LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

DAUDZKANĀLU SKAŅAS KODEKA FUNKCIJU TESTU AUTOMATIZĀCIJA

KVALIFIKĀCIJAS DARBS PROGRAMMĒŠANĀ

Autors:

Ēriks Burtnieks, eb15032

Darba vadītāja:

Vadošā pētniece Dr. dat. Vineta Arnicāne

RĪGA, 2017

ANOTĀCIJA

Dokuments sniedz ieskatu skaņas pults un kodeka testu automatizācijā. Dokumentā aprakstītā programmatūra ir šo iekārtu funkciju testu automatizācijas skripti.

Šo skriptu mērķis ir atvieglot testēšanas procesu. Testu automatizācija nodrošina ātrāku, vienkāršāku un efektīvāku testēšanas procesu. To galvenā priekšrocība ir iespēja tos atkārtoti izmantot, piemēram, kādas funkcijas izmaiņu gadījumā – tie nodrošina to, ka kādu funkciju izmaiņu gadījumā var ātri un ērti pārbaudīt vai konkrētās izmaiņas nav izmainījušas kādas citas funkcijas darbību un ļauj ātrāk atrast, identificēt un salabot jebkādas šāda veida problēmas.

Gala rezultāts uzlabo iekārtu testu nodrošinājumu un uzlabo izstrādes procesu noņemot testēšanas slogu no testētājiem un ļaujot tiem pievērsties citu testu izpildei un uzlabojot gala programmatūras produkta kvalitāti.

Atslēgas vārdi: Testēšana, automatizācija, skaņas iekārtas, skaņas programmatūra

ABSTRACT

AUTOMATIZATION OF TESTS FOR FUNCTIONS OF MULTICHANNEL AUDIO CODEC

This document provides insight into automatization of tests of audio console and codec. The software described by this document are the automatization scripts for the functions of these hardware devices.

The aim of these scripts is to ease the testing process. Automatization of tests provides faster, simpler and more effective testing process. The main advantage is the possibility to reuse them in case of for example any changes in a specific function – running these tests after said changes provides a quick and handy way to check if the changes have not had some unexpected and unwanted effect on other function and allows for a faster spotting, identifying and fixing of any problems of such nature.

The final result improves the test coverage of the hardware and improves the development process by taking some burden off from testers thus allowing them to focus on other, not yet automated tests thus covering more functions and improving the quality of the software end product.

Keywords: Testing, automatization, audio hardware, audio software

SATURS

1. VĀRDNĪCA....6
2. IEVADS....7
   2.1. Nolūks....7
   2.2. Darbības sfēra....7
   2.3. Saistība ar citiem dokumentiem....7
   2.4. Dokumenta pārskats....7
3. VISPĀRĪGAIS APRAKSTS....8
   3.1. Testējamās sistēmas apraksts....8
   3.2. Programmatūras perspektīva....9
   3.3. Programmatūras funkcijas....9
   3.4. Lietotāja raksturiezīmes....9
   3.5. Vispārējie ierobežojumi....9
   3.6. Pieņēmumi un atkarības....10
4. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA....11
   4.1. Funkcionālās prasības....11
      4.1.1. Vispārīgā informācija....11
      4.1.2. Prasību interpretācija....11
   4.2. Programmatūrā izmantotās valodas vispārīgs apraksts....12
   4.3. Valodas komandu apraksts....13
      4.3.1. Konfigurācijas komandas....13
      4.3.2. Ievades komandas....14
      4.3.3. Izvades komandas....16
   4.2. Nefunkcionālās prasības....17
      4.4.1. Veiktspējas prasības....17
      4.4.2. Uzturamība....17
5. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS....18
   5.1. Vispārīgs apraksts....18
   5.2. Uzbūve....18
   5.3. Konfigurācijas daļas uzbūve....19
   5.4. Pārbaudes daļas uzbūve....20
   5.5. Pārbaudes daļas skripta izveide....21
6. TESTĒŠANA....25
7. PROGRAMMATŪRAS PROJEKTA DARBA ORGANIZĀCIJA....27

8. KVALITĀTES NODROŠINĀŠANA 28
9. KONFIGURĀCIJAS PĀRVALDĪBA 29
10. DARBIETILPĪBAS NOVĒRTĒJUMS 30
11. SECINĀJUMI 31
PIELIKUMS 32
1. Testa scenārija apraksts .....
2. Programmatūras pirmkoda fragments .....

1. VĀRDNĪCA

Git – Programmatūras versiju kontroles sistēma, kas nodrošina koda glabāšanu attālināti, nodrošina rezerves kopiju un iepriekšējo versiju pieejamību, kā arī atvieglo koda pieejamību izstrādes komandas ietvaros.

GPIO – *General Purpose Input-Output* – vispārēja ievadizvade. Darba ietvaros šis saprotams kā ārējs panelis skaņas pultij, kas nodrošina papildus funkcionalitāti.

Interpretators – Programmatūra, kas nodrošina pēc definētiem noteikumiem veidota strukturēta teksta secīgu apstrādi, veicot definētas darbības.

Kodeks – *Codec* – Iekārta digitālas datu straumes šifrēšanai un atšifrēšanai no analogā formāta.

Tīkla mezgls – *Node* – Tīkla skaņas iekārta, kas nodrošina dažādu iekārtu savstarpējo darbību.

Skaņas pults/konsole – *Console* – Iekārta dažāda veida skaņas ierakstu studijas darbību vadībai.

Skripts – *Script* – Programmatūras veids, izmantots lai automatizētu liela apjoma vienkāršu komandu kopumus.

2. IEVADS

2.1. Nolūks

Kvalifikācijas darbs ir izstrādāts ar mērķi uzskatāmā un kodolīgā veidā aprakstīt daudzkanālu skaņas kodeka funkciju testu automatizācijas procesu. Procesa gaitā esoši testa scenāriji atbilstoši to prasībām, kas ietver darbību un to rezultātu pārbaudi, tiek automatizēti.

2.2. Darbības sfēra

Dokumentā aprakstītā programmatūra ir testu automatizācijas skripti, izstrādāti ar mērķi paātrināt, atvieglot un padarīt precīzāku skaņas kodeka funkciju darbības pārbaudi, kas nodrošina vienkāršāku un ātrāku iespējamo problēmu atrašanu un tādējādi likvidēšanu.

2.3. Saistība ar citiem dokumentiem

Dokuments ir izstrādāts balstoties uz standartiem, kas definēti "LVS 68:1996 Programmatūras prasību specifikācijas ceļvedis" dokumentā.

2.4. Dokumenta pārskats

Dokuments ir sadalīts 11 nodaļās. 3 nodaļas veltītas darba izklāstam – vispārīgs apraksts, programmatūras prasību specifikācija un programmatūras projektējuma apraksts. 5 nodaļas veltītas darba izstrādes procesa analīzei – tās satur informāciju par testēšanu, darba organizāciju, kvalitātes nodrošinājumu, konfigurācijas pārvaldību kā arī darbietilpības novērtējumu. 2 nodaļas veltītas ievadam un nobeigumam, kā arī 1 nodaļa, kura satur vārdnīcu. Dokumentam ir arī 2 pielikumi, kuros atrodas dotā testa scenārija oriģinālais apraksts un gala programmatūras pirmkods. Detalizēts pārskats atrodams satura rādītājā.

3. VISPĀRĪGAIS APRAKSTS

3.1. Testējamās sistēmas apraksts

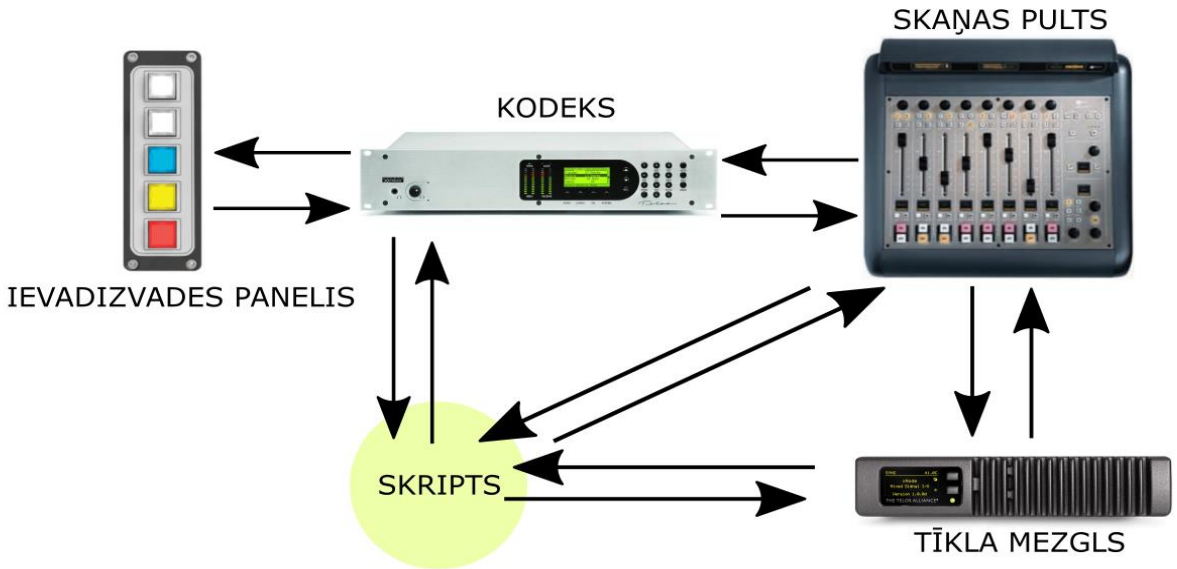
Sistēma var sastāvēt no daudzu veidu iekārtām. Šī darba ietvaros tiek izmantoti četri iekārtu veidi – skaņas pults, kodeks, attālinātās ievadizvades panelis un tīkla mezgls.

Skaņas pults ir studijas vadības panelis. Tā sastāv no vairākiem kanāliem, katram kanālam ir iespējams neatkarīgi mainīt tā konfigurāciju, skaņas līmeni, skaņas avotu, kā arī signāla galamērķi. Darba ietvaros pults tiek izmantota kā bāze visām darbībām.

Kodeks ir iekārta, kas nodrošina analoga signāla kodēšanu un atkodēšanu digitālā formātā. Darba ietvaros kodeks nodrošina attālinātās vadības paneļa funkcionalitāti, kas arī ir veidojamo automatizēto testu galvenais objekts.

Attālinātās ievadizvades panelis – ārēja iekārta, ar kuras palīdzību var izmantot dažāda veida papildus funkcijas.

Tīkla mezgls – iekārta, kas nodrošina daudzas tīklā esošajām skaņas iekārtām nepieciešamās funkcijas. Darbā to izmanto tikai skaņas līmeņa pārbaudēm.



3.1 attēls

Attēlā 3.1 vispārīgā veidā parādīts kā iekārtas un automatizācijas skripts ir savstarpēji savienoti. Kā redzams, skripts izveido tiešus savienojumus ar trim no minētajām iekārtām. Katrai

no šīm iekārtām skripts var sūtīt darbību uzdevumus vai pieprasīt to stāvokli. Ievadizvades panelis tiek pārbaudīts caur kodeka savienojumu.

Savā starpā izveidoto iekārtu savienojumu mērķis ir nodrošināt dažāda veida darbību izraisītu datu plūsmu. Attiecīgi, skriptā nosūtot darbības uzdevumu skaņas pultij, šīs darbības rezultāts var ietekmēt arī pārējo iekārtu stāvokli. Šajā gadījumā skaņas pults tiek savienota ar kodeku, kurš savukārt tiek savienots ar ievadizvades paneli, tādējādi nodrošinot galveno testējamo savienojumu. Skaņas pultij tiek arī pievienots tīkla mezgls, ar kura palīdzību tiek noteikts skaņas līmenis. Tīkla mezglu ir iespējams aizvietot ar jebkādu citu iekārtu, kurā iespējams noteikt skaņas līmeni, piemēram, cita skaņas pults. Šādam savienojumam gan būtu jāveic skripta modifikācijas.

3.2. Programmatūras perspektīva

Programmatūras produkts ir testu automatizācijas skripts – dalāmu, neatkarīgu testu kopums ar vienotu sākotnējo konfigurāciju.

3.3. Programmatūras funkcijas

Automatizēti veikt iepriekš definētu testa scenāriju izpildi un rezultātu ierakstīšanu. Aprakstītie testa scenāriji pārbauda skaņas kodeka ievadizvades funkcijas. Testus iespējams pielāgot dažādām iekārtām un to daļām. Testi atvieglo un paātrina testēšanas procesu, kā arī uzlabo gala produkta kvalitāti, jo tos iespējams ar nelielu resursu patēriņu ieslēgt vairākkārt izstrādes cikla laikā, lai pārbaudītu vai kāda pievienotā funkcionalitāte nav ietekmējusi jau esošo.

3.4. Lietotāja raksturiezīmes

Programmatūras lietotājs ir testētājs. Testētājam ir pieredze testa scenāriju veidošanā, izpildīšanā un analīzē. Testētājam ir zināšanas un pieredze automatizētu testu veidošanā no esošiem scenārijiem un to izpildīšana un analīze. Testētājam ir pieredze tieši šajā valodā veidotu automatizētu testu rezultātu analīzē, kā arī zināšanas par šo testu veidošanu un modificēšanu.

3.5. Vispārējie ierobežojumi

Programmas produkta lietošanai nepieciešams specializēts interpretators, kā arī tiešs pieslēgums testējamajai iekārtai, dokumentā aprakstīto testa skriptu un scenāriju gadījumā 3 iekārtām – skaņas pultij, kodekam, kurš ļauj sūtīt un pārbaudīt komandas uz ievadizvades paneli, kā arī kādai iekārtai, uz kuras pārbaudīt skaņas līmeni. Šī darba ietvaros veidoto skriptu gadījumā arī nepieciešams, lai skaņas pults konfigurācija nesatur iepriekš izveidotus skaņas avotu tipus, jo

šī programma tos veido un dzēš darbības laikā (kas reizē arī nodrošina šīs funkcionalitātes vienkāršotu pārbaudi).

3.6. Pieņēmumi un atkarības

Tiek pieņemts, ka lietotājam ir zināšanas par skaņas pults un ievadizvades paneļa darbību un interpretatora veidoto rezultātu analīzi balstoties uz testa scenāriju. Tiek pieņemts, ka lietotājam ir tieša pieeja skaņas pultij un ievadizvades panelim, kā arī lietotājam ir pieejams interpretators vai iespēja to izmantot.

4. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

4.1. Funkcionālās prasības

4.1.1. Vispārīgā informācija

Funkcionālās prasības tika uzdotas strukturētā formā. Tās sadalītas vairākos lielos testa scenārijos, kuram katram ir iespējamas vairākas apakšsadaļas. Darba gaitā tika izstrādāta viena šāda apakšsadaļa, kuras ietvaros tika automatizēti testi kodeka attālinātās kontroles ievadizvades paneļa funkcijām 9 dažādiem skaņas avota veidiem. Katram skaņas avota veidam tika uzrakstīti aptuveni 12 atsevišķi testa soļi, katram no soļiem veicot gan priekšnosacījumu izpildi, gan funkciju ievadi, gan rezultātu pārbaudi. Testa scenāriji tiek doti angļu valodā, šajā nodaļā ir ievietots neliels fragments gan oriģinālvalodā (angļu), gan tulkojums latviešu valodā un skaidrojums par šī scenārija interpretēšanu un skripta izveidi atbilstoši šīm prasībām.

Pilna specifikācija oriģinālvalodā ir pieejama pielikumā Nr. 1 (Testa scenārija apraksts).

4.1.2. Prasību interpretācija

Šeit redzams fragments oriģinālvalodā no scenārija specifikācijas un tā tiešs tulkojums:

Channel is ON. GPIO: press and release panel button #2 (OFF)

- *Console turns the channel OFF: audio, indications*
- *Panel lamp #2 (Off) lights up*
- *Panel lamp #5 (Stop) flashes for 100 ms*
- *Panel lamp #4 (Start) stays off*
- *Panel lamp #3 (Preview) follows the preview state*
- *Panel lamp #1 (On) goes out*

Kanāls ir ieslēgts. Ievadizveds panelis: nospiediet un atlaidiet paneļa pogu Nr. 2 (Izslēgt)

- *Konsoles kanāls tiek izslēgts: skaņa, apgaismojums*
- *Paneļa lampa Nr. 2 (Izslēgts) ieslēdzas*
- *Paneļa lampa Nr. 5 (Stop) ieslēdzas uz 100 milisekundēm*
- *Paneļa lampa Nr. 4 (Sākt) paliek izslēgta*

- *Paneļa lampa Nr. 3 (Priekšskatījums) saglabā iepriekšējo stāvokli*

- *Paneļa lampa Nr. 1 (Ieslēgts) izslēdzas*

Līdzīgā formātā ir uzdots viss scenārijs. Tā interpretēšana tiek sākta ar avota tipa noskaidrošanu. Dotais piemērs to nesatur, jo šis avota tips tiek norādīts vienreiz pirms visiem šī tipa testiem. Piemēra situācijā tas nav nozīmīgi, jo ir doti visi sagaidāmie notikumi, bet ir svarīgi lai iepriekš uz kanāla būtu izvēlēts tieši norādītais avota tips, jo dažādiem tipiem atšķiras iekārtu uzvedība dažādu komandu gadījumā.

Turpinot interpretēšanu, svarīgākais, kam jāpievērš uzmanība ir tieši scenārija soļa apraksts, kas satur divu veidu svarīgu informāciju – priekšnosacījumu pirms soļa izpildes un soļa izpildes laikā veicamo darbību. Priekšnosacījums vienmēr tiek norādīts sākumā un darbība seko tam. Šajā piemērā priekšnosacījums ir sekojošs – kanālam ir jābūt ieslēgtam. Tas nozīmē, ka veicot darbību situācijā, kad kanāls ir, piemēram, izslēgts, visdrīzāk rezultāts būs pilnīgi cits un tests neizpildīsies. Priekšnosacījumam seko darbība – uz ievadizvades paneļa jānospiež un jāatlaiž poga Nr. 2 – izslēgt.

Priekšnosacījuma izpildes iekļaušana testa skriptā ir atkarīga no iepriekšējā testa soļa – ja tā izpildes rezultātā priekšnosacījums izpildās, nav nepieciešams to īpaši norādīt. Savukārt ja iepriekšējā soļa rezultātā iekārta ir cita stāvoklī, nepieciešams izpildīt nepieciešamās darbības, lai iekārta nonāktu nepieciešamajā stāvoklī. Darbības izpilde savukārt jānorāda vienmēr.

Nākošais solis ir rezultāta prasību analīze, jeb visas aprakstam sekojošās rindiņas. To secība scenārijā nav svarīga, galvenais ir pareizi veikt visas pārbaudes. Šajā piemērā ir 6 rindiņas – pirmā norāda, ka darbības rezultātā pults kanālam ir jāizslēdzas un attiecīgi jāizslēdz arī izejošā skaņa un apgaismojums. Pārējās rindiņas norāda darbības, kam būtu jānotiek ar ievadizvades paneļa lampiņām. Tās var ieslēgties, izslēgties, nemainīt stāvokli vai īslaicīgi ieslēgties.

4.2. Programmatūrā izmantotās valodas vispārīgs apraksts

Programmatūras izstrādē izmantota īpaši izveidota specializēta valoda, kurā ērti lietojamā veidā iespējams automatizēti pārbaudīt gala programmatūras un aparatūras darbību. Valoda ir interpretēta un šo interpretāciju veic valodā “Ruby” īpaši šim nolūkam radīts interpretators.

Valodas galvenās funkcijas ir:

- Savienojuma izveide ar iekārtu
- Iekārtu konfigurācija

- Dažādas palīgfunkcijas, piemēram, aizture
- Komandu ievade
- Programmatūras/aparatūras stāvokļa pārbaude
- Programmatūras/aparatūras stāvokļa salīdzināšana ar testa scenārijā prasīto

4.3. Valodas komandu apraksts

Valodā pieejamās komandas var iedalīt trīs grupās – palīgkomandas, piemēram, sākotnējas konfigurācijas uzstādīšanai, iekārtu savienojumu izveidei u.tml., darbību izpildes jeb ievades komandas, kā arī rezultātu pārbaudes jeb izvades komandas.

4.3.1. Konfigurācijas komandas

Pirmais komandu veids ir komandas, kuras tiek izmantotas vispārējai skripta konfigurācijai un lietojamībai, piemēram, iekārtu savienojumi, palaišanas parametri, aiztures, izvade uz ekrāna un citas. Tām nav vienotas struktūras kā iepriekš izklāstītajiem komandu veidiem.

Daži šāda veida komandu piemēri:

```
S1 = ${consoleaddress}:93

sleep 0.75

<< Starting test
```

Komanda ‘S1 = ’ ir domāta, lai piesaistītu skriptam kādu konkrētu iekārtu. Tai sekojošā virkne ‘${consoleaddress}’ savukārt ir parametrs, kurš skriptam jānodod to palaižot. Šajā gadījumā tā būtu pults adrese tīklā. Tam seko porta numurs uz iekārtas, tā kā šī konkrētā iekārta tiks izmantota ievadizvades pārbaudēm, tiek izmantots ports 93. Pēc šīs rindiņas nolasīšanas visur skriptā turpmāk atbilstoši iepriekš aprakstītajam var izmantot komandu ‘S1:’ lai sūtītu komandas un nolasītu rezultātus tieši no šīs iekārtas.

Komanda ‘sleep’ liek interpretatoram uz norādīto laika sprīdi apturēt skripta lasīšanu, tādējādi ļaujot komandām netraucēti izpildīties un atgriezt rezultātus. Tas vajadzīgs vairāku iemeslu dēļ – tīkls var būt noslogots un radīt traucējumus un noildzi komandu plūsmā, rezultāti, kā jau iepriekš minēts, tiek izvadīti secīgi un tādējādi to pilnvērtīgai nolasīšanai ir nepieciešams sagaidīt tos visus, kā arī, lai pats skripts lieki nenoslogotu tīklu un netraucētu citiem paralēli ieslēgtiem skriptiem. Noildzes laiks tiek norādīts aiz komandas un ir konfigurējams ļoti plaši, sākot

no dažām milisekundēm līdz pat vairākām minūtēm. Tā kā šajā darbā noildze tiek izmantota ļoti daudz, svarīgi atrast līdzsvaru starp veiksmīgu komandu izpildi un skripta ātrdarbību.

Komanda '<<' norāda izvadi uz ekrāna un skripta rezultātu žurnālā (ko interpretators veido paralēli skripta izpildei). Aiz šīs komandas var sekot jebkāds teksts, tomēr svarīgi, lai tam būtu saturīga jēga, tādējādi atvieglojot testa rezultātu analīzi.

Ir pieejama arī komanda koda komentēšanai, jeb komanda, kura norāda interpretatoram, ka sekojošā rindiņa ir jāizlaiž. To apzīmē ar '#', un tā ir ļoti svarīga pārskatāma un modificējama koda izveidei.

4.3.2. Ievades komandas

Darbību izpildes komandas tiek izmantotas, lai kādai iekārtai nosūtītu uzdevumu, kurš tai pēc komandas saņemšanas jāizpilda. Šīs komandas ir atšķirīgas dažādiem iekārtu veidiem, piemēram, komandas, kas domātas skaņas pultij nav iespējams izmantot skaņas kodeka darbību veikšanai un otrādi. Tādējādi ievades komandas ir atkarīgas no nepieciešamajā iekārtā esošās programmatūras. Neatkarīgi no iekārtas veida, visas darbību izpildes komandas tiek sūtītas vienādi – kodā katrai iekārtai iedarbinot skriptu tiek piešķirts konkrēts iekārtas numurs balstoties uz attiecīgās iekārtas adresi tīklā, un visas komandas tiek sūtītas izmantojot šo iekārtas numuru. Piemēram:

```
S1: SET FaCH#5 ON_State=ON
```

Šajā piemērā 'S1' apzīmē iepriekš definētu iekārtu. Iekārta šajā gadījumā ir skaņas pults. Ievades komanda ir 'SET' – tā norāda, ka tiks mainīta kāda no skaņas pults funkciju vērtībām. Pēc komandas seko skaņas kanāla numurs, ko norāda ar 'FaCH#X', kur X ir numurs. Tas norāda kuram skaņas kanālam sekojošā darbība ir adresēta. Pēc kanāla numura tiek rakstīta darbība. Piemērā tiek mainīta kanāla vispārējā stāvokļa vērtība – tas var būt vai nu ieslēgts, vai izslēgts. Šī komanda to ieslēdz.

Vēl kāds piemērs skatāms no ievadizvades paneļa komandām:

```
S2: GPI 5 Lhhhh
```

```
S2: GPI 5 Hhhhh
```

Līdzīgi kā iepriekš, vispirms tiek norādīta mērķa iekārta, šajā gadījumā ievadizvades panelis. Tālāk attiecīgi seko komanda 'GPI', *General Purpose Input,* jeb vispārēja ievade, kas norāda, ka kāda darbība jāizpilda. Pēc tam seko paneļa kanāla numurs, līdzīgi kā iepriekšējā

piemērā. Beigās esošā virkne ‘lhhhh’ norāda gala stāvokli, kur ‘L’ vai ‘l’ nozīmē poga nospiesta un ‘H’ vai ‘h’ nozīme poga nav nospiesta. Lielie un mazie burti ievades komandā nav svarīgi no komandu izpildes viedokļa, bet uzlabo koda lasāmību, jo pārskatāmā veidā norāda ar kuru pogu tiek veikta darbība, tādēļ vēlams tos izmantot. Svarīgi arī pievērst uzmanību soļa aprakstam – ja tajā ir norādīts, ka poga tiek nospiesta un šī paša soļa ietvaros arī atlaista, tas jāizdara, lai nerastos liekas kļūdas. Arī virknes secība ir svarīga un atbilst paneļa pogu/lampiņu numurējumam testa scenārija aprakstā, t.i. šajā gadījumā tiek nospiesta un tad atlaista poga Nr. 1. Šāda veida ievades rezultāts parasti ir atkarīgs no skaņas avota tipa, jo katram tipam paneļa pogu funkcijas nedaudz atšķiras.

Nedaudz atšķirīgā formātā tiek ievadītas komandas uz konfigurācijas WEB lapām – tās visas apzīmē rindiņa, kura sākas ar simbolu virkni ‘WWW’. Pēc tās seko nepieciešamā komanda. Šeit redzami daži piemēri:

```
WWW= #follow#http://192.168.1.123/outputs user a
WWW get_form 1
WWW select lwout_2 255
WWW text lwch_2 25091
WWW button lw_save
```

Ja pēc ‘WWW’ seko ‘= #follow#’, tad tiek norādīts, ka jāatver tam sekojošā WEB lapa, kurai tiek norādīta adrese. Atvērt iespējams jebkādu adresi. Pēc adreses vēl var sekot arī autorizācijas dati, dotajā piemērā ‘user a’, kur ‘user’ ir lietotājvārds un ‘a’ ir parole.

Ir pieejama arī komanda ‘get_form’, kura norāda, ka iepriekš ielādētajā lapā darbības tiks veiktas attiecīgajā formā. Nepieciešamo formu atrod izmantojot tīmekļa pārlūkā izmantojot pirmkoda skatīšanas funkciju. Atkarībā no lapas forma var būt viena vai vairākas, svarīgi ir atrast un norādīt pareizo.

Vēl ir pieejamas komandas konkrētu darbību veikšanai šajā formā. Vispirms norāda ievades elementa veidu, piemēram, teksta lauks, radio pogas, izvēles rūtiņa, pogas u.tml. Aiz ievades elementa veida norādes seko nepieciešamā elementa identifikators formas ietvaros. Arī visu šo informāciju atrod izmantojot pārlūka pirmkoda skatīšanas funkciju. Atkarībā no ievades elementa veida pēc tā var sekot arī nepieciešamā vērtība. Piemēram, izvēlnes sarakstam tas ir vajadzīgās izvēles identifikators, arī to atrod caur pārlūku. Teksta laukam savukārt uzreiz tiek norādīta tā vērtība. Pogas WEB lapās tiek lietotas lai attiecīgajā formā veiktās izmaiņas apstiprinātu, ja

elementa veids ir norādīts kā poga, tad tā tiek nospiesta, līdz ar to nekāda papildus vērtība netiek norādīta.

4.3.3. Izvades komandas

Rezultātu pārbaudes jeb izvades komandas tiek izmantotas, lai pēc darbību izpildes varētu pārbaudīt, vai gala rezultāts sakrīt ar sagaidāmo. Pārbaudes komandas atgriež rezultātu 'OK', ja pēdējās iekārtas atbildes sakrīt ar gaidāmo rezultātu, vai 'NOK', ja nesakrīt. Skriptā gaidāmajiem rezultātiem jābūt tieši tādiem pašiem kā iekārtas atbildei, t.i. secībai jāsakrīt. To pārsvarā var nodrošināt tikai palaižot skriptu un redzot reālo rezultātu un attiecīgi to pielabojot. Tās tiek strukturētas nedaudz atšķirīgi no ievades komandām – jebkura izvades komandas tiek sākta ar simbolu '?'. Piemērs:

```
? LAST_S1 EVENT FaCH#5 ON_State=ON
```

Pēc tam atkarībā no pārbaudes veida seko pārbaudes tips – piemērā to apzīmē 'LAST_S1', kur 'LAST' norāda uz iekārtas pēdējās izvades saturu un 'S1' atkal norāda uz iepriekš definētu iekārtu – svarīgi, lai tiktu pārbaudīta tā pati iekārta uz kuru tika sūtīta komanda, citādi rezultāts būs negatīvs. Pēc tam seko sagaidāmais rezultāts – šīs iekārtas (skaņas pults) gadījumā, vispirms tiek sagaidīts atslēgas vārds 'EVENT', kas norāda ka iekārtā notika kaut kāda veida izmaiņas, ko izraisīja iepriekš ievadīta komanda. Pēc šī atslēgas vārda jau atkal seko kanāla numurs un tam seko izmaiņas rezultāts. Svarīgi piebilst, ka pārbaude neizdosies, ja uz iekārtas nekādas izmaiņas nenotiks, piemēram, ja iekārta jau pirms darbības izpildes atradās attiecīgajā stāvoklī.

Līdzīgā veidā tiek pārbaudīts arī ievadizvades paneļa rezultāts:

```
? LAST_S2 GPI 5 Lhhhh|nGPI 5 Hhhhh|nGPO 1 Llhhh|nGPO 1 lHhhh
```

Šeit pārbaudes kļūst daudz sarežģītākas. Ievadizvades panelis katru lampiņu ieslēdz vai izslēdz atsevišķi un noteiktā secībā, un katras lampiņas, kura darbības rezultātā var mainīt stāvokli, pārbaudei ir jāizveido pārbaude. Parasti, bet ne vienmēr, lampiņu stāvokļi tiek mainīti sākot no pirmās un beidzot ar pēdējo lampiņu, no koda viedokļa raugoties – no kreisās puses uz labo. Ja secība kāda iemesla dēļ ir savādāka, to var pamanīt tikai testējot izmaiņas un attiecīgajā vietā šo secību nomainīt. Šajā piemērā pēc 'LAST_S2' komandas seko virkne ar pārbaudēm, šeit četras, atdalītas ar jaunas rindas simbolu '|n', jo izmaiņas tiek izvadītas secīgi, katra savā rindiņā. Pārbaudīts tiek uzreiz viss pēdējo ievades un izvades komandu bloks. Pirmās divas pārbaudes ir attiecībā uz iepriekš aprakstīto ievadi, kur vispārējās ievades kanāla Nr. 5 pirmā poga tiek nospiesta

un atlaista. Nākošās divas pārbaudes, ‘GPO’, jeb *General Purpose Output* – vispārējā izvade, pārbauda lampiņu stāvokļa maiņas. Pirmā komanda pārliecinās vai atbilstoši lampiņu secībai vispirms iedegās pirmā lampiņa, savukārt otrā komanda pārbauda vai pēc tam izdziest otrā lampiņa. Atbilstoši soļa prasībām komandu skaits var mainīties.

Valodas ietvaros ir iespējams veikt arī pārbaudes iekārtu konfigurācijas WEB lapās, bet šī darba ietvaros tas netiek darīts, tādēļ šī iespēja sīkāk aprakstīta netiks.

4.2. Nefunkcionālās prasības

4.4.1. Veiktspējas prasības

Lai iespējami paātrinātu testēšanas procesu un lieki nenoslogotu tīklu, testa skriptiem jāizpildās iespējami īsākā laikā. Tā kā komandu rezultāti parādās ar nelielu aizkavēšanos, skriptos ir komandas, kas skriptā izraisa īsu aizturi, lai sagaidītu rezultātus. Veiktspēja tiek paātrināta veidojot šīs aiztures maksimāli īsas.

4.4.2. Uzturamība

Programmatūrai ir jābūt viegli modificējamai un funkcijām – iespējami neatkarīgam vienai no otras, lai tās varētu vajadzības gadījumā izmantot citos skriptos. Tāpat tai jābūt pārskatāmai un tajā jāspēj viegli atrast konkrēti scenārija soļi, tieši ar mērķi izpildīt iepriekš minētās prasības.

5. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

5.1. Vispārīgs apraksts

Šī nodaļa vispārīgā veidā apraksta gala programmatūras produkta – skripta – struktūru un mērķus. Tā analizē arī skripta paraugu, izskaidrojot tā lasīšanu, mērķa izpratni un modifikācijas iespējas un metodes.

5.2. Uzbūve

Atbilstoši dotā scenārija prasībām, jāpārbauda iekārtu darbība 9 dažādiem skaņas avotu tipiem, kā arī pāris pārbaudes jāveic iekārtām bez iestatīta skaņas avota. Lai uzlabotu testēšanas iespējas, kods ir jāizveido divējādi – pirmkārt, visi testi vienā datnē, kas nodrošina pilnu scenārija pārbaudi veicot tikai vienu darbību, otrkārt, katram avotam izveidojot savu datni, lai nodrošinātu daudz ātrāku pārbaudi, piemēram, pārtestējot tikai atsevišķus avota tipus vai mainot kādu iekārtu funkciju, kas var ietekmēt tikai atsevišķus avotu tipus. Šī ir svarīga prasība, jo pārbaužu skaits ir samērā liels un var aizņemt daudz laika. Tas ne tikai potenciāli apgrūtina un paildzina testēšanas procesu, bet arī ievieš risku, ka ilgās testēšanas laikā var gadīties kāda kļūme ar iekārtām vai tīklu tādējādi spiežot visu testu sākt no jauna.

Gan vienotā, gan sadalītās datnes ir jāsadala divās daļās – pirmā daļa ir sākotnējā konfigurācija, kas nodrošina savienojumus ar iekārtām, iekārtu iestatīšanu noteiktos stāvokļos, savstarpējo savienojumu izveidi kā arī globālu skaņas avotu izveidi, lai nodrošinātu skaņas pārbaužu iespējas. Konfigurācijas galvenais mērķis ir savienojumi starp galvenajām iekārtam – skaņas pulti un kodeku, lai varētu veikt vispārējās ievadizvades pārbaudes, skaņas pārbaudes ir sekundāra funkcija. Šī pirmā daļa ir nepieciešama, lai atvieglotu skripta startēšanu. Tā nodrošina, ka iekārtas tiek pilnībā nokonfigurētas skripta iekšienē. Bez šīs konfigurācijas daļas pilnīgi visa konfigurēšana būtu jāveic manuāli, kas ir ļoti ilgstošs un sarežģīts process, turklāt to veicot manuāli ir ļoti viegli izveidot kļūdainu savienojumu, kā rezultātā tests visdrīzāk neizpildīsies korekti un kā cēloņa atrašana var būt ļoti laikietilpīga. Ir apzināts risks, ka atsevišķas konfigurācijas darbības interpretatora ierobežojumu dēļ ir ļoti sarežģīti veikt automātiski, bet šādu darbību skaitam ir jābūt minimālam un pie skripta jābūt skaidrām norādēm kā tās veikt manuāli.

Otrā jebkuras datnes daļa ir pārbaudes. Vienotā datne pēc būtības tiek izveidota kā skripta pamats, jo iekļauj pilnu scenārija pārbaudi. Individuālās datnes pēc tam jāizveido no šīs datnes

skripta. Sākotnējās skripta versijas izstrādē, šo var veikt kā vienu no pēdējiem soļiem, lai izvairītos no paralēlas daudzu datņu uzturēšanas, bet skripta izmantošanas un uzturēšanas periodā jebkādas izmaiņas būtu jāveic visās datnēs, kuras šīs izmaiņas skar. Tas apgrūtina uzturēšanu, bet iepriekš minēto iemeslu dēļ uzlabo testēšanas kvalitāti. No scenārija viedokļa katram avotam ir savas pārbaudes, jo dažādiem skaņas avotu tipiem ir savādāka noklusētā konfigurācija attiecībā uz darbībām, kuras iespējams izpildīt ar skaņas pulti un ievadizvades paneli un līdz ar to katram avota tipam ir nedaudz atšķirīgi testēšanas mērķi. Vienotajā datnē katra šīs daļas būtu jāveido pilnīgi neatkarīgas no citām, lai tās vēlāk varētu vienkārši sadalīt un lietot neatkarīgi. Arī katru atsevišķo testa soli jāspēj izmantot atsevišķi un bez sākotnējās konfigurācijas daļas, lai, piemēram, to izmantotu kādā pilnīgi citā testā, bet tad konfigurācija jāveic attiecīgā testa ietvaros vai manuāli un var nākties pievienot dažas darbības katra soļa konfigurācijā.

5.3. Konfigurācijas daļas uzbūve

Svarīgākā un sarežģītākā daļa skriptā ir tieši konfigurācija. Tai ir iespēju robežās jābūt pilnīgai – visai konfigurācijai jānotiek no skripta, jo process var aizņemt ļoti daudz laika, ja jāveic manuāla konfigurācija. Galvenais mērķis, kā jau minēts iepriekš, ir iekārtu savstarpējie savienojumi. Šī skripta izpildē ir iesaistītas 3 dažādas iekārtas – skaņas vadības pults, kodeks un tīkla mezgls. Skaņas vadības pults un kodeks tiek izmantoti dažāda veida darbību veikšanai, savukārt tīkla mezglu izmanto, lai pārbaudītu skaņas līmeni pēc šīm darbībām.

Iedarbinot skriptu, tam jānorāda parametri – iekārtu adreses tīklā, skaņas kanālu numuri, pults kanāls. Katrai iekārtai konfigurācijas tīmekļa lapā tad balstoties uz šo informāciju ir pareizi jāizveido savienojumi ar pārējām iekārtām. Skripta darbības laikā galvenajā datu plūsmas kanālā caur kodeku tiks sūtītas komandas uz ievadizvades paneli. Šo galveno kanālu var definēt kā izejas skaņas plūsmu. Šis kanāls pultij jāpievieno veidojot skaņas avotus – katram skaņas avotam to norāda kā tā galveno avotu. Tas nodrošina uzticamu, nemainīgu skaņas un datu plūsmu fonā. Kodeka konfigurācijā jānorāda, ka tam komandas jāņem tieši no šī kanāla.

Tad no pults jāizveido savienojums ar tīkla mezglu, vairākiem tā kanāliem priekš vairākiem darbību veidiem, piemēram, priekšskatījums vai saziņa ar studijas mikrofonu. Arī šeit katram darbības veidam jāizveido atsevišķs datu plūsmas izejas kanāls, kuru pievieno attiecīgajai skaņas pults funkcijai un tīkla mezglā pieslēdz vienam skaņas kanālam, uz kura tiek veiktas pārbaudes. Visa šī konfigurācija jāveic caur iekārtu konfigurācijas tīmekļa lapām, izmantojot iepriekš aprakstītās ‘WEB’ komandas.

5.4. Pārbaudes daļas uzbūve

Katrai skaņas avota tipa testu daļai, gan vienotajā datnē, gan sadalītajās, jāiekļauj vismaz 3 darbības – jāizveido skaņas avots, jāveic tā pārbaudes, jāizdzēš skaņas avots. Lai to nodrošinātu, katram skaņas avota tipam skriptā jāizveido vismaz 3 mazāki bloki – pirmais bloks, kurš caur WEB saskarni izveido pārbaudāmo skaņas avotu. Tam seko bloks ar visām scenārijā norādītajām pārbaudēm un pēdējais bloks, kurš izdzēš skaņas avotu – pirmkārt, lai nepiesārņotu iekārtas konfigurāciju, otrkārt, lai atvieglotu skripta modifikāciju, jo skaņas avoti tiek veidoti secīgi un katram tiek piešķirts unikāls identifikators – ja avotu pēc izveides izdzēš, nākamais avots saņems tādu pašu identifikatoru, savukārt avotu atstājot katrs nākamais saņems citu identifikatoru, kas ļoti apgrūtina tā pareizu atrašanu pēc izveides. Šo svarīgi ņemt vērā veidojot skriptu, jo iekārtā bez definētiem skaņas avotiem izveidojot jaunu avotu var paredzami zināt tā identifikatoru, kas atvieglo skripta izveidi.

Svarīgi nodrošināt, lai katra skaņas avota tipa testi būtu neatkarīgi viens no otra, t.i. nedrīkst rasties situācija, kad iepriekšējam testam beidzoties nākošais nedarbojas, jo iekārta nav pareizi iestatīta. Vislabāk to nodrošināt katru testa daļu uzsākot no noteikta, universāli izmantota sākuma stāvokļa un testa daļas beigās atgriezt iekārtas šajā stāvoklī.

Šeit minētajai struktūrai ir jābūt universālai visā skriptā, visiem skaņas avotu tipiem, lai nodrošinātu labu lasāmību, un atvieglotu modifikācijas iespējas, jo pēc šādas vienotas struktūras ir viegli saprast kur sākas un kur beidzas konkrētas scenārija daļas pārbaude.

Pēc vienota principa būtu jāveido arī katra individuālā testa soļa pārbaudes. To struktūra jāveido balstoties uz *Gherkin* testu valodas struktūru – katram scenārija solim ir trīs daļas – *given (dots), when(kad), then(tad).* Netieši tas tulkojās kā – pieņemot, ka izpildās ‘šis’ nosacījums, tad veicot ‘šādu’ darbību, rezultāts ir ‘šāds’. Tā ir ļoti izplatīta prakse testu veidošanā, arī ārpus jau minētās *Gherkin* valodas, jo ievērojami uzlabo lasāmību un struktūru, nodrošinot vienotu stilu viscaur kodam. Struktūras galvenais mērķis ir nodalīt svarīgākās testa daļas redzamā veidā vienu no otras. Pirmā daļa (dots) nodrošina priekšnosacījumu izpildi – iespējami garantē, ka katrā izpildes reizē rezultāts būs balstīts uz vienu un to pašu konfigurāciju. Otrā daļa (kad) izpilda pašu darbību. Pēdējā daļa (tad) pārbauda iekārtu stāvokli pēc darbības izpildes.

Tabulā 5.1. norādīti visi skaņas avotu tipi un to scenārija soļu apjoms, kas tieši norāda arī veicamā darba apjomu.

Scenārija pārskats 5.1.tabula

Skaņas avota tips	**Scenārija soļu apjoms**
Līnijas avots (*Line source)*	17
Operatora mikrofons (*Operator mic)*	11
Studijas/Monitora 2 viesa mikrofons (*Studio/Mon2 guest mic)*	10
Kodeka avots (*Codec source)*	12
CR viesa mikrofons (*CR guest mic)*	10
CR producenta mikrofons (*CR producer mic)*	11
Datora atskaņotājs (*Computer player)*	17
Tālruņa avots (*Phone source)*	16
Ārējais mikrofons (*External mic)*	11
Kanāls bez avota (*Channel without source)*	2

Tādējādi kopā darba izstrādes gaitā jāautomatizē 10 savstarpēji nesaistīti testa scenāriji, katrā no 2 līdz 17 dažādiem soļiem – darbībām un to pārbaudēm. Rezultātā sagaidāms, ka tiek izveidotas kopumā 10 datnes ar skriptiem – viena, kurā visi testi ir apkopoti vienuviet, palaižami kā globāla scenārija pārbaude, un deviņas, katram skaņas avota tipam sava. Tā kā kanāls bez avota satur tikai divus soļus, to var īpaši nenodalīt atsevišķā datnē un tā vietā veikt tā testus, piemēram, katra skripta beigās, lai pārliecinātos, ka ar konkrēto avotu saistītā konfigurācija ir pareizi noņemta.

5.5. Pārbaudes daļas skripta izveide

Šeit redzams dotajā valodā izveidots koda fragments:

```
1. << Channel is ON. GPIO: press and release panel button #2 (OFF)
2. # Given: Channel is ON.
3. S2: SET FaCH#1 ON_State=ON
4. sleep 0.75
5. ? LAST_S1 GPO 1 Llhhh|nGPO 1 lHhhh|nGPO 1 lhhLh|nGPO 1 lhhHh
6. sleep 0.75
7. # When: GPIO: press and release panel button #2 (OFF)
8. S1: GPI 5 hLhhh
9. sleep 0.75
10. # Then: panel #1 OFF, #2 ON, #3 KEEP, #4 NONE, #5 FLASH, Audio OFF
11. ? LAST_S1 GPI 5 hLhhh|nGPO 1 Hhhhh|nGPO 1 hLhhh|nGPO 1 hlhhL|nGPO 1 hlhhH
12. sleep 0.75
13. S3: MTR OCH 2
14. S3: MTR OCH 2
15. sleep 0.75
16. ? PEEK L:-1000 R:-1000
17. sleep 0.75
18. S1: GPI 5 hHhhh
```

19. sleep 0.75
20. ? LAST_S1 GPI 5 hHhhh
21. sleep 0.75

Šis fragments ir neliela daļa no gala skripta un kalpo kā paraugs, uz kuru balstoties būtu jāveido visas pārbaudes. Šī fragmenta scenārija apraksts ir atrodams 1. pielikumā un nodaļā 4.1.2. Prasību interpretācija.

Šajā piemērā var uzskatāmi redzēt, kā iepriekš definētā struktūra izmantota kodā. Vispirms uz ekrāna un žurnālā tiek izvadīts soļa apraksts. To nodrošina 1. rindiņā esošais kods. Šīs rindiņas saturs arī nodrošina vieglu konkrēta soļa atrašanu skriptā, izmantojot jebkāda teksta redaktora meklēšanas funkciju, tādēļ šīs rindiņas saturu nepieciešams nokopēt no scenārija apraksta. Pēc šīs rindiņas seko atbilstoši aprakstam un scenārija solim izveidots kods.

Pirmkārt jāveic priekšnosacījuma definēšana, jeb ‘pieņemot, ka izpildās nosacījums’. Priekšnosacījums ir pareizi jāinterpretē no scenārija apraksta. Tas vienmēr ir tieši norādīts scenārija soļa pirmajā teikumā. Šajā piemērā aprakstā norādīts, ka ‘kanāls ir ieslēgts’. Šis arī ir priekšnosacījums. Pašu priekšnosacījumu norāda kā komentāru ar atslēgas vārdu ‘Given’, kas norāda, ka sekojošais kods veic sākotnējā stāvokļa iestatīšanu. Pēc šī komentāra seko attiecīgās darbības, šajā gadījumā kanāls tiek ieslēgts (3. rindiņa) un tiek pārbaudīts, vai attiecīgā darbība notika (5. rindiņa). Šī pārbaude tiek veikta tieši tāpat, kā darbības rezultāta pārbaude, kura aprakstīta tālāk dokumentā. Pārbaudes mērķis ir arī notīrīt iekārtas atmiņu no pēdējām tās izvadītajām darbībām, jo tās tiek saglabātas un traucē vēlākām pārbaudēm. Vēl tiek izmantotas nelielas aiztures, lai korekti sagaidītu rezultātu. Piemērā šis bloks ir no 2. līdz 6. rindiņai.

Otrkārt notiek darbība izpilde, jeb ‘veicot darbību’. Arī to identificē komentārs, šoreiz ar atslēgas vārdu ‘When’ – tas norāda, ka sekojošais bloks ir darbības izpilde. Tā saturs arī tiek ņemts no scenārija apraksta un ir tā otrais teikums. Šajā gadījumā ‘Uz ievadizvades paneļa nospiest un atlaist otro pogu’. Piemērā šis ir bloks no 7. rindiņas līdz 9. rindiņai. Darbības izpildi nodrošina komanda ‘S1: GPI 5 hLhhh’ 8. rindiņā – tā norāda, ka uz iekārtas, kas skripta konfigurācijas sadaļā definēta kā ‘S1’, šajā piemērā šī iekārta ir kodeks, piekto ievadizvades kanālu jānosūta komanda nospiest pogu Nr. 2. Pēc tam uz brīdi tiek izsaukta izpildes aizture (9. rindiņa), lai sākot nākošā bloka darbību visi rezultāti būtu sagaidīti.

Treškārt tiek pārbaudīti rezultāti, jeb ‘tad sagaidāms rezultāts’. Arī šo daļu identificē komentārs ar atslēgas vārdu ‘Then’, kas norāda, ka sekojošais kods pārbaude iepriekš veiktās darbības rezultāta pārbaudi. Šis ir apjoma ziņā lielākais bloks, no 10. līdz 17. rindiņai. Tā nosaukums ir vispārināts sagaidāmo rezultātu apraksts – tas domāts, lai ātri būtu iespējams saprast

gaidāmo rezultātu, bet papildus tam vēlams izmantot arī pilno scenārija soļa aprakstu, kur ir precīzi aprakstītas sagaidāmās iekārtu darbības. Šī īsā apraksta interpretācijas metode ir definēta katra skripta datnes sākumā, informatīvo komentāru daļā.

11. rindiņā esošā komanda ir ievadizvades paneļa pārbaude – uz iekārtu 'S1', kur iepriekš tika nosūtīta ievades komanda, tiek nosūtīts pieprasījums pēc pēdējām izvadītajām komandām. Vispirms pārbauda, vai pogas nospiešana tika reģistrēta, uz ko norāda 'GPI 5 hLhhh' – iekārtas atbildei šeit būtu jābūt tieši tādai, kā komanda, ko nosūtīja iepriekš. Tad pārbauda izvades lampiņu stāvokļu izmaiņas. Tas notiek secīgi, ar vairākām komandām, jo lampiņu stāvokļa izmaiņas tiek reģistrētas pa vienai. Kā jau iepriekšējā nodaļā minēts, lielākoties tas notiek sākot ar pirmo un beidzot ar piekto lampiņu, bet ir iespējami izņēmumi, tādēļ var nākties tās sakārtot citā secībā pēc skripta testēšanas.

Šajā piemērā sagaidāmais rezultāts ir šāds – 1. lampiņa izslēdzas, 2. lampiņa ieslēdzas, 3. lampiņa saglabā savu stāvokli, 4. lampiņa neveic nekādu darbību un 5. lampiņa ieslēdzas uz 100 milisekundēm, kā arī skaņa tiek izslēgta. Pēc šiem nosacījumiem skriptā tiek veiktas pārbaudes uz konfigurācijā definēto ievadizvades paneļa kanālu, šajā gadījumā tas ir pirmais kanāls. Ieslēgšanos uz 100 milisekundēm interpretatora ierobežojumu dēļ nav iespējams pārbaudīt precīzi, tādēļ pēc aiztures tiek pārbaudīts, vai atbilstošā lampiņa ieslēdzās un izslēdzās.

Skaņas pārbaudi veic uz citas, iepriekš konfigurācijā pievienotas iekārtas. Piemērā tā definēta kā 'S3'. Šai iekārtai tiek nosūtīta komanda (13. un 14. rindiņa), kas liek tai izvadīt pašreizējo skaņas līmeni otrajā izvades kanālā. Komandu izsauc divreiz, jo konkrētajā situācijā izmantotais iekārtas veids mēdz reģistrēt komandas izsaukumu ar aizkavēšanos, lielākoties tīkla un pašas iekārtas slodzes dēļ. Divi izsaukumi nodrošina paredzamākus rezultātus. Pēc vēl vienas nelielas aiztures tiek pārbaudīts iekārtas izvadītais rezultāts (16. rindiņa), tas norādīts decibelos kreisajā un labajā kanālā. To nodrošina -1000 šajā gadījumā izprotams kā -100dB, jeb skaņas nav. Ja sākotnējā konfigurācija iestatīta pareizi, ieslēgtu kanālu gadījumā šis līmenis būtu aptuveni -20dB, jeb -200 kodā.

Kā pēdējais solis, bez atsevišķas sadaļas (no 18. līdz 21. rindiņai), ir pogas atlaišana, kam arī veic ātru pārbaudi, lai komanda nepaliktu atmiņā un netraucētu nākošā soļa izpildi. Pēc scenārija apraksta pogas atlaišanai būtu jānotiek uzreiz pēc nospiešanas, kas norādīts arī otrajā sadaļā, tomēr tas apgrūtina standartizētu un uzticamu rezultātu iegūšanu, jo jebkādi nelieli traucējumi tīklā var gadījuma kārtā šīs komandas ievades reģistrēšanas notikumu izvadīt neparedzamā vietā pa vidu starp izvades rezultātiem. Tā kā šāda tipa pārbaudēs pogas stāvoklis neietekmē tiešo rezultātu un

tādējādi nemaina rezultāta būtību, tad nospiešana pēc visu pārbaužu veikšanas ir pietiekami laba pieeja. Teorētiski ir iespējams pogas atlaišanas darbību veikt reizē ar nākamo pogas nospiešanas darbību, kā tas sākotnēji tika darīts, bet šāda pieeja ļoti apgrūtina skaņas pārbaudes.

Pēc šāda principa jāveic visa scenārija automatizācijas skripta izveide.

6. TESTĒŠANA

Programmatūras testēšana tiek veikta manuāli, salīdzinot tās sniegtos rezultātus ar testa scenārijā aprakstīto darbību. Jebkādas nesakritības vai nepareizi rezultāti tad tiek analizēti un novērtēti, lai saprastu vai kļūda ir funkcionalitātē vai programmatūrā un tiek veiktas attiecīgās darbības, lai šo kļūdu novērstu. Šis ir samērā laikietilpīgs process, jo scenārija darbība ir samērā ilga ar daudz pārbaudēm, tādēļ analīze jāveic ļoti rūpīgi, lai nepieļautu kļūdainu rezultātu parādīšanos reāla testa laikā. Lai uzlabotu darba efektivitāti, analīze un labojumi notiek paralēli skripta darbībai, t.i. tiek palaists skripts, atrastas svarīgākās kļūdas un tad tās tiek labotas, tikmēr paralēli palaižot iepriekšējās darbības laikā izveidotās izmaiņas. Tas nodrošina pastāvīgu darba plūsmu, nodrošinot lielāku darba produktivitāti un efektīvāku ierobežotā laika izmantošanu.

Tabulā 6.1. redzami daži izstrādes procesa beigu daļā veiktie skripta pilnas izpildes rezultāti. Redzams kopā veikto pārbaužu skaits, ieskaitot pārbaudes, kas tiek veiktas priekšnosacījumu uzstādīšanai, un kopējais pareizo un nepareizo rezultātu skaits – visi nepareizie rezultāti tiek analizēti, noteikts cēlonis (iekārtas kļūda, specifikācijas kļūda vai skripta kļūda) un veikti attiecīgi labojumi. Pirms katras atkārtotas izpildes tiek salabota tikai daļa no kļūdām, tādēļ nepareizo rezultātu skaits samazinās pakāpeniski. Redzams arī tas, ka pievienojot jaunas pārbaudes vai pārveidojot esošās skaits attiecīgi mainās.

Pēdējos rezultātos absolūtais vairākums nepareizo rezultātu ir saistīti ar iekārtu savstarpējās konfigurācijas problēmām priekš skaņas pārbaudēm.

Testēšanas rezultāti 6.1. tabula

Datums	Pārbaužu skaits	Pareizi rezultāti	Nepareizi rezultāti
11.05.2017 18:16	224	157	67
11.05.2017 19:02	228	181	47
13.05.2017 12:25	253	192	61
13.05.2017 13:24	254	198	56
13.05.2017 15:04	255	212	43
18.05.2017 18:53	255	230	25
23.05.2017 18:01	255	239	16

Savukārt tabulā 6.2. redzams gala programmatūras testu rezultātu sadalījums. Šie rezultāti iegūti no atsevišķajās datnēs esošajiem skriptiem. Nepareizie rezultāti ir dažu veidu skaņas

pārbaudes, kuru konfigurācija izrādījās ļoti sarežģīta no darba loģikas puses, tādēļ neizdevās tās automatizēt.

Skaņas avotu tipu gala testēšanas rezultāti 6.2. tabula

Skaņas avota tips	Pārbaužu skaits	Pareizi rezultāti	Nepareizi rezultāti
Līnijas avots	45	45	0
Operatora mikrofons	30	28	2
Studijas mikrofons	28	28	0
Kodeka avots	31	30	1
CR Viesa mikrofons	26	26	0
CR Producenta mikrofons	30	28	2
Datora atskaņotājs	37	35	2
Tālrunis	29	29	0
Ārējais mikrofons	26	26	0

7. PROGRAMMATŪRAS PROJEKTA DARBA ORGANIZĀCIJA

Izstrādes darbs tiek organizēts patstāvīgi – sākotnēji bija zināms konkrēts testa scenārijs, tā ietvaros izpildāmās darbības un sagaidāmais rezultāts. Šie scenāriji tika pārrakstīti kodā, kurš ar interpretatora palīdzību tiek pārveidots konkrētās darbībās uz testējamās iekārtas un pārbaudīta tās darbība. Darba izstrādes gaitā regulāri tika pārrunāti dažādi ar izstrādes valodu un tās izmantošanu saistīti jautājumi ar izstrādes vietā esošiem valodu pārzinošiem cilvēkiem.

Koda izstrāde pārsvarā tika veikta patstāvīgi ārpus izstrādes vietas pāris reizes nedēļā ierodoties izstrādes vietā, lai pārbaudītu progresu un veiktu nepieciešamas korekcijas, kā arī veiktu izmaiņu un jaunu testa soļu pievienošanu konfigurācijas pārvaldības sistēmā *Git*.

8. KVALITĀTES NODROŠINĀŠANA

Programmatūra ir viegli modificējama nepieciešamības gadījumā – pirmkods ir komentēts un strukturēts balstoties uz testa scenārija aprakstu, un ja scenārijā tiek veiktas izmaiņas, pirmkodā ar teksta redaktora meklēšanas funkciju ir iespējams viegli atrast nepieciešamo pārbaudes soli un to izmainīt balstoties uz jaunajām prasībām.

Izstrādājot programmatūru tika ņemti vērā ieteikumi no izstrādes vietā strādājošiem zinošiem programmētājiem un testētājiem, kā arī tika ievēroti vispārpieņemti principi attiecībā uz koda strukturēšanu, lasāmības un rediģējamības nodrošināšanu.

Valodas iespēju robežās nodrošināta koda dalāmība, lai tā atsevišķas daļas būtu izmantojamas vairākkārt arī citos testa scenārijos. Kods ir strukturēts tā, lai tas būtu pārskatāms – katrs testa solis ir atdalīts kā koda bloks un satur komentāru ar soļa aprakstu no specifikācijas. Katra skaņas avota tipa soļu bloks arī ir atdalīts ar komentāriem, lai būtu viegli pēc vajadzības kādu koda fragmentu izņemt no koda vai to izmantot kādā citā testa scenārijā, ar nosacījumu, ka tiek nodrošināta nepieciešamā sāknēšanas informācija – iekārtas adrese, kanāla numurs, kā arī daži parametri interpretatora korektai ieslēgšanai.

9. KONFIGURĀCIJAS PĀRVALDĪBA

Konfigurācijas pārvaldība tiek veikta Git versiju kontroles sistēmā. Jaunas versijas pievienošana parasti notiek 2-3 reizes nedēļa, ierodoties darba izstrādes vietā. Tad patstāvīgi veiktās izmaiņas tiek pārbaudītas un pievienotas versiju kontrolē, pievienojot īsu, kodolīgu komentāru, par attiecīgās izmaiņas saturu. Piekļuve sistēmai tiek veikta caur īpašu termināļa programmatūru, izmantojot tajā pieejamās komandas.

Kopumā izstrādes laikā jauna versija Git serverī tika ievietota ap 10-15 reizēm, parasti pēc lielāka apjoma izmaiņu veikšanas, piemēram, jauna skaņas avota tipa testu bloks, koda pārstrukturēšana vai lasāmības uzlabošana. Darba beigās tika veikta arī koda sadala pa vairākām datnēm, tās visas arī tikai pievienotas Git serverī.

10. DARBIETILPĪBAS NOVĒRTĒJUMS

Novērtējot programmatūras prasības, izstrādes procesa darbietilpības apjoms tiek novērtēts ar 12 personnedēļām jeb 3 personmēnešiem. Apjoms sadalās šādi:

- 1 personnedēļa darījumloģikas izpratnei un analīzei

- 1 personnedēļa programmatūras projektējuma izstrādei,

- 2 personmēneši jeb 8 personnedēļas programmatūras izstrādei,

- 1 personnedēļas programmatūras testēšanai (neiekļaujot kļūdu labošanu),

- 1 personnedēļa programmatūras kvalitātes uzlabošanai

Darbs tiek izstrādāts bez jebkādas iepriekšējas pieredzes testu automatizācijā un konkrētās valodas izmantošanā. Šie ir apgrūtinoši faktori, kuri ņemti vērā veicot darbietilpības aprēķinu. Ir iespējams, ka šāda apjoma programmatūras apjoms nozarē pieredzējušam programmētājam būtu novērtējams ar mazāku darba apjomu.

11. SECINĀJUMI

Ir izveidota programmatūra, kas atvieglo testēšanas procesu uzņēmumā.

Darba izstrāde bija ļoti vērtīga pieredze testu automatizācijas procesā. Tā rezultātā tika apgūta gan testu automatizācijas pamatus, gan specifiskas zināšanas aparatūras testēšanā.

Lielākais ieguvums ir pamatprasmes šajā nozarē, kas jau šobrīd noder arī patstāvīgajā darbavietā, kur ir iespējams iesaistīties testu automatizācijā un palīdzēt problēmu risināšanā šajā jautājumā.

Izstrādes laikā tika pilnveidota prasme organizēt darbu, jo paralēli bija jāapvieno patstāvīgais darbs ar šī darba izstrādi, kas izdevās ļoti veiksmīgi.

PIELIKUMS

1. PIELIKUMS
Testa scenārija apraksts

Remote Control

Source remote control via GPIO – operations and configuration:
-functions: availability by source types; commands to console {ON, OFF, Mute, Talk to CR}; indications from console {ON lamp, OFF lamp, Muted lamp, Talk to CR active lamp}
-aspects: source profile settings – editing, saving, loading; console operations - receiving and sending commands
-check reaction: UI displays

Line source GPIO

-Console: Take the source on a fader
-->Panel lamp #2 (Off) is lit
-->Panel lamps #1,3,4,5 are off
-Channel is OFF. Console: turn the channel ON
-->Panel lamp #4 (Start) flashes for 100 ms
-->Panel lamp #5 (Stop) stays off
-->Panel lamp #3 (Preview) follows the preview state
-->Panel lamp #2 (Off) goes out
-->Panel lamp #1 (On) lights up
-Channel is ON. GPIO: press and release panel button #1 (ON)
-->The channel is already ON, no operation
-->All panel lamps #1-5 keep their states, no flashes
-Channel is ON. Console: press and release the channel ON button
-->The channel is already ON, no operation
-->All panel lamps #1-5 keep their states, no flashes
-Channel is OFF. GPIO: press and release panel button #1 (ON)
-->Console turns the channel ON: audio, indications
-->Panel lamp #5 (Stop) stays off
-->Panel lamp #4 (Start) flashes for 100 ms
-->Panel lamp #3 (Preview) follows the preview state
-->Panel lamp #1 (On) lights up
-->Panel lamp #2 (Off) goes out
-Channel is ON. GPIO: press and release panel button #2 (OFF)
-->Console turns the channel OFF: audio, indications
-->Panel lamp #2 (Off) lights up
-->Panel lamp #5 (Stop) flashes for 100 ms
-->Panel lamp #4 (Start) stays off
-->Panel lamp #3 (Preview) follows the preview state
-->Panel lamp #1 (On) goes out
-Channel preview is OFF. Console: turn preview on
-->Panel lamp #3 (Preview) lights up
-->Panel lamps #4,5 (Start, Stop) stay off
-->Panel lamps #1,2 (On, Off) follow the channel On/Off state
-Channel is ON. Console: turn the channel OFF
-->Panel lamp #2 (Off) lights up
-->Panel lamp #5 (Stop) flashes for 100 ms
-->Panel lamp #4 (Start) stays off
-->Panel lamp #3 (Preview) follows the preview state

-->Panel lamp #1 (On) goes out
-Channel preview is OFF. GPIO: press panel button #3 (Preview)
-->Console turns channel preview ON: audio, indications
-->Panel lamp #3 (Preview) lights up
-->Panel lamps #4,5 (Start, Stop) stay off
-->Panel lamps #1,2 (On, Off) follow the channel On/Off state
-Channel preview is ON. Console: turn preview off
-->Panel lamp #3 (Preview) goes out
-->Panel lamps #4,5 (Start, Stop) stay off
-->Panel lamps #1,2 (On, Off) follow the channel On/Off state
-GPIO panel button #3 (Preview) is depressed, and channel preview is ON. GPIO: release panel button #3 (Preview)
-->Panel lamp #3 (Preview) goes out
-->Console turns channel preview OFF: audio, indications
-->Panel lamps #4,5 (Start, Stop) stay off
-->Panel lamps #1,2 (On, Off) follow the channel On/Off state
-Channel is ON. GPIO: press and release panel button #4 (Reset)
-->Console turns the channel OFF: audio, indications
-->Panel lamp #2 (Off) lights up
-->Panel lamp #4 (Start) stays off
-->Panel lamp #5 (Stop) stays off, and does NOT flash
-->Panel lamp #3 (Preview) follows the preview state
-->Panel lamp #1 (On) goes out
-GPIO ready only: Channel is OFF, no source on the channel. While keeping panel button #5 (READY) released, take a source
-->After loading the source: All panel lamps, including #2 (Off) are off
-->After loading the source: Both the channel ON and OFF lamps on surface are off
-GPIO ready only: Channel is OFF. GPIO: press and release panel button #5 (READY)
-->For the duration of the panel button press: The OFF lamp on surface lights up
-->For the duration of the panel button press: Panel lamp #2 (Off) lights up
-->For the duration of the panel button press: After releasing the panel button the both lamps go out.
-GPIO ready only: Channel is ON, and the GPIO panel button #5 (READY) is released. Console: turn the channel OFF
-->The channel ON lamp on surface goes out
-->The OFF lamp on surface stays off
-->Panel lamp #2 (Off) stays off
-GPIO ready only: Channel is ON, and the GPIO panel button #5 (READY) is pressed. Console: turn the channel OFF
-->The channel ON lamp on surface goes out
-->The OFF lamp on surface lights up
-->Panel lamp #2 (Off) lights up
-GPIO ready only: Channel is OFF, and the GPIO panel button #5 (READY) is depressed. Console: turn the channel ON
-->The OFF lamp on surface goes out
-->Panel lamp #2 (Off) goes out
-->The channel ON lamp on surface lights up

Operator mic GPIO

-Console: Take the source on a fader

-->The console must reset the GPIO panel lamps: - Panel lamps #1,3,4,5 are off
-Channel is OFF. Console: turn the channel ON
-->- Panel lamp #1 (ON) lights up
-->- Panel lamp #2 (OFF) goes out
-->- Panel lamps #3,4,5 keep their previous states
-Channel is OFF. GPIO: press and release panel button #1 (ON)
-->Console turns the channel ON: audio, indications
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #2 (Off) goes out
-->Panel lamp #1 (On) lights up
-GPIO panel button #3 (TALK TO MON2) is depressed. GPIO: release panel button #3
-->Console deactivates operator mic talking to Mon2: audio, indications
-->Panel lamp #3 (TALK to Mon2) goes out
-->Panel lamps #1,2,4,5 keep their previous states
-Channel is ON. Console: turn the channel OFF
-->Panel lamp #2 (Off) lights up
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #1 (On) goes out
-Channel is ON. GPIO: press and release panel button #2 (OFF)
-->Panel lamps #3,4,5 keep their previous states
-->Console turns the channel OFF: audio, indications
-->Panel lamp #2 (Off) lights up
-->Panel lamp #1 (On) goes out
-GPIO: press panel button #4 (MUTE)
-->Console mutes the channel input: audio, indications
-->Console ON lamp goes out
-->Panel lamp #4 (MUTE) lights up
-->Panel lamp #1 (On) goes out
-->Panel lamps #2,3,5 keep their previous states
-GPIO: press panel button #3 (TALK TO MON2)
-->Console activates operator mic talking to Mon2: audio, indications
-->Panel lamp #3 (TALK to Mon2) lights up
-->Panel lamps #1,2,4,5 keep their previous states
-GPIO panel button #4 (MUTE) is depressed. GPIO: release panel button #4
-->Console unmutes the channel input: audio, indications
-->Console ON lamp lights up
-->Panel lamp #4 (MUTE) goes out
-->Panel lamp #3 (MUTE) lights up
-->Panel lamps #2,3,5 keep their previous states
-GPIO: press panel button #5 (Talk to Previewed sources)
-->Console activates operator mic talking to sources in talkback state: audio, indications
-->Panel lamps #1,2,3 keep their previous states
-GPIO panel button #5 (Talk to Previewed sources) is depressed. GPIO: release panel button #5
-->Console deactivates operator mic talking to sources in talkback state: audio, indications
-->Panel lamp #5 (TALK to sources) goes out
-->Panel lamps #4,5 goes out, lamps #1,2,3 keep their previous states

Studio/ Mon2 guest mic GPIO

-Channel is OFF. Console: turn the channel ON
-->Panel lamp #1 (ON) lights up

-->Panel lamp #2 (OFF) goes out
-->Panel lamps #3,4,5 keep their previous states
-Channel is OFF. GPIO: press and release panel button #1 (ON)
-->Console turns the channel ON: audio, indications
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #2 (Off) goes out
-->Panel lamp #1 (On) lights up
-Console: Take the source on a fader
-->Panel lamp #2 (Off) is lit
-->Panel lamps #1,3,4,5 are off
-Channel is ON. Console: turn the channel OFF
-->Panel lamp #2 (Off) lights up
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #1 (On) goes out
-Channel is ON. GPIO: press and release panel button #2 (OFF)
-->Console turns the channel OFF: audio, indications
-->Panel lamp #2 (Off) lights up
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #1 (On) goes out
-GPIO: press panel button #3 (TALK to CR)
-->Console activates talk to CR: audio, indications
-->Panel lamp #3 (TALK to CR) lights up
-->Panel lamps #1,2,5 keep their previous states,lamp #4 lights up
-GPIO panel button #3 (TALK to CR) is depressed. GPIO: release panel button #3
-->Console deactivates talk to CR: audio, indications
-->Panel lamp #3 (TALK to CR) goes out
-->Panel lamps #1,2,5 keep their previous states, lamp #4 goes out
-GPIO: press panel button #4 (MUTE)
-->Console mutes the channel input: audio, indications
-->Panel lamp #4 (MUTE) lights up
-->Console ON lamp goes out
-->Panel lamp #1 (On) goes out
-->Panel lamps #2,3,5 keep their previous states
-GPIO panel button #4 (MUTE) is depressed. GPIO: release panel button #4
-->Console ON lamp lights up
-->Console unmutes the channel input: audio, indications
-->Panel lamp #4 (MUTE) goes out
-->Panel lamp #1 (ON) lights up
-->Panel lamps #2,3,5 keep their previous states
-GPIO: press and release panel button #5 (NOT CONNECTED)
-->No effect on audio,indications, panel lamps,...

Codec source GPIO

-Channel is OFF. Console: turn the channel ON
-->Panel lamp #1 (ON) lights up
-->Panel lamp #2 (OFF) goes out
-->Panel lamps #3,4,5 keep their previous states
-Channel is OFF. GPIO: press and release panel button #1 (ON)
-->Console turns the channel ON: audio, indications
-->Panel lamp #1 (On) lights up

-->Panel lamp #2 (Off) goes out
-->Panel lamps #3,4,5 keep their previous states
-Console: Take the source on a fader
-->The console must reset the GPIO panel lamps: Panel lamp #2 (Off) is lit
-->The console must reset the GPIO panel lamps: Panel lamps #1,3,4,5 are off
-Channel is ON. Console: turn the channel OFF
-->Panel lamp #2 (Off) lights up
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #1 (On) goes out
-GPIO panel button #3 (TALK to CR) is depressed. GPIO: release panel button #3
-->Console deactivates talk to CR: audio, indications
-->Panel lamp #3 (TALK to CR) goes out
-->Panel lamps #1,2,5 keep their previous states, lamp #4 goes out
-GPIO: press panel button #4 (MUTE)
-->Console mutes the channel input: audio, indications
-->Panel lamp #4 (MUTE) lights up
-->Panel lamps #1,2,3,5 keep their previous states
-Channel is ON. GPIO: press and release panel button #2 (OFF)
-->Console turns the channel OFF: audio, indications
-->Panel lamp #1 (On) goes out
-->Panel lamp #2 (Off) lights up
-->Panel lamps #3,4,5 keep their previous states
-GPIO: press and hold panel button #3 (Talk to CR)
-->Console activates talk to CR: audio, indications
-->Panel lamp #3 (TALK to CR) lights up
-->Panel lamps #1,2,5 keep their previous states, lamp #4 lights up
-GPIO panel button #4 (MUTE) is depressed. GPIO: release panel button #4
-->Console unmutes the channel input: audio, indications
-->Panel lamp #4 (MUTE) goes out
-->Panel lamps #1,2,3,5 keep their previous states
-GPIO: press panel button #5 (Talk to source)
-->Panel lamp #5 (TALK to source) lights up
-->Console activates talk to source: audio, indications
-->Panel lamps #1,2,3,4 keep their previous states
-Console: release the channel Talkback button
-->Panel lamp #5 (TALK to source) goes out
-->Panel lamps #1,2,3,4 keep their previous states
-GPIO: release panel button #5 (Talk to source)
-->Console deactivates talk to source: audio, indications
-->Panel lamp #5 (TALK to source) goes out
-->Panel lamps #1,2,3,4 keep their previous states

CR guest mic GPIO

-Channel is OFF. GPIO: press and release panel button #1 (ON)
-->Console turns the channel ON: audio, indications
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #2 (Off) goes out
-->Panel lamp #1 (On) lights up
-Console: Take the source on a fader
-->Panel lamp #2 (Off) is lit

-->Panel lamps #1,3,4,5 are off
-Channel is ON. GPIO: press and release panel button #2 (OFF)
-->Console turns the channel OFF: audio, indications
-->Panel lamp #2 (Off) lights up
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #1 (On) goes out
-Channel is OFF. Console: turn the channel ON
-->Panel lamp #1 (ON) lights up
-->Panel lamp #2 (OFF) goes out
-->Panel lamps #3,4,5 keep their previous states
-Channel is ON. Console: turn the channel OFF
-->Panel lamp #2 (Off) lights up
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #1 (On) goes out
-GPIO panel button #3 (TALK to CR) is depressed. GPIO: release panel button #3
-->Console deactivates talk to CR: audio, indications
-->Panel lamp #3 (TALK to CR) goes out
-->Panel lamps #1,2,5 keep their previous states, lamp #4 goes out
-GPIO: press panel button #4 (MUTE)
-->Console mutes the channel input: audio, indications
-->Console ON lamp goes out
-->Panel lamp #4 (MUTE) lights up
-->Panel lamp #1 (On) goes out
-->Panel lamps #2,3,5 keep their previous states
-GPIO: press panel button #3 (TALK to CR)
-->Console activates talk to CR: audio, indications
-->Panel lamp #3 (TALK to CR) lights up
-->Panel lamps #1,2,5 keep their previous states, lamp #4 lights up
-GPIO panel button #4 (MUTE) is depressed. GPIO: release panel button #4
-->Console unmutes the channel input: audio, indications
-->Console ON lamp lights up
-->Panel lamp #4 (MUTE) goes out
-->Panel lamp #1 (ON) lights up
-->Panel lamps #2,3,5 keep their previous states
-GPIO: press and release panel button #5 (NOT CONNECTED)
-->No effect on audio,indications, panel lamps,...

CR producer mic GPIO

-Console: Take the source on a fader
-->Panel lamp #2 (Off) is lit
-->Panel lamps #1,3,4,5 are off
-Channel is OFF. Console: turn the channel ON
-->Panel lamp #1 (ON) lights up
-->Panel lamp #2 (OFF) goes out
-->Panel lamps #3,4,5 keep their previous states
-Channel is OFF. GPIO: press and release panel button #1 (ON)
-->Console turns the channel ON: audio, indications
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #1 (On) lights up
-->Panel lamp #2 (Off) goes out

-Channel is ON. Console: turn the channel OFF
-->Panel lamp #2 (Off) lights up
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #1 (On) goes out
-Channel is ON. GPIO: press and release panel button #2 (OFF)
-->Console turns the channel OFF: audio, indications
-->Panel lamp #2 (Off) lights up
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #1 (On) goes out
-GPIO: press panel button #3 (Talk to Monitor2)
-->Console activates producer mic talking to Mon2: audio, indications
-->Panel lamp #3 (TALK to Mon2) lights up
-->Panel lamps #1,2,5 keep their previous states, lamp #4 lights up
-GPIO: press panel button #4 (MUTE)
-->Console mutes the channel input: audio, indications
-->Console ON lamp goes out
-->Panel lamp #1 (On) goes out
-->Panel lamps #2,3,5 keep their previous states
-->Panel lamp #4 (MUTE) lights up
-GPIO panel button #3 (Talk to Monitor2) is depressed. GPIO: release panel button #3
-->Console deactivates producer mic talking to Mon2: audio, indications
-->Panel lamp #3 (TALK to Mon2) goes out
-->Panel lamps #1,2,5 keep their previous states, lamp #4 goes out
-GPIO panel button #4 (MUTE) is depressed. GPIO: release panel button #4
-->Console unmutes the channel input: audio, indications
-->Console ON lamp lights up
-->Panel lamp #4 (MUTE) goes out
-->Panel lamp #1 (On) lights up
-->Panel lamps #2,3,5 keep their previous states
-GPIO: press panel button #5 (Talk to Previewed sources)
-->Console activates producer mic talking to sources in talkback state: audio, indications
-->Panel lamp #5 (TALK to sources) lights up
-->Panel lamps #1,2,3 keep their previous states, lamp #4 lights up
-GPIO panel button #5 (Talk to Previewed sources) is depressed. GPIO: release panel button #5
-->Console deactivates producer mic talking to sources in talkback state: audio, indications
-->Panel lamp #5 (TALK to sources) goes out
-->Panel lamps #1,2,3 keep their previous states, lamp #4 goes out

Computer player GPIO

-Console: Take the source on a fader
-->The console must reset the GPIO panel lamps: Panel lamp #2 (Off) is lit
-->Panel lamps #1,3,4,5 are off
-Channel is OFF. Console: turn the channel ON
-->Panel lamp #4 (Start) flashes for 100 ms
-->Panel lamps #1,5 (Next, Stop) stay off
-->Panel lamp #3 (Preview) follows the preview state
-->Panel lamp #2 (Off) goes out
-Channel is OFF. GPIO: press and release panel button #1 (ON)
-->Console turns the channel ON: audio, indications
-->Panel lamp #4 (Start) flashes for 100 ms

-->Panel lamps #1,5 (Next, Stop) stay off
-->Panel lamp #3 (Preview) follows the preview state
-->Panel lamp #2 (Off) goes out
-Channel is ON. Console: press and release the channel ON button
-->Channel stays turned ON: audio, indications
-->Panel lamps #2,4,5 (Off, Start, Stop) stay off
-->Panel lamp #1 (Next) flashes for 100 ms
-->Panel lamp #3 (Preview) follows the preview state
-Channel is ON. GPIO: press and release panel button #1 (ON)
-->Channel stays turned ON: audio, indications
-->Panel lamp #1 (Next) flashes for 100 ms
-->Panel lamps #2,4,5 (Off, Start, Stop) stay off
-->Panel lamp #3 (Preview) follows the preview state
-Channel is ON. Console: turn the channel OFF
-->Panel lamp #2 (Off) lights up
-->Panel lamp #5 (Stop) flashes for 100 ms
-->Panel lamp #1,4 (Next, Start) stay off
-->Panel lamp #3 (Preview) follows the preview state
-Channel is ON. GPIO: press and release panel button #2 (OFF)
-->Console turns the channel OFF: audio, indications
-->Panel lamp #2 (Off) lights up
-->Panel lamp #5 (Stop) flashes for 100 ms
-->Panel lamps #1,4 (Next, Start) stay off
-->Panel lamp #3 (Preview) follows the preview state
-Channel preview is ON. Console: turn preview off
-->Panel lamp #3 (Preview) goes out
-->Panel lamps #1,4,5 (Next, Start, Stop) stay off
-->Panel lamp #2 (Off) follows the channel On/Off state
-Channel preview is OFF. Console: turn preview on
-->Panel lamp #3 (Preview) lights up
-->Panel lamps #1,4,5 (Next, Start, Stop) stay off
-->Panel lamp #2 (Off) follows the channel On/Off state
-Channel is OFF, no source on the channel. While keeping panel button #5 (READY) released, take a source
-->After loading the source: Both the channel ON and OFF lamps on surface are off
-->Panel lamp #2 (Off) is off
-Channel preview is OFF. GPIO: press panel button #3 (Preview)
-->Console turns channel preview ON: audio, indications
-->Panel lamp #3 (Preview) lights up
-->Panel lamps #1,4,5 (Next, Start, Stop) stay off
-->Panel lamp #2 (Off) follows the channel On/Off state
-Channel is ON. GPIO: press and release panel button #4 (Not used)
-->The console must ignore this GPIO input: no changes in audio signal handling
-->no changes on surface lamps or displays
-->no changes on GPIO panel lamps
-Channel is OFF. GPIO: press and release panel button #5 (READY)
-->For the duration of the panel button press: The OFF lamp on surface lights up
-->Panel lamp #2 (Off) lights up
-->After releasing the panel button the both lamps go out.

-GPIO panel button #3 (Preview) is depressed, and channel preview is ON. GPIO: release panel button #3 (Preview)
-->Console turns channel preview OFF: audio, indications
-->Panel lamp #3 (Preview) goes out
-->Panel lamps #1,4,5 (Next, Start, Stop) stay off
-->Panel lamp #2 (Off) follows the channel On/Off state
-Channel is ON, and the GPIO panel button #5 (READY) is depressed. Console: turn the channel OFF
-->The channel ON lamp on surface goes out
-->The OFF lamp on surface stays off
-->Panel lamp #2 (Off) stays off
-Channel is ON, and the GPIO panel button #5 (READY) is released. Console: turn the channel OFF
-->The channel ON lamp on surface goes out
-->The OFF lamp on surface lights up
-->Panel lamp #2 (Off) lights up
-Channel is OFF, and the GPIO panel button #5 (READY) is released Console: turn the channel ON
-->The OFF lamp on surface goes out
-->Panel lamp #2 (Off) goes out
-->The channel ON lamp on surface lights up

Phone source GPIO

-Console: Take the source on a fader
-->Panel lamps #1,3,4,5 are off
-->The console must reset the GPIO panel lamps: Panel lamp #2 (Off) is lit
-Channel is OFF. Console: turn the channel ON
-->Panel lamp #1 (On) lights up
-->Panel lamp #4 (Start) flashes for 100 ms
-->Panel lamp #5 (Stop) stays off
-->Panel lamp #3 (Preview) follows the preview state
-->Panel lamp #2 (Off) goes out
-Channel is OFF. GPIO: press and release panel button #1 (ON)
-->Panel lamp #5 (Stop) stays off
-->Console turns the channel ON: audio, indications
-->Panel lamp #4 (Start) flashes for 100 ms
-->Panel lamp #3 (Preview) follows the preview state
-->Panel lamp #1 (On) lights up
-->Panel lamp #2 (Off) goes out
-Channel is ON. Console: turn the channel OFF
-->Panel lamp #2 (Off) lights up
-->Panel lamp #4 (Start) stays off
-->Panel lamp #5 (Stop) flashes for 100 ms
-->Panel lamp #3 (PREVIEW) follows the preview state
-->Panel lamp #1 (On) goes out
-Channel is ON. GPIO: press and release panel button #2 (OFF)
-->Console turns the channel OFF: audio, indications
-->Panel lamp #2 (Off) lights up
-->Panel lamp #5 (Stop) flashes for 100 ms
-->Panel lamp #4 (Start) stays off

-->Panel lamp #3 (PREVIEW) follows the preview state
-->Panel lamp #1 (On) goes out
-Channel preview is ON. Console: turn preview off
-->Panel lamp #3 (Preview) goes out
-->Panel lamp #5 (Stop) flashes for 100 ms
-->Panel lamp #4 (Start) stays off
-->Panel lamps #1,2 (On, Off) follow the channel On/Off state
-Channel preview is OFF. GPIO: press panel button #3 (Preview)
-->Console turns channel preview ON: audio, indications
-->Panel lamp #3 (Preview) lights up
-->Panel lamp #4 (Start) flashes for 100 ms
-->Panel lamp #5 (Stop) stay off
-->Panel lamps #1,2 (On, Off) follow the channel On/Off state
-Channel preview is OFF. Console: turn preview on
-->Panel lamp #3 (Preview) lights up
-->Panel lamp #4 (Start) flashes for 100 ms
-->Panel lamp #5 (Stop) stay off
-->Panel lamps #1,2 (On, Off) follow the channel On/Off state
-GPIO panel button #3 (Preview) is depressed, channel preview is ON, and channel is OFF. GPIO: release panel button #3 (Preview)
-->Console turns channel preview OFF: audio, indications
-->Panel lamp #3 (Preview) goes out
-->Panel lamp #5 (Stop) flashes for 100 ms
-->Panel lamps #1 (ON), #4 (Start) stays off
-->Panel lamps #2 (OFF) stays on
-GPIO panel button #3 (Preview) is depressed, channel preview is ON, and channel is ON. GPIO: release panel button #3 (Preview)
-->Console turns channel preview OFF: audio, indications
-->Panel lamp #3 (Preview) goes out
-->Panel lamps #2 (OFF), #4 (Start), #5 (Stop) stays off
-->Panel lamps #1 (ON) stays on
-Channel is ON. GPIO: press and release panel button #4 (Reset)
-->Console turns the channel OFF: audio, indications
-->Panel lamp #2 (Off) lights up
-->Panel lamp #4 (Start) stays off
-->Panel lamp #5 (Stop) stays off, and does NOT flash
-->Panel lamp #3 (Preview) follows the preview state
-->Panel lamp #1 (On) goes out
-Channel is OFF, no source on the channel. While keeping panel button #5 (READY) released, take a source
-->After loading the source: Both the channel ON and OFF lamps on surface are off
-->After loading the source: All panel lamps, including #2 (Off) are off
-Channel is ON, and the GPIO panel button #5 (READY) is released. Console: turn the channel OFF
-->The channel ON lamp on surface goes out
-->The OFF lamp on surface lights up
-->Panel lamp #2 (Off) lights up
-Channel is OFF. GPIO: press and release panel button #5 (READY)
-->For the duration of the panel button press: The OFF lamp on surface lights up

-->Panel lamp #2 (Off) lights up
-->After releasing the panel button the both lamps go out.
-Channel is ON, and the GPIO panel button #5 (READY) is depressed. Console: turn the channel OFF
-->The channel ON lamp on surface goes out
-->The OFF lamp on surface stays off
-->Panel lamp #2 (Off) stays off
-Channel is OFF, and the GPIO panel button #5 (READY) is depressed. Console: turn the channel ON
-->The OFF lamp on surface goes out
-->Panel lamp #2 (Off) goes out
-->The channel ON lamp on surface lights up
-GPIO 4/5 functions

External mic GPIO

-Channel is OFF. Console: turn the channel ON
-->- Panel lamp #1 (ON) lights up
-->- Panel lamp #2 (OFF) goes out
-->- Panel lamps #3,4,5 keep their previous states
-Console: Take the source on a fader
-->The console must reset the GPIO panel lamps: Panel lamp #2 (Off) is lit
-->Panel lamps #1,3,4,5 are off
-Channel is OFF. GPIO: press and release panel button #1 (ON)
-->Console turns the channel ON: audio, indications
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #1 (On) lights up
-->Panel lamp #2 (Off) goes out
-Channel is ON. Console: turn the channel OFF
-->Panel lamp #2 (Off) lights up
-->Panel lamps #3,4,5 keep their previous states
-->Panel lamp #1 (On) goes out
-GPIO panel button #4 (MUTE) is depressed. GPIO: release panel button #4
-->Console unmutes the channel input: audio, indications
-->Console ON lamp lights up
-->Panel lamp #4 (MUTE) goes out
-->Panel lamp #1 (ON) lights up
-->Panel lamps #2,3,5 keep their previous states
-Channel is ON. GPIO: press and release panel button #2 (OFF)
-->Console turns the channel OFF: audio, indications
-->Panel lamp #1 (On) goes out
-->Panel lamp #2 (Off) lights up
-->Panel lamps #3,4,5 keep their previous states
-GPIO panel button #3 (TALK to CR) is depressed. GPIO: release panel button #3
-->Console deactivates talk to CR: audio, indications
-->Panel lamp #3 (TALK to CR) goes out
-->Panel lamps #1,2,5 keep their previous states, lamp #4 goes out
-GPIO: press panel button #3 (TALK to CR)
-->Console activates talk to CR: audio, indications
-->Panel lamp #3 (TALK to CR) lights up
-->Panel lamps #1,2,5 keep their previous states, lamp #4 lights up

-GPIO: press panel button #4 (MUTE)
-->Console mutes the channel input: audio, indications
-->Console ON lamp goes out
-->Panel lamp #1 (On) goes out
-->Panel lamps #2,3,5 keep their previous states
-->Panel lamp #4 (MUTE) lights up
-GPIO: press panel button #5 (NOT CONNECTED)
-->No any changes
-GPIO: release panel button #5 (NOT CONNECTED)
-->No any changes

Channel without source

-Turn the channel OFF, no source on the channel.
-->Console OFF lamp stays off all the time. No lights appeared of the OFF lamp.
-Turn the channel ON, no source on the channel.
-->Turn the channel ON No lights appeared of the ON lamp
-> Source profile configuration is not saved after pressing "Cancel" when creating new or editing source

2. PIELIKUMS
Programmatūras pirmkoda fragments

```
# Version                 1.3.1
# Updated                 25.05.2017
# Author                  Eriks Burtnieks
# Last changed      Eriks Burtnieks
#-------------------------------------------------------------------------------------------------------------------------------#
# Automated tests for remote control test case, Line source type
#
# IMPORTANT: - MUST BE CHECKED MANUALLY
# - Console must containt nothing under /sources
# - /remote on console must not have any source using the 25090-25099 channel range
#
# PARAMETERS:
# GPIO iPort address                    AS add
# Console address                       AS conadd
# Audio validation console address      AS acadd
# Channel/fader number                  AS channel
# Primary source for channels           AS psrc
# Backfeed source                       AS bfchannel
#
# Source creation automatic, needs for console to have no sources set!
# Full run takes around 5-10 minutes.
# Any single test block can be taken out and run separately if needed, but the setup block needs to be taken in any case
(marked as # SETUP # - # ENDSETUP # block)
#
# PHONE source and audio checks not reliable
#
# Every test step follows the following structure (based on Gherkin Given/When/Then structure):
#
# << <Description from scenario>
# # Given: <Pre-condition>
# - Pre-condition setup and check, unless already in such state from previous step
# # When: <Action to be performed>
# - Action done
# # Then: <Expected result short description>
# - Check the results of the action
#
# Expected result description guide:
# # ON - turns ON
# # OFF - turns OFF
# # KEEP - previous state persists
# # FLASH - flashes for 100 ms
# # NONE - no action
# # Audio - ON or OFF, the target device is generally Program 1 output, unless step specifies otherwise (Talk to
CR/Preview/etc)
#
# Version history: (lists only major changes/improvements)
# 1.0 - Initial version, GPIO checks ready
# 1.1 - Audio checks ready, but need adjustments
# 1.2 - GPI changed to implicit press-release where applicable
# 1.3 - Re-structured to Gherkin structure to drastically improve readability and ease of modification
# 1.3.1 - Split up source types in different files
# 1.x - TODO: Configuration automated
#
#-------------------------------------------------------------------------------------------------------------------------------#

# SETUP #

<< Starting "Remote Control" test

%ver Remote Control
```

```
# Initializing target devices, setting up inputs/outputs, authorization and clearing cache
S1= ${add}:93
S1: LOGIN a
S2= ${conadd}:4010
S2: LOGIN a
S3= ${acadd}:93
S3: LOGIN a
S1: ADD GPI
S1: ADD GPO
S3: MTR ADD
sleep 0.75
? LAST_S1.add
? LAST_S2.add
? LAST_S3.add
peek_acceptable_error 15
S2: SET FaCH#${channel} Fader_Gain=0
S2: SET FaCH#${channel} ON_State=OFF
S2: SET FaCH#${channel} Asg_PREV=OFF
S2: SET FaCH#${channel} Asg_PGM1=OFF
S1: GPI 5 hhhhh
sleep 0.75

# Create primary source
WWW= #follow#http://${conadd}/outputs user a
WWW get_form 1
WWW select lwout_2 255
WWW text lwch_2 25091
WWW select lwMode_2 2
WWW button lw_save

# Binding Program 1 Output (Main audio)
WWW= #follow#http://${conadd}/outputs user a
WWW get_form 1
WWW select lwout_0 1
WWW text lwch_0 25092
WWW select lwMode_0 2
WWW button lw_save
WWW= #follow#http://${acadd}/cgi-bin/cgi_dsts user a
WWW get_form 0
WWW text R001_RURL 25092
WWW button Apply

# Binding Talk to CR Output
WWW= #follow#http://${conadd}/outputs user a
WWW get_form 1
WWW select lwout_3 13
WWW text lwch_3 25093
WWW select lwMode_3 2
WWW button lw_save
WWW= #follow#http://${acadd}/cgi-bin/cgi_dsts user a
WWW get_form 0
WWW text R002_RURL 25093
WWW button Apply

# Binding Preview
WWW= #follow#http://${conadd}/outputs user a
WWW get_form 1
WWW select lwout_4 12
WWW text lwch_4 25094
WWW select lwMode_4 2
WWW button lw_save
WWW= #follow#http://${acadd}/cgi-bin/cgi_dsts user a
WWW get_form 0
WWW text R003_RURL 25094
```

```
WWW button Apply

# Binding Talk to Previewed Sources
WWW= #follow#http://${conadd}/outputs user a
WWW get_form 1
WWW select lwout_1 255
WWW text lwch_1 25099
WWW select lwMode_1 2
WWW button lw_save
WWW= #follow#http://${conadd}/sources user a
WWW get_form 0
WWW select id new_codec
WWW button Create
WWW get_form 0
WWW text srcn backfeed
WWW text psid 25099
WWW select lpen 1
WWW button Apply
S2: SET FaCH#${bfchannel} ON_State=OFF
S2: SET FaCH#${bfchannel} src_id=1
sleep 0.75

# Binding GPIO 1 channel
WWW= #follow#http://${conadd}/gpio user a
WWW get_form 0
WWW text gpio0_psid 25091
WWW button local_gpio_save

<< Turning on Program 1 for channel audio validation
S2: SET FaCH#${channel} Asg_PGM1=ON
sleep 0.75
? LAST_S2 EVENT FaCH#${channel} Asg_PGM1=ON
sleep 0.75

# Hardware configuration setup goes here

<< Clearing command cache
clear S1
clear S2
clear S3
<< Cache cleared

# ENDSETUP #

<< Starting tests

# Line source GPIO test block
<< ---Line source GPIO

# Create Line source
<< Creating Line source

WWW= #follow#http://${conadd}/sources user a
WWW get_form 0
WWW select id new_line
WWW button Create
WWW get_form 0
WWW text srcn 0_line
WWW text psid ${psrc}
WWW select lpen 1
WWW button Apply

<< Line source created
```

```
<< Console: Take the source on a fader
# Given: Channel OFF, no source
S2: SET FaCH#${channel} src_id=0
sleep 0.75
? LAST_S2 EVENT FaCH#${channel} src_id=0,src_name=""
sleep 0.75
# When: Console: Take the source on a fader
S2: SET FaCH#${channel} src_id=2
sleep 0.75
# Then: Source set on channel, panel #2 ON, #1,#3,#4,#5 OFF
? LAST_S2 EVENT FaCH#${channel} src_id=2,src_name="0_line"
? LAST_S1 GPO 1 hLhhh
sleep 0.75

<< Channel is OFF. Console: turn the channel ON
# Given: Channel is OFF.
# When: Console: turn the channel ON
S2: SET FaCH#${channel} ON_State=ON
sleep 0.75
# Then: panel #1 ON, #2 OFF, #3 KEEP, #4 FLASH, #5 NONE
? LAST_S1 GPO 1 Llhhh|nGPO 1 lHhhh|nGPO 1 lhhLh|nGPO 1 lhhHh
sleep 0.75

<< Channel is ON. GPIO: press and release panel button #1 (ON)
# Given: Channel is ON.
# When: GPIO: press and release panel button #1 (ON)
S1: GPI 5 Lhhhh
sleep 0.75
# Then: No operations.
? LAST_S1 GPI 5 Lhhhh
sleep 0.75
S1: GPI 5 Hhhhh
sleep 0.75

<< Channel is ON. Console: press and release the channel ON button
# Given: Channel is ON.
# When: Console: press and release the channel ON button
S2: SET FaCH#${channel} ON_State=ON
sleep 0.75
# Then: No operations.
S1: GPO 1
sleep 0.75
? LAST_S1 GPO 1 lhhhh
sleep 0.75

<< Channel is ON. GPIO: press and release panel button #2 (OFF)
# Given: Channel is ON.
# When: GPIO: press and release panel button #2 (OFF)
S1: GPI 5 hLhhh
sleep 0.75
# Then: panel #1 OFF, #2 ON, #3 KEEP, #4 NONE, #5 FLASH, Audio OFF
? LAST_S1 GPI 5 hLhhh|nGPO 1 Hhhhh|nGPO 1 hLhhh|nGPO 1 hlhhL|nGPO 1 hlhhH
sleep 0.75
S3: MTR OCH 2
sleep 0.75
S3: MTR OCH 2
sleep 0.75
? PEEK L:-1000 R:-1000
sleep 0.75
S1: GPI 5 hHhhh
sleep 0.75
? LAST_S1 GPI 5 hHhhh
sleep 0.75
```

```
<< Channel is OFF. GPIO: press and release panel button #1 (ON)
# Given: Channel is OFF.
# When: GPIO: press and release panel button #1 (ON)
S1: GPI 5 Lhhhh
sleep 0.75
# Then: panel #1 ON, #2 OFF, #3 KEEP, #4 FLASH, #5 NONE, Audio ON
? LAST_S1 GPI 5 Lhhhh|nGPO 1 Llhhh|nGPO 1 lHhhh|nGPO 1 lhhLh|nGPO 1 lhhHh
sleep 0.75
S3: MTR OCH 2
sleep 0.75
S3: MTR OCH 2
sleep 0.75
? PEEK L:-202 R:-202
sleep 0.75
S1: GPI 5 Hhhhh
sleep 0.75
? LAST_S1 GPI 5 Hhhhh
sleep 0.75

<< Channel is ON. Console: turn the channel OFF
# Given: Channel is ON.
# When: Console: turn the channel OFF
S2: SET FaCH#${channel} ON_State=OFF
sleep 0.75
# Then: panel #1 OFF, #2 ON, #3 KEEP, #4 NONE, #5 FLASH
? LAST_S1 GPO 1 Hhhhh|nGPO 1 hLhhh|nGPO 1 hlhhL|nGPO 1 hlhhH
sleep 0.75

<< Channel preview is OFF. GPIO: press panel button #3 (Preview)
# Given: Channel preview is OFF.
# When: GPIO: press panel button #3 (Preview)
S1: GPI 5 hhLhh
sleep 0.75
# Then: panel #1 KEEP, #2 KEEP, #3 ON, #4 KEEP, #5 KEEP, Audio ON (Preview)
? LAST_S1 GPI 5 hhLhh|nGPO 1 hlLhh
sleep 0.75
S3: MTR OCH 4
sleep 0.75
S3: MTR OCH 4
sleep 0.75
? PEEK L:-202 R:-202
sleep 0.75

<< GPIO panel button #3 (Preview) is depressed, and channel preview is ON. GPIO: release panel button #3 (Preview)
# Given: GPIO panel button #3 (Preview) is depressed, and channel preview is ON.
# When: GPIO: release panel button #3 (Preview)
S1: GPI 5 hhHhh
sleep 0.75
# Then: #1 KEEP, #2 KEEP, #3 OFF, #4 KEEP, #5 KEEP, Audio OFF (Preview)
? LAST_S1 GPI 5 hhHhh|nGPO 1 hlHhh
sleep 0.75
S3: MTR OCH 4
sleep 0.75
S3: MTR OCH 4
sleep 0.75
? PEEK L:-1000 R:-1000
sleep 0.75

<< Channel preview is OFF. Console: turn preview on
# Given: Channel preview is OFF.
# When: Console: turn the preview on
S2: SET FaCH#${channel} Asg_PREV=ON
sleep 0.75
# Then: #1 KEEP, #2 KEEP, #3 ON, #4 KEEP, #5 KEEP
```

```
? LAST_S1 GPO 1 hlLhh
sleep 0.75

<< Channel preview is ON. Console: turn preview off
# Given: Channel preview is ON.
# When: Console: turn preview off
S2: SET FaCH#${channel} Asg_PREV=OFF
sleep 0.75
# Then: #1 KEEP, #2 KEEP, #3 OFF, #4 KEEP, #5 KEEP
? LAST_S1 GPO 1 hlHhh
sleep 0.75

<< Channel is ON. GPIO: press and release panel button #4 (Reset)
# Given: Channel is ON.
S2: SET FaCH#${channel} ON_State=ON
sleep 0.75
? LAST_S1 GPO 1 Llhhh|nGPO 1 lHhhh|nGPO 1 lhhLh|nGPO 1 lhhHh
sleep 0.75
# When: GPIO: press and release panel button #4 (Reset)
S1: GPI 5 hhhLh
sleep 0.75
# Then: #1 OFF, #2 ON, #3 KEEP, #4 KEEP, #5 KEEP, Audio OFF
? LAST_S1 GPI 5 hhhLh|nGPO 1 Hhhhh|nGPO 1 hLhhh
sleep 0.75
S3: MTR OCH 2
sleep 0.75
S3: MTR OCH 2
sleep 0.75
? PEEK L:-1000 R:-1000
sleep 0.75
S1: GPI 5 hhhHh
sleep 0.75
? LAST_S1 GPI 5 hhhHh
sleep 0.75

<< Enabling GPIO ready-only
# Enable ready-only
WWW= #follow#http://${conadd}/sources?id=2 user a
WWW get_form 0
WWW checkbox rden checked
WWW button apply
<< Enabled

<< GPIO ready only: Channel is OFF, no source on the channel. While keeping panel button #5 (READY) released,
take a source
# Given: GPIO ready only: Channel is OFF, no source on the channel.
S2: SET FaCH#${channel} src_id=0
sleep 0.75
? LAST_S2 EVENT FaCH#${channel} src_id=0,src_name=""
sleep 0.75
# When: While keeping panel button #5 (READY) released, take a source
S2: SET FaCH#${channel} src_id=2
sleep 0.75
# Then: Source set on channel, #1,#2,#3,#4,#5 OFF
? LAST_S2 EVENT FaCH#${channel} src_id=2,src_name="0_line"
sleep 0.75
? LAST_S1 GPO 1 hHhhh
sleep 0.75

<< GPIO ready only: Channel is OFF. GPIO: press and release panel button #5 (READY)
# Given: GPIO ready only: Channel is OFF.
S1: GPO 1
sleep 0.75
? LAST_S1 GPO 1 hhhhh
```

```
sleep 0.75
# When: GPIO: press and release panel button #5 (READY)
S1: GPI 5 hhhhL
sleep 0.75
# Then: panel #2 ON, after release panel #2 OFF
? LAST_S1 GPI 5 hhhhL|nGPO 1 hLhhh
sleep 0.75
S1: GPI 5 hhhhH
sleep 0.75
? LAST_S1 GPI 5 hhhhH|nGPO 1 hHhhh
sleep 0.75

<< GPIO ready only: Channel is ON, and the GPIO panel button #5 (READY) is released. Console: turn the channel OFF
# Given: GPIO ready only: Channel is ON, and the GPIO panel button #5 (READY) is released.
S2: SET FaCH#${channel} ON_State=ON
sleep 0.75
? LAST_S2 EVENT FaCH#${channel} ON_State=ON
sleep 0.75
? LAST_S1 GPO 1 Lhhhh|nGPO 1 lhhLh|nGPO 1 lhhHh
sleep 0.75
# When: Console: turn the channel OFF
S2: SET FaCH#${channel} ON_State=OFF
sleep 0.75
# Then: Channel OFF, panel #2 OFF
? LAST_S2 EVENT FaCH#${channel} ON_State=OFF
? LAST_S1 GPO 1 Hhhhh|nGPO 1 hhhhL|nGPO 1 hhhhH
sleep 0.75

<< GPIO ready only: Channel is ON, and the GPIO panel button #5 (READY) is pressed. Console: turn the channel OFF
# Given: GPIO ready only: Channel is ON,and the GPIO panel button #5 (READY) is pressed.
S2: SET FaCH#${channel} ON_State=ON
sleep 0.75
? LAST_S1 GPO 1 Lhhhh|nGPO 1 lhhLh|nGPO 1 lhhHh
sleep 0.75
S1: GPI 5 hhhhL
sleep 0.75
? LAST_S1 GPI 5 hhhhL
sleep 0.75
# When: Console: turn the channel OFF
S2: SET FaCH#${channel} ON_State=OFF
sleep 0.75
# Then: Channel OFF, panel #2 ON
? LAST_S1 GPO 1 Hhhhh|nGPO 1 hLhhh|nGPO 1 hlhhL|nGPO 1 hlhhH
sleep 0.75

<< GPIO ready only: Channel is OFF, and the GPIO panel button #5 (READY) is depressed. Console: turn the channel ON_State
# Given: GPIO ready only: Channel is OFF, and the GPIO panel button #5 (READY) is depressed.
S1: GPI 5 hhhhH
sleep 0.75
? LAST_S1 GPI 5 hhhhH|nGPO 1 hHhhh
sleep 0.75
# When: Console: turn the channel ON
S2: SET FaCH#${channel} ON_State=ON
sleep 0.75
# Then: Channel ON, panel #1 ON
? LAST_S2 EVENT FaCH#${channel} ON_State=ON
? LAST_S1 GPO 1 Lhhhh|nGPO 1 lhhLh|nGPO 1 lhhHh
sleep 0.75

# NON_STEP: Turn off before removing source
S2: SET FaCH#${channel} ON_State=OFF
```

```
sleep 0.75
? LAST_S2 EVENT FaCH#${channel} ON_State=OFF
? LAST_S1 GPO 1 Hhhhh|nGPO 1 hhhhL|nGPO 1 hhhhH

<< Disabling GPIO ready-only
# Disable ready-only
WWW= #follow#http://${conadd}/sources?id=2 user a
WWW get_form 0
WWW checkbox rden unchecked
WWW button apply
<< Disabled

# Delete Line source
<< Deleting Line source

WWW= #follow#http://${conadd}/sources user a
WWW get_form 1
WWW checkbox rm-2 checked
WWW button src-remove

<< Line source deleted

<< Channel without source

S2: SET FaCH#${channel} src_id=0
S2: SET FaCH#${channel} ON_State=ON
? LAST_S2 EVENT FaCH#${channel} src_id=0,src_name=""|nEVENT FaCH#${channel} ON_State=ON
sleep 0.75
#Turn the channel OFF, no source on the channel
S2: SET FaCH#${channel} ON_State=OFF
sleep 0.75
? LAST_S2 EVENT FaCH#${channel} ON_State=OFF
sleep 0.75

S2: SET FaCH#${channel} ON_State=OFF
sleep 0.75
#Turn the channel ON, no source on the channel
S2: SET FaCH#${channel} ON_State=ON
sleep 0.75
? LAST_S2 EVENT FaCH#${channel} ON_State=ON
sleep 0.75

<< Resetting to default state
S2: SET FaCH#${channel} ON_State=OFF
S1: GPI 5 hLhhh
S2: SET FaCH#${channel} src_id=0
S2: SET FaCH#${channel} Asg_PGM1=OFF
<< Set to default completed

# Deleting backfeed source
WWW= #follow#http://${conadd}/sources user a
WWW get_form 1
WWW checkbox rm-1 checked
WWW button src-remove

<< Finished successfully
```

Kvalifikācijas darbs “Daudzkanālu skaņas kodeka funkciju testu automatizācija” izstrādāts Latvijas Universitātes Datorikas fakultātē.

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

Autors: Ēriks Burtnieks _____ 29.05.2017.

Rekomendēju darbu aizstāvēšanai

Darba vadītāja: Dr. dat., Vineta Arnicāne _____ ___.05.2017.

Recenzents: *B. dat., Jānis Knets*

Darbs iesniegts 29.05.2017.

Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: *Darja Solodovņikova*_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

___.06.2017. prot. Nr. _____

Komisijas sekretārs(-e): _____