LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

IEGULTO SISTĒMU TESTĒŠANA UN IEVIEŠANA

MAĢISTRA DARBS

Autors: **Jānis Mažuika**
Stud. apl. Nr. jm06014
Darba vadītājs: profesors Dr.dat. Jānis Bičevskis

RĪGA 2012

ANOTĀCIJA

Telemetrijas iegultā sistēma ir specifiski pielāgots kontroles modulis, kura uzdevums ir informācijas apmaiņa kopējā sistēmā. Autors iepazīstina ar nozares specifiku, izvēlēto iekārtas arhitektūru un funkcionalitātes specifiku. Iegultās sistēmas testēšana ietver iekārtas elektrisko slēgumu pārbaudi, programmatūras testēšanu, datu transporta protokolu testēšanu, pārsūtāmo datu formāta testēšanu un sistēmtestēšanu. Autors iepazīstina ar patstāvīgi izstrādāto sistēmu iekārtas rūpnieciskajai testēšanai, kas ietver aparātprogrammatūras moduli iegultajai sistēmai un personālās darbstacijas lietojumprogrammu. Darbā izklāstīta galīga automāta testēšanas metode iegulto sistēmu komunikācijas algoritmu testēšanai.

Atslēgvārdi: telemetrija, iegultas sistēmas testēšana, iekārtas testēšana

ABSTRACT

Embedded system testing and deployment

Embedded system of telemetry - specifically adapted control module, which is responsible for information sharing in a common system. Author presents the characteristics of the telemetry, embedded systems architecture and functional specificity. Embedded systems testing includes an electric circuit, firmware, data transport protocol, the transmitted data format and common system testing. The author presents the developed system for industrial testing equipment, which includes firmware module for embedded system and personal desktop application with graphical user interface. The author works out a final machine testing method for embedded systems communication algorithm testing purpose.

Keywords: telemetry, embedded systems testing, hardware testing

AUTOREFERĀTS

Autors šī maģistra darba izstrādes ietvaros ir:

- izklāstījis telemetrijas iegultās sistēmas specifiku un konkrētas pielāgotās iegultās sistēmas projektējumu, uzskaitot ietverto funkcionalitāti;
- izvērtējis konkrētas iegultās sistēmas testēšanas metodes, balstoties uz iekārtas un aparātprogrammatūras projektējumu;
- izzinājis rūpnieciskās ražošanas un testēšanas vidi rūpnīcā, kas nosaka iekārtas testēšanas sistēmas saskarnes un funkcionalitātes prasības;
- izstrādājis iekārtas testēšanas sistēmu, kas paredzēta izmantošanai rūpnieciskās ražošanas procesa ietvaros;
- izklāstījis galīga stāvokļu testēšanas metodes, kā arī testu ģenerēšanas paņēmienus, izmantojot sistēmas galīga stāvokļa automāta grafu.

Galvenais autora ieguldījums šī darba ietvaros ir pielāgotas iekārtas testēšanas sistēmas izstrāde, kas ietver aparātprogrammatūras moduli, tā integrēšanu kopējā iegultajā sistēmā, kā arī darbstacijas lietojumprogrammu testu vadības grafiskajai lietotāja saskarnei.

Darbs tiek noslēgts ar galīga stāvokļa testēšanas metodes realizācijas izklāstu, papildinot to ar algoritmiem automatizētai testu kopas ģenerēšanai, lai aprakstīto paņēmienu izmantotu komunikācijas protokolu testēšanai iegultajai sistēmai.

SATURA RĀDĪTĀJS

APZĪMĒJUMU SARAKSTS

ADC - ierīce, kas pārveido ieejas analogo signālu diskrētajā kodā (ciparu signālā), no angļu val. *Analog-to-Digital Converter.*

ASCII - Amerikas informācijas apmaiņas standartkods, no angļu val. *American Standard Code for Information Interchange*.

DEX - datu apmaiņas protokols audita un notikumu datu saņemšanai no tirdzniecības automāta, no angļu val. *Data Exchange*.

EEPROM - uz pusvadītājiem bāzēta lasāmatmiņa, no angļu val. *Electrically Erasable Programmable Read Only Memory*.

EVA-DTS - tirdzniecības automātu datu formāta standarts, no angļu val. *European Vending Association Data Transfer Standard*.

FLASH - elektroniski dzēšama un pārprogrammējama atmiņa patstāvīgai datu glabāšanai.

FFT - Furjē transformācija, no angļu val. *Fast Fourier Transform.*

FPGA - pārprogrammējamu loģisku elementu matricas balstīts mikroshēmu tips, no angļu val. *Field programmably Gate Array.*

GSM - mobilo sakaru standarts, no angļu val. *Global System for Mobile communications.*

GPS - globālā pozicionēšanas sistēma, no angļu val. *Global Positioning System.*

HID - datora perifērijas (*USB*) iekārtas apzīmējums, no angļu val. *Human Interface Device.*

HTTP - hiperteksta pārsūtīšanas protokols, no angļu val. *Hypertext Transfer Protocol.*

I2C - zema ātruma datu kopnes protokols, no angļu val. *Inter-Integrated Circuit.*

ICD - aparātprogrammatūras izstrādes vides MPLABX datu apmaiņas protokols ar mikroprocesoru.

IMEI - mobilās iekārtas identifikators *GSM* tīklā, no angļu val. *International Mobile Equipment Identity.*

IMSI - *SIM* kartes mobilais identifikators mobilajā tīklā, no angļu val. *International Mobile Subscriber Identity*.

MDB - 8 bitu seriālais iekārtu slēguma datu pārraides protokols, no angļu val. *Multi-Drop Bus.*

NFC - bezkontakta datu apmaiņas risinājums iekārtām, no angļu val. *Near Field Communication.*

PROT-A - tirdzniecības automātu datu apmaiņas protokols (*Exec vai Protocol-A*).

QR - matricas svītru kods, jeb divdimensionāls kods, no angļu val. *Quick Response Code.*

RAM - brīvpiekļuves atmiņa, no angļu val. *Random Access Memory*.

RS232 - telekomunikāciju datu transporta protokola standarts.

SIM - GSM mobilo telefonu operatora viedkarte, no angļu val. *Subscriber Identity Module.*

SMS - teksta ziņojumu sūtīšanas serviss, no angļu val. *Short Message Service*.

UART - individuālu seriālo savienojumu tips no angļu val. *Universal Asynchronous Receiver/Transmitter.*

USB - universālās seriālās kopnes standarts no angļu val. *Universal Serial Bus.*

VMC - tirdzniecības automāta galvenais vadības modulis (iekārta), no angļu val. *Vending Machine Controller*.

XML - paplašināmā iezīmēšanas valoda, no angļu val. *Extensible Markup Language.*

IEVADS

Iegultās sistēmas testēšana ietver ne tikai aparātprogrammatūras testēšanu noteiktos līmeņos, bet arī iekļauj projektētās iekārtas fizisko komponenšu slēgumu testēšanu. Iekārtas projektēšana un prototipu ražošanas, kā arī aparātprogrammatūras procesi ir cieši saistīti un savstarpēji atkarīgi.

Autors izvēlējies tirdzniecības automātu telemetrijas iegultās sistēmas testēšanu kā pētāmo objektu, lai analizētu iekārtas testēšanas specifiku, piedāvātu individuālu risinājumu rūpnieciskajai iekārtas testēšanai, kā arī izklāstītu aparātprogrammatūras testēšanas paņēmienus, tai skaitā galīga automāta testēšanas metodi.

Tirdzniecības automātu industrijai attīstoties, tiek pielāgoti iegulto sistēmu telemetrijas risinājumi, lai uzlabotu šo automātu operatoru darba efektivitāti. Attālināta tirdzniecības automātu statusa novērošana sniedz iespēju iepriekš paredzēt piegādājamo preču skaitu, lai operatoram nodrošinātu tikai nepieciešamo klāstu ar precēm vai produktiem, kurus nepiecienāms uzpildīt konkrētajiem tirdzniecības automātiem. Būtiska nozīme ir audita atskaites attālinātai iegūšanai no šiem automātiem, kas atvieglo operatora darba funkcijas, kā arī ļauj nodalīt automāta uzpildīšanas procesu no audita iegūšanas. Izveidojot speciāli pielāgotu iegultās sistēmas risinājumu tirdzniecības automātu attālinātai apsekošanai, nepieciešams atbilstoši funkcionalitātei izprojektēt iekārtu un izstrādāt aparātprogrammatūru. Sakarā ar iekārtas plānoto saderību ar dažādu ražotāju un modeļu tirdzniecības automātiem, aparātprogrammatūru nepieciešams iepriekš testēt, kā arī paredzēt individuālus pielāgojumus saderībai ar konkrētiem tirdzniecības automātiem.

Uzsākot telemetrijas iegultās sistēmas ražošanu, nepieciešams izveidot iekārtas testēšanas sistēmu, kas paredzēta rūpnieciskai iekārtas ražošanai. Šāda sistēma nodrošinātu informāciju par aparatūras funkcionālo stabilitāti, izvēlēto iekārtas komponenšu kvalitāti un citiem raksturlielumiem. Iekārtas testēšanas sistēma ir ļoti specifiski pielāgota iekārtas projektējumam, tādēļ nepieciešams to individuāli izstrādāt.

Apzinot konkrētās tirdzniecības automātu telemetrijas iegultās sistēmas specifiku, autors izvirza mērķi apzināt iegultās sistēmas testēšanas specifiku, kā arī izstrādāt konkrētu sistēmu iekārtas rūpnieciskai testēšanai, kas ietver darbstacijas lietojumprogrammu, kā arī aparātprogrammatūras moduli testēšanas protokola komandu izpildei iekārtai. Papildus autors pēta iegultās sistēmas testēšanas metodes, īpaši pievēršot uzmanību galīga automāta testēšanai, kuru plānots pielietot iegultās sistēmas komunikācijas un datu apmaiņas protokolu testēšanai.

Darbam autors iepazīstina ar konkrētu mikroprocesora modeli, pamatojot tā izvēli ar konkrētās funkcionalitātes atbalstu. Autors apraksta iegultās sistēmas projektējuma un aparatūras izstrādes procesu, kā arī izskaidro aparātprogrammatūras ielādes iekārtā procesu.

Tiek sniegts ieskats tirdzniecības automātu telemetrijas nozares specifikā, uzskaitot nepieciešamo iekārtas un aparātprogrammatūras funkcionalitāti. Izklāstīts aparatūras pielietojums un priekšrocības, kā arī definēti ārējie faktori, kas ietekmē iegultās sistēmas darbības stabilitāti tirdzniecības automātu telemetrijas nozarē.

Autors pievēršas iegulto sistēmu testēšanas uzdevumam, apskatot gan tradicionālo testēšanas līmeņu uzskaitījumu, gan pieminot specifisku testēšanas paņēmienu pielietojumu nozīmi konkrētās iegultās sistēmas testēšanai.

Darba ietvaros autors izklāsta patstāvīgi izveidotās iekārtas testēšanas sistēmas uzbūvi, funkcionalitāti un pamato izvēlētās realizācijas metodes, kuras balstītas uz reālas ražošanas rūpnīcas sniegtajiem ieteikumiem sistēmas izveidei. Darbā ietverts atsevišķu programmatūras moduļu funkcionalitātes izklāsts, kas papildināts ar diagrammu attēliem.

Visbeidzot autors apskata galīga automāta testēšanas metodi, lai izvērtētu tās pielietojumu datu transporta protokolu un datu formāta testēšanai tirdzniecības automātu telemetrijas iekārtas kombināciju testēšanai.

1. IEGULTAS SISTĒMAS IZSTRĀDE

1.1. Mikroprocesora izvēle

Pasaulē pieejami daudzi mikroprocesoru ražotāji, pazīstamākie no tiem ir *ARM, Atmel, Intel, MIPS, Microchip, PowerPC, Texas Instruments* un *Toshiba*. Mikroprocesoru izvēli nosaka dažādi faktori, galvenokārt veiktspēja, procesora instrukciju izpildes pārtraukumu noildze, iekļauto moduļu un speciālo funkciju reģistru kopums, pieejamo tapiņu skaits un izmaksas.

1.2. PIC mikroprocesora saime

Autors apskata *Microchip* mikroprocesora *PIC24F* saimi, kam raksturīga jaunās paaudzes 16 bitu arhitektūra.

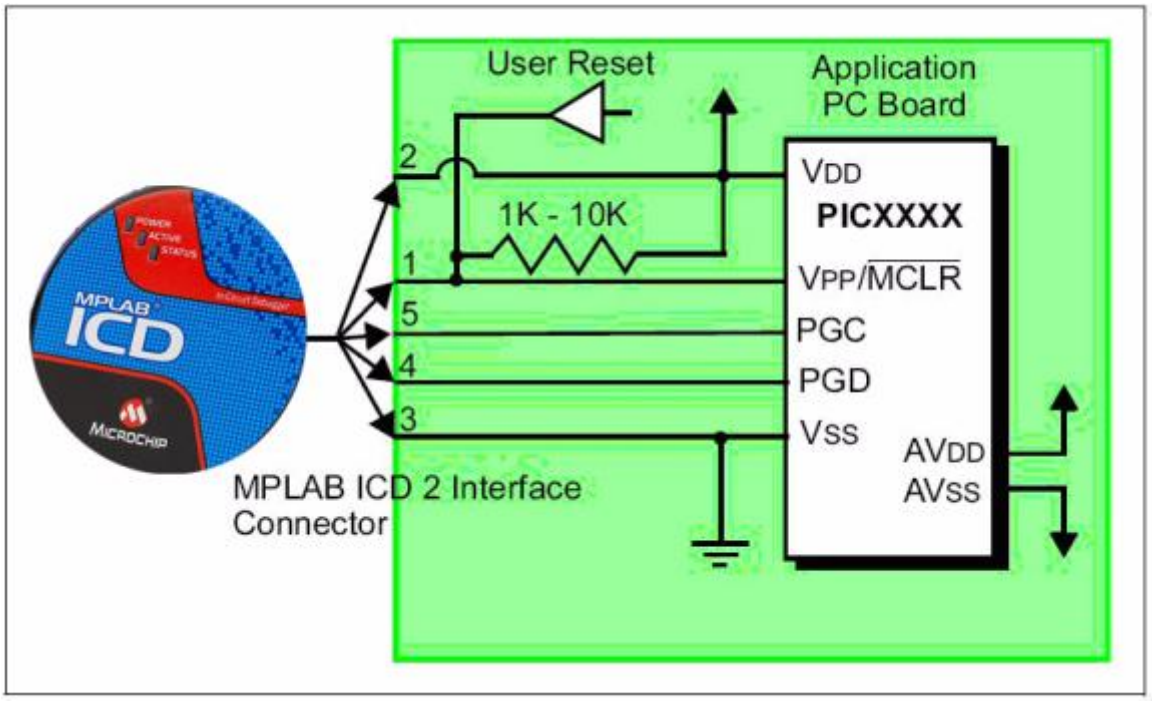
PIC24 tikai mazliet atpaliek ātrdarbības ziņā no digitālajiem signāla procesoriem (*dcPIC30*) un atšķiras ar to, ka trūkst digitālā signāla apstrādes instrukcijas (*FFT* (Furjē transformācija, no angļu val. *Fast Fourier Transform*)) un citas.

Programmas kods tiek izstrādāts valodā *C*, kompilējot ar *C30* kompilatoru, kas ģenerē instrukciju un ar to saistīto datu sakārtota kopu, taču maldīgs ir uzskats, ka iegūtā aparātprogrammatūra izpildās lēnāk nekā tiešā veidā rakstīts asamblera kods priekš *ASM30*, jo *C30* kompilators ietver optimizācijas atbalstu. *PIC24* mikroprocesoru sērija nodrošina programmas instrukciju izpildes ātrumu līdz 120 MHz un programmas atmiņas izmēru līdz pat 256 kilobaitiem, kas ir pietiekoši iegulto sistēmu programmēšanai, piemēram, telemetrijas iegulto sistēmu ierīcēm.

PIC24 mikroprocesori iedalās divās klasēs: *PIC24F* paredzēts kā lētāks risinājums, kas spēj nodrošināt apmēram 40 MHz, taču *PIC24H* sērija paredzēta sistēmām, kur nepieciešama liela veiktspēja, taču tā cena ir divkārt lielāka (četri līdz pieci ASV dolāri) nekā *PIC24F*. Cenu nosaka arī mikroprocesora tapiņu skaits (ārējā saskarne ar citām sistēmas komponentēm) un līdz ar to arī paredzētā datu kopņu funkcionalitāte priekš šīm tapiņām. Toties *PIC24F* arhitektūra paredz divus *UART* (individuālu seriālo savienojumu tips no angļu val. *Universal Asynchronous Receiver/Transmitter*) un divus *I2C* (zema ātruma datu kopnes protokols, no angļu val. *Inter-Integrated Circuit*) moduļus, kā arī divus līdz 16 bitiem vai vienu līdz 32 bitiem taimerus, savukārt uz ātrdarbību vērstajam *PIC24H* variantam šāda funkcionalitāte iztrūkst.

Microchip mikroprocesors ir saderīgs ar *ICD* (aparātprogrammatūras izstrādes vides MPLABX datu apmaiņas protokols ar mikroprocesoru) ķēdes slēgumu. Lai ielādētu

aparātprogrammatūru, visām piecām tapiņām jābūt attiecīgi pievienotām, taču vēl bez tām nepieciešams patstāvīga enerģijas avota pieslēgums mikroprocesoram (sk. 1.1 att.).

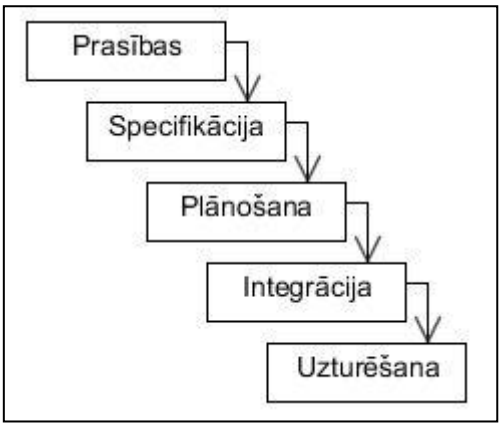


1.1 att. **ICD2 ķēdes slēgums *Microchip* mikroprocesoram** [1]

1.3. Iegultās sistēmas projektēšana

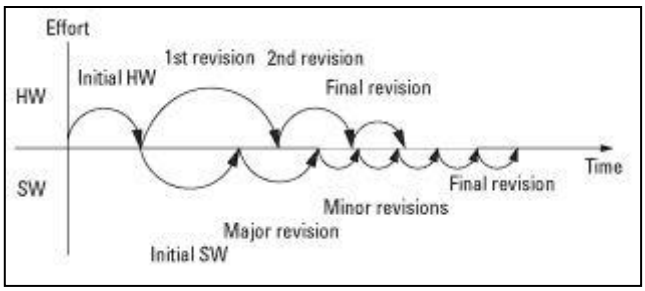
1.3.1. Iegulto sistēmu projektēšana

Iegulto sistēmu projektēšana un izstrāde var tikt veikta, vadoties pēc ūdenskrituma modeļa (sk. 1.2 att.). Parasti procesu uzsāk produkta pasūtītājs vai potenciālais lietotājs, uzskaitot nosacījumus, kuriem jāatbilst plānotajam gala produktam. Tad seko specifikāciju izstrādes fāze, atšķirībā no iepriekšējā etapa, dokumentus - specifikācijas sastāda sistēmas izstrādātāji, nevis lietotājs. Plānošanas fāzē izstrādātājs projektē iekārtas moduļus un sadala tos pēc funkcionalitātes, kā arī plāno algoritmu pielietojumu sistēmai. Integrācijas fāzē notiek testēšana, kas fiziskie komponenti ir gatavi, komponenti tiek savienoti un tiek izpildīti funkcionālie testi, kas iepriekš sagatavoti specifikāciju fāzē. Tālāk seko uzturēšanas fāze, kurā tiek novērsti vēlāk atklātie trūkumi un kļūdas [2; 3].



***1.2** att.* **Ūdenskrituma modelis iegulto sistēmu izstrādē**

Iegulto sistēmu izstrādei tiek piedāvāts arī alternatīvs paņēmiens ar atšķirīgu fāzu iedalījumu (sk. 1.3 att.). Šīs ilustrācijas mērķis ir parādīt aparatūras un aparātprogrammatūras attīstības soļus, kad iesākumā aparātprogrammatūras izstrādātājiem nākas gaidīt, līdz tiek piedāvāta pirmā aparatūras versija, tad seko paralēla iekārtas un programmatūras izstrāde, līdz beidzot izstrādāta gala versija [2]. Šis modelis uzskatāmi attēlo reālu iekārtas projektēšanas un aparatūras izstrādes procesu, kas parasti turpina attīsties, kamēr jau uzsākta aparātprogrammatūras izstrāde, kuras dēļ nereti ierosina izmaiņas iekārtas fizisko komponenšu komplektācijā.



***1.3** att.* **Alternatīvs modelis iegulto sistēmu izstrādē** [2]

1.3.2. FPGA iespējas

Iegulto sistēmu projektēšanai un izstrādei ar vien vairāk tiek izmantota *FPGA* (pārprogrammējamu loģisku elementu matricas balstīts mikroshēmu tips, no angļu val. *Field programmably Gate Array*) komponente. Tā sastāv no programmējamiem loģikas elementiem, kas veido šīs iekārtas konfigurāciju.

FPGA izmantošana nodrošina ērtu iegultās sistēmas modelēšanas paņēmienu izmantojot grafiskos rīkus vai arī kādu no digitālo sistēmu aprakstošajām valodām, piemēram, *Verilog*. Uzmodelēto iekārtu iespējams ērti simulēt un testēt, kā arī izmaiņu gadījumā nav

nepieciešams veikt fiziskas darbības ar iekārtu, tā vietā vienkārši pārkompilējot projektu un atkārtojot simulāciju vai testus. Salīdzinājumā ar fiziskas iekārtas prototipa testēšanu, tas ir daudz ērtāk, ātrāk un iespējams labāk simulēt iekārtas saskarnes ieejas vērtības, precīzi nosakot laika intervālus un citus uzstādījumus.

FPGA lietošanai nepieciešama speciāli izstrādāts rīks, kas saderīgs ar konkrētā izstrādātāja modeli, piemēram, Amerikas tehnoloģiju kompānija *Xilinx* piedāvā izstrādes vidi, kas saderīga ar *Spartan – 3 (FPGA)* plati.

Šo tehnoloģiju izmanto ne tikai sistēmas izstrādes laikā, lai prototipētu integrālās sistēmas shēmas, bet arī to iekļauj kā komponenti galaproduktā (iegultā sistēmā), kas paver iespējas mainīt šīs komponentes konfigurāciju, kad produkts jau ir nonācis pie lietotāja.

Mūsdienās *FPGA* tehnoloģija ir strauji attīstījusies, jo ievērojami pieaudzis programmējamo loģikas elementu skaits uz vienu *FPGA* iekārtu, kā arī attīstīta ietvertā funkcionalitāte (piedāvātie gatavie loģikas elementu bloki – iepriekš izstrādātās elementu sagataves, piemēram, skaitītāji, bīdes reģistri un citi) [2].

1.4. Mikroprocesora saskarne

Izmantojot konkrēta mikroprocesora modeli, programmatūras izstrādātājam nepieciešams rūpīgi izplānot ierobežotās saskarnes resursu izmantošanu, lai sasniegtu izvirzītos funkcionalitātes mērķus iegultās sistēmas realizācijai. Mikroprocesoru ārējā saskarne, tapiņas, ir ierobežots resurss, turklāt katrai no tām (signālu ievades un izvades tapiņas) piekārtots programmatūras atbalsts, kas atšķiras pēc plānotā datu apmaiņas protokola īpatnībām un iespējamo papildus programmatūras funkcionalitāti, piemēram, *Microchip PIC24FJ256GB110* mikroprocesors aprīkots ar 100 tapiņām, no kurām ieejas un izejas signāliem paredzētas 84, turklāt tas atbalsta dažādas perifērijas iekārtu slēgumus, to skaitā:

- četrus neatkarīgus *UART* slēgumus;
- trīs *SPI* kopnes;
- trīs *I2C* slēgumus.

Svarīgi izplānot tapiņu pielietojumu, jo tās, kurām specifiskais datu slēgums ir paredzēts, atbalsta arī ar to saistīto papildu funkcionalitāti, piemēram, kontrolsummu rēķināšanas metodes, ja izmanto *UART* slēgumu *RS232* (telekomunikāciju datu transporta protokola standarts) realizācijai, kas atvieglo programmatūras izstrādātājam darbu, turklāt nodrošinot procesora izpildei speciālu pārtraukumu, kad dati saņemti, kā arī speciālu reģistru, kurā tiek piedāvāta kontrolsummas vērtība saņemtajiem datiem. Šādi pielāgojumi (mikroprocesora atbalsts programmatūrai) arī pieejami ierobežotā skaitā, turklāt specifiskām tapiņām, tādēļ ir svarīgi iepriekš izplānot sistēmas arhitektūru atbilstoši pieejamo resursu daudzumam un

plānotajai sistēmas funkcionalitātei, turklāt labā prakse ir vienmēr paredzēt saderīgumu ar tālākas attīstības nosacījumiem sistēmai.

Perifērijas slēgumiem paredzēto tapiņu grupas savstarpēji pārklājas, tas nozīmē, ka vienlaicīgi vienam mikroprocesoram nav iespējams realizēt visu iespējamo perifērijas iekārtu kopumu, taču nepieciešams izplānot lietderīgāko kombināciju no tām, pēc iespējas izmantojot mikroprocesora piedāvāto funkcionalitātes atbalstu, citādi brīvi izvēlēto tapiņu gadījumā programmatūrai nepieciešams realizēt dublētu funkcionalitāti.

1.5. Aparātprogrammatūras izstrāde

Iegulto sistēmu programmēšanā plaši izmantota prakse ir stāvokļu mašīnas mehānisma ieviešana programmas vadības plūsmas organizēšanai. Šāds paņēmiens atvieglo sistēmas modeļa, grafa, izveidi, lai atspoguļotu programmas darbību, ko vēlāk iespējams pielietot programmatūras testēšanai, piemēram, galīga stāvokļa testēšanas metodei.

Sarežģītu sistēmu programmēšanai, kurām vienlaicīgi nepieciešams nodrošināt komunikāciju ar vairākām citam iekārtam (piemēram, *USB* (universālās seriālās kopnes standarts no angļu val. *Universal Serial Bus*) savienojums ar darbstaciju, *MDB* (8 bitu seriālais iekārtu slēguma datu pārraides protokols, no angļu val. *Multi-Drop Bus*) savienojums ar citu iekārtu un vēl komandu apmaiņa ar *GSM* (mobilo sakaru standarts, no angļu val. *Global System for Mobile communications*) moduli), ieteicams izmantot paralēlu vairāku uzdevumu apstrādes sistēmu, iepriekš izplānojot nepieciešamo veiktspēju un ātrdarbību, ko tieša veidā nosaka mikroprocesora instrukciju izpildes ātrums un veicamo uzdevumu patērētais laiks. Iespējams realizēt „*round-robin*" algoritmu vairāku iepriekš definētu uzdevumu apstrādei, kur katrs no šiem uzdevumiem uzskatāms par neatkarīgu stāvokļu mašīnu.

Mikroprocesora arhitektūra parasti piedāvā izmantot pārtraukumus perifērijas tapiņu saņemto datu apstrādei, taču kopējais sistēmai pieejamais pārtraukumu skaits parasti ir ierobežots, turklāt mazāks par visu realizējamo perifērijas saskarņu skaitu, tādēļ programmatūras izstrādātājam (vai programmatūras projektējuma veidotājam) nepieciešams rūpīgi plānot arī šo resursu.

1.6. Programmatūras ielāde

MPLABX vidē *PIC* mikroprocesoram izstrādāto pirmkodu kompilējot un sasaistot objektmoduļus ar *C30* kompilatoru, tiek iegūts bināra formāta izejas datne (aparātprogrammatūras instrukciju un konfigurācijas kopums). Izmantojot papildus *C30* kompilatora rīku, iespējams iegūt arī *ASCII* (Amerikas informācijas apmaiņas standartkods,

no angļu val. *American Standard Code for Information Interchange*) kodējuma datni, kura ietver heksadecimālā pierakstā programmas instrukciju un datu reprezentāciju.

Programmas atmiņas piekļuvei var tikt izmantots speciāls rīks – programmators (piemēram, *Pickit3*), kas saderīgs ar *ICD2* kopni un tās datu apmaiņas protokolu, lai tiešā veidā ieprogrammētu mikroprocesoru. Fiziski pievienots programmators var tikt lietots programmas atmiņas nolasīšanai no mikroprocesora, rakstīšanai vai dzēšanai. Lai izmantotu šo rīku, nepieciešams izmantot *MPLABX* programmatūras izstrādes vidi ar attiecīgu konfigurāciju vai arī atsevišķu lietojumprogrammu, kas paredzēta konkrētā programmatora izmantošanai.

Ir izstrādāts atbalsts funkcionalitātei, kas ļauj piekļūt programmas atmiņas apgabalam no pašas izpildāmās programmas. Lai to darītu, nepieciešams pielietot īpašu sintaksi specifisko funkciju izsaukumiem, turklāt veidot šāda veida darbības ar programmas atmiņu programmiski, nepieciešams ievērot piesardzību, plānojot pareizi izmantotos algoritmus, lai neapdraudētu sistēmas darbību, kā arī būtiski ieplānot algoritmus tā, lai atmiņas apgabali netiktu cikliski pārrakstīti bez liekas vajadzības, jo programmas atmiņas pārrakstīšanas reižu skaits ir ierobežots, to pārsniedzot nav garantēta aparatūras stabilitāte.

Veicot attālinātu programmas ielādi vai atjaunināšanu sistēmai, labā prakse ir šim nolūkam izveidot neatkarīgu moduli, kurš tiek glabāts programmas atmiņas adrešu apgabalā, kurš netiek programmiski pārrakstīts, šādā veidā izvairoties no sistēmas avārijām, kas varētu rasties kļūdainu programmas atmiņas apgabalu izmantošanas gadījumā. Turklāt lejupielādējot programmas datus, tos vispirms nepieciešams apkopot atsevišķā pagaidu atmiņā (piemēram, ārējā *FLASH* (elektroniski dzēšama un pārprogrammējama atmiņa patstāvīgai datu glabāšanai) atmiņas modulī), pārbaudīt saņemto datu kontrolsummas un tikai beigās pārrakstīt uz tam paredzēto programmas atmiņas īsto adrešu apgabalu.

2. TELEMETRIJA UN NOZARES SPECIFIKA

2.1. Telemetrija

2.1.1. Telemetrijas IT nozares specifika

Telemetrija ir informācijas tehnoloģiju nozare, kas saistīta ar datu vai mērījumu attālinātu pārsūtīšanu, izmantojot koptīklu. Šādu tehnoloģiju pielieto ēku vadības sistēmās, elektroiekārtu efektivitātes celšanas risinājumos un citās attālinātas kontroles vai monitoringa sistēmās. Telemetrijas risinājumi ievērojami atvieglo iekārtu uzraudzīšanas procesu, dod iespēju laicīgi novērtēt un plānot resursus, sniedz iespēju reālā laikā un attālināti iejaukties iekārtu vai pat sarežģītu sistēmu darbībā, kas nereti var būt izšķiroši, lai novērstu pēkšņi radušās problēmas. Bez telemetrijas šādas iespējas nebūtu, tā vietā nāktos sūtīt apkalpojošo personālu pēc nepielāgota grafika un tas savukārt būtu neelastīgs risinājums ar nelietderīgi izmantotiem cilvēku resursiem. Telemetriju plaši izmanto *vendinga* nozarē, kur izšķiroša ir spēja nekavējoties reaģēt uz saņemtajiem statusa ziņojumiem no iekārtām un plānot iekārtu klātienes apsekošanas grafiku vairākas nedēļas uz priekšu, turklāt ņemot vērā citus dinamiskus faktorus.

2.1.2. Telemetrija vending nozarei

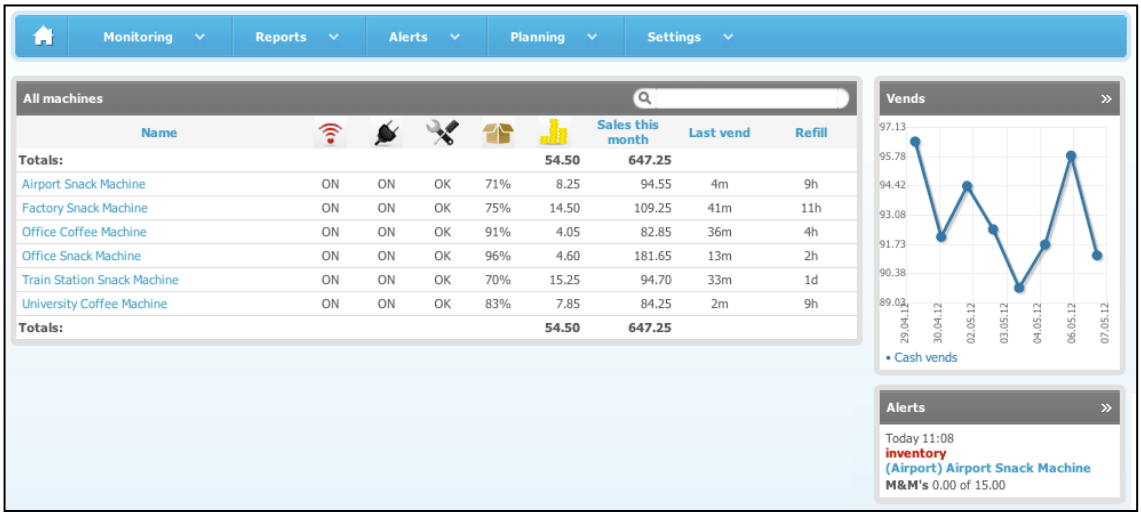
Vendings ir preču un pakalpojumu pārdošana, izmantojot tirdzniecības automātus, ar kuriem iespējams norēķināties uz vietas. Pirmo pasaulē zināmo tirdzniecības automātu radījis 1. gadsimta grieķu inženieris un matemātiķis Hero (no Aleksandrijas), šī iekārta apmaiņā pret monētu izsniegusi svētītu ūdeni. Iekārtas mehānisms sastāvējis no svirām un vārstiem, kas regulējuši ūdens plūsmu [4]. Industriālā laikmeta uzplaukumā tirdzniecības automātus ieviesa arī Anglijā un Amerikā, taču Latvijā tie kļuva sastopami tikai 90. gados, Latvijā biedrība „*Latvian Vending Association*" reģistrēta 2010. gada 30. novembrī [5].

Šāds tirdzniecības veids paver jaunas iespējas produktu un pakalpojumu tirdzniecības sfērā. Tas nodrošina ātrāku un ērtāku preču un pakalpojumu iegādi, izslēdz lieku komunikāciju ar pārdevējiem, sniedz iespēju iegādāties produktus pat darba vietā vai uzgaidāmā telpā. Vēl ērtāka pakalpojumu pirkšana iespējama ar mobilo lietojumprogrammu palīdzību. Tie ir tikai daži piemēri, toties visizplatītākais ir pārtikas produktu un cigarešu tirdzniecības aparātu klāsts, jo šis ir ļoti pieprasītas preces, kuras nepieciešams saņemt fiziski, to var aizstāt tikai ar elektronisku pirkumu, piemēram, interneta vietnē un piegādi līdz pasūtītājam, kas parasti saistās ar papildus līdzekļu un laika resursiem.

Pārtikas produktu *vendingam* ir raksturīga problēma, kas saistīta ar preču un produktu realizācijas termiņiem, uzturēšanas temperatūru un realizācijas apjomu novērošanu un prognozēšanu. Lai apsekotu veiktos darījumus, kontrolētu transakciju bilanci un saņemtu brīdinājumus par tirdzniecības automāta statusu, tiek uzstādīta telemetrijas ierīce. Tā tiek pievienota pie tirdzniecības automāta datu kopnes, lai nodrošinātu saziņu ar tirdzniecības automāta galveno kontroles moduli, kā arī ar citām pievienotajām iekārtām. Telemetrijas iekārtu parasti ievieto drošā vietā tirdzniecības automāta iekšpusē, pievienojot to arī pastāvīgam elektroenerģijas avotam, kura statuss arī tiek novērots, lai telemetrijas iekārta, kura nodrošināta ar bateriju, spētu nosūtīt ziņojumu par tirdzniecības automāta statusu strāvas pazušanas gadījumā. Galvenie telemetrijas uzdevumi tirdzniecības automātos ir veikto pirkumu atskaite, tirdzniecības automāta cenu saraksta un preču nosaukumu attālināta uzturēšana, kā arī atlikušo produktu skaita novērošana. Papildus tiek nodrošināta šāda funkcionalitāte:

- ziņojumu sūtīšana par automāta defektiem;
- apsekot automāta sastāvdaļu zādzību gadījumus;
- atskaite par naudas uzkrājumu automātā;
- brīdinājumi par apkalpojošā personāla vizīti pie automāta.

Telemetrijas iekārta parasti izmanto bezvadu savienojumus datu pārsūtīšanai tīklā, lai veiktu datu apmaiņu ar serveri, kurš apstrādā informāciju un nodrošina tīmekļa lietojumprogrammas grafisko lietotāja saskarni (sk. 2.1 att.). Piekļuve informācijas uzskaites sistēmai tiek piešķirta reģistrētiem klientiem, kas ir telemetrijas abonementi. Parasti tie ir tirdzniecības automātu izplatītāji un uzturētāji, kuri ieinteresēti attālināti apsekot tirdzniecības iekārtas, lai veiksmīgi plānotu savu grafiku iekārtu apmeklēšanai un preču papildināšanai.



2.1 att. ***Vending*** **telemetrijas klienta konta paraugs** [6]

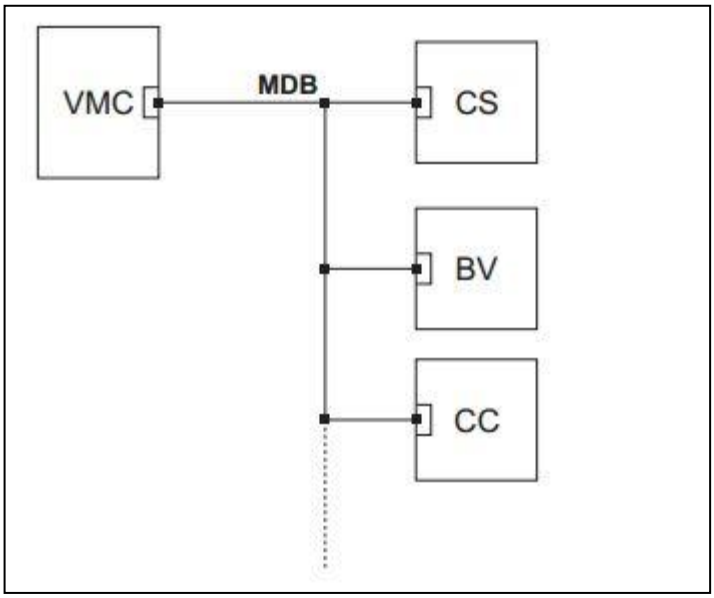
Telemetrija ir starptautiski pieejams pakalpojums, jebkuras valsts tirdzniecības automātu izplatītājs var uzraudzīt jebkurā valstī izvietotās, savam kontam piesaistītās, iekārtas. Tas nosaka tīmekļa lietojumprogrammas valodu atbalsta nepieciešamību lietotāja saskarnei, kā arī liek nodrošināt bezvadu tīkla savienojumu kvalitāti jebkurā valstī, tas savukārt atkarīgs no telekomunikāciju operatora izvēles un izcenojumiem.

Telemetrijas pakalpojumu sniedzējs ir ieinteresēts piesaistīt lielāko un izplatītāko *vending* iekārtu uzturētāju, taču lai to panāktu būtiski ir piedāvāt visizdevīgākos līguma nosacījumus, tādēļ telemetrijas iekārtām jābūt īpaši pielāgotām, kā arī to ražošanai jābūt pēc iespējas lētākai. Lai panāktu šādu rezultātu, nepieciešams speciāli pārdomāt nepieciešamo funkcionalitāti un uzprojektēt pielāgotu telemetrijas iekārtu. Protams, arī nepieciešams izveidot serveru atbalstu un izstrādāt ērtu, daudzfunkcionālu tīmekļa lietojumprogrammu sistēmas uzraudzībai un klientu kontu uzturēšanai..

2.2. Iegulto sistēmu risinājumi

2.2.1. Iegultās sistēmas uzdevums telemetrijā

Vending telemetrijas iegulto sistēmu galvenais uzdevums ir attālināta datu apmaiņa ar serveru sistēmu un saderīgums ar tirdzniecības automāta datu kopni. Iekārta tiek fiziski pieslēgta pie *VMC* (tirdzniecības automāta galvenais vadības modulis (iekārta), no angļu val. *Vending Machine Controller*) datu kopnes (sk. 2.2 att.), līdzīgi kā pārējās iekārtas, kas veido tirdzniecības automāta komplektāciju, piemēram, monētu maiņas automāts, banknošu lasītājs un citas iekārtas.



*2.2 att.**MDB* **slēguma paraugs** *vending* **automāta komplektējošām iekārtām** [7]

Šajā situācijā vispiemērotākais risinājums telemetrijas iekārtai ir iegulta sistēma, kas aprīkota ar mikroprocesoru, *GSM* moduli, papildu *FLASH* atmiņas moduli datu uzglabāšanai, neatkarīgu papildu elektroenerģijas avotu, antenu (atkarīgs no iekārtas novietojuma pret bezvadu savienojuma raidītāju) un papildu perifērijas iekārtām. Atšķirībā no personālās darbstacijas, šādas iekārtas ražošanas izmaksas ir mazākas, tā aizņem mazāk vietas, patērē mazāk elektroenerģiju – 3.3V (iekārta ir darbināma pat ar baterijas strāvas avotu), vieglāk uzstādāma un neizdala tik daudz siltuma, jo netiek darbinātas liekas komponentes.

Iegultā sistēma nodrošina pielāgotu funkcionalitāti un optimālas izmaksas, taču galvenais trūkums ir ierobežotie atmiņas resursi un mikroprocesora veiktspēja, taču tā pilnībā pietiekama pamata funkcionalitātes nodrošināšanai. Nepārtraukti nepieciešams rūpīgi plānot programmatūras arhitektūru, lai nodrošinātu gan atpakaļ saderīgumu, gan jaunpienākušo funkcionalitāti, efektīvi sadalot ierobežotos resursus un nepārslogojot mikroprocesoru.

Iegultās sistēmas rūpnieciskās ražošanas izmaksas ir ievērojami zemākas nekā personālās darbstacijas pielāgošana konkrētas iekārtas funkcionalitātes veikšanai, piemēram, mikroprocesora *PIC24F* cena, pērkot vairumā, ir četri līdz pieci ASV dolāri. Mikroprocesora modeļa izvēli nosaka dažādi faktori, piemēram, instrukcijas vārda garums, *RAM* (brīvpiekļuves atmiņa, no angļu val. *Random Access Memory*) izmērs, programmas atmiņas izmērs, ārējās saskarnes mikroprocesora tapiņu skaits, tapiņu atbalstītā funkcionalitāte no programmiskā viedokļa (digitālie ieejošie vai izejošie signāli, *ADC (*ierīce, kas pārveido ieejas analogo signālu diskrētajā kodā (ciparu signālā), no angļu val. *Analog-to-Digital Converter), UART, RS232* un pārtraukumu atbalsts, kā arī speciālo reģistru funkcionalitāte).

2.2.2. Iegultās sistēmas raksturlielumi

Svarīgi definēt kritērijus, pēc kuriem var savstarpēji salīdzināt konkurējošo iegulto sistēmu kvalitāti un funkcionalitāti. *Vending* telemetrijas nozarē iekārtas galvenie raksturlielumi ir šādi:

- fizisko savienojumu saderīgums ar tirdzniecības automātu modeļiem (datu kopnes savienojumi – spraudņu veidi);
- atbalstīto datu transportu protokolu skaits (piemēram, *MDB, DEX* (datu apmaiņas protokols audita un notikumu datu saņemšanai no tirdzniecības automāta, no angļu val. *Data EXchange*));
- atbalstīto pārsūtāmo datu formātu skaits.

Saderīgums starp dažādiem tirdzniecības automātu firmu modeļiem vienam un tam pašam fiziskam datu kopnes savienojumam un datu protokolam var atšķirties. Lai gan datu transporta protokolu veidi ir rakstiski dokumentēti, tomēr to realizācija dažādās iekārtās un

atšķirīgu ražotāju modeļos spēji atšķiras. Lai nodrošinātu atbalstu atsevišķiem tirdzniecības automātiem, nepieciešams testēt saderīgumu ar katru iekārtu un pat iekārtas modeli atsevišķi, lai pielāgotos ieviestajām atšķirībām no dokumentācijas vai arī ietvertu īpatnības, kuras nav skaidri definētas protokolu specifikācijā. Līdzīga situācija ir arī pārsūtāmo datu formāta atbalstam (piemēram, *EVA-DTS* (tirdzniecības automātu datu formāta standarts, no angļu val. *European Vending Association Data Transfer Standard*)), kas ir rakstiski pieejams, taču atsevišķas sadaļas iespējams dažādi interpretēt, tas savukārt arī nosaka nepieciešamību testēt saderīgumu ar katru modeli atsevišķi, lai uzkrātu pieredzi un spētu nodrošināt visaptverošu atbalstu plašam tirdzniecības automātu klāstam.

Iegultā telemetrijas iekārta parasti tiek aprīkota ar perifērijas ierīcēm – pogām un arī gaismas diodēm vai ekrānu, ja nepieciešams. Iegultās sistēmas iekārtas parasti satur speciālu pogu, kas paredzēta iekārtas pārstartēšanai, taču vēl specifiski nozarei pielāgota lietotājam paredzēta funkcionālā poga (vai vairākas). Funkcionālo pogu telemetrijā izmanto, piemēram, tirdzniecības automātu uzpildītājs, lai reģistrētu savu fizisko klātbūtni pie automāta, lai serveri tiktu informēti, ka produktu uzpilde ir veikta un var tikt veikta atskaites ģenerēšana. Līdzīgi poga var ietvert funkcionalitāti pilnas veikto transakciju uzskaites iegūšanai no tirdzniecības automāta galvenā kontroles modeļa, kas arī var tikt pārsūtīts uz serveri. Gaismas diodes nepieciešamas, lai lietotājs varētu noteikt iekārtas stāvokli – darbības režīmu, tīkla savienojuma statusu, kā arī, lai redzētu vizuālu atbildes signālu uz pēc pogas piespiešanas. Sarežģītākas iekārtas var tikt aprīkotas ar ekrānu, bet tas savukārt iespaido ražošanas izmaksas, aparātprogrammatūras sarežģītību un mikroprocesora modeļa izvēli.

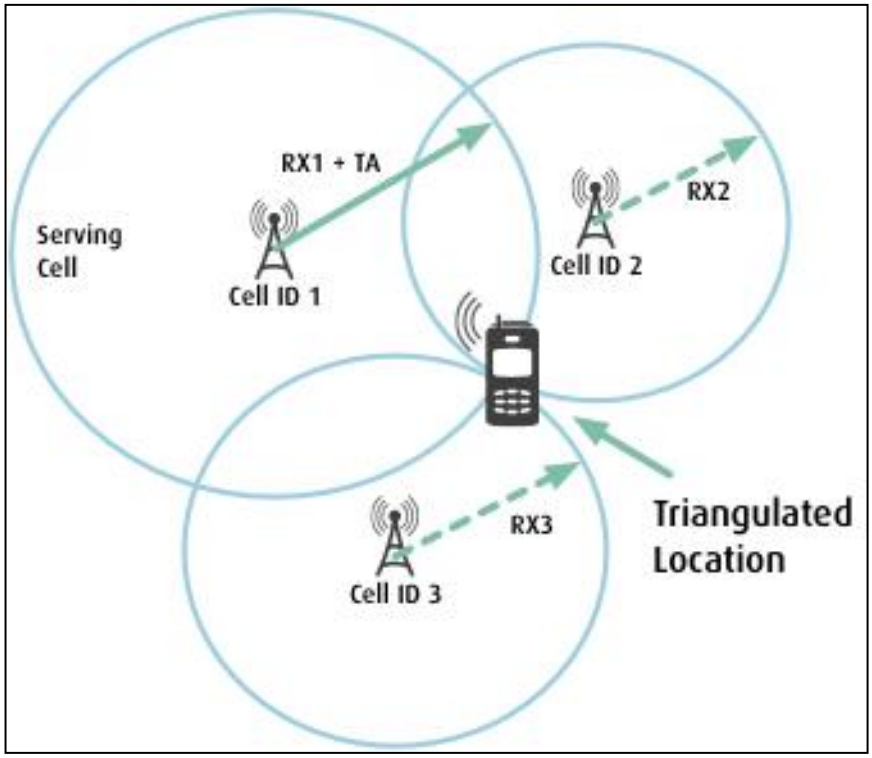
Papildus iekārta var tikt aprīkota ar skaņas signālu, kā arī ar temperatūras un citiem sensoriem. Pieslēgto sensoru skaits mikroprocesoram ir ierobežots, jo tikai dažas no ieejas signālu tapiņām atbalsta *ADC* režīmu vērtību nolasīšanai.

Būtiska ir fizisko spraudņu saderīgums ar tirdzniecības automātu datu kopni. Iegultās sistēmas arhitektūrai jābūt pielāgotai dažādu spraudņu veidu atbalstam, lai gan mikroprocesora tapiņu skaits ir ierobežots – nepieciešams atbalstīt dažāda veida dinamiski konfigurējamu režīmu atbalstu fiziski vienām un tām pašām mikroprocesora tapiņām. Piemēram, jābūt iespējai iekārtu saslēgt ar *DEX* vai *PROT-A* (tirdzniecības automātu datu apmaiņas protokols *(Exec vai Protocol-A*)) savienojumiem.

Iegultās sistēmas priekšrocība ir attālināta aparātprogrammatūras atjaunināšana, kas telemetrijas nozarē ir iespējama, pateicoties tīklu savienojumam ar serveri. Lai paredzētu šādu funkcionalitāti nepieciešams īpaši izstrādāt programmatūras projektējumu un izprojektēt programmas atmiņas pielietojumu, kā arī var būt nepieciešams nodrošināt papildus ārējās atmiņas resursus. Klātienes programmatūras atjaunināšana *vending* telemetrijas iekārtām var

izrādīties pārāk grūti plānojams un realizējams pasākums, jo tirdzniecības iekārtu izvietojums un apraudzīšanas grafiks var būt sagatavots ļoti neizdevīgs operatīvai šī uzdevuma veikšanai.

Telemetrijas iekārtu fiziskā novietojuma drošības problēma uzskatāma par atrisinātu, jo tirdzniecības automāti rūpnieciski veidoti slēgti, aprīkoti ar necauršaujamu stiklu, papildus starpsienu elektronisko komponenšu aizsardzībai un nodrošināti ar atslēgu mehānismu, kā arī veidoti smagnēji, lai apgrūtinātu to zādzības mēģinājumus. Neskatoties uz to, automāti tiek zagti, taču arī šajā sakarā var palīdzēt telemetrijas iekārta, ja tā izmanto telekomunikāciju operatora pakalpojumus, jo pēc *GSM* moduļa datiem iespējams noteikt aptuvenu atrašanās vietu, balstoties uz sakaru torņiem, kuri attiecīgajā brīdī ir uztverami un identificējami. Šādu tehnoloģiju sauc par triangulāciju (sk. 2.3 att.), tā tiek plaši pielietota mobilo tālruņu izsekošanai, neizmantojot *GPS* (globālā pozicionēšanas sistēma, no angļu val. *Global Positioning System*).



2.3 att. **GSM triangulācija** [8]

Datu pārsūtīšanas drošības nodrošināšana atkarīga no tīkla savienojuma veida. Izmantojot interneta pieslēgumu (ar fizisku savienojumu vai bezvadu risinājumu), nepieciešams rūpēties par datu aizsardzību, izmantojot šifrēšanas paņēmienus, kas saistīti ar apjomīgu matemātisko darbību veikšanu, kas savukārt ietekmētu mikroprocesora izvēli, ņemot vērā nepieciešamo ātrdarbību. Izplatīta ir metode ierīkot papildu mikroprocesoru tieši šifrēšanas funkcionalitātes veikšanai, taču tas sarežģī iegultās sistēmas arhitektūru un paredz programmatūrai nodrošināt paralēlu procesu sinhronizāciju. Savukārt, izmantojot mobilo

telekomunikāciju operatoru kā starpnieku datu pārsūtīšanai, tiek veiksmīgi atrisināta datu aizsardzības problēma, jo iekārtai nav pašai jārūpējas par datu drošību, kā arī nav nepieciešams rūpēties par fizisko datu savienojumu nodrošināšanu un pieejamību vietās, kur uzstādīti tirdzniecības automāti. Mobilā operatora datu pārsūtīšanas pakalpojumi ir dārgāki, taču stabilāki un pie telemetrijas iekārtas pārsūtāmo datu apjoma veiksmīgākais risinājums ir tieši *GSM* moduļa izmantošana ar piesaistītu mobilā operatora izdalīto *SIM* (GSM mobilo telefonu operatora viedkarte, no angļu val. *Subscriber Identity Module*) karti.

2.3. Ārējie faktori telemetrijas iekārtām

2.3.1. Telemetrijas sistēmas serveri

Iegultās sistēmas veido tikai daļu no kopējas sistēmas apjoma, arī telemetrijā liela nozīme ir serveru darbībai, ar kuriem iegultās sistēmas iekārtas sazinās un apmainās ar datiem. Nepieciešams plānot serveru saimi tā, lai tiktu risināta optimāla slodzes dalīšana, lai serveri vienmēr būtu sasniedzami no ārējā tīkla, lai savienojumu skaits tiktu paredzēts un plānots iepriekš, laikus novēršot savienojumu problēmas. Iegulto sistēmu atmiņas resursi ir ierobežoti, tādēļ ilgstošs datu savienojuma pārtraukums ar serveri var izraisīt datu iztrūkumu, ja novecojušie dati tiktu pārrakstīti ar nākamo atskaišu datiem, kamēr nav iespējams izveidot savienojumu ar serveri. Tādēļ iegulto sistēmu darbība ir cieši saistīta ar serveru saimniecību.

Jebkurai iegultās sistēmas aparātprogrammatūrai nepieciešams veiksmīgi sazināties ar serveri, tādēļ jebkurām izmaiņām gan iegultās sistēmas pusē, gan serveru pusē jābūt atpakaļ saderīgām, lai neradītu problēmas vecāku programmatūras moduļu darbībā.

2.3.2. Pakalpojumu iekārtas

Telemetrijas iegultās sistēmas tiešā veidā veic datu apmaiņu ar *VMC* moduli. Tas nosaka, ka iekārtas korekta darbība ir tiešā veidā atkarīga arī no paša tirdzniecības automāta darbības. Vienmēr pastāv risks, ka *VMC* var avarēt vai arī kļūdaini funkcionēt, šādu iespēju jau iepriekš jāparedz un jāveido atbalsts aparātprogrammatūrā, piemēram, sūtot atskaites ziņojumu serverim par notikušo problēmu, lai to vēlāk izskatītu un atrisinātu. Šādās situācijās nepieciešams attālināti reaģēt un atkārtot neveiksmīgo darbību (piemēram, uzkrāto transakciju vēstures iegūšanu no tirdzniecības automāta), taču, ja pēc vairākiem mēģinājumiem tiek konstatēta aparatūras problēma tirdzniecības automātam, nepieciešams nosūtīt operatoru, lai tas klātienē sagatavo atskaiti par veiktajām transakcijām, un, ja nepieciešams, veic visas nepieciešamās darbības, lai aparatūru atgrieztu normālā (stabilā) stāvoklī.

Pastāv arī tādi telemetrijas risinājumi, kur iegulto sistēmu ar tīkla pieslēgumu aizvieto speciāli apmācīta persona, kas veic regulārus tirdzniecības iekārtas fiziskus klātienes apmeklējumus un ievāc informāciju par automāta veiktajām transakcijām, izmantojot pārnēsājamo zibatmiņu vai līdzīgu sistēmu, kuru pēc tam dienas beigās nogādā serveru telpā un lejupielādē informāciju no šī datu nesēja. Tas, protams, ir ļoti novecojis apkalpošanas modelis.

2.3.3. VM izplatītāji un operatori

Tirdzniecības automātu izplatītāji iepērk aparatūru no ražotājiem un izvieto preču pārdošanai izdevīgās vietās, pielāgojoties pieprasījumam un mērķauditorijai. Galvenais uzdevums ir rūpēties par šo automātu stāvokli, apsekojot atlikušo preču daudzumu, ievācot sakrātās monētas un banknotes no tirdzniecības automātu komplektējošajām iekārtām. Operatori, kas veic automātu apsekošanu, ne vienmēr ir uzticami, tādēļ ir svarīgi zināt ne tikai to, kāds naudas apjoms ir atgriezts automāta izplatītājam ar šī personāla starpniecību, bet arī precīzu summu, cik pircēji ir ievietojuši nomētu un banknošu lasītājā. Jo teorētiski pastāv iespēja, ka pircēji var būt ievietojuši vairāk naudu nekā nepieciešams, taču var gadīties, ka persona, kas naudu iekasē no automāta, daļu no tās ir piesavinājusies nelegāli darba laikā. Telemetrijas sistēmas, kas iegūst informāciju par veiktajām transakcijām no tirdzniecības automāta centrālā vadības moduļa un informāciju par ievietotiem naudas līdzekļiem no monētu un banknošu lasītāja, sniedz pilnu atskaites informāciju par to, cik tieši naudai jābūt nonākušai līdz izplatītājam.

Pastāv arī problēma, ka tirdzniecības automātus apkalpojošie operatori nemaz neveic objektu apgaitu, šim nolūkam paredzēta poga uz telemetrijas iekārtas korpusa, lai brīdī, kad tirdzniecības automāts tiek apkopts, operators varētu reģistrēties, un lai serveru sistēma varētu precīzi reģistrēt laiku, kad apkope veikta.

2.3.4. Norēķinu iespējas un tehnoloģijas

Tirdzniecības automātu preču un pakalpojumu izmantotāji ir ieinteresēti brīvi izvēlēties ērtāko norēķinu veidu, taču novecojusī aparatūra galvenokārt paredzēta norēķiniem ar monētām un banknotēm. Izplatītāji ir ieinteresēti turpināt lietot jau iepirktos un novecojušos tirdzniecības automātu modeļus, taču inovatīvu risinājumu ieviešana var tikt veikta, izmantojot telemetrijas iekārtu funkcionalitāti.

Tirdzniecības automāts var tikt aprīkots ar *NFC* (bezkontakta datu apmaiņas risinājums iekārtām, no angļu val. *Near Field Communication*) lasītāju (vai uzlīmi), kas pieslēgts telemetrijas iekārtai. Tādā veidā nodrošinot pircēju identifikācijas iespēju, piemēram,

izsniedzot bezmaksas produktus ierobežotā skaitā konkrētajam pircējam. Ja izmanto pieeju, ka tirdzniecības automāts aprīkots ar *NFC* uzlīmi, tad iespējams veidot sistēmu, ka pircējs izmanto sertificētu mobilo ierīci ar apstiprinātu lietojumprogrammu, lai transakcija tiktu veikta ar mobilās iekārtas starpniecību, izmantojot *NFC* uzlīmi kā tirdzniecības automāta identifikāciju, bet produkta vai pakalpojuma izvēle notiek mobilajā aplikācijā (var tikt piesaistīta „virtuālajam maciņam"). Veiksmīga darījuma rezultāta serveris nosūta komandu uz tirdzniecības automātu konkrētā darījuma uzsākšanai, lai pircējs saņemtu preci vai pakalpojumu. Trūkums šai tehnoloģijai ir tas, ka tikai ierobežots skaits pieejamo mobilo tālruņu atbalsta *NFC* tehnoloģiju.

Mūsdienās populāra kļuvusi arī *QR* (matricas svītru kods, jeb divdimensionāls kods, no angļu val. *Quick Response Code*) koda tehnoloģija: divdimensionāls standartizēts simbols, kuru izstrādājis „*Denso wave*" 1994. gadā ar mērķi „kods, kas viegli nolasāms lasītājam" [9]. Arī šādu tehnoloģiju iespējams izmantot līdzīgi kā *NFC* uzlīmi uz tirdzniecības automāta, vienīgi nav nepieciešama īpaši aprīkota mobilā iekārta, jo *QR* koda nolasīšanai nav nepieciešams speciāls papildu aprīkojums, bet gan plaši pieejamā foto tehnoloģija, kas pieejama lielākajai daļai mobilo telefonu aprīkojumā, protams, papildinot iekārtu ar datu savienojuma pieslēgumu, lai lejupielādētu informāciju, kas piesaistīta nolasītajam simbolam.

Protams, ierasts norēķinu paņēmiens mūsdienās ir *SMS* (teksta ziņojumu sūtīšanas serviss, no angļu val. *Short Message Service*) maksājumi, arī tos iespējams izmantot tirdzniecības automātos pateicoties telemetrijas iekārtām, kas paver iespēju attālināti kontrolēt tirdzniecības automātu un šādi apstiprināt pirkumu, par ko iepriekš kāds ir norēķinājies.

3. IEKĀRTAS TESTĒŠANA

3.1. Iekārtas plates testēšana

Iegultās sistēmas iekārtas elektrisko slēgumu testēšana ir neatņemama iekārtas projektēšanas fāzes sastāvdaļa, taču arī rūpnieciskās ražošanas laikā ir aktuāli pārbaudīt katras iekārtas plati atsevišķi, lai konstatētu rūpnieciskos defektus.

Plates projektēšanas laikā elektriskie plates savienojumi tiek pārbaudīti ar oscilogrāfu, tiek rūpīgi sekots līdz signālu stiprumam un tiek noteikti cēloņi iespējamām signālu nobīdēm vai traucējumiem. Atklātās nepilnības tiek novērstas un izlabotas projektējuma modelim (slēgumu shēmai), lai nākamajā ražošanas kārtā (vai prototipu sērijā) iekļautu apstiprināto labojumu.

Kad projektēšanas fāze beigusies, ikviena ražotā iekārta pirms lietošanas jāpārbauda, apsekojot uzlodēto komponenšu darbību, jo vienmēr pastāv risks, ka plate var būt bojāta.

Ja rūpnīcā tiek noteikts, ka plate ir bojāta, tad to noraksta uz pārbaudi, kur speciālists individuāli noskaidro, vai nav iespējams novērst bojājumu, veicot rokas vadības pārlodēšanu. Parasti iekārtas (uzlodētās plates) elektrisko savienojumu testēšanu veic laika posmā starp plates izgatavošanu un tapiņu lodējumu pārklāšanu ar laku. Šo testēšanu veic speciāli pielāgota aparātprogrammatūra, kas tiek ielādēta mikroprocesorā ar mērķi pārbaudīt katru no fiziskajiem moduļiem (plates komponentēm) atsevišķi.

Piemēram, digitālo kontaktu spraudņa ligzdā testēšanas laikā ievieto speciāli izveidotu spraudni, lai ikviena datu izejošā tapiņa tiktu savienota ar ienākošo tapiņu pa pāriem, kas nodrošina atbalss efektu, ja iekārta mēģina izsūtīt datus. Gadījumā, ja atbalss dati netiek saņemti tūlīt pēc to izsūtīšanas, tad tiek konstatēts plates bojājums uz šis datu maģistrāles.

Savukārt atmiņas moduli testē, ierakstot atmiņā noteiktā apgabalā datus, kurus pēc tam secīgi arī nolasa, vienlaikus pārbaudot atmiņas kļūdu reģistrus un nolasīto datu vērtību. Tiek pārbaudīti dažādi atmiņas reģioni un pēc iespējas dažādākas atmiņas moduļa funkcionalitātes iespējas un speciālo reģistru vērtības.

Plates testēšana ietver arī enerģijas avota sprieguma mērīšanu, kā arī rezerves enerģijas avota slēgumus, kas mēdz būt kontrolējami ar vadāmiem slēdžiem. Piemēram, bateriju kā enerģijas avotu var gan pieslēgt, gan atslēgt, izmantojot aparātprogrammatūrai pieejamos speciālo funkciju reģistrus, turklāt iespējams sekojot līdz strāvas sprieguma izmaiņām, ja tiek iesaistīta ienākošā mikroprocesora analogā signāla tapiņa. Iegultās sistēmas darbībā šai funkcionalitātei ir liela nozīme, jo baterija kalpo kā rezerves enerģijas avots, lai iekārta spētu sūtīt serverim ziņojumu, kad tirdzniecības automātam pārtraukta elektrības padeve, turklāt

nepieciešams arī ziņot, kāds ir tā brīža atlikušais baterijas enerģijas līmenis telemetrijas iekārtas darbības ilguma prognozēšanai.

Svarīgi iekļaut arī *GSM* moduļa elektrisko slēgumu un paša moduļa funkcionalitātes testus, jo šī komponente ir ļoti jūtīga pret pievadītā sprieguma stiprumu, kas var radīt funkcionalitātes nestabilitāti, traucējumus un citas blakusparādības. Tādēļ nepieciešams testu laikā ievietot *SIM* karti, lai *GSM* moduli notestētu ar dažādas sarežģītības pakāpes funkcionalitātes testiem, kas ietver veiksmīgu pāreju uz iespējotu moduļa režīmu, *SIM* kartes noteikšanas funkcionalitāti, datu iegūšanu no *SIM* kartes adrešu grāmatiņas ierakstiem, reģistrēšanos tīklā un uztveramo torņu atpazīšanu un citiem testiem. Ievietotā *SIM* karte var nebūt vēl aktivizēta, tādā gadījumā nav iespējams notestēt datu savienojuma izveidi ar serveri (rūpnieciskās testēšanas laikā *SIM* kartes ievietošana un izņemšana ievērojami paildzinātu ražošanas procesu, kas būtiski sadārdzinātu ražošanas izmaksas, taču ideālā gadījumā tas būtu ieteicams).

Testu laikā nepieciešams pārliecināties arī par perifērijas iekārtu, gaismas diožu un skaņas signāla darbību. Šie testi nav veicami pilnībā automatizēti, jo nepieciešams cilvēka novērtējums vai citu sensoru sistēma, kas novērtētu to darbību.

3.2. Mikroprocesora elektriskā slēguma testēšana

3.2.1. Elektriskā slēguma komponentes

Mikroprocesora tapiņu slēgumi un plates komponenšu sastāvs var atšķirties iekārtas projektēšanas un izstrādes laikā, kad iespējams pievienot atsevišķu atkļūdošanas moduli, kas paredzēts speciālam reāla laika atkļūdošanas režīmam, kura laikā izpildāmo instrukciju kopums tiek regulēts no personālās darbstacijas. Šādā veidā aparātprogrammatūras izstrādātājs spēj apturēt mikroprocesora darbību jebkurā momentā un spēj noteikt reģistru saturu, kā arī citas statusa vērtības, kas noder atkļūdošanas procesā. *PIC* mikroprocesoru atkļūdošanai paredzēts izmantot *MPLABX* vidi, kas pieejama bez maksas, atbalsta apmēram astoņus simtus dažādu 8, 16 un 32 bitu *Microchip* mikroprocesoru modeļus, iekļaujot bagātīgu grafisku teksta rediģēšanas funkcionalitāti un simulācijas rīku aparātprogrammatūras darbināšanai [10].

Mikroprocesoru ražotāji piedāvā speciāli komplektētas sistēmu plates ar atkļūdošanas modeļiem, lai demonstrētu produkta iespējas un piedāvātu ērtu veidu, kā programmatūras izstrādātājs var izmēģināt ieplānotos konceptus kontrolētos apstākļos.

3.2.2. Enerģijas avotu slēgumi

Enerģijas avota elektrisko slēgumu testēšana platei ir atbildīgs process, jo iekārtas funkcionalitāte un precizitāte ir atkarīga no pieliktā sprieguma līmeņa, kuram jābūt pieļaujamajās normas robežās. Labā prakse ir reaģēt uz kritiski zema sprieguma līmeņa iestāšanos, kontrolētā veidā atspējojot sistēmu, veicot svarīgo datu pieglabāšanu un korektu atsevišķu plates moduļu atspējošanu, kas var saturēt pat vairākus izpildes soļus ar konkrētiem laika intervāliem, piemēram, *GSM* moduļa korekta atspējošana.

Iekārtas plate var saturēt sarežģītus slēgumus, kas paredzēti aparātprogrammatūras kontrolei enerģijas avota izvēlē – ārējā barošana un baterija, jeb tikai baterija. Ir svarīgi notestēt, vai visi plates slēdži pareizi funkcionē - vai iespējams pilnībā atvienot iekārtu no katra no enerģijas avotiem, vai iespējams atvienot abus enerģijas avotus reizē. Nepieciešams arī novērtēt sprieguma līmeni katrā no šim kombinācijām, nosakot, vai plates lodējumi nav saplūduši kopā, vai nav pārrāvumi elektriskajā ķēdē un vai baterijas spraudnis funkcionē atbilstoši.

3.3. Plates komponenšu testēšana

3.3.1. Perifērijas testēšana

Speciālu testēšanas pieeju nosaka plates perifērijas komponentes. Rūpnieciskās ražošanas laikā ir iespējams izmantot robotu pogu nospiešanai, gaismas sensorus diožu testēšanai un skaņas sensoru skaņas signāla pārbaudei. Mazos apjomos veicot ražošanu pietiek ar cilvēkresursu - darbinieku, kam uzticēts veikt šos testus, kā arī vadu pievienošanu un atvienošanu sagatavošanai testiem.

3.3.2. Komplektācijas testēšana

Rūpnieciskās ražošanas posmā, kad plate ir veiksmīgi notestēta, lodējumu vietas tiek noklātas ar laku. Tad plate tiek ievietota sagatavotajās korpusa detaļās, šajā gadījumā korpuss sastāv no divām aptverošām daļām un vēl kustīga plastmasas elementa – pogas. Korpusa ģeometrija ir precīzi izplānota, lai tā neaizsegtu kontaktu ligzdas un lai gaismas diodes precīzi iegultos paredzētajās vietās.

Pēc korpusa salikšanas nepieciešams veikt pēdējo iekārtas testu – pārbaudīt, vai visas perifērijas iekārtas ir paredzētajās vietās, vai korpuss nav brāķēts, vai, ievietojot plati korpusā, tā nav bojāta. Šo testu veic ar vizuālu novērtējumu, kā arī darbinot programmatūru, kas ielādēta platē, un veicot darbības ar iekārtas perifērijas iekārtām. Piemēram, nospiežot pogu, tiek uzsāki testi, bet testu rezultāts tiek izspīdināts uz gaismas diodēm. Konkrētajā gadījumā

iekārtā ievietota *SIM* karte tiks aktivizēta tikai iekārtas pārdošanas brīdī, bet citādi šo testu laikā būtu iespēja nosūtīt uz serveri datu paketi, kas apstiprina iekārtas darbību. Kad tiks aktivizēta *SIM* karte, iekārta būs jau iepriekš reģistrēta servera datubāzē un tik atpazīts ziņojumam pievienotais identifikācijas numurs.

4. IZSTRĀDĀTĀ SISTĒMA IEKĀRTAS TESTĒŠANAI

4.1. Iekārtas rūpnieciskās testēšanas programmatūra

Autors patstāvīgi izplānoja un izstrādāja iegultās sistēmas rūpnieciskās testēšanas programmatūru, kas ietver personālās darbstacijas un arī iegultās sistēmas programmnodrošinājumu. Galvenais mērķis bija radīt sistēmu, ar kuras palīdzību tiktu veikti paštestēšanas uzdevumi visām saražotajām iekārtām, lai pārliecinātos par lodējumu kvalitāti un plates komplektējošo modeļu savienojumiem.

Autoram nebija iepriekšējas pieredzes aparātprogrammatūras izstrādei, kas paredzēta lietošanai mikroprocesoru iegultajai sistēmai. Iegultās sistēmas pirmkoda apjoms, kas paredzēts iekārtas testēšanas komandu protokola apstrādei un testa funkcionalitātes izpildei iekārtas komponentēm, sastāda 1000 pirmkoda rindas programmēšanas valodā *C*. Izstrādātā darbstacijas daudz-pavedienu lietojumprogramma ar grafisku lietotāja saskarni testēšanas moduļa vadībai, izmantojot *USB* datu kopni, sastāda apmēram 5000 pirmkoda rindiņu programmēšanas valodā *C#*, neskaitot pielāgoto *USB HID* (atora perifērijas (USB) iekārtas apzīmējums, no angļu val. *Human Interface Device*) bibliotēku, kas arī atbalsta daudz-pavedienu izpildes režīmu. Autors vēlas atzīmēt, ka aparātprogrammatūras projektējums tika projektēts divreiz, turklāt ietverot programmas apjoma samazināšanu un funkcionalitātes izmaiņas, kas saistītas ar testēšanas komandu apmaiņas protokola pilnveidošanu.

Galvenās prasības testēšanas sistēmai bija grafiskas lietojumprogrammas izveide personālai darbstacijai, kas vizuāli attēlotu interaktīvu lietotāja saskarni perifērijas testu laikā, kā arī demonstrētu paralēli notiekošo iekārtas moduļu testu statusu, iekārtas testēšanas laikā iegūto informāciju, kas identificē testējamo iekārtu, tajā skaitā unikālo iekārtas sērijas numuru, kā arī datus no ievietotās *SIM* kartes adrešu grāmatiņas. Kā arī iekārtas testēšanas rezultāta atskaiti nosūtītu uz serveri, izmantojot teksta marķēšanas valodu *XML* (paplašināmā iezīmēšanas valoda, no angļu val. *Extensible Markup Language*), testu rezultātā nepieciešams uzskatāmi norādīt, vai iekārta pozitīvi izturējusi visus testu uzdevumus, turklāt testēšanas laikā nepieciešams vizuāli ilustrēt darbības, kuras jāveic lietotājam, kurš testē iekārtu: iekārtas kontaktu pievienošanas secība, perifērijas lietošanas pamācība. Papildus uzlikts nosacījums, ka lietojumprogrammas saskarnei jāatbalsta vairākas valodas, jo programmatūru paredzēts lietot dažādās valstīs.

Tika izvēlēta secīgu dialogu lietojumprogrammas grafiskā saskarne, jo pēc pārrunām ar rūpnieciskās ražošanas nodaļās uzraugu, kļuva skaidrs, ka cikliski atkārtojošā darba specifika nosaka pēc iespējas izvairīties no pasīvas lēmumu pieņemšanas (lietotāja iejaukšanās tikai problēmas situāciju novērošanas gadījumā), bet gan nemitīgi skaidri un uzskatāmi intervēt

lietotāju par iekārtas statusu (gaismas diožu, perifērijas un skaņas signālu testēšana), lai pēc iespējas izslēgtu cilvēcisko faktoru – lietotāja neuzmanību, testējot simtiem iekārtu vienas dienas laikā. Tika kopīgi nolemts, ka rūpnīcas darba videi vispiemērotākais risinājums darbstacijas lietojumprogrammas pogu īsinājumam būtu taustiņš „*ENTER*", taču retākajos gadījumos, kad jāsniedz negatīvs novērtējums, lietojumprogrammas grafiskās saskarnes poga, kas signalizēs kļūdu vai neatbilstību.

Papildus tika pieņemts ierosinājums lietojumprogrammu uzstādīt darbstacijai ar portatīvo svītru kodu skeneri, kuru plānots izmantot, lai nolasītu rūpnīcas teritorijā piešķirto plates identifikatoru, kas būtu noderīgs testu laikā, lai spētu identificēt iekārtas plati pat tad, ja saziņa ar iekārtu neizdotos nemaz. To paredzēts izmantot kā lietotāja ievades lauka informācijas aizpildītāju pirmajā iekārtas testēšanas solī.

4.2. Darbstacijas lietojumprogramma

4.2.1. Uzdevuma nostādne

Lietojumprogrammai jābūt intuitīvai, jāparedz, ka lietotājam var būt uzmanības traucējumi dēļ apkārt notiekošā rūpnīcas vidē, jābūt paredzētām visām iespējamām izņēmuma situācijām, ar kādām lietotājs var saskarties programmas lietošanas laikā, sevišķi, ja lietojumprogrammu izmantotu nekorekti un ignorētu nosacījumus.

Galvenā nostādne – lietojumprogrammas grafiskajai saskarnei jābūt ērti uztveramai, tekstiem viegli salasāmiem un vienkārši formulētiem. Programmai jāveic pakāpeniska lietotāja intervēšana, pa to laiku paralēli jāveic pēc iespējas ātrāka un organizētāka testu uzvedumu apstrāde un datu apmaiņa ar testējamo iekārtu, jo rūpnieciskā testēšana ir maksas pakalpojums, kura izmaksas mērogojamas pret pavadīto laiku, testējot iekārtu.

Pēc apspriedes ar rūpnīcas darbiniekiem, tika kopīgi izlemts, ka iekārtu testēšana notiks secīgi – pa vienai iekārtai pēc kārtas, kas ir būtisks nosacījums lietojumprogrammas arhitektūras izstrādei. Iekārtas pievienošanas un atvienošanas komandas ar vizuāliem palīgmateriāliem demonstrēs lietojumprogramma, kad tas būs nepieciešams.

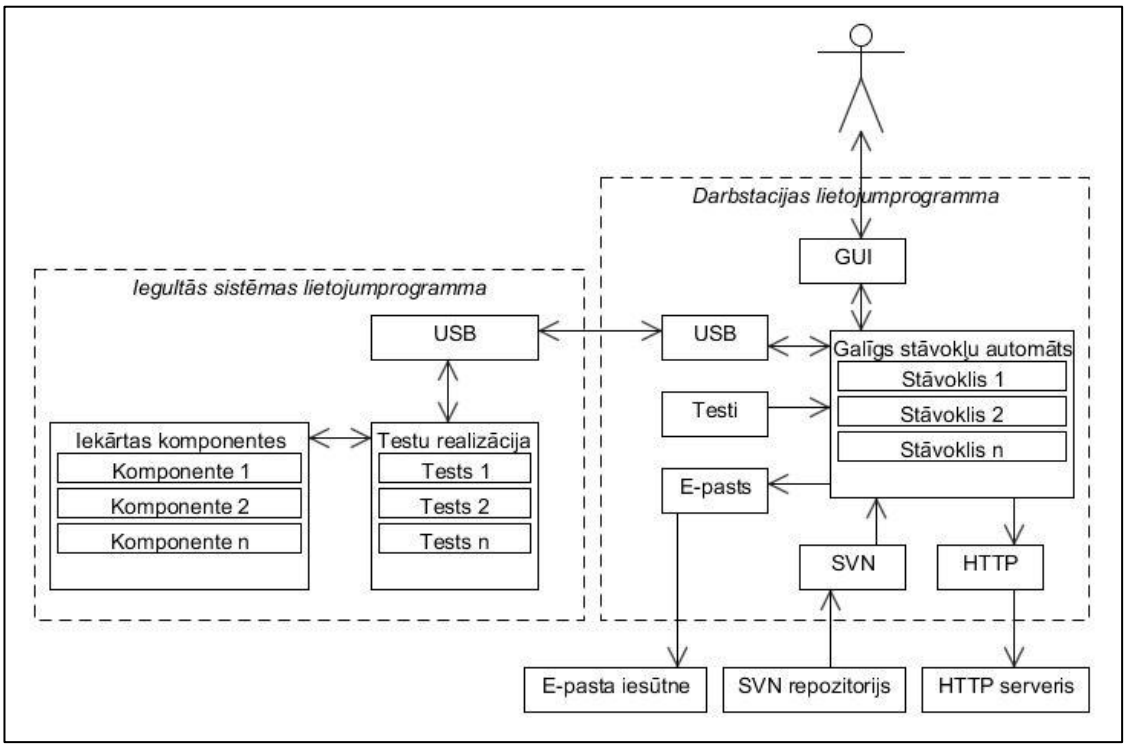
4.2.2. Platforma

Lietojumprogrammu paredzēts lietot *Windows XP* operētājsistēmā, darbstaciju plānots aprīkot ar interneta pieslēgumu un nepieciešamo *USB* savienojumu iekārtas testēšanai. Autors izvēlējās objektorientēto programmēšanas valodu *C#*, izmantojot *.Net* ietvaru un bibliotēkas, kā arī iepriekš speciāli izstrādāto *USB* datu apmaiņas bibliotēku, kas izstrādāta iegultās

sistēmas komunikācijas atbalstam ar personālo darbstaciju, izmantojot iepriekš saskaņotus *USB* iekārtu (*HID*) identifikatorus.

4.2.3. Lietojumprogrammas moduļi

Lietojumprogramma satur vairākus neatkarīgus moduļus, katrs no tiem nodrošina specifisku funkcionalitāti. Tajā skaitā ieviests grafiskās lietotāja saskarnes tekstu atbalsts vairākām valodām, kuras iespējas dinamiski pievienot pēc vajadzības kā atsevišķus resursus, turklāt atbalstīta iespēja jebkurā lietojumprogrammas izpildes momentā pārslēgties starp jebkuru no šim valodām. Autors uzsver, ka lietojumprogrammas pirmkods ir veiksmīgi sadalīts atsevišķos moduļos (sk. 4.1 att.), izmantojot objektorientētas programmēšanas valodas (*C#*) priekšrocības, šāda metode izmantota arī grafisko komponenšu veidošanā.

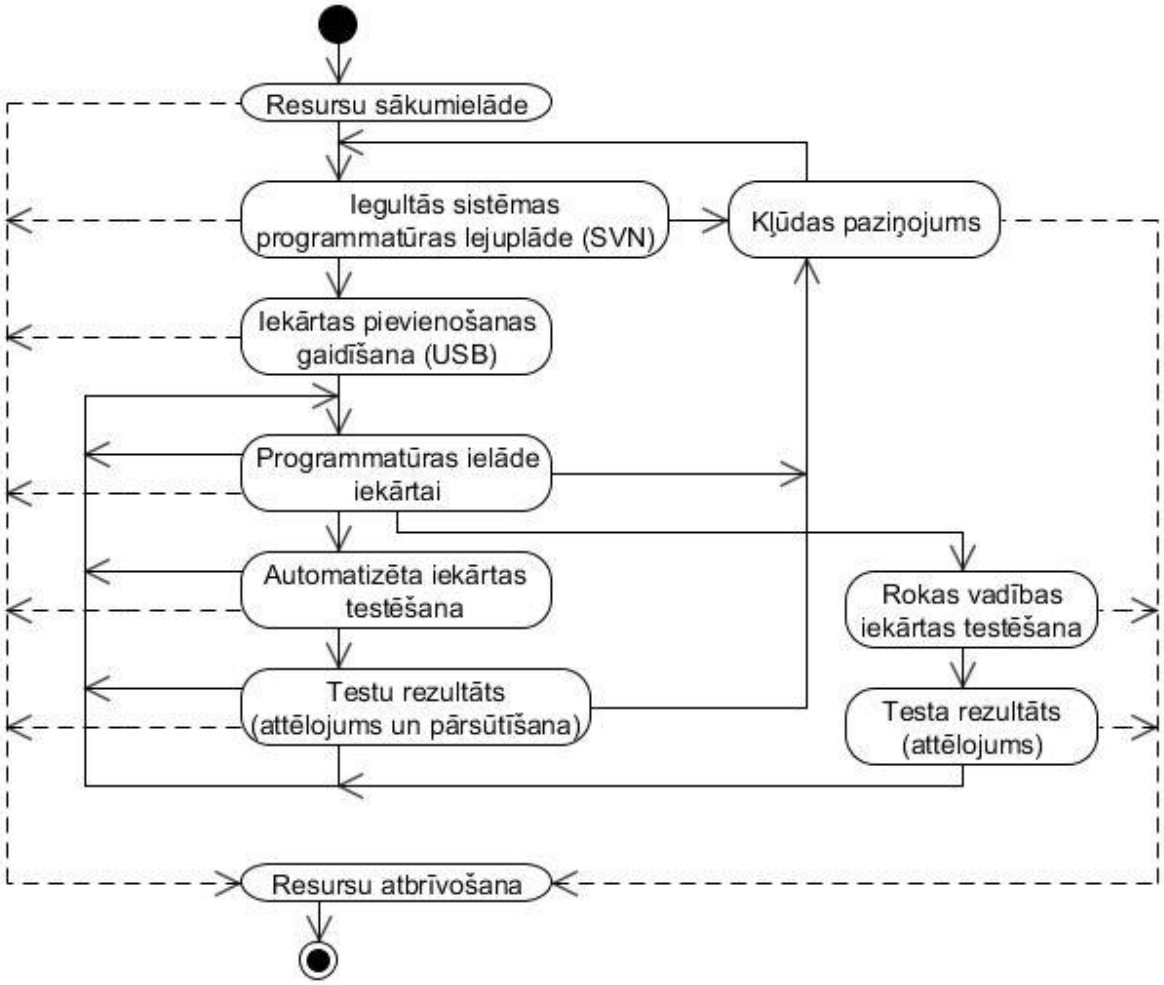


4.1 att. **Lietojumprogrammas moduļi**

4.2.4. Lietojumprogrammas vadības modulis

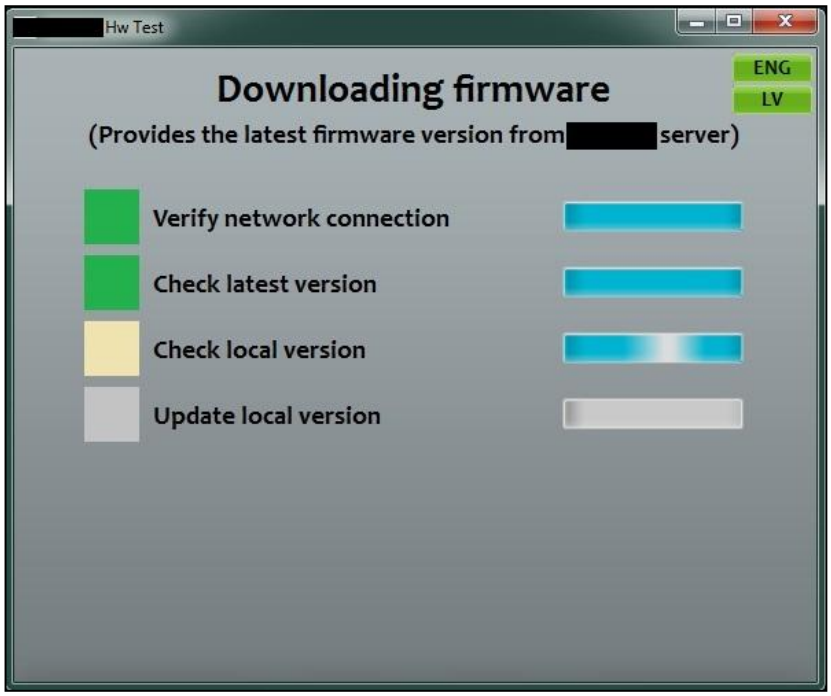
Galvenais modulis, kas ietver programmatūras loģiku veidots pēc galīga stāvokļu automāta projektējuma, kas realizēts objektorientētā pieejā, kur katrs stāvoklis ir definēts kā klase ar tai realizētajām metodēm, turklāt izmantojot klašu mantošanas paņēmienu, novērsta programmas pirmkoda dublēšanās. Protams, izveidots mehānisms, kas katram stāvoklim liek izpildīt funkcionalitāti ieejot stāvoklī un pametot to. Šis modulis nosaka lietojumprogrammas stāvokļus un definē transakcijas (stāvokļu pārejas), turklāt tas saista visus pārējos

lietojumprogrammā realizētos moduļus, kā arī veic kļūdu situāciju pārtveršanu un apstrādi.



4.2 att. **Lietojumprogrammas vadības plūsmas diagramma**

Lietojumprogrammai uzsākot darbību, notiek resursu datņu ielāde, tas ietver dinamisko bibliotēku ielādi un grafisko objektu tekstu un attēlu sagatavošanu darbam (sk. 4.2 att.). Nekavējoties lietojumprogramma uzsāk pēdējās pieejamās iegultās sistēmas programmatūras lejupielādi no *SVN* repozitorija (šī procesa laikā lietotājam nav iespēja ietekmēt sistēmas darbību, ja neskaita lietojumprogrammas pilnīgu aizvēršanu) (sk. 4.3 att.).

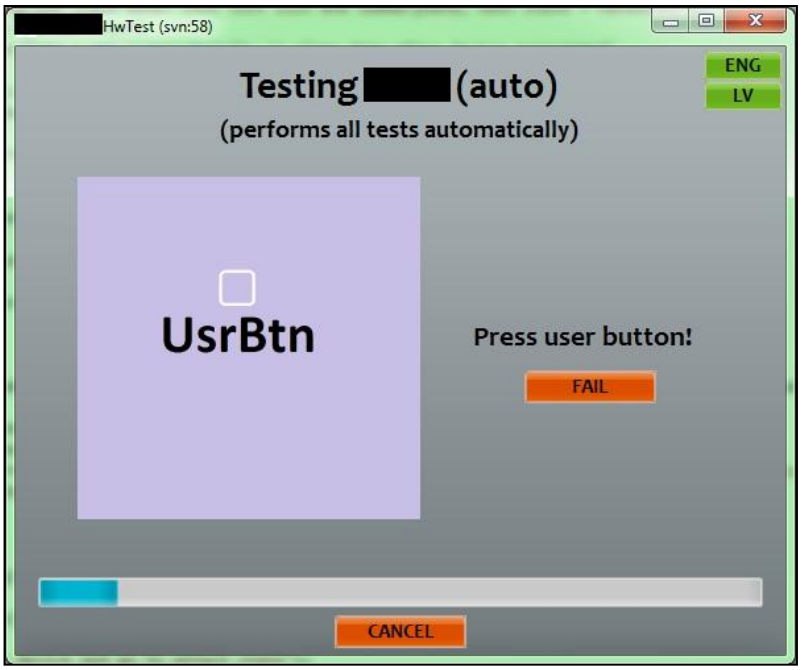


4.3 att. **Lietojumprogrammas *SVN* datu lejupielādes ekrānuzņēmums**

Pēc veiksmīgas lejupielādes lietojumprogramma piedāvā lietotājam ievadīt testējamās iekārtas plates identifikatoru (svītru koda skenerim paredzēts lauks), kā arī tiek gaidīts *USB* ierīces (konkrētā *HID* identifikatora, kas sastāv no produkta un izplatītāja numuri) pievienošanas notikums (sk. 4.4 att.).

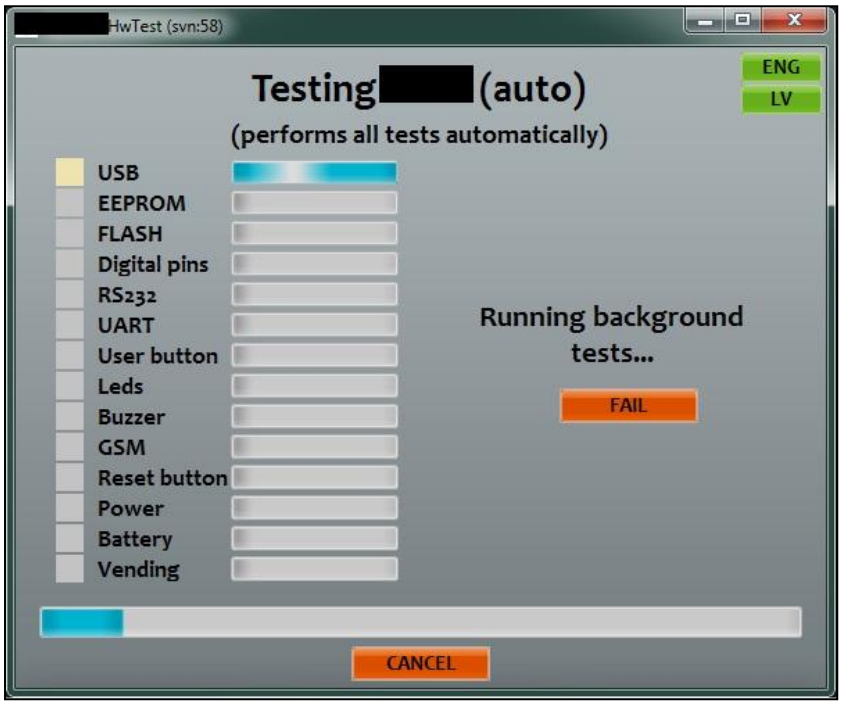
4.4 att. **Lietojumprogrammas iekārtas pievienošanas ekrānuzņēmums**

Kad pievienota kāda no atbalstītajām iekārtām (iegultās sistēmas testējamā iekārta), pēc noklusējuma tiek uzsākta iegultās sistēmas programmatūras ielāde (kas iepriekš tika lejupielādēta no *SVN* repozitorija). Tad tiek veikta automatizēta iekārtas testēšana, kura sastāv no specifiskiem apakš stāvokļiem, kuri tiks aplūkoti atsevišķi. Automatizētas testēšanas laikā tiek vizuāli attēloti paziņojumi, kas papildina iekārtas testēšanu ar lietotāja intervēšanu par iekārtas komponentu darbību, kas ietver lietotāja novērtējumu, piemēram, skaņas signāla un diožu darbības novērtēšana, apstiprinot ar „jā" vai „nē" piedāvāto apgalvojumu, kas ilustrēts ar testējamās iekārtas komponentes attēlu (sk. 4.5 att.).



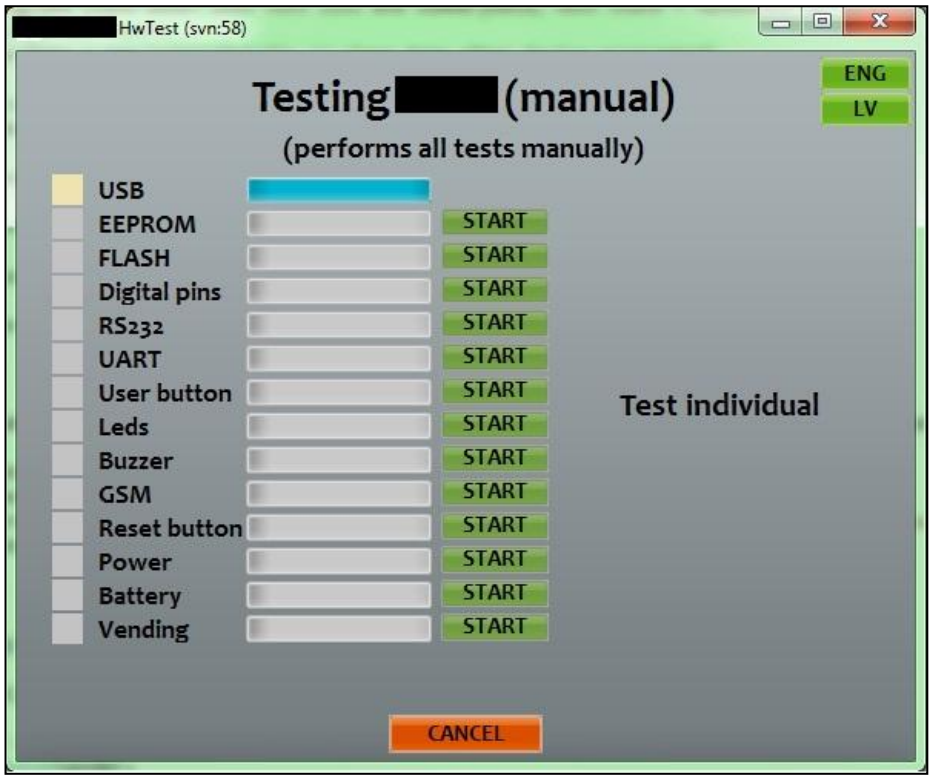
4.5 att. **Lietojumprogrammas lietotāja dialoga ekrānuzņēmums (tajā redzams pagaidu attēls iekārtas perifērijas fotouzņēmuma vietā)**

Paralēli lietotāja saskarnes darbībai tiek izpildīti citi iekārtas testi, kuru statusu iespējams vērot reālā laikā ar progresa indikatoru palīdzību (sk. 4.6 att.).



4.6 att. **Lietojumprogrammas paralēlo testēšanas procesu statusa indikatoru ekrānuzņēmums**

Pēc visu testu izpildes lietojumprogramma attēlo gan kopējo rezultātu (slēdziens par iekārtas atbilstību), gan katra testa rezultātu, kas sniedz informāciju par atklātajiem iekārtas plates vai citiem komponenšu defektiem. Brīdī, kad tiek attēlots rezultāts, lietojumprogramma nosūta atskaiti uz *HTTP* (hiperteksta pārsūtīšanas protokols, no angļu val. *Hypertext Transfer Protocol*) serveri. Atvienojot iekārtu vai arī izmantojot lietotāja saskarni lietojumprogramma pārslēdzas atpakaļ uz nākamās iekārtas pievienošanas režīmu. Ieviesta iespēja palaist rokas vadības testēšanas režīmu, kas piedāvā izvēlēties atsevišķa testa veikšanu, kas paredzēts atkārtotai individuālai komponenšu testēšanai pirms vai pēc iekārtas (vai plates, vai elektroniskās komponentes) defekta novēršanas (sk. 4.7 att.).



4.7 att. **Lietojumprogrammas rokas vadības testēšanas režīma ekrānuzņēmums**

Lietojumprogramma atsevišķi attēlo kļūdas paziņojumus ar informatīvu saturu gadījumos, kad:

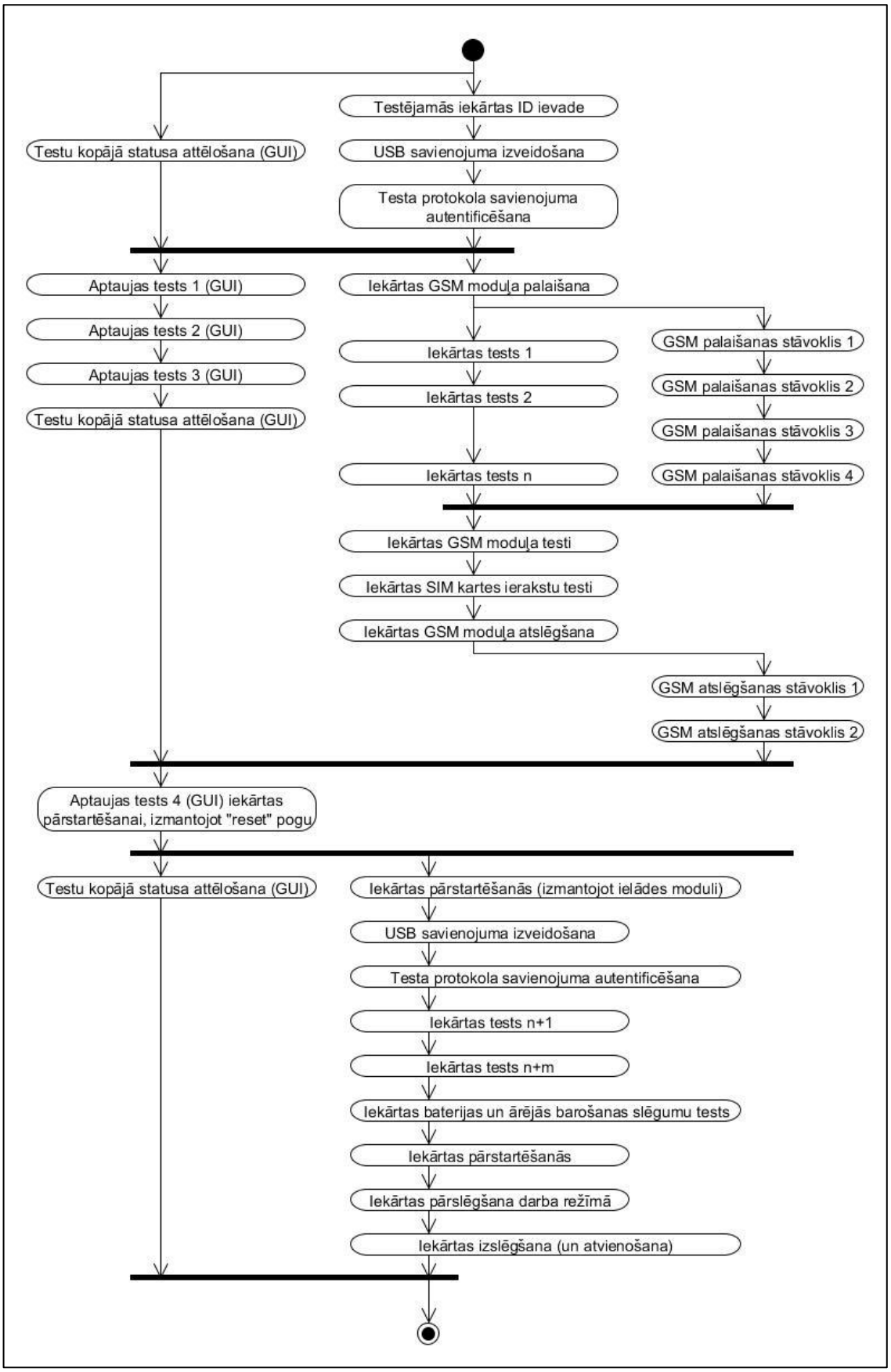
- nav pieejams tīkla savienojums programmatūras lejupielādei;
- *SVN* repozitorija funkcionalitātes problēma iestājusies;
- kļūdas situācija ielādējot programmatūru iekārtai;
- tīkla savienojums ir pieejams, taču nav iespējams nosūtīt atskaiti *HTTP* serverim.

Aizverot lietojumprogrammu tiek atbrīvoti izmantotie resursi un saglabāti pašreizējie lietojumprogrammas grafiskās lietotāja saskarnes konfigurācijas vērtības, piemēram, loga atrašanās koordinātas uz darba virsmas, kā arī pašreizējā izvēlētā saskarnes valoda.

4.2.5. Testu modulis

Autors iepazīstina ar testēšanas moduļa darbību automatizētā režīmā, kas lietotājam liek pievienot iekārtu (*USB* savienojums), automātiski uzsākot iekārtas testēšanas gaitu. Ikreiz iekārtu, pievienojot atkārtoti, vai arī pēc iekārtas pārstartēšanas tiek veikta iekārtas autentifikācija, lai pievienotā iekārta un darbstacija sinhronizētu savienojuma un gatavības stāvokļus. Automatizēto iekārtas plates elementu (*EEPROM* (uz pusvadītājiem bāzēta lasāmatmiņa, no angļu val. *Electrically Erasable Programmable Read Only Memory*),

FLASH, GSM, utt.) testu laikā paralēli tiek intervēts lietotājs par plates perifērijas elementu darbību. Tā kā paralēli notiekošo testu skaitā ietilpst *GSM* moduļa palaišana, testēšana un atslēgšana, lietotājam tiek attēlota testu norises gaita ar starprezultātiem, līdz *GSM* modulis ir iztestēts un veiksmīgi atslēgts, tikai tad lietotājam tiek piedāvāts nospiest „*reset*" pogu (un speciāli paturēt nospiestā stāvoklī 1 sekundi), kas izraisa iekārtas pārstartēšanos, turklāt iekārtai nonākot palaišanas un programmas ielādes moduļa režīmā, šis notikums tiek reģistrēts kā atsevišķs *USB HID* iekārtas pieslēgšanas notikums, kuru reģistrē darbstacijas lietojumprogramma. Iekārtai tiek nosūtīta komanda atgriezties testēšanas režīmā un iekārtas testēšana tiek turpināta, ietverot baterijas un ārējās barošanas testus, kuru veikšanas laikā netiek lietots *GSM* modulis, jo neveiksmīgu testu izpildes gadījumā iekārtas darbība var tikt pārtraukta, bet neplānota iekārtas atslēgšana *GSM* moduļa darbības laikā nav ieteicama. Pēc visu iekārtas testu pabeigšanas iekārtai tiek nosūtīta komanda atgriezties normālā darba režīmā un izslēgties (kā alternatīva *USB* atvienošanai), šis notikums tiek reģistrēts un iekārtas testēšana ir pabeigta (sk. 4.8 att.).



4.8 att. **Lietojumprogrammas testu moduļa vadības plūsmas diagramma**

4.2.6. SVN modulis

Uzsākot darbu ar lietojumprogrammu, nepieciešams pārliecināties, vai ir lejupielādēta pati pēdējā pieejamā produkcijas versija iegultās sistēmas programmatūrai, kura tiks automātiski ielādēta iekārtās pirms testu veikšanas. Šim nolūkam nepieciešams izmantot tīkla savienojumu un pieslēgties *SVN* repozitorijam, kas iepriekš sagatavots – publiski pieejams, bet aprīkots ar lietotāja autentifikāciju. Šo funkcionalitāti nodrošina *SVN* modulis, kurš balstīts uz repozitorija klienta funkcionalitātes bibliotēkas, kas publiski pieejama interneta vietnē bez maksas un speciāli izstrādāta *C#* lietojumprogrammu izstrādei. „*SharpSvn"* bibliotēka ir balstīta uz „*Apache 2.0*" licences, kas paredz atvērtā koda principu un izmantošanu komerciālos projektos [11]. Izstrādātā moduļa funkcionalitāte ietver servera repozitorija versijas noteikšanu un lokālās darba kopijas sinhronizāciju ar servera repozitoriju. Visa iekļautā funkcionalitāte realizēta asinhronā veidā, par darbību pabeigšanu ziņojot ar notikuma izsūtīšanu (*event*), kas paredzēts saderīgumam ar grafisko lietotāja saskarni lietojumprogrammā. Turklāt repozitorija saturu veido programmatūras binārās datnes visiem atbalstītajiem dažādajiem iegulto sistēmu veidiem, ietverot speciālus identifikatorus datņu nosaukumos.

4.2.7. **USB** *modulis*

Izveidotais *USB* datu apmaiņas modulis nodrošina divu veidu funkcionalitāti:

1. protokolu datu apmaiņai ar iekārtu iegultās sistēmas programmas ielādei
2. datu apmaiņu ar iekārtu testēšanas orientētu komandu pārsūtīšanai.

Katram no šiem mērķiem izdalīts atsevišķs *HID* iekārtas identifikators, pēc kā noteikt pieslēgtās iekārtas režīmu, kā arī iekārtas plates projektējuma versijas numuru.

Programmatūras ielādes modelis nodrošina *USB* protokola izmantošanu, lai iegultajā sistēmā ielādētu lietojumprogrammu tam paredzētajā programmas atmiņas sektorā. Šis modulis izstrādāts tā, lai katra no realizētajām funkcijām būtu pieejama asinhronā režīma, turklāt iespējams no šī moduļa saņemt notikumus (*event*), kas satur informāciju par izpildāmā procesa procentuālo veikuma līmeni. Iespējams ne tikai ielādēt programmas atmiņas saturu iegultās sistēmas iekārtā, bet arī pārbaudīt kontrolsummu programmas saturam, kā arī nosūtīt specifiskas kontroles komandas uz iekārtu, lai veiktu, piemēram, automātisku iekārtas pārstartēšanos lietojumprogrammas režīmā pēc programmas atmiņas ielādes.

4.2.8. Iekārtas testēšanas modulis

Lietojumprogrammas galvenais uzdevums ir veikt iegultās sistēmas iekārtas testēšanu, kas parasti notiktu automatizētā režīmā, pēc iespējas bez aizkaves notestējot lielu apjomu ar

iekārtām, secīgi tās pieslēdzot, ielādējot un notestējot vienu pēc otras. Taču ar to vien nepietiek, jo ir nepieciešams režīms rokas vadības iekārtas komponenšu atlases veida testēšanai, jo situācijā, kad noteikta bojāta iekārtas komponente, nepieciešama iespēja ātri un bez liekām papildus darbībām notestēt tieši konkrēto elementu uz iekārtas plates, lai apstiprinātu bojājumu. Kā arī plates elektronisko lodējumu korekcijas laikā vēlams precīzi un vairakkārt pārbaudīt atsevišķo komponenti, turklāt bez citiem papildu testiem, kas varētu patērēt daudzkārt vairāk laiku. Šim nolūkam lietojumprogramma atbalsta rokas vadības testēšanas režīmu, kas piedāvā sarakstu ar iekārtas komponentēm, lai tiktu uzsākta norādītās komponentes individuāla testēšana.

Iekārtas testēšana realizēta ar autora ieviestu protokolu, kas sastāv no testēšanas vadības komandu pārsūtīšanas uz testējamo iekārtu no lietojumprogrammas un testu statusa, rezultāta sūtīšanu no testējamās iekārtas atpakaļ uz lietojumprogrammu. Galvenais pamatprincips – darbstacijas lietojumprogramma nosaka testu izpildes gaitu un vizuāli reprezentē statusu un rezultātus, taču iegultās sistēmas programmatūra satur testu realizācijas loģiku, kur katram no testiem ir izplānoti un ieviesti atsevišķi nodalāmi soļi, par kuru uzsākšanu tiek nosūtīta statusa informācija uz darbstacijas lietojumprogrammu. Iegultās sistēmas programmatūrā realizētie testi satur mainīgos, kuru vērtības tiek saņemts no darbstacijas lietojumprogrammas ar testu uzsākšanas komandām, šī testēšanas konfigurācija ir daļa no pārsūtāmas testu atskaites, jo satur informāciju par veiktu testu uzstādījumiem, kas kopā ar rezultātiem un citiem iegūtajiem mērījumiem sniedz kopēju informāciju par testu norisi un uzstādījumiem, no kuriem arī ir atkarīgi iegūtie rezultāti.

4.2.9. Testu rezultāta pārsūtīšanas modulis

Veicot automatizētu iekārtas testēšanu rūpnīcā, standarta procedūras ietvaros paredzēts identificēt testējamo iekārtu, nolasot iekārtas identifikatoru ar svītru kodu skeneri, savukārt tas, pievienots pie darbstacijas, veic lietotāja teksta ievades funkciju aktīvajā teksta ievades laukā lietojumprogrammai. Skenera teksta ievades funkcionalitāte tiek izmantota testējamās iekārtas *USB* savienojuma gaidīšanas stāvoklī. Turpmākā iekārtas testēšana uzkrāj veikto komponenšu testu rezultātus un, lietojumprogrammai attēlojot testu rezultātus, tiek automātiski sagatavota testu rezultātu atskaite, kura tiek pārsūtīta uz serveri. Lai nodrošinātu minēto funkcionalitāti, autors izveidojis trīs apakš-moduļus, kas savstarpēji saistīti:

- komponenšu testu parametri un konfigurācija (veicamo testu uzstādījumi);
- atskaites objekta pārveidošana *XML* formātā;
- datu pārsūtīšana uz *HTTP* serveri.

Testu parametri un konfigurācija ir pilnībā nodalīti no pārējas sistēmas, ir plānots izveidot atbalstu šo parametru ielādei no datnes vai arī lejupielādei no attālināta *HTTP* servera. Pašreizējā lietojumprogrammas realizācija paredz iepriekšnoteiktu testu konfigurāciju, kuru nav iespējas mainīt lietojumprogrammas izpildes laikā. Šo konfigurācijas objektu papildinot ar iegūtajiem testu rezultātiem, tiek iegūts pilnīgs testu atskaites kopums, kurš tiek pārveidots *XML* formāta datnē, lai to tālāk pārsūtītu uz serveri (serveris apkopo veikto testu rezultātus). Šo datu pārsūtīšana balstās uz tīkla savienojuma pieejamību, taču autors izstrādājis mehānismu, kas uzkrāj neizsūtītās atskaites līdz brīdim, kad atkal pieejams tīkla savienojums, lai neaizkavētu iekārtu testēšanu rūpnīcā, jo jebkura aizkave var radīt finansiālus zaudējumus firmai, kas pasūtījusi iekārtu testēšanu. Dati tiek pārsūtīti uz *HTTP* serveri, lai tiek tiktu apkopoti. Šie dati satur datuma un laika ierakstus, kas sniedz precīzu informāciju par to, kurā brīdi dati tika sagatavoti (tika beigt iekārtas testēšana), lai jebkurā momentā būtu iespējams veikt liela apjoma atskaites ģenerēšanu no šiem datiem, lai sniegtu pārskatu par testēto iekārtu partijas stāvokli, kas ir īpaši noderīgi, ja veiktas plates arhitektūras izmaiņas vai labojumi un nepieciešams novērtēt kvalitātes rādītājus.

4.2.10. E-pasta sūtīšana

Lietojumprogrammai realizēts e-pasta sūtīšanas modulis, kas sasaistīts ar avārijas situāciju apstrādi lietojumprogrammas darbības laikā. Šādu nepieciešamību autors pamato ar ierobežotu iespēju novērot lietojumprogrammas darbību rūpnieciskās ražošanas apstākļos, jo nepieciešams lietojumprogrammā ietvert iespēju attālināti fiksēt kļūdu gadījumus, lai tos pēc iespējas ātrāk atrisinātu, lieki neapgrūtinot lietojumprogrammas lietotāju, intervējot par radušos situāciju. Kļūdu (avāriju) gadījumā tiek uzģenerēts e-pasta saturs ar vispārēju informāciju par darbstaciju, lietojumprogrammas darbības vēsturi, izmantoto bibliotēku versijām un pielietotās iegultās sistēmas *SVN* repozitorija versijas numurs, kas raksturo iespējamo kļūdu cēloņus, kas varētu rasties attīstot testēšanas *USB* protokolu un lietojot nesaderīgas lietojumprogrammas un iegultās sistēmas programmas versijas nākotnē. Izstrādātais e-pasta sūtīšanas modulis izmanto speciāli tam izveidotu kontu, kā arī sūta uz speciāli izmantotu saņēmēja kontu (*Gmail*). Šāda konta ienākošo vēstuļu brīdinājumus iespējams pieslēgt viedtālrunim, kas nodrošina ērtu iespēju jebkurā brīdī reaģēt uz saņemto situācijas parakstu vēstules tekstā.

4.2.11. Grafiskās lietotāja saskarnes modulis

Izveidots modulis grafiskās lietotāja saskarnes vadībai un specifiskajai funkcionalitātei, lai nodalītu lietojumprogrammas tehnisko loģiku no grafiskās attēlošanas. Vizuālo

komponenšu iekļautie teksti sasaistīti ar resursu datnēm, kas individuāli katrai no atbalstītajām valodām, tādā veidā nodrošinot dinamisku saskarni ar iespēju pārslēgties starp piedāvātajām valodām. Grafiskā lietotāja saskarne veidota tā, lai secīgi pildot iekārtu testēšanu, lietotājam būtu iespēja lietot klaviatūras darbstacijas taustiņus kā alternatīvu kursora izmantošanai, kas atvieglo darbu lietotājam un ietaupa kopējo pavadīto laiku lietojumprogrammas darbināšanai.

4.3. Iegultās sistēmas aparātprogrammatūra

4.3.1. Sāknēšanas un programmatūras ielādes modulis

Iegultās sistēmas programmatūra sastāv no diviem galvenajiem moduļiem:

1. sāknēšanas (un programmatūras ielādes) modulis;
2. lietojumprogrammas modulis (iekārtas testēšanas un darba režīma funkcionalitāte).

Iegultās sistēmas programmatūra veidota tā, lai to varētu attālināti atjaunot, izmantojot tīkla savienojumu. Šim nolūkam izveidots speciāls iegultās sistēmas programmatūras modulis, kas vienmēr tiek iedarbināts pirmais, kad iekārta tiek ieslēgta. Šī moduļa uzdevums ir noteikt, vai „*reset*" poga nav tikusi ilgstoši turēta nospiestā stāvoklī pirms iekārtas iepriekšējās izslēgšanās, tādā veidā nosakot, vai nepieciešams iekārtu pārslēgt programmatūras ielādes režīmā vai arī turpināt darbu normālā režīmā.

Iekārtas nostādīšana programmatūras ielādes stāvoklī paredz speciāli izstrādātu komandu apmaiņas protokolu ar darbstacijas lietojumprogrammu, kas ietver datu ielādi, programmas datu kontrolsummu pārbaudi un iekārtas atgriešanas normālā stāvoklī komandas. Vēl iespējams noteikt, vai ir iepriekš lejupielādēta programma, kas saglabāta ārējā *FLASH* atmiņas modulī, tad iespējama programmas kopēšana uz mikroprocesora programmas atmiņas sektoru.

Ja „*reset*" pogas turēšana nospiestā stāvoklī vismaz 1 sekundi nav notikusi (kondensators nav pietiekami uzlādēts un mikroprocesoram pie tapiņas nav pietiekams sprieguma līmenis), tiek fiksēta iekārtas pārlikšana normālā režīmā, t.i., tiek izpildīts lēciens uz programmas adresi, kas nosaka iegultas sistēmas lietojumprogrammas uzsākšanu.

Aprakstītais sāknēšanas (un programmatūras ielādes) modulis ir izstrādāts tā, lai tas būtu atpakaļ saderīgs ar iepriekšējām iegultās sistēmas lietojumprogrammas versijām, lai programmas apjoms būtu pēc iespējas mazāks (jo tas tiek ielādēts tikai un vienīgi ar *Pickit3* programmatoru, to dara izmantojot *ICD2* slēgumu), jo ir būtiski samazināt ielādei

nepieciešamo laiku (oru programmatūras moduli pēc tam iespējams ielādēt, izmantojot *USB* kopni, izmantojot pēdējo programmatūras versiju).

4.3.2. Iegultās sistēmas aparātprogrammatūra

Iekārtas testēšanas nolūkam autors patstāvīgi izstrādājis programmatūras moduli programmēšanas valodā *C*, kura uzdevums ir testēšanas komandu saņemšana no darbstacijas (izmantojot *USB* datu kopni) un veicamo testu realizācija konkrētai iekārtas plates versijai.

Iegultās sistēmas aparātprogrammatūra jau bija izstrādāta iepriekš, tādēļ testēšanas modulim bija nepieciešams nodrošināt pilnīgu saderīgumu ar to, taču testēšanas funkcionalitātei bija nepieciešams būt pilnībā izolētai no pārējā pirmkoda un citiem lietojumprogrammas moduļiem. Svarīgi bija izmantot jau realizēto funkcionalitāti, kas nodrošināja, piemēram, *GSM* moduļa darbības stāvokļu uzturēšanu, tādā veidā izvairoties no pirmkoda dublēšanas un liekas programmas atmiņas izmantošanas, kas ir ļoti būtisks nosacījums iegulto sistēmu programmēšanai. Protams, atsevišķos gadījumos esošo funkcionalitāti bija nepieciešams pārveidot, lai to padarītu universāli pielietojamu testa modulim tāpat kā pamata funkcionalitātei.

Testēšanas moduļa izstrādes sākumposmā tika noteikta prasība to ielādēt specifiskā programmas atmiņas adresē, lai pēc tā izmantošanas šo atmiņu varētu atkārtoti izmantot citiem mērķiem, taču šis uzstādījums tika pārdomāts, jo iegultās sistēmas testēšanas moduļa funkcionalitāti tika nolemts atstāt vienmēr pieejamu, to var ieslēgt, nosūtot iekārtai komandu jebkurā tās darbības momentā, lai veiktu iekārtas testus ne tikai ražošanas procesa ietvaros

Testēšanas modulim bija nepieciešams nodrošināt saderīgumu ar visām līdzšinējām iekārtas plates versijām un to fizisko komponenšu kombinācijām. Arī visām nākotnē paredzētajām plates versijām tiks ieviests atbalsts testēšanas moduļa konfigurācijai un funkcionalitātei (līdzīgi arī darbstacijas lietojumprogrammai).

Izstrādājot šo moduli tika ņemts vērā fakts, ka iekārta tiek testēta un nav garantijas, ka visas elektriskās komponentes strādā stabili, nav arī zināms, vai enerģijas avota slēgumi ir atbilstoši funkcionējoši. Tā kā testēšanas laikā tiek testēti gan elektriskie slēgumi, gan arī pati atmiņa (*EEPROM* un *FLASH*), līdz ar to testēšanas modulim netiek izmantota iespēja pieglabāt savu stāvokli vai citu mainīgo vērtības. To, vai iekārta pārstartējas, paredzēts noteikt darbstacijas lietojumprogrammai, tādēļ iegultās sistēmas testēšanas modulis īpaši pielāgots uzdevumu pildīšanai bez ārējās atmiņas izmantošanas, tas nerūpējas kopējo testēšanas stāvokli, bet tikai secīgi izpilda uzdotā testa soļus un atgriež rezultātu vai arī restartējas un atsāk darbu no sākuma.

Testēšanas modulis izmanto *USB* kopni, lai saņemtu no darbstacijas komandas par testiem, kurus jāuzsāk, un arī lai saņemtu šo testu konfigurācijas parametrus, piemēram, laika ierobežojumus testu izpildei. Testa soļu izpildes laikā (testiem definēti atsevišķi stāvokļi) testēšanas modulis secīgi sūta pašreiz izpildāmā testa progresa vērtības uz darbstaciju, lai darbstacijas lietojumprogramma spētu lietotāja grafiskajā saskarnē attēlot testēšanas gaitu reālā laikā. Tā kā *USB* kopnes darbības stabilitāte izpildot testus, arī nav zināma, ir izveidots protokols, kā darbstacija un iekārta izveido savienojumu un uzsāk testus. Darbstacijas lietojumprogramma šo procesu uzrāda kā *USB* testu, taču tā laikā iekārtas testēšanas modulis uzsāk savienojumu, gaida speciālo simbolu no darbstacijas lietojumprogrammas un atbild tam ar speciālu atslēgvārdu, kopā nosūtot arī atbalstīto testu numurus, lai darbstacijas lietojumprogramma zinātu, cik no visiem iespējamajiem testiem ir definēti konkrētajai iegultās sistēmas aparātprogrammatūrai. Tikai pēc šī procesa veiksmīgas pabeigšanas savienojums ir izveidots un darbstacijas lietojumprogramma uzsāk testu sākšanas un parametru komandu sūtīšanu secībai testu un citas testēšanas funkcionalitātes nodrošināšanai. Šī procesa kopējais ilgums ir atkarīgs arī no darbstacijas spējas apstrādāt *HID* iekārtas pieslēgšanu *USB* portam, tādēļ šī sinhronizācija starp darbstacijas lietojumprogrammu un iegultās sistēmas aparātprogrammatūru ir tik nozīmīga.

Testēšanas modulis sūta lietojumprogrammai uz darbstaciju testēšanas laikā informāciju par pašreizējā testa norisi un rezultātu, taču papildus tam asinhroni sūta arī šādu notikumu informāciju:

- nospiesta lietotāja poga iekārtai;
- no *GSM* moduļa iegūti *IMEI* (mobilās iekārtas identifikators GSM tīklā, no angļu val. *International Mobile Equipment Identity*) un *IMSI* (SIM kartes mobilais identifikators mobilajā tīklā, no angļu val. *International Mobile Subscriber Identity*) identifikatori;
- no *GSM* moduļa iegūti operatora izvēles dati;
- no *GSM* moduļa iegūts pievienotās *SIM* kartes adrešu grāmatiņas ieraksts.

Šādas informācijas pārsūtīšana uz darbstacijas lietojumprogrammu notiek, līdz ko tas iespējams, taču tas neietekmē pašreizējā izpildāmā testa funkcionalitātes izpildi. Svarīgi piebilst, ka *GSM* moduļa lietošana nosaka laikietilpīgu tā palaišanas procesu un atvienošanas stāvokļu izpildi, tādēļ iegultās sistēmas testēšanas modulis paredz atsevišķas komandas izpildi – ieslēgt vai atslēgt *GSM* moduli, jo pēc noklusējuma testu laikā *GSM* modulis ir izslēgts, piemēram, kamēr tiek veikti enerģijas avotu elektrisko slēgumu testi.

Iekārtas testēšanas modulis aparātprogrammatūrā atbalsta speciālu „testu", kura laikā iekārta tiek nostādīta parastā darba režīmā, tas tiek izpildīts iekārtas testēšanas beigās. Kopš tā

izpildes brīža lietojumprogramma ignorē visas testēšanas komandas, bet jebkurā brīdī ir iespējams iekārtai nosūtīt speciālu komandu, kas iekārtu atkal nostāda testēšanas režīmā, to darba testēšanas lietojumprogramma brīdī, kad tās darbības laikā pie *USB* kopnes pievieno testējamo iekārtu.

5. APARĀTPROGRAMMATŪRAS TESTĒŠANA

5.1. Testēšanas veidi un to pielietojums

5.1.1. Testēšanas līmeņu klasifikācija

Programmatūras testēšana tiek iedalīta vairākos līmeņos pēc tā, kurā no programmatūras izstrādes posmiem tie tiek veikti, kā arī pēc testēšanas specifikas. Šis iedalījums attiecas uz visa veida programmatūru, tai skaitā, arī uz iegulto sistēmu, jeb aparātprogrammatūru. Galvenokārt izšķir šādus līmeņus pēc testa mērķa:

- vienību testēšana;
- integrācijas testēšana;
- sistēmas testēšana;
- sistēmas integrācijas testēšana [12].

Vienību testēšana ietver atsevišķu programmatūras moduļu funkcionalitātes testēšanu ar mērķi pārliecināties, vai tie atbilst lietojamības nosacījumiem, piemēram, specifiskas funkcijas testēšana, ietverot iespējamās parametru vērtības, sagaidot attiecīgu funkcijas rezultātu vai izvades datus. Integrācijas testēšana pārbauda atsevišķu moduļu saderīgumu grupās, tiek testēts tas, vai savstarpēji saistīti moduļi spēj atbilstoši mijiedarboties savstarpēji. Sistēmas testēšanas līmenis nosaka kopējas sistēmas atbilstību izvirzītajām prasībām, tiek pārbaudīts, vai sistēma visa kopumā funkcionē atbilstoši (vai visi ietvertie moduļi kopumā funkcionē kā plānots). Visbeidzot sistēmas integrācijas līmenis testēšanai nosaka testēšanas procesu, kas pārbauda, kā konkrētā sistēma integrējas ar citām sistēmām plašā mērogā. Lai veiktu sistēmas integrācijas testēšanu, ikvienai tajā ietvertajām sistēmām jābūt iepriekš notestētām ar visiem iepriekš uzskaitītajiem testēšanas līmeņiem [12].

Programmatūras izstrādē testēšanas loma ir vērsta uz kvalitātes nodrošināšanu. Lai to sekmīgi realizētu, nepieciešams vispirms sagatavot testēšanas plānu katram no testēšanas procesiem, lai uzskaitītu, kas tieši jāpārbauda izpildot norādītos testpiemērus. Izstrādājot šādu plānu, jāpatur prātā, ka nav iespējams atrast pilnīgi visus sistēmas defektus, kā arī tas, ka nekad nepietiek laika, lai pilnībā visu iespējamo notestētu. Tādēļ nepieciešams organizēt testēšanu, izstrādājot testēšanas plānu, kas paredz plānotu resursu sadali, izvirzīto prioritāšu mērķu sasniegšanai.

Aparātprogrammatūrai piemīt īpašība, ka tā parasti ir saistīta ar apkārtējo vidi vai arī to kontrolē kāda cita sistēma, bet tās pašas galvenā funkcionalitāte balstīta uz konkrētu iekārtu kontrolēšanu vai datu ieguvi. Lai gan aparātprogrammatūras testēšana ir stipri atkarīga no šīs iegultās sistēmas veida, funkcionalitātes, sfēras un citiem nosacījumiem, tomēr saistībā ar

testēšanu visām iegultajām sistēmām ir līdzīgas problēmas un pielietojamie risinājumi [13], piemēram, kopīga iezīme iegultajām sistēmām ir ierobežotie resursi (programmas atmiņa, operatīvā atmiņa, procesora ātrdarbība un citi), tipisks risinājums ir testēšanas sistēmas iznešana ārpus iegultās sistēmas moduļa, izmantojot pieejamās datu kopnes un citus savienojumus (*ICD2* slēgums *PIC* mikroprocesoram ar izstrādes vides rīku, kas darbināms uz personālās darbstacijas).

Izplatīts iegulto sistēmu testēšanas paņēmiens ir modeļa bāzēta testēšana (parasti balstīta uz iegultās sistēmas nefunkcionālo īpašību kopu – asinhrono darbību, sinhronizāciju un laika ierobežojumiem), kas savukārt iedalās dažādās dimensijās un kategorijās, piemēram:

- programmas vadības plūsmu testēšana;
- datu plūsmu testēšana;
- pirmkoda zaru nosacījumu testēšana.

Galvenais kritērijs modeļa bāzētas testēšanas veida izvēlei ir programmatūras pirmkoda pieejamība, kas nosaka, vai testējamās programmatūras izveidotais modelis veidots (vai ģenerēts) pēc pirmkoda zaru un nosacījumu plūsmas, jeb veidots pēc vispārināta programmas funkcionalitātes attēlojuma, kas ietver nodalītus stāvokļus un pārejas starp tiem (transakcijas) ar ieejas – izejas datu nosacījumiem.

Būtiski ir testēt arī programmas veiktspēju pie noteikta apjoma apstrādājamo datu kopuma, jo realizētā programmas funkcionalitāte var ietvert nepārdomātu sistēmas resursu sadali, kas var novest programmu avārijas stāvoklī pie konkrētas pārslodzes. Šādu gadījumu nepieciešams notestēt pirms programmatūras ieviešanas fāzes.

Iegultās sistēmas var ietvert arī lietotāja saskarni, izmantojot perifērijas komponentēs, tādēļ zināmā mērā nepieciešams arī veikt saskarnes testēšanu, vadoties gan pēc lietotāja dokumentācijas, gan citiem kritērijiem, kas nosaka sistēmas ērtu un intuitīvu saskarni. Lietojamības testēšanas laikā iespējams atklāt nepilnības, kuras iepriekš nav paredzētas, tādēļ vēlams testēšanas procesā iesaistīt potenciālo sistēmas lietotāju, lai tas sniegtu atzinumu un dalītos ar ieteikumiem saskarnes uzlabošanā.

Jebkura aparātprogrammatūrai vai lietojumprogramma jātestē arī no drošības viedokļa. Drošības testēšanas galvenais nolūks ir šādu programmatūras defektu apzināšanas veidu izmantošana:

- kvalitātes nodrošināšana;
- sistēmas administrēšana;
- neaizsargātības novērtējums [14].

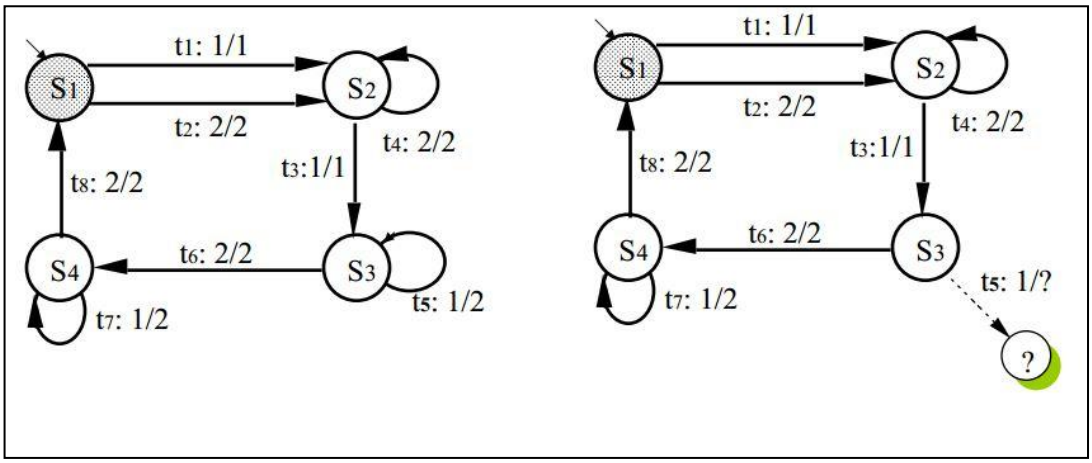
Drošības testēšana ietver programmas saskarnes datu ievades funkcionalitātes pirmkoda caurskatīšanu, kā arī programmas darbības testēšanu ar neatļautu vai neparedzētu ievaddatu saturu. Līdzīgi nepieciešams testēt arī programmatūras komunikāciju savienojumus ar citām sistēmām. Piemēram, ja aparātprogrammatūra izmanto *GSM* moduli, tad nepieciešams pārliecināties, vai nav iespējams kaitnieciski ietekmēt programmatūras darbību, ja tiktu nosūtīta *SMS* uz šo iekārtu ar datiem, kas saturētu, piemēram, citu *GSM* moduļa vadības komandu virkni, ar kuru būtu iespējams ietekmēt *GSM* moduļa konfigurāciju vai *SIM* kartē saglabāto datu saturu (datu injekcijas metode). Parasti drošības testēšana ietver metodes sistēmas darbības traucēšanai, datu pārtveršanai vai pilnīgai sistēmas sagraušanai, tas nosaka specifiskas zināšanas testu piemēru sastādīšanas personālam konkrētajā iegultās sistēmas sfērā.

5.1.2. MPLABX un testēšanas rīki

Autora aplūkotā *Microchip* mikroprocesora pirmkoda izstrādes un kompilēšanas rīka *MPLABX* funkcionalitāte līdz šim neatbalsta saskarni testu veidošanai, ģenerēšanai izpildei vai organizēšanai. *PIC* mikroprocesora testēšanas atbalstu jānodrošina pašam programmatūras izstrādātājam, paredzot speciālu funkcionalitāti aparātprogrammatūrā, piemēram, izstrādājot funkciju kopu, kas tiek palaistas testa režīmā un testu rezultāti, piemēram, tiktu pārsūtīti vai izmantojot *USB* savienojumu ar darbstaciju vai arī izmantojot kādu citu saskarni. *MPLABX* vide neatbalsta arī programmas vadības plūsmas diagrammu ģenerēšanu vai arī cita veida pirmkoda testēšanu, kas balstīta uz zaru nosacījumu un to kombināciju testēšanu, taču šim nolūkam iespējams izmantot kādu citu tīmeklī pieejamu rīku, piemēram, *Understand, Graphviz, Navigator* vai citu. Taču pārsvarā šie rīki piedāvā vizualizēt pirmkodu grafa veidā, kā arī novērtēt pirmkoda atbilstību programmēšanas labās prakses nosacījumiem, bet ne visi no šiem rīkiem atbalsta automatizētu testēšanu.

5.2. Galīga stāvokļu testēšana

Eksistē formālas metodes testpiemēru ģenerēšanai, lai efektīgi testētu sistēmas darbību, piemēram, galīga stāvokļu testēšana. Lai to izmantotu, nepieciešams izveidot galīgu stāvokļu automātu, t.i., sistēmas modeļa reprezentāciju, balstoties uz kuru, uzģenerē testu kopu un to izmanto sistēmas testēšanai, lai noteiktu, vai uzdotais sistēmas galīga stāvokļu automāts atbilst konkrētajai sistēmai (reālajai programmas darbībai). Tādā veidā iespējams gan notestēt programmas darbību visos stāvokļos, gan arī notestēt programmas darbību, ko izraisītu neatļautas stāvokļu pārejas. Šādam sistēmas modelim, stāvokļu grafam, jābūt determinētam, lai nerastos problēmas, testējot atsevišķus stāvokļus (sk. 5.1 att.).



5.1 att. **Testējamā sistēma un tās modelis - stāvokļu grafs** [15]

Galīga stāvokļu automāta testēšanas formālās metodes, kuras var tikt pielietotas determinēta galīga automāta modelim:

- *Transition Tour (T)* metode;
- *Distinguishing Sequence (D)* metode;
- *Characterizing Set (W)* metode;
- *Unique Inpit/Output Sequence (UIO)* metode;
- *Single UIO (SUIO)* metode;
- *Multiple UIO (MUIO)* metode.

Lai veiktu galīgu stāvokļu testēšanu, izveidotajam automāta modelim jābūt:

- galīgam (visi stāvokļu un visas stāvokļu pārejas ir sasniedzamas);
- determinētam (no jebkura stāvokļa jebkuram ieejas alfabēta burtam eksistē ne vairāk kā viena pāreja);
- minimālam (nav divu vai vairāku vienādu stāvokļu);
- pabeigtam (eksistē ceļš – galīgs blakus esošu stāvokļu pāreju, bet ne obligāti atšķirīgu, virknējums starp jebkuriem diviem izvēlētiem stāvokļiem) [16].

Galīgai stāvokļu testēšanai nav nepieciešama detalizēta informācija par testējamās sistēmas realizāciju. Lai izmantotu šo testēšanas paņēmienu, nepieciešams:

1. ģenerēt galīga automāta ievades datus un sagaidāmos izvades datus;
2. nodrošināt šo ievades datu padošanu galīgam automātam ieejā;
3. veikt testējamās sistēmas reālo izvadu datu salīdzināšanu ar galīgā automāta izvades datiem izmantotajai ieejas datu un izstaigāto stāvokļu kopai.

Izveidotā galīga stāvokļu automāta modelim jāsakrīt ar reālo sistēmu, kas tiek testēta, vienīgi iespējams, ka savstarpēji atšķiras izmantoto stāvokļu skaits, kas reālajai sistēmai

parasti ir lielāks, līdz ar to arī atšķiras stāvokļu pāreju (transakciju) skaits un sarežģītība [16; 17].

Testam paredzētos ievades datus ģenerē ar iepriekš minētajiem paņēmieniem, piemēram, *Transition Tour (T)* metodi. Lai to izdarītu, nepieciešams no galīgā automāta modeļa iegūt simetrisku, minimizētu grafu, savukārt pielietojot šo grafa apstaigāšanas metodi, tiek iegūts ceļš, kas ietver grafa virsotnes. *Transition Tour (T)* metodes pamatā ir algoritms, lai iegūtu tādu grafa virsotņu apstaigāšanas ceļu, kas ietver ikvienu virsotni vismaz vienu reizi.

Grafa apstaigāšanas metodes, ar kurām uzģenerēt ieejas datu kopu galīgajam automātam, atšķiras. Lai sīkāk pētītu grafa apstaigāšanas ceļus, nepieciešams nodefinēt jēdzienus:

- grafa virsotņu apstaigāšanas ceļa izmaksas – visu apstaigāto virsotņu vērtību summa, ņemot vērā to, cik reizes katrā virsotnē nonākts;
- grafa virsotnes izmaksas – ieejas un izejas simbolu pāri.

Tātad visas grafa virsotnes apstaigāt iespējams dažādos veidos, turklāt ar atšķirīgām izmaksām, tādēļ jāpiemin, ka eksistē algoritmi, kā iegūt vismazāko izmaksu grafa visu virsotņu apstaigāšanas ceļu. Šāds algoritms ir, piemēram, *Chinese Postman Tour (CPT)* [16].

Izmantojot galīga automāta modeli sistēmas testēšanai, praktiskam nolūkam nepieciešams nodrošināt iespēju automātam atgriezties stabilajā sākuma stāvoklī pēc testpiemēra darbināšanas. Šīs metodes trūkums ir tas, ka tiek notestēti tikai sistēmas izvadi konkrētajos stāvokļos.

5.2.1. *Transition Tour* (T) metode

Konkrētam galīgam automātam S, *Transition Tour* grafa virsotņu secība, kas sākas ar grafa sākuma stāvokli, apstaigā ikvienu virsotni vismaz vienu reizi un atgriežas atpakaļ sākuma stāvoklī.

Ar šādu metodi iespējams noteikt pilnīgi visas iespējamās testējamās sistēmas izejas kļūdas, taču nav garantijas, ka visas stāvokļu pārejas kļūdas tiks noteiktas, jo nav garantēts, ka tika izlietotas pilnīgi visas virsotņu pārejas [15].

5.2.2. Distinguishing Sequence (DS) *metode*

Šī metode balstās uz atrastu konkrētu ievades secību galīgam automātam, ja produkcija, ko ražo automāts atšķiras, kad ieejas secība tiek pielietota katram atšķirīgam stāvoklim. Šī ievades secība tiek lietota kā stāvokļa identificēšanas secība.

Ar šādu testēšanas paņēmienu ir iespējams noteikt visas izvades kļūdas, kā arī visas stāvokļu pāreju kļūdas, taču stāvokļa identificējošo (unikālo katram stāvoklim) ieejas secību ne vienmēr ir iespējams atrast dotajam galīgajam automātam [15].

5.2.3. Characterizing Set (W) *metode*

Šī metode ietver divas ieejas secību kopas: W (simbolu kopa minimālam galīgam automātam, tā sastāv no ieejas secības, kas dažādi izpildīties automātam uz katru stāvokļu pāri) un P (ieejas secības, ka katra stāvokļu pāreja no A uz B pie ievades x, eksistē ievades secības p un p.x šajā ievades kopā P, ka p noved galīgo automātu tā sākuma stāvoklī A) kopu [15].

5.2.4. Unique Inpit/Output Sequence (UIO) *metode*

Tradicionāla *UIO* metode var tik izmantota, ja katram specifikācijas stāvoklim eksistē ieejas secība, ka automāta izejas produkcija (kad tas sākotnēji atrodas dotajā stāvoklī), atšķiras no jebkuru citu stāvokļu iegūtas secības. Testa secība galīgam automātam tiek veikta, pārbaudot tā grafa katru malu ar testa apakšvirkni, neatkarīgi no iepriekšējām malām. Arī šai metodei ir plašs pielietojums komunikācijas protokolu uzturošu sistēmu testēšanai, jo testēšanā izmanto (simbolu) ieejas secību un izejas produkciju. Īpaši uzmanība tiek vērsta uz grafa apstaigāšanas iespējām un apstaigāto virsotņu skaitu, tādēļ tiek izmantoti paņēmieni, kā iespaidot izvēlēta testa ceļa garumu [15; 18].

5.3. Galīga stāvokļa automāta modelēšana

5.3.1. Iegultās programmatūras izstrādes vides rīki

Iegultās sistēmas programmatūras izstrādei tiek izmantots rīks pirmkoda sagatavošanai un mērķa platformas procesora arhitektūrai paredzēts kompilators izmantotajai programmēšanas valodai, iegultajām sistēmām raksturīgas programmēšanas valodas *C* un *C++*, kas tiek pamatos ar programmatūras izpildes ātrdarbību un pielāgojamību ierobežotajiem platformas resursiem.

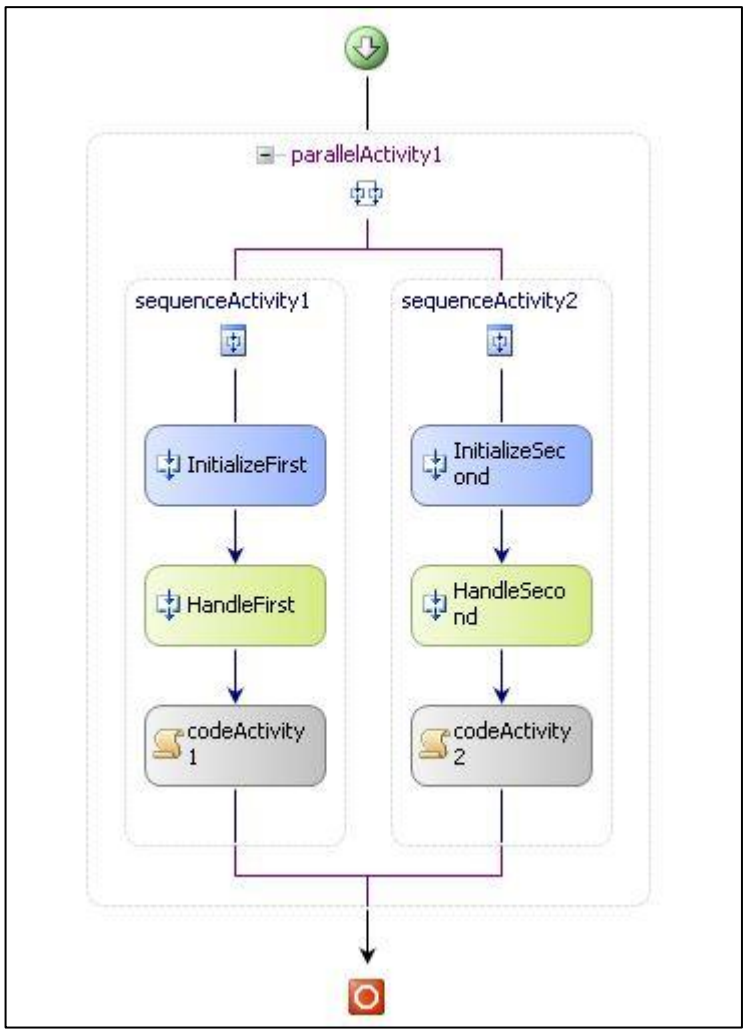
Autora aplūkotajā piemērā, *Microchip* mikroprocesoru saimei ir paredzēta pirmkoda izstrādes vide *MPLABX*, kura nodrošina grafisku teksta redaktoru, kas papildināts ar dažādiem spraudņiem papildu pirmkoda uzturēšanas moduļiem, piemēram, pieejams atbalsts *SVN* repozitorija uzturēšanai un izmaiņu apsekošanai. Šī pirmkoda izstrādes vide ir saderīgums ar valodas *C* (*ANSI* standarts) kompilatoru, kas paredzēts *PIC*24F mikroprocesoru arhitektūras saderīgumam. Minētā platforma paredzēta programmatūras izstrādei un ar tās atbalstu ir

iespējams veikt programmatūras darbināšanu, izmantojot reāla laika atkļūdošanas funkcionalitāti ar iespēju atlikt pirmkoda tekstā pārtraukumpunktus un atspoguļot konkrētā brīža programmas mainīgo pieņemtās vērtības.

Ar šādu funkcionalitāti iespējams veikt pašrocīgu funkcionālo testēšanu, taču ar to nav pietiekami, lai veiktu automatizētu programmatūras testēšanu vai arī pielietotu galīga stāvokļa testēšanas metodi.

5.3.2. Modernas programmēšanas valodas izstrādes vides rīks

Autors piedāvā modernas programmēšanas valodas *C Sharp* izstrādes vides iespēju izklāstu, lai nodemonstrētu iespējas programmatūras izstrādes un testēšanas nolūkam. Kā piemērs izvēlēta programmatūras izstrādes vide *Microsoft Visual Studio 2010* programmēšanas valodas *C Sharp* lietojumprogrammu izstrādei, lietojot *.Net 4.* versijas ietvaru.



5.2 att. ***Windows Workflow Foundation*** **(WF4) rīks** [19]

Sākot ar *.Net 4*. versijas platformas atjauninājumu paku Nr.1, ir pieejams *Windows Workflow Foundation (WF4)* rīks, ar kuru iespējams izmantot grafisko programmatūras izstrādes vides dzini, lai izstrādātu stāvokļu mašīnu (automātu - grafu), kas definē programmatūras stāvokļus un transakcijas (sk. 5.2 att.). Grafiskā rīka funkcionalitāte līdzinās jau iepriekš pieejamajam grafiskajam *Designer* rīkam, ar kura palīdzību iespējams ērti veidot lietojumprogrammas grafisko saskarni, tajā pašā laikā automātiski uzģenerējoties programmas pirmkodam. *Windows Workflow Foundation (WF4)* rīks piedāvā intuitīvā un vizuāli ērti uztveramā veidā izveidot programmas uzvedības modeli, kas tiktu balstīts uz galīga skaita stāvokļiem, transakcijām starp tiem un darbībām (notikumiem). Citiem vārdiem sakot, iespējams izstrādāt lietojumprogrammu, kuras kontroles modulis ir galīga automāta grafiska reprezentācija [20].

Pirmkārt, ieguvums no šāda paņēmiena ir tas, ka programmatūras izstrādātājam ir iespējas tiešā veidā nodefinēt galīgu automātu, nav nepieciešams modelēšanu nodalīt no programmatūras izstrādes, turklāt viss veicams uzskatāmi un līdzvērtīgi jebkuram grafiska atbalsta plūsmu diagrammu zīmēšanas rīkam.

Otrkārt, pēc programmatūras izstrādes fāzes ir skaidri zināms, ka izstrādātā sistēma pilnībā atbilst tam, kas uzskatāmi reprezentēts izveidotajā grafiskajā modelī. Tas nozīmē, ka nav atsevišķi jāpierāda programmatūras atbilstību plānotajam programmatūras projektējumam.

Treškārt, šis izstrādes vides rīks piedāvā izmantot definēto stāvokļu diagrammu, lai veiktu testēšanu automatizētā vai rokas vadības testēšanas veidā, iespējams veikt arī iepriekš definētus, kā arī automātiski ģenerētus testus. Pie tam iespējams arī veikt dažādas pārbaudes, kas nosaka grafiski izveidotā modeļa atbilstību galīga stāvokļu (automāta) testēšanas pamatnosacījumiem. Pieejamā testēšanas funkcionalitāte līdzinās vienību testēšanas ietvara *JUnit* iespējām programmēšanas valodā *Java*, jo iespējamas veikt arī šādas darbības

- Testu scenāriju definēšana;
- Iespējama programmas vadības nodošana ārējām sistēmām
- Grafiska lietotāja saskarne testu atspēlēšanai, rezultātu pārraudzīšanai un testu uzturēšanai.

Autors vēlas vērst uzmanību tieši uz iespēju ļaut rīkam automātiski ģenerēt testus, balstoties uz grafiski izveidoto programmas vadības modeli, kas ietver automātisku stāvokļu apstaigāšanu, izmantojot definētās transakcijas (ar modelī ietvertajām ieejām un izejām, kā arī piesaistīto grafisko funkcionalitāti), lai iegūtu testēšanas rezultātu – definētā automāta uzvedības iepriekšdefinēto nosacījumu pārbaudi.

5.3.3. Iegulto sistēmu sasaiste ar citiem rīkiem

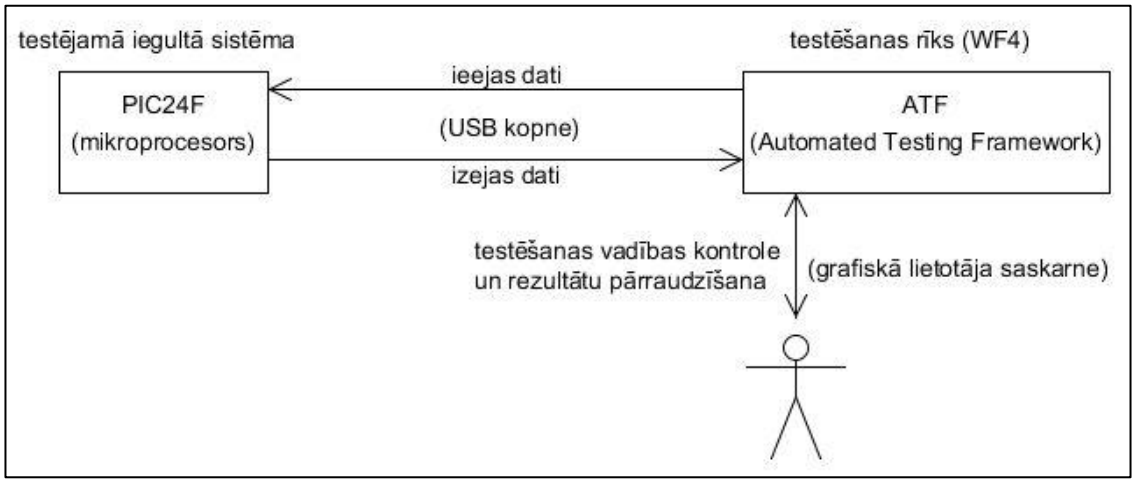
Microchip iegulto sistēmu programmēšanai iespējams izmantot publiski pieejamās bezmaksas programmatūras bibliotēkas, to skaitā *USB HID* iekārtas protokola implementāciju, lai nodrošinātu komunikāciju ar citu (vadošo) aparatūru, izmantojot *USB* datu kopnes standartu.

Izmantojot *USB* savienojumu starp iegulto sistēmu un personālo darbstaciju, iespējams izveidot sistēmu, kura sastāv no iegultās sistēmas programmatūras un darbstacijas lietojumprogrammas ar grafisku lietotāja saskarni. Galvenais ieguvums no šāda paņēmiena ir tas, ka iespējams izmantot, piemēram, *Windows Workflow Foundation (WF4)* rīku, lai testētu iegultas sistēmas darbību. Autors šādā veidā piedāvā realizēt divus atšķirīgus testēšanas paņēmienus:

- iegultās sistēmas programmatūras testēšana;
- iegultās sistēmas izmantošana, lai testētu tai pieslēgtu citu iekārtu.

5.3.4. Iegultās sistēmas testēšanas paņēmiens

Lai veiktu iegultās sistēmas galīga automāta testēšanu, izmantojot *Windows Workflow Foundation (WF4)* rīku, nepieciešams izveidot grafisku stāvokļu diagrammu, kas raksturo iegultās sistēmas stāvokļus, transakcijas un ieejas – izejas datus. Iegūtajam galīgajam automātam nepieciešams veikt testpiemēru ģenerēšanu, lai izveidotu testu kopu un veiktu automāta korektuma pārbaudes. Tad nepieciešams sasaistīt testējamās iegultās sistēmas izvadu ar grafiskās testēšanas rīka izvadu, kā arī līdzīgi savienot ievades, lai, veidot testpiemēru atspēlēšanu, varētu tikt salīdzināta abu sistēmu darbība pēc ieejas un izvades datiem, jo tāds ir iegulto sistēmu testēšanas paņēmiens (*Black box testing*) (sk. 5.3 att.).



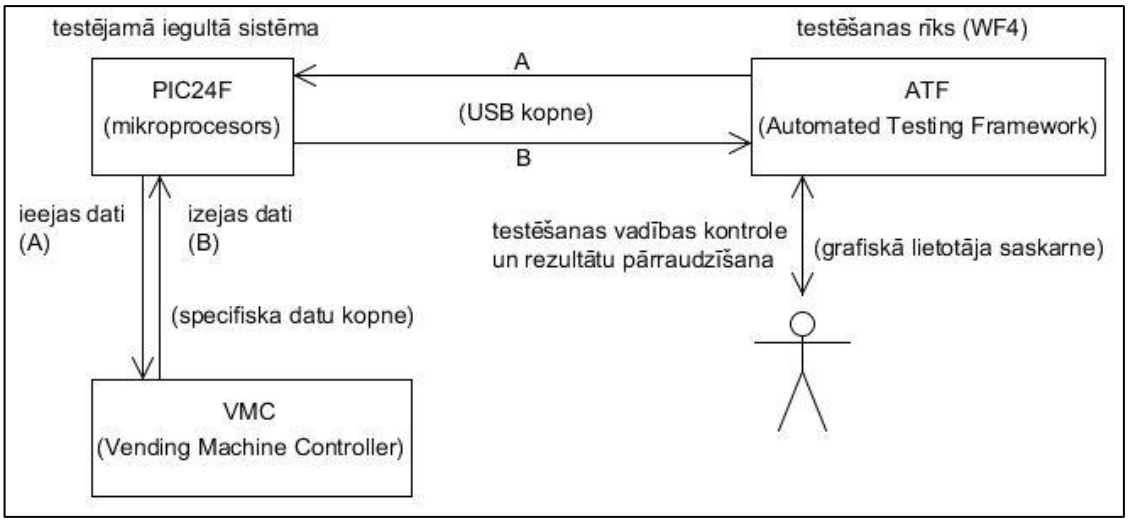
5.3 att. **Iegulto sistēmu galīga automāta testēšana ar WF4 rīku**

5.3.5. Iegultās sistēmas izmantošana citas iekārtas testēšanai

Autors piedāvā realizējamu risinājumu, kā notestēt iegultas sistēmas (iekārtas, kurai nav pieejams programmatūras pirmkods) darbību, balstoties uz tās izejas un ieejas datu pārbaudi, ja zināma stāvokļu pārejas diagramma šai iekārtai.

Piemēram, gadījumā, ja nepieciešams lietot iekārtu, kuras dokumentācija nosaka konkrēta komunikācijas datu protokola izmantošanu, ir svarīgi zināt, vai šis protokols ir implementēts atbilstoši dokumentācijai, bez atkāpēm no tā. Lai to pārbaudītu un pierādītu iekārtas darbības korektumu, nepieciešams no sākuma iegūt pielietojamā datu pārsūtīšanas protokola stāvokļu pārejas diagrammu, kas interpretējams kā formāla specifikācija. Galīga automāta stāvokļu pārejas diagrammu nepieciešams izveidot, piemēram, *Windows Workflow Foundation (WF4)* rīka grafiskajā vidē un pirms testu ģenerēšanas un atspēlēšanas veikt šādus priekšdarbus:

- sagatavot iegultās sistēmas programmatūru iekārtai, kas tiks izmantota kā starpnieks datu pārsūtīšanai starp testējamo iekārtu (savienota specifiskā veidā, izmantojot kādu datu kopnes veidu) un darbstaciju, uz kuras tiks darbināta grafiskā testēšanas vide;
- savienojumus nepieciešams veidot kā norādīts attēlā (sk. 5.4 att.);
- nepieciešams attiecīgi sasaistīt testēšanas rīka ieejas un izejas ar saņemtajiem datiem no iekārtas, lai veiktu datu salīdzināšanu darbināto testu stāvokļos.



5.4 att. **Iegultai sistēmai pieslēgtas iekārtas (VMC) galīga automāta testēšana**

5.4. Datu transporta protokolu un formāta testēšana

Vending telemetrijas iekārtai var tikt pieslēgtas citas iekārtas, šo savienojumu veidi atšķiras gan fiziski, gan izmantotā datu transporta protokola un datu formāta ziņā. Izmantojot galīga automāta testēšanas paņēmienu, iespējams notestēt ne tikai *vending* telemetrijas iekārtas funkcionalitāti, bet arī tai pieslēgto iekārtu funkcionalitātes atbilstību.

Lai veiktu galīga automāta testēšanu datu transporta (komunikācijas) protokolam, nepieciešams izveidot modeli – grafu konkrētajam gadījumam. Izmantojot kādu no iepriekš minētajām metodēm, iespējams uzģenerēt testu kopu, tad notestēt iekārtas saņemtos un nosūtītos datus.

Vending telemetrijas iekārtai svarīgs uzdevums ir veikto pirkumu atskaites iegūšana no tirdzniecības automāta vadības moduļa, savukārt šie dati pieejami speciālā marķēšanas valodās formātā *EVA-DTS* (sk. 5.5 att.). Arī šo datu formāta atbilstības noteikšanai iespējams izmantot galīga automāta testēšanas metodi, uztverot datus kā formālu gramatiku, izveidojot tai akceptoru un ģenerējot testpiemērus.

Audit Report:

DXS*XYZ1234567*VA*V0/6*1	Application header (Mandatory Header)
ST*001*0001	Transaction Set header (Mandatory Header)
ID1*ABC98765*****11223344	Identification data segment
VA1*1000*200	Vend Transaction data segment
VA3*50*1	Free Vend Transaction data segment
CA3*****1250*250*400*6	Cash In data segment
CA4***250*0	Cash Out data segment
PA1*1*50*****	Product 1 Information data segment
PA4*1	Product 1 Free Vend Transaction data segment
PA7*1*CA*0*50*72*3600*11*550	Product 1 Cash Transaction data segment
PA7*1*DA*1*50*756*37800*121*6050	Product 1 Cashless 1 Transaction data segment
PA7*1*DA*2*25*33*825*4*100	Product 1 Cashless 1 Transaction data segment
PA1*2*50*****	Product 2 Information data segment
PA4*0	Product 2 Free Vend Transaction data segment
PA7*2*CA*0*50*40*2000*10*500	Product 2 Cash Transaction data segment
PA7*2*DA*1*50*600*30000*100*5000	Product 2 Cashless 1 Transaction data segment
PA7*2*DA*2*25*30*750*2*50	Product 2 Cashless 1 Transaction data segment
EA2*EGS****1	Event Information
EA3*********24*10	Number of Reads Information
G85*1234	Record Integrity Check (Mandatory)
SE*16*0001	Transaction Set trailer (Mandatory Trailer)
DXE*1*1	Application trailer (Mandatory Trailer)

5.5 att. **EVA-DTS formāta audita atskaite no tirdzniecības automāta (ar skaidrojumiem labajā attēla pusē)**

6. SISTĒMTESTĒŠANA

6.1. Iekārtas testēšana kopējā pakalpojumu sistēmā

6.1.1. Iegultās sistēmas loma kopējas sistēmas darbībā

Telemetrijas sistēmu veido vairāki neatkarīgi moduļi, kas savstarpēji veic datu apmaiņu, lai nodrošinātu sistēmas lietotājam informāciju. Lai nodrošinātu visas kopējās sistēmas atbilstošu darbību, nepieciešams ne tikai nodrošināt katra atsevišķa moduļa testēšanu, bet arī rodas nepieciešamība veikt sistēmtestēšanu, lai pārliecinātos par visas sistēmas darbības atbilstību funkcionālajai specifikācijai. Ir būtiski noskaidrot, vai katrs izstrādātais modulis sistēmā pilda savu funkcionalitāti pareizi un vai katrs moduļu pāris savstarpēji spēj apmainīties ar informāciju. Tātad nepieciešams testēt moduļu savstarpējo saderīgumu, ko demonstrē sistēmtestēšanas rezultāts. Piemēram, *vending* telemetrijas iekārtai nepieciešams sazināties ar diviem citiem moduļiem – *VMC* un telemetrijas serveri. Tas ir būtiski, pārliecināties, vai iekārta spēj korekti iegūt datus no *VMC*, izmantojot datu transporta protokolu un datu formāta marķēšanas valodu, jo pastāv iespēja, ka *VMC* sistēmā, piemēram, ir pieļauta atkāpe no protokola specifikācijas, kas savukārt nozīmē to, ka perfekti pēc specifikācijas izveidota un notestēta telemetrijas iekārta nav spējīga veiksmīgi veikt datu apmaiņu ar *VMC*. Līdzīgi nepieciešams testēt telemetrijas iekārtas saziņas protokolu ar datu serveri, lai pārliecinātos, ka šie moduļi ir saderīgi, ne tikai katrs pats par sevi atbilstošs definētajai funkcionalitātei vai datu apmaiņas protokola specifikācijai.

Apskatot iegultās sistēmas telemetrijas iekārtu kopējā sistēmā, redzams, ka vienlaicīgi sistēmu veido vairākas iegultās sistēmas vienlaicīgi, kurām jābūt atpakaļ saderīgām gan funkcionalitātes, gan datu apmaiņas protokolu ziņā ar serveri, jo iegulto sistēmu programmatūras versijas tiek nemitīgi izlaistas ar uzlabojumiem un kļūdu labojumiem. Tādēļ nepieciešams unikāli identificēt katru no iekārtām, kā arī identificēt iekārtas programmatūras versiju un konfigurāciju. Nepieciešams veikt sistēmtestēšanu ar jebkuru jaunu programmatūras versiju, lai testēšanas režīmā novērotu sistēmas saderīgumu un atklātu neplānotus blakus efektus, ja tādi rastos. Tikai pēc veiksmīgas testēšanas iespējams programmatūras versiju apstiprināt kā stabilu un laist apgrozībā (ieļaut tās attālinātu vai pašrocīgu uzstādīšanu iekārtām).

Šī iemesla dēļ kopējas sistēmas funkcionalitāte ietver iespēju reālā laikā atlasīt to sistēmā reģistrētu iekārtu kopumu, kas šajā laika momentā lieto konkrēto programmatūras versiju. Šāda informācija paver iespēju novērot sistēmas darbības kvalitāti un iegūtos

rezultātus pamatot vai sasaistīt ar konkrētu programmatūras versiju izmantošanu, kas ļauj spriest par veikto programmatūras izmaiņu iespaidu uz sistēmas darbību un produktivitāti.

6.1.2. Iegultās sistēmas testēšana, izmantojot pārējo sistēmu

Iegultās sistēmas tipa iekārtu iespējams testēt, izmantojot mākslīgi radītus stāvokļus pārējā sistēmā. Konkrētajā gadījumā iespējams ietekmēt iekārtas darbību, secīgi veicot manipulācijas ar telemetrijas sistēmas servera lietotāja saskarni, kā rezultātā iespējams panākt iegultās sistēmas atsevišķas funkcionalitātes izpildi vai pat vairāku secīgu darbību apkopojumu konkrētā testpiemēra ietvaros. Šādā veidā iespējams testēt gan sistēmu kopumā, gan mākslīgi radīt produkcijas apstākļus sistēmas darbībā, kas var izpausties gan kā eksperimentāla, gan kā plānota testēšana. protams, šādā veidā iespējas veikt slodzes testēšanu pietuvinoties reāliem apstākļiem, nevis kā to dara testējot iegulto sistēmu pakļaujot individuāli slodzes testiem.

Ir acīmredzami, ka šādā veidā iespējams pielietot dažādās tīmekļa tehnoloģiju funkcionalitātes testēšanas automatizācijas rīku iespējas, lai, izmantojot grafisko lietotāja saskarni, veiktu saistītas iegultās sistēmas programmatūras testēšanu (kuras automatizēta testēšana nav tik vienkārši realizējama).

SECINĀJUMI

Izstrādājot iegultās sistēmas aparatūru un tai pielāgotu aparātprogrammatūras risinājumu, svarīgi ir ne tikai testēt programmnodrošinājumu, bet arī jau izstrādes laikā pilnveidot, pielāgot un testēt iekārtas projektējumu, lai tas pilnībā sekmētu aparātprogrammatūrai nepieciešamo funkcionalitāti. Aparātprogrammatūras izstrāde veicama tiklīdz pieejams iekārtas prototips, savukārt izstrādājot aparātprogrammatūru, iekārtas projektējumu iespējams pakāpeniski pilnveidot un arī papildināt īpašos gadījumos, lai rezultātā tiktu izstrādāta prasībām atbilstoša iegultā sistēma.

Iegultās sistēmas testēšanas specifika paredz speciālu testēšanai paredzētu moduļu iekļaušanu aparātprogrammatūrā, lai nodrošinātu testēšanas funkcionalitāti un testēšanai paredzēto saskarni ar, piemēram, darbstacijas lietojumprogrammu, kas papildina testēšanas funkcionalitāti un attēlo testēšanas rezultātus.

Iegulto sistēmu testēšanas metodes atkarīgas no aparātprogrammatūras pirmkoda pieejamības, jo izmantojot pirmkodu, iespējams ģenerēt programmas vadības plūsmu diagrammu, izmantojot zaru nosacījumus un to kombinācijas, lai balstoties uz to ģenerētu testus. Pretējā gadījumā iegultās sistēmas testēšana var tikt balstīta uz ievaddatiem un izvaddatiem, veidojot aptuvenas abstrakcijas pakāpes programmas darbības stāvokļu grafu, uz ko balstoties iespējams veikt galīga automāta testēšanu pēc ievaddatiem un izvaddatiem.

Iegultās sistēmas saskarne ar pārējo sistēmu paver iespējas testēt aparātprogrammatūras funkcionalitāti, izmantojot citas sistēmas testu automatizēšanas iespējas, piemēram, izmantojot tīmekļa lietojumprogrammas lietotāja saskarni, lai ar attālinātas vadības komandām testētu iegultās sistēmas funkcionalitāti.

Uzsākot rūpniecisku iegultas sistēmas ražošanu, nepieciešams ieviest risinājumu iekārtas fizisko komponenšu un to slēgumu testēšanai. nepieciešams nodrošināt testēšanu arī perifērijas iekārtām, ko iespējams veikt ar sensoru vai robotu palīdzību, aizvietojot cilvēka resursu izmantošanu un pilnībā automatizējot procesu. Izmantojot cilvēku resursus testēšanai, jārūpējas par grafisko lietotāja saskarni un testēšanas vadības sistēmu. Šādas testēšanas sistēmas izveide var tikt mērogojama kā sarežģīta sistēma, turklāt var prasīt ievērojamus programmatūras izstrādes resursus, kā arī apjomīgu testēšanu, ko sarežģī lietotāja grafiskās saskarnes testēšanas apjoms.

Darba ietvaros izdevās apzināt iegulto sistēmu projektēšanas pamatnosacījumus, kā arī iepazīties ar *Microchip* mikroprocesora *PIC24F* projektējumu un ietverto funkcionalitāti. Iegūtas zināšanas šim mikroprocesoram paredzētās aparātprogrammatūras izstrādē, kā arī tās praktiski pielietotas patstāvīgi izveidojot testēšanas moduli, kas iekļauts iegultās sistēmas

programmatūras projektējumā un tiek praktiski izmantots iekārtas rūpnieciskās ražošanas procesā.

Izstrādāta arī darbstacijas lietojumprogramma lietotāja grafiskās saskarnes nodrošināšanai ar aparātprogrammatūras testēšanas moduli. Lietojumprogrammas izstrādei autors izmantojis stāvokļu automāta vadības moduli, kas nodrošina grafiskās saskarnes skatu satura vadības un ietvertās funkcionalitātes vadību.

Autoram izdevās apzināt iegulto sistēmu testēšanas metodes un izvērtēt konkrētu testēšanas metožu pielietojumu tirdzniecības automātu telemetrijas sistēmu testēšanai.

Papildus autors izpētīja galīga automāta testēšanas paņēmienu, kuru plānots pielietot iegulto sistēmu komunikācijas protokolu testēšanai un pārsūtīto datu formāta apstiprināšanai.

Turpmākie pētījumu virzieni saistīti ar galīgu automātu modelēšanu tirdzniecības automātu telemetrijas iegulto sistēmu komunikācijas protokoliem, lai izveidotu rīku, kas nodrošina testēšanas funkcionalitāti, kuru plānots izmantot, lai ne tikai testētu telemetrijas iekārtas darbību, bet arī, lai apstiprinātu citu tirdzniecības automāta iekārtu atbilstību konkrētajiem komunikācijas protokoliem.

LITERATŪRAS SARAKSTS

1. Getting Started with MPLAB ICD. [Online] Microchip. http://www.microchip.com/stellent/groups/sitecomm_sg/documents/market_communication/~export/en022522~18~LCDDezTTP~snippet_layout/132968-2.bmp.

2. **Ron Sass, Andrew G. Schmidt.** *Embedded systems design.* s.l. : Elsevier, 2010. p. 11. 978-0-12-374333-6.

3. **Noergaard, Tammy.** *Embedded Systems Architecture.* bez viet. : Elsevier, 2005. lpp. 7.,590. - 592.,. 0-7506-7792-9.

4. **Shuttleworth, Martyn.** Heron's Inventions. *Experiment-Resources.com.* [Online] 2011. http://www.experiment-resources.com/heron-inventions.html.

5. Uzņēmumu datu bāzes. *Lursoft.* [Tiešsaiste] 2010. gada 30. 11. http://www.lursoft.lv/uznemuma-pamatdati/latvian-vending-association/40008168695.

6. Web software for online minitoring and analysis. [Tiešsaiste] Vendon. http://vendon.net/static/images/screenshots/monitoring_devices.png.

7. Operating Instructions for the currenza clip MDB currenza clip Multi Interface. *Crane Payment Solutions GmbH.* [Online] http://www.nri.de/download/PDF_Englisch/BA_c_clip_EN.pdf.

8. Telesoft Technologies Products to Drive Skyrocketing Location-Based Services Market. *Prlog Press Release Distribution.* [Tiešsaiste] Prlog, 2008. gada 2. 12. http://www.prlog.org/10149430-hinton-locator-uses-network-data-for-location-via-triangulation.jpg.

9. About QRcode.com. *QR Code.com.* [Online] http://www.denso-wave.com/qrcode/index-e.html.

10. Development Tools Main Page. [Tiešsaiste] Microchip, 2012. gada. [Citēts: 2012. gada 16. 05.] http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002.

11. Project home. *CollabNet.* [Online] 2012. http://sharpsvn.open.collab.net/.

12. **Kshirasagar Naik, Priyadarshi Tripathy.** *Software testing and quality assurance Theory and practice.* s.l. : Wiley, 2008. pp. 16., 51 - 53., 158. - 159. 978-0-471-78911-6.

13. **Bart Broekman, Edwin Notenboom.** *Testing Embedded Software.* Sogeti : s.n., 2003. pp. 4. - 6. 0-321-15986-1.

14. **Ari Takanen, Jared D. Demott, Charles Miller.** *Fuzzing for Software Security Testing and Quality Assurance.* Norwood : Artech house, 2008. lpp. 1. - 6. 978-1-59693-214-2.

15. **Dssouli, Rachida.** Model Based testing:FSM-based Testing. [Online] 10 2007. [Cited: 5 11, 2012.] http://users.encs.concordia.ca/~dssouli/COEN345F07%20Folder/FSM-Testing.pdf.

16. **Ural, Hasan.** *Formal methods for test sequence generation.* bez viet. : Butterworth-Heinemann Ltd, 1992. lpp. 311. - 317. 0140-3664/92/005311.

17. **Edwards, Stephen H.** *Black-Box Testing Using Flowgraphs: An Experimental Assessment.* Blacksburg : Wiley, 2000. pp. 2. - 6. 24061-0106.

18. **Sun, Xiao.** *Protocol conformance testing using unique input/output sequences.* Singapore : River Edge, NJ : World Scientific, 1997. pp. 33. - 38. 9810228325 9789810228323.

19. Lessons to Get Started on Windows Workflow Foundation. [Online] [Cited: 5 10, 2012.] http://static.flickr.com/97/257927810_520523a6dc_o.png.

20. **Jacobs, Ron.** *Building State Machine Workflows with Windows Workflow Foundation.* Atlanta : Microsoft, 2011.

Maģistra darbs: **„Iegulto sistēmu testēšana un ieviešana”**

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____
(Autora paraksts)

Ar savu parakstu apliecinu, ka esmu lasījis augstāk minēto maģistra darbu un atzīstu to par **p i e m ē r o t u / n e p i e m ē r o t u** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu maģistrantūrā.

Darba vadītājs(-ja): _____
(Vadītāja paraksts)

Darbs iesniegts **maģistrantūras sekretariātā**: _____.
(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.
Studiju metodiķe: _____.
(Metodiķes paraksts)

Recenzents: profesors, Dr.dat. Guntis Arnicāns

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____, vērtējums _____
(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____
(Sekretāra paraksts)