UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Boštjan Luzar

RFID PRISTOPNA KONTROLA

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE

RAČUNALNIŠTVO IN INFORMATIKA

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: 00246/2012
Datum: 05.04.2012

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

Kandidat: **BOŠTJAN LUZAR**

Naslov: **RFID PRISTOPNA KONTROLA**
RFID ACCESS CONTROL

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Načrtujte preprost sistem RFID pristopne kontrole. Sistem naj bo zasnovan na uporabi mikrokrmilnika Atmel ATmega2560 ter 125 kHz pasivnih RFID značk.
Za razvoj sistema uporabite Arduino Mega ADK razvojno ploščico, razširitveno ploščico Arduino Ethernet Shield ter generični čitalnik RFID značk, ki za povezavo s krmilnikom uporablja protokol Wiegand26.
Sistem naj podpira omrežno integracijo, upravljanje dostopov preko spletnega vmesnika ter hranjenje dnevnikov na pomnilniški kartici SD.

Mentor:
prof. dr. Patricio Bulić

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Luzar Boštjan,

z vpisno številko 63070136,

sem avtor/-ica diplomskega dela z naslovom:

RFID PRISTOPNA KONTROLA

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek) 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 13.9.2012 Podpis avtorja/-ice: _____

Zahvala

Zahvaljujem se prof. dr. Patriciu Buliću za mentorstvo in pomoč pri izdelavi diplomske naloge.

Zahvaljujem se tudi staršema za podporo in priganjanje skozi vsa leta študija in vsem prijateljem in sošolcem za vse trenutke, ki smo jih med študijem doživeli skupaj.

Hvala!

Kazalo vsebine

Kazalo slik

Kazalo preglednic

Povzetek

Cilj diplomske naloge je bila seznanitev s postopkom razvoja aplikacij zasnovanih na mikrokrmilnikih s pomočjo razvojne platforme in IDE okolja Arduino. Skozi praktično delo v okolju Arduino smo realizirali logiko, ki lahko na podlagi 125 kHz RFID značk avtorizira dostope do posameznih lokacij oz. prostorov.

Na področju sicer obstaja veliko rešitev, vendar pa večina zaradi modularne zgradbe in načinov komunikacije zahteva obsežno strojno in programsko opremo. Izdelana rešitev na podlagi mikrokrmilnika, ki podpira omrežno integracijo, upravljanje dostopov preko spletnega vmesnika, hranjenje dnevnika dostopov na zunanji SD kartici pa predstavlja zaključeno celoto, ki je možno hitro in enostavno namesti na lokacijo.

Ključne besede: Arduino, mikrokrmilnik, RFID, pristopna kontrola, HTTP, NTP.

Abstract

The goal of the thesis was to learn about the procedure of developing applications based on microcontrollers using the Arduino development platform and the IDE environment. Through practical development in the Arduino environment we realized a logic which is capable to authorize access to specific locations and areas based on 125 kHz RFID tags.

Although many solutions exist, most of them require a lot of hardware and software because of their modular design and communication types, the solution realized based on a microcontroller which supports basic network integration capability, access management through a web interface and access audit trail on an external SD card represents a complete unit, which can be installed to a location fast and easy.

Keywords: Arduino, microcontroller, RFID, access control, HTTP, NTP**.**

1. Uvod

1.1. Pristopna kontrola

Pristopna kontrola [1] označuje sisteme, ki se uporabljajo za omejevanje dostopa do posameznih prostorov. Tehnologija in struktura sistemov je različna, vendar imajo vsi sistemi ob posameznih končnih točkah (vratih) podoben zaključek; razpoznavno (čitalno) enoto in električno ključavnico. Poznamo različne razpoznavne enote na podlagi zvočne ali vizualne identifikacije, identifikacije s kodo, najbolj pa so razširjene enote, pri katerih se identificiramo s kartico.

Sisteme pristopne kontrole je smiselno uporabiti v prostorih, kjer se giblje več uporabnikov, oz. v prostorih, do katerih dostop se omejuje in jih je potrebno nadzirati. Z uporabo sistema pristopne kontrole praviloma v primerjavi z uporabo ključev zagotavljamo enostavnejši in hitrejši dostop, višji nivo varnosti in nadzora. V večini sistemov imamo poleg omejitve posameznih dostopnih pravic tudi možnost časovne omejitve dostopov posameznika, ter možnost hranjenja zgodovine prehodov skozi prehode. Razpoznavne kartice, uporabljene pri razvoju izdelka, s frekvenco delovanja 125 kHz, kot podatek hranijo enolično razpoznavno kodo (enolična tovarniško definirana koda), na podlagi katere se izvaja avtorizacija posameznikov v sistemu. Isto kartico se lahko uporabi tudi v ostalih sistemih, ki so velikokrat prisotni poleg sistemov pristopne kontrole (registracija delovnega časa, plačevanje s kartico,…). Značke, delujoče na frekvenci 125kHz, ne vsebujejo dodatnega pomnilnika v katerega bi bilo mogoče vpisovati podatke.

Novejše kartice, ki delujejo na frekvenci 13.56 MHz, so opremljene s kompletno procesno enoto, notranjim bralno pisalnim pomnilnikom, zato hranijo večje količine podatkov, kar karticam še povečuje možnosti za uporabo. Sodobnejše kartice večinoma vsebujejo tudi dodaten procesor, ki je optimiziran za področje kriptografije in omogočajo vzpostavitev šifrirane komunikacije med kartico in čitalnikom. Razdelitev internega pomnilnika na sektorje, ki so lahko zakodirani z ločenimi ključi, omogoča varno in neodvisno uporabo v okolju, kjer se kartica uporablja v več aplikacijah, ter hranjenje podatkov občutljive vsebine.

Ključavnice in vrata v sistemih so večinoma zasnovana tako, da ob odpovedi sistema ali napajanja omogočajo dostop s ključi, izhodi iz prostorov pa praviloma niso omejeni zaradi varnosti uporabnikov ob katastrofah oz. nesrečah.

1.2. Zgradba tipičnega sistema pristopne kontrole

Tipična zgradba sistema pristopne kontrole, zasnovanega na RFID tehnologiji :

1) Električni prejemnik
   Večina končnih točk v sistemu pristopne kontrole je opremljeno s t.i. električnim prejemnikom, nameščenim v okovje vrat. Razvojna ploščica uporabljena v diplomski nalogi je za direktno krmiljenje klasičnih ne-impulznih prejemnikov prešibka, zato je za krmiljenje prejemnika potrebno na izhod mikrokrmilnika dodati še ustrezno ojačevalno

vezje za pretvorbo TTL napetostnega signala v ustrezen močnostni signal (običajno se uporablja enosmerna napetost 12V s tokovno zmogljivostjo okoli vrednosti 1A).

2) Čitalnik kartic
   Čitalnik kartic je zadolžen za branje razpoznavnih kartic (značk) in sporočanje vrednosti modulom višje v hierarhiji. V diplomski nalogi smo zaradi razpoložljivosti uporabljali čitalnik in značke s frekvenco delovanja 125kHz. Izbrani generični čitalnik odčitano kodo kartice posreduje skladno z Wiegand26 protokolom [4] .

3) Odobritvena enota
   Odobritvena enota poleg procesne enote vsebuje tudi pomnilnik, v katerem hrani pravice dostopa za dodeljene čitalnike. Ko enota prejme prebrano kodo kartice od čitalnika, glede na vsebino pravic v svoji odobritveni tabeli za posamezni čitalnik dogodek zabeleži in ga sporoči strežniku. Nato ob ustreznih dostopnih pravicah aktivira električni prejemnik in s tem uporabniku omogoči odpiranje vrat. Praviloma so sodobne odobritvene enote vključene v LAN (Local Area Network) omrežje, s čitalniki pa komunicirajo preko RS232/485 vodila.
   Sodobne odobritvene enote za vse svoje podrejene čitalne enote vsebujejo tabelo kartic z dostopnimi pravicami in urniki. Tako uporabniki ob izpadu strežnikov izpada direktno ne občutijo. Prav tako večina odobritvenih enot omogoča beleženje dogodkov na čitalniku.

4) Strežniki
   Na strežnikih povprečnega sistema pristopne kontrole poleg servisov programa, praviloma teče še podatkovna baza, ki hrani vse informacije o nastavitvah strojne opreme, uporabniške dostopne pravice in dnevnike dogodkov. Za normalno upravljanje inteligentne enote so sistemi pristopne kontrole opremljeni z namensko uporabniško aplikacijo. Rok hranjenja informacij o gibanju zaposlenih v podjetjih je zakonsko določen, zato je potrebno poskrbeti za varnostno kopiranje in brisanje starejših podatkov v sistemu.

Če želimo postaviti manjši sistem, oziroma obstoječi sistem s tako zapleteno hierarhijo razširiti na drug objekt, se nam nabere veliko število lokacij, novih povezav in težav, kar lahko predstavlja preveliko finančno investicijo, da bi se tak poseg izplačal.

V diplomski nalogi smo želeli preskočiti vse korake, ki so za implementacijo manjšega sistema moteči in smo se odločili, da so nujno potrebni le naslednji elementi: električni prejemnik, čitalnik in mikrokrmilnik z razširitvami. Čeprav se preprosta rešitev z zmogljivostmi večjih sistemov zagotovo ne more primerjati, bi se lahko tak tip rešitve v posameznih primerih finančno in funkcionalno izkazal kot zadovoljivo nadomestilo.

V praksi bi to lahko pomenilo drastično zmanjšanje kompleksnosti celotnega sistema, saj bi se lahko izognili postavitvi celotne hierarhije strojne opreme (celotno hierarhijo razen strežnika je potrebno postaviti v vsakem objektu), medtem ko za postavitev enote iz diplomske naloge (ali podobnega tipa) enoto enostavno povežemo v LAN omrežje.

Enota ob inicializaciji prebere konfiguracijo in pravice dostopa iz SD (Secure Digital) Flash pomnilniške kartice, zato bi jo lahko za oddaljene točke, kjer se ne izvaja sprememb v dostopni politiki uporabili samostojno brez omrežnega dostopa. V tem primeru bi onemogočili sinhronizacijo časa preko NTP (Network Time Protocol) protokola, ter se odpovedali upravljanju s pravicami preko spletnega vmesnika.

2. RFID

RFID (Radio Frequency IDentification) [2] je tehnologija, ki omogoča identifikacijo RFID značke preko radijskih valov. RFID značka je elektronski čip z enolično identifikacijsko številko, ki definira posamezno identiteto, ki uporablja sistem npr.: osebo, žival, predmet.

RFID čitalnik kartic (kombinacija oddajnika in sprejemnika) neprekinjeno oddaja in prejema povratni signal. V prisotnosti RFID značke prebrani signal na čitalniku ni enak oddanemu, razlika (rezultat modulacije) med signaloma pa označuje enoličen zapis kartice.

Osnovni signal, ki ga oddajnik neprekinjeno pošilja, je sinusoida določene frekvence (nosilni signal). Ko značka, prisotna v oddajnem polju čitalnika, sprejme dovolj energije (indukcija), se aktivira in prične z modulacijo nosilnega signala glede na binarno identifikacijsko kodo, ki je zapisana v spominskem polju. Naslednja naloga čitalnika je demodulacija prejetega signala. Kot pri vseh brezžičnih tehnologijah je potrebno ustrezno poskrbeti tudi za algoritme proti trkom, če na komunikacijskem kanalu hkrati oddaja več naprav (prisotnost dveh ali več kartic).

Čitalniki praviloma ne vsebujejo informacij o pravicah odobritev, zato so opremljeni le z izhodi, kjer ustrezno prebrano kodo kartice oddajo naslednjemu členu v hierarhiji.

2.1. RFID značke

RFID značke označujejo kombinacijo mikročipa, ki v pomnilniku hrani razpoznavno kodo kartice in zunanje antene, zaprte v kompaktnem ohišju. Velikost značk močno variira. Najmanjše so v rangu velikosti riževega zrna, največje in najbolj pogoste pa srečamo v obliki identifikacijskih kartic (Slika 1). Značke se poleg ugotavljanja identitete živali, ljudi, vozil in paketov uporabljajo tudi za spremljanje inventarja.

Slika 1: Različne oblike pasivnih RFID značk [[12],[13],[14],[15],[16]]

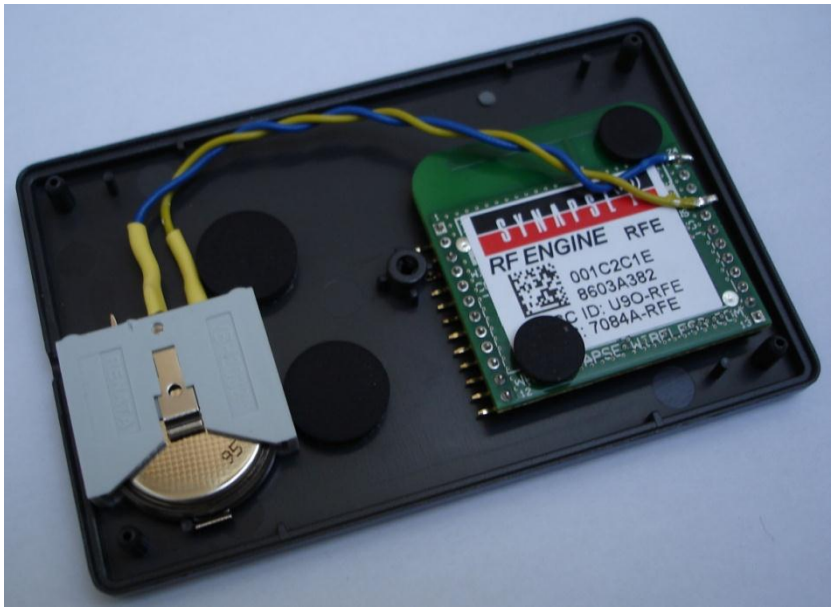
2.1.1. Aktivne in pasivne RFID značke

Aktivne RFID značke poleg mikročipa in antene vsebujejo tudi baterijo. Nekaterim izmed njih je možno baterijo tudi zamenjati. Večina jih je hermetično zaprtih, njihova življenjska doba je omejena s trajanjem baterije in večinoma znaša nekaj let. Stanje baterije značke se pri večini modelov oddaja hkratno s kodo kartice, kar lahko pripomore k pravočasni zamenjavi kartice. V hladnejših razmerah je enota zaradi upočasnjenih kemičnih procesov v bateriji lahko neaktivna.

Koda aktivne kartice se oddaja v predhodno določenem intervalu, ki je običajno okoli ene sekunde. Domet aktivnih kartic je v primerjavi s pasivnimi neprimerljivo boljši, do nekje 30m, medtem ko lahko s pasivnimi značkami dosegamo razdalje le do nekje 15cm. Žal se to zelo pozna na ceni značke, ki je v večinoma vsaj 10-20x višja od pasivne različice.

Dimenzije aktivnih značk so predvidoma večje od pasivnih, večinoma je razlika najbolj opazna pri debelini kartice (Slika 2).

Obstajajo aplikacije, kjer aktivne in pasivne značke nastopajo kot par in je za ustrezno avtorizacijo potrebna hkratna prisotnost aktivne in pasivne značke.

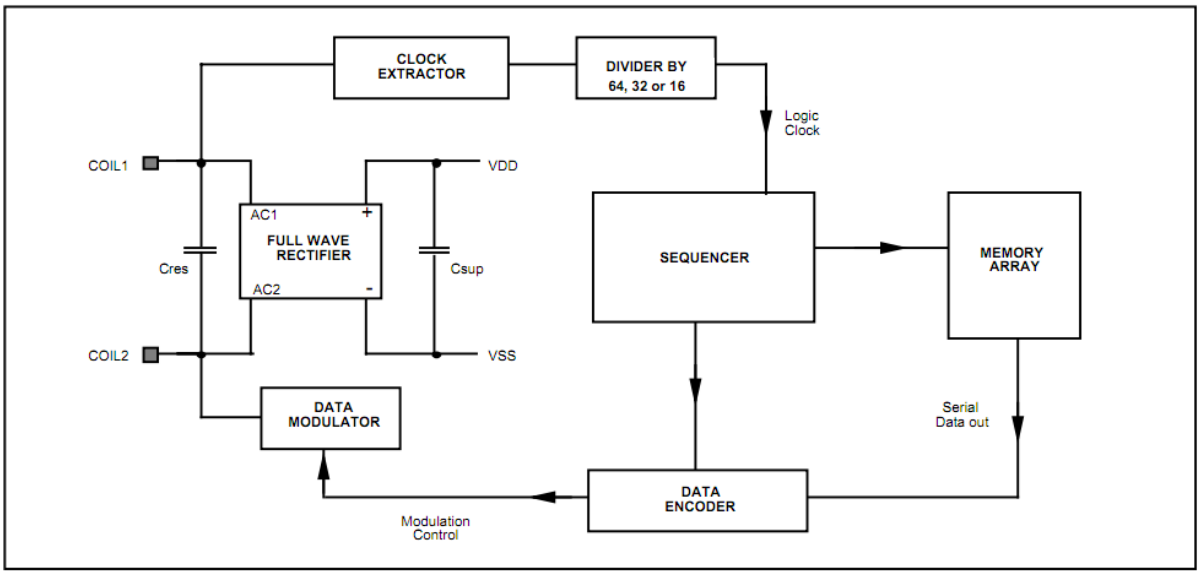


Slika 2: Primer RFID aktivne značke, ki omogoča zamenjavo baterije [17]

2.1.2. EM Marin H4102

Značke, ki smo jih uporabljali pri izdelavi naloge, so zasnovane okoli CMOS (Complementary metal–oxide–semiconductor) integriranega vezja EM Marin H4102 [3]. Gre za enega izmed najbolj razširjenih tipov pasivne značke, kjer se vezje, prisotno v magnetnem polju čitalnega mesta, napaja z energijo, inducirano preko zunanje tuljave - antene kartice. Napajalna napetost se usmeri s kombinacijo polnovalnega usmernika (full wave rectifier) in kondenzatorja (Slika 3). Tovrstne 125kHz značke so omejene s pomnilnikom, v katerega je možno zapisovati samo enkrat

(tovarniški vpis enolične kode kartice) in ne omogočajo hranjenja drugih podatkov. Prav tako ne podpirajo kodirane komunikacije s čitalnikom.



Slika 3: Blokovna shema pasivne RFID kartice zasnovane okoli vezja EM Marin H4102 [3]

Generični RFID čitalnik, uporabljen v diplomski nalogi, kot vsi ostali RFID čitalniki pasivnih kartic neprekinjeno oddaja signal, sinusoido osnovne frekvence 125kHz. V kartici se na podlagi induciranega sinusnega nihanja določi urin takt, katerega se za potrebno višjo nadaljnjo uporabo v postopku procesiranja v sami kartici razdeli z delilnikom faktorja 16, 32 ali 64.

Nadaljnja logika, sestavljena iz sekvenčnika (sequencer) in kodirnika (data encoder), skrbi, da iz pomnilnika (memory array) v kodirnik (data encoder) prehajajo ustrezni podatki (ID kode kartice).

V kodirniku se podatki kodirajo po enem izmed treh podprtih algoritmov kodiranja: Manchester, Biphase ali PSK. Ko se zaključi kodiranje zadnjega (64.) bita, se proces ponovi s prvim bitom v pomnilniku, vse dokler je v vezju inducirana napajalna napetost.

Izhod iz kodirnika določa frekvenco, s katero modulator modulira signal oddan preko zunanje antene.

Naslednja naloga čitalnika je, da iz induciranega signala čitalnika demodulira kodo kartice.

2.1.3. Wiegand26 protokol

Generični čitalnik, ki smo ga uporabili pri izdelavi diplomske naloge, sporoča kodo kartice preko komunikacijskega protokola tipa Wiegand. Standard definira vmesnik, ki je uveljavljen že od leta 1980.

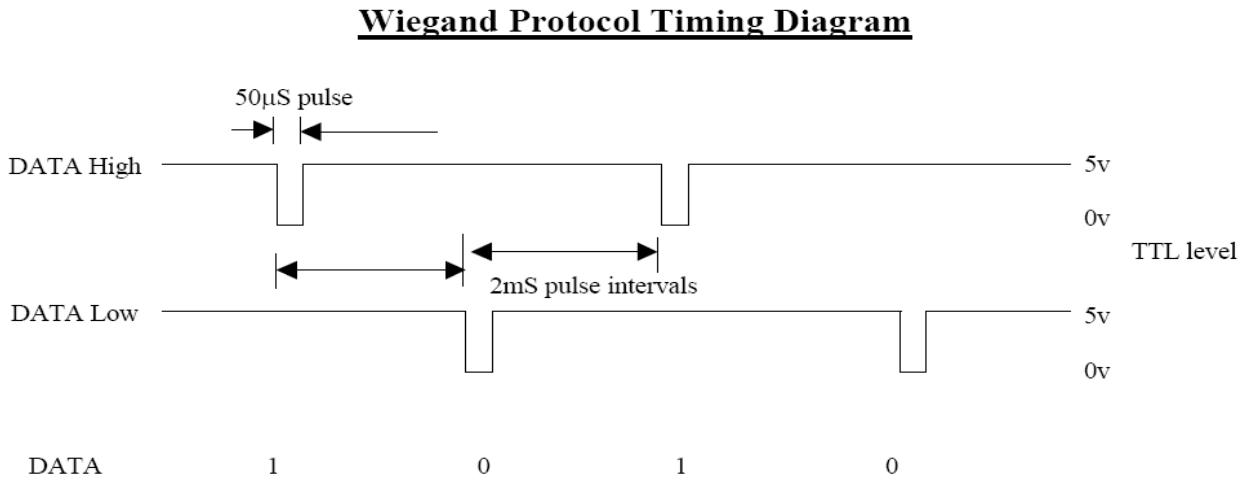
V vmesniku sta definirana dva aktivno nizka (active low) digitalna izhoda TTL napetostnih nivojev: DATA0 in DATA1. Nizka vrednost na izhodu DATA0 predstavlja binarni podatek tipa 0, nizka

vrednost na liniji DATA1 predstavlja binarni podatek tipa 1. Prepovedano je hkratno aktivno stanje obeh izhodov [4].

Čitalnik posamezno binarno mesto šifre kartice (0 ali 1) predstavi z nizkim pulzom na ustreznem izhodu. Po zaključenem pošiljanju celotne kode kartice čitalnik počaka približno sekundo pred ponovnim pošiljanjem šifre kartice po komunikacijskem vmesniku. Ob hitrem izmenjevanju kartic se sekundna zakasnitev ne upošteva in se kode posredujejo brez zakasnitve .

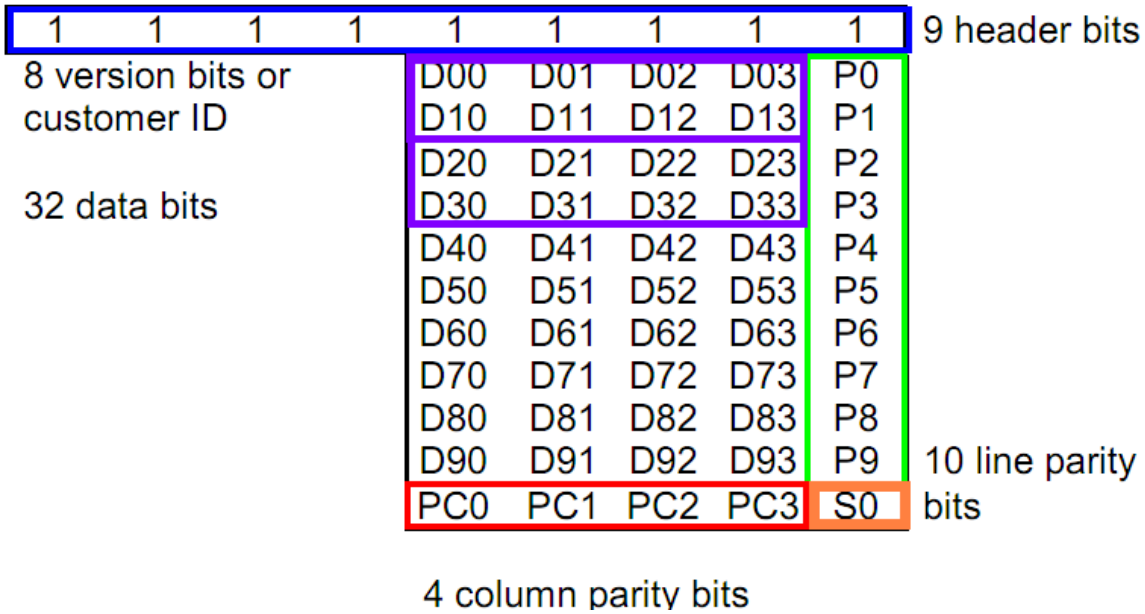
Zakasnitve v protokolu niso strogo definirane (oz. jih različna literatura navaja drugače), v grobem pa naj bi bil čas pulza krajši od 100us, interval med pulzi pa naj bi bil v rangu nekaj ms (Slika 4).

Ob prisotnosti dveh ali več značk v dosegu čitalnika, čitalnik ustrezno reagira in ne posreduje nobene izmed identifikacijskih kod kartic.



Slika 4: Wiegand protokol – fizični nivo [4]

Ob prvih uspešnih branjih kod kartic, ki smo jih na Arduino prejeli od čitalnika, smo bili presenečeni, da je njihova dolžina znašala 26 bitov. Glede na dokumentacijo čipa EM Marin H4102 smo pričakovali kodo dolžine 8 bajtov. Po natančnem pregledu dokumentacije smo zaključili, da je odčitana vrednost pravilna.



Slika 5: Zapis ID kode kartice na RFID znački H4102 [3]

Čitalnik iz prebranega moduliranega podatka po končanem branju odstrani podatke (Preglednica 1), skladno s protokolom Wiegand 26.

Ang. oznaka	Dolžina	Namen
Header bits	9 bitov	9 uvodnih bitov (vrednost=1), začetek prenosa, kombinacija se znotraj podatkovnih bitov zaradi paritete ne more ponoviti
Line parity	10 bitov	Pariteta posamezne vrstice
Column parity	4 biti	Stolpčna pariteta
S0	1 bit	Stop bit (vrednost=0)
Prva 2 bajta podatka (MSB)	16 bitov	Ker smo uporabili čitalnik, ki deluje po protokolu Wiegand26, se iz podatkovnih bitov odstrani 2 bajta, z največjo težo podatkov MSB (Most Significant Bit). To lahko storimo iz praktičnega razloga, ker je možnost, da bi s tem vplivali na rezultat, zelo majhna. Ostane nam 24 bitov podatka s strani LSB (Least Significant Bit), s čimer lahko predstavimo čez 16 milijonov (2^24) enoličnih kombinacij oz. RFID značk. Seveda obstaja možnost, da bi prebrali več kartic z isto 24 bitno kodo (drugačni MSB biti), vendar je verjetnost v povprečnih sistemih zanemarljiva.

Preglednica 1: Pridobivanje ID kode iz pomnilnika kartice

Uporabnim 24 bitom informacije prebrane iz kartice se doda še dva paritetna bita, na prvo MSB ter na zadnje LSB mesto. MSB paritetni bit je pariteta prvih dvanajstih bitov informacije, bit na LSB mestu pa predstavlja pariteto za ostale podatkovne bite (Slika 5). Paritete pri branju kode v programu mikrokrmilnika nismo preverjali in smo jo obravnavali kot običajen podatkovni bit. Tako so veljavne kode znotraj programa dolge 26 bitov.

Wiegand vmesnik zaradi razmeroma dolgih zakasnitev omogoča relativno dolge podatkovne linije. Različna literatura navaja zanesljivo delovanje tudi na razdaljah, ki presegajo 100m.

3. Arduino

3.1. Razvojni ploščici Arduino Uno in Arduino Mega ADK

Za razvoj izdelka smo uporabljali dva različna tipa razvojne ploščice Arduino Uno in Arduino Mega ADK [6]. Razvoj smo zaključili na ploščici Arduino Mega ADK. Kompatibilnosti programa s ploščico UNO zaradi pomanjkanja statičnega SRAM pomnilnika v celoti nismo zagotovili. Primerjava osnovnih parametrov obeh razvojnih ploščic je podana v spodnji preglednici (Preglednica 2).

Parameter	Arduino Mega ADK	Arduino Uno
Urina frekvenca	16 MHz	16 MHz
Število splošnih digitalnih vhodov/izhodov	54	14
Število digitalnih vhodov/izhodov, ki podpirajo PWM izhode	15	6
Število digitalnih vhodov, ki podpirajo analogno -> digitalno pretvorbo	16	6
Količina Flash pomnilnika	256 KB	32 KB
Količina statičnega SRAM pomnilnika	8 KB	2 KB
Količina EEPROM pomnilnika	4 KB	1 KB
Število serijskih portov (UART)	4	1

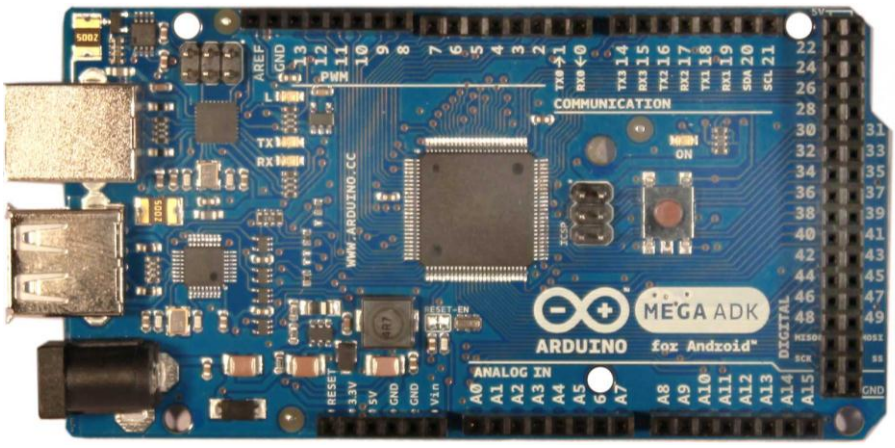
Preglednica 2: Osnovni podatki Atmel krmilnikov - integriranih na razvojni ploščici

Razvojna ploščica Arduino Mega ADK (Slika 6) je zasnovana na obojestranski tiskanini okoli Atmel ATmega2560 mikrokrmilnika. Tiskanina se lahko napaja z zunanjim virom enosmernega napajanja s priporočeno napetostjo 7-12V (izjemoma se kratkotrajno lahko uporabi tudi napetost 7-20V), ali pa se mikrokrmilnik napaja direktno mimo regulatorja napetosti preko USB vrat (5V DC).

Razvojna ploščica je opremljena tudi z naslednjima čipoma:

Atmel ATmega8U2: mikrokrmilnik, ki je programiran, da deluje kot pretvornik med USB in serijsko povezavo (komunikacija med IDE okoljem in osrednjim mikrokrmilnikom).

MAX3421e: vezje, okoli katerega je realiziran gostiteljski USB vmesnik za povezavo z Android telefoni.



Slika 6: Arduino Mega ADK razvojna ploščica [18]

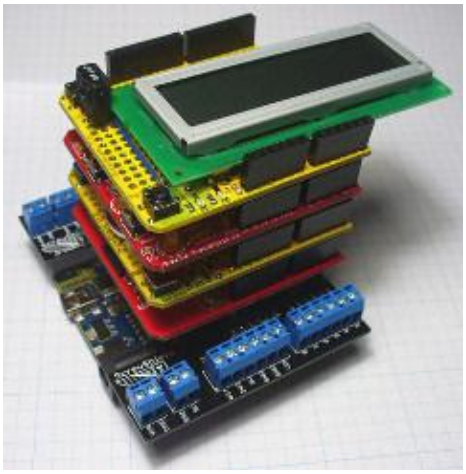
3.2. Logična razdelitev funkcij vezja Arduino Mega ADK

Vezje je logično razdeljeno po družini posebnih funkcij, katere opravlja posamezen sklop vhodov/izhodov:

- Napajalni sklop
  - GND
  - Stabiliziranih 5V DC
  - Stabiliziranih 3.3V DC
- Analogni sklop – vhodi/izhodi zmožni izvajanja 10 bitne analogno-digitalne pretvorbe (1024 vrednosti) med dvema napetostima; ponavadi v razponu 1.1V, 2.56V, 5V oz. med 0V in vrednostjo, ki je definirana na AREF pinu
- Digitalni sklop - (klasični digitalni pini z dodatnimi funkcionalnostmi):
  - PulseWidthModulation – 8 bitni PWM izhodi
  - External Interrupts – vhodi z možnostjo detekcije (zunanjih) prekinitev, aktivacija ob različnih tipih dogodka:
    - rising edge (prehod iz digitalne 0 v digitalno 1)
    - falling edge (prehod iz digitalne 1 v digitalno 0)
    - state change (sprememba stanja tipa rising edge ali falling edge)
    - low (zaporedno proženje glede na interval ob nizkem stanju na pinu)
- Komunikacijski sklop
  - Serijska komunikacija (Universal Asynchronous Receiver/Transmitter)
  - USB gostitelj (MAX3241E)
  - SPI komunikacija definirana preko integrirane programske »SPI_library« knjižnice (SS=pin53,MOSI=pin51,MISO=pin50,SCK=52).
  - Two Wire Interface (TWI)
  - Led dioda

3.3. Razširljivost

Arduino je zasnovan modularno. Zasnova omogoča uporabniku poljubno dopolnjevanje razvojne ploščice s t.i. rezinami (shields) [7]. Rezine so zasnovane tako, da omogočajo enostaven priklop, so enostavne in poceni za izdelavo. Ker nadzor in komunikacija z rezinami večinoma poteka preko SPI vodila, lahko nekatere rezine uporabljamo (fizično) sočasno (Slika 7). Za pravilno delovanje je potrebno poskrbeti za aktivacijo SPI komunikacije izbrane rezine (4.1).



Slika 7: Rezine - razširitve osnovne razvojne ploščice [19]

3.4. Bootloader

Za delo preko Arduino okolja je potrebno na glavni mikrokrmilnik v Flash pomnilnik naložiti bootloader - kratek program, ki skrbi za nalaganje kode (skic) iz programskega okolja Arduino. Tip bootloaderja je definiran z izbiro glavnega mikrokrmilnika razvojne ploščice.

Mikrokrmilniki, dobavljeni v paketu z uradno prototipno ploščico Arduino, so že pred-programirani z ustreznem bootloaderjem. Kompatibilne mikrokrmilnike s predhodno nameščenim bootloaderjem lahko programiramo tudi brez razvojne ploščice preko poljubnega tipa ICSP (In-Circuit Serial Programming) programatorja.

4. Fizične povezave

4.1. Serijski periferni vmesnik (Serial Peripheral Interface)

SPI [8] je sinhronski (enoten urin signal si delijo vse naprave na komunikacijskem kanalu), relativno preprost serijski prenos podatkov. Večinoma ga uporabljamo za komunikacijo med krmilnikom in napravo. Lahko ga uporabimo tudi za komunikacijo med dvema krmilnikoma. SPI komunikacija vedno določa gospodarja in sužnja (master-slave) in poteka po treh podatkovnih linijah:

- SCLK (Serial Clock) – linija, namenjena prenosu skupnega urinega signala,
- MISO (Master In Slave Out) – linija, namenjena prenosu podatkov od sužnja proti gospodarju,
- MOSI (Master Out Slave In) – linija, namenjena prenosu podatkov od gospodarja proti sužnju.

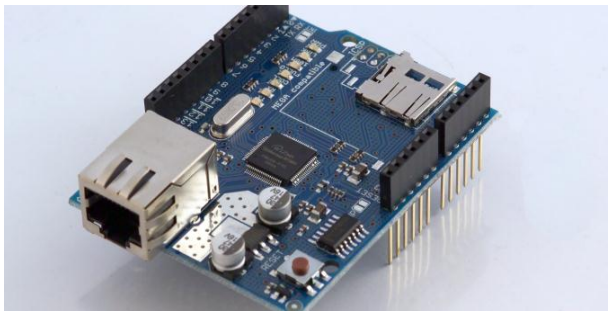
Za uspešno SPI komunikacijo je potrebno paziti pri:

- definiciji vrstnega reda podatkov (po komunikacijskem kanalu se najprej prenese najbolj pomemben (MSB) bit ali najmanj pomemben (LSB) bit),
- določitvi faze ure (sinhronizacija obeh naprav na isti dogodek, začetek urinega cikla (rising-edge) oz. konec urinega cikla (falling-edge),
- definiciji hitrosti komunikacije (delilnik ure), tukaj je potrebno paziti tudi na čas zajema ure (zajemanje stanja ure, ko je stanje ure stabilno),
- določitvi polaritete sinhronizacijske ure (izbira neaktivnega stanja ure; dogovor ali je ura aktivna v nizkem ali visokem stanju).

4.2. Rezina Arduino Ethernet (Arduino Ethernet shield)

Arduino lahko s pomočjo Ethernet rezine [10] povežemo v lokalno omrežje. Modul je zasnovan okoli WizNet W5100 čipa [9], ki je bil razvit za enostavno implementacijo 10Mbit ali 100Mbit omrežne povezave za vgrajene (embedded) naprave. W5100 je razvit v skladu z IEEE 802.3 10BASE-T in 100BASE-TX standardoma. Trdo-ožičeni sklad podpira TCP, UDP, IPV4, ICMP, IGMP, ARP in PPPoE protokole.

Modul kot večino ostalih razširitev na osnovno tiskanino preprosto nataknemo, ne potrebuje dodatnega napajanja, z glavnim mikrokrmilnikom pa komunicira preko SPI vodila. Tiskanina je opremljena s standardnim RJ45 ženskim priključkom (Slika 8). Kot pri vseh uradnih Arduino razširitvah so vse sheme, dokumentacija in programske knjižnice razširitvene rezine javno dostopne.



Slika 8: Razširitvena rezina Arduino Ethernet Shield [21]

Pri razvijanju si lahko pomagamo tudi s skupino informativnih LED diod, implementiranih na rezini:

- PWR – prisotnost napajanja,
- LINK – vzpostavljena omrežna povezava,
- FULLD – indikator za tip povezave (duplex, simplex),
- 100M – indikator za 100M (ali 10M) povezavo,
- RX – indikator prejemanja podatkov,
- TX – indikator oddajanja podatkov,
- COLL – indikator omrežnih trkov (kolizij).

Rezina Ethernet je opremljena tudi z režo, ki podpira MicroSD/SDHC Flash pomnilniške kartice. Za dostop do datotek in nadzor nad kartico smo uporabljali v okolje integrirano SD knjižnico, ki omogoča delo s FAT16 datotečnim sistemom na Flash pomnilniški kartici.

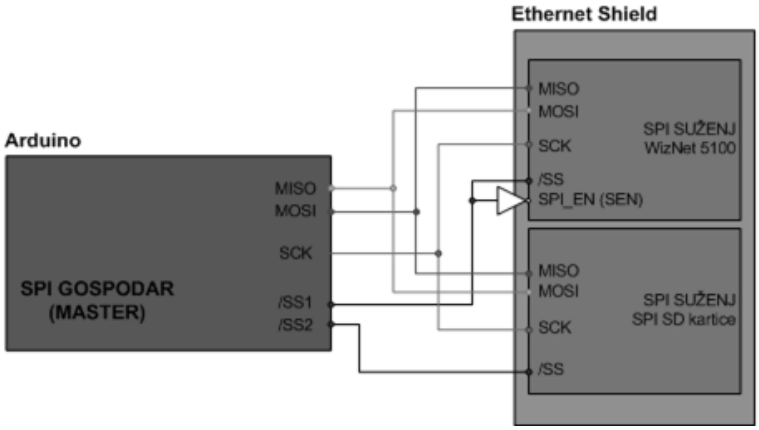
Hkratna komunikacija preko SPI vodila je nevarna (prepovedana), potrebno je poskrbeti, da na vodilu ob vsakem trenutku posluša samo en SPI suženj. Za to je bilo poskrbljeno znotraj programa.

Če želimo komunicirati s SD kartico, je potrebna aktivacija (aktivno nizka) /SS vhoda na vmesniku pomnilniške kartice. To storimo z izhodom SS2 na Arduinu (pin 4).

WizNet5100 potrebuje za vzpostavitev SPI komunikacije ustrezno nastavitev dveh vhodov:

- /SlaveSelect je potrebno postaviti na nizko stanje,
- SPI_Enable je potrebno postaviti na visoko stanje.

Da so se razvijalci Ethernet rezine izognili uporabi dveh izhodov za upravljanje WizNet5100 krmilnika, so na strani Arduina uporabili le eno linijo (pin 10) za naslavljanje /SS vhoda. Invertiran signal linije so povezali na vhod SPI_Enable Wiznet5100 krmilnika (Slika 9).



Slika 9: Linije SPI komunikacija z Ethernet čipom in SD kartico

4.3. Povezava čitalnika kartic na Arduino razvojno okolje

Ob prvih meritvah, ki smo jih z merilnim inštrumentom opravili na čitalniku kartic, smo bili nekoliko skeptični, saj smo izmerili, da izhoda W0 in W1 ne ustrezata definiciji po Wiegand26 standardu (2.1.3). Njuna vrednost je nihala v območju 2-4V, iz česar smo sklepali, da je čitalnik okvarjen in poizkusili z zamenjavo. Z nadomestnim čitalnikom obnašanje izhodov podobno.

Izhoda W0 in W1 smo sklenili preko pull-up uporov vrednosti 2.2kΩ povezati na 5V (mirovno stanje Wiegand izhodov je po standardu definirano visoko).

Na mikrokrmilnik smo naložili preprost program, ki je ob zunanji prekinitvi preko serijskega vmesnika v Arduino okolje poslal številčno oznako:

- ob »falling-edge« prekinitvi na prekinitvenem vhodu 0 (Arduino-pin2); oznako »0«,
- ob »falling-edge« prekinitvi na prekinitvenem vhodu 1 (Arduino-pin3); oznako »1«.

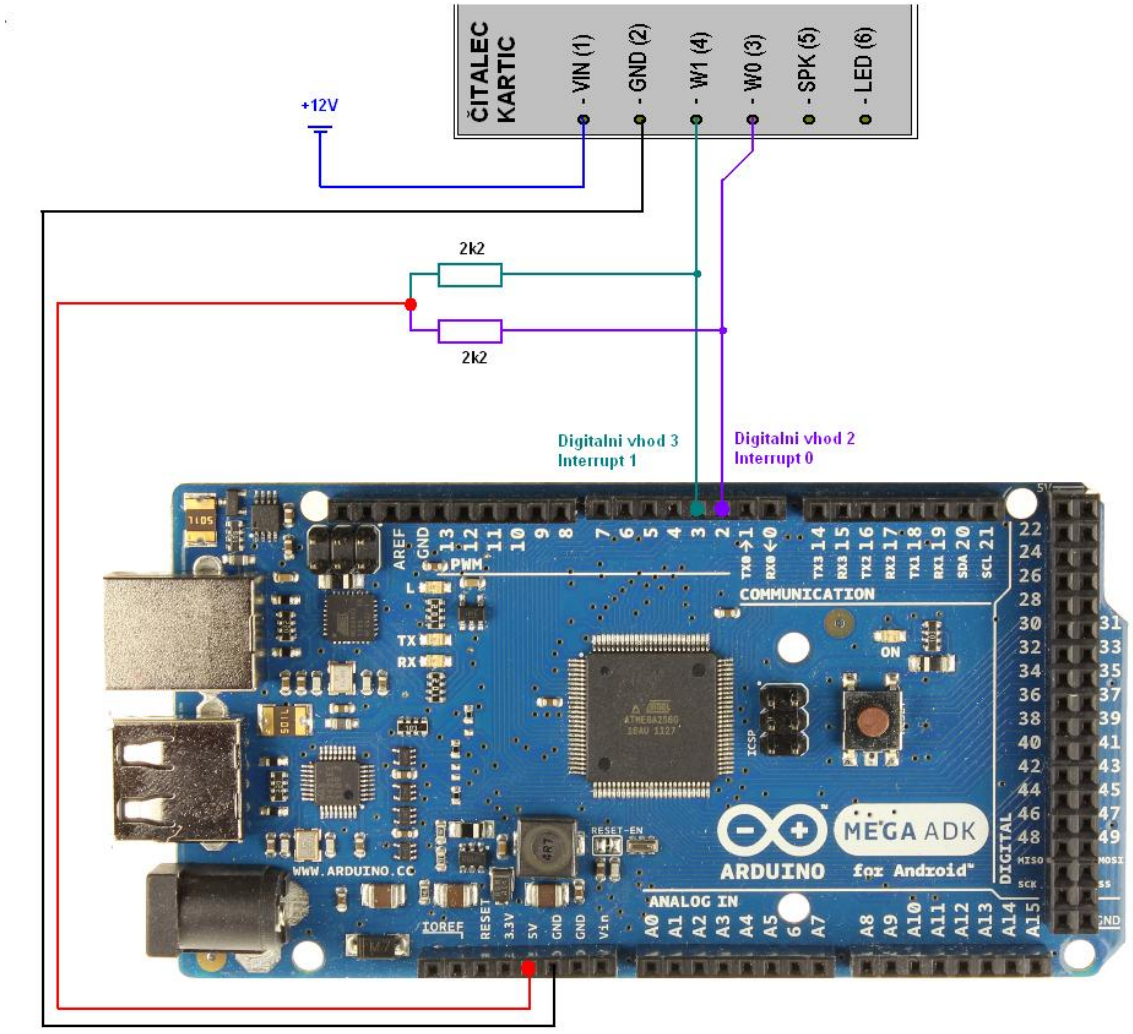
Delovanje programa smo preizkusili na prototipni ploščici: oba prekinitvena vhoda smo preko pull-up upora povezali na 5V, nato pa smo s preklapljanjem stikala, ki je vhod povezovalo na GND, simulirali prekinitve čitalnika. Program se je odzival po pričakovanjih (Preglednica 3).

Vhod mikrokmilnika	Zunanja prekinitev	Sprememba stanja na vhodu	Spročilo prejeto preko serijskega vmesnika	Želen rezultat
2	0	Nizko->Visoko	/	DA
2	0	Visoko-Nizko	0	DA
3	1	Nizko->Visoko	/	DA
3	1	Visoko->Nizko	1	DA

Preglednica 3: Predstavitev komunikacije čitalnika kartic z Arduinom (simulacija)

Ob poizkusu odčitavanja kartice po vezavi, predstavljeni na spodnji shemi (Slika 10), smo bili presenečeni, saj smo za vsako kartico prejemali enolično kodo, rezultat odčitavanja je bil ponovljiv, vendar so se kode razlikovale v dolžini. Vzrok napake je bil v programu, saj je pošiljanje sporočila preko serijskega vmesnika trajalo predolgo, ker ga je naslednja prekinitev prekinila. Napako smo odpravili s tem, da smo v prekinitveni rutini shranjevali podatke v spremenljivko, ki smo jo izpisali šele po koncu prejemanja podatkov.

Izhod na čitalniku W0 smo povezali na prvi prekinitveni vhod Arduino mikrokrmilnika (pin 2). Izhod na čitalniku W1 smo povezali na drugi prekinitveni vhod (pin 3). Napajanje čitalnika smo zagotovili z zunanjim 12V usmernikom.



Slika 10: Povezava čitalnika z razvojno ploščico Arduino Mega 2560 [18]

5. Izdelava programa

5.1. Programsko okolje Arduino

Programsko okolje Arduino za programiranje mikrokrmilnikov, ki smo ga uporabljali pri izdelavi izdelka diplomske naloge, je izpeljanka prototipnega okolja Wiring. Zasnovano je kot odprtokodna rešitev, podprto je na večini večjih operacijskih sistemov in je definirano s tremi komponentami:

- s programskim jezikom Processing,
- z integriranim Wiring razvojnim okoljem (Integrated Development Enviroment) in
- z dodelano dokumentacijo za tiskanino, definirano po principu Single Board Microcontroler.

Obstaja več različnih izpeljank okolja Wiring, ki podpirajo različne družine in proizvajalce mikrokrmilnikov, izbrano okolje Arduino pa omogoča uporabo mikrokrmilnikov proizvajalca Atmel.

Kodi, ki jo ustvarimo v Arduino IDE okolju se imenuje skica (sketch). Prednost IDE okolja je, da si pri pisanju skic lahko pomagamo z ogromno zbirko knjižnic. Obširna zbirka knjižnic s primeri in dokumentacijo je integrirana že v okolje. Knjižnice, hranjene v obliki .cpp in .h datotek, se prevedejo ob vsakem prevajanju skice. Morebitna opozorila zaradi napake v knjižnici, ki smo jo popravljali, se nam izpišejo znotraj IDE okolja, tako da zunanjega prevajalnika za popravljanje/izdelavo knjižnic ne potrebujemo. Novo knjižnico integriramo v okolje preprosto s kopiranjem datotek z izvorno kodo v samostojno podmapo v mapi s knjižnicami.

Za uspešno prevajanje skice je potrebno znotraj skice definirati dve funkciji:

- setup() – funkcija za inicializacijo okolja in spremenljivk, ki se izvede samo enkrat, takoj ob začetku izvajanja programa in
- loop() – glavna zanka programa.

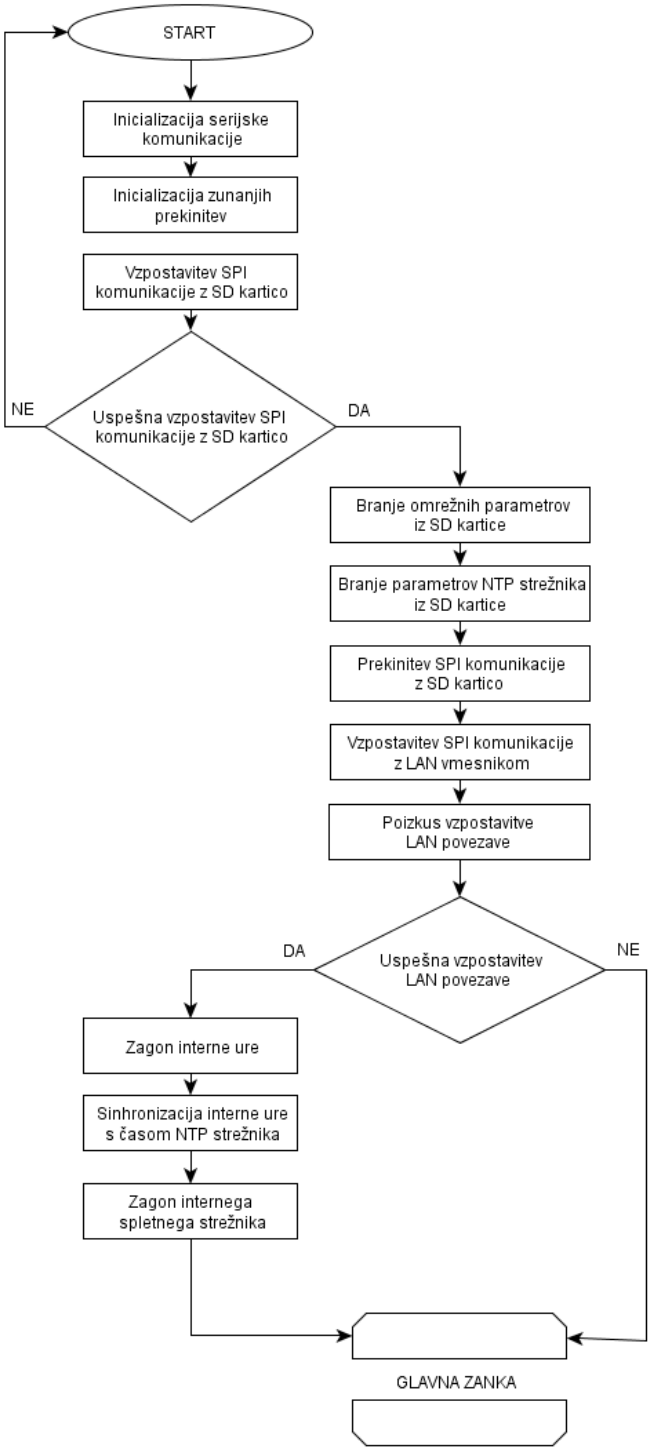
5.2. Programiranje v okolju Arduino

Okolje Arduino nas je pritegnilo s svojo preprostostjo. Obstoječi primeri so nas hitro vpeljali v delo z integriranimi knjižnicami in začutili smo občutek preprostosti. Vendar smo kot novinci v okolju mikrokrmilnikov kmalu naivno spoznali, da je včasih tudi za malenkostno spremembo v delovanju obstoječih knjižnic potrebna korenita sprememba v knjižnico. Tako se tudi s programiranjem v okolju Arduino ne izognemo pregledovanju tehnične dokumentacije.

Vendar je večina knjižnic integrirana napisana pregledno, tako da smo spremembe v knjižnicah izvedli brez večjih težav. Večinoma smo v knjižnicah izvajali le manjše popravke ter odstranjevali neuporabljene elemente, da smo nekoliko prihranili na sistemskih virih. Celotna izvorna koda je tako izrabila približno 32KB/256KB Flash pomnilnika.

Osnovna skica v okolju Arduino za uspešno prevajanje zahteva definicijo dveh funkcij setup() in loop(). Funkcija setup() je namenjena izvedbi inicializacije in se od vseh funkcij, ki jih definira uporabnik, izvede prva. Po izvedbi funkcije setup() se neprekinjeno izvaja funkcija loop() – glavna funkcija programa. Podrobna predstavitev aktivnosti znotraj obeh funkcij z grafično predstavitvijo (Slika 11) sledi v naslednjem podpoglavju.

5.3. Inicializacija - funkcija setup()



Slika 11: Groba predstavitev inicializacije programa

5.3.1. Inicializacija serijske komunikacije z okoljem Arduino

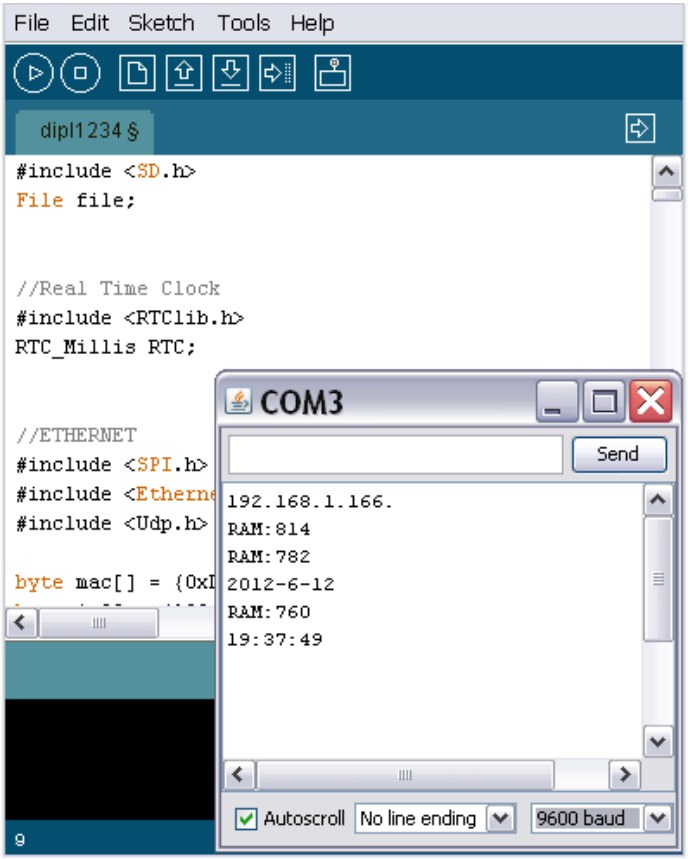
Knjižnica Serial je osnova za komunikacijo Arduina z računalnikom ali ostalimi zunanjimi napravami. Na vse Arduino ploščice je integriran vsaj en univerzalni serijski port – USART. Komunikacija poteka z IDE okoljem preko vhodov/izhodov 0(RX) in 1(TX) digitalnega sklopa Arduina. Ob uporabi serijske komunikacije torej odpade uporaba vhodov/izhodov 0 in 1 za ostale namene. Razvojni ploščici Arduino ADK za komunikacijo z ostalimi napravami preostanejo še trije UARTI.

Za pretvorbo med serijsko in USB komunikacijo skrbi program, ki se izvaja na drugem mikrokrmilniku Atmega8U2. Za potrebe komuniciranja IDE programskega okolja z osrednjim mikrokrmilnikom Arduino ploščice se ob namestitvi okolja ustvari virtualni USB vmesnik, ki omogoči komunikacijo s programskim okoljem Arduino.

Knjižnico Serial smo uporabljali za pomoč pri razhroščevanju in sledenju toka programa. Ker smo uporabljali nizko hitrost komunikacije 9600 baudov/s, smo morali biti previdni, saj bi lahko ob daljših komunikacijah zamudili časovno kritične dogodke.

Knjižnica je privzeto nastavljena za komunikacijo preko prvega UART-a in se s privzetimi nastavitvami znotraj skice aktivira z enim samim klicem funkcije.

Slika 12 prikazuje okno za nadzor komunikacije IDE okolja z Arduinom preko USB vmesnika.



Slika 12: Arduino IDE - Serijska komunikacija z razvojno ploščico

5.3.2. Inicializacija zunanjih prekinitev mikrokrmilnika

Nekatere izmed nalog (funkcij) v skici se izvajajo dlje časa, zato bi s preverjanjem stanja zunanjih znotraj glavne zanke programa občasno zamudili delno ali celotno prejemanje kode kartice od čitalnika.

Zato smo si pri branju kode kartice pomagali z zunanjimi (strojnimi) prekinitvami mikrokrmilnika. Ob prekinitvi procesor shrani vse potrebne podatke, ki jih potrebuje za nadaljnjo izvajanje in takoj prične z izvajanjem prekinitvene rutine. Po končani obdelavi prekinitvene rutine nadaljuje s predhodno nalogo.

Glavni mikrokrmilnik ATMega2560 je opremljen s šestimi vhodi, ki lahko prožijo servisne procedure ob zaznanih zunanjih prekinitvah. Knjižnica za upravljanje poskrbi za preprosto delo s prekinitvami brez prebiranja tehnične dokumentacije mikrokrmilnika.

5.3.3. SPI knjižnica

SPI knjižnica, integrirana v okolje, definira vse različne konstante in funkcije, ki se uporabljajo za:

- nastavitev hitrosti prenosa oz. frekvence urinega delilnika (metoda SPI.setClockDivider),
- nastavitev vrstnega reda podatkov (metoda SPI.setBitOrder),
- nastavitev podatkovnega načina glede na fazo in polariteto ure (metoda SPI.setClockDivider),
- inicializacijo prenosa (metoda SPI.begin),
- prenos podatkov bajta podatkov (SPI.transfer),
- zaključek prenosa podatkov (SPI.end).

Zaradi omogočanja fizične kompatibilnosti z različnimi razvojnimi ploščicami fizični vhodi/izhodi niso definirani znotraj knjižnice, temveč znotraj datoteke v okolju ''pins_arduino.h''. Zato ob zamenjavi razvojne ploščice ni bilo potrebno opraviti nobenih sprememb.

```
#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
const static uint8_t SS   = 53;
const static uint8_t MOSI = 51;
const static uint8_t MISO = 50;
const static uint8_t SCK  = 52;
#else
const static uint8_t SS   = 10;
const static uint8_t MOSI = 11;
const static uint8_t MISO = 12;
const static uint8_t SCK  = 13;
#endif
```

5.3.4. SD kartica

Integracija SD knjižnice, ki bazira na popularni odprtokodni knjižnici SdFatLib, omogoča uporabo nekaterih njenih segmentov v uporabniku zelo preprosti obliki. Knjižnica vsebuje globalni SD objekt, preko katerega lahko iz skice preprosto:

- izvedemo inicializacijo SPI komunikacije z SD kartico (SD.begin()),
- preverimo obstoj mape ali datoteke,
- odpremo datoteko za branje ali pisanje (dodajanje, prepisovanje),
- ustvarimo/brišemo mape ali datoteke,
- se sprehajamo po vsebini in iščemo vsebino znotraj datotek.

Prvi korak k vzpostavitvi SPI komunikacije z SD Flash pomnilniško kartico je aktivacija digitalnega izhoda 4 razvojne ploščice, kar aktivira aktivno nizki vhod /SS SD kartice (kateri se uporablja z okoljem Uno in Mega ADK).

Nato z uporabo knjižnice preverimo fizično prisotnost kartice, ustreznost datotečnega sistema, prisotnost potrebnih konfiguracijskih datotek ter fizično prisotnost seznama, na podlagi katerega avtoriziramo posamezne RFID kartice.

Na SD kartici smo kreirali 8 datotek, ki se uporabljajo tekom izvajanja programa. Ker smo bil pri razvoju z Arduino UNO razvojno ploščico omejeni z 2KB statičnega rama, smo imena datotek zapisali v okrajšanih različicah (Preglednica 4). Ob prehodu iz ploščice UNO na ploščico Mega ADK, kljub neizkoriščenemu pomnilniku, preimenovanja nazaj na celotna imena datotek nismo izvedli.

Ime datoteke	Uporaba
1.b	Datoteka, ki vsebuje html kodo spletne strani, ki se predstavi kot index v primeru, da imamo v konfiguracijski datoteki ''l.b'' izbran omejen prikaz vsebin preko HTTP strežnika (dostop do logov in ogled trenutno veljavnih kartic).
2.b	Datoteka, ki je nadgradnja datoteke ''1.b'' in se jo preko HTTP strežnika pošlje odjemalcu v primeru, da imamo v konfiguracijski datoteki dodan parameter ''WE=1'', kar poleg dostopa do osnovnih vsebin omogoča še dodajanje ali brisanje kartic na seznam dostopnih kartic ''d.b''.
d.b	Datoteka, v kateri so hranjeni vrstični zapisi s kodo kartice, opisom in pripadajočim statusom (aktivna, neaktivna). Primer aktivnega vrstičnega zapisa: a11101110011111101000011000;BostjanLuzar Primer neaktivnega vrstičnega zapisa: a11101110011111101000011000;BostjanLuzar- rezervna kartica
h.b	V datoteki se hrani privzeti HTTP odgovor, ki predstavlja statusno kodo 200. Branje in pošiljanje te vsebine se izvede na začetku vsakega odgovora HTTP zahtevi.
i.b	Datoteka namenjena hranjenju IP naslova.
l.b	V datoteki so zapisani podatki z varnostnimi parametri v formatu, ki je podobnem xml zapisu. Hranjenje gesla, IP in MAC naslova računalnika, iz katerih je omogočen dostop preko HTTP vmesnika. Datoteka prav tako določa izbiro med okrnjenim in polnim spletnim vmesnikom (polni spletni vmesnik omogoča tudi manipulacijo s pristopnimi pravicami).
l.txt	Dnevniška datoteka, v katero se kronološko vpisujejo registracije in poizkusi registracij na čitalniku.
n.b	V datoteki se hrani naslov NTP strežnika, od katerega ob inicializaciji programa pridobimo trenutni datum in čas.

Preglednica 4: Predstavitev osnovnih podatkovnih datotek, hranjenih na izhodišču SD kartice

5.3.5. Konfiguracija osnovnih omrežnih parametrov

V Flash pomnilnik mikrokrmilnika smo v primeru odsotnosti SD kartice definirali privzete omrežne parametre:

- definicija privzetega MAC (Media Access Control) naslova

  byte mac[] = { 0xDE, 0xAD, 0xBE, 0xAB, 0xCE, 0xDA }; ,

- definicija privzetega IPv4 naslova

  byte ip[] = { 192,168,1, 177 };,

- definicija IP naslova privzetega NTP strežnika (komunikacija poteka preko standardnega NTP porta 123).

  byte timeServer[] = { 89,143,246,30 }; //ntp1.siol.net.

Do spletnega vmesnika (HTTP strežnika) lahko dostopamo le preko nestandardnega porta 40. Nastavitev je definirana v spremenljivki, v izvorni kodi skice. Spremembe nastavitve preko HTTP vmesnika ali preko konfiguracijske datoteke na SD kartici nismo predvideli.

V proceduri setup() ob zagonu programa ob uspešni inicializaciji komunikacije z SD kartico preverimo obstoj konfiguracijske datoteke ''ip.b''. Ob uspešnem odpiranju tekstovne datoteke ''ip.b'' iz datoteke preberemo ustrezne pravice, kjer so zapisani poljubni omrežni parametri za katere želimo, da jih prevzame sklad Arduinove Ethernet rezine.

Branje IP in NTP naslova smo realizirali na rezini UNO z 1KB statičnega (SRAM) pomnilnika, zato sta podatka zapisana vsak v svoji datoteki, v kateri je predviden zapis dejanskega parametra v obliki podatkov ločenih z vejico, ker je pomenilo, da smo lahko podatek prebrali z razmeroma nizko izrabo pomnilnika.

Parametri, ki služijo pri ugotavljanju identitete in avtorizacije dostopov ter akcij html odjemalcev, se hranijo v datoteki ''l.b''. Branje parametrov smo izvedli že na platformi Mega ADK z 8KB statičnega (SRAM) pomnilnika, kjer se nam s porabo pomnilnika ni bilo potrebno ukvarjati. Tako so podatki hranjeni v zapisu, ki je zelo podoben .xml datoteki, in se pri branju iz datoteke uporabljajo nekoliko bolj procesno obremenjujoče funkcije za delo z nizi.

```
<g>geslo</g>
<i>192.168.1.2</i>
<m>001617DB38FF</m>
<w>WE=1</w>
```

Niz pod ključem <g></g> predstavlja geslo, katerega mora poznati uporabnik spletnega mesta za upravljanje z dostopnimi pravicami, če je omogočen način upravljanja s pravicami preko spletnega vmesnika.

Polje pod ključem predstavlja IP naslov oddaljenega računalnika, ki je avtoriziran, da dostopa do spletnega vmesnika. Če HTTP zahteva ne pride s pravega IP naslova, je odjemalec obveščen, da njegov dostop ni avtoriziran.

Polje pod ključem <m></m> predstavlja MAC naslov oddaljenega računalnika, ki je avtoriziran, da dostopa do spletnega vmesnika. Če HTTP zahteva ne izhaja iz ustreznega MAC naslova, je odjemalec obveščen, da njegov dostop ni avtoriziran.

5.3.6. Ethernet

Izhodišče realizacije razreda Ethernet kot celote bazira na naslednjih izvornih datotekah:

- w5100.h,
- w5100.cpp,
- socket.h,
- socket.cpp.

V prvih dveh datotekah so definirane funkcije, ki so potrebne na najnižjem nivoju za inicializacijo, upravljanje, ter prenašanje podatkov preko vtičnikov (socket) WizNet5100 vmesnika. V drugih dveh izvornih datotekah so realizirane osnovne funkcije za delo s TCP ali UDP vtičniki (odpiranje, zapiranje, branje, pisanje, itd.). Strojna oprema omejuje število istočasno aktivnih vtičnikov na 4 izvode.

5.3.7. Sinhronizacija časa preko NTP protokola

RTClib.h

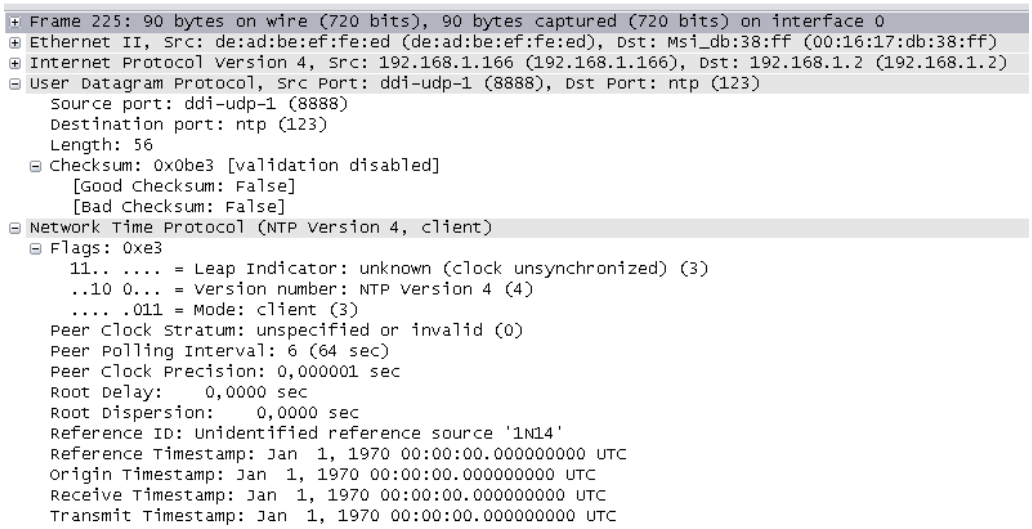
Knjižnica RTClib.h [11] ni vključena v Arduino IDE okolje. Knjižnica in navodila za integracijo v Arduino IDE okolje so javno na voljo na spletnem naslovu HTTPs://github.com/adafruit/RTClib/ za uporabo brez omejitev. V knjižnici je definiran razred DateTime, katerega smo racionalizirali za uporabo na Arduino UNO razvojni ploščici. Po prehodu na tiskanino Mega ADK smo v skici definirali dve metodi, ki v obliki tekstovnega niza vrneta trenuten čas in datum, katerega hrani razred DateTime.

Udp.h

Realizacija UDP knjižnice temelji na vseh predhodno opisanih knjižnicah, kar pripomore h kratki in enostavno berljivi izvorni kodi. UdpClass razred je unikatno definiran s številko vtičnika (0-4) ter portom preko katerega se izvaja branje ali pisanje podatkov.

Prvi korak pridobivanja trenutnega datuma in časa je pošiljanje UDP paketa, dolgega 48 bajtov, na naslov želenega NTP strežnika preko standardnega porta 123. Poslani paket je izvod razreda UdpClass. Vsebina paketa je statično definirana.

Po poslanem paketu program čaka sekundo in nato preveri, če je bil prejet odgovor. Ob uspešnem prejetju NTP odgovora program iz paketa izlušči prejeti čas (bajti 40-43 v paketu). Če je vrednost prejetega datuma večja od 23.1.2012 (čas prve uspešne NTP sinhronizacije, ki smo jo uspeli izvesti preko Arduino UNO platforme), program to sprejme kot veljavno vrednost in ustvari izvod razreda DateTime (RTClib.h) s prejeto vrednostjo in upoštevanjem popravka lokalnega časovnega pasu. Slika 13 prikazuje zajeti paket, ki je bil testno poslan na naslov enega izmed lokalnih računalnikov za analizo.



Slika 13: Primer poslane NTP zahteve

5.3.8. HTTP strežnik

Izhodišče realizacije Ethernet knjižnice kot celote bazira na knjižnicah w5100.h in socket.h. V prvi knjižnici so definirane funkcije, ki so potrebne na najnižjem nivoju za inicializacijo, upravljanje, ter prenašanje podatkov preko WizNet5100 vmesnika. V drugi knjižnici so realizirane osnovne funkcije za delo s TCP ali UDP vtičniki (odpiranje, zapiranje, branje, pisanje,…). Strojna oprema omejuje število istočasno aktivnih vtičnikov na 4. Obe knjižnici služita kot temelj v naslednjih razredih, uporabljenih v izvorni kodi izdelka diplomske naloge:

Ethernet.h

V razredu je več definicij funkcije Ethernet.begin(), izvedba ustrezne izmed definicij (glede na podane omrežne parametre) s klici drugih funkcij v celoti poskrbi za vzpostavitev omrežne povezave. Izvede se inicializacija W5100 krmilnika, nastavita se MAC in IP naslov, določi se privzeti prehod ter maska podomrežja.

Client.h

Razred Client spremlja dogajanje na izbranem TCP vtičniku. V razredu je definiranih veliko javnih funkcij, s katerimi nadzorujemo in spremljamo stanje, beremo in pišemo podatke preko izbranega vtičnika.

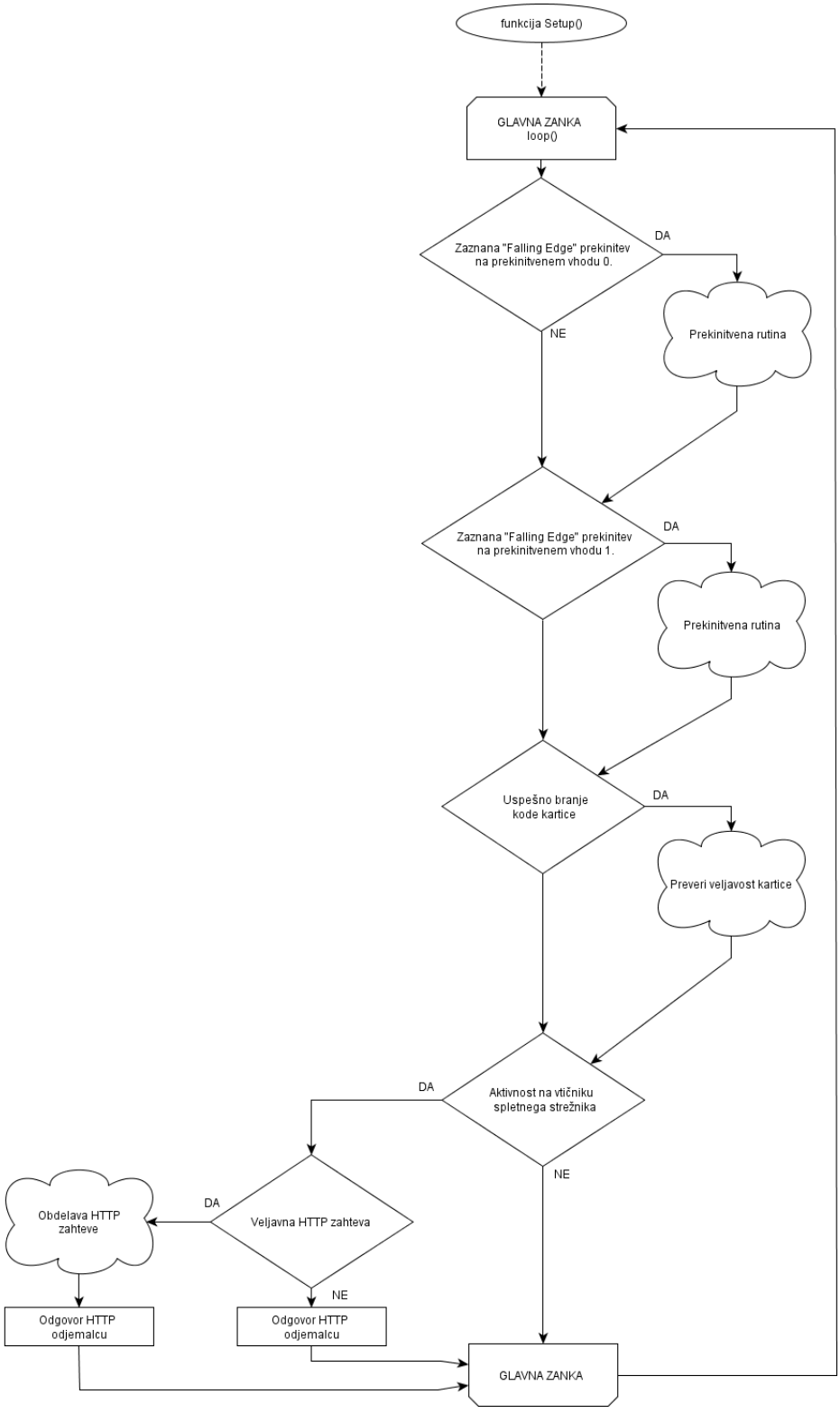
V razredu smo definirali dve dodatni kratki javni funkciji, ki vrneta IP in MAC naslov odjemalca, kar smo uporabili za primerjavo z vnosom avtoriziranega odjemalca v datoteki na SD kartici ''l.b''.

Server.h

Razred, ki je namenjen spremljanju aktivnosti na izbranem TCP vtičniku. V izvorni kodi izdelka smo definirali nestandardni port 40. V razredu je definiranih veliko javnih funkcij, s katerimi beremo in pišemo podatke. V diplomski nalogi se javne metode razredov Server.h in Client.h dopolnjujejo. Implementacija strežnika je zasnovana na primeru, ki je že integriran v okolje Arduino.

5.4. Glavna zanka programa – funkcija loop()

V glavni zanki programa se zaporedno izvajajo vse aktivnosti (Slika 14), ki zagotavljajo funkcionalnost izdelka. Naloge, ki se izvajajo v ozadju, skrbijo za ustrezno delovanje spletnega strežnika, prednostno nalogo pa predstavljajo aktivnosti, ki skrbijo za ustrezno razpoznavanje kartic in ustrezno avtorizacijo dostopov.



Slika 14: Groba predstavitev izvajanja glavne zanke programa

5.4.1. Branje kode kartice

Za čitalnik kartic je potrebno zagotoviti 12V enosmerno napajanje, kar pomeni dodatni zunanji usmernik. Na srečo ima Arduino na tiskanini integriran pozitivni regulator, ki skrbi za znižanje in glajenje enosmerne napetosti zunanjega napajanja (5V), ki se uporablja za napajanje razvojne ploščice in se tako lahko za napajanje uporabi le en zunanji usmernik.

Čitalnik ima poleg D0 in D1 izhoda še 2 uporabna vhoda: vhod za aktiviranje LED diode in vhod za aktiviranje piskača (oba vhoda sta definirana kot aktivno nizka). Odločili smo se, da ju ne uporabimo.

Po Wiegand standardu imata izhoda D0 in D1 definirano visoko mirovno stanje, vendar je bilo stanje v našem primeru nedefinirano. Potencial posameznega izhoda smo preko pull-up upora v vrednosti 2.2kΩ izenačili na napajalno vrednost Vdd (5V) in izhoda povezali na vhoda Arduina, označena z Interrupt0 (digitalni vhod 2) in Interrupt1 (digitalni vhod 3). Gre za vhoda, ki podpirata zunanje prekinitve.

Prekinitev je tip dogodka (asinhroni signal, v tem primeru nizek pulz na liniji D0 ali na D1), ki sporoča, da se je zgodil dogodek, ki se obravnava prednostno. Stanje procesorja se shrani in izvede se prekinitvena rutina. Po končanem izvajanju prekinitveno rutine se delovanje programa vrne nazaj v izhodiščno stanje. Arduino Mega ADK (ATmega2560) je opremljen s šestimi zunanjimi prekinitvami. Realizirali smo le možnost priklopa za enega čitalnika, vendar bi lahko zaradi štirih prostih prekinitvenih vhodov hitro dodali še podporo za 2 dodatna čitalnika. Nekoliko zamudnejša bi bila sprememba spletnega vmesnika in načina hranjenja pravic za vsak čitalnik. Razvojna ploščica UNO žal podpira samo 2 zunanji prekinitvi (1 čitalnik).

Okolje vsebuje integrirano knjižnico, s katero smo si pomagali pri delu s prekinitvami. Knjižnica je uporabniku prijazna in poskrbi za vse nastavitve registrov (omogočanje prekinitev,frekvenca vzorčenja,…) za uporabo prekinitev. Določiti moramo le tip prekinitve na posameznem vhodu (Interrupt0 ali Interrupt1). Za obe liniji smo določili prekinitev tipa FALLING (falling edge) (sprememba stanja iz 5V->0V), lahko bi uporabili tudi tip prekinitve RISING (rising edge, sprememba na liniji iz 0V->5V). Podprt je tudi tip prekinitve LOW, ta pa v našem primeru ni najbolj uporaben.

Pri branju kartice RFID po protokolu Wiegand26 znaša maksimalen čas med prekinitvama 2ms. V vsaki prekinitveni rutini smo trenutnemu binarnemu nizu dodali novo vrednost (novo prebrana 0 ali 1) in ponastavili časovnik, ki je meril čas od zadnje prekinitve.

Če čas zadnje prekinitve (po definiciji Wiegand protokola znaša maksimalno dovoljen čas 2ms) znaša več kot 50ms, se branje kartice zaključi; uspešno le pod pogojem, da je bilo prebranih točno 26 bitov. Po uspešnem branju smo v programu izklopili prekinitve, da niso motile izvajanja nadaljnje obdelave kode prebrane kartice, po koncu obdelave (preverjanja veljavnosti kartice) pa smo jih ponovno aktivirali.

Ostale naloge, ki se izvajajo v glavni zanki, so posledično postavljene v ozadje. So nižje prioritete, zato se lahko zgodi, da HTTP zahteva ne bo servisirana, ker bo program izvajal prednostno nalogo. Slika 15 predstavlja postopka, ki se izvedeta ob zaznani prekinitvi na katerem izmed prekinitvenih vhodov.

Slika 15: Groba predstavitev branja kode kartice

5.4.2. Obdelava prebrane kode kartice

Ko uspešno preberemo vseh 26 bitov identifikacijske kode kartice, znotraj prekinitvene rutine začasno izklopimo vse prekinitve, ker nočemo zaznavati nobenih novih prekinitev dokler nismo zaključili z vsemi akcijami, ki se tičejo trenutne kartice.

V prekinitveni rutini se nato program preusmeri v izvajanje funkcije, ki na SD kartici odpre tekstovno datoteko ''d.b'', v kateri so po vrsticah zapisane kode kartic in informacije o karticah, ki imajo pravico prehoda skozi čitalno mesto.

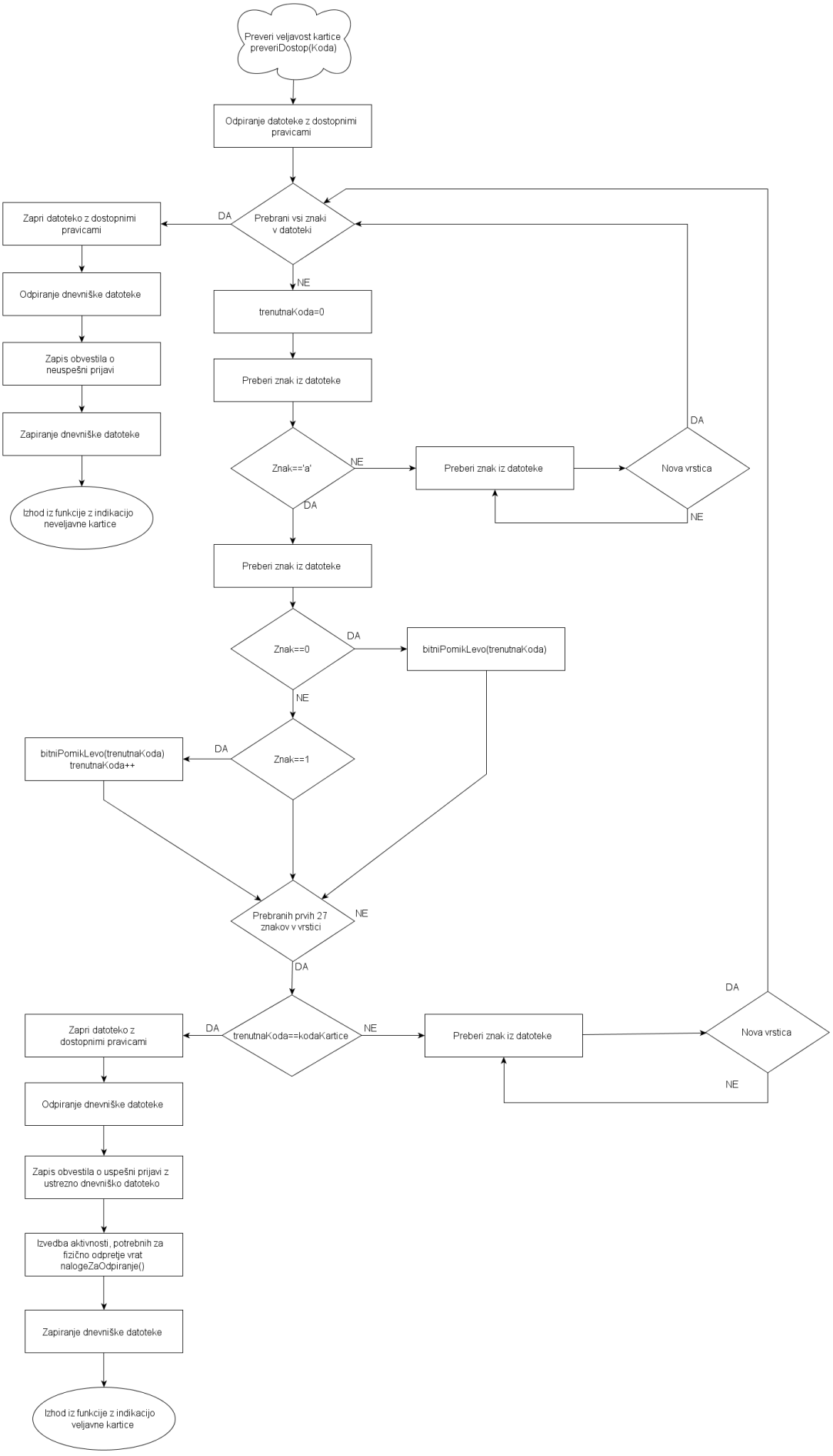
Primer zapisa dostopnih pravic je predstavljen v poglavju ''5.3.4 SD kartica''.

Branje datoteke poteka zaporedno, datoteka lahko vsebuje dvojnike, saj se ob prvem pravilnem ujemanju binarne kode nadaljnje branje ustavi.

Če datoteka ''d.b'' vsebuje kodo kartice, se v datoteko log1.csv pripne sporočilo o uspešni avtorizaciji in poljuben digitalni izhod (v našem primeru 13) postavi v visoko stanje – za poljubno časovno trajanje.

Če v datoteki ''d.b'' ne najdemo ustrezne vrstice z identično kodo kartice, potem v dnevniško datoteko ''l.txt'' zabeležimo poizkus odprtja vrat in kodo kartice, ki smo prejeli od čitalnika.

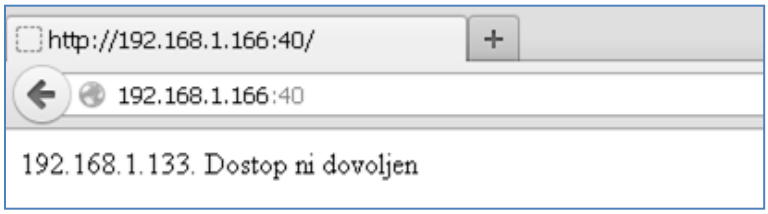
Pri pisanju v dnevniško datoteko pripišemo tudi trenuten sistemski čas. Slika 16 predstavlja aktivnosti, ki se izvedejo znotraj funkcije preveriDostop() po končanem branju kartice (prejetih 26 bitih kode kartice). Funkcija je namenjena ugotavljanju avtorizacije posamezne kartice in izvedbi ustreznih nadaljnjih akcij (beleženje aktivnosti, odpiranje vrat,…).



Slika 16: Ugotavljanje avtorizacije posamezne prebrane kartice

5.4.3. HTTP strežnik (web vmesnik)

V glavni zanki programa smo definirali neprekinjeno zaznavanje aktivnih HTTP odjemalcev. Ob prejemu novega sporočila s pomočjo funkcij, ki smo jih dodatno definirali v razredu Client, program preveri ali je paket prišel iz avtoriziranega odjemalca (iz pravilnega IP in MAC naslova, definiranega znotraj datoteke na SD kartici ''l.b''). Ob neveljavnih parametrih odjemalca strežnik poda sporočilo, da dostop ni avtoriziran ter povezavo z odjemalcem prekine (Slika 17).



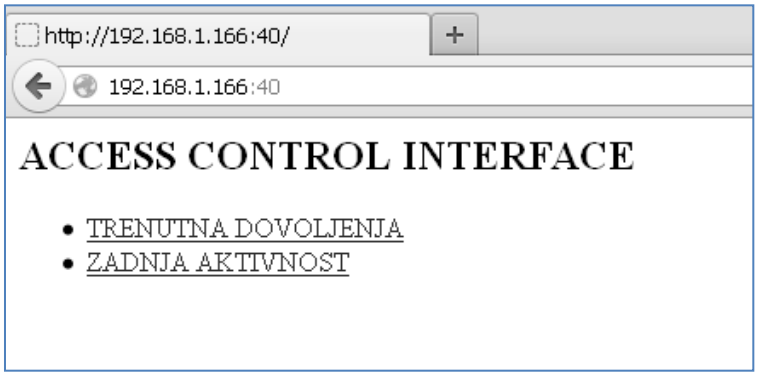
Slika 17: Sporočilo, ki ga prejme odjemalec, ki ni pooblaščen za delo na sistemu

Ob uspešni avtorizaciji prejeto zahtevo znak za znakom shranimo v niz. Glede na vrednost prejete zahteve, tipa zahteve (POST ali GET), želenega URL naslova, nastavitev v datoteki ''l.b'' in definicije načina delovanja spletnega odjemalca (polni ali omejeni dostop), strežnik odjemalcu ustrezno odgovori.

Zahtevi po indeksni strani ali zahtevi po neveljavni strani (koda 404) strežnik odgovori s html kodo, ki izpiše preprost meni s funkcijami (Slika 18).

Pod povezavo ''TRENUTNA DOVOLJENJA'' si lahko ogledamo seznam kartic, ki so bile predhodno že dodane v sistem. Seznam za vsako kartico vsebuje tudi opis in indikator veljavnosti. Ob kliku na povezavo se izpiše vsebina datoteke ''l.b''. Datoteka se hrani na SD kartici.

Pod povezavo ''ZADNJA AKTIVNOST'' si lahko ogledamo vsebino datoteke ''l.txt''. Datoteka se hrani na SD kartici in vsebuje podatke o zadnji aktivnosti (zgodovina prehodov) na čitalnem mestu.

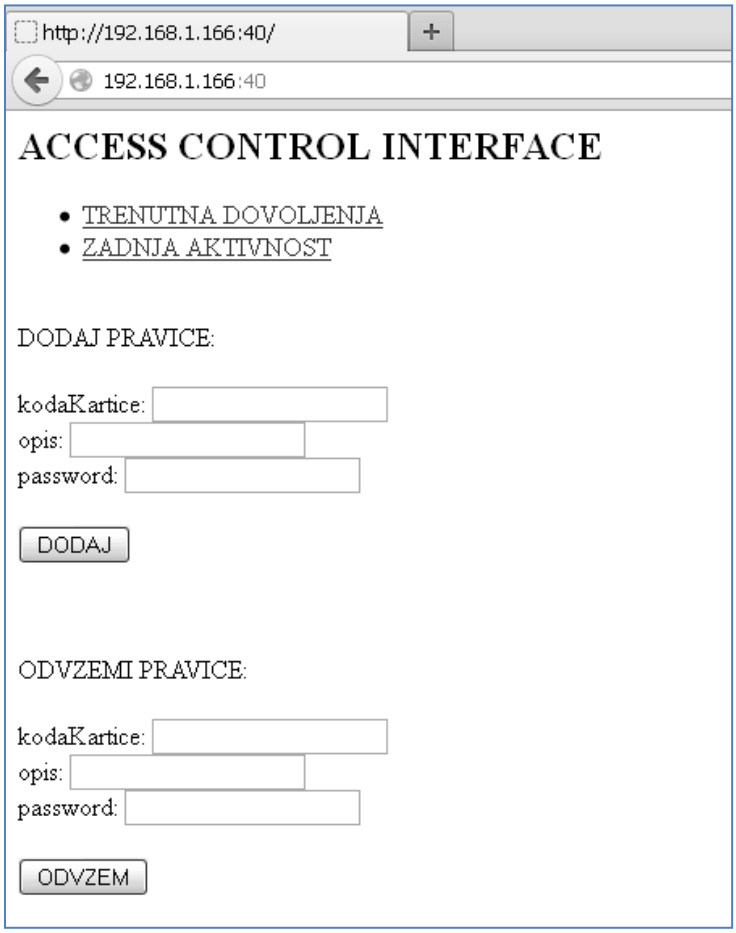


Slika 18: Možnosti omejenega načina delovanja HTTP vmesnika

Če je aktivirano dodajanje oz. odvzemanje dostopnih kartic, se v osnovnem meniju prikažeta še dva dodatna segmenta:

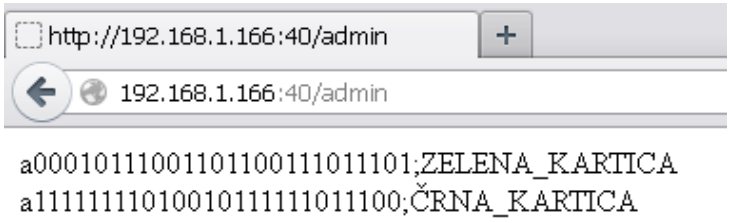
- ''DODAJ PRAVICE'',
- ''ODVZEMI PRAVICE''.

Za izvedbo akcij, iz polnega načina delovanja spletnega vmesnika, je potrebno zahtevo potrditi tudi z vnosom ustreznega gesla, ki je definirano v datoteki ''l.b'' (Slika 19).



Slika 19: Omogočeno dodajanje pravic karticam v polnem načinu delovanja

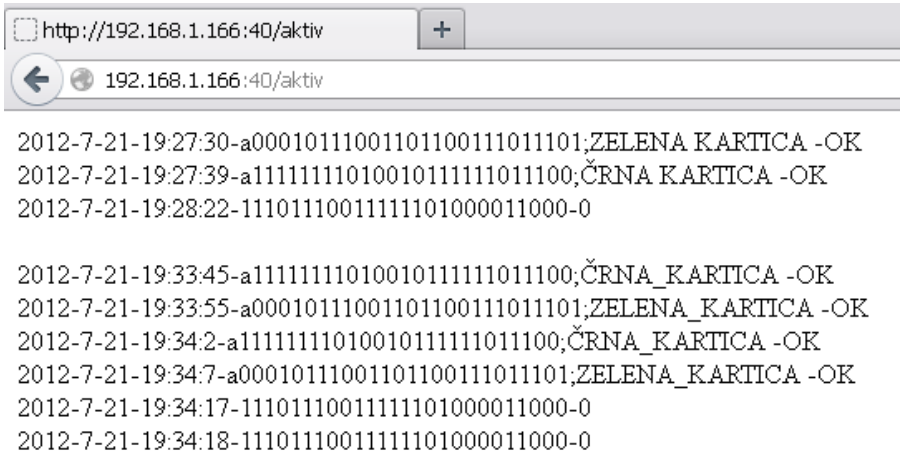
S klikom na povezavo ''TRENUTNA DOVOLJENJA'' se nam preko spletnega vmesnika naloži vsebina datoteke ''d.b''. V datoteki so vpisane vse kartice, katere so aktivne oz. so bile aktivne v preteklosti (Slika 20).



Slika 20: Ogled trenutnih dovoljenj kartic preko HTTP vmesnika

Razberemo lahko, da sta trenutno v sistemu aktivni samo dve kartici. Prav tako lahko iz vmesnika razberemo tudi posamezno kodo in oznako kartice.

S klikom na povezavo ''ZADNJA AKTIVNOST' se nam preko spletnega vmesnika naloži vsebina datoteke ''l.txt''. V datoteki so kronološko zapisane vse registracije ter neuspešni poizkusi registracij (Slika 21).

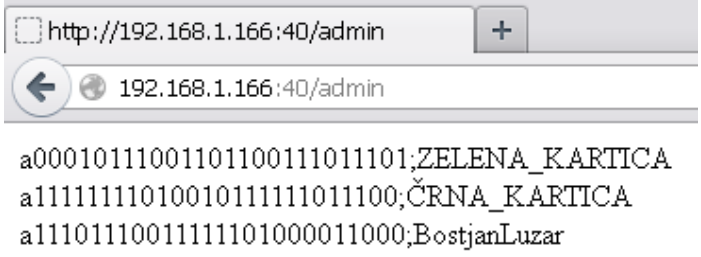


Slika 21: Izpis dnevniške datoteke preko HTTP vmesnika

V dnevniku opazimo uspešne registracije kartice, ki so v sistemu vodene pod oznakama ''ZELENA KARTICA'' IN ''ČRNA KARTICA'' z vrstično oznako ''OK''. Primeri neuspešne registracije se v dnevnik zapišejo z vrstično oznako ''0''. Pričakovano je kartica s kodo 11101110011111101000011000 za sistem nepoznana in ne omogoča dostopa.

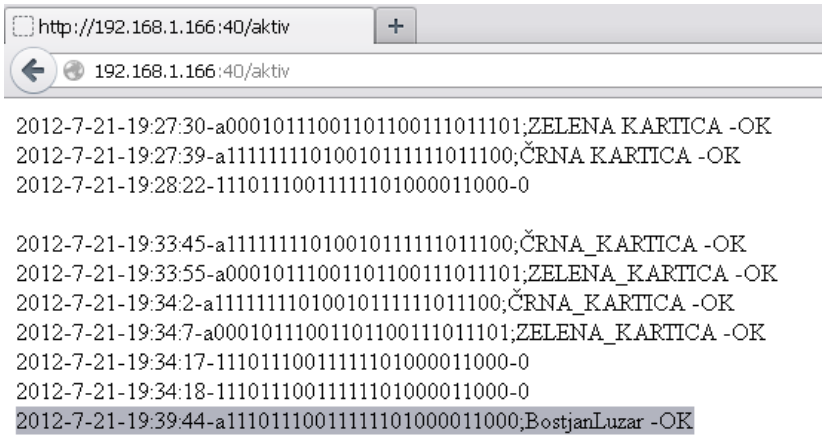
V polnem načinu delovanja HTTP vmesnika lahko preprosto dodamo dostopne pravice novi kartici (Slika 22) oz. ponovno aktiviramo obstoječo kartico, ki je bila predhodno dezaktivirana.

Slika 22: Dodajanje pravic kartici (s kodo 11101110011111101000011000) preko HTTP vmesnika



Slika 23: Nova aktivna kartica z dovoljenji za prehod

Slika 23 predstavlja seznam kartic, katere so avtorizirane na čitalniku. Lahko vidimo, da je bila kartica z napisom ''BostjanLuzar'' uspešno uvrščena na seznam.

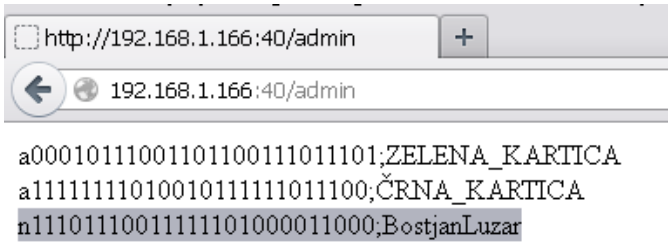


Slika 24: Registracija z novo vpisano kartico

Slika 24 prikazuje, da je sistem kartico s kodo 11101110011111101000011000 sedaj uspešno prepoznal in dostop odobril 21.7.2012 ob 19:39:44.

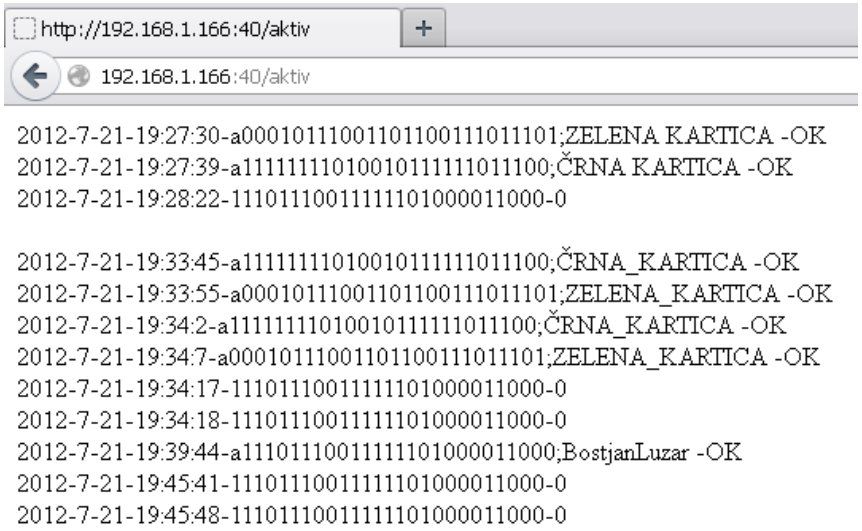
Preko spletnega vmesnika lahko kartici dostopne pravice odvzamemo. Zapis kartice se zaradi hranjenja zgodovine ne izbriše iz datoteke. Spremeni se samo prvi znak v vrstici, ki hrani zapis o specifični kartici iz vrednosti ''a'' (a-ktivna) v ''n'' (n-eaktivna). Slika 25 predstavlja primer odvzema pristopnih pravic kartici z oznako ''BostjanLuzar''.

Slika 25: Odvzemanje pravic izbrani kartici



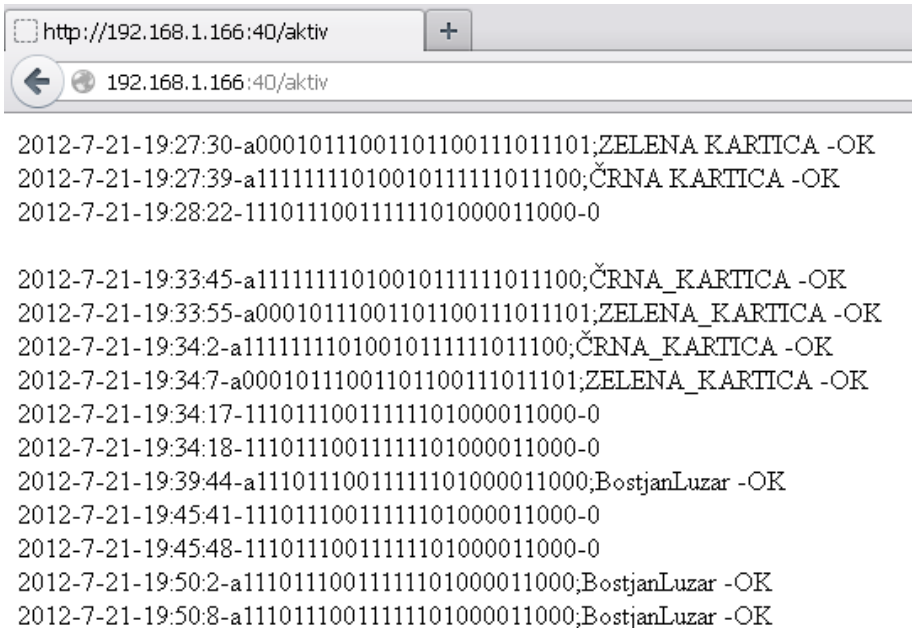
Slika 26: Onemogočen zapis kartice z oznako ''BostjanLuzar'' v datoteki ''d.b''

Slika 26 prikazuje stanje v datoteki ''d.b'' potem, ko smo preko spletnega vmesnika kartico onemogočili. Slika 27 prikazuje izpisek iz dnevniške datoteke, kjer lahko vidimo dva neuspešna poizkusa registracije z izbrano kartico. Vmesnik v tem primeru ne izpiše oznake kartice (ob neuspešni registraciji se zabeležita le čas in koda kartice).



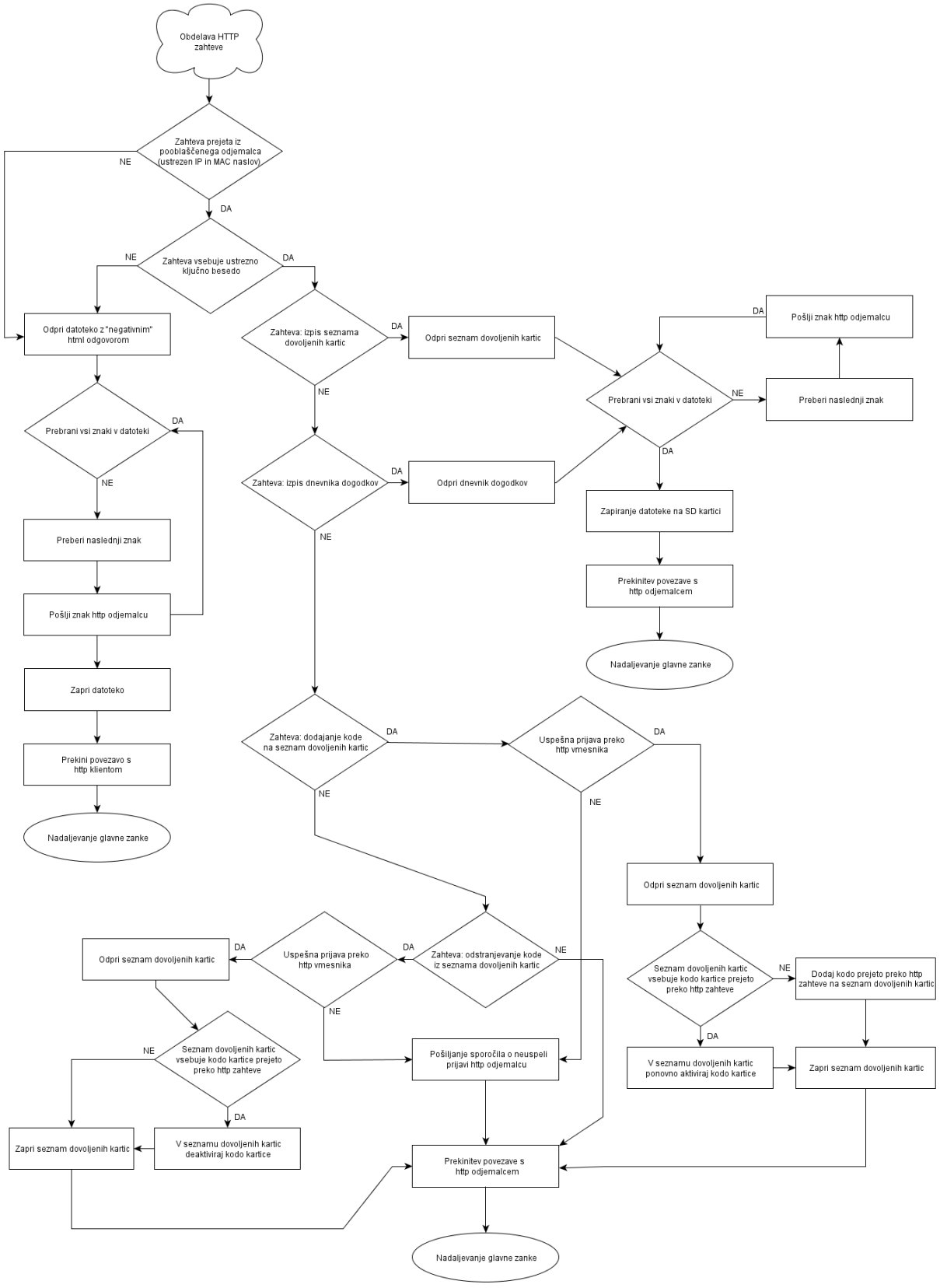
Slika 27: Dvojen neuspešen poskus registracije z dezaktivirano kartico

Slika 28 prikazuje dve uspešni registraciji s kartico, potem ko smo zapis kartice preko spletnega vmesnika ponovno aktivirali. Vmesnik je zasnovan tako, da ob ponovni aktivaciji oznake kartice zaradi hranjenja zgodovine ne moremo spremeniti.



Slika 28: Dvojna uspešna registracija s predhodno ponovno aktivirano kartico

Slika 29 prikazuje potek aktivnosti, ki se izvedejo za obdelavo posamezne http zahteve. Ker se naloga izvaja v ozadju jo lahko prekinitev na enem izmed prvih dveh prekinitvenih vhodih ustavi. To se odraža kot začasna nedosegljivost spletnega vmesnika.



Slika 29: Groba predstavitev obdelave prejete HTTP zahteve

6. Sklepne ugotovitve

Skozi razvoj izdelka smo se seznanili z novim konceptom programiranja mikrokrmilnikov v okolju Arduino. Okolje s svojimi knjižnicami in standardno razvojno platformo omogoča hiter, enostaven in transparenten razvoj funkcionalnosti, ki so s klasičnim razvojem zelo zamudne in v nekaterih primerih težko dosegljive.

Žal pri programiranju z uporabo knjižnic, katerih avtorji nismo sami, izgubimo pregled nad podrobnostmi, ki se izvajajo v ozadju. Posledično je včasih že manjša sprememba globoko v knjižnici zaradi nepoznavanja lahko zelo zamudna.

Med razvijanjem izdelka na razvojni ploščici Arduino UNO smo se neprekinjeno ukvarjali z optimizacijo sistemskih virov. Največjo težavo je zaradi tekstovne narave tematike predstavljalo pomanjkanje statičnega SRAM pomnilnika. Zato sem se med razvojem izdelka odločil za nakup svoje razvojne ploščice Arduino Mega ADK. Zaradi štirikrat večje kapacitete statičnega SRAM pomnilnika glavnega mikrokrmilnika za približno dvojno ceno različice UNO se nam je zdela različica Mega ADK primernejša izbira. Kompatibilnost izdelka z različico UNO žal nismo zagotovili.

Osnovni cilj razvoja smo dosegli, vendar smo mnenja, da bi morali izdelek do polne uporabnosti še dodelati, predvsem na področju varnosti. Največjo težavo predstavlja nešifrirana (plain-text) HTTP komunikacija, saj Atmelov mikrokrmilnik žal ni dovolj zmogljiv, da bi lahko komunikacijo izvajali preko protokola HTTPS.

Iskanje ustrezne rešitve bi lahko obsegalo novo diplomsko nalogo. Smo mnenja, da bi bil primeren začetek za raziskovanje področje začasnih gesel, generiranih na podlagi trenutnega časa (Time-Based One-Time Password).

Potrebno bi bilo tudi poskrbeti za šifriranje vsebine na SD kartici (vsaj datoteke z varnostnimi nastavitvami). Tudi za platformo Arduino obstajajo razne kriptografske knjižnice zato ocenjujemo, da implementacija nebi smela biti težavna (vsaj na platformi z mikrokrmilnikom Atmel Mega 2560).

Poleg izbire ustreznega ohišja in napajalnega vezja bi bila potrebna tudi realizacija modula, ki bi ustrezen TTL signal (izhod mikrokrmilnika Atmel Mega 2560) ojačal na ustrezen nivo, s katerim bi lahko direktno krmilili elektronski prejemnik (ključavnico) v okovju vrat.

Viri

[1] Access Control (2012). Dostopno na: HTTP://en.wikipedia.org/wiki/Access_control.

[2] RFID (2012). Dostopno na: http://en.wikipedia.org/wiki/Radio-frequency_identification

[3] H4102, Read Only Contactless Identification Device, EM Microelectronic-Marin (2000). Dostopno na HTTP://www.zygo.btinternet.co.uk/H4102_C.pdf.

[4] Wiegand Interface (2012). Dostopno na HTTP://en.wikipedia.org/wiki/Wiegand_interface.

[5] Magswipe, IB Technology (2005). Dostopno na HTTP://ibtechnology.co.uk/pdf/magswipe_dec.PDF.

[6] Arduino (2012). Dostopno na HTTP://arduino.cc/en/.

[7] Arduino Shield List (2012). Dostopno na HTTP://shieldlist.org/.

[8] Serial Pheripheral Interface Bus (2012). Dostopno na: HTTP://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus.

[9] W5100 datasheet, WIZnet (2006). Dostopno na: HTTP://hackable-devices.org/media/products_data/documents/documents/2010/05/19/W5100_Datasheet.pdf.

[10] Arduino Ethernet Shield (2012). Dostopno na: HTTP://arduino.cc/en/Main/ArduinoEthernetShield.

[11] RTClib, Adafruit Industries (2012). Dostopno na: HTTPs://github.com/adafruit/RTClib/.

[12] HTTP://upload.wikimedia.org/wikipedia/commons/thumb/c/c7/Microchip_rfid_rice.jpg/220px-Microchip_rfid_rice.jpg

[13] HTTP://sweb.cityu.edu.hk/sm2240/2009/09/RFIDtagA.jpg

[14] HTTP://www.avonwood.co.uk/upload/125KHz-30mm-Disk-Tag.jpg

[15] HTTP://www.easybizchina.com/picture/product/201007/12-0250d140-fca0-48b7-b3fa-d369f695c846.jpg

[16] HTTP://www.idplate.com/Images/products/RFID_standard-large(0pu6x3).jpg

[17] HTTP://www.ns-tech.co.uk/blog/wp-content/uploads/2010/01/tag_nocover.jpg

[18] HTTP://arduino.cc/en/uploads/Main/ArduinoADKFront.jpg

[19] HTTP://shieldlist.org/templates/images/stacked-shields.jpg

[20] HTTP://arduino.cc/en/uploads/Hacking/PinMap2560.png

[21] HTTP://arduino.cc/en/uploads/Guide/ArduinoWithEthernetShield.jpg