UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA

ZAKLJUČNA NALOGA

GRAFIČNA VODNA ZAVESA

KRISTJAN UGRIN

UGRIN 2014

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Grafična vodna zavesa

(Graphic water printer)

Ime in priimek: Kristjan Ugrin
Študijski program: Računalništvo in informatika
Mentor: izr. prof. dr. Peter Korošec

Koper, avgust 2014

Ključna dokumentacijska informacija

Ime in PRIIMEK: Kristjan UGRIN

Naslov zaključne naloge: Grafična vodna zavesa

Kraj: Koper

Leto: 2014

Število listov: 47

Število slik: 18 Število tabel: 1 Število referenc: 31

Mentor: izr. prof. dr. Peter Korošec

Ključne besede: grafična vodna zavesa, elektronika, krmilnik, programska oprema

Izvleček: Grafična vodna zavesa je naprava, ki omogoča izris grafike s pomočjo padajočih curkov vode. V uvodu je v splošnem predstavljena grafična vodna zavesa, sledi opis strojne opreme ter razvoj programske opreme za grafično vodno zaveso. Razvoj programske opreme za grafično vodno zaveso je nadaljnje razdeljen in opisan po poglavjih. Najprej sledi opis uporabljenih orodij za razvoj le te, opis razvoja programske opreme za mikrokrmilnik ter razvoj uporabniška programske opreme. Uporabniška programska oprema kronološko opisuje razvoj od osnovne namembnosti do dodatnih razširitvenih modulov, ki dopolnjujejo grafično vodno zaveso. V zaključku je opisan še povzetek celotnega dela in možnosti za razvoj projekta v prihodnosti.

Key words documentation

Name and SURNAME: Kristjan UGRIN

Title of the final project paper: Graphic water printer

Place: Koper

Year: 2014

Number of pages: 47

Number of figures: 18 Number of tables: 1 Number of references: 31

Mentor: Assoc. Prof. Peter Korošec, PhD

Keywords: water printer, electronics, microcontroller, software

Abstract: Water printer is a device, which prints graphical objects using water jets. There is a general depiction of a graphical water printer in the introduction, followed by a description of hardware and software development. Software development is further divided and described in chapters. First, a description of used tools for software development, software development for microcontroller and last, a development of user software. User software chapter chronologically describes the development from elementary purpose to expansion modules, which enhance the graphical water printer. A full project summary and possible future development follows in the concluding chapter.

ZAHVALA

Zahvaljujem se mentorju izr. prof. dr. Petru Koroščcu za sprejem pod mentorstvo in svetovanje. Poleg tega se zahvaljujem kolegu Janiju Černetu za poglavitno idejo, izdelavo strojne opreme, vsesplošno sodelovanje ter podporo pri projektu, ravno tako se zahvaljujem Simonu Pribcu za izdelavo načrta elektronike za upravljanje z ventili. Zahvala gre tudi staršem za podporo v študijskih letih.

KAZALO VSEBINE

KAZALO PREGLEDNIC

KAZALO SLIK

KAZALO ALGORITMOV

SEZNAM KRATIC

CPU – (angl.) Central Processing Unit, centralna procesna enota
CSS – (angl.) Cascading Style Sheets, prekrivni slogi
FAT – (angl.) File Allocation Table, datotečni sistem s tabelo dodeljevanj datotek
HTML – (angl.) HyperText Markup Language, Označevalni jezik za oblikovanje večpredstavnostnih dokumentov
HTTP – (angl.) HyperText Transfer Protocol, protokol za prenos hiperteksta
IDE – (angl.) Integrated Development Environment, integrirano razvojno okolje
I/O – (angl.) Input / Output, Vhod / Izhod
MVC – (angl.) Model-View-Controller, Model-Pogled-Krmilnik
RAM – (angl.) Random Access Memory, pomnilnik z naključnim dostopom
RGB – (angl.) Red Green Blue, rdeča modra zelena
SD – (angl.) Secure Digital, trajno zapisljiva vrsta pomnilnika
SMS – (angl.) Short Message Service, sistem za pošiljanje krajših besedilnih sporočil po mobilnem omrežju
USB – (angl.) Universal Serial Bus, univerzalni zaporedni vmesnik

1 UVOD

Voda je v naravi življenjskega pomena, vendar pa ima lahko tudi estetski pomen. Naravni vodni slapovi so atraktivni za številne množice, zato smo s tako imenovano grafično vodno zaveso želeli ujeti čar vodnih slapov, ter jih razširiti s praktično uporabnostjo v obliki interaktivnega sporočanja za privabljanje posameznikov ali skupin.

Grafična vodna zavesa je naprava, ki posnema slap, tako da v natančno določenem zaporedju in časovnih razmikih preklaplja (odpira in zapira) vodne ventile. Na tak način omogoča nadzorovano ustvarjanje vodnih curkov in s tem zmožnost sporočanja s pomočjo oblikovanja padajoče vode, tako da le ta prikazuje dele besedila ali celo grafike.

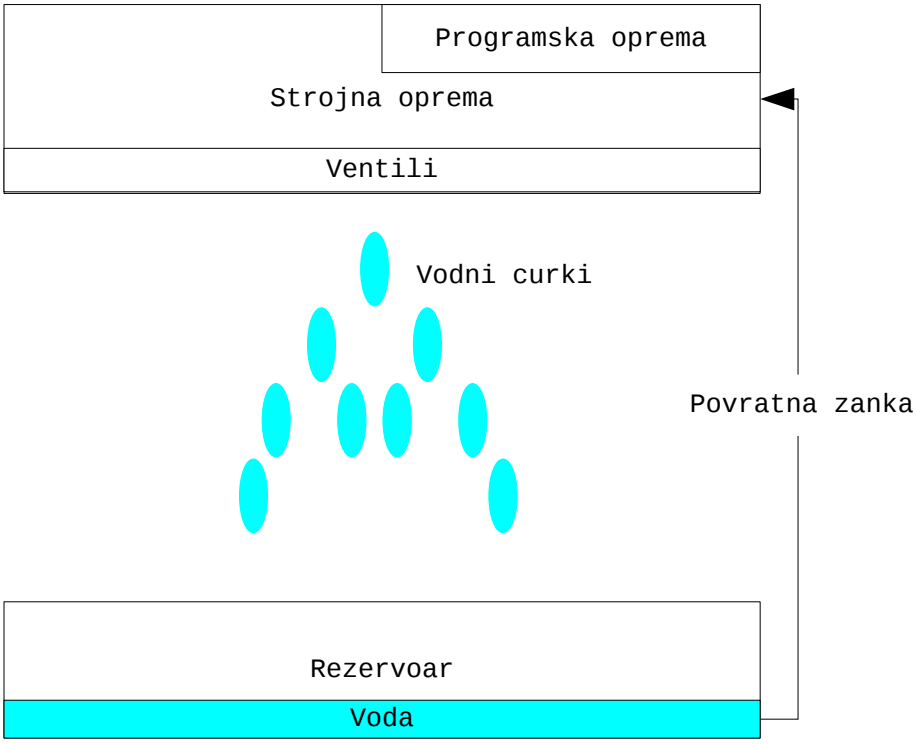
Interakcija z napravo je možna preko spletnega vmesnika, ki deluje na tabličnem računalniku, namiznem računalniku ali drugi napravi s podporo spletnim tehnologijam. Z vgrajeno ustrezno tehnologijo grafična vodna zavesa omogoča tudi prebiranje in prikaz kratkih tekstovnih sporočil neposredno iz pametnega telefona ali kar od ponudnika mobilnih storitev.

Tako lahko z grafično vodno zaveso opazovalcem na ustvarjalen način prikazujemo kratka grafična ali tekstovna sporočila, udeležencem dogodka aktivno prikazujemo ključne dele predstavitve, ali pasivno v ozadju dopolnjujemo aranžma.

V zaključni projektni nalogi bo predstavljen razvoj programske opreme za grafično vodno zaveso. Na začetku bo opisan strojni del grafične vodne zavese, osnovno delovanje naprave (po principu črne škatle), ter vmesnik elektronike za upravljanje izhodnih ventilov, katerega krmili mikrokrmilnik. V nadaljevanju sledi pojasnitev izbire primernega mikrokrmilnika, opis uporabljenih razvojnih orodij ter razvoja programske opreme za mikrokrmilnik. Razvoj programske opreme bo predstavljen v več delih, glede na dodano zmožnost posamezne različice (komunikacija z elektroniko, uporaba izmenljivega medija za shranjevanje podatkov, komunikacija preko USB priključka). Sledil bo opis razvoja programske opreme za upravljanje z grafično vodno zaveso ter razširitvami, ki grafični vodni zavesi dodajo praktično uporabnost (pretvarjanje grafičnih formatov v interni format, pošiljanje grafike na vodno zaveso preko USB priključka, interaktivni spletni vmesnik, pretvorba znakovnih nizov v grafični format ter prejemanje kratkih sporočil). V zaključku bodo predstavljene ideje in možnosti za nadaljnje delo na projektu grafična vodna zavesa.

2 GRAFIČNA VODNA ZAVESA

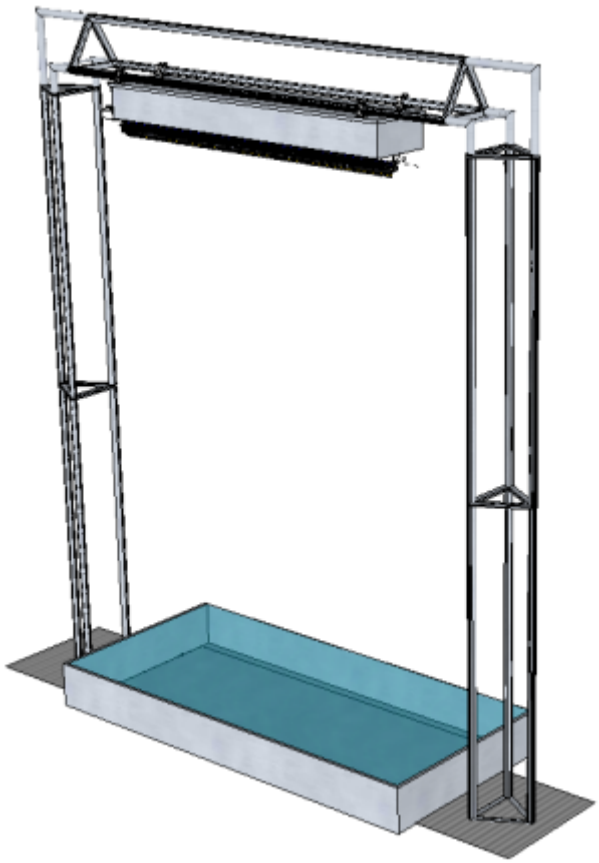
Grafična vodna zavesa je naprava, ki spušča vodne curke v določenem časovnem zaporedju, za izris padajoče grafike ali besedila po zraku. Za doseganje tega učinka je grafična vodna zavesa sestavljena iz treh komponent: strojne opreme, programske opreme ter vode – medij za vizualni izris podatkov, kot je prikazano na sliki 1 [1].



Slika 1: Grafična vodna zavesa

Strojna oprema sestoji iz elektronike, ventilov ter priključka cevi za dovod vode. S priključitvijo vodnega rezervoarja na strojno opremo tvorimo povratno zanko, po kateri kroži voda od rezervoarja prek dovoda ventilov, ter do ventilov samih. Elektronika preklaplja ventile, ki spuščajo vodne curke. Delovanje elektronike vodi programska oprema mikrokrmilnika. Mikrokrmilnik s programsko opremo lahko samostojno vodi delovanje grafične vodne zavese z branjem grafičnega podatkovnega zapisa iz medija (opisano bo več vrst dostopa do podatkovnih zapisov). Pretvorba grafike in besedila v zapis grafične vodne zavese pa se vrši na PC-ju (angl Personal Computer).

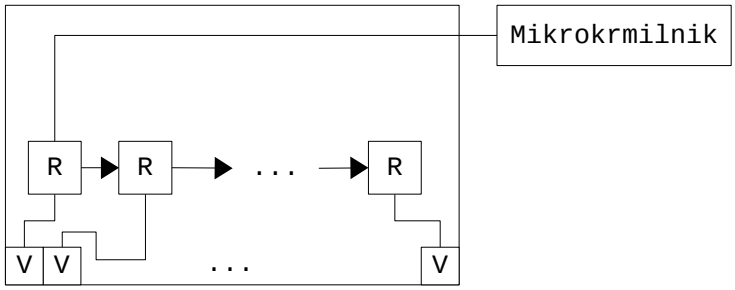
Strojna oprema grafične vodne zavese je zaprta v ogrodje, ki se lahko postavi na želeno višino s pomočjo primernih nosilcev, kot je razvidno iz slike 2.



Slika 2: Fizična postavitev grafične vodne zavese

3 STROJNA OPREMA

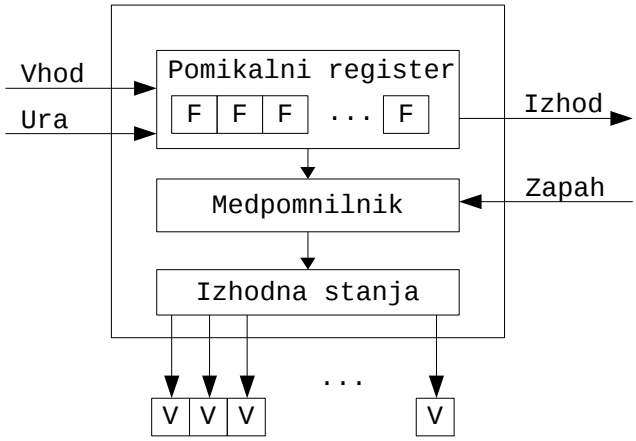
Strojno opremo sestavlja elektronika s pomikalnim registrom (angl. Shift register) [2], ki sprejema podatke o stanju ventilov ter jih ustrezno zapira ali odpira. Podatki o stanju ventilov so predstavljeni kot polje bitov (angl. Bit array), kjer posamezen bit binarno določa stanje ventila (0 zaprt, 1 odprt). Vsaka strojna konfiguracija grafične vodne zavese ima vnaprej določeno število ventilov, zato je tudi dolžina polja bitov vnaprej določena. Polje bitov tvori mikrokrmilnik, s katerim vodi padanje vodnih curkov, kot je razvidno iz slike 3. Polje bitov se pošilja zaporedno vezanim pomikalnim registrom (R). Stanje bitov v pomikalnih registrih, se preslika na izhode (V).



Slika 3: Strojni del z mikrokrmilnikom

3.1 Pomikalni register

Pomikalni register, ki je uprizorjen na sliki 3, je vezje zaporedno vezanih flip flop elementov [3], ki bere, shranjuje in pomika bite iz vhoda v določeni smeri. V vsakem trenutku lahko preberemo shranjeno stanje iz poljubnega flip flop elementa (možna stanja so 0 ali 1). V našem primeru uporabljamo pomikalni register z zapahi (angl. Latched shift register), ki med flip flop elementi ter izhodi vsebuje še medpomnilnik. Medpomnilnik omogoča ohranjanje stanja na izhodih med pomikanjem bitov med posameznimi flip flop elementi.



Slika 4: Pomikalni register z zapahi

Ob prehodu urine periode na priključku z oznako Ura, se obstoječi biti v pomikalnem registru pomaknejo v smeri izhoda za eno mesto. V začetni flip flop element (oznaka F) se prebere in shrani bit iz vhoda. Ob poljubnem prehodu urine periode na priključku z oznako zapah, se stanje v pomikalnem registru preslika v medpomnilnik. Stanje na izhodnih priključkih je vedno enako stanju v medpomnilniku. S takim sistemom lahko nemoteno pošiljamo niz bitov v pomikalni register, izhod pa se spremeni le ob sprostitvi zapaha. Z izhodom B je možno zaporedno povezati več pomikalnih registrov in tako povečati število izhodov (V) ter posledično omogočiti priklop večjega števila naprav.

Pomikalni register smo uporabili v povezavi z mikrokrmilnikom. Z enim digitalnim izhodom mikrokrmilnika, lahko krmilimo večje število ventilov. Brez pomikalnih registrov bi v nasprotnem primeru potrebovali več zaporedno vezanih mikrokrmilnikov, kar je neekonomično.

3.2 Mikrokrmilnik

Mikrokrmilnik je računalnik manjših dimenzij na enem vezju, ki vsebuje centralno procesno enoto (angl. Central processing unit, s kratico CPU), pomnilnik z naključnim dostopom za začasno obdelavo podatkov (angl. Random access memory, s kratico RAM), programabilni ali statični pomnilnik za shrambo programa ter vhodno / izhodne enote (angl. Input / Output, s kratico I/O) [4].

Pri razvoju grafične vodne zavese sta bila uporabljena mikrokrmilnika Arduino ter Teensy, pri čemer je bil v razvojnem ciklu Arduino opuščen (razlogi so navedeni v poglavju 3.3 o zaporedni komunikaciji). Mikrokrmilnik smo načrtno fizično postavili bližje elektroniki grafične vodne zavese, za izognitev morebitnim elektromagnetnim motnjam iz okolice in posledično nepravilnega delovanja celotne naprave.

Pri izbiri mikrokrmilnika smo bili pozorni na fizične zmožnosti, dimenzije, število združljivih programskih knjižic in ceno.

Preglednica 1: Primerjava izbranih mikrokrmilnikov

	Arduino Uno SMD (Atmega328) [5]	**Teensy 3.0 (MK20DX128)** [6]
CPU	8 bit AVR, 16 MHz	32 bit ARM Cortex-M4, 48 MHz
Statični pomnilnik	32 KB	128 KB
RAM	2 KB	16 KB
Digitalni izhodi	14	24
Zanesljiva USB komunikacija	Ne[1]	Da
Velikost	3,6 × 1,8 cm	6,9 × 5,4 cm
Programska podpora	Arduino IDE, SDFat	Kompatibilen z Arduino IDE, SDFat

Preglednica 1: Medtem ko obe končno uvrščeni izbiri ustrezata večini pogojev, Arduino ni primeren zaradi načina implementacije univerzalnega zaporednega vodila (angl. Universal Serial Bus, s kratico USB).

3.3 Zaporedna komunikacija

Zaporedna komunikacija predstavlja sekvenčni prenos bitov po žici. Osnovna komunikacija med napravami se vrši tako, da pošiljatelj pošilja bit za bitom iz medpomnilnika, medtem ko prejemnik na drugi strani žice prejema in polni svoj medpomnilnik (v tem primeru ga imenujemo medpomnilnik zaporednega vmesnika) [7]. Ko se prejemnikov medpomnilnik napolni, ne more prejemati novih bitov in je potrebno pošiljanje začasno zaustaviti – to se rešuje s »stop« bitom, katerega prejemnik sporoči pošiljatelju. Primer implementacije take komunikacije je RS232 protokol [8].

Arduino Uno uporablja Atmega8U2 čip za zaporedno komunikacijo prek USB vmesnika, ki implementira zaporedni protokol, brez signalizacije polnega medpomnilnika [9]. V primeru polnega medpomnilnika, se na novo poslani biti izgubijo oziroma pride do tako imenovane prekoračitve medpomnilnika (angl. Buffer overflow). To v naši projektni nalogi ni zaželeno, saj je cilj grafične vodne zavese konsistenten izris grafike in posledično zahteva natančen prenos podatkov. Zaradi tega razloga je bil izbran Teensy 3.0.

[1]Arduino Uno ne premore zaščite pred prekoračitve medpomnilnika

4 RAZVOJ PROGRAMSKE OPREME

Pri razvoju programske opreme smo najprej popisali želje in zahteve grafične vodne zavese. Z grafično vodno zaveso želimo kot vhod imeti rastrske slike (npr.: png, jpeg), ki jih je enostavno obdelati v poljubnem urejevalniku rastrskih slik ter so največ dvobarvne oziroma konkretno v našem primeru črno-bele. Razlog za omejitev števila barv tiči v tem, da princip izrisovanja temelji na padajočih curkih vode, ki so v določenem trenutku prisotni ali odsotni na določeni točki v zraku. Posledično imamo največ dve možni različni stanji, ki jih lahko popišemo z dvema različnima barvama. Programsko opremo smo ločili na dva dela:

- uporabniško programsko opremo ter
- programsko opremo mikrokrmilnika.

Uporabniška programska oprema zahteva pretvorbo rastrskih slik v primeren podatkovni zapis za mikrokrmilnik, ki ga imenujemo podatkovni zapis grafične vodne zavese. Ker mikrokrmilnik vodi časovno občutljivo pošiljanje podatkov o stanju ventilov, ne želimo dodatnih zakasnitev z obdelavo podatkov pri prebiranju podatkovnega zapisa grafične vodne zavese. Tako se pojavi zahteva po čim bolj preprostem podatkovnem zapisu grafične vodne zavese. Podatkovni zapis grafične vodne zavese je potrebno prenesti na mikrokrmilnik, kar doda zahtevo po komunikaciji uporabniške programske opreme z mikrokrmilnikom s pomočjo zaporedne komunikacije.

Programska oprema mikrokrmilnika mora zagotavljati časovno natančno pošiljanje podatkov elektroniki grafične vodne zavese. Ker smo na mikrokrmilniku precej bolj omejeni s procesorsko močjo ter pomnilnikom v primerjavi s PC-jem, je kritičnega pomena, da v programski opremi mikrokrmilnika ne izvajamo operacij nad prebranimi podatki, oziroma le te prenesemo na uporabniški del programske opreme.

Končni seznam zahtev za celotno programsko opremo grafične vodne zavese je popisan v sledečem seznamu:

- enostaven zapis podatkov (ne glede na medij),
- pretvornik rastrskih slik v enostavni podatkovni zapis grafične vodne zavese,
- prebiranje datotek s podatkovnim zapisom grafične vodne zavese na mikrokrmilniku,
- pošiljanje podatkov iz mikrokrmilnika elektroniki grafične vodne zavese,
- komunikacija z mikrokrmilnikom preko USB-ja in
- razširitev s programskimi moduli.

Za razvoj programske opreme grafične vodne zavese, smo uporabili ustrezna razvojna orodja, katerih opis sledi.

4.1 Razvojna orodja

Razvoj uporabniške programske opreme je potekal v programskem jeziku Java [10]. V te namene je bilo izbrano integrirano razvojno okolje (angl. Integrated Development Environment, s kratico IDE) Eclipse [11]. IDE je celovito okolje za razvoj programske opreme, sestavljeno iz razhroščevalnika, prevajalnika in urejevalnika kode [12]. Ker Javino ogrodje nima specifične implementacije zaporedne komunikacije za operacijski sistem, smo za te namene uporabili RXTX knjižico [13].

Za razvoj programske opreme mikrokrmilnika smo uporabili C jezik ter Arduino IDE [14]. Ker uporabljamo nekoliko drugačen mikrokrmilnik, smo za združljivost Arduino razvojnega okolja s Teensy mikrokrmilnikom namestili Teensyduino programski dodatek [15]. Pisanje same kode za mikrokrmilnik je potekalo v zunanjem tekstovnem urejevalniku, zaradi boljšega označevanja sintakse ter hitrejšega urejanja. Arduino razvojno okolje smo uporabili samo za prevajanje programa ter nalaganje na mikrokrmilnik.

Sledi opis postopnega razvoja programske opreme, in sicer v smeri od programske opreme za mikrokrmilnik do uporabniške programske opreme ter programskih modulov.

4.2 Programska oprema mikrokrmilnika

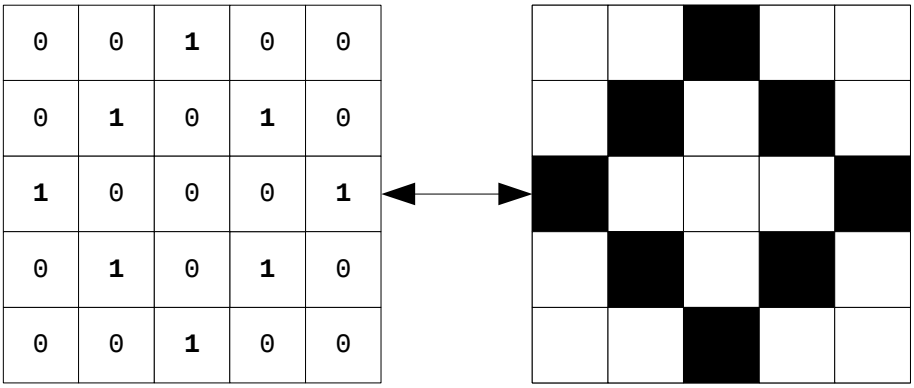
Naloga programske opreme mikrokrmilnika je vodenje elektronike grafične vodne zavese, ki preklaplja ventile. Za natančen izris slike je potrebno ohranjati enak časovni interval med pošiljanjem podatkov, zato je kritičnega pomena, da mikrokrmilnik ne počne ničesar drugega kot osnovne funkcije za doseganje namena. Med te spada branje ter pošiljanje podatkov na elektroniko grafične vodne zavese. Za hitro prebiranje podatkov morajo biti le ti zapisani tudi v čim bolj primernem formatu.

4.2.1 Podatkovni zapis grafične vodne zavese

Grafična vodna zavesa ima vnaprej določeno število ventilov, vsak ventil ima v določenem trenutku eno izmed dveh možnih preklopnih stanj (odprt ali zaprt). Bit je najmanjša možna enota, s katero opišemo stanje ventila v določenem trenutku. Bit z vrednostjo 0 pomeni zaprto, 1 pa odprto stanje. Ker se želimo izogniti dodatni procesorski obdelavi ob prebiranju podatkov, je najenostavnejši način za opis stanja sekvenčno nizanje bitov. Zaradi vnaprej znanega števila ventilov v določeni strojni izvedbi grafične vodne zavese nam ni potrebno v zapisu definirati posebnega stanja bitov, ki določajo prelom vrstice, temveč tega lahko hitro in učinkovito določimo ob samem prebiranju podatkovnega zapisa grafične vodne zavese. Natančno določanje preloma vrstice dobimo z deljenjem po modulu števila ventilov na grafični vodni zavesi s številom prebranih bitov. V primeru, da je ostanek enak 0, smo na točki preloma vrstice.

Enostaven primer popisa petih zaporednih stanj za grafično vodno zaveso s petimi ventili, v matriko bitov, je prikazan na sliki 5. Črno polje predstavlja odprt ventil (bit z

vrednostjo 1), belo pa zaprt (bit z vrednostjo 0). Vsaka vrstica matrike določa stanje ventilov v določenem trenutku (hkraten prehod stanj med vrsticami, z drugo besedo sinhron), stolpec pa stanje posameznega ventila. Težava takega podatkovnega zapisa je vrstni red bitov. Zapis se bere od prvega bita v zapisu naprej, kar pomeni izris od prve vrstice proti zadnji. Na ta način dobimo zrcaljeno sliko, težavo pa bomo rešili na višjem nivoju, da ne vpeljemo dodatnega procesiranja na mikrokrmilniku. S tem principom smo zelo blizu preslikave rastrske slike v lastno definiran bitni zapis, vendar bo to podrobneje opisano v poglavju 4.3.2 o pretvorbi rastrskih slik. V naslednjem poglavju bomo opisali pošiljanje podatkovnega zapisa na elektroniko grafične vodne zavese.



Slika 5: Primer preslikave med zapisom grafične vodne zavese ter stanj ventilov

4.2.2 Krmiljenje elektronike grafične vodne zavese

Grafično vodno zaveso krmilimo s pošiljanjem bitov na vmesnik elektronike grafične vodne zavese. V psevdokodi smo uporabili sledeče spremenljivke za primerne priključke in sicer, clockPin za uro, dataPin za vhod ter latchPin za zapah. Stanje z visoko napetostjo na priključku predstavlja bit z vrednostjo 1 na izhodu, nizko pa bit z vrednostjo 0. Točna napetost visokega ter nizkega stanja je odvisna od samega vezja, za namene tega raziskovalnega projekta, smo samo predpostavili, da visoko stanje predstavlja višjo napetost kot pa nizko stanje. Za začetek smo implementirali krmiljenje ventilov glede na vhodne podatke. Zaradi preprostosti predpostavimo, da so podatki o stanju ventilov fiksno shranjeni v spremenljivki data.

```
1.  set data value

2.  for all bits in data do
3.   if readBits equals number of valves or 0
4.    set latchPin off
5.   endif
6.   shiftOutFast(dataPin, clockPin, data[i])
7.    if readBits equals number of valves
8.     set latchPin on
9.     delay
10.  endif
11. done

12. function shiftOutFast(dataPin, clockPin, dataBit)
13.  set clockPin off
14.  if dataBit == 1
15.   set dataPin on
16.  else
17.   set dataPin off
18.  endif
19.  set clockPin on
20. endfunction
```

Algoritem 1: Psevdokoda za krmiljenje grafične vodne zavese

Primer krmiljenja grafične vodne zavese, je opisan v algoritmu 1. Za vsak bit v spremenljivki data (2. vrstica), ki opisuje stanje ventilov, se izvaja funkcija shiftOutFast. Namen funkcije shiftOutFast je pošiljanje bita v pomikalni register, za to je predhodno potrebno postaviti clockPin v nizko stanje (15. vrstica), zapisati želeno stanje ventila na dataPin priključek (od 16. do 20. vrstice) ter nato clockPin postaviti v visoko stanje (21. vrstica), da pomikalni register premakne obstoječe bite v določeno smer in prebere novo vrednost. V glavni rutini je potrebno po vsaki prebrani vrstici, ki je dolga, kot imamo število ventilov v trenutni konfiguraciji grafične vodne zavese, postaviti latchPin v visoko stanje (8. vrstica). S tem se sproži zapah v pomikalnem registru, nato pa se ventili sinhrono postavijo v prebrano stanje. Priključek latchPin je potrebno ponovno postaviti v nizko stanje, preden začnemo z branjem nove vrstice.

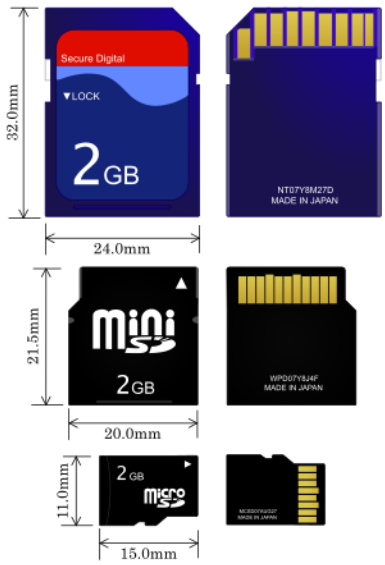
4.2.3 Izmenljiv medij

Naš program v trenutni obliki sicer deluje, toda ni dovolj fleksibilen. V primeru opisanega algoritma 1 bi za vsak nov niz stanj, morali zapisati v spremenljivko data druge vrednosti ter program prevesti in ponovno naložiti v mikrokrmilnik. S tem izgubimo dinamično upravljanje nad podatki stanj in posledično nam v prihodnosti odpade vsaka razširitev s strani uporabniške programske opreme. Zato vpeljemo uporabo izmenljivega medija, ki nam omogoči zamenjavo podatkov stanj brez ponovnega prevajanja in nalaganja programa mikrokrmilnika. V našem primeru je izmenljiv medij mikro SD pomnilniška kartica.

4.2.3.1 SD kartice

SD kartica (angl. Secure Digital, s kratico SD) je pomnilniška kartica, ki vsebuje trajni pomnilnik (angl. Non-volatile memory) [16]. Trajni pomnilnik ohranja zapisane podatke tudi po prekinitvi napajanja le tega. Pomnilnik je razdeljen na različne bloke, v katerih se nahaja zapis podatka. Zaradi te značilnosti pripada skupini bločnih naprav [17]. Kot vse bločne naprave, se pomnilnik na SD kartici naslavlja z naslovom na blok, iz katerega želimo prebrati ali vanj zapisati podatek. Ker podatke želimo združevati v smiselne bloke informacij, potrebujemo datotečni sistem, ki gručam blokov podatkov pripiše dodatne parametre (metapodatke ali angl. Metadata) ter tvori datoteke. Datoteke so standardizirana abstraktna struktura nad podatki, ki so vezane v smiselno celoto [18]. S pomočjo datotek lahko zapišemo stanja, ki nam tvorijo celovit podatkovni zapis grafične vodne zavese. V razvojnem okolju Teensyduino smo za implementacijo datotečnega sistema uporabili SDFat knjižico [19], ki implementira standardiziran FAT datotečni sistem (angl. File Allocation Table, s kratico FAT). Delovanje FAT datotečnega sistema si lahko ogledate v knjigi avtorja Andrew S. Tanebaum z naslovom »Modern Operating Systems«, pod poglavljem »The MS-DOS File System« [20].

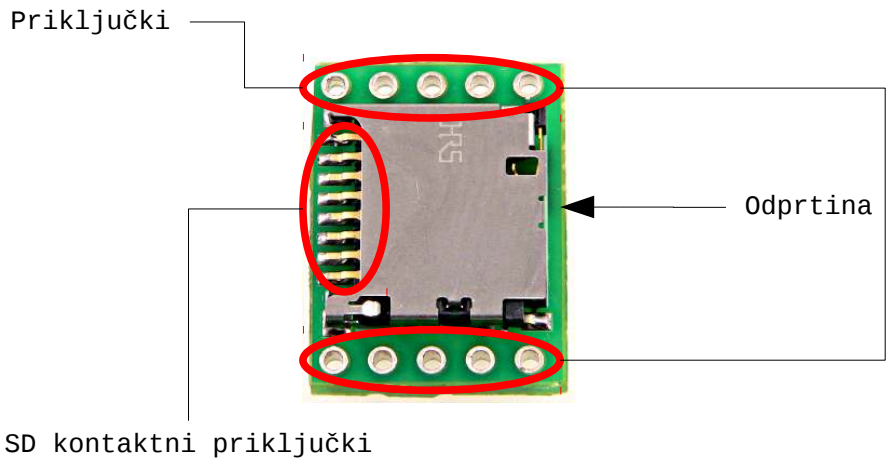
SD kartice so namenjene uporabi na prenosnih napravah kot so npr. telefoni, fotoaparati, tablični računalniki, itd. Obstaja več različic SD pomnilniških kartic, ki se delijo glede na fizično velikost, kot je prikazano na sliki 6. SD (na vrhu), mini SD (na sredini) ter mikro SD (spodnja). V našem primeru smo uporabili mikro SD kartico ter pripadajoči SD fizični vmesnik (angl. Adapter) za pomnilniške kartice.



Slika 6: Različni fizični formati SD kartic

4.2.3.2 SD fizični vmesnik

SD fizični vmesnik je vezje, na katerem leži reža za SD kartice. V reži so kontaktni priključki, ki vodijo k ostalim priključkom z namenom povezave k mikrokrmilniku. Slika 7 prikazuje mikro različico fizičnega SD vmesnika za mikro SD pomnilniške kartice. Le ta se v drugih značilnostih, razen fizične velikosti, ne razlikuje od ostalih fizičnih različic SD vmesnikov. Na desni strani slike pod železnim ščitom je reža za SD kartico (v našem primeru mikro SD). SD kartica se povezuje s fizičnim vmesnikom prek SD kontaktnih priključkov, ki so v tem primeru speljani na priključke ob zgornji in spodnji strani. Stranski priključki v našem primeru služijo za povezavo z mikrokrmilnikom.



Slika 7: Mikro SD fizični vmesnik

Mikro SD kartico uporabimo kot izmenljiv medij, ki jo v povezavi z mikrokrmilnikom lahko naslavljamo in posledično dostopamo do podatkov. Z ustreznimi spremembami obstoječega algoritma 1, lahko SD pomnilniško kartico uporabimo za dostop do podatkov naše grafične vodne zavese.

4.2.4 Krmiljenje pomnilniške kartice

SD pomnilniško kartico, ki je povezana na Teensy mikrokrmilnik, je potrebno ustrezno krmiliti. Primer psevdokode algoritma 2 prikazuje ključne spremembe glede na algoritem 1, ki služijo krmiljenju SD pomnilniške kartica, za dostop do podatkov. Po vključitvi knjižice za upravljanje s SD karticami, lahko dostopamo do vsebine FAT datotečnega sistema. (4. vrstica). Za vsako prebrano datoteko nastavljamo stanja kot v prejšnjem primeru. Ključna sprememba je še razvidna od 16. do 19. vrstice, kjer se poskrbi za zaprtje ventilov po prebrani datoteki in želen premor pred odpiranjem naslednje.

```
1.  import SDFat
2.  init sdCard
3.  sd.rewind

4.  for all files in sd.files do
5.   set data file.content
6.   for all bits in data do
7.    if readBits equals number of valves or 0
8.     set latchPin off
9.    endif
10.   shiftOutFast(dataPin, clockPin, data[i])
11.    if readBits equals number of valves
12.     set latchPin on
13.     delay
14.   endif
15.  done
16.  set latchPin off
17.  shiftOutFast(dataPin, clockPin, 0)
18.  set latchPin on
19.  delay
20. done
```

Algoritem 2: Primer razširitve algoritma 1 za branje datotek iz SD kartice

Z neprekinjenim izvajanjem algoritma 2, program neprekinjeno teče in grafična vodna zavesa izrisuje vodne grafične objekte. Grafična vodna zavesa tako postane avtonomna naprava, vsebino na SD kartici pa lahko upravljamo s pomočjo SD čitalnika na PC-ju. Algoritem 2 bomo izkoristili v nadaljevanju za upravljanje mikrokrmilnika grafične vodne zavese preko USB vmesnika.

4.2.5 Upravljanje mikrokrmilnika preko USB vmesnika

Teensyduino omogoča zaporedno komunikacijo preko USB vmesnika, kar lahko tudi izkoristimo za upravljanje vsebine na SD kartici brez prekinjanja glavnega programa ali napajanja.

Teensyduino implementira metodo za branje iz zaporednega vmesnika, kot je razvidno iz algoritma 3. Metoda serialEvent se kliče, ko so podatki na voljo v medpomnilniku zaporednega vmesnika. Implementacijo lahko uporabimo na dva načina – za neposredno vodenje grafične vodne zavese (mikrokrmilnik kot zaporedni most), ter za že omenjeno upravljanje vsebine na SD kartici. Prvi primer smo opustili, ker se je iz testiranj izkazalo, da USB nima zagotovljenih dovolj nizkih zakasnitev, da bi zadovoljil potrebo po časovni kritičnosti. Podatki lahko zamudijo časovni okvir v katerem bi morali biti na voljo in zamaknejo sinhrone prehode stanj med ventili, kar se izkaže kot popačen izris grafike.

V primeru implementacije za upravljanje s SD kartico, moramo definirati pomen bitov, ki jih preberemo iz medpomnilnika zaporednega vmesnika. Na ta način lahko programska oprema mikrokrmilnika razpozna pomen prejetih bitov kot ukaz. Sama interpretacija bitov na opisan način ni dovolj za npr. pošiljanje vsebine skozi zaporedni

vmesnik, zato potrebujemo še dodatne opisne parametre. Teensyduino omogoča branje 8 bitov oziroma enega zloga (angl. Byte) z vsakim klicem metode Serial.read, ki pripada SDFat knjižici.

Primer bitnega toka iz zaporednega vmesnika, razdeljenega na glavo in zaglavje, je prikazan na sliki 8. Glavo smo razdelili na dva dela, prvih osem bitov uporabimo za opis ukaza, s katerim se programska oprema mikrokrmilnika odloči za ustrezen tok odvijanja programa. S tem želimo popisati in izvajati naslednje akcije:

- Glava z osem bitnim številom 0 ali »00000000« binarno, pomeni brisanje vse vsebine na SD kartici ter
- Glava z osem bitnim številom 1 ali »00000001« binarno, pomeni zapis nove datoteke na SD kartico.

Slika 8: Primer strukture bitnega toka na serijskem vmesniku

Za implementacijo zapisa nove datoteke na SD kartico potrebujemo še naslednjih 64 bitov, ki nam delujejo kot števec za velikost vsebine v zaglavju. V našem primeru zaglavje predstavlja vsebino datoteke.

```
1.  set serialByteCount 0
2.  set actionId -1

3.  function serialEvent()
4.   set incomingByte serial.readByte
5.   if serialByteCount less than 1
6.    set actionId incomingByte
7.   endif
8.   set serialByteCount increment
9.   switch actionId
10.   if 0: deleteFiles()
11.   if 1: writeFile(incomingByte)
12.  endswitch
13. endfunction

14. function deleteFiles()
15.  sd.deleteAll
16.  set actionId -1
17. endfunction

18. function writeFile(incomingByte)
19.  if serial_byteCount greater than 8
20.   if writefile not open
21.    set writefile.filename genFilename()
22.    writefile.open()
23.   endif
24.   writefile.write(incomingByte)
25.   set serialByteCount decrement
26.   if serialByteCount equals 0
27.    set actionId -1
28.    writefile.close()
29.   endif
30.  elseif
31.   shift left byteCounter and merge with incomingByte
32.  endif
33. endfunction

34. function genFilename
35.  set lastFileNumber sd.getLastFile.name
36   set fileName lastFileNumber increment
37.  set fileExtension 'BWPR'
38   return fileName + fileExtension
39. endfunction
```

Algoritem 3: Branje iz medpomnilnika zaporednega vmesnika ter delo s podatki na SD kartici

Implementacija branja iz medpomnilnika zaporednega vmesnika ter razpoznavanje glave in pomena podatkov, je opisana v algoritmu 3. Funkcija serialEvent se izvede, ko so na voljo podatki v medpomnilniku zaporednega vmesnika. Glede na prvi niz 8 bitov v glavi, se odloči za ustrezno akcijo (brisanje datotek, naložitev nove datoteke). Brisanje datotek je opisano v funkciji deleteFiles. Ker je opravilo trivialno, v glavi ne potrebujemo dodatnih podatkov. Za zapis nove datoteke sledi prvim 8 bitom še 64 bitni števec. Vrednost incomingByte se pomika v spremenljivko byteCounter, kot je navedeno v 31. vrstici. V primeru prebranega števca, spremenljivka serialByteCount nosi vrednost večjo od 8, zato

jo uporabimo za nadaljnje usmerjanje programa ter zapis podatkov v datoteko. Datoteki dodelimo ime, ki je vedno naravno število. Ime se povečuje z vsako novo datoteko. Novo datoteko odpremo za pisanje ter vanjo zapisujemo podatke iz medpomnilnika. Ko serialByteCount doseže vrednost 0 pomeni, to pomeni, da je datoteka v celoti zapisana, zato writefile zapremo ter actionId nastavimo na neveljavno vrednost, da program čaka na naslednji ukaz iz glave podatkov.

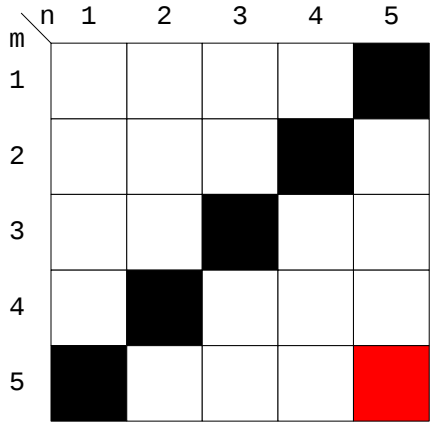
Sedaj imamo na mikrokrmilniku program, ki ga je mogoče upravljati preko USB vmesnika. Sledi opis razvoja uporabniške programske opreme ter zmožnosti, ki nam jih ponuja v navezi s programsko opremo mikrokrmilnika.

4.3 Uporabniška programska oprema

Uporabniška programska oprema je namenjena upravljanju grafične vodne zavese kot celote, torej teče neodvisno od mikrokrmilnika na grafični vodni zavesi. V osnovi je njen namen pretvorba rastrskih slik v zapis grafične vodne zavese, ki ga lahko bere program mikrokrmilnika. Kasneje bomo uporabniško programsko opremo razširili in dodali zmožnost pošiljanja pretvorjenega zapisa preko USB vmesnika, ter si ogledali še nekaj dodatnih razširitvenih modulov, ki jih nam kot celota omogoča. Uporabniška programska oprema teče na PC-ju in je pisana v različnih programskih jezikih. Najprej si oglejmo osnovno definicijo rastrske slike, nato bomo poglavje povezali v uporabniško programsko opremo.

4.3.1 Rastrska slika

Rastrska ali bitna slika, je definira kot matrika raznobarvnih pik, ki skupno tvorijo sliko [21]. Vsaka posamezna pika vsebuje informacijo o njeni barvi. Slika 9 prikazuje primer rastrske slike velikosti m=5 × n=5 s črno diagonalno poševnico ter rdečo piko v spodnjem desnem kotu. Celotno sliko lahko opišemo kot matriko, kjer posamezna pika nosi vrednost o barvi. Po RGB modelu [22] lahko informacijo o barvi razdelimo na tri komponente: rdeča, zelena, modra (angl. Red, Green, Blue s kratico RGB). Posamezna komponenta je predstavljena kot osem bitno število, zato je njena vrednost lahko med 0 in 255, kjer 0 predstavlja najnižjo vrednost komponente, 255 pa najvišjo. Za naslednje primere lahko vsako posamezno piko definiramo kot zapis [rdeča, zelena, modra]. Na primer črno piko v matriki na položaju (2, 4) lahko opišemo z vrednostmi [0, 0, 0], belo piko (2, 2) z vrednostmi [255, 255, 255], rdečo piko (5, 5) pa z vrednostmi [255, 0, 0].



Slika 9: Primer rastrske slike

Grafična vodna zavesa izrisuje grafične podobe s pomočjo kratkih curkov vode. Vsako izrisano podobo lahko v določeni točki opišemo z dvema vrednostnima, in sicer kot prisotnost curka. Zaradi tega smo se pri pretvorbi omejili na obdelavo rastrskih slik z dvema možnima barvama – črno ali belo.

4.3.2 Pretvorba rastrskih slik

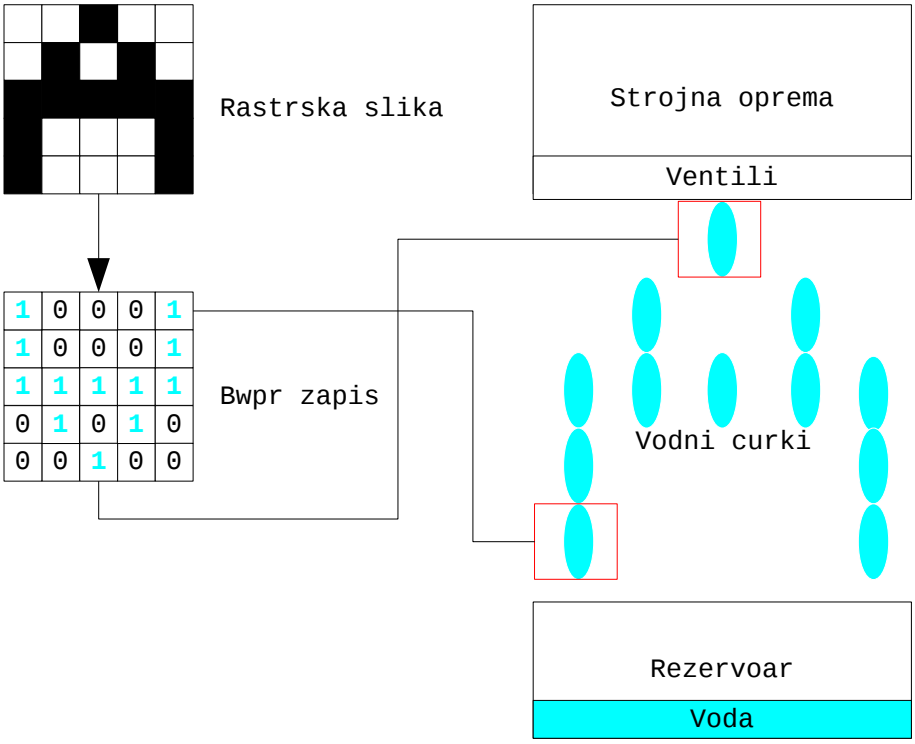
Osnovni namen uporabniške programske opreme je pretvorba rastrskih slik v zapis grafične vodne zavese, katerega smo v nadaljevanju poimenovali binarni zapis grafične vodne zavese (angl. Binary Water Printer, s končnico datoteke »BWPR«). Zato smo najprej razvili pretvornik rastrskih slik v BWPR zapis. Pretvornik je pisan v programskem jeziku Java.

```
1.  set inputDir argument1
2.  set outputDir argument2

3.  function convertFiles()
4.   for all files in inputDir do
5.    set bwprFile convertFile(file)
6.    save bwprFile to outputDir
7.   endfor
8.  endFunction

9.  function convertFile(file)
10.  set rasterMatrix file.data
11.  for all rows in rasterMatrix from last to first do
12.   for all colums in row do
13.    if rasterMatrix[row, column] equals black
14.     set outputBitString append 1
15.    else if rasterMatrix[row, column] equals white
16.     set outputBitString append 0
17.    endif
18.   endfor
19.  endfor
20.  return outputBitString
21. endfunction
```

Algoritem 4: Pretvorba rastrske slike v BWPR zapis

Za pretvorba črno-bele rastrske slike v BWPR zapis, moramo datoteko rastrske slike naložiti v matriko rasterMatrix, kot je prikazano v algoritmu 4. Za vsako vrednost v matriki zapišemo en bit v končni bitni niz – 1 za črno piko [0, 0, 0], 0 za belo [255, 255, 255]. Matriko beremo od zadnje vrstice proti prvi – tako sliko zrcalimo horizontalno, ter rešimo problem, omenjen v poglavju 4.2.1 o zapisu podatkov. Končni bitni niz zapišemo v izhodni imenik na datotečnem sistemu. Algoritem za pretvorbo ponavljamo, dokler ne pretvorimo vseh slik v vhodnem imeniku.

Izhodne datoteke iz izhodnega imenika lahko sedaj naložimo na SD kartico ter jih vstavimo v fizični vmesnik pri mikrokrmilniku, ki nam bo z njegovim programom izrisoval grafiko.



Slika 10: Pretvorba v BWPR zapisa in izris na grafični vodni zavesi

Primer pretvorbe rastrske slike velikosti 5 × 5 v BWPR zapis, in pripadajoč izris na grafični vodni zavesi, je prikazan na sliki 10. Konfiguracija vodne zavese je hipotetična zaradi enostavnejše razlage (število ventilov v pravi izvedbi bi moralo biti deljivo z 8). Vsak bit v vodni zavesi predstavlja curek na izrisani sliki. Ker poznamo število ventilov na grafični vodni zavesi, lahko BWPR zapis pretvorimo nazaj v matriko. Število bitov v nizu delimo s številom ventilov in dobimo število stolpcev v matriki, bitni niz pa prepišemo v matriko. Sedaj bomo uporabniško programsko opremo nadgradili z zmožnostjo upravljanja mikrokrmilnika prek USB vmesnika.

4.3.3 Komunikacija z mikrokrmilnikom

Grafična vodna zavesa je postavljena na višini več metrov, pogosto je njen mikrokrmilnik s SD kartico fizično nedostopen. V ta namen smo uporabniško programsko opremo razširili ter izkoristili implementacijo zaporedne komunikacije na mikrokrmilniku za krmiljenje grafične vodne zavese preko vmesnika USB. Komunikacijski del z mikrokrmilnikom je pisan v programskem jeziku Java.

```
1.  set commandType argument1
2.  set inputDir argument2
3.  initialize Serial

4.  switch commandType
5.   if delete: deleteSD()
6.   if upload: uploadFiles(inputDir)
7.  endswitch

8.  function deleteSD
9.   Serial.send('00000000')
10. endfunction

11. function uploadFiles(inputDir)
12.  deleteSD()
13.  for all files in inputDir do
14.   Serial.send('00000001')
15.   set bwprFile convertFile(file)
16.   Serial.send(bwprFile.byteSize)
17.   Serial.send(bwprFile)
18.  endfor
19. endfunction
```

Algoritem 5: Pošiljanje BWPR zapisa na mikrokrmilnik

Z dopolnitvijo algoritma 4 dodamo zmožnost pošiljanja BWPR zapisa na zaporedni vmesnik (algoritem 5). Glede na izbrano akcijo, program bodisi pošlje ukaz za brisanje SD kartice bodisi sproži postopek nalaganja BWPR zapisa na SD kartico. Za brisanje SD kartice je dovolj, če tvorimo in pošljemo glavo z 8 ničelnimi biti. V primeru pošiljanja datoteke je potrebno tvoriti večjo glavo in najprej poslati niz bitov '00000001', kar mikrokrmilnik sprejme kot ukaz za nalaganje nove datoteke. Sledi 64 bitni del glave, ki opisuje velikost vsebine BWPR zapisa. Na koncu se pošlje še sama vsebina BWPR zapisa. Del algoritma za pošiljanje BWPR datotek je opisan od 14. do 17. vrstice, ponavljamo ga, dokler nismo pretvorili in poslali vsake datoteke z rastrsko sliko v vhodni mapi na datotečnem sistemu. S tem dosežemo upravljanje vsebine na SD kartici in posledično spreminjanje programa predstave grafične vodne zavese brez prekinitve delovanja. Uporabniško programsko opremo bomo postopoma dodatno razširili z interakcijo in dodali prvi interaktivni modul.

4.3.4 Interakcija

Ideja o interakciji z grafično vodno zaveso v realnem času služi kot dodatek k privabljanju posameznikov in množic. Pod interakcijo štejemo dodatne module, ki omogočajo upravljanje vsebine na vodni zavesi za krajši čas. S stališča programske opreme, so to posamezni BWPR zapisi, ki se za razliko od sedanjih ne ponavljajo ter imajo prioriteto nad klasičnimi BWPR zapisi. Na mikrokrmilniku potrebujemo spremembo, ki lahko BWPR zapis loči od ostalih.

```
... ...
9.   switch actionId
10.   if 0: deleteFiles()
11.   if 1 or 2: writeFile(incomingByte)
12.  endswitch
... ...

... ...
34. function genFilename
35.  switch actionId
36.   if 1: set lastFileNumber sd.getLastFile.name of 'BWPR'
37.      set fileExtension 'BWPR'
38.   if 2: set lastFileNumber sd.getLastFile.name of 'TMP'
39.      set fileExtension 'TMP'
40.  endswitch
41.  set fileName lastFileNumber increment
42.  return fileName + fileExtension
43. endfunction
... ...
```

Algoritem 6: Označevanje BWPR zapisa

S spremembo algoritma 3 in vpeljavo razpoznavanja zapisa, dobimo algoritem 6. Pri razpoznavanju samega ukaza uporabimo enak princip kot pri klasičnem BWPR zapisu ali takemu za enkratno uporabo. Zato je sprememba v 11. vrstici minimalna, ne glede na tip BWPR zapisa ga še vedno zapisujemo na SD kartico. Večja sprememba je v funkciji genFilename od 34. vrstice naprej. Algoritem mora primerno označiti BWPR zapis, da lahko pri branju ustrezno določi prioritete izrisa. V 35. vrstici se glede na ukaz določi končnica BWPR zapisa. V primeru, da se gre za BWPR zapis iz interaktivnega modula, ga označimo s končnico »TMP«. Sedaj imamo na voljo BWPR zapise, ločene glede na namembnost, potrebujemo spremeniti še samo izvajanje branja zapisov na SD kartici. S spremembo algoritma 2 dobimo algoritem 7, ki pri branju BWPR datotek ločuje glede na končnico. Datoteke z »BWPR« končnico sestavljajo tekočo grafično predstavitev vodne zavese. Sprememba v 5. vrstici omogoča preskok klasičnega zapisa z »BWPR« končnico in dodeli prioriteto zapisu s »TMP« končnico. Ker zapis s končnico »TMP« izvira iz modula za interakcijo, smo tako dosegli namen dodelitve prioritete modulom za interakcijo.

```
... ...
4.  for all 'BWPR' files in sd.files do
5.   if fileExists 'TMP'
6.    set file sd.getFile('TMP');
7.   endif
8.   set data file.content
9.   for all bits in data do
10.   if readBits equals number of valves or 0
11.    set latchPin off
12.   endif
13.   shiftOutFast(dataPin, clockPin, data[i])
14.    if readBits equals number of valves
15.     set latchPin on
16.     delay
17.   endif
18.  done
19.  set latchPin off
20.  shiftOutFast(dataPin, clockPin, 0)
21.  set latchPin on
22.  delay
23.  if file.extension = 'TMP'
24.   sd.delete(file)
25.  endif
26. done
... ...
```

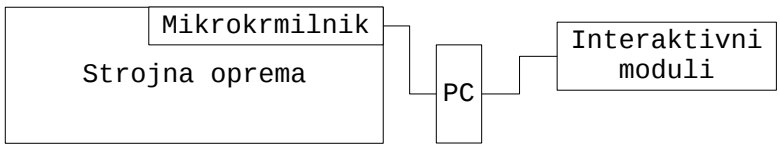
Algoritem 7: Branje BWPR zapisa glede na končnico

Spremembe nam omogočajo vpeljavo prvih implementacij interaktivnih modulov.

4.3.5 Interaktivni modul

Interaktivni modul je modul, ki razširja uporabniško programsko opremo in v realnem času omogoča začasno spreminjanje vsebine na grafični vodni zavesi. Primera interaktivnih modulov, razvitih za grafično vodno zaveso sta:

- spletni vmesnik in
- modul za kratka tekstovna sporočila (angl. Short Message Service, s kratico SMS)



Slika 11: Shema vezave interaktivnih modulov

Interaktivni modul uporablja do sedaj opisano uporabniško programsko opremo ter jo dopolnjuje. Vsak interaktivni modul je povezan s PC-jem, ki sprejema ukaze interaktivnega modula, jih ustrezno obdela ter po potrebi posreduje mikrokrmilniku na grafični vodni zavesi (slika 11). Sledi opis razvoja prvega interaktivnega modula – spletnega vmesnika.

4.3.6 Interaktivni modul: Spletni vmesnik

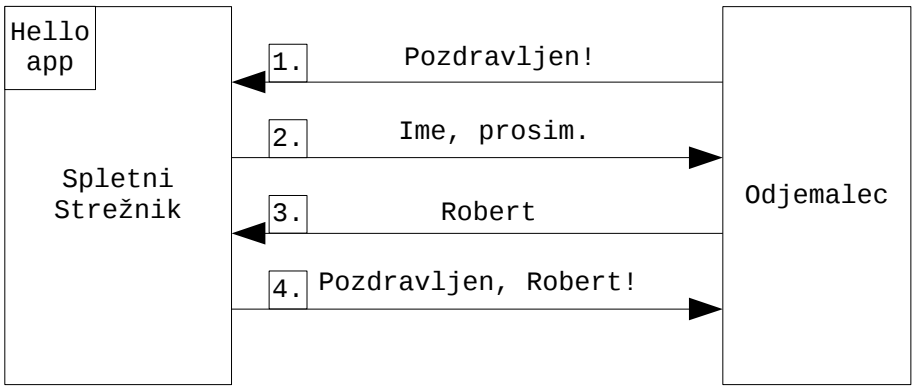
Interaktivni modul za spletni vmesnik (ali interaktivni spletni vmesnik) je podvrsta interaktivnega modula. Iz vidika spletnih tehnologij je to interaktivna spletna aplikacija. Njen namen je prostoročno risanje grafike na zaslone, občutljive na dotik, ter prenos grafike na grafično vodno zaveso. V nadaljevanju bomo na kratko opisali spletne tehnologije, ki so bile uporabljene pri razvoju interaktivnega spletnega vmesnika.

4.3.6.1 Spletne tehnologije

Spletne tehnologije so zelo široko področje, zato jih bomo le površinsko opisali do te mere, da bo zadostovalo za razumevanje nadaljnje snovi tega izdelka. Za poglobljeno poznavanje te tematike je priporočeno branje knjige »Web technologies: a computer science perspective«, avtorja Jackson, Jeffrey C [23].

Okolje spletnih aplikacij je v grobem sestavljeno iz spletnega strežnika ter odjemalca, kjer spletni strežnik nudi storitve, odjemalec pa jih zahteva in koristi. Spletni strežnik je programska oprema, ki poganja spletne aplikacije. Spletne aplikacije, ki so pisane v enem ali več jezikih (kot so npr. PHP, Java, ASP, itd.), tvorijo storitve odjemalcem. Dostop do storitev se vrši preko protokola, ki definira tok podajanja informacij med strežnikom ter odjemalcem. Za uspešno interpretacijo informacij, morata tako strežnik kot tudi odjemalec poznati format vsebine informacij.

Primer delovanja okolja spletnih aplikacij je prikazan na sliki 12. Spletni strežnik nudi eno spletno storitev s pomočjo spletne aplikacije »Hello app«. Naloga spletne aplikacije je pozdravni dialog z odjemalcem, ki teče po protokolu opisanem od 1. do 4. točke. Ravno tako je ključnega pomena prepoznavanje vsebine, če bi odjemalec v zgornjem hipotetičnem primeru poslal »Pozdravljen.« s piko na koncu, spletna aplikacija bodisi ne bi podala odgovora, bodisi bi zavrnila odjemalca z negativnim odgovorom (tudi način zavrnitev spada pod sam protokol o komunikaciji).



Slika 12: Primer okolja spletne aplikacije in dialoga z odjemalcem

Če postopoma preslikamo naš hipotetičen primer v interaktivni spletni vmesnik za grafično vodno zaveso, smo za spletni strežnik uporabili Apache programsko opremo s

podporo PHP skriptnemu jeziku [24]. Spletni strežnik v našem primeru poganja spletno aplikacijo, pisano v PHP skriptnem jeziku. Na drugi strani imamo odjemalec, ki je poljuben spletni brskalnik. Manjka še ključna povezava med obema in sicer protokol o komunikaciji, tega v našem predhodnem hipotetičnem primeru nismo imeli, zaradi preprostosti razlage primera.

Komunikacija med spletnim strežnikom ter brskalnikom poteka po protokolu za prenos hiperteksta (angl. HyperText Transport Protocol, s kratico HTTP). HTTP protokol uporablja zahtevek-odgovor model komunikacije, kar implicira začetno interakcijo s strani odjemalca, nato sledi ustrezen odgovor s strani spletnega strežnika. V glavi sporočila ima parametre, ki opisujejo parametre o odjemalcu ali spletnem strežniku, lahko tudi o samem odgovoru aplikacije. Zahtevek ali odgovor ima lahko tudi neobvezno telo, po katerem se prenaša večja vsebina odgovora, kot npr. tekst odgovora spletne aplikacije. Spletni strežnik s HTTP protokolom lahko uspešno komunicira z brskalnikom, vendar potrebujeta še načine za razumevanje vsebine, dobljene s HTTP zahtevki in odgovori.

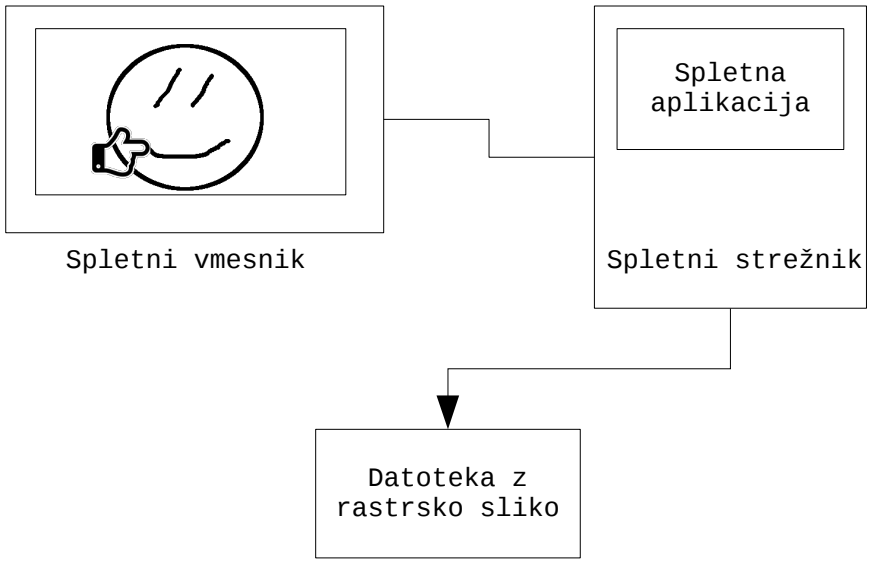
Za razumevanje same vsebine med spletnim strežnikom in brskalnikom imamo skupek standardov. Z njimi lahko spletni strežnik tvori vsebino ter jo s HTTP odgovori pošlje odjemalcu. Odjemalec jih tolmači in prikaže uporabniku, uporabnik pa z nadaljnjo interakcijo ustvarja nove HTTP zahtevke. Tako se dialog med strežnikom in odjemalcem ponavlja. V našem primeru so uporabljeni standardi za vsebino HTML, CSS ter Javascript [23].

Označevalni jezik za oblikovanje večpredstavnostnih dokumentov (angl. HyperText Markup Language, s kratico HTML) je označevalni jezik, uporaben za gradnjo spletnih vmesnikov. Elemente na spletnem vmesniku dodajamo z zapisom ustreznih značk, s tem dobimo osnovno podobo in funkcionalnost spletnega vmesnika. Za grafično upodobitev, uporabimo dodaten jezik prekrivnih slogov (angl. Cascading Style Sheets, s kratico CSS). CSS jezik definira grafično podobo posameznih elementov na spletni strani, s spreminjanjem parametrov kot npr. velikost, pozicijo, barvo, itd. Naš spletni vmesnik ni statično prikazana spletna vsebina po vsakem HTTP zahtevku, z njo želimo prestrezati pritiske na zaslon ter v eni barvi izrisati pike, glede na koordinate pritiska. Potrebujemo, da se del naše spletne aplikacije aktivno izvaja na odjemalčevi strani. V te namene se uporablja JavaScript skriptni jezik (včasih se uporablja tudi drugo ime ECMAScript, iz zgodovinskih razlogov). JavaScript je skriptni jezik, ki se izvaja v brskalniku. Z njim lahko programiramo upravljanje spletne strani v brskalniku, kot npr. tvorba novih asinhronih HTTP zahtevkov (zahtevki in odgovori se pošiljajo v ozadju naložene spletne strani, brez vpliva na izris spletne strani med tem časom), prestrezanje dogodkov vhodnih naprav, ter še mnogo drugih reči.

4.3.6.2 Zasnova interaktivnega spletnega vmesnika

Interaktivni spletni vmesnik je spletna aplikacija, katere realizacija je prikazana na sliki 13. Spletna aplikacija se izvaja na spletnem strežniku, njen spletni vmesnik pa prikazuje v

brskalniku. Namen interaktivnega spletnega vmesnika je prostoročno risanje na zaslon ter shranjevanje rastrske slike v datoteko na strežnik. Datoteka z rastrsko sliko je naprej podana v obdelavo prej opisani uporabniški programski opremi, pod poglavjem 4.3.4 o interakciji.



Slika 13: Interaktivni spletni vmesnik

Spletna aplikacija je pisana v PHP ter Javascript skriptnem jeziku. Uporablja Laravel php [25] ter JQuery javascript [26] programsko ogrodje, ki omogočata boljšo abstrakcijo in ponovno uporabo posameznih delov kode naše spletne aplikacije. Laravel programsko ogrodje, privzeto usmerja načrtovanje spletne aplikacije po Model-Pogled-Krmilnik (angl. Model–View–Controller, s kratico MVC) arhitekturnem principu.

MVC princip sestoji iz modela, pogleda, ter krmilnika [27]. Za lažjo razlago MVC principa, vzemimo za primer fizični avtomobil. Model predstavlja opis podatkov za naš objekt, v našem primeru npr. število prestav, maksimalno število vrtljajev, velikost posode za gorivo, itd. Pogled nam omogoča izris oziroma predstavitev podatkov o modelu, v našem primeru, bi lahko pogled o avtomobilu vseboval sliko samega avtomobila, glede na barvni parameter iz modela. Krmilnik veže model ter pogled skupaj z akcijami, ki jih kliče uporabnik. Za naš primer bi lahko v krmilniku popisali menjavo prestave, vklopa smernika, zasuk volana, itd. Za poglobljene informacije o Laravel programskem ogrodju ter MVC arhitekturnem principu, si lahko ogledate v knjigi avtorja Dayle Rees z naslovom »Laravel: Code Happy« [25].

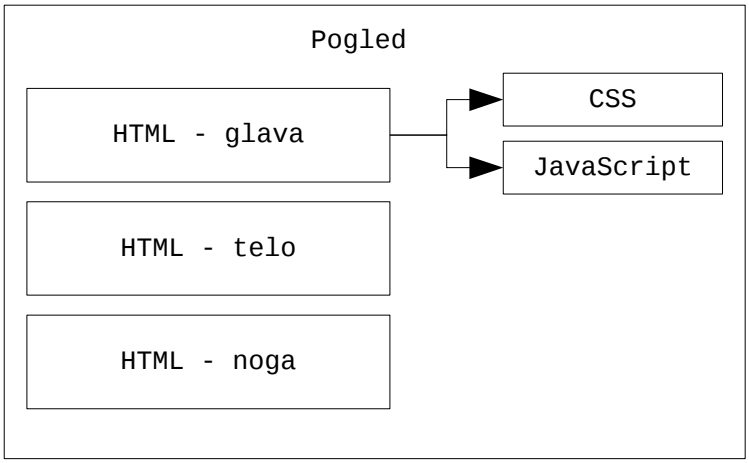
Naša spletna aplikacija vsebuje krmilnik, ki prestreza akcije s strani uporabnika ter privzeti pogled, ki je sestavljen iz HTML kode. Krmilnik je opisan s psevdokodo, kot je opisana v algoritmu 8. Ob pritisku na gumb za pošiljanje prostoročne slike na grafično vodno zaveso, brskalnik pošlje HTTP zahtevek s sliko v telesu, ki kliče metodo saveImage v krmilniku, za shranjevanje rastrske slike na disk. Shranjena slika je nadaljnje obdelana s

strani uporabniške programske opreme, kot je opisano v razdelku o interakciji.

```
1. set outputDir out

2. function index
3.  set view printerInterface
4.  send view to browser
5. endFunction

6. function saveImage(image)
7.  save image to outputDir
8. endFunction
```

Algoritem 8: Psevdokoda krmilnika

Privzeti pogled, ki je pisan v HTML jeziku, se deli na glavo, telo in nogo. Delitev omogoča hitro zamenjavo določenega dela, kar je praktično v našem primeru, kjer želimo v prihodnosti zamenjati CSS definicijo za grafično podobo, glede na odjemalca (ob predpostavki, da ima grafična vodna zavesa drugačen aranžma glede na dogodek). HTML koda vsebuje še reference na JavaScript kodo za prestrezanje koordinat miške na zaslonu, ter CSS kodo za grafično podobo vmesnika, kot je prikazano na sliki 14.



Slika 14: HTML pogled

JavaScript program je tudi zanimiv del spletne aplikacije, ker skrbi za prostoročno risanje glede na koordinate pritiska na zaslonu. JavaScript skriptni program se izvaja ob naložitvi celotnega HTML dokumenta, kot je opisano v algoritmu 9. Program je razdeljen na dva glavna dela in sicer glede na funkcijo onMouseDown ter draw. OnMouseDown prestreza dogodke vhodne naprave s kazalcem ter prebere koordinate kazalca na zaslonu. Metoda onMouseDown kliče metodo draw s koordinatami, ki izriše piko na podani koordinati. Uporabnik pušča enobarvno sled pik po zaslonu, bodisi z vlečenjem prsta po zaslonu na dotik, bodisi z vlečenjem miške, s pritisnjenim levim gumbom miške.

```
1.  on window load do
2.   function draw(x, y)
3.    draw dot on x, y
4.   endFunction

5.   function onMouseDown
6.    set x get X coordinate
7.    set y get Y coordinate
8.    draw(x, y)
9.   endFunction
10. done
```

Algoritem 9: Prestrezanje naprav s kazalcem

Interaktivni spletni vmesnik uporablja brskalnik za izris vsebine, zatorej je kompatibilen s širokim spektrom naprav – od pametnih mobilnih telefonov, do tabličnih računalnikov in PC-jev. Slika 15 prikazuje delujoč interaktivni spletni vmesnik, ki se nahaja na zaslonu občutljivem na dotik. Celotna spletna aplikacija teče na PC-ju.

Slika 15: Primer uporabe interaktivnega spletnega vmesnika

Na sliki 16 je prikazan primer interaktivnega spletnega vmesnika na tabličnem računalniku Apple iPad. V tem primeru strežniški del interaktivnega spletnega vmesnika deluje na oddaljenem PC-ju, tablični računalnik prikazuje le grafični vmesnik za odjemalca.

Slika 16: Primer interaktivnega spletnega vmesnika na tabličnem računalniku Apple iPad

Tekom projektne naloge smo razvili še interaktivni modul za prejem kratkih SMS sporočil, kateri deluje ločeno od interaktivnega spletnega vmesnika.

4.3.7 Interaktivni modul: Kratka SMS sporočila

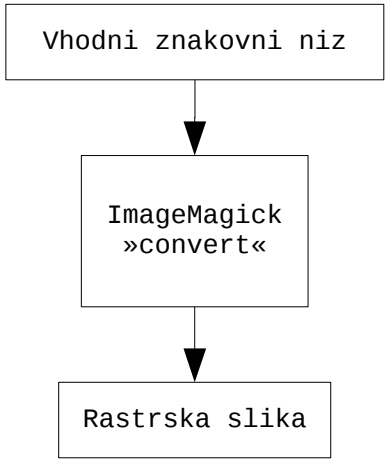
Tekom tega projekta je bil razvit še interaktivni modul za kratka SMS sporočila (ali SMS modul), ki na Android pametnemu telefonu prejema SMS sporočila, jih pošlje modulu za pretvorbo znakov v rastrsko sliko, ter poda uporabniški programski opremi in posledično izriše sliko na grafični vodni zavesi. Sam SMS modul ima tudi možnost prejemanja SMS sporočil neposredno od ponudnika storitev, konkretno je bila implementirana Telekomova storitev M:Vrata [30] tokom dogodka TS Startup 2013 [29]. V nadaljevanju bo opisano delovanje SMS modula s pomočjo aplikacije za Android pametne telefone.

4.3.7.1 Tehnologije

SMS modul, tako kot prejšnji modul interaktivnega spletnega vmesnika, uporablja različne tehnologije za doseganje namena. Poleg spletnih tehnologij, ki pokrivajo HTTP komunikacijo, uporablja še ImageMagick programsko opremo.

ImageMagick je programska oprema za obdelavo rastrskih slik in omogoča ustvarjanje, urejanje ali pretvorbo med različnimi rastrskimi podatkovnimi zapisi [28]. Podpira večje število podatkovnih zapisov, poglavitna prednost ImageMagick programske opreme je izvajanje naštetih operacij brez grafičnega vmesnika. Je tudi priljubljena in razširjena programska rešitev, najdemo jo na mnogo strežnikih s podporo gostovanja PHP spletnim aplikacijam. ImageMagick smo uporabili za pretvorbo znakovnih nizov v rastrsko sliko, kot je prikazano na sliki 17. Za pretvorbo znakovnega niza v rastrsko sliko smo izvedli s pomočjo ImageMagick orodja »convert« (sl. Pretvoriti, preoblikovati). Kljub

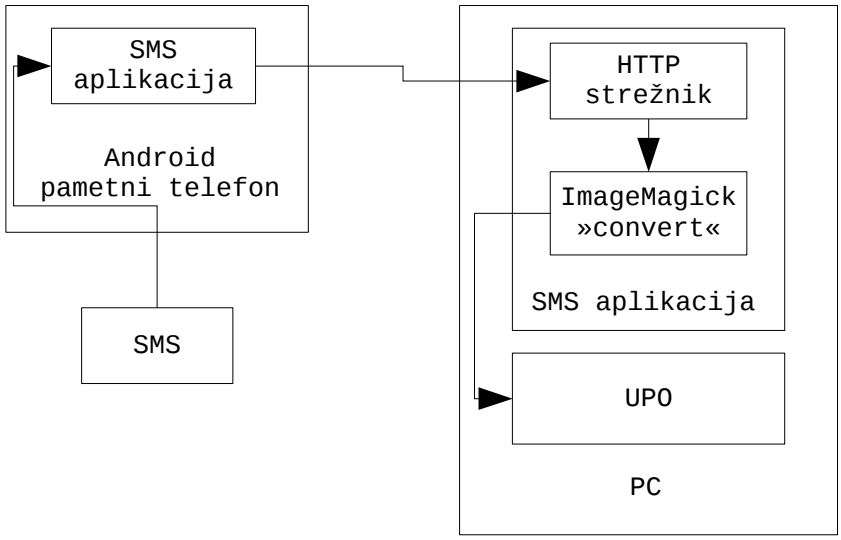
imenu »convert«, ima orodje precej širši nabor zmožnosti kot samo pretvorbo podatkovnega zapisa, kar ime neposredno nakazuje. S pomočjo parametrov k orodju »convert«, lahko izvajamo različne operacije nad vhodnim znakovnim nizom. Za naše namene smo uporabili več parametrov. Prvi parameter je bil »background« z vrednostjo »white«, ki rastrski sliki priredi belo ozadje. Parameter »fill« smo uporabili skupaj z vrednostjo »black«, kar sproži izris vseh podanih grafičnih objektov v črni barvi. S »pointsize« parametrom smo določili velikost pisave (le ta je pa odvisna od postavitve grafične vodne zavese), parameter »size« pa vpliva na fizično velikost slike. Fizična velikost slike je podana glede na komponento širine in višine. Pri grafični vodni zavesi vnaprej poznamo samo širino, ki je podana s številom ventilov na grafični vodni zavesi. Z vrednostjo npr. »64x« ob parametru »size« tako ImageMagick določi širino slike 64 točk, višino pa ImageMagick dinamično prilagodi glede na vsebino. Parameter »interline-spacing« določa razmak v številu pik med vrsticami, »gravity« z vrednostjo »Center« pa poskrbi za sredinsko poravnavo besedila. Zadnji pomemben parameter je »caption«, ki podan znakovni niz pretvori v grafični objekt in izriše rastrsko sliko z besedilom. Sedaj moramo ImageMagick povezati v verigo programske opreme, za tvorbo rastrskih slik na podlagi prejetega znakovnega niza, da bomo lahko prikazovali SMS sporočila.



Slika 17: Uporaba ImageMagick "convert" orodja

4.3.7.2 Zasnova SMS modula

Naloga SMS modula je prejem kratkih SMS sporočil ter pretvorba v rastrsko sliko. Rastrsko sliko, kot v prejšnjem primeru interaktivnega spletnega modula, podamo naprej v obdelavo uporabniški programski opremi.



Slika 18: Shema delovanja SMS modula

SMS modul je razdeljen na več delov. Prvi del aplikacije se izvaja na PC-ju in skrbi za sprejem znakovnega niza ter tvorbo rastrske slike. Drugi del aplikacije je Android aplikacija, ki skrbi za prestrezanje kratkih SMS sporočil ter posredovanje prvemu delu aplikacije. Aplikacija je v celoti pisana v programskem jeziku Java. Slika 18 prikazuje tok delovanja SMS modula. Kratko SMS sporočilo prejme Android pametni telefon, na njem ga prestreže naša SMS aplikacija, ki ga posreduje preko HTTP protokola našemu spletnemu strežniku. Spletni strežnik prejet HTTP zahtevek obdela, izlušči kratko SMS sporočilo ter požene ImageMagick z ustreznimi parametri, za tvorbo rastrske slike z izluščenim besedilom. Rastrska slika je v naslednjem koraku obdelana s strani uporabniške programske opreme.

```
1. set convertParameters default

2. function httpRequest(POST)
3.  set message POST.body
4.  createImage(message)
5. endFunction

7. function createImage(message)
8.  create process 'convert' with convertParamteters
9. endFunction
```

Algoritem 10: Luščenje SMS sporočila ter poganjanje ImageMagick orodja

Primer delovanja SMS modula na PC-ju je opisan v algoritmu 10. Algoritem s funkcijo httpRequest prestreže HTTP zahtevek ter izlušči telo pripetega SMS sporočila. Nato se izvede funkcija createImage s SMS sporočilom, ki požene program »convert« s parametri za ustvarjanje rastrske slike. Rastrska slika je pripravljena za nadaljnjo obdelavo

s strani uporabniške programske opreme.

Tekom dogodka TS:Startup 2013, je bil razvit še pod-modul za prejemanje SMS sporočil neposredno s strani operaterja, konkretno Telekom Slovenije. Premijska storitev M:Vrata omogoča več pod-storitev za komunikacijo z uporabniki. V našem primeru smo implementirali prejem kratkih SMS sporočil [31]. Storitev deluje tako, da pošlje HTTP zahtevek na ciljni strežnik s SMS sporočilom. Poglavitna razlika iz tehničnega vidika je ta, da ne potrebujemo fizičnega telefona. Zaradi sorodnega principa delovanja, obstoječa zasnova SMS modula s HTTP strežnikom ni potrebovala večjih sprememb.

5 ZAKLJUČEK

Projekt programske opreme za grafično vodno zaveso je bil zanimiv zaradi uporabe različnih tehnologij za doseganje namena. Začet je bil pri programski opremi za elektroniko grafične zavese, kjer se je reševalo izzive komunikacije z elektroniko, učinkovitim podatkovnim zapisom, shranjevanjem podatkovnega zapisa, ter komunikacijo prek zaporednega vmesnika. Nadaljnje se je pod projektom razvilo uporabniško programsko opremo, ki skrbi za pretvorbo rastrskih slik v zapis grafične vodne zavese, ter pošiljanja zapisov na mikrokrmilnik. Dodatno se je vpeljalo interaktivne module, ki množici gledalcev omogočajo rokovanje z vsebino in izrisom v realnem času. Za te namene sta bila konkretno razvita dva interaktivna modula, prvi za interakcijo prek naprav z zaslonom občutljivim na dotik (interaktivni spletni vmesnik), drugi za interakcijo preko SMS sporočil (interaktivni SMS modul dodatno). Interaktivni spletni vmesnik združuje več tehnologij pod en modul za izris prostoročno skicirane grafike na grafično vodno zaveso. Javascript program z JQuery programskim ogrodjem skrbi za interakcijo vhodnih naprav s spletnim vmesnikom v brskalniku, PHP aplikacija z Laravel programskim ogrodjem pa skrbi za pravilno izvajanje na strežniškem delu. Spletna aplikacija kot izhod tvori rastrsko sliko, ki je pripravljena na nadaljnjo obdelavo s strani uporabniške programske opreme. Interaktivni SMS modul dodatno razširja zmožnost grafične vodne zavese. Kljub drugačnemu viru izvornih podatkov se z modularnim načrtom lahko enostavno vklopi v uporabniško programsko opremo grafične vodne zavese. S pod-moduli implementira še dva načina prestrezanja kratkih SMS sporočil, bodisi prek Android pametnega telefona, bodisi neposredno s strani operaterja.

V prihodnosti se bomo osredotočili na izdelavo novih interaktivnih modulov, saj vzbujajo največ interesa pri opazovalcih, ki lahko aktivno spreminjajo izpis grafične vodne zavese. Dodatne tehnološke izzive predstavlja ideja o večbarvnem izpisu grafičnih objektov, bodisi enolično barvo glede na celoten grafični objekt, bodisi različne barve na en vodni curek. Enolično določanje barve glede na celoten grafični objekt, bi lahko enostavno rešili z enim opisnim parametrom na začetku podatkovnega zapisa grafične vodne zavese, tako bi mikrokrmilnik ustrezno vklopil primerno barvno razsvetljavo. Za določanje barve glede na posamezen curek, bi pa potrebovali sistem natančno vodenih barvnih žarkov do curka, le ta bi pa moral tudi slediti sami gravitaciji padanja curka, ter ga natančno spremljati. Spremembe na programski opremi bi bile nedvomno večje v drugem primeru, potrebovali bi tudi podatkovni zapis za barvno osvetlitev posameznih padajočih curkov.

Z zaključno nalogo smo poglobili znanje na več področjih, od mikrokrmilnikov, do programiranja le teh, spletnih tehnologij, komunikacijskih vmesnikov, poznavanja različnih programskih ogrodjih ter delovanje določenih elektronskih vezij. Projekt se nam je skozi celoten razvoj zdel zanimiv, zaradi bogatega nabora različnih izzivov in pristopov

reševanja le teh. V prihodnosti upamo na večji razvoj modulov za grafično vodno zaveso, kateri precej vzbudijo zanimanje pri končnemu gledalcu.

6 LITERATURA

[1] Jani Černe, Vodni slap kot tiskalnik (Waterfall as a printer), 19.9.2011, obiskano 3.6.2014 URL http://eprints.fri.uni-lj.si/1504/

[2] All About Circuits, Shift registers – introduction, obiskano 6.7.2014 URL http://www.allaboutcircuits.com/vol_4/chpt_12/1.html

[3] All About Circuits, Shift registers – Serial-in, parallel-out shift register, obiskano 6.7.2014 URL http://www.allaboutcircuits.com/vol_4/chpt_12/4.html

[4] Steve Heath, Embeded Systems Design, second edition. Oxford, Boston: Newnes, 2003.

[5] Atmel, ATMEGA 328, obiskano 8.7.2014 URL http://www.atmel.com/devices/atmega328.aspx

[6] Freescale, K20 Sub-Family Reference Manual, revision 2, February 2012, URL https://www.pjrc.com/teensy/K20P64M50SF0RM.pdf

[7] Sparkfun, Serial communication, obiskano 8.7.2014 URL https://learn.sparkfun.com/tutorials/serial-communication/all

[8] Dallas Semiconductor, Application Note 83, Fundamentals of RS–232 Serial Communications, obiskano 8.7.2014 URL http://www.lammertbies.nl/download/dallas-appl-83.pdf

[9] Binglong's space, Arduino Serial Port Communication, obiskano 8.7.2014 URL http://binglongx.wordpress.com/2011/10/26/arduino-serial-port-communication/

[10] Oracle, What is Java technology and why do I need it?, obiskano 26.8.2014 URL http://www.java.com/en/download/faq/whatis_java.xml

[11] Eclipse Foundation, Eclipse IDE, obiskano 15.7.2014 URL https://www.eclipse.org/

[12] Technopedia, Integrated Development Environment (IDE), obiskano 15.7.2014 URL http://www.techopedia.com/definition/26860/integrated-development-environment-ide

[13] Mfizz, RXTX for java, RXTX fork, obiskano 15.7.2014 URL http://mfizz.com/oss/rxtx-for-java

[14] Arduino, Arduino IDE, obiskano 15.7.2014 URL http://arduino.cc/en/main/software

[15] PJRC, Teensyduino – addon for Arduino IDE, obiskano 15.7.2014 URL https://www.pjrc.com/teensy/teensyduino.html

[16] SD Association, SD standards, obiskano 16.7.2014 URL https://www.sdcard.org/developers/overview/

[17] Andrew S Tanebaum, Modern operating systems 2nd edition. Prentice Hall PTR. Poglavje 5, stran 270, I/O devices

[18] Andrew S Tanebaum, Modern operating systems 2nd edition. Prentice Hall PTR. Poglavje 6, stran 379, File systems

[19] Google Code, A FAT16/FAT32 Arduino library for SD/SDHC cards, 2014 URL https://code.google.com/p/sdfatlib/

[20] Andrew S Tanebaum, Modern operating systems 2nd edition. Prentice Hall PTR. Poglavje 6.4.3, stran 440, The MS-Dos File System (FAT)
[21] Princeton, Raster graphics, obiskano 29.7.2014 URL http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Raster_graphics.html
[22] Princeton, RGB Color model, obiskano 29.7.2014 URL http://www.princeton.edu/~achaney/tmve/wiki100k/docs/RGB_color_model.html
[23] Jackson, Jeffrey C., Web technologies: a computer science perspective. Upper Saddle River: Pearson/Prentice Hall, cop. 2007
[24] The Apache HTTP Server Project, About the Apache HTTP Server Project, obiskano 27.8.2014 URL http://httpd.apache.org/ABOUT_APACHE.html
[25] Dayle Rees, Laravel: Code Happy, Application development with the Laravel PHP Framework for beginners. Leanpub, 7.7.2012.
[26] jQuery, 2.8.2014 URL http://jquery.com/
[27] Jackson, Jeffrey C., Web technologies: a computer science perspective. Upper Saddle River: Pearson/Prentice Hall, cop. 2007. Stran 466, poglavje 8.7.1, MVC Basics
[28] Imagemagick, obiskano 3.8.2014 URL http://www.imagemagick.org/
[29] TS startup, obiskano 3.8.2014 URL http://www.tsstartup.si/ekipe
[30] M:Vrata, obiskano 3.8.2013 URL http://www.telekom.si/podjetniki/poslovne-resitve/premijske-storitve/m-vrata/predstavitev
[31] Telekom Slovenija, Komunikacijski vmesnik med Telekomom Slovenije in zunanjimi ponudniki (prejem vsebin SMS/MMS s strani uporabnika), različica 2, obiskano 3.8.2014 URL http://www.telekom.si/Documents/Sporocilo_na_zahtevo.pdf