti prijave.

# 2 Tehnologija RFID

Radiofrekvenčna identifikacija (kratica RFID) je tehnologija za prenos podatkov med čitalcem in elektronsko oznako z namenom identifikacije, ki za medij prenosa uporablja radiofrekvenčne valove [1].

Začetek uporabe RFID tehnologije naj bi bil v štiridesetih letih 20. stoletja v vojaške namene, ko so Britanci za identifikacijo letal razvili sistem IFF (ang. *Identify Friend or Foe*), da bi lažje prepoznali svoja letala. V vsako britansko letalo so vgradili oddajnik, ki je ob sprejemu signala iz radarskih postaj začel oddajati nek signal nazaj; tako so vedeli, da je letalo domače. Raziskave na področju tehnologije RFID so se nadaljevale v petdesetih in šestdesetih letih 20. stoletja, ko so znanstveniki opravljali raziskave, kako na daljavo prepoznati objekte. Takrat so nastala prva podjetja, ki so se ukvarjala s sistemi za preprečevanje kraje, ki so za delovanje uporabljali radijske valove. Prvi patent v Združenih državah Amerike za tehnologijo RFID naj bi leta 1973 prejel Mario W. Cardullo za aktivno značko RFID s pomnilnikom, ki ga lahko prepišemo. Istega leta je Charles Walton prejel patent za odklepanje vrat brez ključa z uporabo pasivne značke [2]. Uporaba tehnologije RFID se je kasneje razširila v logistiki za sledenje kontejnerjev, vozil, za plačevanje cestnine (ABC v Sloveniji), za sledenje živalim.

Sistemi za radiofrekvenčno identifikacijo uporabljajo elektronske oznake – značke, ki so pripete objektu, ki ga hočemo prepoznati, in čitalce. Čitalci so dvosmerni radijski oddajniki/sprejemniki, ki najprej oddajajo signal, potem pa sprejemajo signal, ki ga oddaja značka. Značke so sestavljene iz antene (navite žice) in vezja, ki vsebuje osnovno vezje za modulacijo signala in trajen pomnilnik [3].

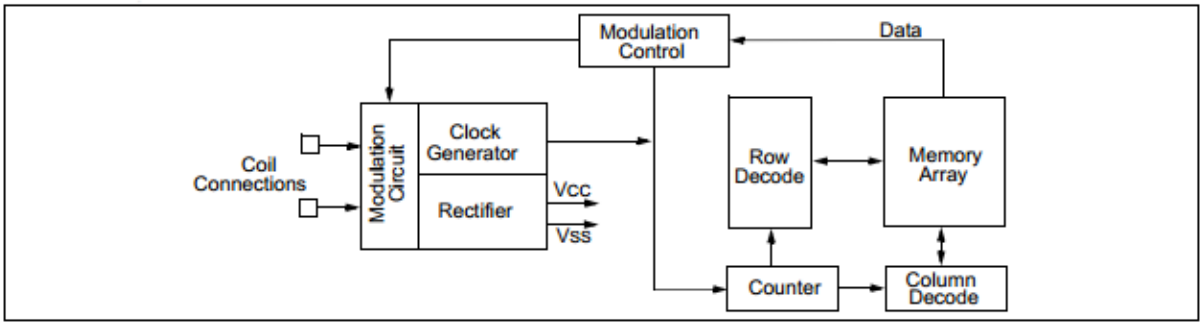

*Slika 1: Pasivne značke*

Poznamo pasivne, aktivne in pasivne značke z baterijo, ki jih ločimo glede na napajanje. Aktivne značke imajo vgrajeno baterijo in periodično pošiljajo signal z identifikacijo značke. Običajno lahko vsebino aktivnih značk spreminjamo, jo večkrat prepišemo. Pasivne značke (prikazane na sliki 1) so cenejše, nimajo baterije in se napajajo z energijo radijskih valov, ki jih pošilja čitalec. Največkrat jih lahko le beremo. Pasivne značke ne morejo začeti komunikacije s čitalcem. Pasivne značke z baterijo se od aktivnih razlikujejo po tem, da začnejo pošiljati identifikacijo takrat, ko so v bližini čitalca RFID. Lahko so priključene na

kakšne senzorje. Značke, ki imajo lastno napajanje, lahko beremo z večje razdalje kot značke brez lastnega napajanja [3].

### *Branje vsebine pasivne značke*

Čitalec konstantno oddaja sinusni val po radijsko-frekvenčnih valovih in preverja, če je prišlo do modulacije signala – modulacija signala bi pomenila prisotnost značke. Ko je značka v dometu čitalca, se v navitju (anteni, tuljavi) na znački inducira napetost. To napetost vezje v znački usmeri in uporabi za napajanje. Ko prejme dovolj energije za normalno delovanje, začne značka generirati urin signal, ki je običajno za nek faktor počasnejši od signala, ki ga oddaja čitalec. Vsako urino periodo se števec, ki naslavlja vrstice in stolpce v pomnilniku značke, poveča. Bit, ki je na izhodu pomnilnika se pošlje v modulacijsko enoto, ki je povezana z izhodnim tranzistorjem, povezanim na anteno. Tranzistor odvaja tok tuljavi, to pa se izrazi kot sprememba amplitude signala, ki ga oddaja čitalec. Čitalec zaznava spremembe v amplitudi in jih obdeluje v skladu s kodiranjem podatkov in metodami modulacije, ki jih uporablja [4]. Shemo vezja pasivne značke lahko vidimo na sliki 2.




*Slika 2: Shema vezja pasivne značke (Vir: [4])*

Najbolj uporabljena kodiranja podatkov v tehnologiji RFID so NRZ, Manchester in Biphase kodiranje, modulacije pa ASK, FSK in PSK modulacija [4].

### *Uporaba tehnologije RFID*

Tehnologija RFID je uporabljena za različne namene:

- za kontrolo dostopa,
- za sledenje dobrinam,
- za sledenje ljudem in živalim,
- za brezkontaktno zaračunavanje,
- kot medij za podajanje informacij (muzeji, knjižnice),
- za meritve v športu,
- in druge namene.

# 3 Zaslon na dotik

Zaslon na dotik je elektronski pripomoček, ki prikazuje informacije na zaslon in lahko zazna dotik in položaj dotika na zaslonu. Izraz dotik splošno pomeni dotik zaslona s prstom oziroma dlanjo, čeprav lahko zaslon na dotik zazna tudi ostale objekte, kot so elektronsko pisalo (ang. *stylus*) (paličica, ki služi za natančnejši dotik zaslona) [5].

Dve glavni značilnosti zaslonov na dotik:

- uporabniku omogoča neposredno interakcijo s prikazano informacijo (posredna bi bila uporaba miške),
- omogoča interakcijo z informacijo brez uporabe vmesnih naprav, ki bi jih bilo treba držati v roki – izjema je elektronsko pisalo.

Poznamo več vrst zaslonov na dotik, ki jih delimo glede na različne metode zaznavanja dotika. Nekateri izmed njih so:

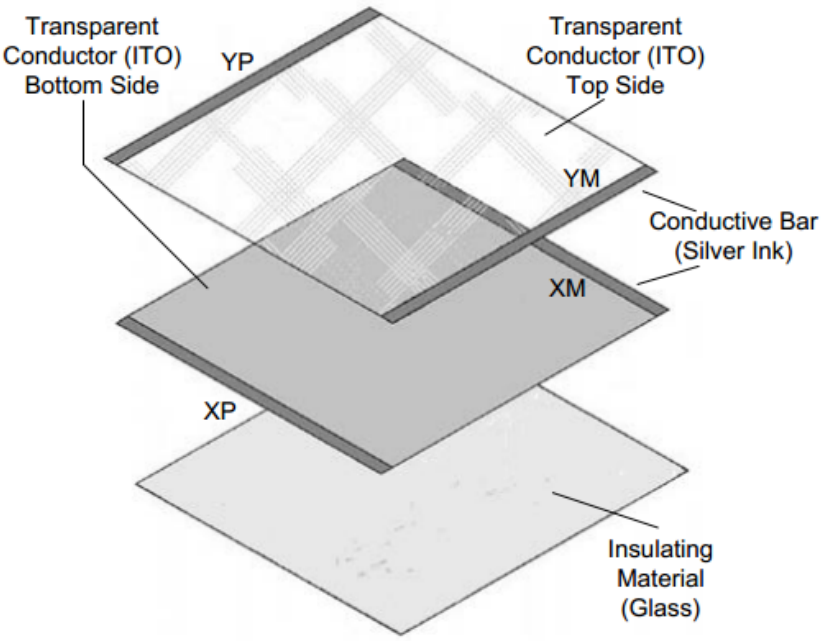
- rezistivni,
- kapacitivni,
- zasloni s projicirano kapacitivnostjo,
- zasloni s površinskimi zvočnimi valovi,
- optični zasloni na dotik,
- zasloni z IR (infrardečo) mrežo.

V zadnjih letih smo priča porastu uporabe zaslonov na dotik. Najdemo jih v mobilnih telefonih, v tabličnih računalnikih, v informacijskih napravah (interaktivni vodiči v muzejih, galerijah), v navigacijskih napravah, v avtomobilski industriji in nenazadnje tudi v industriji bele tehnike.

## *3.1 Rezistivni zasloni na dotik*

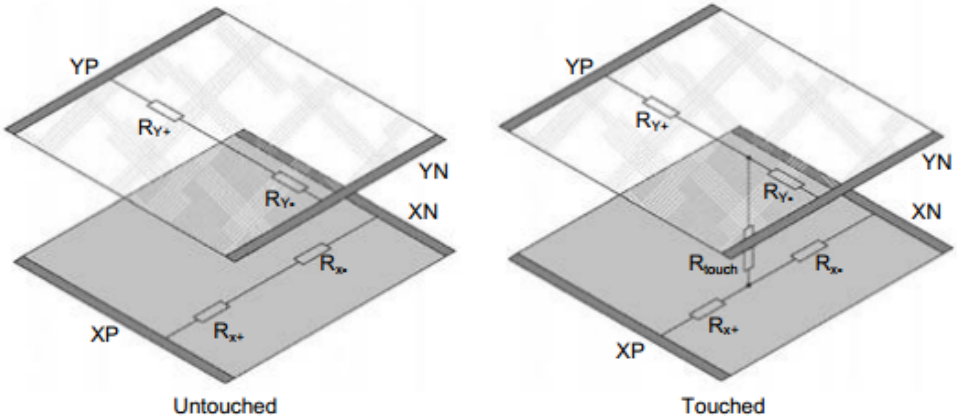
Rezistivni zasloni na dotik so sestavljeni iz zaslona, ki prikazuje informacije, in plošč z elektroniko, ki pretvori informacijo o dotiku v numerični zapis.

Spodnja plošča, ki prekriva zaslon, je običajno izolator iz stekla ali iz akrila. Na to ploščo sta položeni še dve upogljivi plošči iz stekla ali akrila, ki sta na notranjih straneh prevlečeni z električno prevodno in rezistivno plastjo iz indij-kositrovega oksida (ang. *ITO*). Ti dve plasti sta med seboj ločeni s prozornimi, nevidnimi distančniki (ang. *spacers*), ki preprečujejo, da bi se plošči nehote staknili in dali lažno zaznavanje dotika [6].




*Slika 3: Postavitev elektrod na prevodnih plasteh pri zaslonu na dotik (Vir: [7])*

Plošči, prevlečeni s prevodno in rezistivno plastjo, ob stiku začneta prevajati električni tok (slika 4). Vsaka od teh dveh plasti ima po dve elektrodi, ki sta postavljeni v nasprotnih stranicah plošče na prevodni plasti. Ko položimo tanki plošči eno na drugo, štiri elektrode tvorijo pravokotniku podoben lik; elektrode se ne prekrivajo, kar je vidno tudi na sliki 3 [8]. Elektrode so povezane s krmilnikom za zaznavo dotika, ki elektrodam na eni plasti dovaja napetost, na drugi plasti pa na eni od elektrod hkrati meri napetost, ki nastane ob stiku plasti.

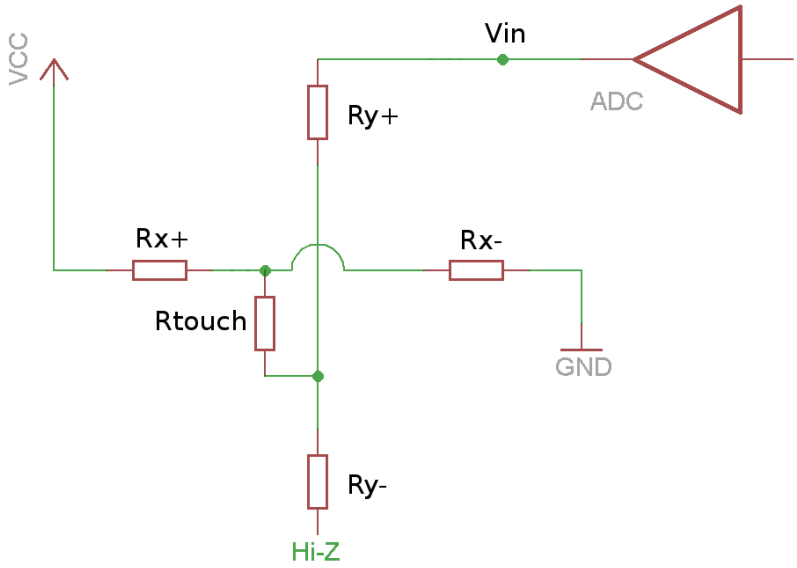



*Slika 4: Stik plasti ob dotiku zaslona na dotik (Vir: [7])*

Krmilnik za zaznavo dotika deluje v splošnem kot analogno/digitalni pretvornik s stikali. S stikali krmilnik preklaplja med funkcijo, ki jo opravlja elektroda: krmilnik lahko na elektrodo dovede napetost ali pa elektrodo poveže z analogno/digitalnim pretvornikom.

Merjenje pozicije dotika poteka tako, da krmilnik eni plasti dovede napetost (eni elektrodi dovede napetost, drugi pa maso), na drugi plasti pa eno elektrodo odklopi, na drugi elektrodi pa meri napetost, ki se pojavi ob stiku plasti, ki je posledica dotika. Napetost, ki jo merimo, se spreminja glede na pozicijo dotika zaradi rezistivne plasti; kontakt, ki je posledica dotika, ustvari napetostni delilnik v tej točki, tako da lahko odčitamo koordinato dotika. Meritev izvajamo za vsako os posebej.

Za merjenje koordinate X krmilnik elektrodi X+ dovede napetost Vcc, elektrodi X- pa maso. Ob stiku prevodnih plasti se v točki dotika ustvari napetostni delilnik, katerega napetost krmilnik prebere na vhodu analogno/digitalnega pretvornika, ki je povezan z elektrodo Y+ (slika 5). Elektroda Y- je med merjenjem koordinate X odklopljena. Ista logika se uporablja za merjenje koordinate Y. Na elektrodi Y+ in Y- dovedemo napetost in maso, napetost ob kontaktu pa beremo na koordinati X+ [9].

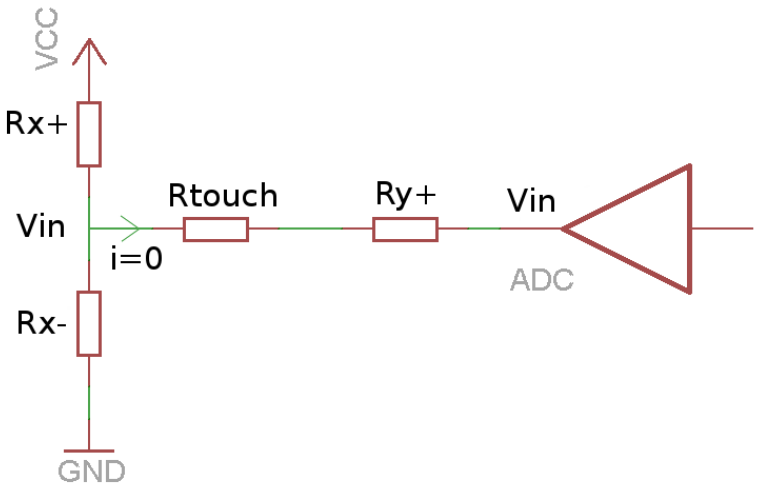



*Slika 5: Prikaz branja koordinate X ob stiku prevodnih plošč*

Zaradi predpostavke, da je vhodna upornost A/D pretvornika neskončna, je tok, ki teče skozi upora Ry+ in Rtouch izredno majhen, zato lahko zanemarimo padec napetosti na uporih Ry+ in Rtouch [7] (slika 6). Napetost Vin, ki jo izmerimo na vhodu A/D pretvornika, je enaka napetosti napetostnega delilnika (enačba 1). Koordinata X je proporcionalna izmerjeni napetosti Vin. Isto velja v primeru za koordinato Y.

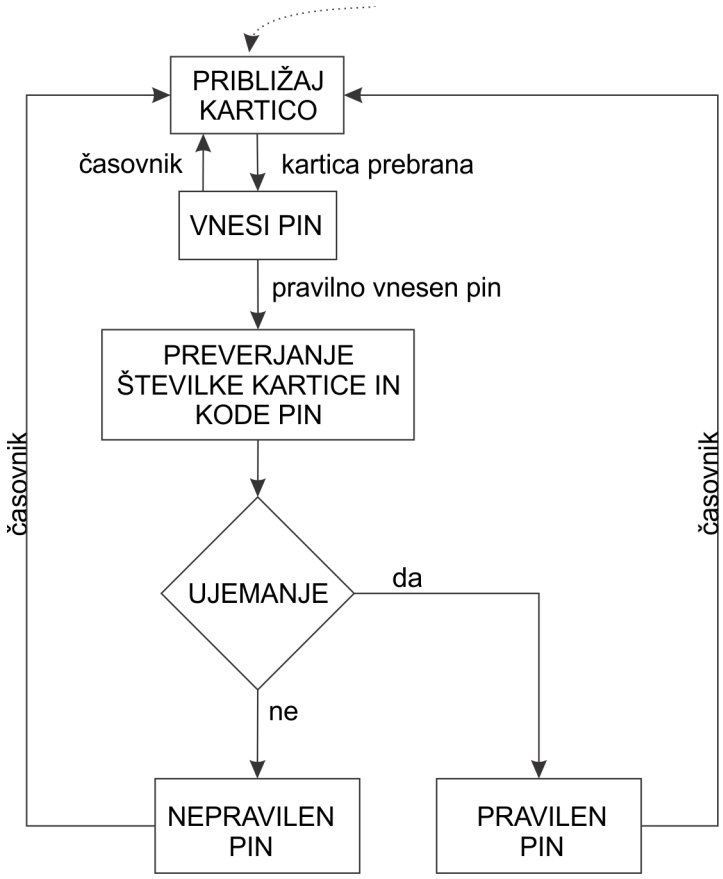
$$Vin = Vcc \times \frac{Rx-}{Rx+ + Rx-} \tag{1}$$

Na sliki 6 je prikazana električna shema za branje koordinate X, ki je ekvivalentna shemi na sliki 5.




*Slika 6: Prikaz branja koordinate X ob stiku prevodnih plošč, drugič*

# 4 Sistem za kontrolo dostopa

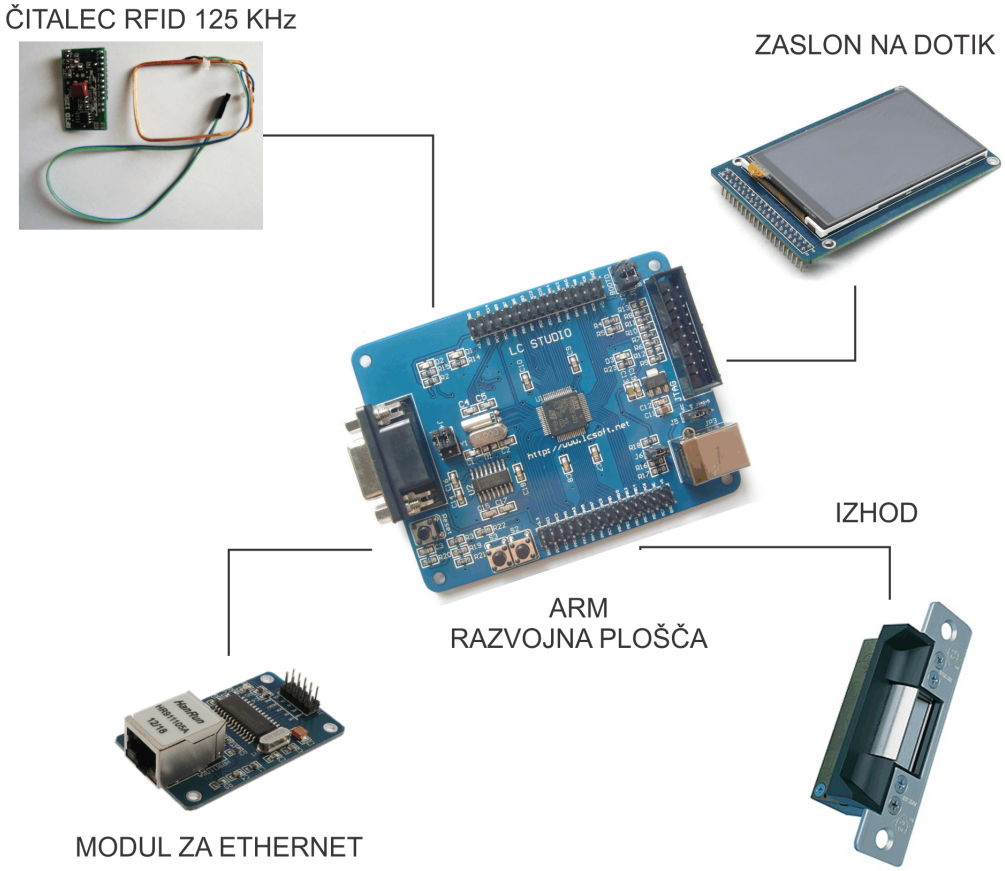



*Slika 7: Shema delovanja vgrajenega sistema*

Gledano skozi oči uporabnika je sistem za kontrolo dostopa preprosta naprava. Ob začetku uporabe naprava na zaslonu obvesti uporabnika, naj približa brezkontaktno kartico. Dokler uporabnik ne približa brezkontaktne kartice k anteni čitalca, se ne zgodi popolnoma nič. S približanjem kartice naprava pokaže obvestilo za vnos kode PIN. Takrat je številka kartice že prebrana. Naprava nato prikaže številčnico. Uporabnik ima za vnos kode PIN omejen čas, po preteku le-tega se naprava vrne v začetno stanje. Ob pravilno vneseni kodi PIN in potrditvi se na zaslonu izpiše, da je vstop dovoljen ali da je vnesena koda PIN napačna. Po določenem času se naprava vrne v začetno stanje. Delovanje naprave je prikazano na sliki 7, kjer ujemanje pomeni, da se številka kartice in koda PIN ujemata z zapisom v podatkovni bazi strežnika, časovnik pa označuje spremembo stanja naprave zaradi preteka časa, ki je namenjen posameznemu stanju.

## *4.1* ***Strojna oprema***

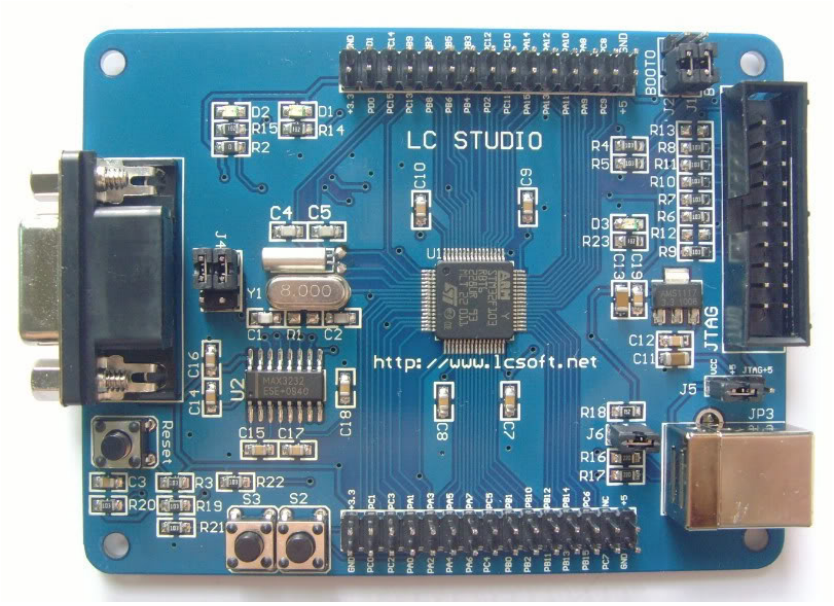
Za izdelavo naprave smo uporabili razvojno ploščo ARM proizvajalca STMicroelectronics s krmilnikom STM32F103RB. Nanjo smo priključili modul za zaslon na dotik, modul za Ethernet in čitalec RFID. V nadaljevanju bodo predstavljene posamezne komponente, ki sestavljajo vgrajeni sistem (slika 8). Povezave modulov z mikrokrmilnikom lahko vidimo na sliki 60 v dodatku B  Shema povezav.




*Slika 8: Moduli, ki sestavljajo vgrajeni sistem*

### 4.1.1 **Razvojna plošča, mikrokrmilnik STM32F103RB**

Za izdelavo sistema kontrole pristopa smo izbrali razvojno ploščo podjetja LC STUDIO, ki je prikazana na sliki 9. Srce razvojne plošče je mikrokrmilnik STM32F103RB proizvajalca STMicroelectronics in spada v serijo ARM Cortex-M3. Po zmogljivostih ga uvrščamo v srednji zmogljivostni razred serije STM32F103.




*Slika 9: Razvojna plošča LC STUDIO z mikrokrmilnikom serije Cortex-M3*

Značilnosti mikrokrmilnika STM32F103RB [10]:

- jedro: 32-bit ARM Cortex-M3, frekvenca 72 MHz,
- pomnilnik Flash: 128 kB,
- pomnilnik SRAM (ang. *static random-access memory*): 20 kB,
- 2x vmesnik SPI (ang. *Serial Peripheral Interface*),
- 3x vmesnik USART (ang. *universal synchronous asynchronous receiver transmitter*),
- 2x vmesnik $I^2C$ (ang. *Inter-integrated circuit*),
- 1x vmesnik USB 2.0 (ang. *Universal Serial Bus*),
- 1x vmesnik CAN (ang. *Controller area network*),
- 2x 12-bitni A/D (analogno/digitalni) pretvornik, 16-kanalni,

- 51x splošno-namenskih vhodov/izhodov (GPIO),
- 4x 16-bitni časovnik.

Na razvojni plošči najdemo:

- 1x priključek USART,
- 2x gumb, 1x reset gumb,
- 1x priključek JTAG (ang. *Joint Test Action Group*) – za programiranje mikrokrmilnika,
- 1x priključek USB – za komunikacijo z računalnikom in za napajanje,
- pini za splošno-namenske vhode/izhode,
- 2x 3,3 V pin, 2x 5 V pin, 4x masa GND (ang. *ground*).

### *Prekinitveni krmilnik NVIC*

Cortex-M3 procesorji imajo vgrajen prekinitveni krmilnik NVIC (ang. *Nested Vectored Interrupt Controller*), ki podpira naslednje funkcije: gnezdenje prekinitev, vektorske prekinitve, dinamično spreminjanje prioritet prekinitvam, maskiranje prekinitev. Vse prekinitve v mikrokrmilniku upravlja krmilnik NVIC [11].

Mikrokrmilnik, ki smo ga uporabili, ima 68 prekinitvenih vhodov (tukaj ni vštetih 16 prekinitvenih linij procesorja). Prekinitvenim virom lahko nastavimo 16 različnih prioritet [10].

### *Vodili AHB in APB*

Osnovni cilj preprostih vodil je, da dovolijo programski opremi, ki teče na centralni procesni enoti, komunikacijo z napravami v vgrajenem sistemu [12].

AHB (ang. *Advanced High-performance Bus*) je visoko zmogljivo vodilo, ki omogoča, da imamo več gospodarjev vodila, kar omogoča več hkratnih prenosov med različnimi gospodarji in sužnji. Gospodarji na vodilu AHB so vodilo DCode, sistemsko vodilo, vodilo DMA1, sužnji pa mostički AHB-APB, SRAM, FSMC (ang. *Flexible Static Memory Controller*) in FLITF (ang. *Flash memory interface*) [13].

APB (ang. *Advanced Peripheral Bus*) je preprosto vodilo namenjeno priključevanju relativno počasnih naprav. Na vodilu je samo en gospodar – mostiček AHB-APB, zato je v nekem trenutku omogočen le en prenos po vodilu. Mostička AHB2-APB1 in AHB2-APB2 povezujeta vodili APB1 in APB2 z vodilom AHB. Sužnji na vodilih so naprave (vmesniki SPI, naprave USART, splošno-namenska vhodno izhodna vrata GPIO …) [13].

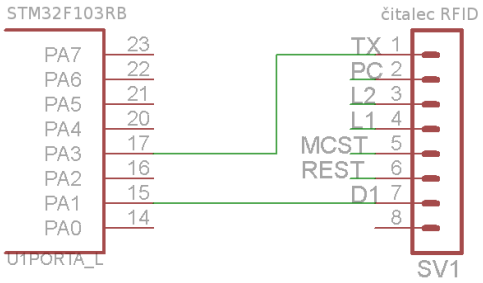
Vodilo APB1 deluje s frekvenco ure 36 MHz, vodili APB2 in AHB pa delujeta s polno hitrostjo 72 MHz [13].

Preden uporabimo katerokoli napravo v mikrokrmilniku, ji moramo najprej omogočiti uro za delovanje. Ob zagonu in ponovnem zagonu so ure vseh naprav onemogočene z namenom varčevanja porabe energije [13].

### 4.1.2 Čitalec RFID

Za branje kode RFID značk smo uporabili 125 KHz mini-modul, ki smo ga kupili na eBayu. Z izjemo osnovnih informacij o delovanju modula, ki so bile v opisu izdelka, informacij o modulu nismo našli. Čitalec bere številke kartic, ki so združljive s 125 KHz/134,2 KHz brezkontaktnimi karticami in podpira kartice serij EM4100, TK4100 in EM4102.

Čitalec (slika 11) nam številko kartice posreduje preko treh izhodov (slika 10), od tega sta dva namenjena priklopu na serijski vmesnik RS232. Izhod TX lahko priklopimo neposredno na mikrokrmilnik, izhod PC pa je namenjen neposrednemu priklopu na računalniški serijski vmesnik RS232, saj ima vgrajeno vezje za pretvorbo napetostnih nivojev. Na tretjem izhodu MCST čitalec pošilja številko kartice v Manchester zapisu.

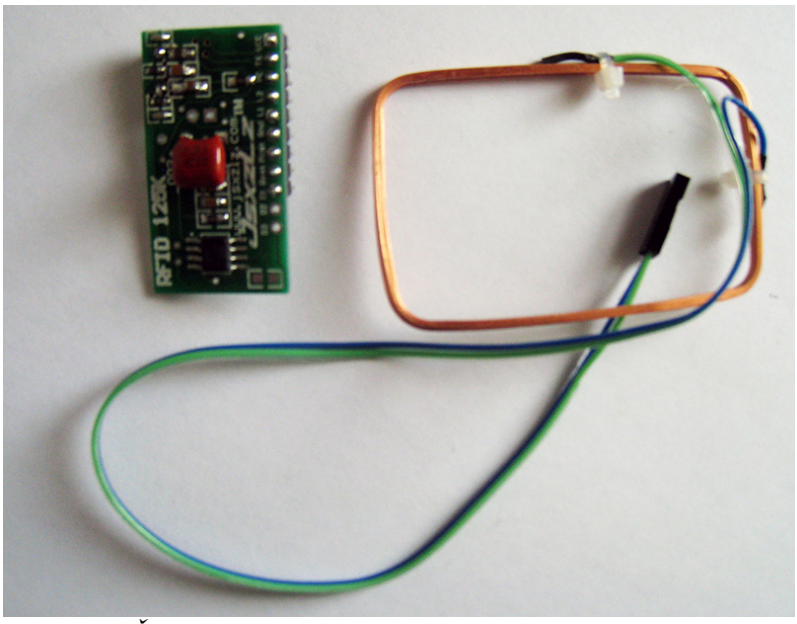



*Slika 10: Shema vezave čitalca RFID z mikrokrmilnikom*

Uporabili smo izhod za serijski vmesnik TX, ki deluje asinhronsko s hitrostjo 9600 baudov in pošilja 8 podatkovnih bitov in 1 stop bit. Priključili smo ga na vmesnik USART2, na pin 3 na vratih A (pin PA3).

Pina L1 in L2 sta namenjena priključitvi antene čitalca RFID, pin REST pa ima funkcijo reseta.

Na čitalcu je še izhod D1 (viden na sliki 10), ki označuje, da je čitalec zaznal kartico. Ko približamo kartico, se izhod D1 postavi v visoko stanje in po največ 6 ms se na izhodih pojavijo prebrani podatki. Po koncu pošiljanja kode kartice se D1 postavi nazaj v nizko stanje. D1 smo priključili na pin 1 na vratih A (pin PA1).

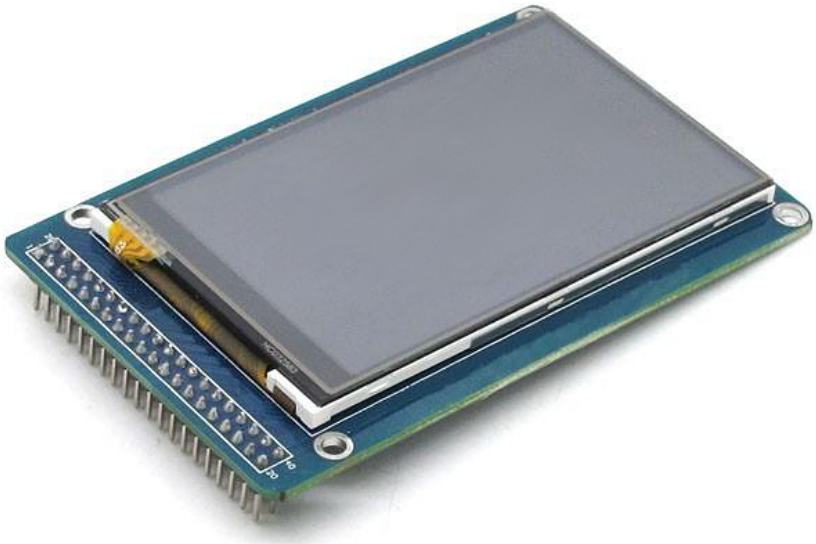



*Slika 11: Čitalec RFID - vezje in antena*

Modul deluje v napetostnem območju od 3 V do 5 V DC. Napajamo ga z napetostjo 3,3 V, ki jo dobimo na razvojni plošči.

### 4.1.3 Zaslon na dotik

Za prikazovanje informacij in zaznavanje dotikov smo uporabili modul TFT LCD (ang. *Thin film transistor liquid crystal display*) (slika 12). Tako kot čitalec RFID, smo tudi ta modul kupili na eBayu, proizvajalec je neznan. Podatke o krmilnikih na modulu smo dobili v opisu oglasa. Na modulu so prispajkani zaslon LCD velikosti 3,2 inča, prikazovalni krmilnik SSD1289 za zaslon LCD, krmilnik za zaznavanje dotikov (ang. *touch screen controller*) ADS7843 ter reža za spominske kartice SD.



*Slika 12: Modul z zaslonom na dotik*

Modul ima vgrajen napetostni regulator 3,3 V, s katerim so povezane vse naprave na modulu. Za delovanje napetostnega regulatorja moramo zagotoviti napetost 5 V, ki jo dobimo na razvojni plošči.

#### *4.1.3.1* ***Zaslon LCD in krmilnik za LCD SSD1289***

Lastnosti zaslona:

- velikost: 3,2 inča,
- ločljivost: 240 x 320 pikslov RGB.

Lastnosti krmilnika SSD1289 [14]:

- prikazovanje do: 262.144 barv pri ločljivosti 240 x 320 pikslov RGB (ang. *Red Green Blue*),
- velikost prikazovalnega (grafičnega) pomnilnika GDDRAM: 172.800 bajtov,
- širina vodila: 8- / 9- / 16- / 18-bitov.

S krmilnikom komuniciramo preko kontrolnih registrov in prikazovalnega pomnilnika.

S pisanjem v kontrolne registre upravljamo z nastavitvami krmilnika. Ob inicializaciji krmilnika je potrebno nastaviti registre za napajanje, izbrati način komunikacije z mikrokrmilnikom ter nastaviti še druge nastavitve [14].

Prikazovalni pomnilnik je pomnilnik, ki vsebuje informacijo o pikslih in barvah pikslov na zaslonu. Sestavljen je iz statičnega RAM-a velikosti $240 \times 320 \times 18/8 = 172.800$ bajtov. Vsakemu pikslu na zaslonu pripada pomnilniška beseda velikosti 18 bitov v prikazovalnem pomnilniku. Za spreminjanje vzorca na zaslonu (pisanje v prikazovalni pomnilnik) je potrebno najprej nastaviti ustrezne kontrolne registre za pozicijo piksla na zaslonu, nato pa zapisati 18- oziroma 16-bitno vrednost barve (razlika je prikazana na sliki 13), za katero želimo, da je prikazana. Krmilnik samodejno osvežuje zaslon s podatki iz prikazovalnega pomnilnika.

Za izris 262.144 barv RGB potrebujemo 18 bitov ( $262.144 = 2^{6+6+6}$ ). Na modulu je za podatkovne linije krmilnika namenjenih 16 pinov (LCD_DB0 do LCD_DB15), zato smo se odločili za širino vodila 16 bitov (izbrali bi lahko še 8 ali 9 bitov). Po 16-bitnem podatkovnem vodilu pa ne moremo naenkrat poslati 18 bitov, kolikor potrebujemo za 262.144 barv. Iz tega sledi, da moramo za polno barvno globino (262.144 barv) uporabiti podatkovno vodilo vsaj dvakrat. Prvič pošljemo 6 bitov za rdečo in 6 bitov za zeleno barvo, drugič pa še 6 bitov za modro barvo. Možno je uporabiti barvno globino 65.536 barv – v tem primeru naenkrat pošljemo 5 bitov za rdečo, 6 bitov za zeleno in 5 bitov za modro barvo, kar je ravno 16 bitov (slika 13). Ta način predstavitve barv se imenuje RGB565. Zaradi hitrejšega delovanja zaslona smo se odločili, da bomo uporabljali 65.536 barv.

| Podatkovni pini na modulu | LCD_15 | LCD_14 | LCD_13 | LCD_12 | LCD_11 | LCD_10 | LCD_9 | LCD_8 | | LCD_7 | LCD_6 | LCD_5 | LCD_4 | LCD_3 | LCD_2 | LCD_1 | LCD_0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Pini na krmilniku LCD | D17 | D16 | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| 18-bitni dostop | R5 | R4 | R3 | R2 | R1 | R0 | G5 | G4 | G3 | G2 | G1 | G0 | B5 | B4 | B3 | B2 | B1 | B0 |
| 16-bitni dostop | R4 | R3 | R2 | R1 | R0 | G5 | G4 | G3 | / | G2 | G1 | G0 | B4 | B3 | B2 | B1 | B0 | / |

*Slika 13: 18- in 16- bitni način zapisa barve piksla*

Na modulu je za priključitev mikrokrmilnika na SSD1289 namenjenih 21 pinov. Od tega je 16 pinov za podatkovno vodilo in 5 kontrolnih pinov, kar lahko vidimo na sliki 14.

Podatkovni pini (16): LCD_DB0 – LCD_DB15.
Kontrolni pini (5): LCD_RS, LCD_WR, LCD_RD, LCD_CS, LCD_RST.

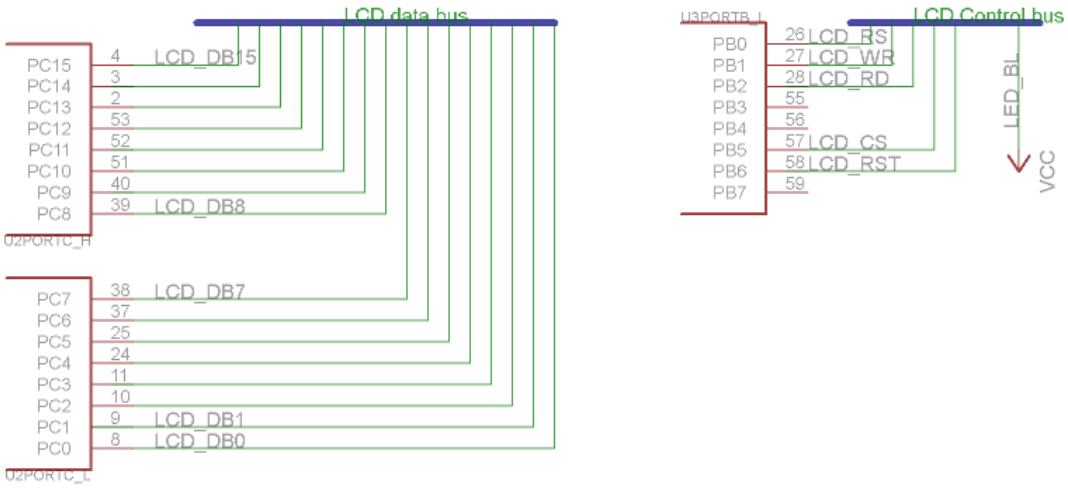
Signal LCD_CS (vhod) služi za izbiro naprave. Aktiven je v nizkem stanju, takrat krmilnik spremlja spremembe signalov na kontrolnih in podatkovnih pinih.

S signalom LCD_RS (vhod) (v tehnični dokumentaciji krmilnika [14] poimenovan kot D/C signal) izbiramo lokacijo, kjer krmilnik piše/bere. Ko je na vhodu visoko stanje, pišemo v prikazovalni pomnilnik, ob nizkem stanju pa v kontrolne registre.

S signalom LCD_WR (vhod) izbiramo branje ali pisanje. Ko je na vhodu nizko stanje, imamo pisanje v kontrolne registre oziroma prikazovalni pomnilnik, z visokim stanjem pa izberemo branje.

Signal LCD_RD (ang. *read strobe signal*) (vhod) uporabljamo pri bralnih ciklih. Aktiven je v nizkem stanju, z njim krmilniku ukažemo, da postavi podatke na podatkovno vodilo.

Signal LCD_RST (vhod) uporabljamo za reset krmilnika SSD1289. Aktiven je v nizkem stanju.




*Slika 14: Shema vezave krmilnika LCD in mikrokrmilnika*

Podatkovne pine krmilnika smo povezali z vrati C (PORTC) na mikrokrmilniku, kontrolne pine pa sledeče: LCD_RS na PB0, LCD_WR na PB1, LCD_RD na PB2, LCD_CS na PB5 in LCD_RST na PB6. Vezavo lahko vidimo na sliki 14.

#### *4.1.3.2* ***Krmilnik za zaznavo dotikov ADS7843***

Modul ima vgrajen krmilnik za zaznavo dotikov ADS7843 podjetja Texas Instruments [9]. Primeren je za priklop na rezistivne zaslone na dotik.

V osnovi krmilnik deluje kot analogno/digitalni pretvornik, ki ima vgrajena stikala, s katerimi nadzira katero os (X,Y) napaja in katero bere. Kadar bere os X, napaja os Y, in obratno.

Lastnosti [9]:

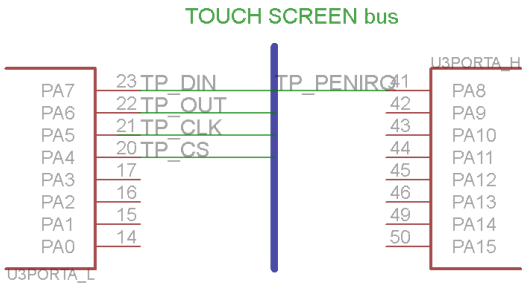
- vmesnik za komunikacijo: SPI,
- 12-bitni analogno/digitalni pretvornik,
- široko področje delovanja: 2,7 V do 5 V,
- majhna poraba električne energije: 750 μW pri hitrosti oddajanja 125 kHz.

Analogni vhod v A/D pretvornik poteka preko multiplekserja 4/1. Posebna konfiguracija stikal, ki imajo zanemarljivo upornost, omogoča, da neizbranemu kanalu analogno/digitalnega pretvornika krmilnik dovede napetost, izbrani kanal A/D pretvornika pa poveže z elektrodo na plasti zaslona, ki bere napetost, ki se pojavi ob kontaktu prevodnih plasti.

Za komunikacijo s krmilnikom uporabljamo vmesnik SPI. Krmilnik smo priključili na mikrokrmilnik na vmesnik SPI1. Krmilnik ob zaznanem dotiku zaslona postavi izhod za indikacijo prekinitve PENIRQ iz visokega v nizko stanje, takrat lahko pošljemo ukaz za branje X in Y vrednosti zaslona. Izhod PENIRQ je povezan na pin PA8 na mikrokrmilniku, vhodni signal TP_DIN za sprejemanje podatkov na pin PA7, izhodni signal TP_OUT za pošiljanje podatkov na pin PA6, vhodni signal TP_CLK za urin pulz na pin PA5 in vhodni signal TP_CS za izbiro naprave na pin PA4. Shema vezave je prikazana na sliki 15.

Krmilniku za branje koordinate X pošiljamo vrednost 0x90, za branje koordinate Y pa 0xD0. V tehničnih podatkih [9] krmilnika sta ukaza za branje koordinat ravno obratna; X beremo z ukazom 0xD0, Y pa z 0x90. Ker ima naš modul različno podane koordinatne osi v krmilniku za LCD in krmilniku za zaznavanje dotika zaslona, smo zaradi lažjega računanja koordinat zamenjali smeri koordinatnega sistema pri krmilniku za zaznavanje dotika zaslona.

Krmilnik v odgovor na ukaz pošilja nazaj 12-bitno vrednost iz A/D pretvornika.

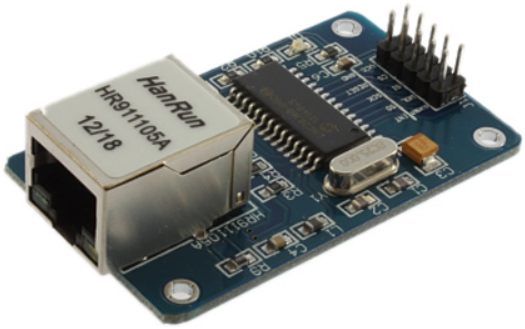



*Slika 15: Shema vezave krmilnika za zaznavanje dotikov z mikrokrmilnikom*

### 4.1.4 **Ethernet modul ENC28J60**

Za komunikacijo s strežnikom na računalniku smo uporabili krmilnik za Ethernet ENC28J60 (slika 16) [15]. Krmilnik služi kot mrežni vmesnik za vse mikrokrmilnike, ki imajo vgrajen vmesnik SPI.

Krmilnik opravlja naloge dveh plasti po modelu OSI (ang. *Open System Interconnection*) [16], linijske (povezovalne) in fizične plasti. Fizična plast skrbi za kodiranje in dekodiranje analognih informacij na liniji, linijska plast pa iz niza bitov, ki ga sprejme iz fizične plasti, sestavi okvirje, ki se pošljejo v omrežno plast. V obratni smeri pakete, ki jih dobi iz omrežne plasti, razstavi v nize bitov, ki jih pošlje fizični plasti. Linijska plast skrbi za kontrolo pretoka paketov ter zaznavanje in popravljanje napak (če je mogoče), ki se pojavijo v linijski plasti.



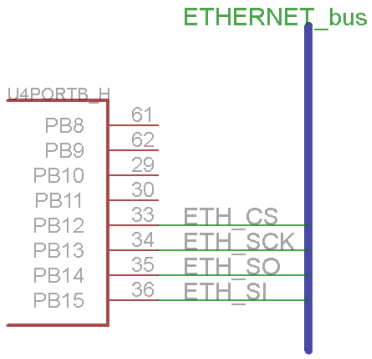

*Slika 16: Ethernet modul ENC28J60*

Značilnosti ENC28J60 [15]:

- komunikacija preko vmesnika SPI s hitrostjo do 10 Mb/s,
- fizična plast: 10BASE-T,
- 8-kB vmesni pomnilnik (ang. *buffer*) za vhodne in izhodne pakete,
- krožni FIFO (ang. *First In, First Out*) medpomnilnik na strojnem nivoju,
- podpira t. i. unicast, multicast in broadcast pakete,

- napetostni nivoji TTL (ang. *Transistor-Transistor Logic*),
- napetost 3,3 V.

Modul smo priključili na vmesnik SPI2, ki zaseda pine PB12, PB13, PB14 in PB15 na mikrokrmilniku. Na pin PB12 smo priključili vhodni signal za izbiro naprave ETH_CS, na pin PB13 vhodni signal za urin pulz ETH_SCK, na pin PB14 izhodni signal za pošiljanje podatkov ETH_SO in na pin PB15 vhodni signal za sprejemanje podatkov ETH_SI. Shema vezave je vidna na sliki 17.




*Slika 17: Shema vezave krmilnika Ethernet z mikrokrmilnikom*

## *4.2* ***Programska rešitev***

Za programiranje STM mikrokrmilnikov družine ARM je na voljo več razvojnih orodij: IAR EWARM [17], Keil MDK-ARM [18], Atollic TrueSTUDIO [19], Raisonance IDE Ride7 [20], Hitex HiTOP IDE/Debugger [21] in drugi.

Ker smo za programiranje mikrokrmilnika STM32F103RB že prej imeli na voljo Keil ULINK2 Debug Adapter [22] – JTAG programator in razhroščevalnik, smo izbrali razvojno orodje Keil MDK-Lite [18]. Lite verzija orodja je na voljo brezplačno, prihaja pa z določenimi omejitvami – razhroščevalnik, simulator, C/C++ prevajalnik in povezovalnik lahko uporabljamo le do velikosti izvorne kode 32 KB.

Da bi se izognili omejitvam MDK-Lite verzije, smo se odločili za uporabo brezplačnih razvojnih programov GNU, ki jih dobimo v paketu Sourcery CodeBench Lite Edition [23] – v tem primeru Lite verzija pomeni, da smo prikrajšani za tehnično podporo, optimizirane knjižnice, simulator, razhroščevalnik in razvojno orodje Sourcery CodeBench [24].

Uporabili smo:

- razvojno orodje: Keil MDK-ARM Lite – µVision 4.60.0.0 [25],
- prevajalnik, povezovalnik: Sourcery CodeBench Lite 2012.03-56, gcc version 4.6.3,
- adapter JTAG: Keil ULINK2.

### 4.2.1 Programiranje mikrokrmilnika

Pri programiranju smo si pomagali s knjižnicami [26] proizvajalca STMicroelectronics. Začeli smo z že pripravljenim začetnim projektom, ki vsebuje nastavitve za osnovno delovanje mikrokrmilnika, kot so: ustrezna nastavitev frekvence sistemske ure in delilnikov (ang. *prescalers*) za urin signal, nastavitev skladovnega kazalca, nastavitev lokacije tabele prekinitvenih vektorjev, ponastavitev vseh čakajočih prekinitev ...

Program se začne z inicializacijo sistemskih nastavitev, kjer nastavimo uro za mikrokrmilnik, delilnike za uro za vodilo AHB in pomnilnik Flash. Te začetne nastavitve smo povzeli po začetnem projektu.

Nato sledi inicializacija naprav v mikrokrmilniku in modulov, ki so priključeni na mikrokrmilnik: krmilnik LCD, krmilnik za zaznavanje dotika, čitalec kartic RFID, krmilnik za Ethernet. Po tem nastavimo še začetne nastavitve za končni avtomat, ki upravlja sistem.

V sklopu začetnih nastavitev program pregleda stanje na pinu PB11. Ta pin smo nastavili kot vhod z vključenim notranjim »pull-up« uporom. Če je prebrano nizko stanje, preberemo nastavitve kalibracije dotikov zaslona iz pomnilnika Flash v pomnilnik RAM, ob visokem stanju pa izvedemo kalibracijsko rutino in rezultate shranimo v pomnilnik Flash. Privzeto je pin PB11 vezan na maso.

Po tem nastavimo prekinitev za DMA (ang. *Direct Memory Access*), prekinitev ob spremembi nivoja na vhodnem pinu PA1, ki je priključen na izhod D1 na RFID čitalcu, in prekinitev časovnika.

Na koncu inicializiramo še sklad uIP. Ko so vse inicializacije končane, začnemo izvajati glavno zanko, kjer se izvaja končni avtomat in podprogram za Ethernet.

### 4.2.2 Konfiguracija krmilnika za LCD SSD1289

Kot smo že prej omenili, so ure vseh naprav ob zagonu privzeto onemogočene, zato je bilo najprej treba omogočiti uro za splošno-namenska vrata B (GPIOB) in C (GPIOC). Vrati B in C sta povezani z vodilom APB2, potrebno je bilo uporabiti ukaz RCC_APB2PeriphClockCmd (slika 18).

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC | RCC_APB2Periph_GPIOB,
ENABLE);
```

*Slika 18: Vklop ure za splošno-namenska vrata B in C*

Z omogočeno uro smo lahko začeli nastavitev naprave: vrat. V kodi (slika 19) smo nastavili pine kot izhodne pine. V pomoč so nam bile knjižnice proizvajalca mikrokrmilnika [26]. Najprej je bilo treba deklarirati strukturo GPIO_InitStructure in ji ustrezno nastaviti elemente. V GPIO_Pin smo zapisali, katere pine želimo nastaviti, z GPIO_Speed_50MHz pa smo nastavili, da se vrednost pina spreminja s hitrostjo 50 MHz. Element GPIO_Mode hrani podatek o funkciji pina. V tem primeru smo nastavili pine za izhodne s parametrom GPIO_Mode_Out_PP, ki nastavi pin kot »Push-pull output«. Nastavitev naprave smo izvedli z ukazom GPIO_Init, ki mu kot parameter podamo vrata, ki jih želimo spremeniti, in strukturo, ki hrani nastavitve.

```
GPIO_InitTypeDef GPIO_InitStructure;

GPIO_InitStructure.GPIO_Pin =      GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 |
                                   GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5 |
                                   GPIO_Pin_6 | GPIO_Pin_7 | GPIO_Pin_8 |
                                   GPIO_Pin_9 | GPIO_Pin_10 | GPIO_Pin_11 |
                                   GPIO_Pin_12 | GPIO_Pin_13 | GPIO_Pin_14|
                                   GPIO_Pin_15;

GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;

GPIO_Init(GPIOC, &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Pin =      CtrlPin_RS | CtrlPin_NWR | CtrlPin_RD |
                                   CtrlPin_NCS | CtrlPin_RST;

GPIO_Init(GPIOB, &GPIO_InitStructure);
```

*Slika 19: Konfiguriranje pinov za krmilnik za LCD*

Uporabili smo celotna vrata C in nastavili pine za izhode – »Push-pull output«. Vrata C so uporabljena za podatkovno vodilo krmilnika za zaslon. Kontrolne signale za krmilnik za zaslon smo generirali s pini na vratih B.

### *Pisanje v kontrolne registre*

Krmilnik upravljamo s pisanjem v kontrolne registre. Za ta namen smo napisali funkcijo (slika 20), ki nastavi kontrolne signale in pošlje podatke po podatkovnem vodilu. Clr_ in ime signala pomeni, da postavimo signal v nizko stanje, Set_ in ime signala pa postavi signal v visoko stanje. Cs ustreza LCD_CS, Rs LCD_RS, Rd LCD_RD in Wr LCD_RW. LCD_Reg označuje naslov v krmilniku, možno je nastaviti registre od 0x00 do 0x4F. V RegValue zapišemo vrednost, ki jo želimo zapisati na naslov v krmilniku. V kodi je uporabljen ukaz GPIO_Write, ki podatke, ki jih želimo poslati, pošlje na vrata C, ki predstavljajo podatkovno vodilo.

```
void LCD_WriteReg(u8 LCD_Reg, u16 LCD_RegValue)

{

      Clr_Cs;

      Clr_Rs;

      Set_Rd;

      GPIO_Write(GPIOC, LCD_Reg);

      Clr_Wr;

      Set_Wr;

      Set_Rs;

      GPIO_Write(GPIOC, LCD_RegValue);
```

```
      Clr_Wr;

      Set_Wr;

      Set_Cs;

}
```

*Slika 20: LCD – funkcija za pisanje v kontrolni register*

S pripravljeno funkcijo za pisanje v kontrolne registre krmilnika lahko začnemo inicializacijo krmilnika. Potrebno je nastaviti nastavitve za napajanje, frekvence osveževanja zaslona, nastaviti barvno globino in način pošiljanja, nastaviti registre za prilagoditev game ter druge. Inicializacije ne bomo podrobneje predstavili, je pa razvidna iz programske kode na sliki 59.

### *Pisanje v prikazovalni pomnilnik*

Poleg pisanja v kontrolne registre poznamo tudi pisanje v prikazovalni pomnilnik, v katerem je za vsak piksel na zaslonu zapisana barva piksla. Pisanje v prikazovalni pomnilnik začnemo (slika 21) s pisanjem v kontrolni register 0x22, s katerim krmilniku povemo, da hočemo pisati v prikazovalni pomnilnik. To naredimo tako, da postavimo signala CS in RS v nizko stanje, signal RD v visoko stanje, na podatkovne linije zapišemo število 0x22 in postavimo signal WR v nizko stanje, s tem označimo, da gre za pisalni dostop. Nato signala WR in RS postavimo nazaj v visoko stanje.

```
void LCD_WriteRAM_Prepare(void)

{

      Clr_Cs;

      Clr_Rs;

      Set_Rd;

      GPIO_Write(GPIOC, 0x22);

      Clr_Wr;

      Set_Wr;

      Set_Rs;

      /* tu se koda nadaljuje z LCD_WriteRAM */

}
```

*Slika 21: Začetna funkcija za zapisovanje v prikazovalni pomnilnik*

Zgornjo funkcijo (slika 21) moramo vedno uporabiti pred prvim zapisom v prikazovalni pomnilnik.

Nato uporabljamo funkcijo LCD_WriteRAM (slika 22), ki zapiše vrednost barve v naslov v prikazovalnem pomnilniku, ki je nastavljen v registrih za koordinati X in Y. Za zapis v

prikazovalni pomnilnik postavimo na podatkovne linije barvno vrednost piksla in s spremembo signala WR najprej v nizko, nato v visoko stanje izvedemo zapis. To funkcijo lahko uporabimo več kot enkrat – pri zapisu sosednjih pikslov, saj krmilnik samodejno poveča naslov v pomnilniku za velikost enega piksla po vsakem pisnem dostopu.

```
void LCD_WriteRAM(u16 RGB_Code)
{
1     /* za prvo pisanje je treba uporabiti LCD_WriteRAM_Prepare */
      GPIO_Write(GPIOC, RGB_Code);
      Clr_Wr;
      Set_Wr;
      /* za zadnje pisanje uporabi LCD_WriteRAM_Close */
}
```

*Slika 22: Funkcija za pisanje v prikazovalni pomnilnik*

Pisanje v prikazovalni pomnilnik zaključimo s postavitvijo signala CS v visoko stanje (slika 23).

```
void LCD_WriteRAM_Close(void)
{
      Set_Cs;
}
```

*Slika 23: Funkcijo za zaključitev pisanja v prikazovalni pomnilnik*

Poleg funkcije za pisanje v prikazovalni pomnilnik je verjetno najbolj uporabna funkcija za nastavljanje naslova v pomnilniku (slika 24), ki vpliva na koordinati X in Y piksla, v katerega pišemo. Koordinato X nastavimo s pisanjem vrednosti (0–239) v kontrolni register 0x4e, koordinato Y(0–319) pa s pisanjem v register 0x4f.

```
void LCD_SetCursor(u8 Xpos, u16 Ypos)
{
  LCD_WriteReg(0x4e, Xpos);
  LCD_WriteReg(0x4f, Ypos);
}
```

*Slika 24: Funkcija za nastavitev naslova v pomnilniku*

Funkcije za zapis podatkov v kontrolne registre in prikazovalni pomnilnik predstavljajo poln nabor funkcij za delo s krmilnikom. Pri izrisovanju vsebine na zaslon smo si pomagali s funkcijami za izris pravokotnikov, krogov, črt, ki smo jih napisali na podlagi znanih algoritmov.

### 4.2.3 Konfiguracija krmilnika za zaznavanje dotika ADS7843

Najprej smo omogočili uro vratom A (slika 25). Tako kot vrata B in C, so vrata A na vodilu APB2.

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
```

*Slika 25: Vklop ure za splošno-namenska vrata A*

Sledila je nastavitev pinov.

```
GPIO_InitTypeDef GPIO_InitStructure;
/* Configure SPI1 pins: SCK, MISO and MOSI */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5  | GPIO_Pin_6 | GPIO_Pin_7;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);
/* TP_CS */
GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_4;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);
/* TP_IRQ */
GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_8 ;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU ;
```

```
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

GPIO_Init(GPIOA, &GPIO_InitStructure);
```

*Slika 26: Konfiguracija pinov krmilnika za zaznavanje dotika*

Pinom PA5, PA6 in PA7 smo s parametrom GPIO_Mode_AF_PP nastavili funkcijo »Alternate function Push-pull output« (slika 26). S tem smo pine odklopili od splošno-namenskih vrat in jih priključili na signal naprave SPI1, ki si z vrati deli pine.

PA5 je priklopljen na pin TP_CLK na krmilniku za zaznavanje dotika, PA6 na TP_OUT in PA7 na pin TP_DIN.

V inicializaciji smo nastavili še izhodni pin PA4, ki je priključen na pin TP_CS, s katerim izbiramo napravo.

Pin PA8 nastavimo kot vhod z vključenim »pull-up« uporom. Ta pin je priključen na signal TP_PENIRQ na krmilniku za zaznavanje dotika, ki je v visokem stanju, ko ni zaznanega dotika, ob stiku plasti pa gre v nizko stanje. Kadar je PA8 v nizkem stanju, lahko prožimo poizvedovanje po vrednostih analogno/digitalnega pretvornika.

Nato smo omogočili uro za vmesnik SPI1 (slika 27).

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_SPI1, ENABLE);
```

*Slika 27: Vklop ure napravi SPI1*

Sledila je nastavitev naprave SPI1.

```
SPI_InitTypeDef  SPI_InitStructure;

/* DISABLE SPI1 */

SPI_Cmd(SPI1, DISABLE);

/* SPI1 Config */

SPI_InitStructure.SPI_Direction = SPI_Direction_2Lines_FullDuplex;

SPI_InitStructure.SPI_Mode = SPI_Mode_Master;

SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b;

SPI_InitStructure.SPI_CPOL = SPI_CPOL_Low;

SPI_InitStructure.SPI_CPHA = SPI_CPHA_1Edge;

SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_256;

SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB;

SPI_Init(SPI1, &SPI_InitStructure);
```

```
/* Enable SPI1 */
SPI_Cmd(SPI1, ENABLE);
```

*Slika 28: Nastavitev naprave SPI1*

Podobno kot za konfiguriranje splošno-namenskih vhodov/izhodov, obstaja knjižnica tudi za vmesnik SPI. S SPI_Mode_Master smo nastavili SPI1 funkcijo gospodarja vodila (slika 28). S SPI_DataSize je bilo nastavljeno, da vmesnik SPI1 pošilja po 8 bitov naenkrat. S SPI_CPOL in SPI_CPHA nastavljamo fazo ure in polariteto ure. Ti dve nastavitvi sta pomembni, saj z njima nastavljamo, kdaj naprave zajemajo podatke na vodilu. Za spreminjanje nastavitev ure smo morali uro napravi SPI1 predhodno onemogočiti. S kombinacijo SPI_CPOL_Low in SPI_CPHA_1Edge smo nastavili, da se zajemanje podatkov proži ob prehodu ure iz nizkega stanja v visoko stanje. S SPI_BaudRatePrescaler_256 smo vključili delilnik za uro, ki uro upočasni na 281.250 Hz. Zadnji parameter, ki ga nastavimo, je SPI_FirstBit_MSB, s katerim nastavimo, da je prvi bit najpomembnejši bit (MSB). Na koncu omogočimo uro.

```
static void WR_CMD (uint8_t cmd)
{
/* Wait for SPI1 Tx buffer empty */
while (SPI_I2S_GetFlagStatus(SPI1, SPI_I2S_FLAG_TXE) == RESET);
/* Send SPI1 data */
SPI_I2S_SendData(SPI1,cmd);
/* Wait for SPI1 data reception */
while (SPI_I2S_GetFlagStatus(SPI1, SPI_I2S_FLAG_RXNE) == RESET);
/* Read SPI1 received data */
SPI_I2S_ReceiveData(SPI1);
}
```

*Slika 29: Pošiljanje ukaza v krmilnik za zaznavanje dotika*

V funkciji WR_CMD (slika 29) je predstavljeno pošiljanje ukaza v krmilnik preko vmesnika SPI1. Najprej je treba počakati, da je register, ki pošilja in sprejema podatke na vmesniku SPI1 prazen. To storimo s funkcijo SPI_I2S_GetFlagStatus, kjer preverimo zastavico TXE v statusnem registru. Funkcija vrne vrednost SET, ko je register za pošiljanje prazen. Nato pošljemo ukaz po vodilu in počakamo na zastavico RXNE, ki označuje, da je register, ki sprejema podatke, poln.

V funkciji RD_AD (slika 30) preberemo podatke iz krmilnika za zaznavanje dotikov. Pred klicanjem te funkcije je treba uporabiti funkcijo WR_CMD, s katero pošljemo ukaz. V funkciji najprej pošljemo po vodilu same ničle. Nato počakamo, da krmilnik pošlje podatke, in jih shranimo v spremenljivko temp, kjer jih pomaknjene za 8 bitov v levo nato shranimo v spremenljivko buf. Še enkrat ponovimo pošiljanje ničel in preberemo podatke iz krmilnika v temp. Iz obeh branj sestavimo 12-bitno število, ki ga vrnemo kot rezultat funkcije.

```
static uint16_t RD_AD(void)
{
uint16_t buf,temp;
/* Wait for SPI1 Tx buffer empty */
while (SPI_I2S_GetFlagStatus(SPI1, SPI_I2S_FLAG_TXE) == RESET);
/* Send SPI1 data */
SPI_I2S_SendData(SPI1,0x0000);
/* Wait for SPI1 data reception */
while (SPI_I2S_GetFlagStatus(SPI1, SPI_I2S_FLAG_RXNE) == RESET);
/* Read SPI1 received data */
temp=SPI_I2S_ReceiveData(SPI1);
buf=temp<<8;
while (SPI_I2S_GetFlagStatus(SPI1, SPI_I2S_FLAG_TXE) == RESET);
/* Send SPI1 data */
SPI_I2S_SendData(SPI1,0x0000);
/* Wait for SPI1 data reception */
while (SPI_I2S_GetFlagStatus(SPI1, SPI_I2S_FLAG_RXNE) == RESET);
/* Read SPI1 received data */
temp=SPI_I2S_ReceiveData(SPI1);
buf |= temp;
buf>>=3;
buf&=0xfff;
return buf;
}
```

*Slika 30: Branje podatkov iz krmilnika za zaznavanje dotika*

### *Branje koordinate dotika*

Ukaz za branje koordinate X: 0x90
Ukaz za branje koordinate Y: 0xD0

Branje koordinate X (slika 31) zgleda takole:

```
pin TP_CS postavimo v nizko stanje

WR_CMD(0x90); // ukaz

rezultat = RD_AD();

pin TP_CS postavimo v visoko stanje
```

*Slika 31: Branje koordinate X*

V spremenljivki rezultat dobimo 12-bitno vrednost A/D pretvornika iz krmilnika za zaznavanje dotika. Branje koordinate moramo izvesti posebej za vsako od obeh osi.

Prvo branje koordinat dotika nam vedno da rezultat z največjim odstopanjem od natančnih koordinat, zato je potrebno večkratno branje istega dotika. Da bi dobili čimbolj natančne koordinate, smo za vsak dotik izvedli 7 branj, koordinate X smo shranili v tabelo koordinat X, koordinate Y pa v tabelo koordinat Y. V primeru da nam ni uspelo prebrati 7 odčitkov, smo dotik razveljavili. Če je bilo branje koordinat uspešno, smo z algoritmom za hitro določitev mediane sortirali obe tabeli in kot rezultat branja dotika vrnili mediani iz obeh tabel [27].

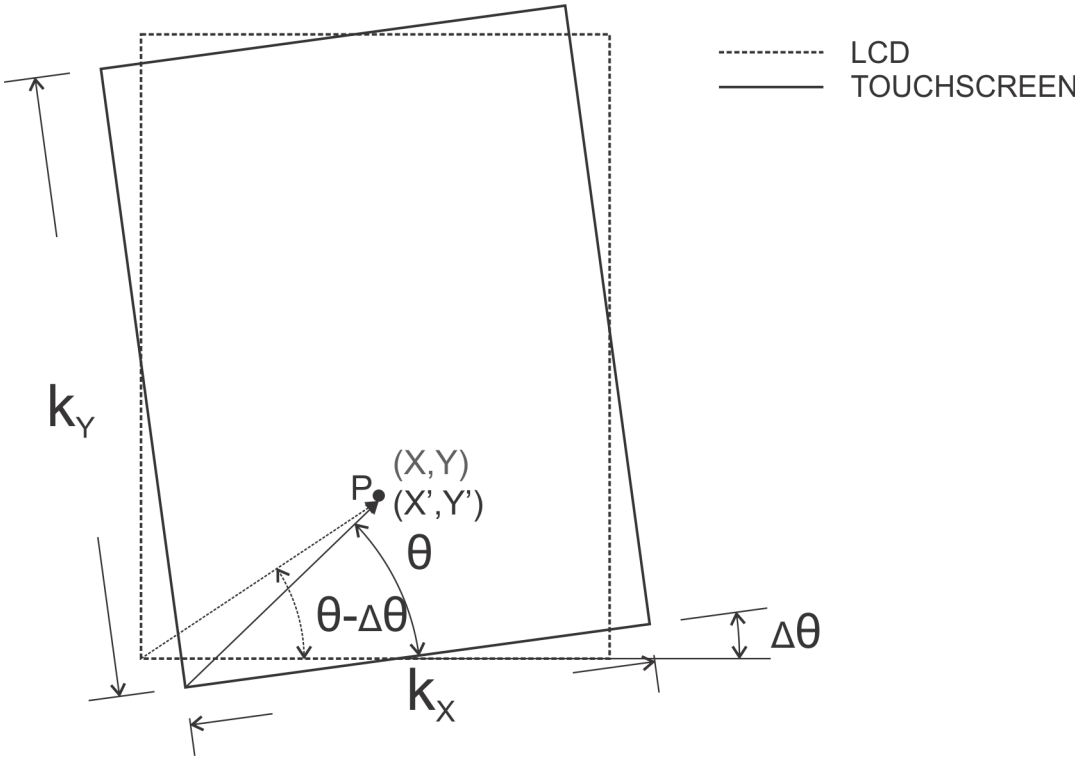
Prebrani položaj dotika je sestavljen iz dveh 12-bitnih števil, na primer T = (2419, 3181). Če poskušamo na zaslon narisati piko v točki T, vidimo, da ne bo šlo. Zaslon prikazuje točke od 0 do 239 v smeri X in od 0 do 319 v smeri Y, točka T pa daleč presega meje našega zaslona. Prebrani položaj dotika je očitno treba še dodatno obdelati; marsikdo bi pomislil na preprosto množenje koordinat z ustreznim skalirnim faktorjem, vendar se v praksi izkaže, da to ni dovolj. Prebrani koordinati moramo pomnožiti s kalibracijsko matriko, ki vsebuje koeficiente, s katerimi izračunamo koordinato na zaslonu. Najprej pa je treba kalibracijsko matriko izračunati v postopku kalibracije zaslona.

## 4.2.4 **Kalibracija zaslona**

Pri napravah, ki uporabljajo rezistivni zaslon na dotik, moramo skoraj vedno opraviti kalibracijsko rutino ob začetku uporabe. Kalibracija (umerjanje) je potrebna, ker je težko točno poravnati koordinate zaslona na dotik s koordinatami zaslona LCD. Neporavnanost opazimo, ko dotik na zaslonu ni pravilno interpretiran; pritisk na določen gumb na zaslonu se izrazi kot pritisk na napačen gumb ali pa se zdi, kot da naprava ne reagira na dotik.

Poznamo več izvorov napak, ki vplivajo na rezultate koordinat zaslona na dotik. Najpomembnejše so električni šum, mehanska neporavnanost in skalirni faktorji. Električni šum povzročajo zaslon in osvetlitev zaslona, okolje in drugi dejavniki. Vpliv električnega šuma običajno rešujemo z nizko prepustnim filtriranjem na vhodih A/D pretvornika. Kalibracija zaslona pomaga pri odpravi napak zaradi mehanske neporavnanosti in skalirnih faktorjev [28].

Opišimo lego zaslona in plošč zaslona na dotik pri mehanski neporavnanosti in skaliranju (slika 32). Zaslon LCD je narisan s prekinjeno črto, plošče zaslona na dotik pa z neprekinjeno črto.




*Slika 32: Mehanska neporavnanost in skaliranje pri zaslonih na dotik*

Točka (X',Y') predstavlja dotik na zaslonu na dotik, (X,Y) pa predstavlja točko na zaslonu LCD. V enačbah, ki sledijo, bomo opisali relacijo med točko dotika na zaslonu na dotik (ang. *touchscreen*) in točko na zaslonu, ki jo hočemo izračunati [29].

Razlaga uporabljenih koeficientov:

$k_X$ – raztezek /skrček (skaliranje) na osi X

$k_Y$ – raztezek/skrček (skaliranje) na osi Y

$R$ – razdalja od izhodišča (0,0) do točke P

$\Delta\theta$ – rotacija plošče zaslona na dotik glede na zaslon LCD

$\Delta X$ – premik v smeri osi X

$\Delta Y$ – premik v smeri osi Y

V enačbah (2) je predstavljen zapis koordinate X na zaslonu LCD z upoštevano mehansko neporavnanostjo in skaliranjem.

$$\begin{aligned} X &= k_X \times R \times \cos(\theta - \Delta\theta) + \Delta X \\ X &= k_X \times R \times \cos(\theta) \times \cos(\Delta\theta) + k_X \times R \times \sin\theta \times \sin(\Delta\theta) + \Delta X \\ X &= k_X \times X' \times \cos(\Delta\theta) + k_X \times Y \times \sin(\Delta\theta) + \Delta X \\ X &= \alpha_X \times X' + \beta_X \times Y' + \Delta X \end{aligned} \tag{2}$$

Koordinate X in Y na zaslonu LCD, ki ju bomo računali po branju iz krmilnika za zaznavanje dotika, so zapisane v enačbah (3).

$$\begin{aligned} X &= \alpha_X \times X' + \beta_X \times Y' + \Delta X \\ Y &= \alpha_Y \times X' + \beta_Y \times Y' + \Delta Y \end{aligned} \tag{3}$$

Koeficienti $\alpha, \beta, X', Y'$ so podani v enačbah (4).

$$\begin{aligned} \alpha_X &= k_X \times \cos(\Delta\theta) \\ \beta_X &= k_X \times \sin(\Delta\theta) \\ \alpha_Y &= -k_Y \times \sin(\Delta\theta) \\ \beta_Y &= k_X \times \cos(\Delta\theta) \\ X' &= R \times \cos(\theta) \\ Y' &= R \times \sin(\theta) \end{aligned} \tag{4}$$

Iz enačb (5) je razvidno, da za izračun koeficientov $\alpha_X, \alpha_Y, \beta_X, \beta_Y, \Delta X, \Delta Y$ potrebujemo vsaj 3 nekolinearne točke. $(X_1, Y_1), (X_2, Y_2), (X_3, Y_3)$ predstavljajo točke na zaslonu LCD, točke $(X'_1, Y'_1), (X'_2, Y'_2), (X'_3, Y'_3)$ pa dobimo z dotikom zaslona na dotik.

$$\begin{aligned} X_1 &= \alpha_X \times X'_1 + \beta_X \times Y'_1 + \Delta X \\ X_2 &= \alpha_X \times X'_2 + \beta_X \times Y'_2 + \Delta X \\ X_3 &= \alpha_X \times X'_3 + \beta_X \times Y'_3 + \Delta X \end{aligned}$$

$$\begin{aligned} Y_1 &= \alpha_Y \times X'_1 + \beta_Y \times Y'_1 + \Delta Y \\ Y_2 &= \alpha_Y \times X'_2 + \beta_Y \times Y'_2 + \Delta Y \\ Y_3 &= \alpha_Y \times X'_3 + \beta_Y \times Y'_3 + \Delta Y \end{aligned} \tag{5}$$

Enačbe (5) lahko zapišemo kot:

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = A \times \begin{pmatrix} \alpha_X \\ \beta_X \\ \Delta X \end{pmatrix}$$

$$\begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = A \times \begin{pmatrix} \alpha_Y \\ \beta_Y \\ \Delta Y \end{pmatrix} \tag{6}$$

in

$$A=\begin{pmatrix} X'_1 & Y'_1 & 1 \\ X'_2 & Y'_2 & 1 \\ X'_3 & Y'_3 & 1 \end{pmatrix} \tag{7}$$

.

Zdaj lahko izračunamo koeficiente, ki jih potrebujemo za izračun točke na zaslonu LCD. Da razrešimo enačbe (6), moramo obe strani pomnožiti z inverzno matriko A (enačba 7). Dobimo dve enačbi (8, 9), iz katerih lahko napišemo nove enačbe za izračun koeficientov. Izpeljavo enačb prepuščamo bralcu.

$$\begin{pmatrix} \alpha_X \\ \beta_X \\ \Delta X \end{pmatrix}=A^{-1}\times\begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \tag{8}$$

$$\begin{pmatrix} \alpha_Y \\ \beta_Y \\ \Delta Y \end{pmatrix}=A^{-1}\times\begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} \tag{9}$$

### *Implementacija kalibracijskega algoritma*

1. potrebno je izbrati kalibracijske točke na zaslonu $(X_k, Y_k) za\ k=1,2,...n \text{ in } n\geq 3$
2. potrebno je poklicati funkcijo za pridobitev koordinat iz krmilnika za zaznavanje dotikov
3. dotaknemo se prve točke $(X_1, Y_1)$ na zaslonu, pridobimo koordinate iz krmilnika za zaznavanje dotikov in jih shranimo v $(X'_1, Y'_1)$
4. prejšnji korak ponavljamo, da pridobimo vse $(X'_k, Y'_k) za\ k=2,3,..n \text{ in } n\geq 3$
5. pokličemo funkcijo za izračun $\alpha_X, \alpha_Y, \beta_X, \beta_Y, \Delta X, \Delta Y$

Za izračun $\alpha_X, \alpha_Y, \beta_X, \beta_Y, \Delta X, \Delta Y$ izračunamo determinante 7 matrik, ki jih sestavljajo kalibracijske točke na zaslonu in točke, prebrane iz krmilnika za zaznavanje dotikov. Iz rezultatov determinant teh matrik sestavimo kalibracijsko matriko posebej za koordinato X in Y. Za vsak dotik pomnožimo prebrane koordinate $(X', Y')$ iz krmilnika za zaznavanje dotikov s kalibracijsko matriko, da dobimo koordinate (10), ki so dokaj natančen približek točkam na zaslonu LCD.

$$X=\begin{pmatrix} X' \\ Y' \\ 1 \end{pmatrix}^{-1}\times\begin{pmatrix} \alpha_X \\ \beta_X \\ \Delta X \end{pmatrix} \qquad Y=\begin{pmatrix} X' \\ Y' \\ 1 \end{pmatrix}^{-1}\times\begin{pmatrix} \alpha_Y \\ \beta_Y \\ \Delta Y \end{pmatrix} \tag{10}$$

Za izračun kalibracijske matrike smo uporabili že implementirani funkciji setCalibrationMatrix in getDisplayPoint [30]. Funkcija setCalibrationMatrix poračuna prebrane točke s kalibracijskimi točkami in sestavi matriko. Točke na zaslonu smo izračunali s funkcijo getDisplayPoint, ki na podlagi točke prebrane iz krmilnika za detekcijo dotika in kalibracijske točke izračuna točko na zaslonu.

Kot že prej omenjeno, imamo za izvedbo kalibracijske rutine zaslona rezerviran pin PB11, ki je vezan na maso. Ob vklopu oziroma ob ponovnem zagonu naprava preveri stanje na vhodnem pinu PB11. Če je nizko stanje, prebere iz pomnilnika Flash v pomnilnik SRAM koeficiente za kalibracijsko matriko, ob visokem stanju pa se izvede kalibracijska rutina, kjer se po treh dotikih kalibracijskih točk poračunajo koeficienti kalibracijske matrike, ki se nato shranijo v pomnilnik Flash.

### 4.2.5 **Konfiguracija čitalca RFID**

Pri nastavitvah za čitalec kartic RFID, ki smo ga priključili na vrata A, nam ni bilo treba vključiti ure, ker smo jo že vključili pri krmilniku za zaznavanje dotika. Mikrokrmilnik in čitalec sta povezana z dvema pinoma, pinom TX, ki je priključen na pin PA3 na mikrokrmilniku, in pinom D1, priključenim na PA1. Oba pina v mikrokrmilniku smo nastavili kot vhoda brez vključenih »pull-up« uporov s parametrom GPIO_Mode_IN_FLOATING (slika 33).

```
GPIO_InitTypeDef GPIO_InitStructure;


GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(GPIOA, &GPIO_InitStructure);


GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

*Slika 33: Konfiguriranje pinov za čitalec RFID*

Sledila je nastavitev naprave USART2 (slika 34). Najprej smo napravi omogočili uro. Ker je naprava USART2 povezana na vodilo APB1, smo tu uporabili ukaz RCC_APB1PeriphClockCmd. Hitrost pošiljanja smo nastavili na 9600 baudov, format pošiljanja/sprejemanja je 8 podatkovnih bitov in 1 stop bit, brez paritete. Uporabljamo asinhronski prenos. S parametrom USART_Mode_Rx omogočimo sprejemanje.

```
RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);
USART_InitTypeDef usart;
```

```
usart.USART_BaudRate = 9600;

usart.USART_WordLength = USART_WordLength_8b;

usart.USART_StopBits = USART_StopBits_1;

usart.USART_Parity = USART_Parity_No ;

usart.USART_Mode = USART_Mode_Rx;

usart.USART_HardwareFlowControl = USART_HardwareFlowControl_None;

USART_Init(USART2, &usart);
```

*Slika 34: Nastavitev naprave USART2*

Nastavitve pinov za uporabo vmesnika USART smo opravili. Za branje podatkov iz USART-a imamo več možnosti: lahko v zanki čakamo na podatke (ang. *polling*), lahko jih beremo ob prekinitvah znak za znakom, lahko pa prepustimo branje napravi DMA. Naš mikrokrmilnik ima vgrajeno napravo DMA, zato smo se odločili, da bomo brali podatke s pomočjo DMA.

### *Nastavitev naprave DMA*

Naprava DMA omogoča, da lahko beremo iz pomnilnika in zapisujemo v pomnilnik, ne da bi obremenjevali procesor. Naprava DMA1 ima 7 kanalov, ki so povezani z A/D pretvornikom, vmesnikom SPI, vmesnikom USART, I2C in časovniki. Pri povezavi mikrokrmilnika s čitalcem kartic mikrokrmilnik samo sprejema podatke in jih ne pošilja. Sprejemnik naprave USART2, ki ga uporabljamo, je priklopljen na kanal 6 v napravi DMA1, zato v nadaljevanju nastavljamo nastavitve za kanal 6. Napravo DMA1 bomo uporabili za branje številke kartice iz čitalca RFID – naprava DMA1 bo podatke, ki jih sprejemamo z napravo USART2, prekopirala v spremenljivko RfidOznaka, kjer hranimo številko prebrane brezkontaktne kartice.

Najprej je bilo treba vključiti uro za napravo DMA1, ki je priključena na vodilo AHB (slika 35). Nato smo z ukazom DMA_DeInit ponastavili napravo na začetne nastavitve. V element DMA_PeripheralBaseAddr smo zapisali naslov, ki kaže na register za sprejemanje podatkov naprave USART2. S tem smo nastavili izvorni naslov. V MemoryBaseAddr smo vpisali naslov spremenljivke RfidOznaka, kamor hočemo, da se v DMA prenosu kopira vsebina. To je ponorni naslov. Z DMA_DIR_PeripheralSRC smo nastavili smer kopiranja, naprava je izvor. V BufferSize smo vpisali 16, toliko bajtov zasede koda kartice, ki jo bomo prebrali s čitalcem kartic. Z DMA_PeripheralInc_Disable smo onemogočili, da bi DMA naprava samodejno povečala naslov ob prenosu, z DMA_MemoryInc_Enable pa nastavili, da se naslov spremeni. V DMA_PeripheralDataSize in DMA_MemoryDataSize vpišemo, da je osnovna enota prenosa bajt. Z DMA_Mode_Circular smo nastavili, da gre za ponavljajoči prenos DMA, ki po zadnjem prenesenem bajtu napravo ponovno nastavi na iste vrednosti, kot smo jih vpisali. DMA_Priority_High nastavi, da imamo visoko prioriteto izvajanja, čeprav to ni pomembno, saj bomo izvajali le en prenos DMA. Z DMA_M2M_Disable smo nastavili, da prenos DMA proži izvorna naprava, v našem primeru USART2.

```
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
DMA_DeInit(DMA1_Channel6);

DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)USART2_DR_Address;
DMA_InitStructure.DMA_MemoryBaseAddr = (u32)&RfidOznaka;
DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
DMA_InitStructure.DMA_BufferSize = 16;
DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
DMA_InitStructure.DMA_Priority = DMA_Priority_High;
DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
DMA_Init(DMA1_Channel6, &DMA_InitStructure);

DMA_ITConfig(DMA1_Channel6, DMA_IT_TC, ENABLE);

USART_DMACmd(USART2, USART_DMAReq_Rx, ENABLE);
USART_Cmd(USART2, ENABLE);
```

*Slika 35: Nastavitev naprave DMA1*

Z DMA_ITConfig smo vključili prekinitev v napravi DMA1 na kanalu 6, ki se bo prožila ob končanem prenosu DMA. Na koncu vključimo napravo USART2.

### *Nastavitev prekinitve na vhodnemu pinu PA1*

Vhodni pin PA1 na mikrokrmilniku je povezan z izhodom D1 čitalca RFID, ki označuje, da je čitalec zaznal kartico in da bo kmalu pričel s pošiljanjem številke kartice. Signal D1 smo uporabili za sprožitev prenosa DMA, ki kopira prebrano številko kartice iz naprave USART2 v spremenljivko RfidOznaka v pomnilnik, tako da smo nastavili zunanjo prekinitev ob prehodu signala D1 iz nizkega v visoko stanje na pinu PA1.

Za nastavitev zunanjih prekinitev moramo najprej nastaviti krmilnik za zunanje prekinitve EXTI (ang. *External interrupt/event controller*), ki se nahaja v procesorju. Krmilnik EXTI sestavlja 19 detektorjev spremembe signala na linijah, ki so: 16 zunanjih prekinitvenih linij, po ena za vsak pin na posameznih vratih, linija za alarm RTC, dogodek USB Wakeup in

PVD (ang. *programmable voltage detector*) izhod. Vsaki vhodni liniji je možno nastaviti tip prekinitve in fronto signala, ki proži prekinitev: prednja fronta (ang. *rising*), zadnja fronta (ang. *falling*) ali oboje. Krmilnik EXTI nastavimo tako, da izberemo linijo, v našem primeru EXTI_Line1 (slika 36), z EXTI_Mode_Interrupt nastavimo prekinitev, ki se bo prožila ob prednji fronti (EXTI_Trigger_Rising). Nato s parametrom ENABLE omogočimo linijo in z EXTI_Init nastavimo krmilnik EXTI.

Nato sledi povezava krmilnika EXTI s krmilnikom za vektorske prekinitve NVIC. To storimo z nastavitvijo krmilnika NVIC. V element NVIC_IRQChannel smo vpisali EXTI1_IRQn, nastavili najnižjo prioriteto 15, skupino prioritet smo nastavili na 0, v NVIC_IRQChannelCmd omogočili prekinitve in nastavitve zapisali v krmilnik NVIC z ukazom NVIC_Init.

V Cortex-M3 jedru so višje prioritete predstavljene z nižjo številko. Najnižja prioriteta v našem primeru je predstavljena s številko 15 (1111 binarno), najvišja prioriteta pa ima številko 0. Pri nastavljanju prioritet imamo možnost uporabe skupine prioritet [31], vendar se pri tem ustrezno spremeni število prioritet. Število bitov za prioritete plus število bitov za skupino prioritet znaša pri STM32F103RB 4 bite.

```
EXTI_InitTypeDef        EXTI_InitStructure;


GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource1);
/* Configure EXTI1 line */
EXTI_InitStructure.EXTI_Line = EXTI_Line1;
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);


/* Enable and set EXTI1 Interrupt to the lowest priority */
NVIC_InitStructure.NVIC_IRQChannel = EXTI1_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 15;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x00;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
```

*Slika 36: Nastavitev krmilnikov EXTI in NVIC*

### *Nastavitev prekinitev za DMA*

```
      NVIC_InitStructure.NVIC_IRQChannel = DMA1_Channel6_IRQn;

      NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 15;

      NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;

      NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;



      NVIC_Init(&NVIC_InitStructure);
```

*Slika 37: Povezava naprave DMA1 s krmilnikom NVIC*

Ker smo pri inicializaciji naprave DMA vključili, da naprava proži prekinitve, jo moramo povezati s prekinitvenim krmilnikom NVIC (slika 37). Če tega ne naredimo, ne bo procesor nikoli šel v prekinitveni servisni program. Nastavitev krmilnika NVIC poteka podobno kot pri povezavi krmilnika EXTI, le da tu nastavimo za prekinitev kanal 6 naprave DMA1 (DMA1_Channel6_IRQn).

### *Branje številke kartice*

Ko približamo kartico anteni čitalca RFID kartic, začne čitalec z branjem kode kartice. Po končanem branju številke kartice postavi izhod D1 v visoko stanje in po 800 mikrosekundah (slika 40) začne s pošiljanjem številke kartice na izhodne pine. V nastavitvah prekinitev smo nastavili prekinitev na zunanjem pinu PA1, ki je povezan na pin D1 na čitalcu. Ob spremembi stanja pina PA1 iz nizkega v visoko stanje se proži prekinitveni servisni program (slika 38), kjer najprej pogledamo, če je prišlo do prekinitve na zunanji liniji 1. Če je, omogočimo prenos DMA na kanalu 6 v DMA1. Nato je treba pobrisati zastavico, ki označuje, da je prišlo do prekinitve, drugače bo prekinitvena zahteva ostala in se bo CPE (centralno procesna enota) spet odzvala nanjo.

```
if (EXTI_GetITStatus(EXTI_Line1) != RESET)

{

      DMA_Cmd(DMA1_Channel6, ENABLE);

      EXTI_ClearITPendingBit(EXTI_Line1);

}
```

*Slika 38: Prekinitveni servisni program, ki se sproži ob prehodu signala D1 iz nizkega v visoko stanje*

Ker smo omogočili prenos DMA, pred tem pa v nastavitvah za napravo DMA1 in krmilnik NVIC omogočili proženje prekinitev ob končanem prenosu DMA, naprava DMA1 ob prebrani številki kartice sproži prekinitev. Prebrane podatke shrani naprava DMA v

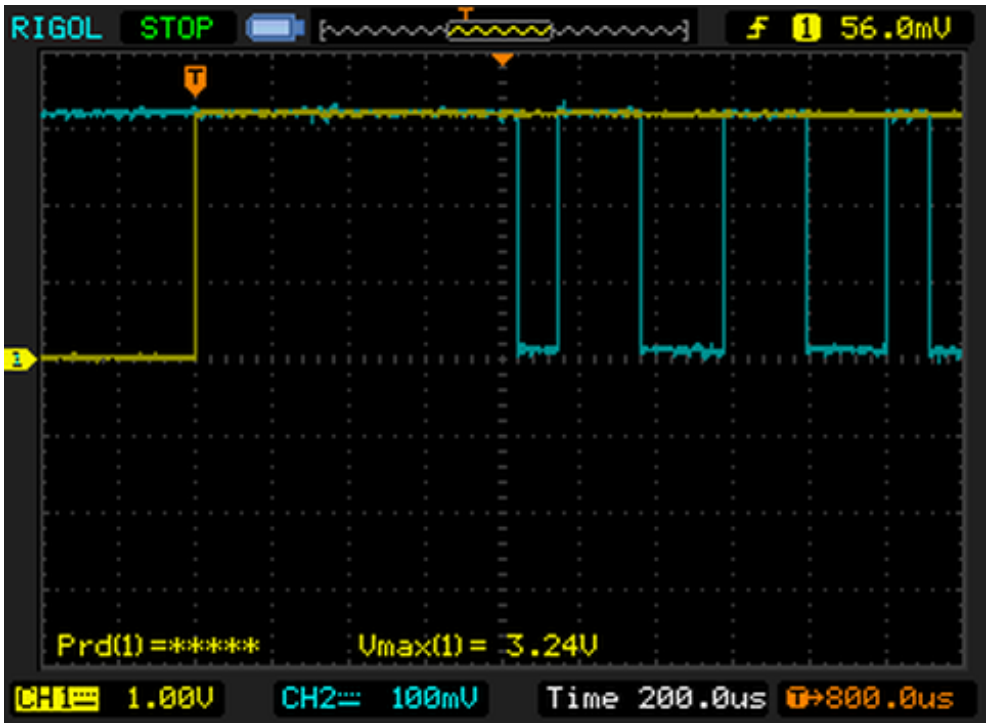
spremenljivko RfidOznaka, ki je definirana kot tabela znakov velikosti 17 bajtov. Prikaz signalov pri branju brezkontaktne kartice vidimo na sliki 41.

V prekinitvenem servisnem programu naprave DMA1 (slika 39) najprej pregledamo, ali je prišlo do prekinitve zaradi končanega prenosa na kanalu 6 v napravi DMA1. Nato z DMA_ClearITPendingBit pobrišemo zastavice, ki označujejo, da je naprava prožila prekinitev. Ker je prenos DMA zaključen, onemogočimo kanal 6 v napravi DMA1.

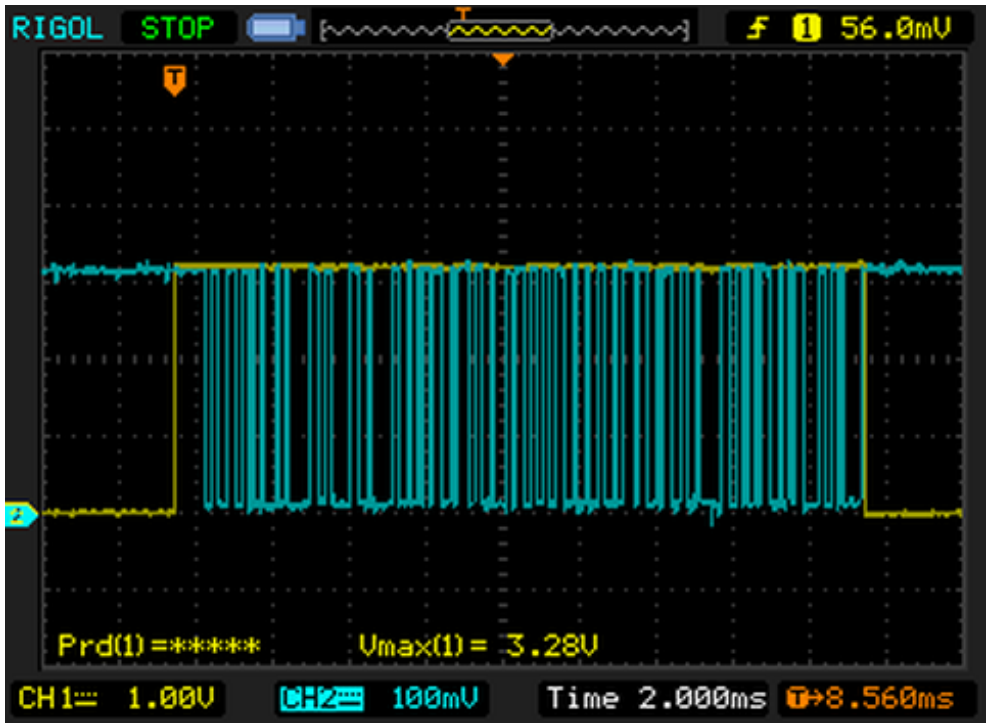
Na koncu prekinitvenega servisnega programa pregledamo še spremenljivko zaslon, ki hrani stanje, v katerem se nahaja končni avtomat. Če smo v stanju LCD_UVOD, kjer čakamo kodo kartice, s pisanjem 1 v spremenljivko kartica_prebrana omogočimo pogoj, da končni avtomat spremeni stanje v LCD_VNESI, kjer beremo kodo PIN.

```
if ( DMA_GetFlagStatus(DMA1_FLAG_TC6) == SET )

{

      DMA_ClearITPendingBit(DMA1_IT_GL6);

      DMA_Cmd(DMA1_Channel6, DISABLE);


      if (zaslon == LCD_UVOD)

      {

            kartica_prebrana = 1;

      }

}
```

*Slika 39: Prekinitveni servisni program, ki se sproži ob zaključku prenosa DMA naprave DMA1*




*Slika 40: Prikaz signala D1 (rumena barva) in signala TX (modra barva) čitalca RFID*




*Slika 41: Prikaz signala D1 (rumena barva) in TX (modra barva) pri branju brezkontaktne kartice*

### 4.2.6 Konfiguracija krmilnika za Ethernet ENC28J60

Krmilnik za Ethernet smo povezali na vrata B. Morali smo vključiti uro napravi SPI2 in ustrezno nastaviti pine PB12, PB13, PB14, PB15. Pin PB12 je uporabljen za izbiro naprave, PB13 za uro vmesnika SPI2, PB14 za sprejemanje podatkov v mikrokrmilnik, PB15 pa za pošiljanje podatkov v ENC28J60.

Nastavitev naprave SPI2 je zelo podobna nastavitvi vmesnika SPI1, le da tu spremenimo delilnik za frekvenco ure, ki jo nastavimo na 9 Mhz. Ker je vmesnik SPI2 na vodilu APB1, ki deluje s frekvenco 36 Mhz, smo izbrali delilnik s faktorjem 4.

Nato smo izvedli inicializacijo krmilnika, kjer je, podobno kot za nastavitev krmilnika LCD, bilo potrebno pisati v kontrolne registre krmilnika za Ethernet. V sklopu inicializacije smo nastavili želeno velikost vmesnega pomnilnika (ang. *buffer*) za pakete, ki jih sprejemamo in pakete, ki jih pošiljamo. Nastavili smo filter za pakete, ki jih sprejemamo: blokirali smo broadcast pakete z izjemo paketov ARP (ang. *Address Resolution Protocol*) in nastavili, da sprejemamo samo unicast pakete za naš naslov MAC (ang. *Media Access Control*). Vključili smo samodejno računanje CRC (ang. *Cyclic Redundancy Check*) v linijskem sloju in nastavili največjo velikost paketa 1500 bajtov, na koncu pa smo vnesli naslov MAC [15].

Pri pisanju funkcij za uporabo krmilnika za Ethernet smo si pomagali z že napisanimi funkcijami [32], ki smo jih spremenili za delovanje z našim mikrokrmilnikom.

Pošiljanje podatkov iz mikrokrmilnika v ENC28J60 se začne z nastavitvijo kazalca za pisanje (slika 42). To naredimo s pisanjem naslova v registra EWRPTL in EWRPTH, kjer se začne pomnilnik za pošiljanje paketov. Spomnimo, vmesni pomnilnik velikosti 8 kB je namenjen hkrati za prejemanje in pošiljanje paketov, sami razdelimo vmesni pomnilnik na dva dela z nastavitvijo kazalcev. Ker smo za največjo velikost paketa določili 1.500 bajtov, smo nastavili TXSTART_INIT na $0x1FFF - 0x600 = 0x19FF$, kjer 0x1FFF pomeni 8 kB, 0x600 pa 1500 bajtov. Podobno kot za kazalec za začetek pisanja nastavimo kazalec, ki kaže na mesto v vmesnem pomnilniku, kjer se pisanje konča. To je odvisno od velikosti podatkov, ki jih pošiljamo. Nato zapišemo ukaz ENC28J60_WRITE_BUF_MEM, ki pripravi krmilnik na zapis podatkov v vmesni pomnilnik. Nato podatke pošljemo s funkcijo enc28j60WriteBuffer in na koncu s postavitvijo bita ECON1_TXRTS damo ukaz krmilniku, da pošlje paket.

```
void enc28j60PacketSend(unsigned int len, unsigned char* packet)

{

      // Set the write pointer to start of transmit buffer area

      enc28j60Write(EWRPTL, TXSTART_INIT&0xFF);

      enc28j60Write(EWRPTH, TXSTART_INIT>>8);


      // Set the TXND pointer to correspond to the packet size given

      enc28j60Write(ETXNDL, (TXSTART_INIT+len)&0xFF);

      enc28j60Write(ETXNDH, (TXSTART_INIT+len)>>8);
```

```
      // write per-packet control byte (0x00 means use macon3 settings)
      enc28j60WriteOp(ENC28J60_WRITE_BUF_MEM, 0, 0x00);

      // copy the packet into the transmit buffer
      enc28j60WriteBuffer(len, packet);

      // send the contents of the transmit buffer onto the network
      enc28j60WriteOp(ENC28J60_BIT_FIELD_SET, ECON1, ECON1_TXRTS);
}
```

*Slika 42: Pošiljanje paketa s krmilnikom za Ethernet*

Za prejem paketa smo uporabili funkcijo, ki je po načinu delovanja podobna funkciji za pošiljanje podatkov. Najprej je treba nastaviti kazalec na naslov v vmesnem pomnilniku, ki kaže na prejeti paket, ki ga še nismo prebrali.

Prebrani paket je sestavljen iz zaglavja in podatkov. Zaglavje, sestavljeno iz 6-ih bajtov, vsebuje kazalec, ki kaže na naslov naslednjega prebranega paketa, in statusni vektor, v katerem so zapisani lastnosti paketa in velikost prejetega okvirja. Zato je pomembno, da v mikrokrmilniku hranimo naslov naslednjega paketa.

Nato preberemo naslov naslednjega prejetega paketa in ga shranimo v pomnilnik. Sledi branje podatka o velikosti paketa, nato pa preberemo vsebino paketa iz krmilnika za Ethernet v mikrokrmilnik. Po branju paketa sprostimo vmesni pomnilnik s pisanjem naslova naslednjega paketa v register ERXRDPT, ki preprečuje pisanje v vmesni pomnilnik od naslova v registru dalje. Na koncu zmanjšamo števec paketov v krmilniku za 1.

Predstavili smo osnovni nalogi krmilnika za Ethernet, ki obvladuje linijski in fizični sloj OSI modela. Za implementacijo mrežnega in transportnega sloja smo uporabili sklad uIP TCP/IP.

### 4.2.7 Sklad uIP TCP/IP

Sklad uIP [33] (izgovori se mikro IP) je implementacija sklada TCP/IP. Namen njegovega razvoja je bil, da bi tudi 8-bitni mikrokrmilniki lahko komunicirali z uporabo zbirke protokolov TCP/IP. Z razvojem sklada uIP je začel Adam Dunkels, kasneje so se razvoju priključili še drugi razvijalci. uIP je licenciran z licenco BSD in sedaj spada v projekt Contiki. Contiki je majhen, visoko prenosljiv operacijski sistem za majhne naprave, ki vključuje uIP sklad, grafične knjižnice, strežnik VNC (ang. *Virtual Network Computing*), najmanjši spletni brskalnik in druge funkcionalnosti. uIP je napisan v programskem jeziku C in vsebuje protokole IP (ang. *Internet Protocol*), ICMP (ang. *Internet Control Message Protocol*), UDP (ang. *User Datagram Protocol*) in protokol TCP (ang. *Transmission Control Protocol*). Združljiv je z dokumenti RFC (ang. *Request for Comments*). Uporabili smo verzijo uIP 1.0.

Za razliko od drugih skladov TCP/IP, ki zavzamejo nekaj sto kilobajtov programske kode in nekaj sto kilobajtov RAM-a, porabi sklad uIP samo nekaj kilobajtov kode in nekaj kilobajtov RAM-a. To je možno, ker sklad uIP vsebuje le najmanjši set funkcionalnosti, ki sestavljajo polno delujoči sklad TCP/IP – uIP ne vsebuje določenih mehanizmov, za katere velja, da so v uporabi le v zelo redkih primerih in jih lahko odstranimo, ne da bi s tem vplivali na pravilnost delovanja [34].

### *Delovanje sklada uIP*

Za komunikacijo preko vmesnika Ethernet potrebujemo poleg sklada TCP/IP tudi krmilnik za Ethernet in aplikacijo, ki bo obdelovala podatke, ki jih želimo prejemati/pošiljati. Vse to imamo, zato si v nadaljevanju oglejmo povezave med posameznimi deli.

Sklad uIP za prejemanje in pošiljanje paketov uporablja globalni vmesni pomnilnik (v kodi poimenovan uip_buf). Ta pomnilnik je po velikosti ravno dovolj velik, da lahko hrani paket največje velikosti. Ko paket prispe po omrežju, ga gonilnik krmilnika za Ethernet zapiše v globalni vmesni pomnilnik in pokliče sklad uIP. Če prejeti paket vsebuje podatke, sklad uIP obvesti uporabniško aplikacijo (v nadaljevanju poimenovana omrežna aplikacija), ki nato obdela prejete podatke. Analogno je pri pošiljanju paketov – omrežna aplikacija pripravi podatke za pošiljanje in jih posreduje skladu uIP, ki podatke ovije v paket in jih zapiše v globalni vmesni pomnilnik, nato pa pokliče gonilnik krmilnika za Ethernet, ki pošlje paket v omrežje.

V primeru neuspešnega pošiljanja paketa mora omrežna aplikacija za novo pošiljanje pripraviti enake podatke kot pri prejšnjem pošiljanju. Sklad uIP ne hrani paketov v pomnilniku za ponovno pošiljanje, ravno tako sprejema samo po en paket iz omrežja naenkrat, ki ga mora omrežna aplikacija obdelati preden se prebere nov paket. Razlog za to se skriva v uporabi globalnega vmesnega pomnilnika, posledica pa je zmanjšanje porabe pomnilnika.

### *Glavna zanka sklada uIP*

Glavna zanka sklada uIP v osnovi počne dve stvari:

- pregleduje, če je prispel nov paket,
- pregleduje, če je prišlo do periodične zakasnitve.

Periodične zakasnitve so uporabljene za določene mehanizme TCP, kot so odložene potrditve (ang. *delayed acknowledgments*), ponovno pošiljanje paketa in druge.

Spodaj lahko vidimo delovanje glavne zanke sklada uIP. Paket najprej preberemo z uporabo ukaza etherdev_read, ki vrne število prebranih bajtov, ki ga zapišemo v spremenljivko uip_len. Ukaz etherdev_read (slika 43) je v našem primeru ukaz enc28j60PacketReceive(1500, uip_buf), kjer 1500 pomeni največjo velikost paketa, uip_buf pa kot že omenjeno, globalni vmesni pomnilnik.

```
void etherdev_pooling(void)

{

      uint8_t i = 0;


      uip_len = etherdev_read();
```

*Slika 43: Glavna zanka sklada uIP, prvič*

Nato pregledamo, če je prispel nov paket – uip_len mora biti večji od nič v tem primeru. Če je, pogledamo tip prispelega paketa. Možna sta dva, prispeli paket je paket IP (UIP_ETHTYPE_IP) ali pa paket za protokol ARP (UIP_ETHTYPE_ARP). To storimo s preverjanjem strukture BUF, ki kaže na globalni vmesni pomnilnik.

Če je prispeli paket tipa IP (slika 44), je treba uporabiti ukaz uip_arp_ipin, ki pregleda, ali je naslov že v tabeli ARP; če je, ga osveži, če ga ni, ga vpiše. Nato je treba uporabiti ukaz uip_input. S tem ukazom sklad uIP obdela paket, pripravi podatke za omrežno aplikacijo in kliče omrežno aplikacijo. Na koncu je treba še enkrat pogledati ali je uip_len večji od nič. Če je, pomeni, da je omrežna aplikacija pripravila podatke za pošiljanje. Ukaz uip_arp_out nato preveri, ali se naslovnik nahaja v tabeli ARP, in podatkom doda zaglavje paketa. Z etherdev_send se nato podatki pošljejo po liniji.

V primeru, da je prispeli paket tipa ARP, je potrebno paket obdelati z ukazom uip_arp_arpin. Ta ukaz nato glede na tip ARP paketa pripravi nov paket, v katerem pošlje naslov MAC našega krmilnika za Ethernet ali pa iz odgovora na povpraševanje za naslov IP vpiše prejeti naslov MAC v tabelo ARP. Podobno kot pri paketu IP, je tudi tu potrebno pogledati uip_len za podatke, ki jih je potrebno poslati po liniji.

```
  if(uip_len > 0)

      {

            if(BUF->type == htons(UIP_ETHTYPE_IP))

            {

                  uip_arp_ipin();

                  uip_input();

                  if(uip_len > 0) {

                        uip_arp_out();

                        etherdev_send();

                  }

            }

            else if(BUF->type == htons(UIP_ETHTYPE_ARP))
```

```
            {
                  uip_arp_arpin();
                  if(uip_len > 0) {
                        etherdev_send();
                  }
      }
      }
```

*Slika 44: Glavna zanka sklada uIP, drugič*

V programski kodi (slika 45) vidimo drugo nalogo glavne zanke – pregledovanje, ali je prišlo do periodične zakasnitve. Če je prišlo, najprej ponastavimo časovnik periodic_timer. Nato za vsako od povezav (možnih je UIP_CONNS sočasnih povezav, v našem primeru je to 10 povezav) z ukazom uip_periodic sprožimo potrebno periodično obdelavo (vezano na čas) za povezavo TCP. Kot prej tudi tu pregledamo, če je potrebno poslati kakšne podatke po liniji. Na koncu pregledamo ali je časovnik arp_timer potekel, če je, z ukazom uip_arp_timer poženemo ustrezno periodično procesiranje v modulu ARP.

```
      else if(timer_expired(&periodic_timer))
      {
            timer_reset(&periodic_timer);
            for(i = 0; i < UIP_CONNS; i++)
            {
                  uip_periodic(i);


                  if(uip_len > 0)
                  {
                        uip_arp_out();
                        etherdev_send();
                  }
            }
                  if(timer_expired(&arp_timer))
                  {
                        timer_reset(&arp_timer);
                        uip_arp_timer();
```

```
                    }

        }

}
```

*Slika 45: Glavna zanka sklada uIP, tretjič*

Glavno zanko sklada uIP, ki je opisana zgoraj, izvajamo v while zanki main funkcije za funkcijo, ki upravlja končni avtomat sistema.

### *Omrežna aplikacija*

Za delovanje sklada uIP je potrebno napisati aplikacijo, ki jo bo sklad uIP klical v funkciji uip_input. V ta namen smo napisali funkcijo eth_appcall, ki upravlja sklad uIP.

Za povezavo sklada uIP in omrežne aplikacije, je potrebno definirati makro (slika 46), ki UIP_APPCALL zamenja z imenom omrežne aplikacije. Brez tega prevajalnik ne prevede kode, saj je omrežna aplikacija v kodi sklada poimenovana z imenom UIP_APPCALL.

```
#define UIP_APPCALL eth_appcall
```

*Slika 46: Makro, ki zamenja UIP_APPCALL z eth_apcall*

Nato je potrebno definirati strukturo (v našem primeru struktura ethapp_appstate, slika 47), v kateri hranimo podatke, katere vsebina se razlikuje glede na povezavo. V našem primeru te funkcionalnosti sklada uIP nismo uporabili, strukturo pa smo morali vendarle definirati, ker drugače prevajalnik ni prevedel kode. Uporaba strukture nam olajša pisanje aplikacij kadar pošiljamo/sprejemamo več paketov v isti povezavi. Strukturo lahko dodatno dopolnimo in v njej hranimo podatke za trenutno povezavo.

```
struct ethapp_appstate

{

      u8_t state;

};



typedef struct ethapp_appstate uip_tcp_appstate_t;
```

*Slika 47: Definicija strukture ethapp_appstate*

Vmesnik API (ang. *Application programming interface*) sklada uIP definira način, kako aplikacija komunicira s skladom uIP. Na voljo je več funkcij, s katerimi testiramo, v katerem stanju se povezava nahaja. S funkcijo uip_connected testiramo, če je bila vzpostavljena nova povezava. Funkcija uip_rexmit preveri, če je prišlo do izteka časovnika za ponovno pošiljanje, funkcija uip_newdata preveri, če je prišel nov paket, in funkcija uip_close preveri, če je nasprotna stran zaprla povezavo. Obstaja še veliko drugih funkcij v vmesniku API.

V funkciji eth_appcall je predstavljena (omrežna) aplikacija (slika 48), ki smo jo napisali za komunikacijo vgrajenega sistema s strežnikom na računalniku. Najprej testiramo, če je bila vzpostavljena nova povezava in ali se je časovnik za ponovno pošiljanje iztekel. To naredimo s klicanjem funkcij uip_connected in uip_rexmit. Če je eden od pogojev pravilen, kličemo funkcijo senddata, ki pripravi niz znakov, sestavljen iz številke naprave, številke kartice in kode PIN. V funkciji senddata s klicem funkcije uip_send zapišemo podatke v globalni vmesni pomnilnik, ki jih nato sklad uIP v funkciji uip_arp_out preoblikuje v paket IP.

S funkcijo uip_newdata preverimo, če je prišel nov paket. Če je, potem preverimo podatke, ki jih pripravi sklad uIP, do katerih pridemo s kazalcem uip_appdata. Če strežnik v odgovor vrne »dovolivstop«, nastavimo novo stanje končnemu avtomatu v LCD_PRAVILEN; če pa v prejetem paketu preberemo »stop«, pomeni, da je prišlo do neujemanja številke kartice in kode PIN, nastavimo stanje LCD_NAPACEN.

Funkcija uip_timedout preveri, če je sklad uIP večkrat ponovno poslal isti paket. Če je, pomeni da imamo težave s povezavo, zato nastavimo novo stanje avtomata LCD_ETH_NAPAKA in zapremo povezavo.

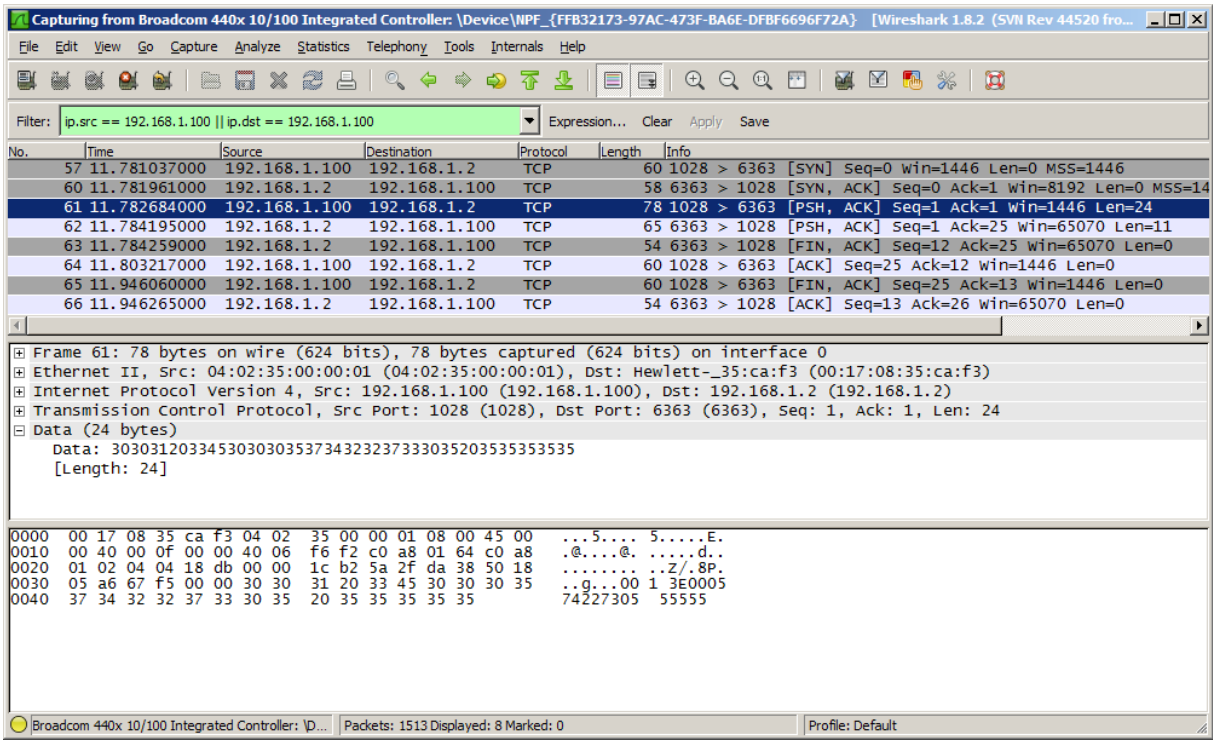
V zadnjem if-stavku s funkcijo uip_closed preverimo, če je strežnik zaprl povezavo, in jo zapremo.

```
void eth_appcall(void)

{

      // serverju posljemo podatke: kodo in pin

      if ( uip_connected() || uip_rexmit() )

      {

            senddata();

            //uip_send(»st_naprave stevilka_kartice koda_PIN«, length);

            return;

      }



      //odgovor streznika

      if ( uip_newdata() )

      {

            /* spremenimo stanje končnega avtomata glede na odgovor
            streznika; LCD_NAPACEN pri neujemanju kode in pina, pri
            ujemanju pa je novo stanje LCD_PRAVILEN */

            ...

      }
```

```
      if (uip_timedout())
      {
            zaslon = LCD_ETH_NAPAKA;
            execute = 1;
            uip_close();
      }

      if (uip_closed())
      {
            uip_close();
      }
}
```

*Slika 48: Omrežna aplikacija*

Na sliki 49 vidimo pošiljanje zahtevka za preverjanje številke kartice in kode PIN. Naprava pošlje po omrežju niz znakov, ki je sestavljen iz treh znakov za številko naprave, enega presledka, 14-ih znakov številke kartice, enega presledka in 5-ih znakov za kodo PIN. V našem primeru je številka naprave 001, številka kartice 3E000574227305, koda PIN 55555.




*Slika 49: Pošiljanje prijavnih podatkov v strežnik*

### *Inicializacija sklada uIP*

Inicializacijo sklada uIP pričnemo s klicem funkcije uip_init (slika 50), ki nastavi začetne nastavitve sklada. Nato je treba nastaviti naslov IP za napravo. To naredimo s klicem funkcij uip_ipaddr in uip_sethostaddr. Funkcija uip_ipaddr kopira naslov IP, predstavljen s 4 bajti v spremenljivko ipaddr. Nato s funkcijo uip_sethostaddr nastavimo napravi naslov IP [34].

S funkcijo uip_ipaddr kopiramo naslov strežnika v spremenljivko rmipaddr, ki jo bomo uporabljali za povezavo s strežnikom.

Na koncu nastavimo naslov MAC, ki ga zapišemo v sklad uIP v tabelo uip_ethaddr.addr.

```
uip_init();


1  uip_ipaddr_t ipaddr, rmipaddr;
// nastavimo ip naprave
uip_ipaddr(&ipaddr, 192,168,1,100);
uip_sethostaddr(&ipaddr);


// nastavimo ip serverja
uip_ipaddr(&rmipaddr, 192,168,1,2);


uip_ethaddr.addr[0] = mymac[0];
uip_ethaddr.addr[1] = mymac[1];
uip_ethaddr.addr[2] = mymac[2];
uip_ethaddr.addr[3] = mymac[3];
uip_ethaddr.addr[4] = mymac[4];
uip_ethaddr.addr[5] = mymac[5];
```

*Slika 50: Inicializacija sklada uIP*

S funkcijo uip_connect vzpostavimo povezavo s strežnikom (slika 51). V našem primeru se povežemo na naslov IP, ki je v spremenljivki rmipaddr, na vrata 6363. Makro HTONS() pretvori 16-bitno število iz načina predstavitve števila v mikrokrmilniku v predstavitev za pošiljanje po omrežju. Funkcijo spodaj kličemo po vnosu kode PIN in pritisku na zeleno tipko.

```
uip_connect(&rmipaddr, HTONS(6363));
```

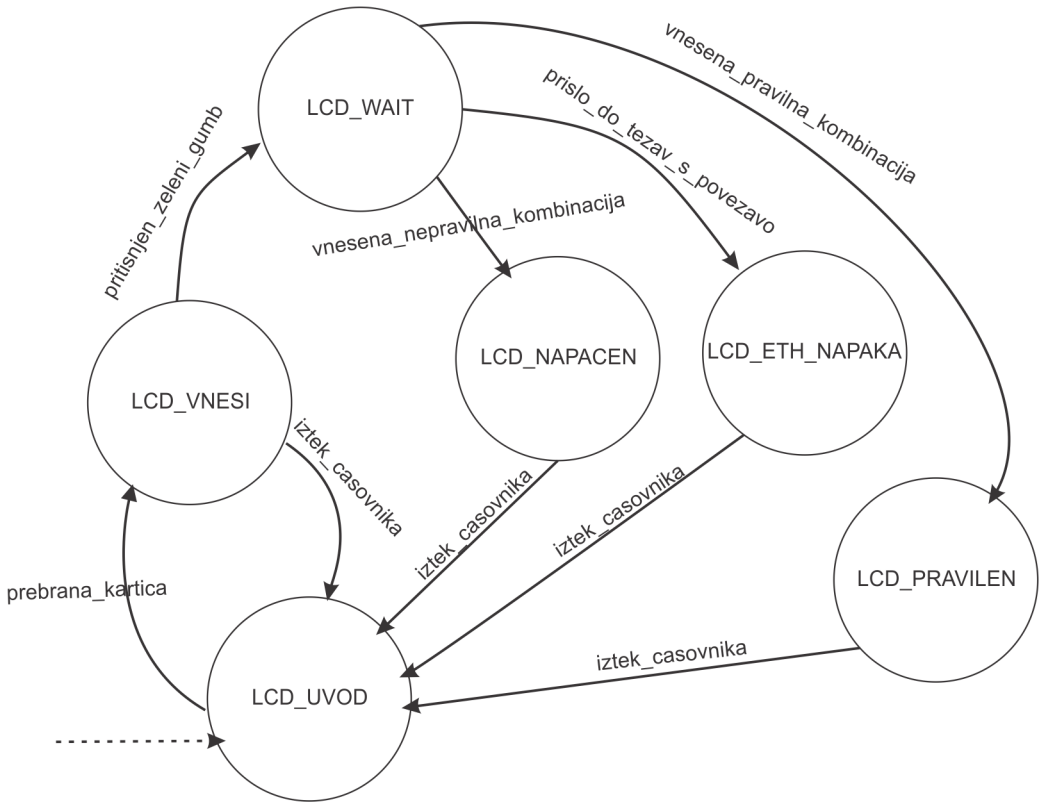
*Slika 51: Povezovanje na strežnik*

### 4.2.8 **Končni avtomat**

Delovanje našega vgrajenega sistema krmili končni avtomat. Avtomat se lahko nahaja v več stanjih: LCD_UVOD, LCD_VNESI, LCD_WAIT, LCD_ETH_NAPAKA, LCD_PRAVILEN, LCD_NAPACEN. Delovanje avtomata je prikazano na sliki 52.

LCD_UVOD: Ob vklopu/ponovnem zagonu naprave postavimo avtomat v začetno stanje LCD_UVOD. Avtomat na zaslon izpiše obvestilo »PRIBLIZAJ KARTICO« ter čaka na številko kartice. Ko približamo brezkontaktno kartico čitalcu kartic, se prožita dve prekinitvi – ob začetku branja (da omogočimo prenos DMA) in po končanem branju številke kartice (onemogočimo napravo DMA). Po končanem branju kode kartice, ki se zapiše v tabelo znakov RfidOznaka, se zapiše še enica v spremenljivko kartica_prebrana in s tem se izpolni pogoj, da avtomat preide v stanje LCD_VNESI.

LCD_VNESI: Avtomat izpiše obvestilo »VNESI PIN«, po eni sekundi pa izriše številčnico na zaslon. V tem stanju avtomat čaka na uporabnikov dotik zaslona. Če v obdobju 5 sekund ni dotika, se avtomat vrne v stanje LCD_UVOD. Ob vsakem dotiku se časovnik, ki odšteva čas do vrnitve v začetno stanje, ponastavi. Nato se preveri koordinate dotika s koordinatami gumbov številčnice. Če se prebrani dotik izrazi kot dotik gumba, se v statusni vrstici številčnice izriše pravokotnik za vsak zaznani pritisk gumba. Branje pritisnjenih gumbov se izvaja do števila petih prebranih številk. Vsak pritisnjen gumb shranimo v tabelo znakov pin. Poleg številk lahko pritisnemo še zeleni in rdeči gumb. Po petih pritisnjenih številkah se omogoči zeleni gumb, ki se iz temno zelene obarva v svetlo zeleno barvo. Ob pritisku na zeleni gumb se naprava poveže na strežnik in spremeni stanje v LCD_WAIT. Ob pritisnjenem rdečem gumbu se pobriše tabelo znakov pin in se počisti statusna vrstica.




*Slika 52: Diagram prehajanja med stanji za končni avtomat vgrajenega sistema*

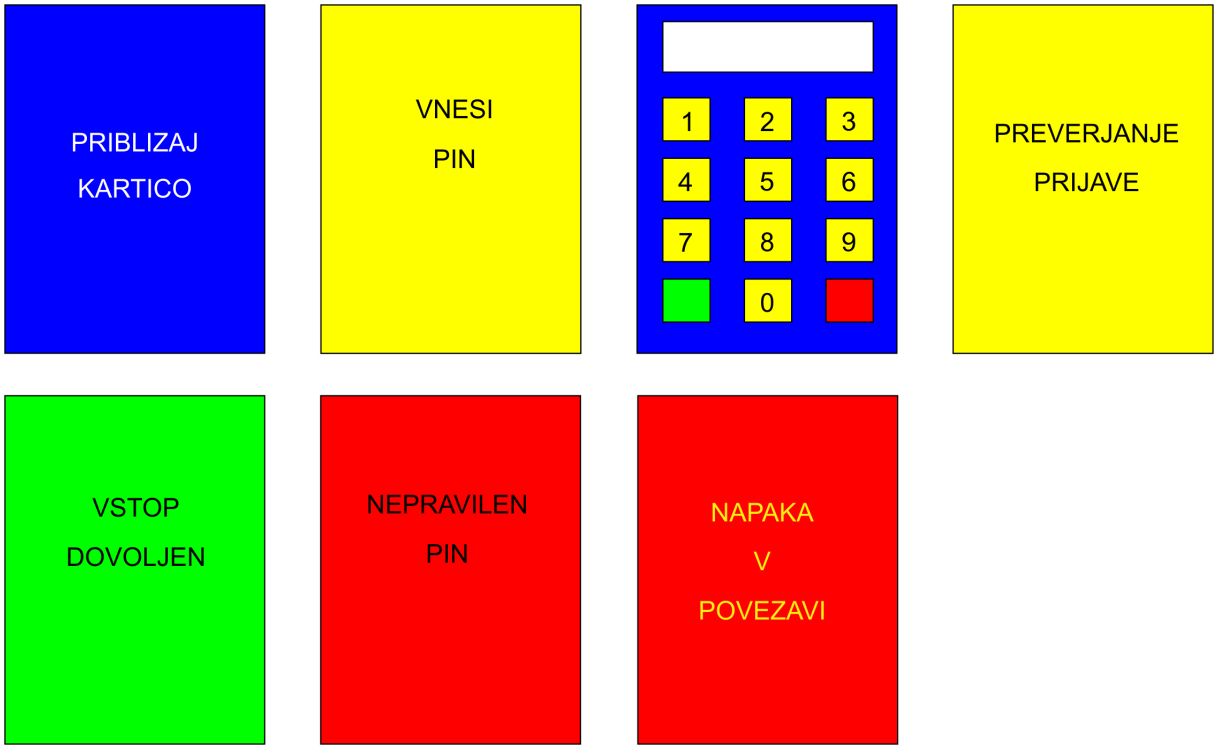
LCD_WAIT: Avtomat izpiše obvestilo »PREVERJANJE PRIJAVE« in čaka na odziv strežnika. V primeru da se par (številka kartice, koda PIN) ujema v bazi strežnika, preide avtomat v stanje LCD_PRAVILEN, ob neujemanju pa v stanje LCD_NAPACEN. Če je prišlo do težav s povezavo na strežnik, gre avtomat v stanje LCD_ETH_NAPAKA.

LCD_PRAVILEN: Avtomat izpiše obvestilo »VSTOP DOVOLJEN«, postavi pin PB10 v visoko stanje in ga po 4 sekundah nazaj postavi v nizko stanje. Nato spremeni stanje avtomata v LCD_UVOD.

LCD_NAPACEN: Avtomat izpiše obvestilo »NEPRAVILEN PIN«, po 4 sekundah pa spremeni stanje avtomata v LCD_UVOD.

LCD_ETH_NAPAKA: Avtomat izpiše obvestilo »NAPAKA V POVEZAVI«, v tem stanju počaka 10 sekund, nato se vrne v začetno stanje LCD_UVOD.

Na sliki 53 so prikazani izpisi na zaslon, ki jih izrisujemo glede na stanje, v katerem se nahaja avtomat.




*Slika 53: Zaslonske maske za posamezna stanja avtomata*

Končni avtomat smo implementirali s switch stavkom in spremenljivko zaslon, ki hrani trenutno stanje (slika 54). V vsakem stanju smo pregledovali, če so izpolnjeni pogoji za skok v novo stanje avtomata, in izvajali nalogo avtomata. Prehode v stanja LCD_PRAVILEN, LCD_NAPACEN in LCD_ETH_NAPAKA izvede omrežna aplikacija, ko se nahajamo v stanju LCD_WAIT.

```
switch (zaslon)

{
```

```
case LCD_UVOD:
      if (execute)
      {
            prikazi_uvodno_zaslonsko_masko();
            execute = 0;
      }
      else if (kartica_prebrana)
      {
            kartica_prebrana = 0;
            zaslon = LCD_VNESI;
            execute = 1;
      }
      break;

case LCD_VNESI:
      if (execute)
      {
            zelena_tipka = 0;
            execute = 0;
      }
      else if (zelena_tipka)
      {
            povezi_se_na_streznik();
            zaslon = LCD_WAIT;
            execute = 1;
      }
      else if (zaznan_dotik)
      {
            preberi_tipkovnico();
            resetiraj_casovnik();
      }
```

```
            else if (casovnik_pretecen) //timed out
            {
                  zaslon = LCD_UVOD;
                  execute = 1;
            }
            break;

      case LCD_PRAVILEN:
            if (execute)
            {
                  prikazi_obvestilo_o_dovoljenem_vstopu();
                  Vklopi_izhod(); /* PB10 */
                  resetiraj_casovnik();
                  execute = 0;
            }
            else if (casovnik_pretecen) //timed out
            {
                  Izklopi_izhod();
                  zaslon = LCD_UVOD;
                  execute = 1;
            }
            break;

      case LCD_NAPACEN:
            if (execute)
            {
                  prikazi_obvestilo_o_napacnem_pinu();
                  execute = 0;
            }
            else if (casovnik_pretecen) //timed out
            {
```

```
                zaslon = LCD_UVOD;
                execute = 1;
          }
          break;

    case LCD_ETH_NAPAKA:
          if (execute)
          {
                prikazi_obvestilo_o_napaki_v_povezavi();
                pocakaj_10_sekund();
                zaslon = LCD_UVOD;
          }
          break;

    case LCD_WAIT:
          if (execute)
          {
                execute = 0;
                prikazi_obvestilo_o_preverjanju_prijave();
          }
          break;
    }
```

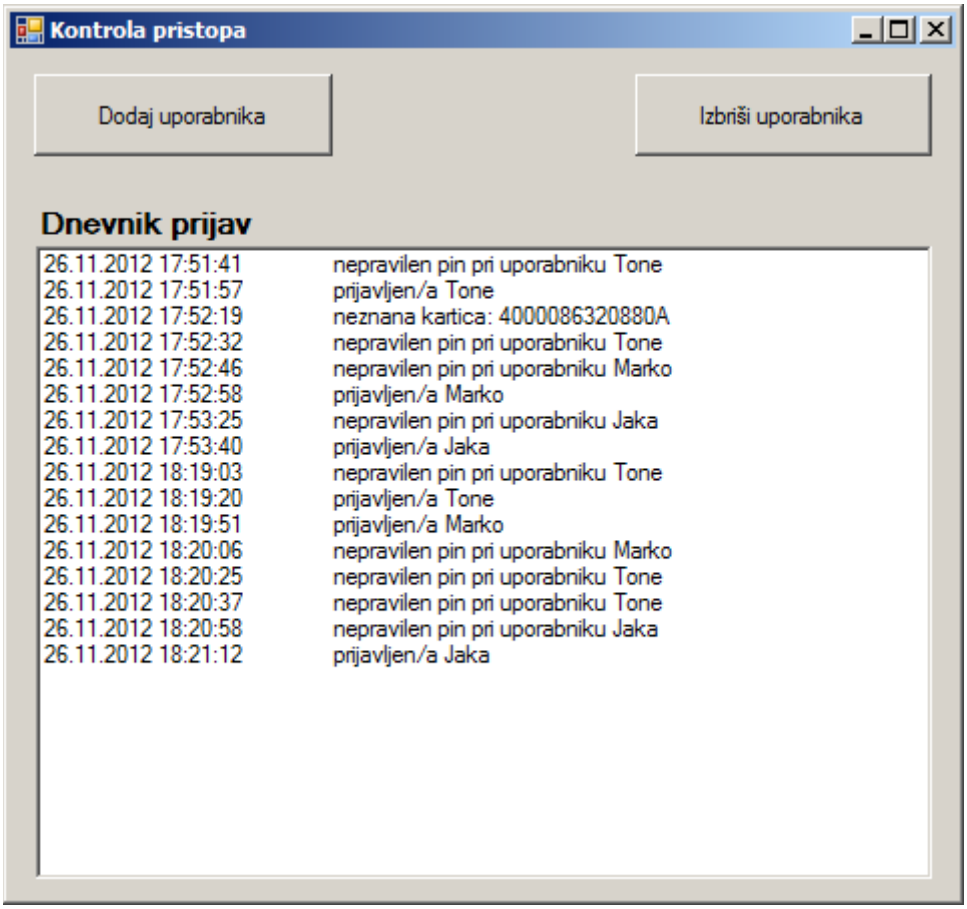
*Slika 54: Psevdokoda končnega avtomata*

### 4.2.9 Strežniški program

Za preverjanje številke kartice in kode PIN smo naredili strežniški program Kontrola pristopa. Program je napisan v programskem jeziku C# in za hranjenje podatkov uporablja podatkovno bazo Microsoft SQL Server Compact 3.5 [35].

Strežnik je sestavljen iz glavnega okna in dveh modalnih oken.

Na sliki 55 vidimo zaslonski izsek glavnega okna programa. Okno je sestavljeno iz dveh gumbov, enega za dodajanje uporabnika in enega za brisanje uporabnika, napisa Dnevnik prijav in seznama. Kot že ime samo pove, gumb »Dodaj uporabnika« odpre modalno okno za dodajanje uporabnika, gumb »Izbriši uporabnika« pa modalno okno, kjer brišemo uporabnike.



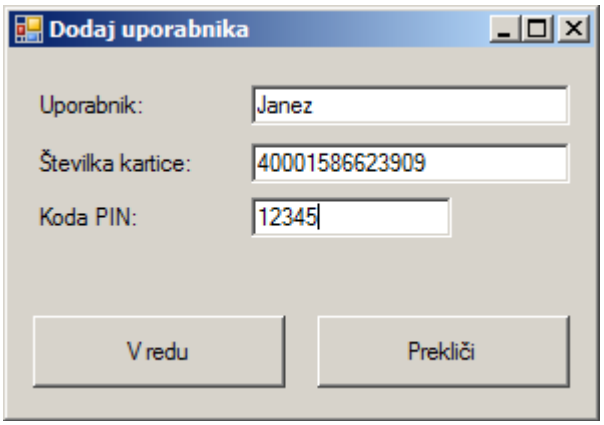

*Slika 55: Strežniški program Kontrola pristopa*

V seznam Dnevnik prijav na glavnem oknu se izpišejo vse poizvedbe vgrajenega sistema za preverjanje prijave, vključno s podatkom o času poizvedbe. Program izpiše različna obvestila:

- o nepravilni kodi PIN: tu pri obvestilu izpiše, pri katerem uporabniku je prišlo do napačne prijave;
- o neznani kartici: kadar kartice ni v sistemu, izpiše številko neznane kartice;

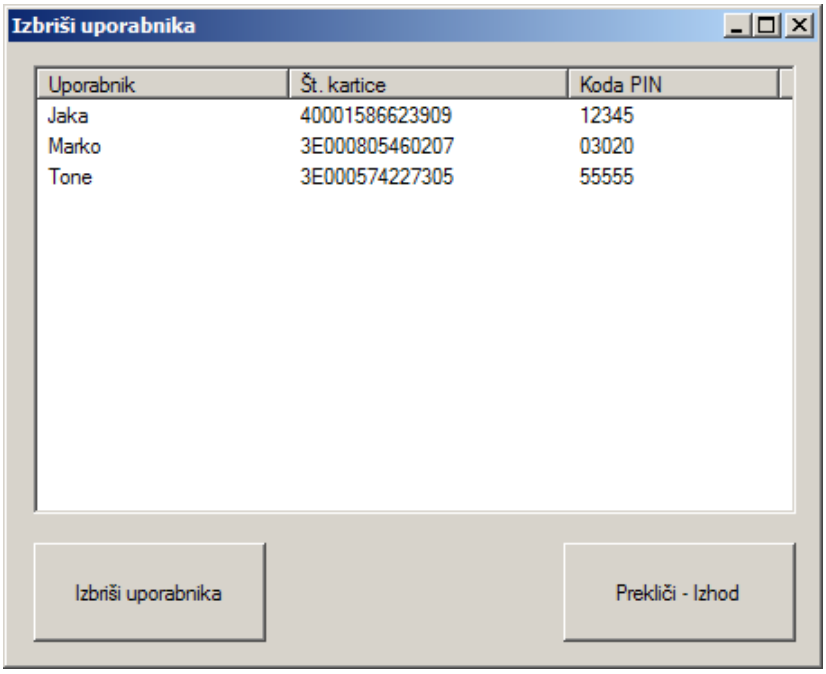
- o prijavi v sistem: kadar se številka kartice in koda PIN ujemata, takrat izpiše prijavljen/a in ime uporabnika.

Na sliki 56 vidimo modalno okno Dodaj uporabnika, ki se prikaže ob kliku na gumb »Dodaj uporabnika«. Skrbnik sistema tu vpiše novega uporabnika v sistem. Potrebno je vpisati ime uporabnika, številko kartice, s katero se bo uporabnik identificiral in kodo PIN. Polje za vnos številke kartice je omejeno na 14 znakov, polje za vnos kode PIN na 5 znakov.



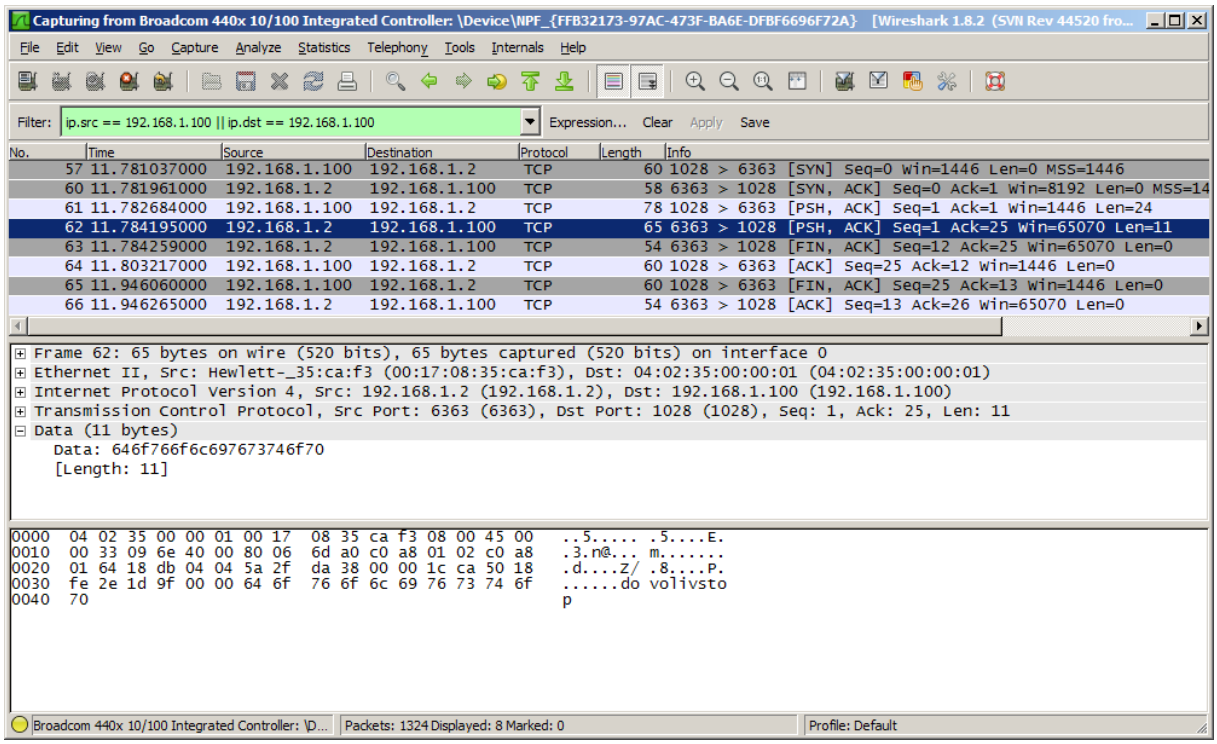

*Slika 56: Okno za vnos novega uporabnika*

Klik gumba »Izbriši uporabnika« na glavnem oknu odpre modalno okno Izbriši uporabnika (slika 57). Tu ima skrbnik sistema popoln pregled nad uporabniki, prikažejo se mu imena uporabnikov, številke kartic in kode PIN. Skrbnik sistema z izbiro uporabnika v seznamu in pritiskom na gumb »Izbriši uporabnika« izbriše uporabnika iz baze podatkov. S klikom na gumb »Prekliči - Izhod« zapre modalno okno.




*Slika 57: Okno za pregled in izbris uporabnikov*

Na sliki 58 je prikazan odgovor strežnika na preverjanje številke kartice in kode PIN, ko obstaja zapis v bazi podatkov – ob uspešni prijavi. Zajeli smo okno programa Wireshark [36], ki smo ga uporabili za analiziranje povezave med vgrajenim sistemom in strežniškim programom.




*Slika 58: Odgovor strežnika po preverjanju prijavnih podatkov*

# 5 Sklepne ugotovitve

Cilj diplomske naloge je bil izdelati vgrajen sistem za kontrolo dostopa, ki za delovanje uporablja tehnologijo RFID in preverja identiteto uporabnika dvostopenjsko. Kljub temu da je na trgu veliko različnih sistemov kontrole pristopa, ki uporabljajo čitalce RFID, v ponudbi nismo zasledili takih, ki bi za interakcijo z uporabnikom uporabljali zaslon na dotik. Tako smo dobili idejo, da bi v sistem vgradili zaslon na dotik, na katerem bi prikazovali obvestila in odčitavali kodo PIN.

Sistem, ki smo ga izdelali, spada po načinu delovanja med »on-line« sisteme, ker za delovanje potrebuje povezavo s strežnikom. Pomanjkljivost »on-line« sistemov je, da ne morejo delovati samostojno, zato ob neuspešni povezavi s strežnikom odpovejo. V takih primerih se za odpiranje vrat poslužujemo navadnih ključev.

Prednost izdelanega sistema za kontrolo dostopa v primerjavi s samostojnimi sistemi je, da ni omejitve števila uporabnikov in nimamo omejitve števila zapisovanj v pomnilnik, čeprav zgornje meje števila zapisovanj v življenjski dobi sistema zelo verjetno tudi pri samostojnih sistemih nikoli ne dosežemo (pomnilniki Flash, EEPROM (ang. *Electrically Erasable Programmable Read-Only Memory*)).

V strežniku smo implementirali le najosnovnejše funkcionalnosti sistemov za kontrolo dostopa. To so: dodajanje, pregled, brisanje uporabnikov in spremljanje uporabe sistema. Na strežnik lahko povežemo več vgrajenih sistemov hkrati, s tem omogočimo skupini uporabnikov, ki so v bazi, da dostopajo do več prostorov. Trenutna verzija strežnika še ne razlikuje med napravami, čeprav vsaka naprava skupaj s prijavnimi podatki pošlje tudi identifikacijsko številko naprave. Z razlikovanjem naprav v strežniku bi lahko strežnik dopolnili tako, da bi za vsak prostor lahko določili nabor uporabnikov, ki jim je dostop omogočen, dodatna funkcionalnost bi bila nastavitev urnika uporabe sistema. Z nadaljnjim razvojem strežnika bi lahko sistem kontrole pristopa dokaj enostavno razširili še v sistem za registracijo delovnega časa.

Tako kot drugi, ima tudi naš sistem pomanjkljivosti. Kot smo že prej omenili, je sistem brez povezave s strežnikom neuporaben, saj deluje samo kot vmesnik za vnos prijavnih podatkov. Trenutno so nastavitve naprave vpisane v izvorni kodi vgrajenega sistema – zaenkrat ni mogoče spreminjati identifikacijske številke naprave in naslovov IP za napravo in za strežnik, tako da je sistem pogojno uporaben oziroma je potrebno ob spremembi nastavitev kodo še enkrat naložiti v napravo. Možna rešitev tega problema bi bila ta, da bi v vsako napravo dodali skrbniškega uporabnika, ki bi po pravilno vpisani kodi PIN imel možnost urejanja nastavitev naprave. Druga rešitev bi bila implementacija spletnega strežnika v napravi, kjer bi po uspešni prijavi imeli možnost spreminjanja nastavitev naprave, podobno kot je to narejeno v novejših modemih in ruterjih. Največja slabost našega sistema za kontrolo dostopa je ravno lažen občutek varnosti, ki jo želimo vnesti v okolje. Izkaže se, da je naš izdelek občutljiv na napad »mož v sredini« (ang. *Man-in-the-middle*), kjer lahko napadalec izve številko kartice in kodo PIN, in najpomembnejše, izve kakšen odgovor pošlje strežnik ob pravilni prijavi in kakšnega ob nepravilni. Pri kakršnikoli komunikaciji, kjer se prenašajo občutljivi podatki, bi morali uporabiti šifriranje povezave s protokolom TLS (ang. *Transport Layer Security*) ali SSL (ang. *Secure Sockets Layer*). Na internetu smo iskali dovolj majhno implementacijo katerega od omenjenih protokolov, vendar vse implementacije potrebujejo več sistemskih

virov, kot jih izbrani mikrokrmilnik ponuja. Tako smo ostali pri nešifrirani povezavi med napravo in strežnikom.

Zastavljeni cilj diplomske naloge smo uspešno uresničili in se pri tem naučili pisanja programov za mikrokrmilnike, ki uporabljajo zaslon na dotik in sklad uIP TCP/IP. Izdelek svoj namen opravlja zadovoljivo, vendar bi v primeru, da bi razvijali komercialni izdelek, obvezno morali uporabiti zmogljivejši mikrokrmilnik, ki bi omogočal uporabo šifrirnih protokolov.

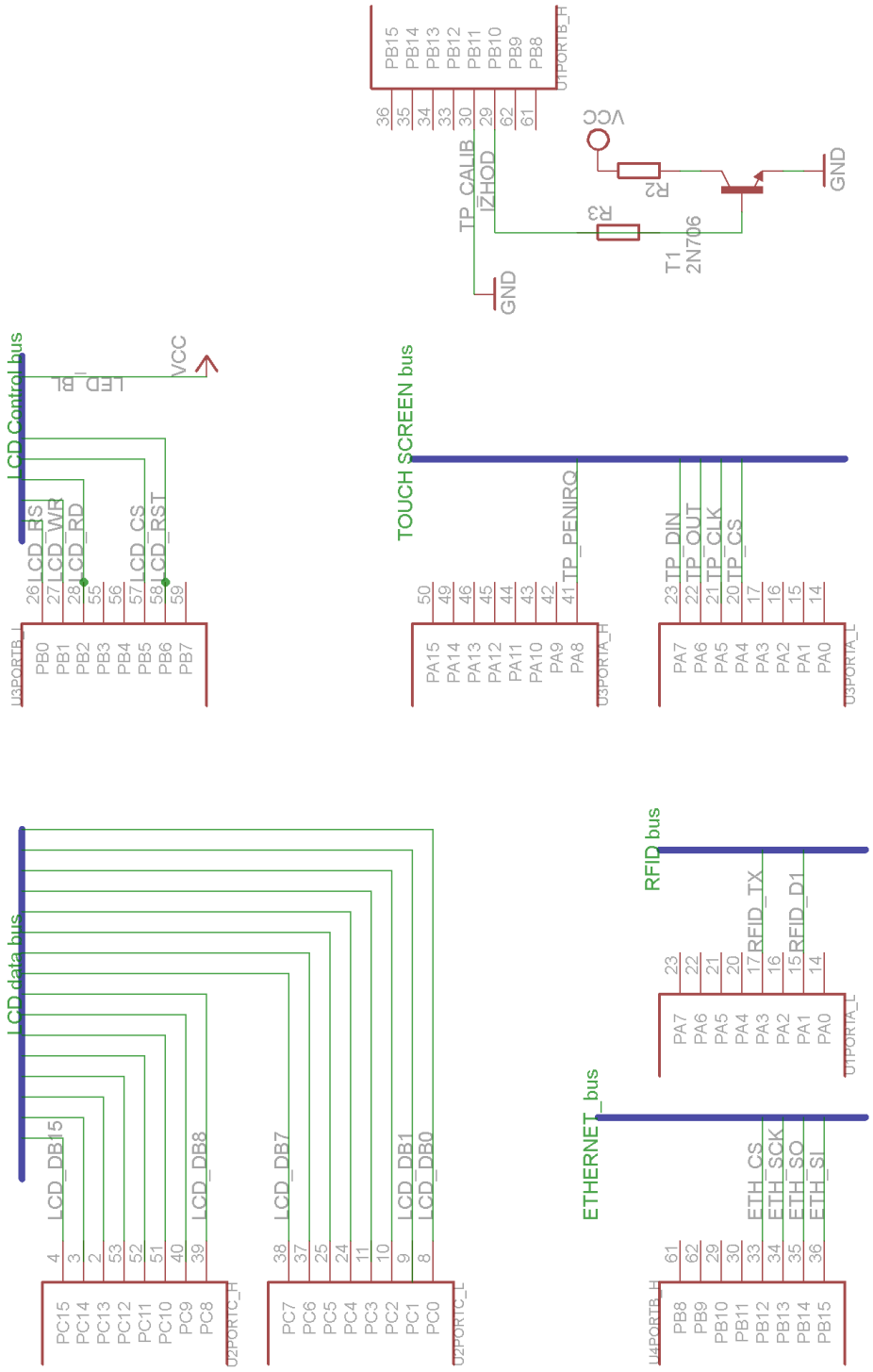
# A Inicializacija krmilnika LCD SSD1289

```
void LCD_Init()

{

      LCD_WriteReg(0x00,0x0001);

      LCD_WriteReg(0x03,0xA8A4);

      LCD_WriteReg(0x0C,0x0000);

      LCD_WriteReg(0x0D,0x080C);

      LCD_WriteReg(0x0E,0x2B00);

      LCD_WriteReg(0x1E,0x00B7);

      LCD_WriteReg(0x01,0x2B3F);

      LCD_WriteReg(0x02,0x0600);

      LCD_WriteReg(0x10,0x0000);

      LCD_WriteReg(0x11,0x6070);

      LCD_WriteReg(0x05,0x0000);

      LCD_WriteReg(0x06,0x0000);

      LCD_WriteReg(0x16,0xEF1C);

      LCD_WriteReg(0x17,0x0003);

      LCD_WriteReg(0x07,0x0233);

      LCD_WriteReg(0x0B,0x0000);

      LCD_WriteReg(0x0F,0x0000);

      LCD_WriteReg(0x41,0x0000);

      LCD_WriteReg(0x42,0x0000);

      LCD_WriteReg(0x48,0x0000);

      LCD_WriteReg(0x49,0x013F);

      LCD_WriteReg(0x4A,0x0000);

      LCD_WriteReg(0x4B,0x0000);

      LCD_WriteReg(0x44,0xEF00);

      LCD_WriteReg(0x45,0x0000);

      LCD_WriteReg(0x46,0x013F);

      LCD_WriteReg(0x30,0x0707);

      LCD_WriteReg(0x31,0x0204);
```

```
      LCD_WriteReg(0x32,0x0204);
      LCD_WriteReg(0x33,0x0502);
      LCD_WriteReg(0x34,0x0507);
      LCD_WriteReg(0x35,0x0204);
      LCD_WriteReg(0x36,0x0204);
      LCD_WriteReg(0x37,0x0502);
      LCD_WriteReg(0x3A,0x0302);
      LCD_WriteReg(0x3B,0x0302);
      LCD_WriteReg(0x23,0x0000);
      LCD_WriteReg(0x24,0x0000);
      LCD_WriteReg(0x25,0x8000);
      LCD_WriteReg(0x4f,0x0000);
      LCD_WriteReg(0x4e,0x0000);
      LCD_WriteReg(0x22, 0x0000);
}
```

*Slika 59: Inicializacija krmilnika LCD SSD1289*

# B Shema povezav




*Slika 60: Shema povezav med komponentami*

# Kazalo slik

Slika 1: Pasivne značke....7
Slika 2: Shema vezja pasivne značke (Vir: [4])....8
Slika 3: Postavitev elektrod na prevodnih plasteh pri zaslonu na dotik (Vir: [7])....10
Slika 4: Stik plasti ob dotiku zaslona na dotik (Vir: [7])....10
Slika 5: Prikaz branja koordinate X ob stiku prevodnih plošč....11
Slika 6: Prikaz branja koordinate X ob stiku prevodnih plošč, drugič....12
Slika 7: Shema delovanja vgrajenega sistema....13
Slika 8: Moduli, ki sestavljajo vgrajeni sistem....14
Slika 9: Razvojna plošča LC STUDIO z mikrokrmilnikom serije Cortex-M3....15
Slika 10: Shema vezave čitalca RFID z mikrokrmilnikom....17
Slika 11: Čitalec RFID - vezje in antena....18
Slika 12: Modul z zaslonom na dotik....18
Slika 13: 18- in 16- bitni način zapisa barve piksla....20
Slika 14: Shema vezave krmilnika LCD in mikrokrmilnika....20
Slika 15: Shema vezave krmilnika za zaznavanje dotikov z mikrokrmilnikom....22
Slika 16: Ethernet modul ENC28J60....22
Slika 17: Shema vezave krmilnika Ethernet z mikrokrmilnikom....23
Slika 18: Vklop ure za splošno-namenska vrata B in C....24
Slika 19: Konfiguriranje pinov za krmilnik za LCD....25
Slika 20: LCD – funkcija za pisanje v kontrolni register....26
Slika 21: Začetna funkcija za zapisovanje v prikazovalni pomnilnik....26
Slika 22: Funkcija za pisanje v prikazovalni pomnilnik....27
Slika 23: Funkcijo za zaključitev pisanja v prikazovalni pomnilnik....27
Slika 24: Funkcija za nastavitev naslova v pomnilniku....28
Slika 25: Vklop ure za splošno-namenska vrata A....28
Slika 26: Konfiguracija pinov krmilnika za zaznavanje dotika....29
Slika 27: Vklop ure napravi SPI1....29
Slika 28: Nastavitev naprave SPI1....30
Slika 29: Pošiljanje ukaza v krmilnik za zaznavanje dotika....30
Slika 30: Branje podatkov iz krmilnika za zaznavanje dotika....31
Slika 31: Branje koordinate X....32
Slika 32: Mehanska neporavnanost in skaliranje pri zaslonih na dotik....33
Slika 33: Konfiguriranje pinov za čitalec RFID....36
Slika 34: Nastavitev naprave USART2....37
Slika 35: Nastavitev naprave DMA1....38
Slika 36: Nastavitev krmilnikov EXTI in NVIC....39
Slika 37: Povezava naprave DMA1 s krmilnikom NVIC....40
Slika 38: Prekinitveni servisni program, ki se sproži ob prehodu signala D1 iz nizkega v visoko stanje....40
Slika 39: Prekinitveni servisni program, ki se sproži ob zaključku prenosa DMA naprave DMA1....41
Slika 40: Prikaz signala D1 (rumena barva) in signala TX (modra barva) čitalca RFID....42
Slika 41: Prikaz signala D1 (rumena barva) in TX (modra barva) pri branju brezkontaktne kartice....42
Slika 42: Pošiljanje paketa s krmilnikom za Ethernet....44
Slika 43: Glavna zanka sklada uIP, prvič....46

Slika 44: Glavna zanka sklada uIP, drugič.....47
Slika 45: Glavna zanka sklada uIP, tretjič.....48
Slika 46: Makro, ki zamenja UIP_APPCALL z eth_apcall.....48
Slika 47: Definicija strukture ethapp_appstate.....48
Slika 48: Omrežna aplikacija.....50
Slika 49: Pošiljanje prijavnih podatkov v strežnik.....50
Slika 50: Inicializacija sklada uIP.....51
Slika 51: Povezovanje na strežnik.....51
Slika 52: Diagram prehajanja med stanji za končni avtomat vgrajenega sistema.....52
Slika 53: Zaslonske maske za posamezna stanja avtomata.....53
Slika 54: Psevdokoda končnega avtomata.....56
Slika 55: Strežniški program Kontrola pristopa.....57
Slika 56: Okno za vnos novega uporabnika.....58
Slika 57: Okno za pregled in izbris uporabnikov.....58
Slika 58: Odgovor strežnika po preverjanju prijavnih podatkov.....59
Slika 59: Inicializacija krmilnika LCD SSD1289.....64
Slika 60: Shema povezav med komponentami.....65

# Literatura


[1] (2012) Radiofrekvenčna identifikacija. Dostopno na: http://sl.wikipedia.org/wiki/Radiofrekven%C4%8Dna_identifikacija

[2] (2012) The History of RFID Technology. Dostopno na: http://www.rfidjournal.com/article/view/1338

[3] (2012) Radio-frequency identification. Dostopno na: http://en.wikipedia.org/wiki/Radio-frequency_identification

[4] (2012) Microchip microID 125 kHz RFID System Design Guide. Dostopno na: http://ww1.microchip.com/downloads/en/devicedoc/51115f.pdf

[5] (2012) Touchscreen. Dostopno na: http://en.wikipedia.org/wiki/Touchscreen

[6] (2012) Resistive touchscreen. Dostopno na: http://en.wikipedia.org/wiki/Resistive_touchscreen

[7] (2012) PSoC 1 - Interface to Four-Wire Resistive Touchscreen. Dostopno na: http://www.cypress.com/?docID=33540

[8] (2012) Touch Screen Controller Tips. Dostopno na: http://www.ti.com/lit/an/sbaa036/sbaa036.pdf

[9] (2012) Touch Screen Controller, ADS7843 Datasheet. Dostopno na: http://www.ti.com/lit/ds/sbas090b/sbas090b.pdf

[10] (2012) STM32F103RB Datasheet. Dostopno na: http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00161566.pdf

[11] J. Yiu, *The Definitive Guide to the ARM Cortex-M3*, Oxford: Elsevier Inc., 2007, pogl. 8

[12] P. Bulić, Prosojnice za predmet Vgrajeni sistemi, 2012. Dostopno na: https://ucilnica.fri.uni-lj.si/course/view.php?id=39

[13] (2012) STM32F101xx Reference Manual. Dostopno na: http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/REFERENCE_MANUAL/CD00171190.pdf

[14] (2012) SSD1289 Datasheet. Dostopno na: http://www.kosmodrom.com.ua/el/STM32-TFT/SSD1289.pdf

[15] (2012) ENC28J60 Datasheet. Dostopno na: http://ww1.microchip.com/downloads/en/devicedoc/39662a.pdf

[16] (2012) OSI model. Dostopno na: http://en.wikipedia.org/wiki/OSI_model

[17] (2012) IAR Embedded Workbench for ARM. Dostopno na: http://www.iar.com/en/Products/IAR-Embedded-Workbench/ARM/

[18] (2012) MDK-ARM Microcontroller Development Kit. Dostopno na: http://www.keil.com/arm/mdk.asp

[19] (2012) Atollic TrueSTUDIO. Dostopno na: http://www.atollic.com/index.php/truestudio

[20] (2012) Raisonance Microcontroller, Ride7. Dostopno na: http://www.raisonance.com/~ride7__microcontrollers__tool~tool__T018:4cw36y8a5c39.html

[21] (2012) Hitex HiTOP IDE/Debugger. Dostopno na: http://www.hitex.com/index.php?id=hitop

[22] (2012) ULINK2 Debug Adapter. Dostopno na: http://www.keil.com/ulink2/

[23] (2012) Sourcery CodeBench Lite Edition. Dostopno na: http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/editions/lite-edition/

[24] (2012) Sourcery CodeBench Overview. Dostopno na: http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/overview/#one

[25] (2012) uVision IDE Overview. Dostopno na: http://www.keil.com/uvision/

[26] (2012) Knjižnice mikrokrmilnika. Dostopno na: http://www.st.com/internet/com/SOFTWARE_RESOURCES/SW_COMPONENT/FIRMWARE/stm32f10x_stdperiph_lib.zip

[27] (2012) STM32PLUS: ADS7843 Touch Screen Driver. Dostopno na: http://andybrown.me.uk/wk/2012/01/07/stm32plus-ads7843-touch-screen-driver/

[28] (2012) How To Calibrate Touch Screens. Dostopno na: http://www.embedded.com/design/configurable-systems/4023968/How-To-Calibrate-Touch-Screens

[29] W. Fang, T. Chang, "Calibration in touch-screen systems" *Analog Applications Journal*, 3Q 2007, str. 5-10, 2007

[30] (2012) Funkcije za kalibracijo zaslona na dotik. Dostopno na: https://bitbucket.org/akent/touchscreen-calibrate

[31] J. Yiu, *The Definitive Guide to the ARM Cortex-M3*, Oxford: Elsevier Inc., 2007, pogl. 7

[32] (2012) avr-uip: Port of uIP tcp/ip stack. Dostopno na: http://code.google.com/p/avr-uip/source/browse/trunk/drivers/?r=183#drivers%2Fenc28j60.tux

[33] (2012) The uIP Embedded TCP/IP Stack. Dostopno na: http://sourceforge.net/projects/uip-stack/

[34] (2012) The uIP 1.0 Reference Manual. Dostopno na: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.154.2510&rep=rep1&type=pdf

[35] (2012) Microsoft SQL Server Compact 3.5 Service Pack 2 for Windows Desktop. Dostopno na: http://www.microsoft.com/en-us/download/details.aspx?id=5783

[36] (2012) Program Wireshark. Dostopno na: http://www.wireshark.org/