UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Bojan Pikl

Implementacija vgrajenega računalnika

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: prof. dr. Saša Divjak

Ljubljana, 2011

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: 00152/2011
Datum: 05.09.2011

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

Kandidat: **BOJAN PIKL**

Naslov: **IMPLEMENTACIJA VGRAJENEGA RAČUNALNIKA**
IMPLEMENTATION OF AN EMBEDDED COMPUTER

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Cilj diplomske naloge je izdelava vgrajenega računalnika. Opišite tehnologije določenih delov, ki so trenutno dobavljivi na trgu. Pri vsakem podsklopu navedite, katero tehnologijo ste izbrali in to utemeljite. Predstavite pripravo operacijskega sistema Windows Embedded CE 6.0R3. Poleg same izdelave slike operacijskega sistema opišite tudi postopek povezovanja ciljnega in razvojnega računalnika pred in po namestitvi operacijskega sistema. Opišite tudi razvoj podsklopov namenskega programa v tehnologiji Silverlight for Windows Embedded.

Mentor:

prof. dr. Saša Divjak

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Bojan Pikl, z vpisno številko 63050081, sem avtor diplomskega dela z naslovom:

Implementacija vgrajenega računalnika

S svojim podpisom zagotavljam, da:

- Sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Saše Divjaka,
- So elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- Soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 28. 9. 2011 Podpis avtorja:

ZAHVALA

Zahvaljujem se mentorju na Fakulteti za računalništvo in informatiko v Ljubljani prof. dr. Saši Divjaku za pomoč in podporo pri izdelavi diplomske naloge.

Posebno bi se rad zahvalil podjetju Robomed d.o.o., kjer so mi omogočili delo na zanimivem projektu in mi nudili pomoč in podporo pri diplomski nalogi.

Zahvaliti se želim tudi staršem in ostalim v družini, ki so mi omogočili študij ter me podpirali ves čas študija.

Najlepša hvala vsem!

Kazalo vsebine

Kazalo slik

Kazalo tabel

Kazalo kode

Terminologija

- BIOS Basic Input Output System – skrbi za nalaganje OS
- BSP Board Support Package – paket z gonilniki
- CF Compact Flash – tip bliskovnega pomnilnika
- CPE Centralno Procesna Enota
- GPIO General Purpose Input Output – splošno namenski vhodi in izhodi
- LCD Liquid Crystal Display – displej s tekočimi kristali
- LVDS Low Voltage Differential Signaling – nizko napetostni digitalni signal
- OS Operacijski Sistem
- OS Image slika operacijskega sistema
- USB Universal Serial Bus – univerzalno serijsko vodilo
- WEC Windows Embedded Compact
- WES Windows Embedded Standard
- XAML eXtensible Application Markup Language – označevalni jezik za aplikacije

Terminology

- BIOS Basic Input Output System – it loads the OS
- BSP Board Support Package – driver package
- CF Compact Flash – flash memory
- CPE CPU - Central Processing Unit
- GPIO General Purpose Input Output
- LCD Liquid Crystal Display
- LVDS Low Voltage Differential Signaling
- OS Operating System
- OS Image Image of Operating System
- USB Universal Serial Bus
- WEC Windows Embedded Compact
- WES Windows Embedded Standard
- XAML eXtensible Application Markup Language

POVZETEK

Cilj sledeče diplomske naloge je izdelava vgrajenega (*embedded*) računalnika. Celotna naloga opisuje razvoj vgrajenega računalnika za medicinski, diodni laser DL30, ki se izdeluje v podjetju Robomed d.o.o. V prvem delu, je opisan potek izbora strojne opreme. Tu so v večji meri opisane tehnologije določenih delov, ki so trenutno dobavljive na trgu. Pri vsakem podsklopu je tudi navedeno katero tehnologijo smo izbrali in utemeljeno zakaj. Drug del diplomskega dela, pa najprej predstavi izdelavo operacijskega sistema Windows Embedded CE 6.0R3. Poleg same izdelave slike operacijskega sistema, je opisan tudi postopek povezovanja ciljnega in razvojnega računalnika, pred in po namestitvi operacijskega sistema. Nazadnje je opisan še razvoj nekaterih podsklopov namenskega programa v tehnologiji Silverlight for Windows Embedded.

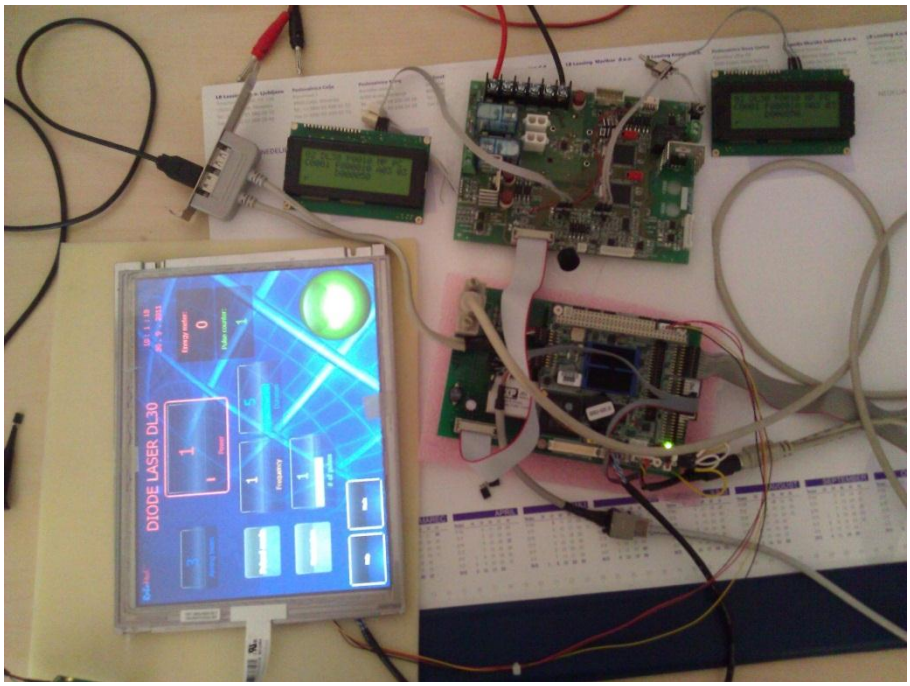
ABSTRACT

The goal of this thesis is to describe a production of an embedded computer. The thesis describes development and production of an embedded computer for the medical diode laser DL30 that is being developed in Robomed d.o.o.. The first part of the thesis describes the choice of hardware devices. I mostly describe the technologies that one can buy on the market. Moreover for every part of the computer installed and developed there is an argument why we selected that exact part. The second part of the thesis describes how to make and deploy an image of the Windows Embedded CE 6.0R3 operating system. Later we find how to connect target computer to the development computer. At the end one can find the development of a few parts of the dedicated application written in Silverlight for Windows Embedded.

1. UVOD

V podjetju Robomed d.o.o. smo se odločili razviti medicinski, diodni laser. Sistem je krmiljen z mikroprocesorjem Atmel ATMEGA128L. Ker želimo napravo izdelati uporabniku prijazno, smo vključili zaslon občutljiv na dotik, preko katerega uporabnik lahko krmili laser. Zaradi enostavnejše izdelave uporabniškega vmesnika smo mikroprocesorju dodali še računalnik arhitekture x86, podjetja Advantech PCM-3343F, ki je povezan z zaslonom občutljivim na dotik. Na sliki 1 vidimo razvojni sistem.

Moja naloga je izdelava vgrajenega računalnik. Najprej sem izbral primerne tehnologije strojne opreme potrebne za izdelavo sistema in poiskal dobavitelje ustrezne strojne opreme na trgu ter pridobil njihove ponudbe. Poiskali smo najboljšo ponudbo. Nato sem na izbran računalnik naložil operacijski sistem Windows Embedded CE 6.0 R3. Ko sem imel delujoč sistem, sem pričel z razvojem aplikacije, s katero rokuje uporabnik pri nastavljanju parametrov diodnega laserja. Aplikacija je razvita do te mere, da lahko krmilimo laser. V načrtu so še določeni dodatki, ki bodo uporabniku olajšali delo z diodnim laserjem, kot so spominske enote za parametre. S tem bo uporabnik lahko pogosto uporabljane nastavitve shranil in jih priklical iz spomina, da mu ne bo treba vedno nastavljati laserja.



Slika 1: Razvojni sitem: zaslon občutljiv na dotik, računalnik PCM – 3343, vezje z računalnikom ARM in krmilno logiko ter povezovalno vezje.

2. STROJNA OPREMA

2.1. Uvod

Pri zasnovi vgrajenega sistema je nekaj časa potrebno nameniti izbiri strojne opreme. Na odločitev vplivajo predvsem okolje in vplivi iz okolja, kot so temperaturno območje, vlažnost prostora, hrup, ki ga računalnik lahko povzroča, povezovanje z drugimi napravami in podobno. Upoštevati moramo tudi potrebe po vhodno/izhodnih enotah. Za sistem DL30 smo se odločali med tremi računalniki. Prva dva sta bila Kontronov JRexplus-LX in pITX-SP (Slika 2), drugi pa Advantechov PCM-3343 (Slika 3). JRexplus-LX je bil opuščen zaradi težav z dobavljivostjo. Prevladal je Advantechov računalnik in sicer zaradi cene in širšega programa produktov. V tem poglavju si bomo ogledali nekaj lastnosti, ki igrajo pomembno vlogo pri izbiri računalnika za vgrajen sistem.

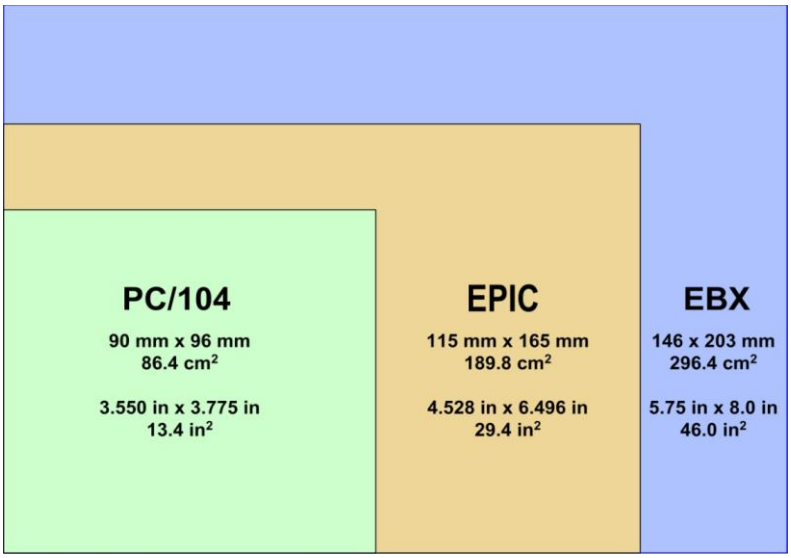
Slika 2: Kontron pITX-SP.

Slika 3: Advantech PCM-3343.

2.2. Formati računalnikov

Računalniki se na prvi pogled najbolj ločijo po velikosti in obliki. Enako tudi velja ko odstranimo ohišje, ki sicer najbolj vpliva na izgled in velikost računalnika. V svetu vgrajenih sistemov bi lahko rekli, da imajo ohišja drugačno vlogo kot v svetu namiznih računalnikov. Namizni računalnik mora biti predvsem lep – prijeten za oko in po možnosti čim tišji. Pri vgrajenih računalnikih na izbiro ohišja vpliva več dejavnikov. Ker je po navadi računalnik le del produkta ni redko, da je brez svojega ohišja in je kar vgrajen v večje ohišje med ostale komponente sistema. Pogosto se uporabnik niti ne zave, da je tudi računalnik del sistema. Kdaj je računalnik vgrajen v svoje ohišje, je odvisno od potreb in zahtev celotnega sistema. Včasih mora biti odporno na primer na vlago in prah. Potemtakem je po navadi tudi brez ventilatorjev, ki skrbijo za prezračevanje sistema in so pogosti pri namiznih računalnikih, kar pa ne velja za vgrajene računalnike. Ker se ohišja pogosto izdelajo namensko za različne sisteme, o njih ne bomo podrobneje govorili.

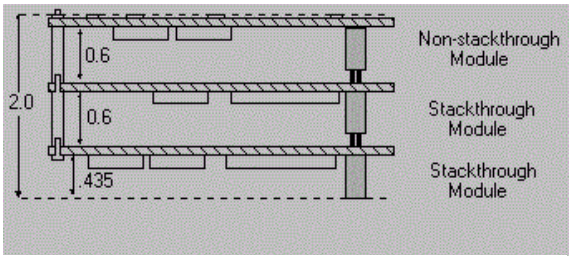
Pri računalnikih je dimenzija računalnika definirana z velikostjo oz. formatom matične plošče. Pri namiznih računalnikih nanjo naknadno vgradimo še centralno procesno enoto, spomin, napajalnik in različne vhodno/izhodne enote. Pri vgrajenih računalnikih procesor, kakor včasih tudi pomnilnik, najpogosteje najdemo že vgrajena na osnovni plošči. Vgrajeni računalniki imajo manj splošno namenskih vhodno/izhodnih vodil. V namiznih računalnikih najpogosteje srečamo računalnike velikosti ATX (Advanced Technology Extended). Ta format določa, velikost računalnika, dimenzij 305 mm x 244 mm, pritrditvene točke na ohišje, postavitev vhodno/izhodnega dela in napajalni del, vključno z napajalnikom samim [1]. Iz tega formata izhaja tudi nekaj drugih formatov npr.: micro-ATX, ki je pogosto uporabljen pri PC računalnikih, ki se uporabljajo za domači kino. Drugi formati so še mini-ITX, nano-ITX, BTX in drugi.



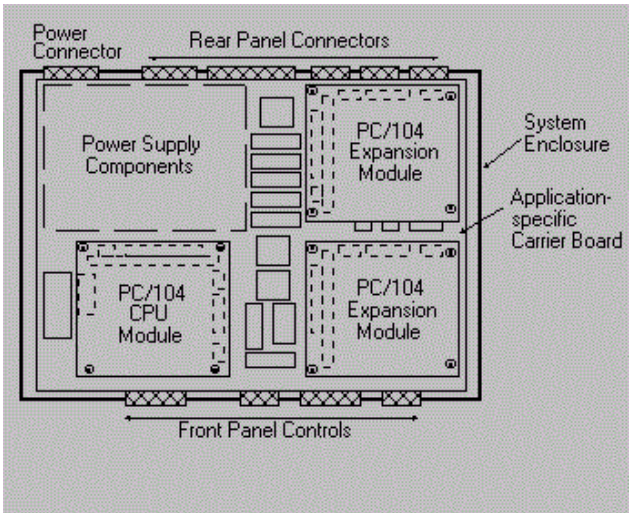
Slika 4: Nekaj velikosti formatov osnovnih plošč.

Pri vgrajenih računalnikih so to pogosto preveliki formati. Mini-ITX in nano-ITX sta še uporabljena, kaj večjega pa redko. Bolj pogosti formati so EPIC, COM Express, pico-ITX, PC/104. Na sliki 4 vidimo primerjavo velikosti med nekaterimi formati. PC/104 označuje poleg formata tudi vodilo za povezavo več modulov. Pri tem ni nujno, da ima vsak modul

CPE enoto, ampak je lahko en modul namenjen le procesiranju grafičnih ukazov, ali pa je namenjen vhodno/izhodnim enotam [2]. Primere povezav vidimo na sliki 5 in 6.



Slika 6: Vertikalna postavitev modulov.



Slika 5: Postavitev modulov drug ob drugem.

Najdemo tudi manjše kot sta Qseven in mobile-ITX. Slednja dva srečamo redkeje. Če ju res ne rabimo imamo pogosto več težav kot koristi, saj so že drugačni konektorji za napajanje, vhodno/izhodne naprave in podobno.

Za naš vgradni sistem smo se odločali med formati PC/104, pITX in pa, za primerjavo dodan ne tako pogost format, JRex (3,5"). Iskali smo čim manjši računalnik, saj smo želeli ohraniti čim manjši sistem. Najmanjši je bil sicer pITX, a to je bil zmogljivejši računalnik z Intel Atom procesorjem, kar pa ni bil pogoj za delovanje našega sistema. Posledično je boljša CPE enota pomenila tudi višji cenovni razred računalnika. Vse tri velikosti so bile še dopustne za sistem. Mere so predstavljene v tabeli 1.

Plošča	**PCM-3343**	**pITX**	**JRex**
Dimenzije	96 mm x 90 mm	100 mm x 72 mm	146 mm x 102 mm
Površina	86,4 mm^2	72 mm^2	151,84 mm^2
Št. Pritrditvenih točk	4	4	6

Tabela 1: Primerjava lastnosti formatov. Kot je že bilo omenjeno je najmanjši format pITX. PC/104 mu je kar precej blizu, JRex pa je že več kot dvakrat večji.

2.3. Procesorji

Za jedro računalnika se pogosto omenja centralno procesno enoto (CPE). Danes nam tržišče ponuja ogromno različnih vrst in tipov CPEjev. Imamo nekaj glavnih segmentov kot so procesorji za strežnike, namizne računalnike, prenosnike, mobilne naprave, vgrajeni procesorji, potem pa so tu še procesorji za super računalnike in še kak segment bi se našel. V tem razdelku bom omenil zgolj del centralno procesnih enot za vgrajene naprave. Ta segment je res precej velik, saj tu najdemo od 4-bitnih procesorjev vse do 32-bitnih, dvojedrnih procesorjev. Podjetje IDC, ki je naredilo analizo trga procesorjev, v poročilu navaja, da bo do leta 2015 na svetu 12,5 bilijonov »pametnih« vgrajenih procesorskih jedr, kar je 6-krat več kot jih bo do tedaj v osebnih računalnikih PC. Med »pametna« procesorska jedra štejejo programabilne sisteme, ki so brezžično povezani in lahko poganjajo visoko nivojski operacijski sistem, kot so pametni televizorji, pametni telefoni, pametni merilniki, pametna mrežna oprema [3].

Če bi želel predstaviti vse CPE enote bi lahko napisal ločeno diplomsko delo. Malo podrobneje bom omenil le Intelov Atom procesor in pa DM&P Vortex86.

Vortex86 centralno procesna enota je na voljo le kot vgradna komponenta in ne kot komponenta, ki bi jo vgradil končni uporabnik. Vortex86 arhitektura je združljiva z x86 arhitekturo. Trenutno so na voljo tri različice 32-bitnih procesorjev : Vortex86SX, VDX in VMX. Prva različica Vortex86 ni imela FPU (Floating Point Unit – enote za računanje s plavajočo vejico). Kljub temu programi, ki delujejo na Linuxih delujejo tudi na Vortex86 procesorjih, ker Linux emulira FPU, kar pa seveda pomeni precej počasnejše izvajanje. VSX in VDX, ki je vgrajen v naš sistem PC-3343F, imata vgrajen severni in južni most, 5 serijskih vrat, USB 2.0, IDE vrata, 10/100M Ethernet, SPI, I2C, PWM, GPIO in JTAG vrata v en sam BGA čip velikosti 27 mm x 27 mm. Tudi BIOS je vgrajen v samem čipu. Temperaturno območje delovanja obeh je od -40°C do +85°C, kar je idealno za industrijske razmere. Ta arhitektura pri hitrosti 800 MHz porabi le 2.02W energije.

Vortex86MX deluje s taktom 1GHz in ima jedro precej podobno VDX. Ta verzija CPE enote ima vgrajena tudi kontrolerja za avdio in video. Znan je tudi pod imenom PMX-1000 [4, 5].

Drug procesor prihaja iz podjetja Intel. Intel je največji proizvajalec procesorjev za PCje na svetu. Močno je prisoten tudi na trgu vgrajenih naprav. Trenuteno zelo popularen procesor za mobilne naprave pri Intelu, je Atom. Intel ponuja v vgradni različici tudi modele znane iz trga prenosnih in namiznih računalnikov kot so i3, i5 in i7.

Intel Atom je na voljo samo kot vgradni procesor in ni na voljo končnim kupcem. Ta procesor je uporabljan v majhnih prenosnikih, tabličnih računalnikih, hibridnih napravah, tankih odjemalcih, pametnih telefonih, in osnovnih namiznih računalnikih. Atom procesor je ponujen v več serijah s porabo energije od manj kot 1 W, do preko 10 W. Predstavljen je bil leta 2008, sprva le kot 32-bitni procesor x86 arhitekture, sedaj pa je na voljo tudi podpora 64-bitnim ukazom. Problem prve generacije je bil ta, da je porabo energije procesorju kvaril chipset, saj je bil pogosto uporabljen Intel 945GSE Express, ki je večji in energijsko bolj potraten. V tem primeru je imel CPE porabo energije 2,5 W in pa chipset kar 11 W. Druga verzija procesorjev Atom ima vgrajen tudi grafični procesor. Skupna poraba energije je od 7 pa do 12 W, kar je tudi do 40 odstotno izboljšanje. Trenutna verzija (Pineview) ima vgrajen procesor za grafiko in tudi kontroler za pomnilnik, pri čemer so s treh čipov prešli na dva.

V tabeli 2 si lahko ogledate primerjavo med obema procesorjema. Podatke za tabelo sem našel na spletu [6]. Iz tabele je razvidno, da je Atom procesor precej boljši kot DM&Pjev.

	Intel Atom N270	DM&P Vortex86
Numerično urejanje	265.76	152.6
Urejanje stringa	38.667	8.4981
Emulacija plavajoče vejice	36.356	23.335
Fourier	4062.5	1869.6
Prirejanje	6.3417	1.711
Nevronske mreže	3.8447	0.81116
LU dekompozicija	206	27.335

Tabela 2: Primerjava Procesorjev Atom in Vortex86. Vsa števila so v iteracijah / sekundo.

Pri sistemu DL30 računalnik služi le za komunikacijo z uporabnikom preko zaslona občutljivega na dotik. Znotraj sistema komunicira še z ARM računalnikom. Ker ne opravlja zahtevnih operacij, je že manj zmogljiv računalnik dovolj zmogljiv za naše potrebe.

2.4. Vodila za video

Najzanimivejši del računalnika je pogosto ekran, saj slika pove več kot tisoč besed. Še zlasti v zadnjem času ko prihajajo vedno večji ekrani, televizorji z boljšo resolucijo, 3D tehnologija in podobno. A kljub vsemu, med vgrajenimi napravami še vedno najdemo precej 'head-less' naprav. S tem izrazom opisujemo naprave, ki nimajo ekrana. Tovrstne naprave so na primer industrijski kontrolerji, kjer se program naloži kar preko omrežja in nato stroj deluje sam. Na drugi strani, na primer, pa že imamo pametni telefon, ki premore četrtino visokoločljivostnega (QHD) ekrana s tehnologijo 3D, brez posebnih očal [7].

Do nedavnega je bilo najpogostejše vodilo za video pri PC računalnikih VGA (Video Graphic Array). To tehnologijo je predstavil IBM že leta 1987. VGA je najbolj razširjen v svetu PC računalnikov, saj se pričakuje, da ga ima vsak računalnik.

Pri vgrajenih računalnikih so bili najprej prisotni le vrstični ekrani, kjer so bili snovalci uporabniških vmesnikov precej omejeni, saj so omogočali le izpis alfanumeričnih znakov. Ti ekrani so bili povezani preko serijskega ali paralelnega vodila, kot ostale vhodno/izhodne naprave.

Kasneje so se pojavili grafični ekrani, ki so povezani preko TTL povezave. TTL (Tranzistor-tranzistor logic) označuje digitalno vezje, ki v tem primeru skrbi za vhodne signale v monitor. Po linijah se prenaša urin signal pikslov, horizontalna in vertikalna sinhronizacija in digitalna vrednost rdeče, zelene in modre barve paralelno.

S to povezavo je bilo kar nekaj težav, zato smo se odločili, da uporabimo monitor z LVDS (Low Voltage Differential Signaling – digitalni signal z nizko napetostjo) povezavo, ki so vse pogostejši. LVDS je serijsko vodilo. Pri LVDSu se prenese 7 bitov na urin signal po eni podatkovni liniji. Osnovni ekrani podpirajo 18-bitov na linijo, boljši pa imajo še četrto podatkovno linijo in podpirajo do 24-bitov na točko. S tem dobimo sliko kvalitete truecolor. Linija je odpornejša na motnje in zato primerna za daljše in hitre komunikacije [8].

Proizvajalci bodo kmalu zamejali vodilo LVDS z vgrajenim in notranjim vodilom DisplayPort. Intel načrtuje, da bo v svojih produktih to uvedel že do leta 2013 [9].

2.5. Vodila za druge vhodno / izhodne enote

Vgrajeni računalnik je pogosto povezan z več zunanjimi napravami. Naprava je lahko senzor, aktuator ali pa tudi drugi računalnik.

Med vgrajenimi računalniki je pogosto serijsko vodilo RS232, znano iz sveta osebnih računalnikov. Danes ga med osebnimi računalniki srečamo že zelo redko. Nadomestilo ga je vodilo USB (Universal Serial Bus). Drugo dokaj pogosto vodilo je Philipsov I^2C (Inter-Integrated Circuit). Intel je kasneje definiral SMBus, ki je izpeljanka I^2C, ima pa strožje definirane protokole. I^2C uporablja dve dvosmerni liniji SDA (Serial Data Line – serijska podatkovna linija) in pa SCL (Serial Clock – serijska ura). Uporablja 7-bitni naslovni prostor s 16 rezerviranimi naslovi, tako da je možnih 112 naprav na enem vodilu. Standardna hitrost je 100 kbit/s, možen pa je tudi način 10 kbit/s [10, 11].

Omenil bom še vodilo SPI (Serial Peripheral Interface Bus – serijski vmesnik za naprave). To sinhrono serijsko vodilo uporablja štiri linije SCLK (Serial Clock – serijska ura), MOSI (Master Output, Slave Input), MISO (Master Input, Slave Output) in SS (Slave Select). Vodilo ni omejeno na prenos 8-bitnih besed. Prenos je sicer hitrejši kot pri I^2C ali SMBus, potrebnih pa je več linij.

Pri daljših razdaljah je pogosto uporabljen CAN (Controller-Area Network – omrežje na območju kontrolerja). To vodilo je bilo narejeno za uporabo v avtomobilu, kjer mikrokontrolerji in naprave komunicirajo med seboj brez nadzornega računalnika. Danes se uporablja tudi v medicinskih napravah in pa pri avtomatizaciji industrijskih naprav. Vsaka naprava mora imeti nadzorni procesor, CAN kontroler in pa sprejemnik/oddajnik. Obstaja tudi različica CANOpen [12].

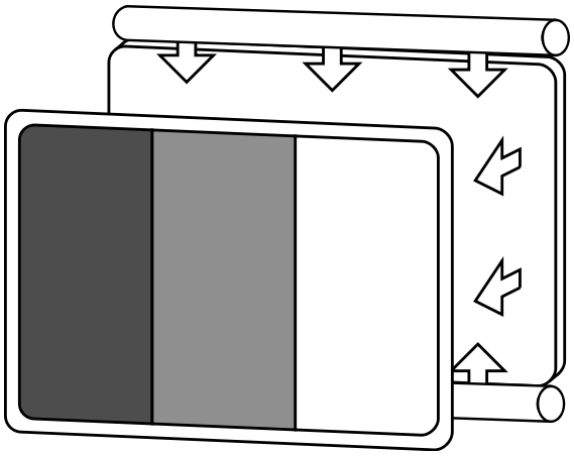
V sistemu DL30 sta računalnika povezana preko vodila RS232, preko katerega prenašata podatke. Poleg tega smo se odločili še za uporabo GPIO vodila (General Purpose Input/Output – splošno namensko vhodno izhodno vodilo). Oba računalnika med katerima smo se odločali imata GPIO vodilo. Advantechov računalnik ima 8 vhodno/izhodnih linij, Kontronov pa 4. V našem sistemu smo uporabili štiri. Te štiri linije imajo točno določene vloge. Ena od linij je na primer aktivna, kadar je katera od naprav v stanju napake.

2.6. Ekrani

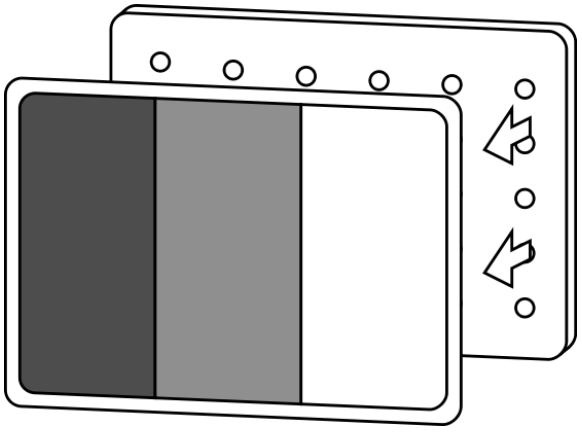
Že v prejšnjem poglavju smo govorili o ekranih. Izbira računalnika je povezana tudi z izbiro ekrana. Računalnik in ekran morata imeti podporo za isto vodilo. Če temu ni tako moramo uporabiti še ustrezen pretvornik. Pretvornik je seveda rešitev, a to pomeni, da je končni produkt toliko večji in dražji. To pa seveda ni naš cilj, zato moramo opremo izbirati nekako vzporedno, da je med seboj kompatibilna. Ekranov poznamo več vrst. Danes so še vedno najpopularnejši LCDji. V mobilnih napravah srečujemo že SLCD in pa OLED ekrane. Ti so energijsko varčnejši, kar je zelo dobrodošlo, saj se naprave pogosto napajajo z baterije. Druga prednost je vidljivost na soncu. LCD ekran je pogosto slabo viden če je pod vplivom direktne svetlobe. Pri naši aplikaciji ti dve lastnosti nista bili tako pomembni, saj se bo DL30 uporabljal v prostorih, napajal pa se bo z omrežja, vseeno pa je dobrodošlo, da je naprava čim bolj energijsko varčna.

Najprej smo definirali velikost in resolucijo ekrana. Odločili smo se za 10,4 inčno velikost, ločljivosti 800 x 600 pik. To je standardna rešitev, ki je enostavno dobavljiva. Za vodilo za povezavo z računalnikom smo izbrali LVDS. To vodilo je po našem mnenju najprimernejše, saj je podprto v veliko ekranih in tudi računalnikih. Več O LVDS vodilu je napisano v poglavju 2.4.

Naslednja lastnost LCD ekranov je osvetlitev. Včasih je prevladovala CCFL tehnologija, danes pa se vse bolj uveljavljajo LED diode. CCFL (Cold Cathode Fluorescent Lamp) osvetlitev je bila cenovno ugodna in se veliko uporablja v elektronskih napravah. Njena prva slabost je potreba po visoki napetosti, ki je potrebna za delovanje, zaradi česar potrebujemo dodaten pretvornik. Posledica je seveda tudi visoka poraba energije. Druga slabost je, da se takšna osvetlitev tudi precej segreje. CCFLev se ne sme preveč ohladiti, saj jim tako skrajšamo življenjsko dobo. Naslednjo razliko vidimo na sliki 7 in sliki 8, ki kaže postavitev svetilnih objektov. Torej imamo prednosti LED diod pri osvetlitvi ekranov kar nekaj: nižja poraba energije, hladnejše delovanje, ni potrebe po dodatnih pretvornikih. Zaradi vsega naštetega smo se odločili izbrati ekran z LED osvetlitvijo [13].



Slika 7: Osvetlitev CCFL.



Slika 8 : Osvetlitev LED.

Poleg tega za našo aplikacijo potrebujemo še občutljivost na dotik. Zasloni z občutljivostjo na dotik so dandanes vse popularnejši. To se še posebej odraža na trgu mobilnih naprav in zabavne elektronike, v avtomobilih ipd. Poznamo kar nekaj tehnologij zaslonov občutljivih na dotik. Tukaj bodo omenjene bodo le tiste, med katerimi smo se odločali in so verjetno tudi

najpogostejše. Prva za katero se nismo odločili je bila občutljivost zaznana z IR diodami. Ob strani imamo linijo IR diod in sprejemnikov. Če je vmes kašna ovira (prst, rokavica, svinčnik itd.) kontroler to zazna in računalniku pošlje podatke o dotiku. Ta tehnologija ima prednosti, visoko prosojnost, zazna različne objekte, kar pa je obenem tudi slabost, saj zazna že muho ali kapljico vode. Ker pa elektromagnetno sevanje vpliva na njeno delovanje, smo jo izključili iz izbora.

Druga je bila SAW (Surface Acoustic Wave) ali tehnologija na podlagi površinskih akustičnih valov. Tehnologija bazira na dveh oddajnikih in sprejemnikih za os X in Y. Kontroler pošlje signal oddajniku, ki ga pretvori v ultrazvočne valove in jih pošle po panelu. Valovi se odbijejo od roba in gredo nazaj do sprejemnika. Če jih prekine prst potem to sprejemnik zazna. Prednost je ta, da je panel samo steklo, kar pomeni zelo majhno poslabšanje slike in dolgo obstojnost. Seveda pa ima ta tehnologija tudi slabosti. Ni občutljiva na trde predmete kot konica svinčnika, lahko pa jo zmoti prah ali voda. Pri DL30 obstaja možnost izpostavljenosti kakšnim kapljicam tekočin zaradi česar se nismo odločili za to tehnologijo.

Zadnji tip občutljivega zaslona, ki ga bomo omenili in ki ni bil izbran, je kapacitiven. Kapacitivni zasloni so vse bolj uporabljeni v mobilnih telefonih saj zelo dobro in natančno zaznajo pritisk s prstom. Tu je panel sestavljen iz dveh plasti elektrod, ki sta povezani na kontroler. Zgornja plast vsebuje vertikalne elektrode, spodnja pa horizontalne. Kontroler meri medsebojno kapacitivnost obeh plasti. Ko se zaslona dotaknemo s prstom se nanj prenese nekaj napetosti in se spremeni potencial med plastema. Kontroler to zazna in pošlje podatke o X in Y poziciji računalniku. Največja ovira za nas je, da kapacitivni zaslon ne zazna pritiska z roko v rokavicah.

Trenutno še vedno precej pogosta tehnologija je rezistivni (uporovni) zaslon občutljiv na dotik. Tudi pri tem imamo dve plasti izdelani iz prozornega uporovnega materiala (ITO – Indium Tin Oxide). Med plastema so prozorne neprevodne točke, ki ju ločujejo. V plasteh so vodila, po katerih potuje signal kjer je bila točka pritiska. Zgornja plast ima na zunanji strani obstojni sloj, na notranji pa prevodni sloj. Ko pritisnemo se prevodni sloj dotakne sloja na steklu pri čemer se ustvari električni stik. To zazna kontroler in pošlje računalniku podatke o dotiku. Slabost te tehnologije je le 75% prepustnost svetlobe in pa možnost poškodbe z ostrimi predmeti. Prednosti sta neobčutljivost na prah, vodo in podobne dejavnike in pa cena. Zaradi teh prednosti je bil izbran uporovni zaslon občutljiv na dotik [14].

2.7. Pomnilni mediji

V računalniku, kot pomnilni medij, najpogosteje srečamo trdi disk. Danes se manjše prenosne naprave, kot so digitalni fotoaparati in GPS naprave, mp3 predvajalniki, mobilni telefoni in podobno, že skoraj vedno poslužujejo bliskovnega pomnilnika. Tudi vedno več vgrajenih naprav uporablja bliskovni pomnilnik. Največja problema bliskovnega pomnilnika sta cena in omejeno število brisanj/pisanj. A pri manjših količinah pomnilnika je cena že popolnoma sprejemljiva. Ta problem ostaja le še pri količini nad 128 GB pomnilnika. Prednosti bliskovnega pomnilnika so neobčutljivost na tresljaje, manjša poraba energije, ni gibljivih delov in velikost.

Vedno popularnejši so SSD diski (Solid State Drive). Ti so ravno tako bliskovni pomnilniki večje kapacitete, ki pa imajo dodan kontroler.

Ker je DL30 majhen in dokaj preprost sistem smo se odločili za bliskovni pomnilnik. Trenuten razvoj poteka s kartico tipa CF (Compact Flash) velikosti 4 GB (Slika 9). Ker pri tem pomnilniku ni gibljivih delov, to pomeni neslišno delovanje. Velika prednost bliskovnega pomnilnika je tudi hitrost delovanja. Trdi diski, magnetni trakovi in optični mediji imajo dokaj visok čas dostopa do podatkov. Pri bliskovnih pomnilnikih je ta čas precej manjši. Res je, da je bliskovni pomnilnik nizkega cenovnega razreda počasnejši, a z višjo ceno pridemo do pomnilnika, ki je precej hitrejši kot trdi diski. Veliko vgrajenih računalnikov ima že vgrajen bralnik bliskovnih pomnilnikov. Tako ima pITX vgrajen čitalnik za kartice tipa SDHC (Secure Digital High Capacity), kar pomeni da bere tudi SD kartice velikosti nad 2 GB. PC-3343F pa ima vgrajen bralnik bliskovnih kartic tipa CF.

Slika 9: Kartica CF, proizvajalca Transcend, ki se uporablja pri razvoju prototipa.

2.8. Zaključek

V tem poglavju smo si ogledali nekaj pomembnejših lastnosti vgrajenih računalnikov. Med seboj smo tudi primerjali dva računalnika pITX in pa PC-3343F. Predvsem zaradi cene in širšega izbora dodatne opreme, smo se odločili za PC-3343F računalnik podjetja Advantech. V tabeli 3 vidimo primerjavo med računalnikoma.

	Kontron pITX	**Advantech 3343F**
CPU:	Atom Z530 (1.6 GHz)	DM&P Vortex86DX (1 GHz)
Chipset:	Intel US15W	DM&P Vortex86DX (1 GHz)
Grafika:	DVI + LVDS	VGA + LVDS + TTL LCD
USB 2.0:	6+1 (host)	4
LAN:	Gigabit LAN	10/100 Mbps
Flash čitalec:	micro SDHC	CF Type I/II
Dimenzije:	100 x 72 mm - Pico ITX	96 x 90 mm - PC/104

Tabela 3: Primerjava računalnikov pITX in 3343F.

Tu bi omenil, da oba proizvajalca nudita podporo operacijskemu sistemu Windows Embedded CE 6.0, ki bo imel glavno vlogo v naslednjem poglavju. Kot bomo videli se uporablja paket gonilnikov BSP (Board Support Package).

3. OPERACIJSKI SISTEM

3.1. Uvod

Sama strojna oprema računalnika je zgolj en del, ki za končnega uporabnika niti ni tako zanimiv. Računalnik dobi uporabnost šele s primerno programsko opremo. Za sistem DL30 smo se odločili uporabiti vgrajeni operacijski sistem.

Najprej smo pomislili na odprtokodni Linux. Njegova najopaznejša prednost je brezplačna uporaba licence kakor tudi programskih orodij. Vendar ima pomanjkljivost, ki se pogosto izkaže za ključno, pomanjkanje podpore. Iz tega razloga smo se odločili za operacijski sistem podjetja Microsoft. Microsoftovo ponudbo lahko razdelimo na dva sklopa operacijskih sistemov za vgrajene računalnike. V prvem, WES (Windows Embedded Standard), imamo dva sistema, ki bazirata na operacijskih sistemih znanih iz sveta namiznih računalnikov. Prvi je Windows Embedded Standard 2009 (modularni XP), drugi pa je Windows Embedded Standard 7, ki je modularna različica novejšega operacijskega sistema Windows 7. Glavna razlika med njima je velikost jedra, ki pri prvem znaša od približno 300 MB, pri drugem pa od 2 GB naprej. Do takšne razlike pride predvsem zato, ker je pri slednjem v samo jedro vključenih več komponent. Pri operacijskem sistemu WES7 so predpostavili, da bo vsaka naprava imela ekran in bo povezana v neko omrežje [15].

Drug sklop vgrajenih Microsoftovih operacijskih sistemov pa je Windows CE, ali z zadnjo različico preimenovan WEC (Windows Embedded Compact). Različica, ki je bila na voljo ko smo se odločali je bila Windows Embedded CE 6.0 R3. Danes je na voljo že Windows Embedded Compact 7. Operacijski sistemi v tem sklopu nimajo z različicami za namizne računalnike nič skupnega. Raziskovalec in namizje spominjata na Windows 2000 in prejšnje različice. Prva verzija operacijskega sistema je bila razvita leta 1996 in se je imenovala Windows CE 1.0. Bila je razvita popolnoma na novo in neodvisno od različic za namizne računalnike.

3.2. Windows Embedded CE 6.0 R3

Windows Embedded CE (Slika 10) (z zadnjo, sedmo različico preimenovan v Windows Embedded Compact), je komponentni operacijski sistem namenjen vgrajenim računalnikom, kar predstavlja precej širok spekter produktov – od naprave za brskanje po spletu, kot je tablični računalnik, ročni skenerji v skladiščih, industrijski kontrolerji, navigacijske naprave, kioski, multimedijski strežniki, prenosni predvajalniki, thin clienti itd. Windows Embedded CE je v javnosti pogosto pomešan z Windows Mobile, ki pa je zgolj izvedenka operacijskega sistema Windows Embedded CE. Večkrat sem naletel na vprašanje kaj pomeni CE v imenu. To vprašanje se je zastavljalo tudi v javnosti, odgovor nanj pa je: nič. Dejansko ne skrajšuje ničesar, namiguje pa na: Compact, Connectable, Compatible, Efficient. Trenutno je operacijski sistem na voljo le za 32-bitne procesorje. Na zadnji različici WEC7 bazira tudi nov Microsoftov operacijski sistem za mobilne telefone Windows Phone 7.

Slika 10: Logo operacijskega sistema Windows Embedded CE 6.0

Glavne odlike operacijskega sistema Windows Embedded CE, prisotne že od prve različice so:

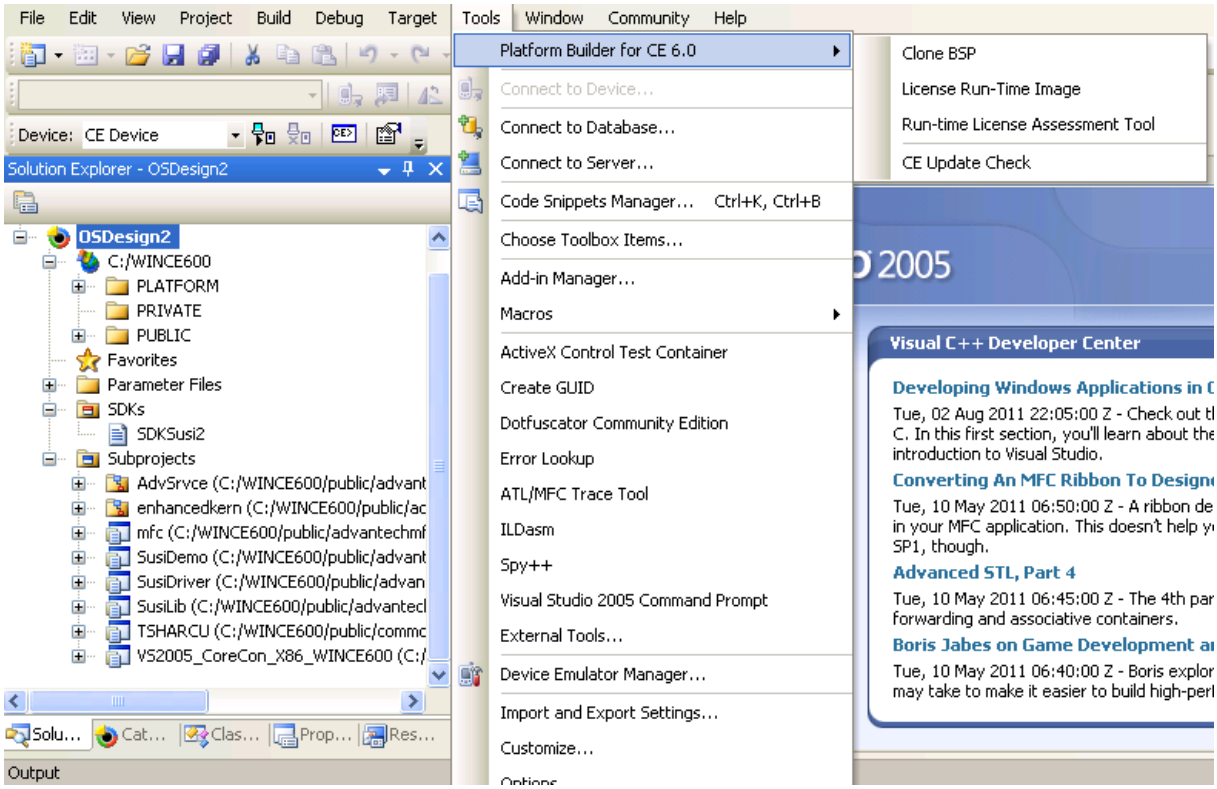
- Majhna velikost slike operacijskega sistema (že od manj kot 350 KB)
- Modularna arhitektura
- Podpora aplikacijam v realnem času
- Delovanje na velikem obsegu procesorjev
- Dober nadzor nad porabo energije
- Dobra razvojna orodja
- Dobra orodja za testiranje in razhroščevanje sistema

V prejšnjih različicah je operacijski sistem nudil podporo le 32 procesom hkrati. Vsak je lahko naslovil 32 MB navideznega pomnilnika. Ti dve omejitvi sta bili v šesti različici odstranjeni. Sedaj lahko sočasno teče kar 32,000 procesov in vsak lahko naslovi kar 2 GB navideznega pomnilnika.

Morda najopaznejša razlika v primerjavi z namiznimi operacijskimi sistemi je modularna arhitektura. Tu ni namestitvenega medija pač pa kar programski paket, ki ga naložimo na razvojni računalnik, kjer nato sami sestavimo operacijski sistem. Programski paket je dodatek razvojnemu programu Visual Studio 2005.

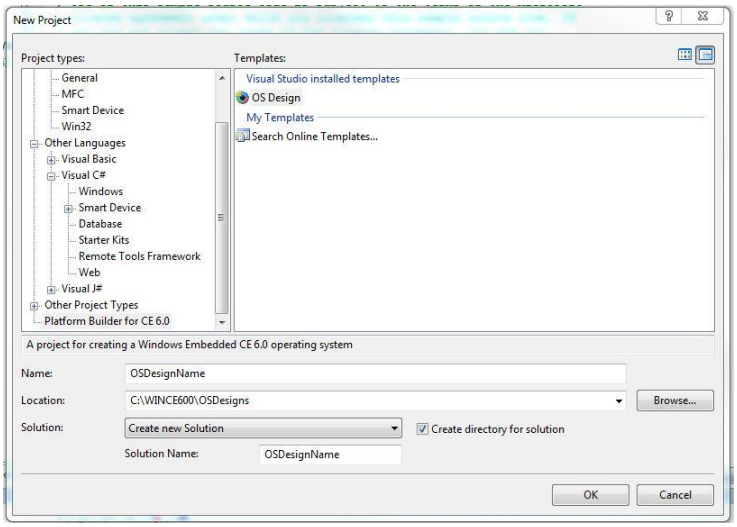
3.3. Izdelava slike operacijskega sistema

Razvojno orodje, ki ga posnamemo kot dodatek v Visual Studio 2005 se imenuje »Platform Builder«. Pomembno je, da Platform Builder pravilno namestimo. Tu sem se sam že srečal s prvim problemom. Platform Builder sem namreč namestil na operacijski sistem Windows 7. Windows Embedded CE 6.0 je bil izdan že pred več kot 3 leti, ko še ni bilo na trgu Windows 7. Problem se je pojavil, ko sem želel vključiti v operacijski sistem še svoj podprojekt. Ta nikakor ni bil v sliki operacijskega sistema in sem porabil kar nekaj časa, da sem ugotovil, kje je napaka, saj ni bilo javljene nobene napake in sem sprva mislil, da sem napačno postopal pri kakem koraku. Kasneje sem z brskanjem po spletnih klepetalnicah spoznal, da se s podobnimi problemi srečujejo tudi drugi, zato sem se odločil, da na razvojni računalnik namestim operacijski sistem Windows XP. Potem sem namestil še Visual Studio 2005 in vse popravke, nato pa še Platform Builder za Windows Embedded CE 6.0 in na koncu še nadgradnjo na 6.0 R3. Ko sem pognal VS2005 in odprl projekt sem dobil okno kot ga vidimo na sliki 11.

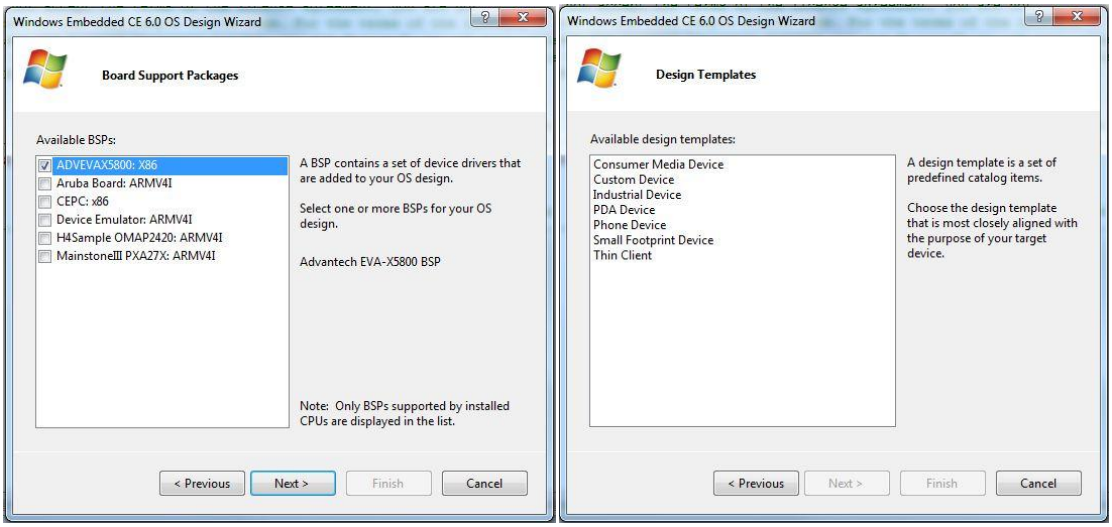


Slika 11: Visual Studio 2005 z nameščenim Platform Builderjem.

Postopek izdelave slike se začne z novim projektom v Visual Studiu (Slika 12). V sklopu odpiranja novega projekta izberemo željeno platformo (BSP) in pa predlogo operacijskega sistema (Slika 13).

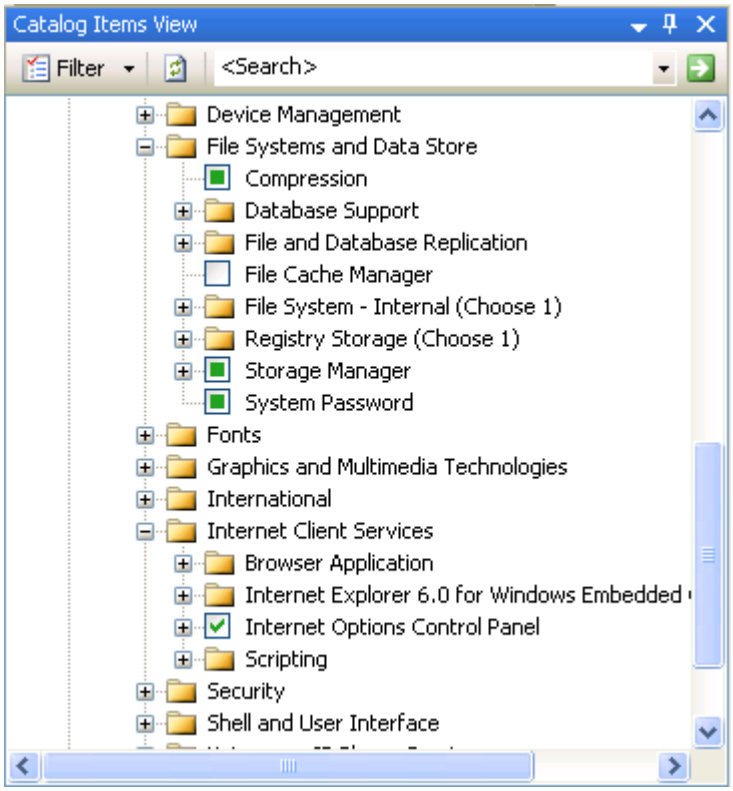


Slika 12: Nov projekt.



Slika 13: Izbira BSP (levo) in predloge (desno).

BSP (Board Support Package) je dejansko skupek gonilnikov in konfiguracijskih datotek, ter sloj, ki skrbi za povezavo med strojno opremo in operacijskim sistemom (OAL – OEM Adaptation Layer) za določeni računalnik. Tu lahko razvijemo tudi svojega, če izdelamo lastni računalnik. Ta korak se po navadi prične tako, da skopiramo že izdelan BSP za čim bolj podobno konfiguracijo. V podjetju smo računalnik kupili in oba proizvajalca prej opisanih računalnikov sta nam ponudila svoj že razvit BSP, tako da izkušenj z razvojem lastnega paketa še nimam [16]. Ko izberemo BSP nam čarovnik ponudi na izbiro še nekaj že izdelanih predlog. Predloge vsebujejo izbrane komponente za pogoste aplikacije. Predlogo si lahko malo prilagodimo že v čarovniku, seveda pa je prilagoditev možna tudi kasneje v pogledu 'Catalog Items View' (Slika 14), kjer izbiramo med vsemi nameščenimi komponentami, gonilniki, aplikacijami, uporabniškimi vmesniki in komponentami, katere so razvili drugi razvijalci. Na voljo imamo preko 700 komponent. Poleg že vgrajenih si lahko namestimo tudi komponente drugih razvijalcev, ali pa izdelamo svoje lastne. Zaradi tako velike količine komponent je zelo koristno iskalno okno kamor vpišemo iskalni niz. Tako lahko po vrsti pregledamo vse zadete komponente, ki ustrezajo naši poizvedbi.

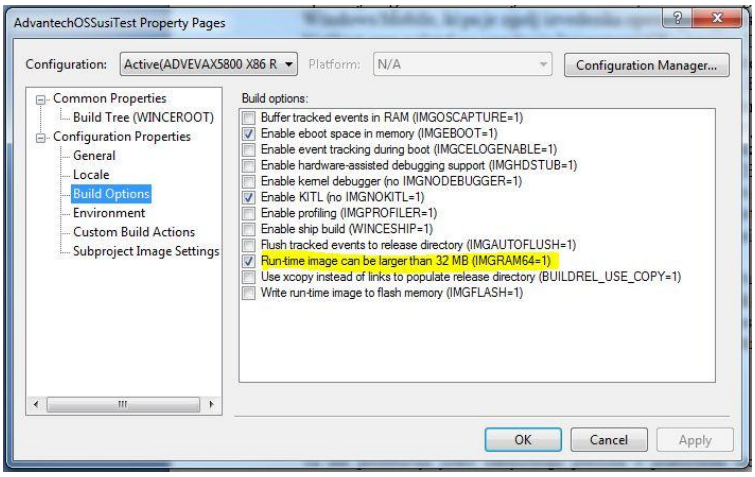


Slika 14: Katalog komponent – 'Catalog Items View'.

Med drugimi bi omenil komponento SYSGEN_XAML_RUNTIME, s katero omogočimo poganjanje aplikacij, ki uporabljajo tehnologijo Silverlight for Windows Embedded. Za demonstracijo so pri Microsoftu vključili tudi Internet Explorer z uporabniškim vmesnikom v XAML tehnologiji (komponenta SYSGEN_IESAMPLE_EXR). Z vključitvijo le tega, Platform Builder sam vključi še XAML pogon, ki je potreben za poganjanje te aplikacije. Tako vidimo, da Platform Builder, sam vključi komponente, ki so potrebne za delovanje drugih, ki jih vključi razvijalec, tako da dobimo delujoč operacijski sistem in ni težav še z medsebojno odvisnostjo komponent.

Nadalje velja omeniti še tako imenovane okoljske spremenljivke (Environment Variables). Spremenljivke delimo na več kategorij kot so BSP spremenljivke, s katerimi nastavljamo parametre BSP paketa, BSP_NO spremenljivke, s katerimi lahko iz slike odstranimo vse komponente določene kategorije (na primer za audio, če naš sistem ne podpira zvoka), IMG spremenljivke, ki umaknejo komponente registra pa ne spreminjajo (uporabljajo se med razvojem) in še nekatere druge [17].

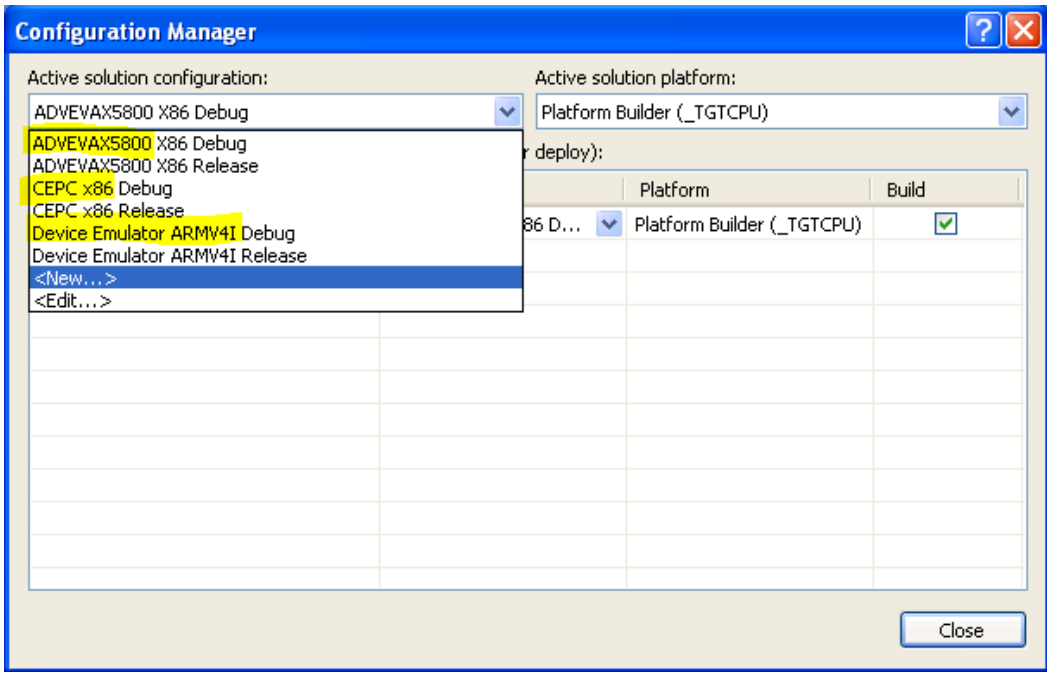
Na sliki 15 lahko vidimo zanimivo spremenljivko IMGRAM64. Če je ta spremenljivka postavljena potem slika operacijskega sistema ne sme preseči velikosti 64 MB. Tu se znova pokaže precejšnja razlika v primerjavi z namiznimi računalniki, kjer sam operacijski sistem dandanes zasede več 1000 MB pomnilnika.



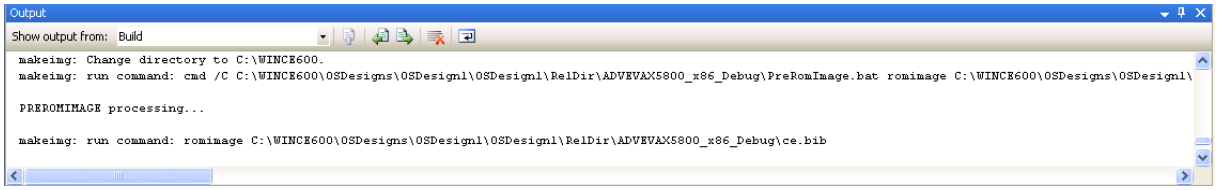
Slika 15: Omejitev velikosti slike na 32 MB.

Ko imamo enkrat vključen BSP (ki nam ga ponudi proizvajalec, ali pa ga izdelamo sami), in vse gonilnike zunanjih naprav (sam sem vključil le še gonilnike za ekran občutljiv na dotik), ter delujočo sliko operacijskega sistema, lahko pričnemo še z vključevanjem lastnih aplikacij, vnosov v registre in nastavljanjem okoljskih spremenljivk. Tu se pogosto vrnemo nazaj in še dodatno optimiziramo sliko operacijskega sistema ter vključimo kake komponente, za katere se pokaže potreba med izdelavo aplikacije.

V primeru, da smo pri prvem koraku v čarovniku izbrali več BSP paketov, torej izdelujemo enako sliko za različne strojne sisteme, lahko izberemo za kateri sistem bomo sliko izdelali, kot je razvidno na sliki 16. Pomembno je, da se zavedamo, da se izdela le slika za trenutno izbran BSP, zato moramo izdelati sliko za vsak BSP posebej. Med procesom same izdelave slike je zelo uporabno okno razhroščevalnika (Slika 17). Tu dobimo rezultat o uspešnosti/neuspešnosti same izdelave slike in, če operacija ni uspešno dokončana, tudi sporočilo o napaki, ki nam je v pomoč med odpravo napake. V primeru napake v korenskem direktoriju (_WINCEROOT) dobimo tudi datoteko Build.err. Če ne pride do napak, imamo le datoteki Build.log, kjer so vsa sporočila razhroščevalnika in pa Build.wrn, kjer so le opozorila.



Slika 16: Izbira platforme za katero bomo izdelali sliko operacijskega sistema. V konkretnem primeru vidimo, da obstaja možnost izbire med različnimi arhitekturami (x86 in ARM v4).



Slika 17: Okno razhroščevalnika, ki prikazuje potek in morebitne napake ter opozorila pri izdelavi slike.

3.4. Prenos slike na ciljni računalnik

Seveda nam sama slika na našem razvojnem računalniku ne služi kaj dosti, če je ne prenesemo na ciljni računalnik. V našem primeru je to PCM-3343F. Windows Embedded CE nam nudi 3 možne načine prenosa: ethernet omrežna povezava, USB ali pa serijski vmesnik RS232. Sam sem se odločil za omrežno povezavo, ker se mi zdi najbolj smiselna in je implementirana na največ računalnikih. Kljub temu, da prenos slike poteka preko omrežja, sem povezal razvojni in ciljni računalnik preko RS232 vrat. Preko teh vrat namreč lahko spremljamo stanje zagonskega programa (loadcepc.exe). Seveda pa se mora zagonski program zagnati nekje na ciljnem računalniku. Tudi tu imamo več opcij, med katerimi sem se odločil za USB pomnilniški medij z naloženim operacijskim sistemom DOS (Disk Operating System) in zagonskim programom loadcepc.exe, ki je del razvojnih orodij za Windows Embedded CE. Nahaja se v sliki za 3,5" disketo (floppy).

Tukaj pa sem se srečal z zanimivim problemom. DOS mi je javil napako v zvezi z datoteko HIMEM.SYS. Po brskanju po spletu sem ugotovil, da je to gonilnik, ki omogoča dostop do pomnilniških naslovov preko 1 MB in naloži jedro DOS-a v HMA (High Memory Area). Intel 8088 in Intel 8086 procesorja sta imela le 20 naslovnih linij, zato nista mogla dostopati do višjih naslovov kot 1 MB. Kasneje so dodali logična vrata (A20), ki so odklopila

mikroprocesorjev enaindvajseti pin, za združljivost s starejšimi programi. Vrata so bila dinamično krmiljena, tako da so se lahko izvajali programi za starejše procesorje, prav tako pa so lahko dostopali do več pomnilnika, z novimi programi [18]. Končno sem ugotovil, da je bil problem pri testnem računalniku, saj tega ni imel implementiranega v svojem BIOSu (Slika 18). Računalnik sem vrnil prodajalcu, ki ga je poslal v Kontron v Nemčijo, kjer so nadgradili BIOS, potem pa so mi ga znova vrnili. Po nadgradnji sem lahko zagnal sistem z USB pomnilnika v DOS.

19 BIOS/CPLD Changes

Some BIOS options are only available with a corresponding CPLD version (see BIOS Setup entry Main/Board Information). The following table shows an overview:

Option	CPLD Version
ACPI Suspend to RAM (S3)	0x0C
Gate A20 Control	0x0E
Power LED Blinking during S3	
Wake On USB	0x0F
Enable/Disable Wake On LAN	0x10
GPIO Control (IRQ Input, Tri-State Output)	0x11
Windows® Automatic Thermal Control	
Power LED Blinking during Recovery	

Slika 18: Prikaz navodil, da nekatere verzije nimajo implementiranih A20 vrat na procesorju.

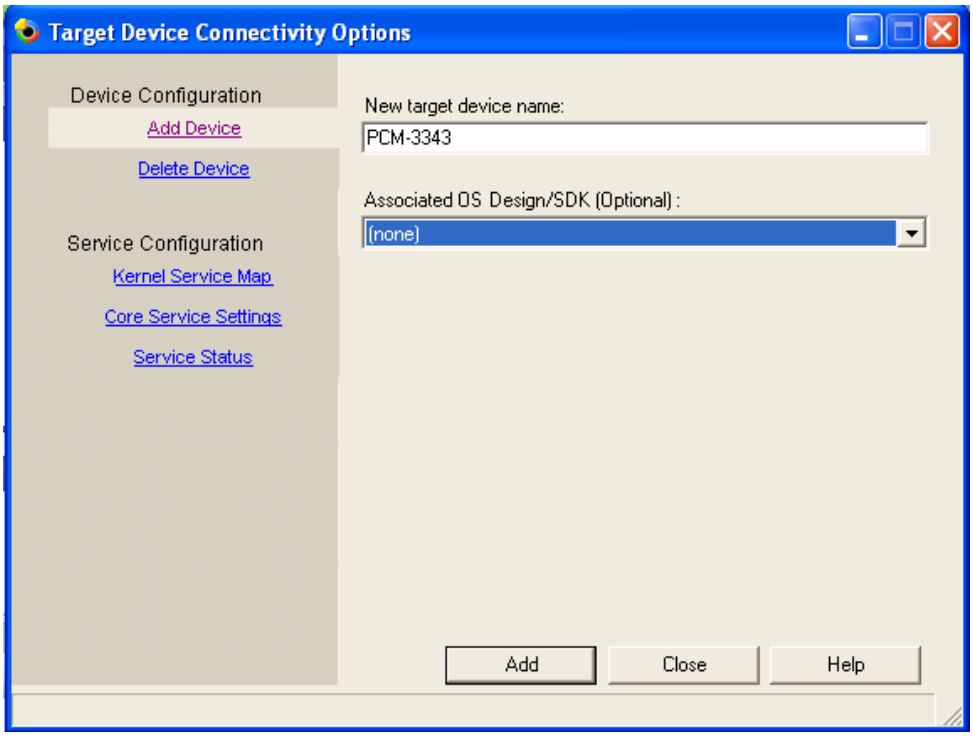
Uporabljal sem svoj USB pomnilnik, ki je bil že starejši. Njegova velikost je bila 256MB. Po nekajtedenskem testiranju sem ga uspel celo uničiti. Pomagalo ni niti formatiranje. Izbral sem drugega, ki pa je bil v velikosti 1 GB. Ta pa z operacijskim sistemom DOS ni deloval. Na srečo sem našel še en starejši pomnilnik v velikosti 128 MB, s pomočjo katerega sem uspešno zagnal DOS.

Po uspešnem zagonu DOS operacijskega sistema na ciljnem računalniku sem pognal program loadcepc.exe z dodatnimi parametri (Koda 1), da prične oddajati 'bootme' paketke v omrežje.

```
Loadcepc.exe /v /eboot.bin
```

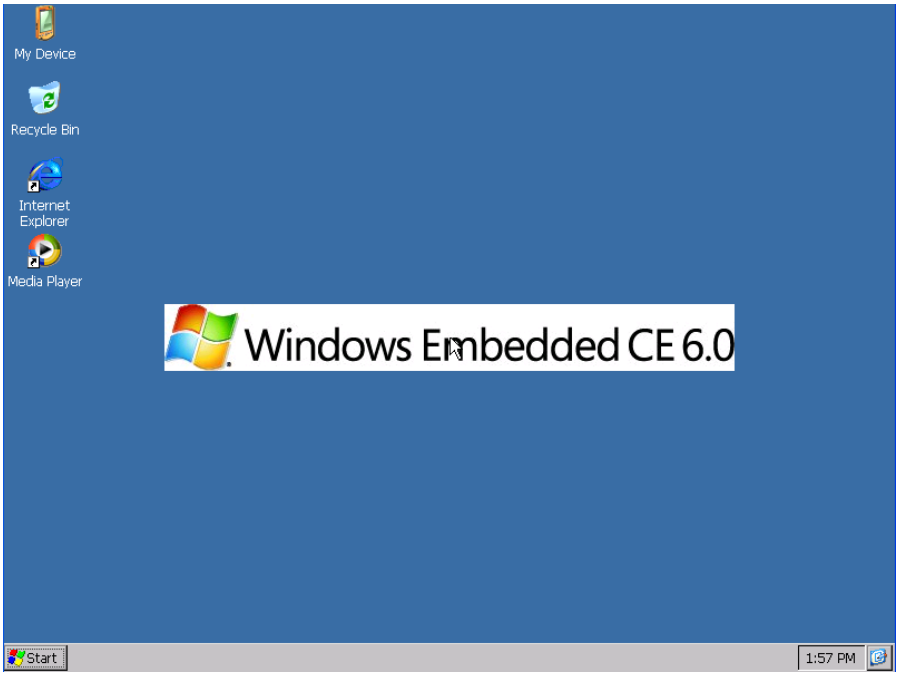
Koda 1: Niz, ki se uporabi za zagon programa loadcepc in prenos slike preko omrežja.

Seveda je potrebno ustrezno nastaviti tudi razvojni računalnik. V oknu za nastavitve povezovanja s ciljnim računalnikom (Target Device Connectivity Options), izdelamo nov profil in vnesemo ime računalnika (Slika 19). Nato za izdelani profil izberemo povezovanje preko omrežja in izberemo dodatne nastavitve. Odpre se nam okno, kamor Visual Studio 2005 izpiše vse naprave, ki v omrežje oddajajo 'bootme' paketke.



Slika 19: Pričetek izdelave novega profila za povezovanje naprave.

Ko izberemo željeno ciljno napravo se računalnika povežeta. Po uspešno vzpostavljeni povezavi se naloži slika operacijskega sistema, ki smo jo izdelali. Če med postopkom ni prišlo do napak, na ciljnem računalniku vidimo sliko namizja, kot je na sliki 20 (v primeru da smo vključili ustrezne komponente).

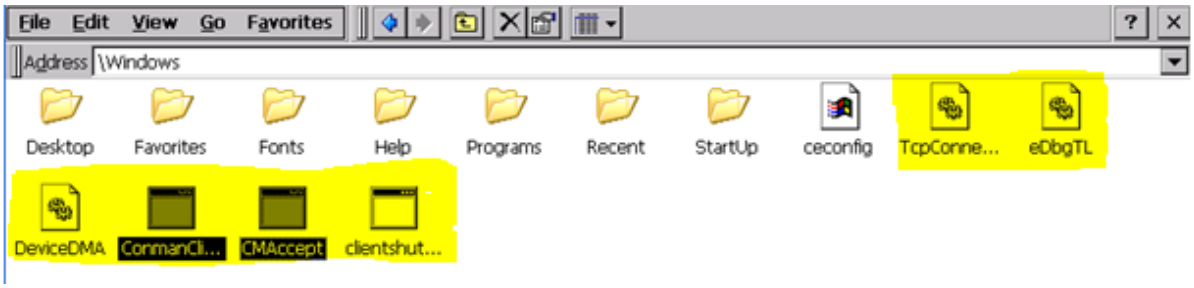


Slika 20: Namizje ciljnega računalnika z operacijskim sistemom Windows Embedded CE 6.0 R3.

3.5. Povezovanje ciljnega in razvojnega računalnika

Pri vgrajenih računalnikih imamo po navadi namensko aplikacijo. To aplikacijo izdelujemo, podobno kot samo sliko operacijskega sistema, na razvojnem računalniku. Ker računalnika nista enaka pa je ne moremo na tem računalniku tudi testirati, zato moramo računalnika znova povezati.

Pri povezavi se nam ponudi več novih možnosti, saj imamo sedaj na ciljnem računalniku nameščen operacijski sistem. Sam sem se odločil za povezovalno okolje 'CoreCon'. To okolje je vključeno v VS2005 in VS2008. Odvisno od razvojnega programa moramo ustrezne datoteke namestiti še na ciljni računalnik. Sam sem izdelal kar podprojekt, ki ob vključitvi v katalogu, ob namestitvi vnese potrebne datoteke v sliko operacijskega sistema, tako da so potrebne datoteke na ciljnem računalniku v mapi 'Windows' (Slika 21) in ni potrebe po prenosu datotek vsakič znova.



Slika 21: Datoteke potrebne za povezovalno okolje CoreCon. Z modro označeni datoteki moramo še zagnati.

Ko je torej ciljni računalnik prižgan in povezan, moramo v mapi Windows v raziskovalcu pognati najprej datoteko ConmanClient2.exe, nato pa še CMAccept.exe. Slednja za kratek čas odpre vrata za dostop do računalnika, zato moramo v roku 30 sekund vzpostaviti povezavo z razvojnega računalnika. Ko je povezava vzpostavljena, na ciljnem računalniku ni potrebno nobenih dodatnih nastavljanj ali potrjevanj, ampak le prevedemo programsko kodo na razvojnem računalniku in poženemo razhroščevalnik. Okolje CoreCon nato samo prenese preveden program na ciljni računalnik in prične z razhroščevanjem.

3.6. Silverlight

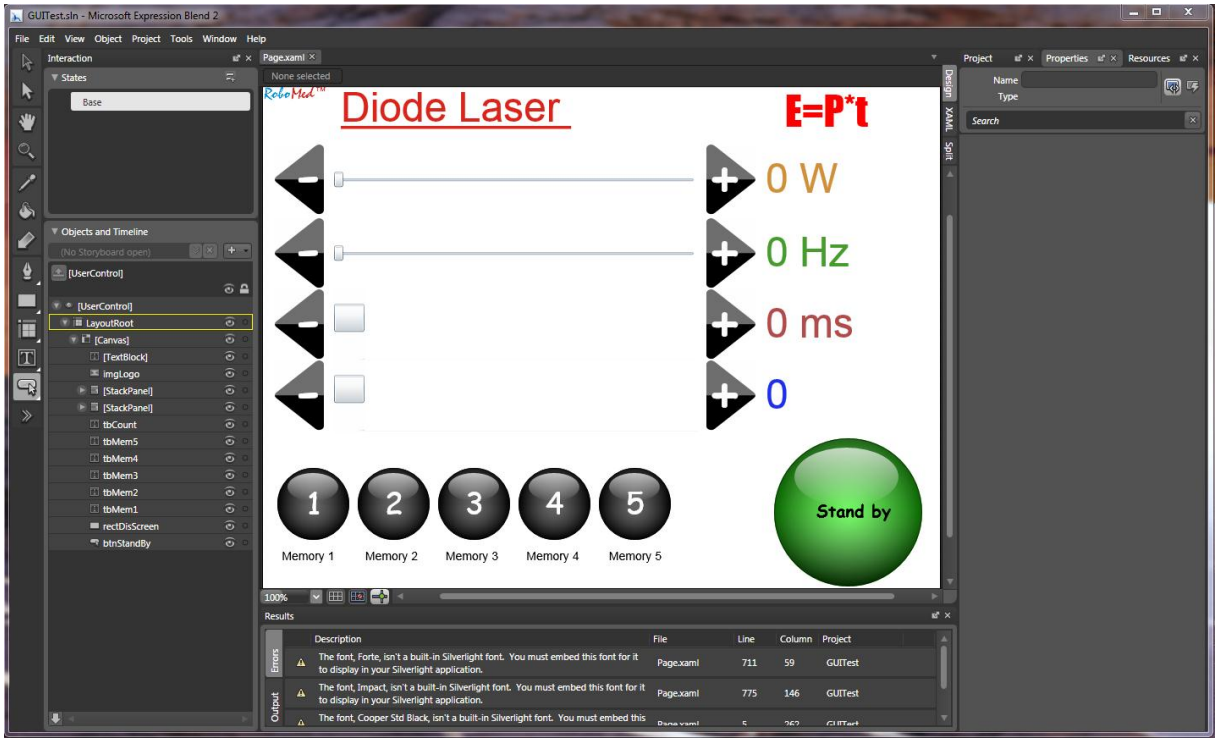
Odmevnejša novost v zadnjih dveh različicah Windows Embedded CE je Silverlight tehnologija, prirejena za Windows Embedded. Tehnologija Silverlight nam olajša izdelavo uporabniškega vmesnika, ki ni v tipično sivih odtenkih, ampak je v stilu naprave same. Prvotno je bila razvita za izdelavo spletnih strani, potem za vgradne naprave, sedaj se jo uporablja še na mobilni platformi Windows Phone 7. Govori se tudi o polni različici za aplikacije namiznih računalnikov (danes že prisotna pod imenom WPF – Windows Presentation Foundation). Posebnost tehnologije Silverlight je ločitev uporabniškega vmesnika od kode. To pomeni, da lahko programer razvija kodo, medtem pa dizajner oblikuje grafični vmesnik. Ko vsak svoj del združita bi morala aplikacija delovati, če sta se oba držala vnaprej dogovorjenih imen. Uporabniški vmesnik za Silverlight je lahko napisan v jeziku XAML – eXtensible Application Markup Language. Primer sintakse je viden v kodi 2.

```
<StackPanel>
  <Button Content="Click Me"/>
</StackPanel>
```

Koda 2: Jezik XAML je označevalni jezik, ki je izpeljanka jezika XML.

Vendar pa to ni edini način. Lahko je tudi pisan v programskem jeziku C, a potem izgubimo prednost ločitve uporabniškega vmesnika od kode.

Kot imamo razvojna orodja za pisanje programske kode v Javi, Cju, itd., imamo tudi orodja za pisanje XAML kode. Z Microsoftovim orodjem Expression Blend 2, je izdelava uporabniškega vmesnika precej poenostavljena.



Slika 22: Uporabniški vmesnik programa Blend 2.

Blend je orodje namenjeno dizajnerjem, kar se vidi že po njegovem vmesniku (Slika 22). Podpira tudi vnašanje datotek iz programov Adobe Photoshop in Adobe Illustrator. To je zelo uporabno, saj so določene stvari precej lažje izvedljive v kakem drugem programu kot v Blend-u.

Omenil bi še podporo za animacije. Tu se pokaže prava moč Silverlighta. Dizajner oziroma programer animacijo namreč le opiše, nakar jo izvede Silverlight pogon. Uporabnik ima 'storyboard' ali časovno tablo, kjer označi stanje v določenem trenutku. Če želimo, na primer, spremeniti velikost pravokotnika v eni sekundi, se le postavimo za eno sekundo naprej in nato povečamo pravokotnik. Ko predvajamo animacijo, se pravokotnik počasi povečuje do izbrane velikosti. Uporabnik mora torej določiti le začetne in končne vrednosti ter časovno skalo. Vmesne korake izvede Silverlight pogon. Pri projektu DL30 animacije sicer niso imele ključnega pomena.

3.7. XAML2CPP

Silverlight je z razvojnimi orodji zelo dobro podprt pri različici za spletne aplikacije, katerih koda je napisana v programskem jeziku C# ali Visual Basic. Pri različici Silverlight for Windows Embedded, pride do problema, ker je podprt edino jezik C, brez okolja .NET. Orodje Blend, jezika C nima podprtega, tako da nam služi le za izdelavo XAML kode in ne programske kode. Seveda pa je potrebno tudi v programu vedeti do katerih objektov imamo dostop. Pri razvoju take aplikacije je zato kar nekaj dela s samo povezavo kode XAML in C. Pri različici Windows Embedded Compact 7 obstaja orodje, ki nam omogoča avtomatizirano izdelavo kode na podlagi datotek iz projekta izdelanega v Blend 3.

A za različico Windows Embedded CE 6.0 R3, nam to orodje ne koristi, saj je že pri preprostih projektih precej veliko dela, da kodo, ki je izdelana za WEC7 priredimo za CE 6.0 R3. Za starejšo različico vseeno obstaja orodje XAML2CPP, ki ga je izdelal Valter Minute. Klub temu, da žal ni podprto s strani Microsofta, je široko uporabljeno s strani programerjev, saj nam napiše kar nekaj vrstic kode, s katero bi morali v nasprotnem primeru sami izgubljati čas.

XAML2CPP je zmogljiv a preprost program. Poženemo ga kar z ukazne vrstice in edini način upravljanja so parametri. Ob zagonu namreč kot parametre le podamo .xaml datoteke, katere želimo uporabiti v svojem programu. V kodi 3 je primer parametrov kakor smo jih uporabili za izdelavo projekta DL30.

```
XAML2CPP.exe Page.xaml ModeItem.xaml
```

Koda 3: Primer parametrov za program XAML2CPP.

Po končanih operacijah, se progam le zapre, nimamo pa niti povratne informacije o uspešnosti izvedbe, tako da moramo biti predvsem pozorni, da so imena podanih datotek pravilna. Program nam izdela kar 5 datotek, izmed katerih ena vsebuje resurse, ena nam olajša nadgrajevanje, v ostalih pa se nahajajo opisi objektov. Naša naloga je le še odpreti nov podprojekt tipa Win32. V naš projekt vključimo le datoteko »XAML2CPP.h« in pa seveda vnesemo tudi nekaj kode. Da lahko prevedemo program, moramo vnesti vsaj vse dogodke, katere smo definirali že v Blendu. Nato program prevedemo in prikaže se nam uporabniški vmesnik, kot smo ga izdelali v programu Blend. Če ga želimo uporabljati moramo seveda dodati še malo logike programu.

XAML2CPP.exe nam v metodi InitWindowParms definira okno z naslovno vrstico. Postavi ga v zgornji levi kot. Postavitev je dobra, ker pa želimo, da je naša aplikacija celozaslonska ne potrebujemo naslovne vrstice. Na srečo je metoda definirana kot virtualna, tako lahko v svoji metodi popravimo potrebno in nam ni treba skrbeti za popravke. Dejstvo je, da vsakič ko spremenimo .xaml datoteko moramo pognati XAML2CPP, ki pa spremeni vse svoje datoteke. V naši metodi oknu odvzamemo naslovno vrstico in mu dodamo razširjen nabor stilov, da postane okno v ospredju kot je vidno v kodi 4. Tako bomo na zaslonu videli le okno naše aplikacije kot ga izdelamo in uporabnik niti nebo vedel, da je v ozadju operacijski sistem.

```
wp->Style           = WS_POPUP;
wp->ExStyle         = WS_EX_TOPMOST;
```

Koda 4: Koda za spremembo stila okna.

3.8. Izdelava start gumba

V tem podpoglavju bo opisana izdelava tako imenovanega start gumba. Prvih nekaj korakov bo izvedenih v Blendu in bodo bolj oblikovalske narave.

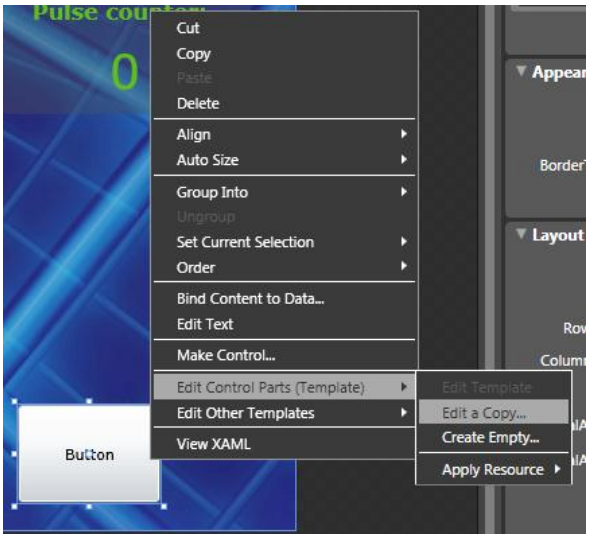
Za opis tega gumba sem se odločil, ker se mu oblika ob kliku programsko spremeni in sem za njegovo izdelavo porabil več časa. Torej gumb ima tri stanja:

- čakam (stand by),
- v pripravljenosti (ready),
- sprožen (emission).

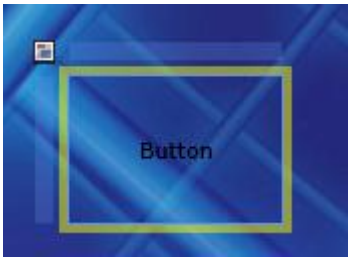
Pri vsakem stanju je barva drugačna. Izdelan gumb z napisom stand by je zelene barve in označuje čakajoče stanje. Stanje v pripravljenosti označuje napis ready in oranžna barva, v zadnjem stanju, ko je DL30 sprožen pa je gumb rdeče barve in ima napis emission. Posebnost zadnjega, sproženega, stanja je tudi ta, da gumb ni občutljiv na dotik. Torej vhoda in izhoda iz zadnjega stanja ne moremo krmiliti preko gumba samega.

Zahtevo za stanje 'emission' dobimo preko splošno namenskih GPIO vrat. Vhod 9 na GPIO vratih je povezan z mikroporcesorjem ARM. Ta vhod je aktiven ko je preko stopalke DL30 v stanju sprožen. Ko ni aktiven je sistem v stanju pripravljenosti.

Pričel sem tako, da sem na zaslon postavil nov gumb in izbral ukaz za urejanje kopije objekta (Slika 23). V tem načinu, v oknu objektov ne vidimo več vseh objektov, ki so v projektu, ampak objekte s katerimi je sestavljen izbran objekt. Vse objekte razen prikazovalnika vsebine sem odstranil (Slika 24), ker nisem želel standardnega, sivega gumba, ampak je bil cilj izdelati podobo okroglega gumba s 3D izgledom.

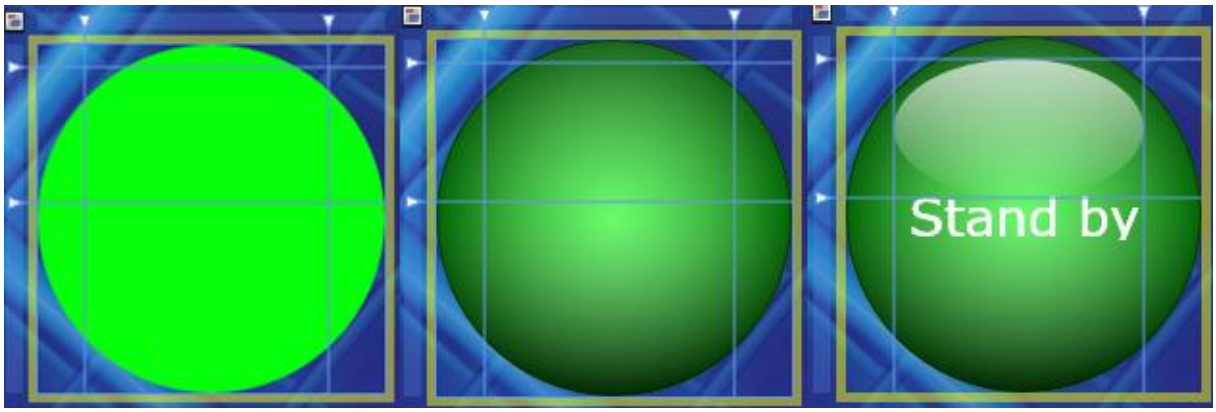


Slika 23: Program Blend nam omogoča prilagajanje izgleda že obstoječih kontrolnikov. Najlažji način je, da izelamo kopijo kontrolnika, ki jo ustrezno poimenujemo in uredimo. Seveda je omogočeno tudi kasnejše urejanje. Možna je tudi izdelava lastnega kontrolnika od začetka. Prednost urejanja kopije je predvsem, da nam ostanejo vsa stanja in dogodki.



Slika 24: Gumb v načinu urejanja, po tem, ko smo odstranili vse objekte, razen prikazovalnika vsebine (teksta).

Izgled start gumba je dosežen s pomočjo petih objektov. Za ozadje je najprej le preprost lik, krog zelene barve. Nato sem dodal dve elipsi z gradientom od prozorne k črni barvi in še eno od prozorne k beli in je tako barva gumba popolnoma prilagodljiva, saj je odvisna le od ozadja in jo lahko spremenimo z nastavljanjem lastnosti. Na koncu sem dobil gumb kot sem si ga zamislil. Postopni koraki so vidni na sliki 25. Dodal sem mu tudi 3D efekt, ko kliknemo na gumb, se ta pogrezne. To sem dosegel s tem, da sem premaknil sence in pa besedilo ko je gumb v stanju pritisnjen.



Slika 25: Postopna izdelava gumba. Prvi objekt je preprost zelen krog, ki mu kasneje z dodajanjem prosojnih slojev dodamo 3D izgled.

Po teh korakih nam program Blend izdela preko 100 vrstic XAML kode, ki opisujejo izgled obeh stanj gumba. Časovno je bilo kar zamudno, da sem ujel položaje senc, ki dajo efekt krogle. Kljub porabljenemu času sem privarčeval ogromno časa v primerjavi s tem, koliko časa bi izdelava takšnega gumba trajala, če bi izgled gumba opisoval s kodo. Sedaj imamo izdelano vizualno podobo gumba, manjka pa nam še programska koda, da bo gumb imel kak pomen.

Najprej sem sprogramiral, da se izgled gumba spremeni glede na stanje (čakam/v pripravljenosti).

```
if(isStdBy)
{
      colorNew = RGBA(0, 255, 225, 255);
      btnvalue.pReadOnlyStringVal = L"Ready";
      visible = XRVisibility_Visible;
}
else
{
      colorNew = RGBA(0, 255, 25, 255);
      btnvalue.pReadOnlyStringVal = L"Stand by";
      visible = XRVisibility_Collapsed;
}
```

Koda 5: Koda v C++, ki določi barvo in napis na gumbu.

V kodi 5 vidimo kodo, ki gumbu priredi ustrezno barvo in ime. Ta del je zanimiv, saj tukaj vplivamo na izgled uporabniškega vmesnika, ki je pisan v jeziku XAML, s kodo pisano v jeziku C++. Spremenljivka *colorNew* je tipa COLOREF, ki je sestaljen iz štirih, eno bajtnih vrednosti, ki predstavljajo stopnjo prosojnosti, modre, zelene in rdeče barve, torej določa barvo. Spremenljivka *btnvalue* je tipa XRValue. Spremenljivka *visible* je tipa XRVisibility, ki vpliva na vidnost komponente, ki onemogoči vse objekte na zaslonu razen start gumba. Ti dve spremenljivki sta enumeratorja za Silverlight for Windows Embedded.

Ko spremenljivkam priredimo vrednosti, jih vnesemo v objekte, kot vidimo v kodi 6.

```
//Change button value
btnStart->SetContent(&btnvalue);
//Change button color
IXRSolidColorBrush* myPaintBrush;
IXRApplicationPtr appl;
GetXRApplicationInstance(&appl);
appl->CreateObject(IID_IXRSolidColorBrush, &myPaintBrush);
myPaintBrush->SetColor(colorNew);

btnStart->SetBackground(myPaintBrush);
```

Koda 6: Prirejanje spremenljivk objektom.

S tem smo dosegli, da ima gumb dve stanji in za vsako primeren izgled.

Pred spremembo stanja moramo v to stanje poslati mikrokontroler, da lahko ustrezno vplivamo, v primeru, da do spremebe stanja v mikrokontrolerju ne more priti. To je sprogramirano tako, da najprej preberemo vse parametre. Tudi za to uporabljamo ukaze, ki so enumeratorji za Silverlight for Windows Embedded, saj moramo prebrati stanja večstanjskih gumbov in vrednosti v okencih za tekst. Nato vse zbrane parametre ustrezno zapakiramo, kar pomeni, da jim dodamo začetek in konec ter pariteto in jih pošljemo preko serijskih vrat. V primeru, da mikrokontroler uspešno prebere vse parametre, nam vrne potrdilo, preko linije na GPIO vratih. Šele tedaj pošljemo gumb start v novo stanje.

S stanja pripravljenosti se lahko pomaknemo nazaj v stanje čakanja, ali pa sprožimo laser in gremo v stanje sprožen. Ker torej želimo spremljati dva dogodka, ustvarimo novo nit. Prva nit procesira dogodke sprožene z uporabniškega vmesnika, druga nit pa skrbi za dogodke na GPIO vratih. Vloga prve niti je že opisana. Druga nit pa je dejansko zanka, ki nadzoruje stanje linje 9 na GPIO vratih. Če je linija v nizkem stanju (tedaj je aktivna), potem označi, da smo v stanju 'sprožen' in izriše gumb kot je prikazan na sliki 26. V kodi 7 vidimo ukaz, ki prikaže rdečo barvo gumba, napis emission in onemogoči občutljivost na dotik.

Slika 26: Prikaz start gumba v stanju 'sprožen'.

```
//Show emmission
EmissionButtonImage->SetVisibility(XRVisibility_Visible);
```

Koda 7: Ukaz za prikaz gumba v stanju 'sprožen'.

3.9. Beleženje napak

Za lažje razhroščevanje sem implementiral tudi sistem za beleženje napak in dnevniških datotek. Med razvojem trenutno uporabljam le beleženje napak, kar mi je že nekajkrat pomagalo odkriti napako in jo odpraviti.

Napisal sem kar svoj razred myLogger in ga spravil v novo .cpp in .h datoteko, tako da je omogočena enostavna uporaba tudi pri drugih projektih. Ko vključimo glavo je uporaba enostavna, kot jo prikazuje kodi 8.

```
#include "myLogger.h"
…
myLogger::Error("SUSI Dll IO couldn't read the door line.");
```

Koda 8: Prikaz klica podprograma za beleženje napake.

V primeru, da pride do napake in se ta del kode izvede, dobimo nov zapis v datoteki error.log. Če datoteka še ne obstaja jo podprogram naredi pred vnosom. Primer zapisa lahko vidimo v kodi 9.

```
-------------------------------------------------
Error entry written on 14.08.2011
 at 07:20:53
SUSI Dll IO couldn't read the door line.
-------------------------------------------------
```

Koda 9: Primer vpisa v datoteko napak.

Kot parameter podamo le željen vpis in pokličemo podprogram. Nato v podprogramu najprej preverimo obstoj datoteke in ustvarimo novo, če le ta še ne obstaja. V primeru, da pri tem koraku pride do napake, izpišemo opozorilo na sporočilno okence. Nato zabeležimo sistemski datum in čas, ter ju skupaj s sporočilom vpišemo v datoteko. Po končanem vnosu, datoteko zapremo. V kodi 10 vidimo izvorno kodo tega podprograma.

```
void myLogger::Error(char *msg)
{
     SYSTEMTIME stTime;

     ofstream errFile ("error.log", ios::app);
     if (!errFile.is_open())
          MessageBox(NULL,TEXT("An error happened and the
system was unable to log it!"),TEXT("Opening the error
file"),MB_OK);

     GetSystemTime(&stTime);
     errFile << " ------------------------------ " << endl;
     errFile << " Error entry written on " << stTime.wDay <<
"." << stTime.wMonth << "." << stTime.wYear << " at " <<
stTime.wHour << ":" << stTime.wMinute << ":" << stTime.wSecond
<< endl;
```

```
      if (msg != NULL)
           errFile << " " << msg << endl;
      errFile << " ------------------------------ " << endl;

      errFile.close();
}
```

Koda 10: Podprogram za beleženje napak. Koda je bila delno prirejena za boljšo preglednost.

4. SKLEP

Sistem DL30, delni razvoj katerega prikazuje ta diplomska naloga, je trenutno še v fazi prototipa. V diplomskem delu je opisan najverjetnejši končni izbor strojne opreme, saj se je v prototipu izkazala za ustrezno. V celoti je predstavljen tudi razvoj operacijskega sistema Windows Embedded CE 6.0 R3, ki je tudi ustrezno prestal dosedanja testiranja. Na koncu je predstavljena še izdelava namenskega programa, ki pa se bo zagotovo še spremenil. Trenutno nam omogoča nastavljanje določenih parametrov in upravljanje z laserjem. V nadaljevanju bo aplikacija omogočala še servisni način, ki bo služil za umerjanje laserja, shranjevanje stanj naprave v dnevnik in pa kopiranje dnevnika na USB pomnilnik. Morebitne dodatne funkcionalnosti bodo še naknadno dodane glede na želje uporabnikov.

5. LITERATURA IN VIRI

[1] (2011) ATX format. Dostopno na: http://en.wikipedia.org/wiki/ATX.

[2] (2011) PC/104 format. Dostopno na: http://www.pc104.org/pc104_specs.php.

[3] (2011) Študija o številu procesorjev leta 2015. Dostopno na: http://www.windowsfordevices.com/c/a/News/IDC-Smart-Technology-World-conference-and-study/.

[4] (2011) Vortex86. Dostopno na: http://en.wikipedia.org/wiki/Vortex86.

[5] (2011) Vortex86. Dostopno na: http://www.vortex86.com/index2.html.

[6] (2011) Primerjava procesorjev. Dostopno na: http://www.fccps.cz/download/adv/frr/vortex/vortex.htm.

[7] (2011) HTC EVO 3D. Dostopno na: http://www.htc.com/www/product/evo3d/overview.html.

[8] (2011) LVDS. Dostopno na: http://en.wikipedia.org/wiki/Low-voltage_differential_signaling.

[9] (2011) Intel DisplayPort. Dostopno na: http://newsroom.intel.com/community/intel_newsroom/blog/2010/12/08/leading-pc-companies-move-to-all-digital-display-technology-phasing-out-analog.

[10] (2011) I^2C. Dostopno na: http://www.i2c-bus.org/.

[11] (2011) I^2C. Dostopno na: http://en.wikipedia.org/wiki/I%C2%B2C.

[12] (2011) CAN. Dostopno na: http://en.wikipedia.org/wiki/Controller_area_network.

[13] (2011) CCFL. Dostopno na: http://en.wikipedia.org/wiki/Cold_cathode in http://en.wikipedia.org/wiki/CCFL_inverter.

[14] (2011) Zasloni občutljivi na dotik. Dostopno na: http://www.tvielectronics.com/Documents/TouchScreenSolutions/TouchScreen_Technology_Comparison.pdf.

[15] (2010) Microsoft Windows Embedded. Dostopno na: http://www.microsoft.com/windowsembedded/en-us/products/windowsce/faq.mspx.

[16] (2011) BSP. Dostopno na:
http://en.wikipedia.org/wiki/Board_support_package.

[17] (2011) Samuel Phung, »Microsoft Windows Embedded CE 6.0«, Wiley Publishing, Inc., 2009.

[18] (2010) HIMEM.SYS. Dostopno na:
http://en.wikipedia.org/wiki/HIMEM.SYS.