UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Luka Hauptman

Elektronski sistem letalskih instrumentov

DIPLOMSKO DELO UNIVERZITETNEGA ŠTUDIJA

Mentor:

prof. dr. Dušan Kodek

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01538/2009
Datum: 15.01.2009

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

Kandidat: **LUKA HAUPTMAN**

Naslov: **ELEKTRONSKI SISTEM LETALSKIH INSTRUMENTOV**
ELECTRONIC FLIGHT INSTRUMENT SYSTEM

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Analizirajte lastnosti obstoječih sistemov letalskih instrumentov in z njimi povezanih mednarodnih predpisov. Na osnovi ugotovljenih lastnosti izberite rešitev sistema za prikazovanje podatkov na letalih z batnimi motorji. Za to rešitev izberite primerne komponente in razvijte računalniški sistem, s katerim je preko senzorjev in prikazovalnika mogoče prikazati vse zahtevane podatke. Razvijte in izdelajte tudi ustrezno programsko opremo s primernim uporabniškim vmesnikom. Sistem preizkusite in preverite pravilnost delovanja.

Mentor:
prof. dr. Dušan Kodek

Fakulteta za računalništvo in informatiko
UNIVERZA V LJUBLJANI
2

Dekan:
prof. dr. Franc Solina

Zahvala

Zahvaljujem se mentorju prof. dr. Dušanu Kodeku za strokovne nasvete, pomoč pri reševanju razvojnih težav in obilico časa, ki mi ga je namenil. Iskrena hvala očetu za finančno podporo in zaupanje v moje sposobnosti. Hvala tudi mami in bratu za podporo v času študija, stricu Pavlu Hauptmanu za pomoč pri izdelavi ohišja ter teti Mateji Piškur za lektoriranje.

Kazalo

Seznam uporabljenih kratic

CCFL – Cold Cathode Fluorescence Lamp

CHT – Cylinder Head Temperature

CNC – Computer Numerical Controlled

ECAM – Electronic Centralised Aircraft Monitoring

EFIS – Electronic Flight Instrument System

EGT – Exhaust Gas Temperature

EICAS – Engine Indications and Crew Alerting System

EL – Electroluminiscent

FAA – Federal Aviation Administration

GA – General Aviation

JAA – Joint Aviation Authorities

JAR – Joint Aviation Requirements

LVDS – Low Voltage Differential Signaling

MFD – Multi-Function Display

ND – Navigation Display

OSD – On-Screen Display

PFD – Primary Flight Display

PSI – Pound per Square Inch

QVGA – Quarter Video Graphics Array

RPM – Revolutions per minute

SD – Secure Digital

SXGA – Super eXtended Graphics Array

TFT – Thin-Film Transistor

UXGA – Ultra eXtended Graphics Array

VMC – Visual Meteorological Conditions

Povzetek

Pričujoče diplomsko delo opisuje temeljne pristope, ki se uporabljajo pri razvoju in izvedbi enostavnega elektronskega sistema letalskih instrumentov, namenjenega prikazovanju podatkov batnega letalskega motorja. Osnovni namen je izdelava delujočega prototipa ter pridobitev znanj, ki bodo služila nadaljnemu razvoju celotnega sistema.

Drugo poglavje predstavi bralcu osnovne koncepte elektronskega sistema letalskih instrumentov, ki ga najdemo na večjih potniških letalih. Podrobneje sta opisana različna pristopa izvedbe sistema dveh največjih proizvajalcev letal, Airbusa ter Boeinga.

V tretjem poglavju je opisana strojna oprema izdelane naprave ter grafični vmesnik. Uporabljena vezja je moč kupiti v prosti prodaji, ohišje pa je bilo izdelano po meri. Ideje pri oblikovanju grafičnega vmesnika so nastale na podlagi rešitev proizvajalcev podobnih sistemov.

Opis programske opreme naprave se nahaja v četrtem poglavju. Poleg razlage pomembnejših funkcij, so predstavljeni tudi posamezni vidiki ustreznosti izvedbe programske kode. Poglavje je namenjeno predvsem tistim, ki želijo uporabiti oz. spreminjati izvorno kodo.

V zaključku je povzeto celotno delo, vključno s predstavitvijo idej za nadaljni razvoj. Predvidena prva faza razvoja obsega dodelavo ter izboljšavo prototipa. Glavni cilj je izdelati sistem, s katerim bo moč uspešno nastopati na trgu elektronskih sistemov letalskih instrumentov.

Ključne besede: elektronski sistem letalskih instrumentov, prikazovanje podatkov batnega letalskega motorja, postopki v sili, grafični vmesnik, izdelava prototipa

Abstract

This thesis describes basic concepts in research and development of a simple electronic flight instrument system, which displays piston engine data to the pilot. The main purpose is to build a functional prototype and acquire knowledge, which will enable us to further develop the system.

The second chapter presents fundamentals of electronic flight instrument systems used in large commercial aircrafts. A detailed description of basic approaches to system implementation used by two of the biggest airplane manufacturers, Airbus and Boeing is given.

Chapter three describes hardware components of the prototype and graphical interface. System housing is the only component that was custom built, while all the integrated circuits were available on the market. The ideas for designing the graphical interface were based on similar products made by different manufactures.

Description of software can be found in chapter four. In addition to the explanation of the main functions, the chapter also describes techniques used to implement certain parts of the software. Users who intend to use or modify the source code will find all the necessary information in this chapter.

The last chapter summarizes all of the findings together with recommendations for further development. This will include completion and improvement of the prototype. The main goal is to develop a system which would enable us to successfully meet the global market demands.

Keywords: electronic flight instrument system, aircraft piston engine data display, emergency procedures, graphical interface, prototype design

1 Uvod

V zadnjih dveh desetletjih je na področju letalskih instrumentov prišlo do revolucionarnih sprememb. Klasične elektromehanske instrumente nadomeščajo elektronski sistemi, ki temeljijo na uporabi mikroprocesorjev in zaslonov. Slednji so bili na začetku razvoja katodni, novejši sistemi pa temeljijo na zaslonih s tekočimi kristali. Ravno LCD-zasloni so pripomogli k množičnejši uporabi elektronskih sistemov v manjših letalih. V primerjavi s katodnimi prikazovalniki so lažji, manjši ter imajo manjšo porabo električne energije. Pri manjših letalih sta velikost in masa elektronskega sistema instrumentov še toliko bolj pomembni, saj imamo pogostokrat opravka z utesnjenimi prostori in največjo dovoljeno skupno maso letala.

Elektronski sistemi letalskih instrumentov [1, 2] (angl. »Electronic Flight Instrument System«, krajše EFIS) so bili sprva omejeni na uporabo v potniških letalih, kjer so bistveno pripomogli k povečanju varnosti v letalskem prometu. V zadnjih letih pa lahko najdemo t. i. »glass cockpit« tudi v letalih splošnega letalstva (angl. »General Aviation«, krajše GA). Omeniti velja, da morajo takšni sistemi, torej sistemi, ki jih uporabljajo v prej omenjenih segmentih letalstva, ustrezati strogim predpisom evropske organizacije Joint Aviation Authorities (krajše JAA) oz. njene ameriške »različice« Federal Aviation Administration (krajše FAA).

Pridobitev ustreznega certifikata je običajno zahtevna, zato se je na trgu v zadnjem desetletju pojavilo več podjetij, ki izdelujejo elektronske sisteme za t. i. eksperimentalna samostojno izdelana in ultra lahka letala. Takšni sistemi so običajno podrejeni manj strogim zahtevam, hkrati pa je njihova uporaba omejena na letenje v vizualnih meteoroloških pogojih (angl. »Visual Meteorological Conditions«, krajše VMC).

EFIS običajno sestavljajo trije podsistemi, in sicer primarni letalni zaslon (angl. »Primary Flight Display«, krajše PFD), večfunkcijski zaslon (angl. »Multi-Function Display«, krajše MFD), ki ga nekateri proizvajalci poimenujejo tudi navigacijski zaslon (angl. »Navigation Display«, krajše ND), ter sistem za nadzor delovanja motorjev in obveščanja posadke (angl. »Engine Indications and Crew Alerting System«, krajše EICAS). Airbusova različica slednjega se označuje s kratico ECAM (angl. »Electronic Centralised Aircraft Monitoring«). EFIS se v predhodno omenjeni obliki običajno nahaja v potniških letalih, medtem ko je pri manjših letalih običajno združevanje dveh sistemov v enega. Izpostaviti velja, da je v nekateri literaturi moč zaslediti definicijo elektronskega sistema letalskih instrumentov kot skupek

primarnega in večfunkcijskega zaslona oz. sistema. V pričujočem delu se uporablja prvotno omenjena razlaga.

Letalstvo me je že od nekdaj navduševalo, lahko bi se celo reklo, da mi je bilo usojeno ob rojstvu. Glavni »krivec« je moj oče, kapetan letala A330/A340, ki sem ga že v otroških letih spremljal na njegovih poletih. Z odraščanjem mi je postajalo letalstvo čedalje bolj zanimivo in razumljivo, po končani srednji šoli pa sem tudi sam pričel leteti. Z diplomskim delom sem želel povezati obe področji mojega zanimanja, torej računalništvo in letenje. Menim, da razširjenost računalniške znanosti na nekaterih področjih letalstva (tu mislim predvsem na manjša športna letala) ni tako izrazita kot drugje. Posledično so zaradi manjše konkurence možnosti za uspešnost bistveno večje.

Osnovni cilj diplomske naloge je bil izdelati sistem za prikazovanje podatkov batnega letalskega motorja kot poenostavljene različice sistema ECAM. Kot zgled je bil uporabljen sistem EMS-D10 podjetja Dynon Avionics [3]. Glavni zahtevi sta bili čim nižji stroški strojnih komponent in kratek razvojni čas. Običajno pa so komponente, ki zahtevajo manj razvojnega časa, dražje. V času nastanka pričujočega dela je na svetovnem trgu nastopalo približno deset podjetij, ki se ukvarjajo z razvojem sistemov EFIS. Cene proizvodov se gibljejo od 1200 pa vse do 12000 €.

Glavni lastnosti končnega izdelka sta preprost uporabniški vmesnik, ki omogoča takojšnjo identifikacijo problema in prikaz ustreznega postopka v sili. Med razvojem sta do izraza prišla dva ključna problema. Prvi je bil (pre)majhna velikost flash pomnilnika mikrokrmilnika ATmega168 in nezmožnost priključitve dodatnega zunanjega pomnilnika, posledica česar je bila omejitev izvedbe vseh želenih funkcionalnosti. Zmogljivost grafičnega krmilnika Picaso MD-1 pa je težave povzročala pri izrisovanju obratov motorja. Hitrost spreminjanja slednjih je dokaj visoka v primerjavi s spremembami temperature in tlaka.

2 Elektronski sistemi letalskih instrumentov

2.1 Zgodovina

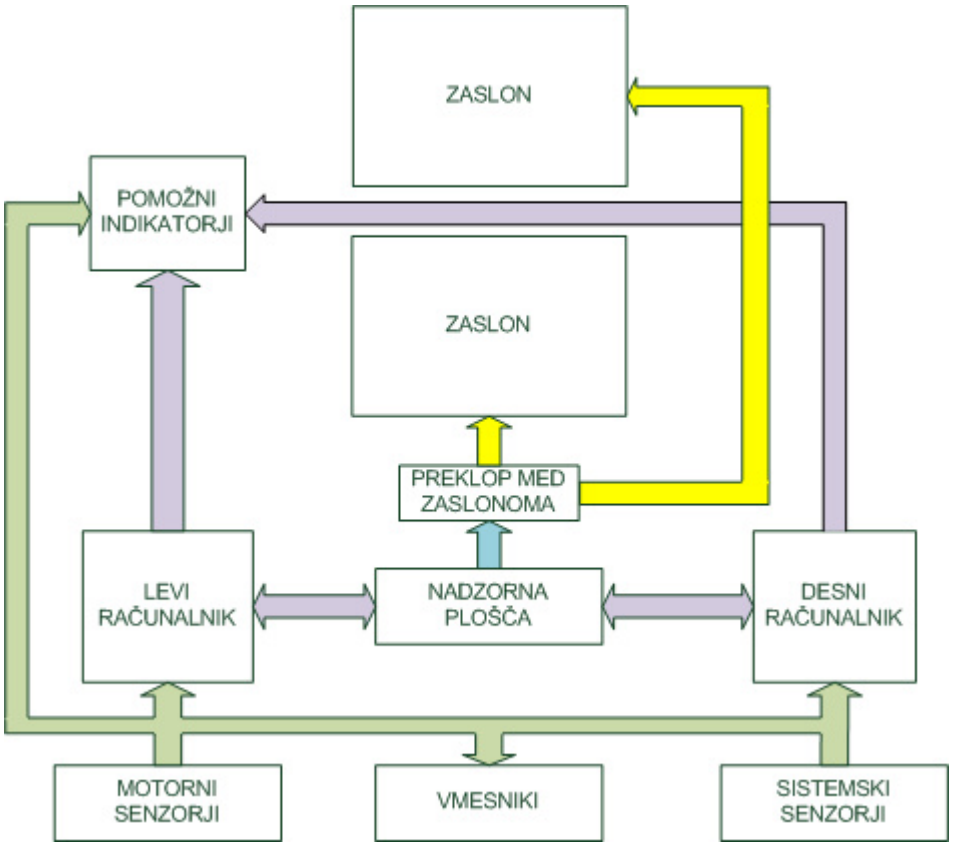
V tem poglavju bomo predstavili elektronski sistem letalskih instrumentov, ki ga najdemo v velikih letalih. Takšni sistemi so kompleksni, saj je glavni poudarek na varnosti, kar pomeni, da se vsi podatki sproti preverjajo, preden se predstavijo človeku. Poleg tega imamo opravka s podvojenimi (tudi potrojenimi) varnostno kritičnimi deli sistema. Pri manjših letalih, kjer imamo opravka z bistveno manjšo stopnjo kompleksnosti, bi bilo tolikšno izpostavljanje varnosti neracionalno.

Glavna naloga vseh elektronskih sistemov letalskih instrumentov, ne glede na vrsto letal, v katerih se uporabljajo, je prikazovanje v določenem trenutku kritičnih podatkov. S tem se zmanjša delovna obremenitev in poveča zmožnost osredotočenja na za varnost bistvene podatke.

Za nadzorovanje delovanja motorjev in drugih sistemov (hidravlični, električni, pnevmatski ...) sta v uporabi sistema EICAS in ECAM. V času uvedbe obeh je bilo razlikovanje med njima smiselno, medtem ko so razlike z razvojem počasi izginjale. Tako lahko danes rečemo, da imamo opravka z eno samo sistemsko arhitekturo, ki jo proizvajalci različno implementirajo. Sistem EICAS, ki je bil prvič predstavljen na letalih Boeing 757 in 767, sestavljata dva katodna zaslona, ki prikazujeta vse podatke o delovanju motorjev in drugih sistemih letala, kar je omogočilo izločitev ustreznih klasičnih elektromehanskih instrumentov. V istem obdobju je tudi evropski konzorcij Airbus predstavil sistem ECAM (na letalu A310). Airbusova rešitev je pomenila kombinacijo elektromehanskih in elektronskih instrumentov. Tako so bili podatki o motorjih prikazani na klasičnih analognih instrumentih, prikaz ostalih sistemov letala pa je bil v elektronski obliki.

2.2 EICAS

Sestavljata ga dva katodna ali LCD-prikazovalnika, nadzorna plošča (angl. »control panel«) in dva računalniška sistema z analognimi in digitalnimi vhodi. Slika 1 prikazuje sistem EICAS.



Slika 1: EICAS

Med delovanjem je le eden od računalnikov (levi ali desni) aktiven, medtem ko je drugi v stanju pripravljenosti. V primeru okvare aktivnega se pomožni aktivira samodejno ali pa ga izberemo ročno. EICAS ves čas delovanja zagotavlja pilotoma primarne parametre delovanja motorjev, sekundarni parametri ter opozorila, previdnosti in druga obvestila pa se izpišejo po potrebi.

Zaslona sta običajno razporejena eden nad drugim, pri čemer vrhnji prikazuje primarne parametre, opozorila in previdnosti, spodnji pa sekundarne (obrati drugostopenjskega kompresorja, pretok goriva, količina olja, tlake in temperature) ter podatke o ostalih sistemih. Vrsta podatka je natančno določena tudi z barvno shemo (podrobnejša razlaga pomena posameznih barv se nahaja v poglavju 3.4).

Nadzorna plošča med drugim omogoča spreminjanje podatkov, prikazanih na spodnjem zaslonu. Čeprav se vsaka napaka v sistemu samodejno shrani v pomnilnik, je mogoče preko nadzorne plošče ročno shraniti vprašljiv dogodek, ki ga lahko vzdrževalci letala kasneje pregledajo. Poleg omenjenih funkcij se na nadzorni plošči nahajajo še stikala za nastavljanje svetlosti zaslonov, brisanje opozoril itd.

Sistem EICAS zagotavlja neprekinjeno spremljanje okoli 400 vrednosti, ki jih dobi preko različnih senzorjev. V primeru okvare katerega izmed nadzorovanih podsistemov izpiše ustrezno opozorilo, pri čemer je to razporejeno v enega izmed treh nivojev. Najvišjo prioriteto (nivo A) imajo opozorila, ki zahtevajo takojšnji korektivni ukrep in so obarvana rdeče. Spremljajo jih tudi kontinuirana zvočna opozorila. Nižjo prioriteto (nivo B) imajo previdnosti (angl. »cautions«), ki zahtevajo takojšnjo pozornost in morebitno akcijo. So jantarjeve barve, zvočni signal pa se ponovi dvakrat. Na najnižjem nivoju so dogodki, ki zahtevajo samo pozornost posadke, so enake barve kot previdnosti in brez zvočnih opozoril.

2.3 ECAM

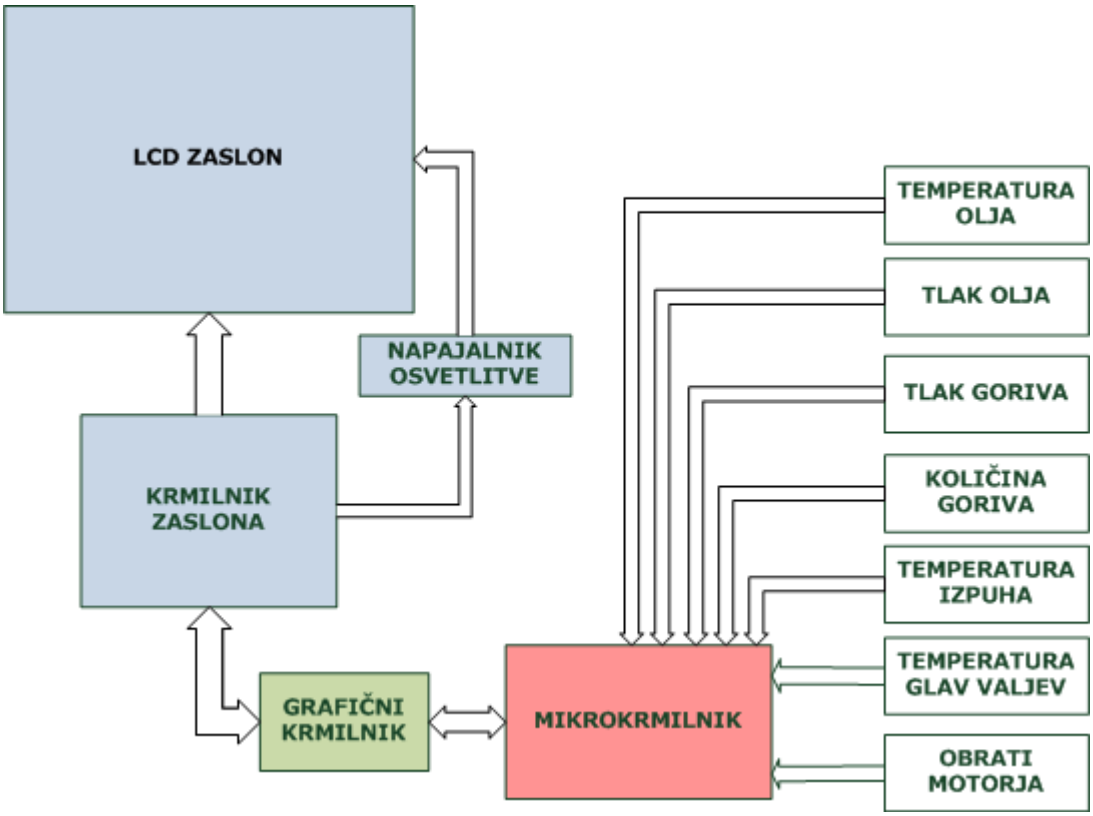
Čeprav ima sistem ECAM v takšni obliki, kot jo bomo opisali, zgolj še zgodovinski pomen, je prav, da prikažem tudi drug pristop k zasnovi sodobnih letalskih instrumentov. Sestavni deli ECAM-sistema so enaki kot pri EICAS-u. Katodna zaslona sta postavljena drug zraven drugega, pri čemer levi prikazuje sistemske informacije, opozorila ter korektivne ukrepe v obliki kontrolnih seznamov (angl. »checklist«). Desni zaslon je namenjen grafičnemu oz. shematičnemu prikazu informacij, ki se nahajajo na levem zaslonu.

Predvsem velja izpostaviti delovanje ECAM-a v primeru okvare katerega izmed sistemov letala. Na levem zaslonu se izpiše opozorilo in seznam ustreznih korektivnih ukrepov. Hkrati se na desnem zaslonu prikaže shema sistema, v katerem je prišlo do okvare. Vsako opozorilo spremlja tudi zvočni signal, ki opozarja na okvaro. Po izvedbi posameznega ukrepa se ustrezno navodilo na levem prikazovalniku nadomesti s sporočilom o uspešni izvedbi. Sprememba se odrazi tudi na shemi, ki je prikazana na desnem zaslonu.

3 Naprava

Osrednji del naprave je razvojna ploščica Arduino Diecimila [4]. Ta je preko serijske povezave povezana z grafičnim krmilnikom Picaso MD-1 [5]. Komunikacija poteka v načinu gospodar – suženj, saj grafični krmilnik samo izvršuje ukaze, ki jih izdaja mikrokrmilnik. Po vsakem uspešno izvedenem ukazu sledi pošiljanje potrditve. V kolikor pride do napake pri izvrševanju ukaza, je odgovor negativna potrditev (angl. »Negative Acknowledge«, krajše NAK). Hitrost serijske povezave je 250000 bitov na sekundo.

Tipala so priključena na ploščico, bodisi preko A/D-pretvornika bodisi neposredno preko ene izmed vhodno/izhodnih digitalnih nožic. VGA-povezava poteka med grafičnim krmilnikom in krmilnikom zaslona. Blok shema celotne naprave je prikazana na sliki 2.Trenutna različica naprave ne podpira interakcije uporabnika s sistemom. Vsa območja posameznih instrumentov so fiksno določena in jih je moč spremeniti s popravkom ustreznih konstant v programski kodi. Slednja je napisana tako, da omogoča kar najbolj enostavno spreminjanje posameznih parametrov sistema.



Slika 2: Blok shema sistema za prikazovanje podatkov batnega letalskega motorja

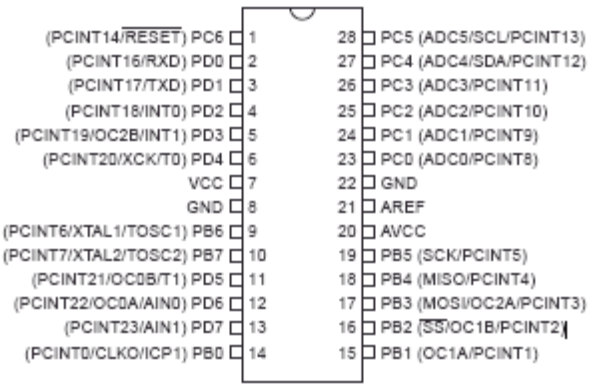
3.1 Arduino Diecimila

Je razvojna ploščica, ki temelji na 8-bitnem mikrokrmilniku ATmega168 proizvajalca Atmel. Slednji deluje pri frekvenci ure 16 MHz in ima 16 KB flash, 512 B EEPROM in 1 KB statičnega bralno-pisalnega pomnilnika. Mikrokrmilnik temelji na izpopolnjeni RISC arhitekturi, ki omogoča izvedbo večine ukazov v eni urini periodi.

Napajanje ploščice je zagotovljeno preko USB-vmesnika ali preko zunanjega enosmernega napetostnega vira, ki ga izberemo z ustrezno postavitvijo mostička PWR_SEL. Priporočena napetost je med 7 in 12 V. Omogočeno je tudi enostavno baterijsko napajanje s povezavo polov baterije z nožicama Vin in Gnd.

Ploščico lahko programiramo z uporabo Arduino razvojnega okolja. Napisano je v programskem jeziku Java in temelji na odprtokodnem projektu Processing. Pri programiranju se uporablja sintaksa jezika ANSI C oz. C++, vendar pa velja izpostaviti, da programerju v primeru slednjega niso na voljo vse standardne knjižnice. Prevajalnik avr-gcc namreč nudi le delno podporo jeziku C++. Končni program prenesemo v flash pomnilnik mikrokontrolerja preko USB-povezave. Zapis programa omogoča zagonski nalagalnik (angl. »bootloader«), ki ga proizvajalec namesti v pomnilnik. Velikost nalagalnika je približno 2 KB, kar pomeni, da ima programer namesto celotnih 16 KB pomnilnika na voljo le 14 KB. Uporaba strojnih programatorjev odpravlja to pomanjkljivost.

Za priključitev vhodno-izhodnih enot je na voljo 14 digitalnih nožic. Vsaka lahko prejme oz. zagotavlja tok jakosti 40 mA pri napetosti 5 V, poleg tega pa ima vezan pull-up upor. Velikost slednjega je med 20 in 50 kΩ in ga omogočimo po potrebi. Nekatere nožice imajo dodatne funkcionalnosti in so prikazane na sliki 3 [6].



Slika 3: Razporeditev nožic mikrokrmilnika ATmega168

Del mikrokrmilnika je tudi 6-kanalni (v primeru pakiranja TQFP in 32-pin MLF je število kanalov 8) analogno-digitalni pretvornik, ki zagotavlja 10-bitno ločljivost. Pri pretvorbi vhodne napetosti je privzeta zgornja meja primerjave napetost 5 V. To lahko spremenimo, če uporabimo nožico AREF.

Serijska komunikacija poteka preko asinhronskega komunikacijskega adapterja (angl. »Universal Asynchronous Receiver Transmitter«, krajše UART), ki je del mikrokrmilnika. Vezje FTDI FT232RL, ki se nahaja na razvojni ploščici Arduino, omogoča vzpostavitev serijske povezave preko USB-vmesnika. S tem je bistveno olajšano vzpostavljanje povezave z napravo, ki nima klasičnega serijskega konektorja DE-9, ima pa USB-vmesnik. Predhodno je potrebno namestiti ustrezne gonilnike.

3.2 Picaso MD-1

Je grafični krmilnik, ki omogoča enostavno izrisovanje slik, teksta in osnovnih geometrijskih likov. Podpira 8-bitno barvno paleto, kar zadošča za prikaz 256 barv. Z ustreznim ukazom lahko spremenimo privzeto QVGA-resolucijo (320 x 240 pik) v VGA (640 x 480 pik). V prihodnosti naj bi bila implementirana tudi SVGA-resolucija (800 x 600 pik). 512 KB statičnega bralno-pisalnega pomnilnika omogoča dvojno izravnavanje [7] (angl. »double buffering«), s katerim implementiramo animacije. Proizvajalec je z namenom, da bi povečal število sličic, ki jih lahko shranimo v pomnilnik, spremenil dejanske resolucije. Tako je dejanska QVGA-resolucija 310 x 210 pik, VGA pa 620 x 420.

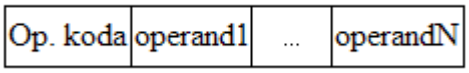
Osnovni princip dvojnega izravnavanja je naslednji: trenutna sličica animacije (angl. »frame«), torej sličica, ki je trenutno izrisana na zaslonu, je shranjena v delu slikovnega izravnalnika (angl. »frame buffer«), ki ga imenujemo prednji izravnalnik (angl. »front buffer«). Naslednjo sličico shranimo v drug del slikovnega izravnalnika, ki ga imenujemo zadnji izravnalnik (angl. »back buffer«). Preklop med zadnjim in prednjim izravnalnikom se običajno izvede s spremembo vrednosti kazalca, ki kaže na del pomnilnika, ki je trenutno izrisan na zaslonu. Naslednji izračun prikazuje število sličic, ki jih lahko v danem trenutku shranimo v pomnilnik:

$$\frac{velikost\ pomnilnika\ (v\ bajtih)}{število\ pik\ (dejanska\ resolucija)\ *\ barvna\ globina\ \ (v\ bajtih)} \tag{1}$$

V primeru VGA-resolucije lahko v pomnilnik shranimo dve sličici, trenutno in prihodnjo.

Picaso MD-1 lahko deluje samostojno ali pa v povezavi z drugo napravo. V primeru samostojnega delovanja ima programer na voljo približno 12 KB flash pomnilnika, kamor lahko shrani program, napisan v programskem jeziku 4D Graphics Language. S tem se odpravi ozko grlo, ki ga predstavlja serijska povezava. V primeru povezave z drugo napravo ima ta vlogo gospodarja, Picaso MD-1 pa vlogo sužnja. Gospodar pošlje nov ukaz po prejemu potrditve uspešne izvedbe predhodnega. Čakanje na potrditev se da odpraviti z ustrezno zakasnitvijo, ki pa jo je potrebno določiti empirično, saj čas, potreben za izvedbo posameznega ukaza, ni dokumentiran.

Picaso MD-1 naj bi v prihodnosti podpiral tudi dostop do micro-SD pomnilniške kartice. S tem bi bila olajšana izdelava kompleksnih animacij, pri katerih bi bile sličice animacije shranjene na pomnilniškem mediju. Glavni program bi skrbel za prenos ustrezne sličice v slikovni izravnalnik. Število vseh ukazov, ki jih podpira Picaso MD-1, je 26. Format ukaza prikazuje slika 4. Največje število operandov v ukazu je 260405, obstajata pa tudi ukaza brez operandov. Seznam nekaterih ukazov je prikazan v tabeli 1 [8].



Slika 4: Format ukaza pri grafičnemu krmilniku Picaso MD-1

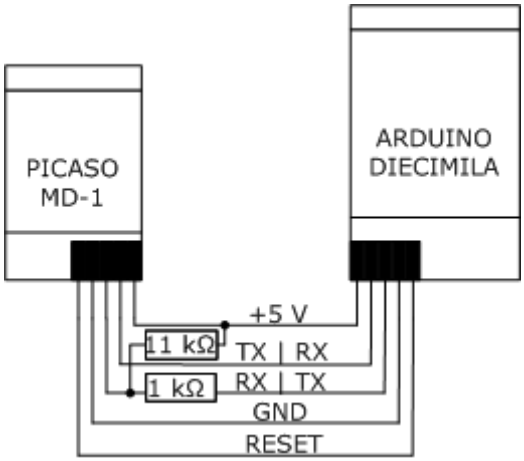
ukaz	največje število operandov	opis
bitmap copy	8	kopiranje vsebine poljubnega dela zaslona iz ene strani v drugo
erase screen	0	izbris celotnega zaslona, pri čemer se uporabi barva, ki je trenutno nastavljena za ozadje
display image	260405	izriše sliko, ki je podana z višino in širino ter barvo vsake posamezne točke (angl. "pixle")
place string of text (formatted)/(unformatted)	261/263	izpis niza ASCII znakov (največ 256) na poljubnem mestu; pri (unformatted) različici ukaza določimo višino in širino znaka
version/device info request	1	zahteva podatkov o vrsti krmilnika, vodorani in navpični ločljivosti zaslona, ...
vsync lock	0	čakanje na naslednjo sličico (angl. "frame") animacije
display control functions	2	nastavitev ločljivosti; izbira strani, ki je trenutno prikazana na zaslonu(angl. "display page"); izbira bralne (angl. "read page") in pisalne strani (angl. "write page")

Tabela 1: Seznam nekaterih ukazov grafičnega krmilnika Picaso MD-1

Dolžina operacijske kode je 8 bitov, medtem ko so operandi 8- oz. 16-bitni. Pri pošiljanju 16-bitnih operandov se upošteva pravilo debelejšega konca [9] (angl. »Big Endian Rule«). Pri večini ukazov operacijska koda natančno določa število operandov, le pri nekaterih ukazih je njihovo število lahko poljubno (vsekakor pa je to število omejeno navzgor in navzdol). Primer ukaza s poljubnim številom operandov je ukaz *Display Image*, opis katerega je v tabeli 1.

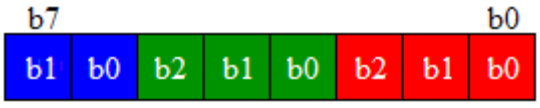
V primeru povezave z zunanjo napravo je način prenosa podatkov asinhronski. Startnemu bitu sledi osem podatkovnih bitov ter stop bit. Paritetni biti se ne uporabljajo. Po vsakem vklopu oz. resetiranju grafičnega krmilnika mora zunanja naprava poslati ukaz *auto-baud*, s čimer se uri sprejemnika in oddajnika sinhronizirata. Enota prenosa so sporočila – ukazi, ki imajo različne dolžine. Kot primer vzemimo ukaz dolžine pet bajtov. Če pošljemo grafičnemu krmilniku štiri bajte, takoj zatem pa nov ukaz, bo odgovor na izvedbo prejšnjega ukaza negativna potrditev. Generator baudne hitrosti (angl. »Baud-rate generator«) Picasa MD-1 podpira hitrosti od 2400 baudov do 1 Mbaud. Shema serijskega vmesnika je prikazana na sliki 5. Ker obstaja nevarnost, da je med inicializacijo mikrokrmilnika ATmega168 vrednost na izhodu nožice *transmit* nedoločena, je potrebna vezava pull-up upora. S tem se izognemo

težavam, ki lahko nastopijo, če Picaso MD-1 nedoločeno vrednost interpretira kot zahtevo za začetek prenosa. Ker je napetost na transmit nožici Arduino večja kot 3,6 V, je potrebna vezava upora vrednosti 1 kΩ, kot zahteva dokumentacija Picasa [8].



Slika 5: Serijska povezava med Picaso MD-1 in Arduino Diecimila

Na koncu velja omeniti še postopek, ki smo ga uporabili pri pretvarjanju barv grafičnega vmesnika v 8-bitno barvno globino. Barve grafičnega vmesnika smo določili na osnovi slik Airbusovega vmesnika, ki pa so bile narisane v 24-bitni barvni globini. Ker Picaso MD-1 podpira samo 8-bitno barvno globino, je bilo potrebno izbrane barve Airbusovega vmesnika pretvoriti. Razporeditev posameznih barv (prikazana je na sliki 6), ki jo uporablja Picaso MD-1, je naslednja: bita 7 in 6 predstavljata modro, biti 5, 4, 3 zeleno, spodnji trije biti pa rdečo barvo.



Slika 6: Razporeditev barv pri Picasu MD-1

Pretvorba 24-bitne barvne globine v 8-bitno:

1. izvedemo logično konjunkcijo med vrednostjo 224 (vrednost E0 v šestnajstiškem sistemu) in biti besede, ki določajo rdečo barvo 24-bitne barvne globine. Enako storimo z biti zelene barve, medtem ko je pri modri vrednost enaka 192 vrednost C0 v šestnajstiškem sistemu);
2. nad dobljenimi vrednostmi opravimo desni logični pomik, ki znaša v primeru bitov rdeče barve pet mest, zelene dve mesti, nad biti modre barve pa pomika ne izvedemo;

3. končno vrednost dobimo tako, da opravimo logično disjunkcijo med vrednostmi, dobljenimi v 2. točki.

3.3 Zaslon

Sestavljajo ga tri glavne komponente, in sicer krmilnik zaslona, napajalnik osvetlitve in TFT LCD-zaslon. Podrobnejši opisi posameznih komponent so v naslednjih poglavjih.

3.3.1 Kontron CRTtoLCD-6

Je krmilnik TFT LCD-zaslona, ki omogoča pretvorbo analognega ali digitalnega signala v RGB LVTTL (Low Voltage Transistor-Transistor Logic) obliko. Obstaja tudi različica, ki podpira zaslone z vmesnikom LVDS (angl. »Low Voltage Differential Signaling«). Za napajanje krmilnika potrebujemo 12 V vir enosmerne napetosti pri tokovni obremenitvi 770 mA. Glavna prednost krmilnika je v njegovi splošnosti, saj je združljiv s precejšnjim številom zaslonov različnih proizvajalcev. Spisek teh najdemo na spletni strani proizvajalca [10].

Krmilnik podpira veliko število ločljivosti, od standardne VGA (640 x 480 pik) pa vse do UXGA (1600 x 1200 pik) v primeru analognega vhodnega signala oz. SXGA (1280 x 1024 pik) pri digitalnem signalu. Poleg že omenjenih lastnosti velja omeniti podporo t. i. On-Screen Displayu (krajše OSD). S priključitvijo posebne tipkovnice, ki jo je mogoče kupiti posebej, lahko uporabnik spreminja nastavitve zaslona, kot so svetlost, pozicija slike, izbira vhodnega signala, nastavitve barv itd.

3.3.2 NEC 65PW061

Pretvornik NEC 65PW061, katerega glavna funkcija je napajanje, je potreben zaradi načina osvetlitve (angl. »backlight«) zaslona Sharp LQ64V3DG01. Slednji namreč uporablja t. i. osvetlitev CCFL (angl. »Cold Cathode Fluorescence Lamp«). Pretvornik oz. napajalnik pretvori enosmerno 12 V napetost, vir katere je krmilnik zaslona CRTtoLCD-6, v izmenično napetost s frekvenco 55 kHz. Efektivna izmenična napetost na izhodu pretvornika znaša 360 V pri tokovni obremenitvi 10 mA.

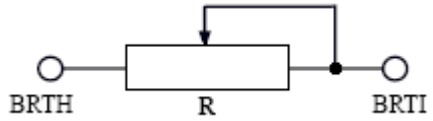
CCFL predstavlja eno izmed možnosti, ki se uporabljajo za osvetlitev LCD-zaslonov. Ostale možnosti obsegajo osvetlitev LED (angl. »Light Emitting Diode«) in EL (angl.

»Electroluminiscent«). Primerjava osvetlitev CCFL in LED je prikazana v tabeli 2. Tako eno kot drugo rešitev lahko najdemo v večini LCD-zaslonov.

	CCFL	**LED**
fizična velikost	++	--
enakomernost osvetlitve	+	-
življenska doba (-30 °C)	--	++
življenska doba (+60 °C)	++	--
vpliv na okolje	--	+
elektro-magnetna interferenca	-	+
napajalna napetost	--	+
cena	++	-

Tabela 2: Primerjava CCFL in LED osvetlitve

Izbrani napajalnik omogoča spreminjanje svetilnosti (angl. »Luminance«) zaslona z regulacijo napetosti ali upornosti [11]. Če uporabimo 10 kΩ potenciometer, ga vežemo med terminala BRTH in BRTI, kot prikazuje slika 7. Kadar je upornost 0 Ω, je svetilnost najmanjša, pri 10 kΩ pa največja.



Slika 7: Regulacija svetilnosti z uporabo potenciometra

3.3.3 Sharp LQ64V3DG01

Je barven, 6.4 palčen transmisiven LCD-zaslon z aktivno matriko, ki temelji na tehnologiji TFT (angl. »Thin Film Transistor«).

Glavne lastnosti zaslona so [12]:

- diagonala zaslona 6.4";
- VGA-ločljivost 640 x 480 pik;
- 18-bitna barvna globina;
- CCFL-osvetlitev;
- sloj proti bleščanju (angl. »Anti-Glare filter«);
- odzivni čas 29 ms;
- kontrastno razmerje 350 : 1;

- svetilnost 350 cd/m^2;
- vidni kot 140 ^{o};
- delovno temperaturno območje -30 ~ 80^o C.

Pri izbiri zaslona je imela pomembno vlogo dobavljivost. LCD-prikazovalniki, ki jih lahko najdemo na tržišču, imajo bodisi aktivno bodisi pasivno matriko. Prvi zasloni s tekočimi kristali so temeljili na pasivni matriki. V zadnjem času jo najdemo predvsem v segmentnih alfanumeričnih in manjših grafičnih prikazovalnikih, medtem ko imajo večji prikazovalniki običajno aktivno matriko. V nadaljevanju so nekoliko podrobneje predstavljeni osnovni principi delovanja obeh vrst matrik [13].

Pri pasivnih zaslonih si lahko točke poenostavljeno predstavljamo kot pomnilniške celice v pomnilniku z naključnim dostopom. Kot pri pomnilniški celici je potrebno tudi pri dostopu do posamezne točke najprej aktivirati ustrezno vrstico in nato stolpec, v katerem se točka nahaja [14]. Takšni vrsti naslavljanja pravimo neposredno naslavljanje. Vrstice in stolpci so dejansko elektrode, ki so nameščene na nasprotnih si polovicah stekla. Med proizvodnim procesom se obe polovici stekla združita, vmes pa so tekoči kristali. Z večanjem ločljivosti zaslona pri nespremenjeni diagonali je potrebno zmanjševati velikost elektrod. Posledično se poveča napetost, potrebna za napajanje zaslona. Višja napetost pa povzroča še dodatno težavo; nenamerno vpliva na sosednje točke. To pomeni, da se poleg izbrane (delno) aktivirajo še nekatere sosednje točke, kar zmanjšuje ostrino slike in kontrastno razmerje.

Zasloni z aktivno matriko odpravljajo pomanjkljivosti pasivne matrike. Osnovni princip delovanja je podoben kot pri pasivni, le da ima sedaj vsaka točka zaslona svoj par tranzistor – kondenzator, ki omogoča posredno naslavljanje. Tranzistor deluje kot stikalo, ki se aktivira, kadar je naslovljena vrstica, v kateri se nahaja. S tem se omogoči električnemu naboju, da steče preko ustreznega stolpca v točko zaslona. Ko je celotna vrstica naslovljena, se stikalo deaktivira, kar prepreči prehajanje električnega naboja. V nasprotju s pasivno matriko pri naslavljanju ne prihaja več do nenamerne aktivacije sosednjih točk. Vloga kondenzatorja je hranjenje električnega naboja skozi celoten osveževalni cikel.

Skozi celotno razlago principov delovanja smo uporabljali izraz točka oz. pika (angl. »pixel«) za označevanje elementa zaslona, ki dejansko sestoji iz treh podtočk (angl. »subpixle«). Pri zaslonih s tekočimi kristali so osnovne tri barve, torej rdeča, zelena in modra, samostojni

elementi, katerih velikost je reda nekaj tisočink milimetra (v konkretnem primeru 68 μm). Velikost posamezne točke oz. podtočke je možno enostavno izračunati po naslednjih enačbah:

$$višina\ točke\ = \frac{višina\ aktivne\ površine\ zaslona}{število\ točk\ zaslona\ v\ vertikalni\ smer} \tag{2.1}$$

$$širina\ točke\ = \frac{širina\ aktivne\ površine\ zaslona}{število\ točk\ zaslona\ v\ horizontalni\ smer} \tag{2.2}$$

$$širina\ podtočke\ = \frac{širina\ točke}{3} \tag{2.2}$$

O osvetlitvi LCD-zaslona je bilo govora že v prejšnjem poglavju, tu pa bomo nekoliko podrobneje opisal osnovne principe delovanja transfleksivnih, transmisivnih in refleksivnih zaslonov. Izbira vira svetlobe temelji na dejavnikih, kot so poraba moči, proizvajanje toplote, fizična velikost, cena ...

Transmisivni zasloni uporabljajo notranji vir (angl. »backlight«), ki je lahko LED-dioda, CCFL ali elektroluminiscenčna žarnica (podrobnejši opisi so v prejšnjem poglavju). Področje uporabe so običajno prostori, kjer ni neposredne izpostavljenosti sončni svetlobi.

Nasprotno velja za refleksivne zaslone, kjer je potrebna večja jakost zunanjega vira svetlobe. Slednja potuje preko slojev LCD-zaslona do nekakšnega ogledala, ki jo odseva v nasprotno smer. Glavna pomanjkljivost je delovanje v temnejših prostorih, kar pa je možno odpraviti z uporabo umetnega vira svetlobe (angl. »frontlight«), ki je lahko enak kot pri transmisivnem zaslonu.

Transfleksivni zasloni predstavljajo kombinacijo predhodno omenjenih načinov. Čeprav izgledajo na prvi pogled idealna rešitev, se izkaže, da delujejo nekoliko slabše kot refleksivni zasloni v zunanjem okolju in transmisivni v zaprtem prostoru. Razlog je v transflektorju, ki prepušča svetlobo notranjega vira, hkrati pa odbija zunanjo svetlobo. Transfleksivne zaslone običajno najdemo v mobilnih telefonih in zapestnih urah.

Minimalne zahteve pri izbiri zaslona so bile: 8-bitna barvna globina, vidni kot 90^o v horizontalni smeri, sloj proti bleščanju, VGA-ločljivost, široko delovno temperaturno

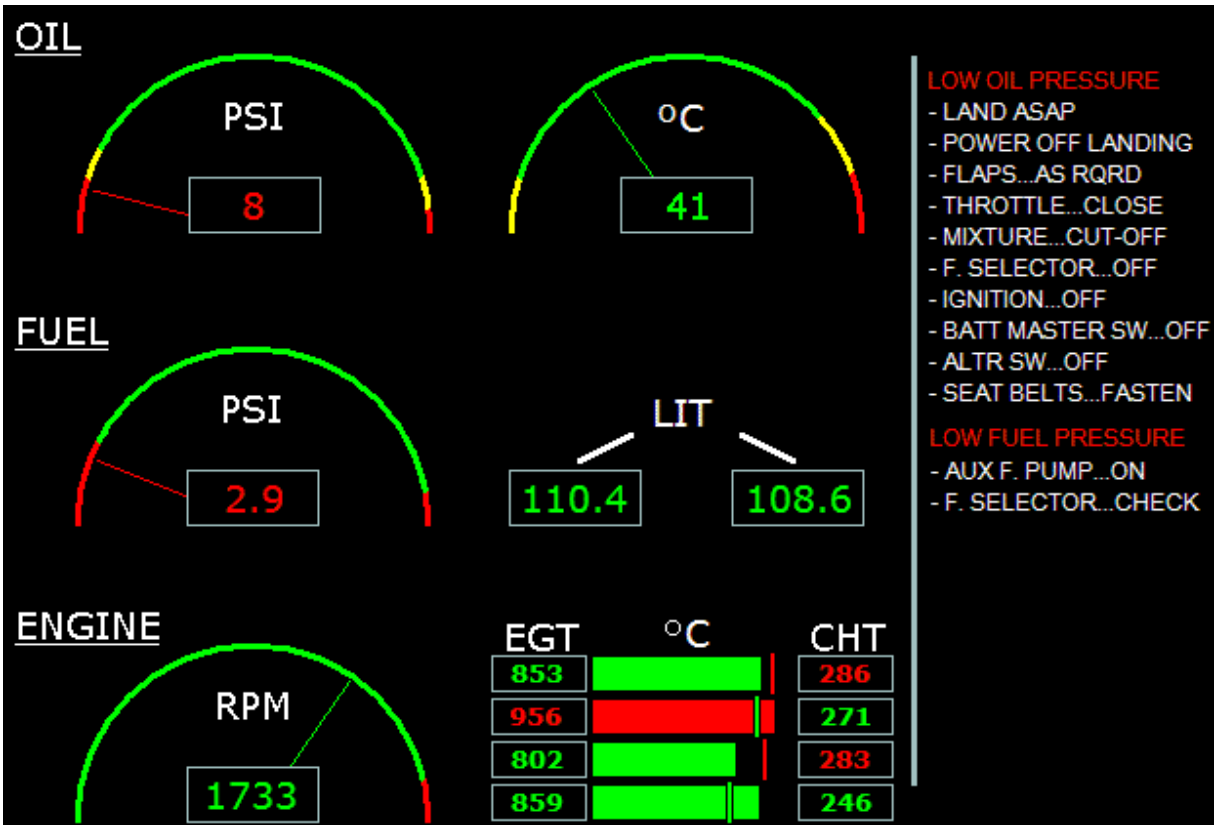
območje, združljivosti s krmilnikom Kontron CRTtoLCD-6 ter čim nižja cena. Odločilna kriterija sta bila združljivost in cena.

3.4 Grafični vmesnik

Naloga grafičnega vmesnika je predstavitev s senzorjev zajetih podatkov o motorju na nedvoumen in intuitiven način. Uporabniku naprave mora omogočati enolično interpretacijo, ki ne zahteva kompleksnega miselnega procesa, s čimer se zagotovi več časa za reševanje morebitnega problema. Grafični vmesnik sestavlja sedem različnih instrumentov, ki prikazujejo podatke, zajete s štirinajstih senzorjev.

Prva faza v procesu razvoja vmesnika je bila izdelava skice. Kot idejna podlaga so bile uporabljene naprave različnih proizvajalcev, največ pozornosti pa je bil deležen sistem EMS-D10 podjetja Dynon Avionics. Razlog je bil v preglednosti in razmeroma preprosti izvedbi vmesnika. Poleg že navedenih zahtev, ki jih mora izpolnjevati vmesnik, je bilo potrebno upoštevati tudi strojne omejitve naprave. Najbolj omejujoča dejavnika sta bila velikost pomnilnika ter zmogljivost grafičnega krmilnika. Gledano s stališča zmogljivosti, bi bilo vmesnik idealno implementirati kot množico sličic, shranjenih v obstojnem pomnilniku, pri čemer bi vsaka prikazovala določeno stanje instrumenta v času. Število sličic bi pri tem moralo ustrezati številu različnih stanj posameznega instrumenta. Ustrezna sličica bi se nato izbrala glede na vrednost, prebrano s senzorja. Ker grafični krmilnik (v času pisanja) ne podpira dostopa do dodatnega pomnilnika, flash pomnilnik krmilnika pa je premajhen, je bilo potrebno vmesnik implementirati kot množico ukazov za risanje likov, črt, teksta ... Takšna izvedba vmesnika pa zaradi večjega števila operacij pomeni slabšo zmogljivost, merjeno v številu sličic na sekundo (angl. »Frames per Second«, krajše FPS).

Na podlagi skice se je izdelala končna različica vmesnika, ki je prikazana na sliki 8. Pri tem je bil uporabljen program Microsoft Paint, ki je kljub svoji preprostosti zadoščal našim potrebam.



Slika 8: Grafični vmesnik sistema za prikazovanje podatkov o letalskem motorju

Instrumenti so smisleno razporejeni v skupine glede na podatke, ki jih prikazujejo. Kot je razvidno s slike 8, se v prvi vrsti nahajata tlak in temperatura olja. Razlog za postavitev tlaka olja v levi zgornji del zaslona je postopek ob zagonu motorja (angl. »Engine Start Checklist«), po katerem je potrebno po zagonu najprej preveriti tlak olja. Prostor desno od navpične črte je namenjen postopkom v sili.

Nekaj besed velja nameniti barvni shemi vmesnika. Slednja je pri večjih (potniških in transportnih) letalih priporočena s strani JAA v okviru predpisov JAR-25 [1, 15] (anlg. »Joint Aviation Requirements«). Čeprav naša naprava ni namenjena predhodno omenjenemu segementu letalstva, je bilo zaradi enotnosti z ostalimi podobnimi sistemi smisleno upoštevati JAA priporočila. Spodnja tabela prikazuje uporabljeno barvno shemo našega sistema, urejeno po padajoči prioriteti. Tako bo trenutna vrednost, ki pomeni opozorilo, obarvana rdeče in ne zeleno.

Vrsta obvestila	**Barva**
opozorilo (angl. »Warning«), nedovoljeno območje	rdeča
previdnost (angl. »Caution«)	rumena
trenutna vrednost	zelena/bela

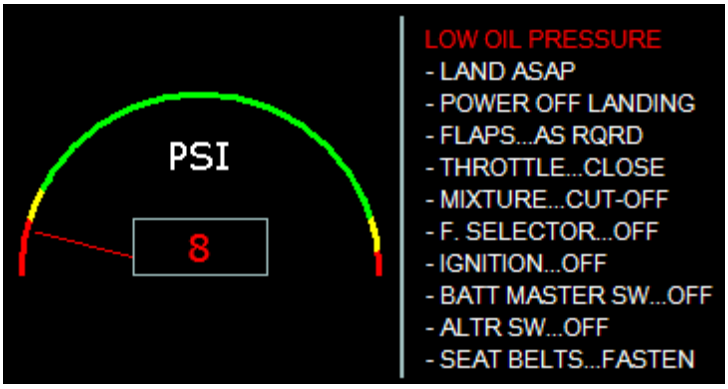
Tabela 3: Barvna shema sistema za prikazovanje podatkov o letalskem motorju

Barva kazalca in številčna vrednost, prikazana na sredini instrumenta, se spreminjata skladno z definiranimi območji instrumenta. Trenutna različica naprave uporabniku ne omogoča samostojnega spreminjanja razpona območij, ampak se ta prilagodijo glede na tip motorja pred prenosom grafičnega vmesnika v pomnilnik naprave.

3.4.1 Tlak olja

Slika 9: Tlak olja

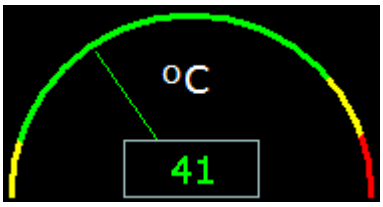
Instrument na sliki 9 prikazuje tlak olja v funtih na kvadratni palec (angl. »Pound per Square Inch«, krajše PSI). Levo rumeno področje prikazuje območje previdnosti med praznim tekom motorja (angl. »idle«). Zelena barva označuje normalno delovno področje, ki mu sledi rumeno obarvano območje previdnosti med zagonom in ogrevanjem motorja.



Slika 10: Postopek v sili v primeru prenizkega tlaka olja

V primeru previsokega ali prenizkega tlaka se barva kazalca in številčna vrednost obarvata rdeče, na desni strani zaslona pa se izpiše ustrezno obvestilo ter pripadajoči postopek v sili (angl. »Emergency Checklist«). Trenutna izvedba vmesnika izpiše zasilni postopek samo primeru previsokega tlaka.

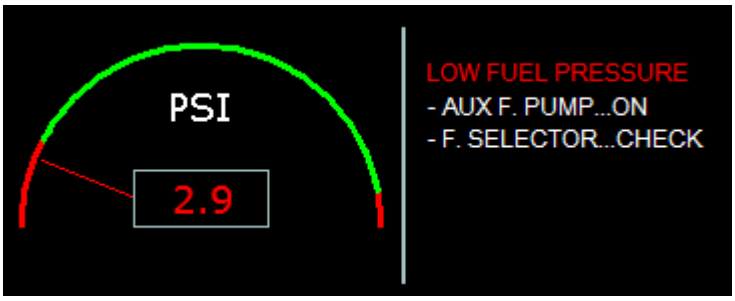
3.4.2 Temperatura olja



Slika 11: Temperatura olja

Podobno kot pri tlaku olja je tudi temperaturno območje olja razdeljeno na več vrednosti. Temperatura je prikazana v stopinjah Celzija. Postopek v sili se izpiše samo v primeru previsoke temperature in je identičen postopku, ki ga prikazuje slika 11.

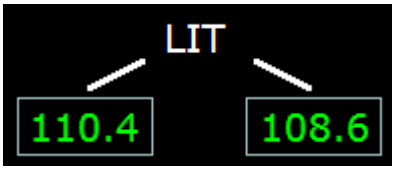
3.4.3 Tlak goriva



Slika 12: Postopek v sili v primeru prenizkega tlaka goriva

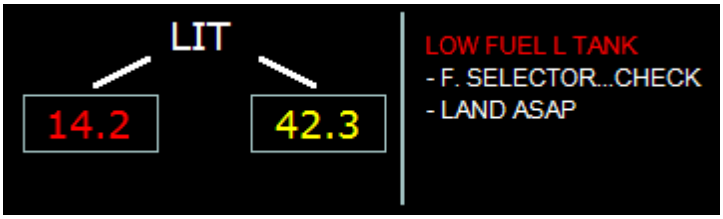
Tlak goriva je prikazan v funtih na kvadratni palec. Rdeče območje določa prenizko oz. previsoko vrednost tlaka. Postopek v sili se izpiše v primeru prenizkega tlaka, kot je prikazano na sliki 12.

3.4.4 Količina goriva



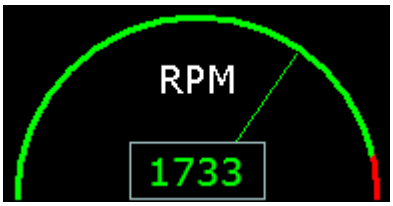
Slika 13: Količina goriva

Instrument na levi sliki prikazuje količino goriva v litrih za levi in desni rezervoar. Takšna predstavitev je primerna za letala, ki imajo rezervoarje v krilih ali pa imajo poleg glavnega še pomožni rezervoar za gorivo. V slednjem primeru je leva vrednost količina goriva v glavnem, desna pa v pomožnem rezervoarju. Vrednost se obarva rumeno, če je količina goriva manjša kot 30 % količine polnega rezervoarja, in rdeče, kadar je manjša kot 10 %. V tem primeru se izpiše tudi zasilni postopek.



Slika 14: Količini goriva v levem oz. desnem rezervoarju sta manjši kot 10% oz. 30% količine polnega rezervoarja

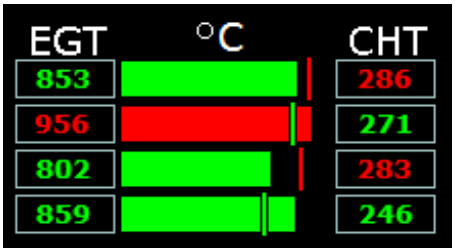
3.4.5 Obrati motorja



Slika 15: Obrati motorja

Predstavitev obratov motorja je podobna prikazom vrednosti v prejšnjih razdelkih. Območje je razdeljeno na dve vrednosti, delovno in prepovedano. Postopkov v sili ni.

3.4.6 Temperatura izpuha in glav valjev



Slika 16: Previsoka temperatura izpuha 2. valja in glave 1. ter 3. valja

Temperatura izpuha (angl. »Exhaust Gas Temperature«, krajše EGT) je prikazana v obliki vodoravnega stolpičnega diagrama, temperatura glav valjev (angl. »Cylinder Head Temperature«, krajše CHT) pa kot drsnik, ki se (neodvisno) giblje po ustreznem stolpcu. Številčne vrednosti so prikazane pod posamezno oznako (temperatura izpuha je prikazana v levem stolpcu, glav valjev pa v desnem). Vse vrednosti so prikazane v stopinjah Celzija. Najvišja vrstica ustreza temperaturi prvega valja. Za oštevilčenje se običajno uporablja vrstni red vžigov v valjih. Zelena barva označuje delovno, rdeča pa prepovedano območje. Trenutna različica vmesnika ne prikaže zasilnega postopka.

3.5 Senzorji

Skupno število senzorjev je štirinajst, pri čemer so štirje namenjeni zajemu temperature izpuha, štirje pa zajemu temperature glav valjev. Frekvenca zajemanja podatkov je spremenljiva in je odvisna od števila operacij, ki jih mora izvesti grafični krmilnik. Ko se prebrana vrednost shrani, se pošlje ukaz za brisanje stare, zatem pa še ukaz za izris nove vrednosti. Večje kot je število sprememb, nižja je frekvenca. Podatki s tipal se v povprečju preberejo približno trikrat na sekundo. Čeprav se je nedeterminiranim vrednostim v splošnem potrebno izogibati, v našem sistemu nenatančno določena frekvenca zajemanja podatkov ne predstavlja nevarnosti, saj se vse vrednosti, z izjemo obratov motorja, spreminjajo počasi. Izpisana vrednost na vsakem instrumentu predstavlja povprečje desetih zaporednih meritev posameznega tipala. V nadaljevanju so opisani osnovni principi delovanja senzorjev, ki se uporabljajo v letalstvu. Morda ni odveč izpostaviti, da so imeli starejši sistemi mehansko povezavo med tipalom in indikatorjem. To je pomenilo, da je bilo potrebno, na primer pri merjenju tlaka olja, zagotoviti ustrezno povezavo med motorjem in indikatorjem v pilotski kabini. Novejši sistemi temeljijo na uporabi oddajnika, ki se nahaja na mestu meritve in sprejemnika, ki pretvori sprejete podatke v ustrezno vrednost.

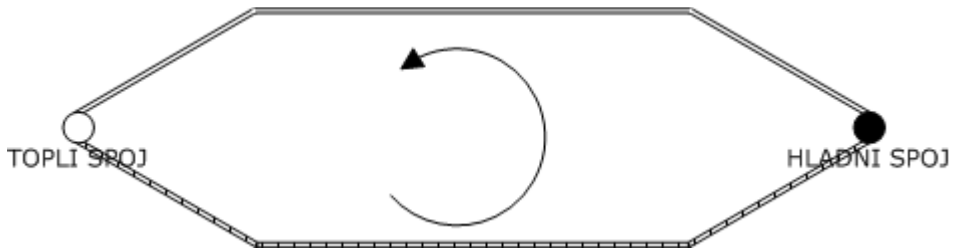
3.5.1 Tlak

Merjenje tlaka pri elektromehanskih instrumentih temelji na uporabi Bourdonove cevi ali mehu podobnih elementov, kjer se raztezanje oz. krčenje prenaša neposredno na indikator. V primeru elektronskih instrumentov pa je potrebno spremembe tlaka prevesti v elektronsko obliko. Slednje je mogoče z uporabo kondenzatorjev, ki imajo namesto ene plošče membrano, občutljivo na spremembe tlaka. Posledično se spreminja kapacitivnost kondenzatorja, ki jo je mogoče pretvoriti v električni tok. Piezouporovni senzorji tlaka imajo razporejene piezouporovne merilnike površinske napetosti po silicijevi membrani. Upori so v vezani v Wheatstonov mostiček, izhod katerega je proporcionalen tlaku. [16]

3.5.2 Temperatura

Temperaturna tipala se v letalstvu uporabljajo za merjenje temperature izpuha, glav valjev, olja, vplinjača, turbine ... Kadar imamo opraviti z visokimi temperaturami (okoli 1000^{o} C v primeru izpuha in 300^{o} C pri glavah valjev), je običajna uporaba termočlenov (angl. »thermocouple«), v primeru nižjih (100^{o} C v primeru olja) temperatur (gorivo, zrak v vplinjaču) pa se uporabljajo električni upori, katerih upornost se spreminja skladno s temperaturo – termistorji. Za povečanje natančnosti termistorjev se uporabljajo uporovni delilniki.

Termočleni temeljijo na Seebeckovem pojavu. Dva vodnika, narejena iz različnih kovin, položimo vzporedno in povežemo na obeh koncih (slika 17).



Slika 17: Termočlen

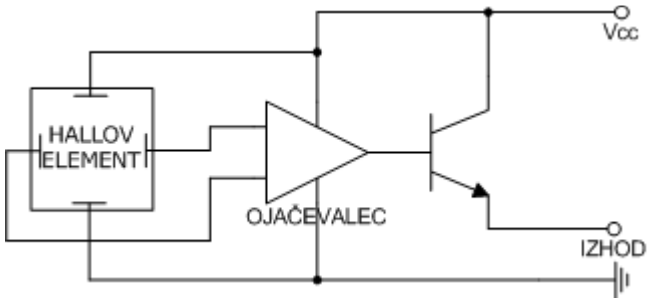
V primeru temperaturne razlike med stičiščema pride do pojava električnega toka in posledično nastanka magnetnega polja. Vroči spoj (angl. »hot junction«) predstavlja tipalo, ki ga namestimo na želeno površino (angl. »surface contact type«) ali pa ga izpostavimo toplotnemu toku (angl. »immersion type«), katerega temperaturo želimo izmeriti. Običajno je izdelano iz zmesi niklja in aluminja oz. niklja in kroma. Hladni spoj (angl. »cold junction«) se

nahaja na mestu temperaturnega indikatorja, kjer merimo električni tok, ki ga pretvorimo v temperaturno indikacijo. [1]

3.5.3 Obrati motorja

Mehanski (magnetni) tahometri in električni generatorski sistem so namenjeni merjenju obratov motorja, kadar se uporabljajo elektromehanski indikatorji. Mehanske tahometre najdemo v starejših batnih motorjih letal, medtem ko je električni generatorski sistem ena izmed najstarejših oblik merjenja obratov motorja, ki se še uporablja na velikih letalih.

Pri elektronskih instrumentih pa se običajno uporabljajo tipala, ki temeljijo na Hallovem pojavu, pri čemer je lahko izhodna vrednost analogna ali digitalna. Analogni Hallov senzor, ki ga prikazuje slika 18, deluje kot pretvornik, ki spreminja izhodno napetost glede na spremembe v magnetnem polju.



Slika 18: Analogni Hallov senzor

Pri digitalnih Hallovih senzorjih zavzame izhodna vrednost eno izmed dveh stanj, običajno predstavljenih z logično 0 in 1. Analogno obliko tipala spremenimo v digitalno z vezavo t. i Schmitt triggerja. [17]

3.5.4 Količina goriva

Količino goriva lahko izrazimo kot maso ali pa kot prostornino. Slednji način se uporablja v manjših letalih, zato je podrobneje opisan v nadaljevanju.

Najpreprostejši sistem temelji na uporabi plovca, ki z zmanjševanjem količine goriva vpliva na varistor ali potenciometer in posledično na električni tok. Glavna pomanjkljivost takšnega sistema je nelinearna indikacija, zaradi česar imamo točen podatek o gorivu samo v primeru majhne količine oz. kadar je rezervoar prazen. Težavo predstavljajo tudi manevri letala ter sprememba višine.

Kapacitivni sistem (angl. »capacitance type system«) odpravlja pomanjkljivosti predhodno omenjenega načina merjenja količine goriva. V osnovi sestoji iz vira izmenične napetosti, kondenzatorja, nameščenega v rezervoar, ojačevalca in indikatorja. Del sistema so referenčne enote (angl. »reference units«), ki zmanjšujejo napako, povezano z morebitno spremembo dielektričnosti goriva. Temeljna podlaga delovanja kapacitivnega sistema je uporaba goriva in zraka kot dielektrika v kondenzatorju. Kapacitivnost kondenzatorja je podana z naslednjo enačbo:

$$C = \varepsilon \times \frac{S}{d} \tag{3}$$

Pri konstantni površini plošč kondenzatorja (S) in razmiku med ploščama (d), je kapacitivnost (C) odvisna le od dielektričnosti (ε). Tako je v primeru polnega rezervoarja edini dielektrik med ploščama gorivo, v primeru praznega pa zrak. Iz tega sledi, da je kapacitivnost rezervoarja moč izračunati kot vsoto kapacitivnosti zraka in goriva. Za ustrezno indikacijo količine goriva je potrebno kapacitivnost pretvoriti v električni tok. Po enačbi 2 je tok (I) v izmeničnem tokokrogu odvisen le od spremembe kapacitivnosti (C) pri konstantni napetosti (V) in frekvenci (ν).

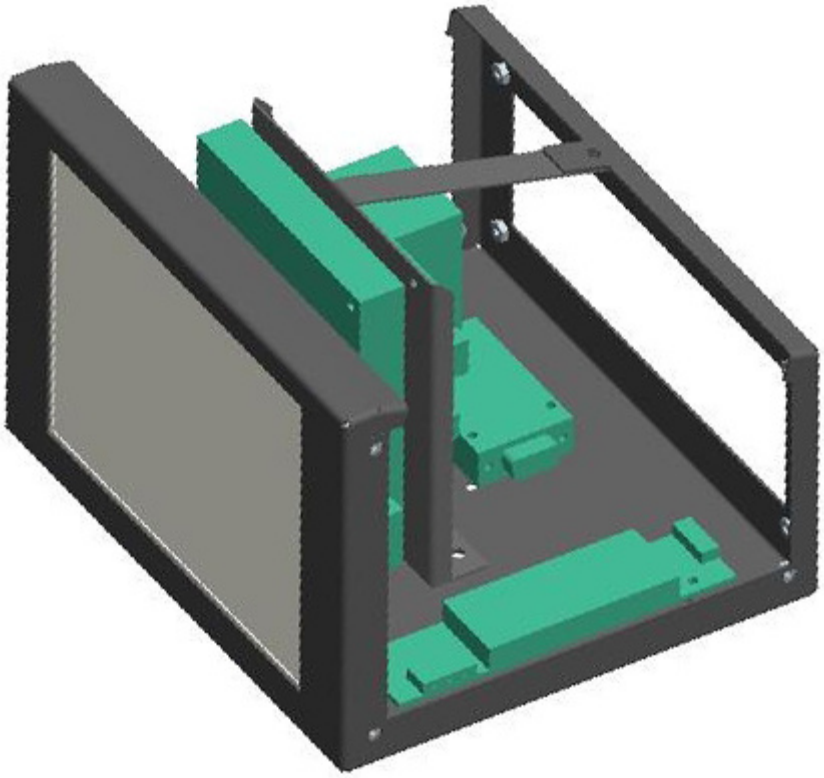
$$I = 2\pi \times \nu \times V \times C \tag{4}$$

Celotno vezje, ki presega okvire naše razlage, je predstavljeno v [1].

3.6 Ohišje

Ščiti strojne komponente pred zunanjimi vplivi, kot so vlaga, nizke oz. visoke temperature in fizične poškodbe, hkrati pa omogoča vgradnjo v letalo. Komponente so v ohišju razporejene tako, da omogočajo enostaven dostop v primeru morebitne zamenjave. Izdelano je iz kakovostne jeklene pločevine, debeline 1 mm, s pomočjo CNC-naprave. Na zadnji strani je predvidena namestitev dveh konektorjev, kamor bomo priklopili senzorje, električno napajanje in ozemljitev. Drugi konektor bo namenjen izključno temperaturnim tipalom izpuha in glav valjev. Če bi se med uporabo sistema izkazalo, da odvajanje toplote ni zadostno, se predvideva vgradnja aktivnega hlajenja. Ohišje prikazujeta sliki 19 in 20.

Slika 19: Zaprto ohišje



Slika 20: Odprto ohišje

4 Programska oprema

Celotna programska oprema je napisana v programskem jeziku ANSI C. Sestavljajo jo tri datoteke, in sicer knjižnica *Ems.h*, ki vsebuje vse grafične funkcije ter funkcije za obdelavo podatkov, datoteka *Ems_globals.h*, v kateri se nahajajo globalne spremenljivke, in glavna datoteka *Ems.cpp* (kljub končnici *.cpp* je uporabljena sintaksa jezika C in ne C++). Podrobnejši opisi sledijo v nadaljevanju.

Ključnega pomena pri programiranju je bila optimizacija velikosti programa. Kot je bilo že večkrat omenjeno, je bilo potrebno zaradi majhne velikosti pomnilnika posebno pozornost posvetiti zmanjšanju velikosti izvorne kode. Eden od načinov, s katerim smo dosegli želeno, je bil povečanje števila funkcij. Tako so bili deli kode, ki so se pojavili v programu vsaj dvakrat, združeni v novo funkcijo. Posledično se je bilo potrebno pogostokrat ubadati s problemom zmanjšanja števila argumentov. Težavo so namreč predstavljale tiste funkcije, znotraj katerih smo klicali druge. Slednje so običajno imele drugačen seznam argumentov kot prve.

Pretirana uporaba gnezdenih funkcij in funkcij z velikim številom argumentov v splošnem ni priporočljiva. Delo, ki ga mora ob vsakem klicu opraviti centralna procesna enota, se bistveno poveča. Zmanjšanje zmogljivosti lahko pričakujemo v primerih, ko je potrebno pred skokom na gnezdeno proceduro oz. funkcijo shraniti veliko število programsko dostopnih registrov, tudi kadar vemo, da se ti ne bodo spremenili. Pomanjkljivostim navkljub je bilo zmanjšanje velikosti izvorne kode občutno.

Pazljiva izbira ustreznih podatkovnih tipov je bil naslednji način, ki se je uporabljal pri optimizaciji. Tako so bile na primer pri prvi različici programa koordinate objektov na zaslonu definirane kot nepredznačne 16-bitne vrednosti (ker je največja vrednost 640, je ne moremo predstaviti s samo osmimi biti). V procesu razvoja se je izkazalo, da je bilo potrebno zaradi spremenjene izvedbe ene izmed funkcij vrednosti koordinat definirati kot predznačene.

Tudi podatki, ki jih zajemamo s senzorjev, so bili deležni posebne pozornosti. Tako so bili kot zgornja meja intervala vrednosti, ki jih lahko zavzamejo posamezni podatki, definirani obrati motorja. Spodnjo mejo je na začetku predstavljala vrednost 0, kasneje pa se je izkazala potreba tudi po negativnih vrednostih. Čeprav so pri pravilno delujočem motorju vrednosti, ki jih zajemamo, pozitivne, bo v naslednji različici naprave dodan še senzor zunanje

temperature. Glede na povedano se za shranjevanje zajetih podatkov uporabljajo 16-bitne predznačene celoštevilske spremenljivke.

Tretja metoda zmanjševanja velikosti programa je bila implementacija nekaterih funkcij, ki so vsebovane v standardnih knjižnicah jezika C. Lastna definicija funkcije *sprintf()* je pomenila zmanjšanje velikosti izvorne kode za 1262 bajtov v primerjavi z uporabo funkcije, ki je del knjižnice *stdio.h*.

Čeprav je bila primarnega pomena minimizacija velikosti programske kode, smo hkrati poizkušali doseči tudi skrajšanje izvajalnega časa programa. Nekateri prijemi, ki smo jih uporabili, vključujejo zmanjšanje operacij s števili v plavajoči vejici. Slednja so, kjer je to smiselno, predstavljena kot cela števila, ki jim dodamo spremenljivko, ki predstavlja število decimalnih mest.

Uporaba polj (angl. »array«) je priporočljiva, saj pripomore k zadostitvi principa lokalnosti dostopov. Spremenljivke, ki so predstavljale logično enoto, so zato shranjene v eno- in dvodimenzijska polja. Pri zankah s kompleksnimi pogoji se le-ti izračunajo pred vstopom v zanko, s čimer se izognemo vsakokratnemu računanju vrednosti, ki se med izvajanjem zanke ne spreminja. Ena izmed možnih optimizacij, ki je zaradi omejenega pomnilniškega prostora nismo implementirali, je vnaprejšen izračun kotnih funkcij danega kota.

V primeru pogojnih stavkov s kompleksnimi pogoji so slednji razporejeni po padajočih verjetnostih njihove izpolnitve. Tako je kot prvi pogoj izbran tisti, za katerega obstaja največja verjetnost, da bo izpolnjen. Kot zadnji način optimizacije naj omenimo še zmanjšanje števila lokalnih spremenljivk. S tem dosežemo, da lahko vse spremenljivke neke funkcije shranimo v registre procesorja, s čimer se posledično zmanjša število pomnilniških dostopov. Dodatna prednost je v manjši zasedenosti sklada in hitrejši izvršitvi skočnega ukaza, saj je v primeru klica gnezdene funkcije potrebno shraniti lokalne spremenljivke. [18]

4.1 Ems.cpp

Datoteka Ems.cpp vsebuje izvorno kodo glavnega programa, katere zapis v psevdokodi je naslednji:

1. incializiraj globalne in ostale spremenljivke na privzete vrednosti
2. prični serijski prenos in zapiši grafični vmesnik v privzeto (stran 0) pisalno stran (angl. »write page«) grafičnega krmilnika
3. prikaži na zaslonu pravkar zapisano stran in zapiši grafični vmesnik še v stran 1
4. ponavljaj(neskončno):
   4.1. preberi vrednosti s senzorjev in jih shrani v tabelo, za vsako stran posebej
   4.2. preveri, ali se vrednost povečuje ali zmanjšuje
   4.3. na zaslonu prikaži stran 1 in izberi pisalno stran 0
      4.3.1. za vsako vrednost, zapisano v stran 0, določi barvo
      4.3.2. izriši vrednosti in izračunaj pozicije naslednjih
      4.3.3. če se vrednosti nahajajo v prepovedanem območju, izpiši ustrezen postopek v sili
   4.4. na zaslonu prikaži stran 0 in izberi pisalno stran 1
      4.4.1. ponovi podtočki 4.3.1 in 4.3.2 še za stran 1

Na začetku izvorne kode najdemo dekleracije makrojev, ki določajo največje dovoljene vrednosti posameznih instrumentov (kot je bilo že predhodno omenjeno, so najmanjše vrednosti enake nič). Uporabljajo se pri definiciji števila senzorjev ter posameznih območij instrumentov (podrobnejši opis sledi v nadaljevanju). Makrojem sledijo dekleracije in inicializacije globalnih spremenljivk, ki so zbrane v datoteki *Ems_globals.h*. Tu gre predvsem za definicije posameznih barv, velikosti črk in števila zasilnih postopkov.

Vrednosti, prebrane s senzorjev, shranimo v polje *input_value[NOOFINPUTS][4]*, kjer pomeni spremenljivka *NOOFINPUTS* število različnih senzorjev. Če si polje predstavljamo kot matriko, je v našem primeru število stolpcev enako štiri. V prvem stolpcu so shranjene vrednosti, ki so trenutno prikazane na pisalni strani 0. Podobno velja za drugi stolpec, kjer so vrednosti, prikazane na pisalni strani 1. Tretji stolpec vsebuje končno vrednost (dejansko gre za vrednost, ki jo ob vsakem obhodu zanke preberemo s posameznega tipala), ki jo morata doseči vrednosti v prvem in drugem stolpcu. Zadnji stolpec pa pove, ali se trenutno prikazana vrednost glede na končno povečuje ali zmanjšuje.

Območja previdnosti, prepovedana ter delovna območja vsakega instrumenta so shranjena v dvodimenzionalnem polju *limits_ime_instrumenta[stevilo_obmocij][2]*. Elementi, katerih indeks druge dimenzije je 0, so zgornja meja območja, na mestu z indeksom 1 pa je shranjena barva območja. Barvo kazalca ali drsnika in številčne vrednosti posameznega podatka se določi s primerjavo zgornje meje območja ter vrednosti, ki je shranjena v trenutni pisalni strani.

Vsak postopek v sili je shranjen kot polje kazalcev na posamezen niz znakov. Nizi so lahko dolgi največ 256 znakov, zaradi omejitve, ki jo določa ukaz *place string of text* grafičnega krmilnika. Za izpis ustreznega postopka se uporablja polje *warning_table[NOOFWARNINGS][2]*. Spremenljivka *NOOFWARNINGS* pomeni število različnih postopkov v sili, medtem ko je v prvem stolpcu polja shranjena ordinata postopka, zapisanega na zaslonu, v drugem pa podatek o tem, ali je postopek izpisan ali ne. Zaradi poenostavitve algoritma za izpis zasilnega postopka, se ta hkrati zapiše v obe pisalni strani glede na vrednost, ki se nahaja v prvem stolpcu polja *input_value*. Pomanjkljivost takšne izvedbe je razvidna v primeru, ko je vrednost drugega stolpca polja *input_value* že v prepovedanem območju, vrednost prvega pa še ne. V tem primeru se zasilni postopek ne izpiše.

4.2 Ems.h

V nadaljevanju sledijo opisi funkcij, ki so bistvene za razumevanje delovanja programske opreme. Glede na to, da je bil prvotni namen diplomskega dela pridobitev osnovnih znanj s področja razvoja elektronskega sistema letalskih instrumentov, so bili pri izvedbi nekaterih funkcij uporabljeni načini programiranja, katerih uporaba ni priporočljiva. Namreč, prihodnje različice grafičnega vmesnika bodo po vsej verjetnosti deležne bistvenih sprememb, tako da je stopnja ponovne uporabe programske kode precej nizka. Večji poudarek je na učenju konceptov delovanja in ne toliko na izvedbi.

- *getAnalogData* vrne celoštevilsko predznačeno 16-bitno vrednost pomnoženo s številom decimalnih mesti, ki je podan kot argument funkcije. Funkcija na podlagi linearne aproksimacije pretvori vrednost, ki jo prebere z ustreznega kanala (številka kanala je podana kot argument) A/D-pretvornika. Kot argumenta sta podana še naklonski koeficient

premice *k* in odsek na ordinatni osi *n*, ki sta predstavljena kot 32-bitni števili v plavajoči vejici. Pretvorba tipa *float* v celoštevilsko vrednost je eksplicitna.

- *numberToString* vrne kazalec na niz znakov. Uporablja se za pretvorbo celoštevilske 16-bitne predznačene vrednosti, ki je podana kot argument, v niz znakov. Dinamično alokacijo pomnilniškega prostora, ki ga zasede niz, dosežemo s klicem funkcije *malloc*. Argument slednje je dolžina niza, izražena v bajtih, ki jo izračunamo kot desetiški logaritem števila, ki ga želimo pretvoriti. V prvi različici programa je bila za pretvorbo vrednosti uporabljena funkcija *sprintf*, ki pa smo jo z namenom zmanjšanja velikosti programske kode samostojno implementirali. Sprostitev alociranega pomnilnika je potrebno opraviti eksplicitno po končani izvedbi funkcije *numberToString*. Če bi programsko kodo spreminjali programerji, ki niso sodelovali pri razvoju, obstaja nevarnost prenehanja delovanja programa, zaradi t. i. »memory leak«. Drugi argument funkcije je število decimalnih mest, ki se uporablja pri pretvorbi števila v plavajoči vejici, zapisano kot celoštevilska vrednost v niz znakov.

- *waitACK* čaka na prvo vrednost, ki se pojavi v serijskem medpomnilniku (angl. »serial buffer«). Vrednost je lahko poljubna in ni nujno ACK, kot bi lahko sklepali po imenu funkcije. Razlog za takšno izvedbo je v dejstvu, da je edina naprava, ki je v serijski povezavi z mikrokrmilnikom, grafični krmilnik. Ker so vsi izdani ukazi pravilni (za kar mora v celoti poskrbeti programer), tudi ni potrebe po ločevanju pozitivne in negativne potrditve. Slednja se namreč pošlje samo v primeru nepravilnega ukaza, ne pa tudi v primeru pravilnega ukaza, ki ga krmilnik ni uspel izvesti. Tako se lahko vedno predpostavlja, da je odgovor grafičnega krmilnika pozitivna potrditev. Po prejemu potrditve se medpomnilnik izprazni.

- *displayText* izpiše niz znakov, podanih kot argument funkcije. Ker se funkcija uporablja tudi za izpis številskih vrednosti, jih je potrebno sredinsko poravnati znotraj pravokotnika posameznega instrumenta glede na spreminjajoče se število števk, kot je prikazano na sliki 8 v poglavju 3.4. Posledično se premaknejo tudi nizi znakov, ki ne predstavljajo zapisa števila. Ta premik je potrebno upoštevati pri določanju pozicije nizov, kar lahko privede tudi do negativne vrednosti katere od koordinat. Zato so slednje definirane kot predznačene celoštevilske vrednosti, za razliko od ostalih funkcij, kjer so nepredznačene.

- *drawDial* poskrbi za izris kazalca glede na kot, podan v stopinjah in središčno koordinato instrumenta (oba podana kot argumenta funkcije). Končno koordinato črte, ki predstavlja kazalec, dobimo z izračunom vrednosti sinusa in kosinusa podanega kota. Rezultat se ustrezno zaokroži in pretvori v celoštevilsko vrednost. Začetna koordinata pa se izračuna kot presečišče premice, ki je nosilka daljice oz. črte, ki predstavlja kazalec in pravokotnik v središču instrumenta. Razlog za uporabo takšnega pristopa je v povečanju števila izrisanih sličic na sekundo. Če bi bila začetna koordinata kazalca enaka središčni koordinati instrumenta, bi črta prekrila številsko vrednost, ki je zapisana v sredini instrumenta in pravokotnik, ki jo obdaja. Tako bi bilo potrebno ponovno izrisati vrednost in pravokotnik. Čeprav uporablja naša izvedba za izračun presečišča kotno funkcijo tangens, je število sličic večje, kot bi bilo v primeru uporabe predhodno opisane možnosti.

- Osnovna naloga funkcije *animateDial* je izbris kazalca in ponoven izris na novi poziciji. Jedro funkcije se izvede samo v primeru, ko se vrednost, ki jo želimo prikazati, spreminja (bodisi narašča bodisi zmanjšuje). Brisanju kazalca sledi izpis labele, ki označuje mersko enoto (PSI, ^{o}C, RPM). Slednje je potrebno, kadar se kazalec nahaja v položaju, ki prekriva mersko enoto. Težava namreč nastopi pri brisanju, saj se poleg kazalca izbriše tudi del labele. Omeniti velja, da se v primeru naše implementacije slednja ponovno izpiše ne glede na pozicijo kazalca. Razlog za takšno odločitev je manjša velikost programske kode in neopazno zmanjšanje zmogljivosti. Glede na povečevanje oz. zmanjševanje vrednosti izračunamo naslednji položaj kazalca, ki ga nato izrišemo v ustrezni barvi (podana je kot argument funkcije). Sledi klic funkcije *numberToString*, brisanje stare številske vrednosti ter izpis nove. Nize znakov izbrišemo z ustrezno velikim polnim pravokotnikom črne barve. Izkazalo se je, da je čas izvedbe ukaza grafičnega krmilnika *draw rectangle* manjši kot pri ukazu *place string of text*.

  Psevdokoda izračuna naslednje pozicije kazalca:

```
1. če se vrednost povečuje/zmanjšuje, potem
   1.1. če ((vrednost na strani 0 ≤/≥ vrednosti na strani 1) ali
        (stran = 1)) potem
```

```
1.1.1. če (vrednost na strani 0 = vrednost na strani 1)
       potem (naslednja pozicija = trenutna +/- 1)
1.1.2. sicer (naslednja pozicija = trenutna +/- 2)
1.1.3. če (na strani 0 ≥/≤ vrednost na strani 1) potem
       (naslednja pozicija = končna vrednost)
  1.1.3.1. če (vrednost na strani 0 = vrednost na strani 1)
           potem (vrednost se ne spreminja)
```

- *animateChtEgt* deluje podobno kot predhodno opisana funkcija. Prva pomembnejša razlika je v zanki, katere število obhodov je enako dvakratnemu številu valjev motorja. Jedro funkcije se tako v primeru štirivaljnega motorja izvede osemkrat. Prvi štirje obhodi zanke so namenjeni animiranju vrednosti temperature izpuha, naslednji štirje pa temperaturi glav valjev. Glede na povedano morajo biti posamezne temperature v polju *input_value* shranjene v pravilnem vrstnem redu. Naslednja pomembnejša razlika, glede na predhodno funkcijo, je v načinu brisanja in izrisovanja. V primeru povečevanja temperature izpuha trenutno prikazane vrednosti, ponazorjene s pravokotnikom, ne izbrišemo, temveč samo izrišemo naslednjo. Ker s tem prekrijemo drsnik, moramo tudi tega ponovno izrisati. V primeru zmanjševanja vrednosti je potrebno pred prikazom naslednje izbrisati trenutno vrednost oz. pravokotnik, ki jo ponazarja. Tudi pri animiranju temperature glav valjev je potek podoben. Ne glede na to, kako se vrednost spreminja, najprej izbrišemo drsnik na trenutni poziciji. S tem hkrati izbrišemo tudi del pravokotnika, na katerem se drsnik nahaja (za lažjo predstavo, si lahko bralec ogleda sliko 8 v poglavju 3.4). Zato najprej ponovno narišemo celoten pravokotnik (dejansko bi morali izrisati samo manjkajoči del, vendar pa bi s tem bistveno povečali kompleksnost in velikost programske kode) nato pa še drsnik. Prikazovanje številskih vrednosti je izvedeno na enak način kot pri funkciji *animateChtEg*.

- *displayGauge* izriše polkrožni instrument, vključno z napisom in pravokotnikom v sredini. Algoritem za izris polkroga temelji na risanju množice črt. Izhodiščno in končno točko posamezne črte izračunamo v odvisnosti od kosinusa oz. sinusa kota, ki zavzame vrednost na intervalu [0, PI]. Spremembe kota so v korakih po 0,003 radiana, kar pomeni, da je interval razdeljen na 1047 vrednosti. Na prvi pogled je število izračunov, ki jih je potrebno opraviti, (pre)veliko. Slednje postane še bolj očitno, če upoštevamo, da

ATmega168 nima enote za računanje s števili v plavajoči vejici (angl. »Floating Point Unit«, krajše FPU). Vendar pa velja izpostaviti, da instrumente izrišemo samo pri inicializaciji posamezne pisalne strani, nato pa ne več. Če bi želeli pospešiti izris, bi lahko število v plavajoči vejici pretvorili v celoštevilsko vrednost. Dodatno pospešitev je mogoče doseči tudi s shranjevanjem vnaprej izračunih vrednosti kotnih funkcij. Žal pa ta rešitev zahteva več pomnilniškega prostora, ki ga pri naši napravi močno primanjkuje. Zgoraj omenjene pomanjkljivosti naše izvedbe bi lahko odpravili z uporabo t. i. *midpoint* algoritma, ki temelji na Bresenhamovem algoritmu za risanje črt [19]. Algoritem uporablja samo celoštevilske vrednosti in za delovanje ne potrebuje kotnih funkcij. Z delitvijo krožnice na osmine lahko izkoristimo njeno simetričnost. Algoritem hkrati riše istoležne točke krožnice v posameznih oktantih kroga. Ravno simetričnost je tista, ki nam otežuje uporabo algoritma. V našem primeru je krožnica sestavljena iz krožnih lokov različnih barv. Če bi želeli, da sta istoležni točki v ustrezni barvi, bi bilo potrebno bistveno spremeniti *midpoint* algoritem. To pa nas zopet privede do povečanje velikosti programske kode.

- *displayWarning* izpiše ustrezen postopek v sili kot rezultat primerjave med trenutno in mejno vrednostjo. Algoritem izpiše vse tiste zasilne postopke oz. obvestila, za katere ugotovi, da niso trenutno prikazani in izpolnjujejo pogoj za izpis (trenutna vrednost se nahaja v prepovedanem območju). Zaradi poenostavitve izvedbe se postopek hkrati zapiše v obe pisalni strani. Nato se poišče y-koordinata že prikazanega obvestila, ki ima najnižjo pozicijo v stolpcu obvestil (iščemo največjo vrednost y-koordinate). Tej vrednosti prištejemo ustrezen odmik in tako dobimo pozicijo, kamor bomo zapisali naslednji zasilni postopek. V primeru brisanja poiščemo najmanjšo vrednost tiste y-koordinate, na kateri še ni zapisanega obvestila. »Lepotna napaka« takšne izvedbe je v že izpisanih postopkih. Ti namreč ohranijo svoj položaj tudi po brisanju obvestil, ki so se nahajala nad njimi, kar ima za posledico prazen prostor.

  Psevdokoda algoritma *displayWarning*:

```
1. če ((vrednost1 < vrednost2) in postopek ni prikazan), potem
   1.1. zapiši zasilni postopek na pozicijo, ki mu je določena v
        pisalno stran 0
```

1.2. zapiši zasilni postopek na pozicijo, ki mu je določena v pisalno stran 1
1.3. zabeleži, da je postopek prikazan
1.4. poišči največjo y-koordinato že prikazanega postopka
1.5. vsem neprikazanim postopkom določi pozicijo, dobljeno v točki 1.4, in ji prištej ustrezen odmik

2. sicer če((vrednost1 > vrednost2) in postopek je prikazan), potem
2.1. izbriši zasilni postopek na strani 0
2.2. izbriši zasilni postopek na strani 1
2.3. zabeleži, da postopek ni prikazan
2.4. poišči najmanjšo y-koordinato neprikazanega postopka
2.5. vsem neprikazanim postopkom določi pozicijo, dobljeno v točki 2.4

5 Sklepne ugotovitve

Ideja o izdelavi elektronskega letalskega instrumenta je bila sprva videti težko uresničljiva. Podjetja, ki izdelujejo takšne naprave, imajo zaposlene visoko specializirane kadre z izkušnjami pri razvoju in izdelavi elektronskih naprav. Že sama misel, da bi bilo mogoče v »domači dnevni sobi« razviti napravo, ki bila tudi praktično uporabna, je bila absurdna. Navkljub začetnim pomislekom je končni izdelek presenetljivo zadovoljil vsa naša pričakovanja. Čeprav bo potrebno v izdelek, s katerim bo mogoče konkurenčno nastopati na prodajnem trgu, vložiti še veliko truda, menimo, da smo na dobri poti k uresničitvi zastavljenega cilja.

Razvoj bo potekal postopoma in bo v prvi fazi zajel dodelavo in izboljšavo obstoječe naprave. V kolikor bi bili rezultati spodbudni, je za prihodnost načrtovana tudi izdelava primarnega letalnega in navigacijskega sistema. S tem bi dobili zaokroženo celoto, ki bi jo tržili kot elektronski sistem letalskih instrumentov.

Prva izboljšava, ki bo implementirana v naslednji različici, bo zamenjava mikrokrmilnika ATmega168. Čeprav je zmogljivost slednjega, gledano s stališča izvrševanja ukazov, povsem zadovoljiva, je velikost obstojnega pomnilnika preskromna. V pomoč ni tudi dejstvo, da ne podpira priključitve dodatnega zunanjega pomnilnika. Zamenjava bo verjetno mikrokrmilnik iz družine ARM (angl. »Advanced RISC Machine«), kjer bo eden izmed glavnih kriterijev izbire ravno pomnilnik.

Zamenjave bo verjetno deležen tudi grafični krmilnik Picaso MD-1. Razlog je predvsem v njegovi razmeroma slabi razširjenosti in posledično preverjenosti. S povečanjem kompleksnosti grafičnega vmesnika bo sčasoma postala problematična tudi zmogljivost krmilnika. Ena od zahtev je neposredna povezava LCD-zaslona z grafičnim krmilnikom preko vmesnika LVDS ali TTL, s čimer bi odpravili analogni VGA-signal.

Z razvojem lastnih vezij bi bistveno znižali končne stroške naprave, saj so posamezne komponente (mikrokrmilnik, pomnilnik, periferne enote) precej cenejše kot že izdelani sistemi. Poveča se tudi prilagodljivost strojne opreme našim potrebam. Tako glavni kriterij za izbiro ne bo samo funkcionalnost, temveč tudi poraba, delovno območje itd.

Dodatno znižanje stroškov je mogoče doseči tudi z zmanjšanjem števila komponent. Tu gre predvsem za odstranitev delov sistema, ki jih lahko nadomestimo z enim samim. V primeru

naše naprave bi bila smiselna uporaba mikrokrmilnika z vgrajenim LCD-krmilnikom, kot sta na primer Atmel AT91SAM9261 ali NXP LH7A400. Tako bi lahko LCD-zaslon priključili neposredno na mikrokrmilnik, s čimer bi se znebili analogne VGA-povezave.

Naslednja različica grafičnega vmesnika bo morala nuditi večjo prilagodljivost. To pomeni, da bo uporabnik sam izbral tip motorja in posamezne parametre, kar se bo ustrezno odrazilo na vmesniku. S tem bi dobili popolnoma univerzalno napravo, ki bi podpirala veliko število različnih letalskih batnih motorjev. Na voljo sta dve možnosti, in sicer razširitev naprave s preprosto enovrstično tipkovnico, ki bo omogočala vnos parametrov, ali pa izdelava programske opreme. Ta bi med drugim lahko uporabniku omogočala lastno razporeditev instrumentov, spremembo formata izpisa obvestil ... Spremembe bi se nato preko USB-vmesnika shranile v pomnilnik naprave.

Sistemu postopkov v sili bo potrebno dodati zvočne signale. Tako bi vsakemu opozorilu ustrezal enoličen zvočni signal, ki bi še dodatno opozarjal na prisotnost nevarnosti. Čeprav je predlogov za izboljšave še veliko, menimo, da smo izpostavili najpomembnejše. Ravno ideje so namreč tiste, ki predstavljajo temelj za uspešno prihodnost.

Primarni cilj je vgradnja sistema v letalo, s čimer bi lahko pričeli fazo testiranja v realnem okolju. Ključnega pomena je odziv uporabnikov in njihovi predlogi za morebitne izboljšave. V kolikor bi se pokazalo veliko zanimanje lastnikov letal, se predvideva povečanje števila članov razvojne skupine. Kljub nekoliko (pričakovano) zadržanim začetnim odzivom, upamo, da se bo naprava izkazala kot zanesljiv, kakovosten in funkcionalen produkt, ki bo v bližnji prihodnosti predstavljal resno konkurenco ostalim podobnim izdelkom na trgu.

Priloga A: Načrt naprave

Priloga B: Kazalo slik

Priloga C: Kazalo tabel

Literatura in viri

[1] Oxford Aviation Services Limited, *Aircraft General Knowlegde 4 – 022 Instrumentation*, Frankfurt: Jeppesen GmbH, 2001.

[2] (2009) Wikipedia: Electronic Flight Instrument System. Dostopno na: http://en.wikipedia.org/wiki/EFIS

[3] (2009) Dynon Avionics – Affordable Glass Cockpit Avionics. Dostopno na: http://www.dynonavionics.com

[4] (2009) Arduino Diecimila. Dostopno na: http://arduino.cc/en/Main/ArduinoBoardDiecimila

[5] (2009) 4D Systems, µVGA-PICASO-MD1. Dostopno na: http://www.4dsystems.com.au/prod.php?id=15

[6] (2009) Atmel, ATmega48/88/168 Product Datasheet, revision P. Dostopno na: http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf

[7] (2009) Osnove 3D pospeševanja. Dostopno na: http://hercules.uni-mb.si/articles/Recenzije%5COsnove%203D%20pospe%C5%A1evanja.pdf

[8] (2009) 4D Systems, µVGA-PICASO-MD1 Users Manual revision 1.2. Dostopno na: http://www.4dsystems.com.au/downloads//micro-VGA/uVGA-PICASO-MD1/Docs/Pdf/uVGA-PICASO-MD1_Serial_Users_Manual_Rev1.2.pdf

[9] D. Kodek, *Arhitektura računalniških sistemov*, Ljubljana: BI_TIM, 1994, str. 149.

[10] (2009) Kontron, CRTtoLCD-6 Flatpanel Configurator. Dostopno na: http://emea.kontron.com/products/boards+and+mezzanines/flatpanel+controller/crttolcd6.html

[11] (2007) NEC LCD Technologies, Ltd., Inverter 65PW061 Datasheet. Dostopno na: http://www.farnell.com/datasheets/123128.pdf

[12] (2006) Sharp Microelectronics of the Americas, Sharp LQ64V3DG01 Documentation. Dostopno na: http://document.sharpsma.com/files/LQ064V3DG01_SS_110206.pdf

[13] (1993) Fundamental Liquid Crystal Display Technology: an introduction for the basic understanding of LCDs. Dostopno na: http://www.repairfaq.org/filipg/LINK/F_Theory_LCD1.html

[14] D. Kodek, *Mikroprocesorski sistemi*, Ljubljana: BI-TIM, 1993, pogl. 7.2.

[15] (2009) Joint Aviation Authorities, JAR-25. Dostopno na: http://www.jaat.eu/publications/jars/JAR%2025.pdf

[16] (2009) How they work – The pressure sensor. Dostopno na: http://www.sensorland.com/HowPage004.html

[17] (2009) Honywell, Hall Effect Sensing and Application. Dostopno na: http://content.honeywell.com/sensing/prodinfo/solidstate/technical/chapter2.pdf

[18] (2009) Optimizing C and C++ Code. Dostopno na: http://www.eventhelix.com/realtimemantra/basics/optimizingcandcppcode.htm

[19] (2009) Wikipedia: Midpoint circle algorithm. Dostopno na: http://en.wikipedia.org/wiki/Midpoint_circle_algorithm

Izjava o avtorstvu diplomskega dela

Izjavljam, da sem diplomsko delo izdelal samostojno pod vodstvom mentorja prof. dr. Dušana Kodeka. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali. Soglašam, da je diplomsko delo v elektronski obliki dostopno v zbirki »Dela FRI«.

Ljubljana, 10. marec 2009 Luka Hauptman