

Liepājas Valsts tehnikums

**Testēšanas rīks “TT/CU Skeneris”**

Programmatūras projekta tehniskā dokumentācija

Profesionālā kvalifikācija ......................................................................

Grupas nosaukums ......................................................................

Projekta izstrādātājs ............................................................

/vārds, uzvārds, paraksts/

Eksāmena datums 2024. gada 21. jūnijā

Liepāja 2024

**Saturs**

[1. Uzdevuma formulējums 5](#_Toc165200997)

[2. Programmatūras prasību specifikācija 7](#_Toc165200998)

[2.1. Produkta perspektīva 7](#_Toc165200999)

[2.2. Sistēmas funkcionālās prasības 7](#_Toc165201000)

[2.2. Sistēmas nefunkcionālās prasības 19](#_Toc165201001)

[2.3. Gala lietotāja raksturiezīmes 19](#_Toc165201002)

[3. Izstrādes līdzekļu, rīku apraksts un izvēles pamatojums 20](#_Toc165201003)

[3.1. Izvēlēto risinājuma līdzekļu un valodu apraksts 20](#_Toc165201004)

[3.1.1. Izstrādes programmatūras 20](#_Toc165201005)

[3.1.2. Izstrādes darbstacija 21](#_Toc165201006)

[3.1.3. Produkta komponentes un ierīces 21](#_Toc165201007)

[3.1.4. Programmatūras programmēšanas valodas un bibliotēkas 22](#_Toc165201008)

[3.2. Iespējamo risinājuma līdzekļu un valodu apraksts 23](#_Toc165201009)

[4. Sistēmas modelēšana un projektēšana 24](#_Toc165201010)

[4.1. Sistēmas struktūras modelis 24](#_Toc165201011)

[4.1.1. Pievienota mikročipa noteikšana 25](#_Toc165201012)

[4.1.2. Real-Time-Clock komponentes pārbaude 25](#_Toc165201013)

[4.1.3. EEPROM komponentes pārbaude 26](#_Toc165201014)

[4.1.4. ControlUnit releju pārbaude 26](#_Toc165201015)

[4.1.5. ControlUnit Wiegand karšu lasītāja pārbaude 27](#_Toc165201016)

[4.1.6. ControlUnit mikročipa pārbaudes rezultāts 27](#_Toc165201017)

[4.1.7. TTunit mikročipa pārbaudes rezultāts 28](#_Toc165201018)

[4.1.8. Pārbaudes atkārtošana 28](#_Toc165201019)

[4.2. Klašu diagramma / ER diagramma 29](#_Toc165201020)

[4.3. Funkcionālais un dinamiskais sistēmas modelis 31](#_Toc165201021)

[4.3.1. Stāvokļu diagramma 31](#_Toc165201022)

[5. Lietotāju ceļvedis 32](#_Toc165201023)

[5.1 TTunit mikročipa pārbaudes sagataves process 32](#_Toc165201024)

[5.2 ControlUnit mikročipa pārbaudes sagataves process 33](#_Toc165201025)

[5.3 ControlUnit mikročipa Wiegand karšu lasītāja pārbaudes process 34](#_Toc165201026)

[5.4 Saskarnes pamācība 37](#_Toc165201027)

[6. Testēšanas dokumentācija 39](#_Toc165201028)

[6.1. Izvēlētās testēšanas metodes, rīku apraksts un pamatojums 39](#_Toc165201029)

[6.1.1. Melnās kastes testēšanas rīku apraksts un pamatojums 39](#_Toc165201030)

[6.1.2. Baltās kastes testēšanas rīku apraksts un pamatojums 40](#_Toc165201031)

[6.2. Testpiemēru kopa 40](#_Toc165201032)

[6.3. Testēšanas žurnāls 40](#_Toc165201033)

[7. Secinājumi 41](#_Toc165201034)

[7.1.1. Rezultāta novērtējums 41](#_Toc165201035)

[7.1.2. Uzdevumu sasniegšanas analīze 41](#_Toc165201036)

[7.1.3. Darba apjoms 42](#_Toc165201037)

[8. Lietoto terminu un saīsinājumu skaidrojumi 43](#_Toc165201038)

[9. Literatūras un informācijas avotu saraksts 44](#_Toc165201039)

[Pielikumi 45](#_Toc165201040)

**Ievads**

Šis dokuments tiek uzrakstīts, ievērojot "Liepājas Valsts tehnikums" mācību programmas "Programmēšanas tehniķis" kvalifikācijas eksāmenu prasības. Projekta mērķis ir izstrādāt augstas kvalitātes testēšanas rīku "TT/CU skeneris" programmatūru, kas atbilst šīs izglītības iestādes prasībām.

Galvenais aspekts, ko šis dokuments piedāvā, ir plaša programmatūras produkta analīze, ietverot perspektīvu, līdzekļus un modeļus, lai sasniegtu maksimālo efektivitāti un funkcionālo pilnveidi. Var uzskatīt, ka šis projekts ietverēs inovatīvus risinājumus, kas būs saskaņots ar profesijas nozares labākajām praksēm.

Izstrādātās programmatūras nepieciešamība izriet no vajadzībām izstrādātāja prakses vietas uzņēmumā “SIA InPass”. Pēc pasūtītāja velmēm izstrādātājs ir apņēmies radīt produkta programmatūru, kas efektīvi pārbaudīs dažādus mikročipus, piemēram, TTunit un ControlUnit. Šie mikročipi ir būtiski izstrādātāja prakses vietā, veicot dažādu signālu pārraides un nodrošinot svarīgas funkcionalitātes, piemēram, releju ieslēgšanos un karšu nolasīšanu.

Produkts tiks izstrādāts, lai nodrošinātu pārbaudi dažādām iekšējām mikročipa funkcionalitātēm un komponentēm, kas tiek izmantoti izstrādātāja prakses vietā. Tā kā produkta gala lietotāji būs darbinieki no attiecīgā uzņēmuma, izstrādātājs pievērsīs īpašu uzmanību funkcionalitātes un ergonomikas izstrādei.

Produkts sastāvēs no mikrokontrolliera, pogas, displeja un plastmasas ietvara, nodrošinot integrētu risinājumu ar elegantu un modernu dizainu. Ergonomiskā un vienkāršā saskarne būs lietotājam draudzīga, un tā tiks precīzi aprakstīta angļu valodā ar dažādiem nozares profesionālismiem, lai atbilstu starptautiskajiem standartiem un vienkāršotu lietošanas pieredzi darbiniekiem.

Produkta saskarnes dizains tiks pielāgots, lai efektīvi mijiedarbotos ar pogu sistēmu, kas ļaus lietotājam intuitīvi progresēt caur testa posmiem un iegūt galīgos komponenšu darbības stāvokļa rezultātus. Izstrādātājs vēlas nodrošināt, ka produkts ne tikai izpilda savu pamatfunkciju, bet arī piedāvā lietotājiem ērtu un efektīvu darbību, atspoguļojot izstrādātāja apņemšanos sniegt augsta līmeņa risinājumu.

# Uzdevuma formulējums

Šajā nodaļā tiek detalizēti aprakstīts uzdevuma formulējums vispārējam programmatūras produktam, kas paredzēts mikročipu pārbaudei - TTunit vai ControlUnit. Produkts vēlāk seko precīzi definētiem procesiem, kas ietver dažādu funkcionalitātes pārbaudes veikšanu.

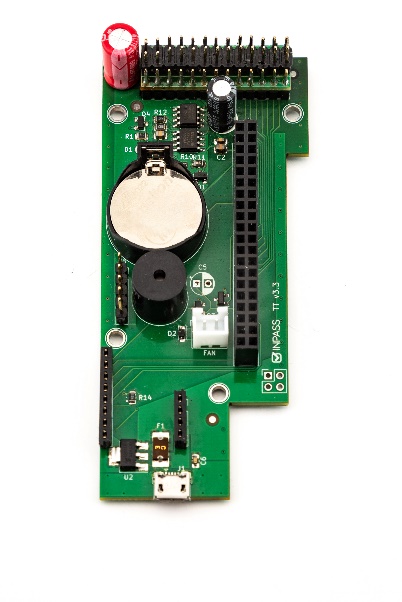
Sākot ar primāro uzdevumu, produkts (skatīt 1. attēlu) tiek veidots ar mērķi noskaidrot TTunit vai ControlUnit mikročipa funkcionalitātes stāvokli, īstenojot dažādus funkcionalitātes testus. Produkta saskarne uz displeja tiek izstrādāta angļu valodā un pielāgota ar dažādiem profesionālismiem.



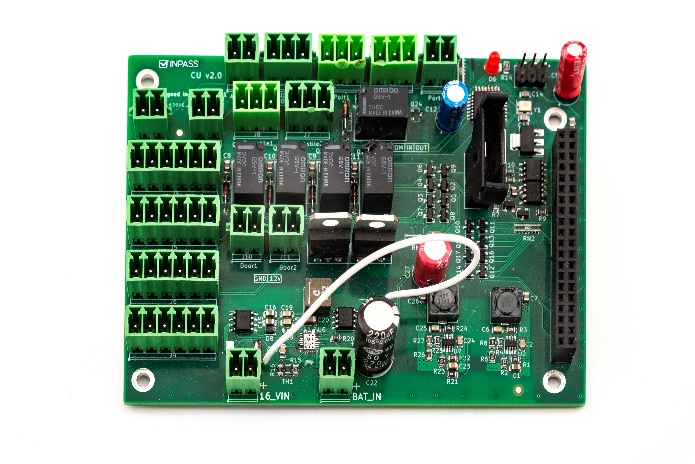
**1.attēls. Izstrādājamais produkts.**

Produkta testēšanas process tiek inicializēts, savienojot mikrokontrolieri ar TTunit vai ControlUnit mikročipu. Šajā posmā, izmantojot pogas un displeja interfeisu, produkts pakāpeniski progresē caur testu, identificējot mikročipu un veicot atbilstošus testus, kuru rezultāti tiek rūpīgi dokumentēti un pārbaudes beigās tiek izvadīts uz displeja.

Pievienojot produktu pie TTunit mikročipa (skatīt 2.attēlu), uz displeja tiek sniegta signalizācija par veiksmīgu pievienošanos, un tiek izpildīti vairāki testi, tostarp Real-Time-Clock, EEPROM un zumera testēšana. Savukārt, savienojot produktu ar ControlUnit mikročipu (skatīt 3. attēlu), displejā tiek attēlots, ka ir pievienots ControlUnit mikročips, un tiek veikti vēl papildus pārbaudes ar vairākiem sarežģītiem procesiem, ietverot Real-Time-Clock, EEPROM, pogu releju, durvju releju, RFID karšu lasītāja un signalizācijas stāvokļa releja pārbaudi.



**2.attēls. TTunit mikročips**

****

**3.attēls. ControlUnit mikročips**

Pārbaudes rezultāti tiek skaidri atspoguļoti displejā, norādot, vai pārbaude ir veiksmīga vai neveiksmīga. Gadījumā, ja mikročips netiek pievienots produktam, tiek izvērsti dažādi scenāriji, lai informētu lietotāju par mikročipa neesošo pievienošanos. Arī gadījumā, ja viena no esošām mikročipa komponentēm, kas konfliktē ar mikročipa identificēšanu, ir daļēji bojāta, tad displejā tiek izvadīta iespējamā komponente, kas izraisa problēmu.

Produkts nodrošina, kad kāda no mikročipa testējamajām komponentēm nav darba kārtībā. Tad displejā tiek izvadīts paziņojums par neesošo komponentes darbību. Lietotājam tiek piedāvāta iespēja restartēt testu, izmantojot displeja interfeisu un pogu, ļaujot veikt papildu testēšanu pēc nepieciešamības un pārliecības.

# Programmatūras prasību specifikācija

Šajā nodaļā tiek aprakstītas vairākas apakšnodaļas : produkta perspektīva, kur tiek aprakstīta vispārējā produkta būtība, sistēmas funkcionālās un nefunkcionālās prasības nosaka visas produkta iespējamās funkcionalitātes, gala lietotāja raksturiezīmes iezīmē produkta palaišanas brīža gala lietotāja raksturiezīmes.

## Produkta perspektīva

Produkta primārā funkcija ir pārbaudīt dažādu mikročipu (TTunit vai ControlUnit) funkcionalitātes stāvokli. Veicot šo pārbaudi, produkts nodrošina galējo rezultātu atbilstošam mikročipam norādot to komponenšu stāvokli, kā arī specifiskos gadījumos (skatīt 15.attēlu) norādīs pievienoto ierīču stāvokli. Pārbaudes beigās produkts nodrošina lietotājam veikt vairākkārtēju pārbaudes iespējamību.

## Sistēmas funkcionālās prasības

**P.1. Galvenā mikročipu pārbaudes funkcija**

Mērķis: Uzsākt mikročipu pārbaudes procesu un iegūt rezultātus.

Ievaddati: Uzsākt rīku ar kādu no mikročipiem, skatīt procesu pēc 5. nodaļas.

Apstrāde:

1. Pārbauda kādas ir pievienotās komponentes pie mikročipa;
2. Izveido sākuma kadru ar jauna kadra funkciju (skatīt P.4.);
3. Pārbauda kāds mikročips ir pievienots pie rīka;
4. Uz ekrāna izvada mikročipa pievienotās komponentes heksadecimālās vērtības;
5. Attiecīgi pēc pievienotā mikročipa veic pārbaudes funkcijas (skatīt P.5 - P.12);
6. Pēc mikročipu pārbaudes funkciju izpildes informē, lai lietotājs nospiež fizisku pogu rīka labajā pusē(skatīt 4.attēlu).



**4.attēls. Fiziskas pogas nospiešana**

Izvaddati:

1. Uz displeja tiek izvadīts sākuma kadrs attiecīgi no pievienota mikročipa (skatīt 5. attēlu);



**5.attēls. Sākuma kadrs uz displeja**

1. Uz displeja tiek izvadīts jauns kadrs mikročipa komponenšu adreses (skatīt 6.attēlu);



**6.attēls. Mikročipa komponenšu adreses kadrs**

1. Uz displeja tiek izvadīti attiecošie pārbaudes funkciju kadri, atkarībā no pievienotā mikročipa, skatīt pārējās funkcionālās prasības;
2. Uz displeja tiek izvadīts informatīvais kadrs, kas liecina, lai lietotājs vēlreiz nospiež pogu mikročipa pārbaudes apstiprināšanas nolūkiem.

**P.2. Pirms testa EEPROM pārbaude**

Mērķis: Pārbaudīt mikročipa EEPROM komponenti pirms galvenā testa sākšanas priekš precīzas EEPROM komponentes pārbaudes.

Ievaddati: Uzsākt rīku, skatīt procesu pēc 5. nodaļas.

Apstrāde:

1. Ieraksta vērtību “test” EEPROM komponentes atmiņā;
2. Pārbauda vai ierakstītā vērtība sakrīt ar vērtību “test” nolasot no EEPROM komponentes atmiņas;
3. Atgriež vērtību, kas liecinātu vai EEPROM komponente precīzi raksta un nolasa datus.

Izvaddati: Tiek atgriezta vērtība attiecīgi EEPROM komponentes rakstīšanas, lasīšanas funkcionalitātei.

**P.3. Pirms testa rīka un ControlUnit mikrokontrollieru komunikācijas pārbaude**

Mērķis: Pārbaudīt ControlUnit un rīka komunikāciju pirms galvenā testa sākšanas priekš precīzas ControlUnit un rīka komunikācijas pārbaudes.

Ievaddati: Uzsākt rīku, skatīt procesu pēc 5. nodaļas

Apstrāde:

1. Pievieno četru baitu komandu – “get”, “board\_version” četru baitu komandu sarakstam (skatīt P.10);
2. Nosūta četru baitu komandu uz ControlUnit mikrokontrolliera (skatīt P.11.);
3. Nolasa atbildi no ControlUnit mikrokontrolliera ar UART palīdzību
4. Atgriež nolasīto vērtību no ControlUnit mikrokontrolliera.

Izvaddati: Tiek atgriezta vērtība attiecīgi ControlUnit mikrokontrolliera atbildei.

**P.4.  Jauna kadra izveide**

Mērķis: Izvadīt uz displeja jauna kadra izveidi

Ievaddati: Progresējot caur programmai funkcija tiek izsaukta specifiskos brīžos ar specifisku indeksu (skatīt 1.tabulu).

**1.tabula**

**Jauna kadra ievaddati**

|  |  |
| --- | --- |
| Mainīgā nosaukums | Vērtība |
| Page | Cipars vai ‘res’ |

Apstrāde:

1. Funkcijai izsaucoties pārbauda kāds mikročips ir pievienots pie rīka un attiecoši izvada uz ekrāna mikročipa nosaukumu;
2. Izveido InPass logo un to uz displeja izvada;
3. No dota indeksa izvada uz displeja testa numuru vai rezultātu, ja ir dots cipars kā indekss, tad uz displeja izvada “x. test” vai, ja ir dots “res” kā indekss tad uz displeja izvada “Results”.

Izvaddati:

1. Ja specifiskais indekss ir cipars un ir pievienots TTunit, tad izvada jaunizveidoto kadru uz displeja (skatīt 7. attēlu);



**7.attēls. Jauns kadrs ar cipara indeksu un pievienotu TTunit**

1. Ja specifiskais indekss ir “res” un ir pievienots ControlUnit, tad uz displeja izvada jaunizveidoto kadru (skatīt 8.attēlu);



**8.attēls. Jauns kadrs ar rezultāta indeksu un pievienotu ControlUnit**

**P.5. Real-Time-Clock komponentes pārbaude**

Mērķis: Ļaut lietotājam pārbaudīt Real-Time-Clock komponentes stāvokli.

Ievaddati:  Izpildīt P.1 un nospiezt fizisku pogu rīka labajā malā (skatīt 4.attēlu)

Apstrāde:

1. Galvenā mikročipu pārbaudes funkcija palaiž Real-Time-Clock pārbaudes funkciju;
2. Funkcijas process ieraksta Real-Time-Clock komponentes noteiktu datumu;
3. Funkcijas process nolasa no Real-Time-Clock komponentes iepriekš ierakstīto datumu;
4. Funkcijas process pārbauda vai nolasītais Real-Time-Clock komponentes datums sakrīt ar Real-Time-Clock komponentes ierakstīto datumu iekš programmā.

Izvaddati:



**9.attēls. Real-Time-Clock pārbaudes kadrs**

Real-Time-Clock komponentes pārbaudes laikā uz displeja tiek izvadīts kadrs (skatīt 9. attēlu).

**P.6. EEPROM komponentes pārbaude**

Mērķis: Ļaut lietotājam pārbaudīt EEPROM komponentes stāvokli.

Ievaddati: Pabeigt P.5. darbības procesu;

Apstrāde:

1. Galvenā mikročipu pārbaudes funkcija (skatīt P.1.) palaiž EEPROM pārbaudes funkciju.
2. Funkcijas process ieraksta iekš EEPROM komponentes nejaušus datus katrā EEPROM komponentes lapā.
3. Funkcijas process nolasa no EEPROM komponentes iepriekš ierakstītos datus.
4. Funkcijas process pārbauda vai nolasītie EEPROM komponentes dati sakrīt ar EEPROM komponentes ierakstītiem datiem iekš programmā.

Izvaddati:



**10.attēls. EEPROM pārbaudes kadrs**

EEPROM komponentes pārbaudes laikā uz displeja tiek izvadīts kadrs (skatīt 10. attēlu).

**P.7. Rīka un ControlUnit komunikācijas pārbaude**

Mērķis: Ļaut lietotājam pārbaudīt rīka un ControlUnit komunikācijas funkcionalitātes stāvokli.

Ievaddati: Pabeigt P.6. darbības procesu;

Apstrāde:

1. Galvenā mikročipu pārbaudes funkcija palaiž rīka un ControlUnit komunikācijas pārbaudes funkciju;
2. Funkcijas process palaiž citu funkciju, kas pārbauda ControlUnit mikrokontrolliera komunikācijas funkcionalitāti (skatīt P.9.);
3. Iepriekš minētā funkcija izvada vērtību, kas nosaka rīka un ControlUnit komunikācijas stāvokli;
4. Pēc izvadītās vērtības rīks liecina vai rīka un ControlUnit komunikācija ir ejoša vai neejoša un šo stāvokli ievada sarakstā priekš mikročipa pārbaudes rezultāta noteikšanas;
5. Izveido jaunu kadru (skatīt P.4.) un uz displeja izvada ka notiek rīka un ControlUnit komunikācijas pārbaude.

Izvaddati:



**11.attēls. Rīka un ControlUnit komunikācijas pārbaudes kadrs**

Rīka un ControlUnit komunikācijas pārbaudes laikā uz displeja tiek izvadīts kadrs (skatīt 11. attēlu).

**P.8. ControlUnit releju komponenšu pārbaude**

Mērķis: Ļaut lietotājam pārbaudīt ControlUnit releju komponenšu stāvokli.

Ievaddati: Pabeigt P.7. procesu un jābūt pievienotam ControlUnit mikročipam pie rīka.

Apstrāde:

1. Rīks nosūta specifisku četru baitu komandu uz ControlUnit mikrokontrollieri, lai aktivizētu “Turnstile1\_a” releju un uz displeja izvada releja nosaukumu.
2. Pēc “Turnstile1\_a” releja aktivizēšanas rīks nosūta nākamo četru baitu komandu uz ControlUnit mikrokontrollieri, lai aktivizētu “Turnstile1\_b” releju un uz displeja izvada releja nosaukumu.
3. Pēc “Turnstile1\_b” releja aktivizēšanas rīks nosūta nākamo četru baitu komandu uz ControlUnit mikrokontrollieri, lai aktivizētu “Turnstile2\_a” releju un uz displeja izvada releja nosaukumu.
4. Pēc “Turnstile2\_a” releja aktivizēšanas rīks nosūta nākamo četru baitu komandu uz ControlUnit mikrokontrollieri, lai aktivizētu “Turnstile2\_b” releju un uz displeja izvada releja nosaukumu.
5. Pēc “Turnstile2\_b” releja aktivizēšanas rīks nosūta nākamo četru baitu komandu uz ControlUnit mikrokontrollieri, lai aktivizētu “Button1” releju un uz displeja izvada releja nosaukumu.
6. Pēc “Button1” releja aktivizēšanas rīks nosūta nākamo četru baitu komandu uz ControlUnit mikrokontrollieri, lai aktivizētu “Button2” releju un uz displeja izvada releja nosaukumu.

Izvaddati: Tiek aktivizētas visas sešu releju darbības un ik katru releja pārbaudes brīdi uz displeja tiek izvadīts releja nosaukums.

**P.9. Rīka un ControlUnit mikrokontrollieru komunikācijas pārbaudes funkcija**

Mērķis: Ļaut lietotājam pārbaudīt rīka un ControlUnit savstarpējo mikrokontrollieru komunikāciju.

Ievaddati: Funkcija tiek palaista iekļaujoties P.7. procesā.

Apstrāde:

1. Rīka mikrokontrollieris pievieno komandu sarakstam (skatīt P.8) četru baitu komandu (skatīt 2. tabulu);

**2.tabula**

**Ievaddati no rīka uz ControlUnit**

|  |  |
| --- | --- |
| Metode | Komanda |
| “get” | “board\_version” |

1. Nosūta doto četru baitu komandu uz ControlUnit mikrokontrollieri ar ControlUnit mikrokontrolliera komandu nosūtīšanas funkciju (skatīt P.9);
2. Pārbauda vai tiek saņemta atbilde no ControlUnit mikrokontrolliera un atgriež vērtību attiecīgi vai ControlUnit mikrokontrollieris atgriež atbildi.

Izvaddati: Tiek atgriezta vērtība attiecīgi ControlUnit mikrokontrolliera atbildei.

**P.10. Četru baitu komandu saglabāšana sarakstā**

Mērķis: Ļaut lietotājam pievienot četru baitu komandu sarakstā.

Ievaddati: Funkcija tiek palaista iekļaujoties P.3. vai P.7. vai P.9 procesā.

Apstrāde: Sadala četru baitu komandu ik pa četriem baitiem (skatīt 3. tabulu) un to pievieno sarakstam.

**3.tabula**

**Četru baitu sadalīšana**

|  |  |  |  |
| --- | --- | --- | --- |
| Sekvence | Tips | Atslēga | Vērtība |
| Komandu izpildes izkārtojums indekss | “SIGNAL” | “turnstile1\_a”, "turnstile1\_b", "turnstile2\_a", "turnstile2\_b",  "button1",  "button2" | Cipars |
| Komandu izpildes izkārtojums indekss | “GET” | "board\_version" | Cipars |

Izvaddati: Tiek pievienota četru baitu komanda sarakstam.

**P.11. ControlUnit mikrokontrolliera komandu saraksta nosūtīšana**

Mērķis:  Ļaut lietotājam palaist komandu no rīka uz ControlUnit.

Ievaddati: Funkcija tiek palaista iekļaujoties P.3. vai P.7. vai P.9 procesā.

Apstrāde: Caur UART nosūta saglabāto četru baitu komandu sarakstu uz ControlUnit mikrokontrolliera.

Izvaddati: Tiek nosūtīta komanda uz ControlUnit mikrokontrolliera un tas to izpilda.

**P.12. Karšu lasītāju funkcionalitātes pārbaude uz ControlUnit**

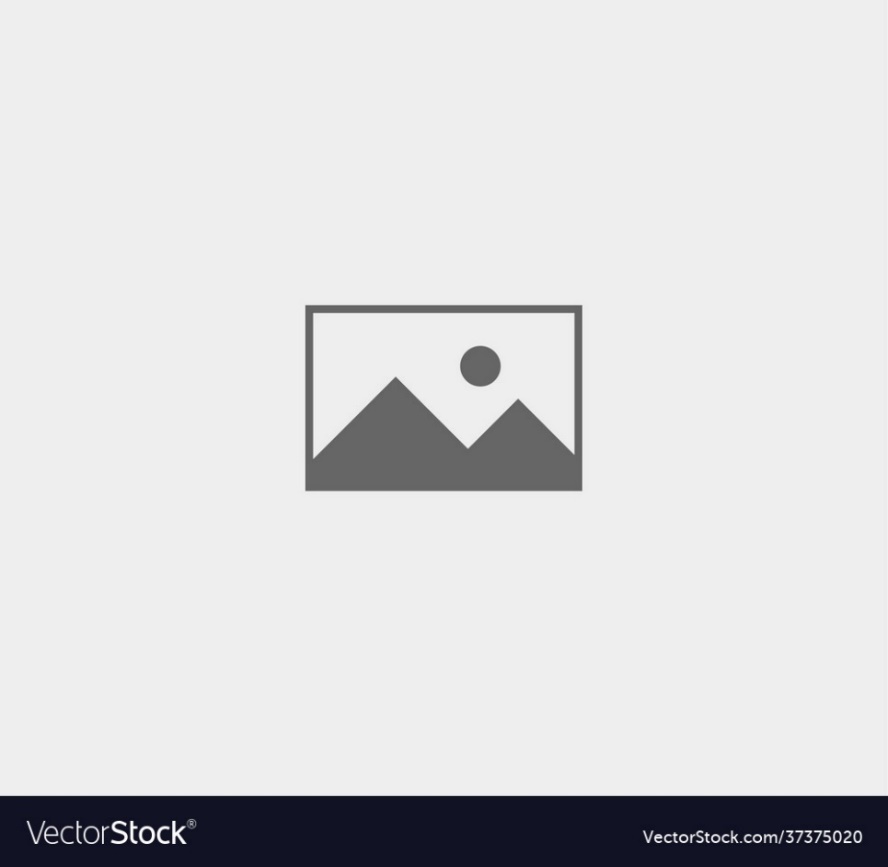
Mērķis: Ļaut lietotājam pārbaudīt karšu lasītāja funkcionalitāti četrās ControlUnit mikročipa adresēs.

Ievaddati:

1. Pabeigt P.9. procesu un jābūt pievienotam ControlUnit mikročipam;
2. Pārbaudīt pirmo ControlUnit mikročipa adresi un noskenēt karti, skatīt 5. nodaļu;
3. Nospiest fizisku pogu rīka labajā pusē (skatīt 4.attēlu);
4. Pārbaudīt otro ControlUnit mikročipa adresi un noskenēt karti, skatīt 5. nodaļu;
5. Nospiest fizisku pogu rīka labajā pusē;
6. Pārbaudīt trešo ControlUnit mikročipa adresi un noskenēt karti, skatīt 5. nodaļu;
7. Nospiest fizisku pogu rīka labajā pusē;
8. Pārbaudīt ceturto ControlUnit mikročipa adresi un noskenēt karti, skatīt 5. nodaļu;
9. Nospiest fizisku pogu rīka labajā pusē.

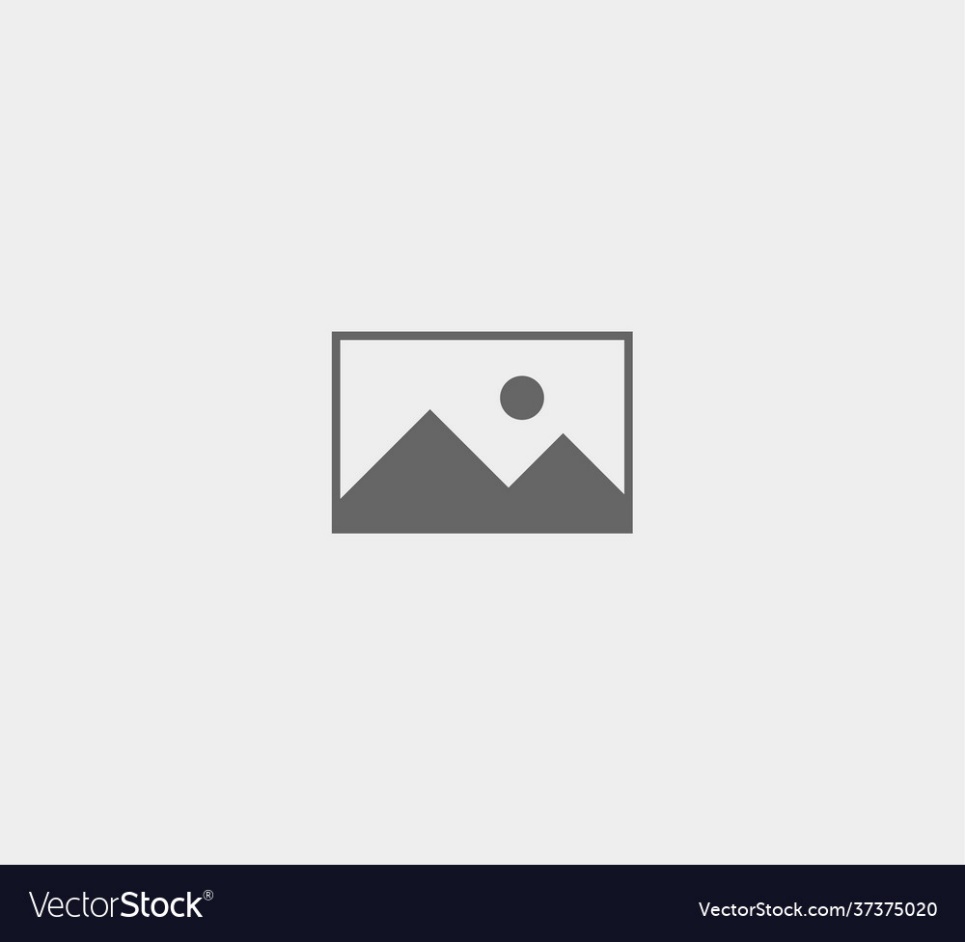
Apstrāde:

Šie četri punkti atkārtojas attiecīgi ControlUnit mikročipa četrām adresēm, jeb šie četri punkti atkārtosies četras reizes:



**12.attēls. Karšu lasītāja kadrs**

1. Uz rīka displeja tiek izvadīts karšu lasītāja kadrs (skatīt 12.attēlu);



**13.attēls. Karšu numura izvade uz displeja**

1. Pēc karšu skenēšanas uz displeja tiek izvadīts karšu numurs, apgriezts kartes numurs, kartes tips ar wiegand protokola palīdzību (skatīt 13. attēlu);
2. Pēc pogas nospiešanas tiek, atkarīgi vai karšu lasītājs ir nolasījis kartes informāciju, tiek saglabāts sarakstā karšu lasītāja adreses indekss, kas liecina vai karšu lasītājs nostrādājis.
3. Tiek izveidots nākošā kāršu lasītāja kadrs uz displeja.

Pēc ceturtās pogas nospiešanas reizes funkcija tiks pārtraukta.

Izvaddati: Uz displeja tiks izvadīta kartes informācija pēc katras karšu lasītāja adreses pārbaudes un tiks saglabāta karšu lasītāja funkcionalitātes stāvoklis sarakstā, attiecīgi vai karšu lasītājs nolasa informāciju.

**P.13. Testa rezultāta izvadīšana uz displeja**

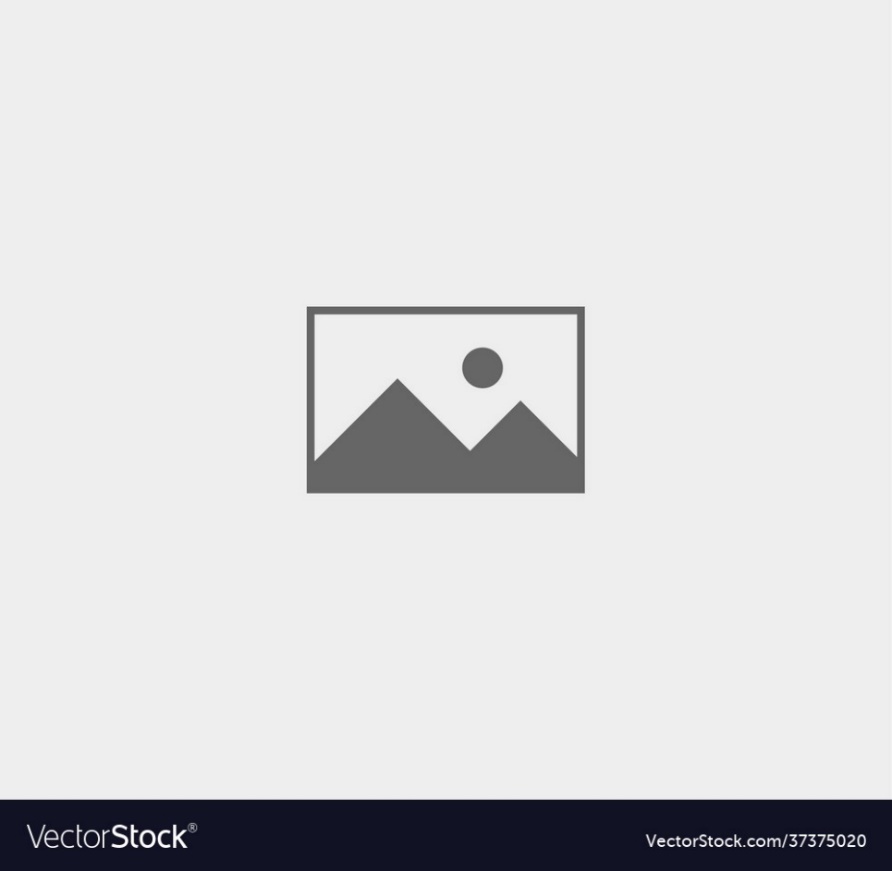
Mērķis: Ļaut lietotājam saprast mikročipa pārbaudes rezultātus.

Ievaddati:

1. Ja ir pievienots ControlUnit mikročips, tad pabeigt P.12. procesu;
2. Ja pievienots TTunit, tad pabeigt P.6. procesu;
3. Nospiest fizisku pogu rīka labajā malā (skatīt 4.attēlu).

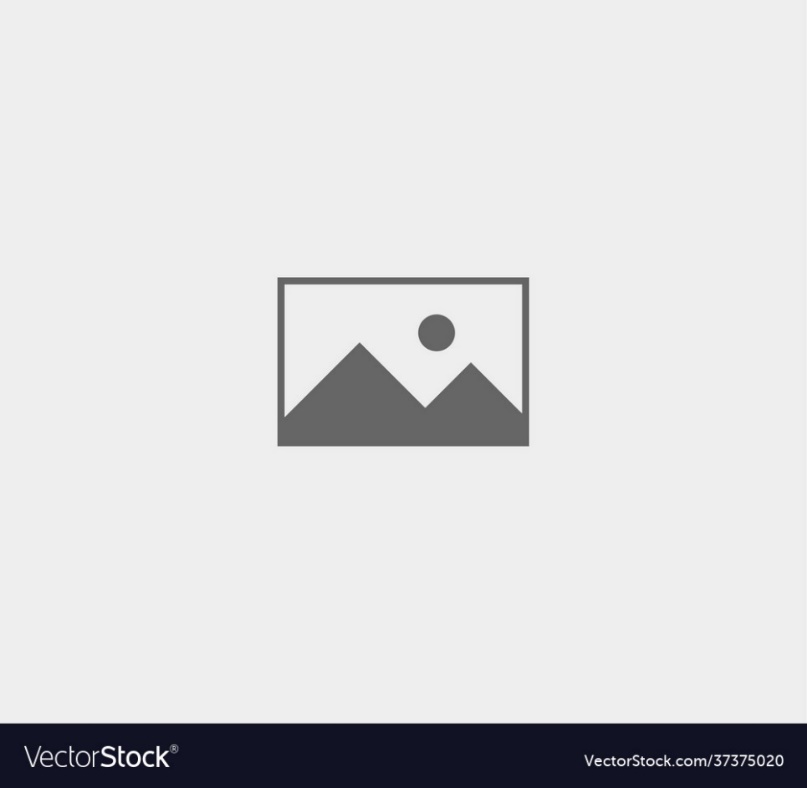
Apstrāde:

1. Pārbauda kurš mikročips ir pievienots pie rīka;
2. Atkarīgi no pievienota mikročipa, ja ControlUnit, tad tiek aktivizēts signalizācijas stāvokļa relejs izmantojot P.8. un P.9. funkcijas, taču, ja TTunit, tad tiek aktivizēts zumers kurš atrodas uz TTunit mikročipa, lai liecinātu mikročipa pārbaudes rezultāta apstiprināšanu;
3. Tiek izveidots jauns kadrs (skatīt P.2.) ar ievaddatiem – “res”;
4. Tiek veikta pārbaude vai mikročipa pārbaude bijusi veiksmīga ar P.11. funkciju;



**14.attēls. Veiksmīga mikročipa pārbaudes rezultāta kadrs**

1. Ja mikročipa pārbaudes rezultāts ir veiksmīgs, tad uz displeja tiek izvadīts zaļā krāsā “The scanner test was successful!” un uz displeja tiek izvadīts visas mikročipa komponenšu pārbaudes rezultāti (skatīt 14.attēlu);



**15.attēls. Neveiksmīga mikročipa pārbaudes rezultāta kadrs**

1. Ja mikročipa pārbaudes rezultāts ir neveiksmīgs, tad uz displeja tiek izvadīts sarkanā krāsā “The scanner test was unsuccessful!” un uz displeja tiek izvadīts visas mikročipa komponenšu pārbaudes rezultāti (skatīt 15.attēlu);
2. Tiek dota iespēja lietotājam atkārtot mikročipa pārbaudi nospiežot fizisku pogu rīka labajā malā.

Izvaddati:

1. Lietotājs tiek informēts vai mikročipa pārbaudes rezultāts ir veiksmīgs vai neveiksmīgs;
2. Lietotājs tiek informēts kuras mikročipa komponentes darbojas korekti, kuras nekorekti;
3. Lietotājam dota iespēja atkārtot mikročipa pārbaudi.

**P.14. Mikročipa testa rezultāta pārbaudes funkcija**

Mērķis: Pārbaudīt vai testa process ir veiksmīgs vai neveiksmīgs.

Ievaddati: Progresēt cauri no P.1. līdz P.11. un iekš P.11 šī funkcija tiek palaista.

Apstrāde:

1. Pārbauda kāds mikročips ir pievienots pie rīka;
2. Pārbauda vai testa procesa laikā visi pievienoti dati sarakstā liecina ka katrs tests ir nostrādājis.

Izvaddati: Funkcija atgriež vērtību, kas nosaka vai tests ir veiksmīgs vai neveiksmīgs

**P.15. Mikročipa pārbaudes atkārtošanas sagatavošanas funkcija**

Mērķis: Attīrīt visus datus par iepriekšējo mikročipa pārbaudi.

Ievaddati: Nospiest fizisku pogu rīka labajā malā (skatīt 4.attēlu) pēc P.11. izpildes.

Apstrāde: Attīra visas sarakstes un mainīgos, kas iepriekš glabāja datus par iepriekšējo mikročipa pārbaudi.

Izvaddati: Visas iepriekšējās mikročipa pārbaudes sarakstes un mainīgie tiek attīrīti.

## Sistēmas nefunkcionālās prasības

1. Testēšanas rīka izstrādes procesa dokumentācijai ir jābūt noformētai un izstrādātai atbilstoši Latvijas Valsts standartam LVS 68:1996;
2. Saskarnei jābūt ērtai un ergonomiskai, lai ātrā laika posmā lietotājs spētu pārbaudīt mikročipu;
3. Testēšanas rīka displeja dominējošās krāsas – zaļa un balta;
4. Testēšanas rīka displeja izmantotā valoda – Angļu valoda ar dažādiem izmantotiem profesionālismiem;
5. Testēšanas rīkam jābūt izstrādātam iekš plastmasas ietvara;
6. Produktam jābūt izstrādātam Python programmēšanas valodā un rīkam jātiek palaistam uz Micropython programmaparatūru.

## Gala lietotāja raksturiezīmes

Testēšanas rīka galvenā funkcija ir pārbaudīt mikročipus TTunit un ControlUnit. Tā kā mikročipus veido uzņēmums “SIA InPass”, tos nav iespējams iegādāties tirgū un vienīgie gala lietotāji būs uzņēmuma darbinieki. Var secināt, ka gala lietotājs būs tehniskais speciālists ar vajadzību pārbaudīt mikročipus.

Gadījumā, ja pasūtītājs būs nolēmis, ka testēšanas rīks tiek tirgots plašajā tirgū un testēšanas rīks tiek papildināts ar vēl vairākām funkcionalitātēm, tad testēšanas rīka izstrādes procesa dokumentācija jāpapildina.

# Izstrādes līdzekļu, rīku apraksts un izvēles pamatojums

Šajā nodaļā aprakstīti produkta izstrādes līdzekļi, dažādie izmantotie rīki, kā arī to izvēles pamatojumu.

## Izvēlēto risinājuma līdzekļu un valodu apraksts

## Izstrādes programmatūras

Izstrādes laikā tika izmantotas dažādas programmatūras. Katras programmatūras unikālās īpašības ļāva izstrādātājam ātri un efektīvi atrisināt dažādus izstrādes strupceļus. Šīs ir izmantotās izstrādes programmatūras:

**Thonny** - integrētā izstrādes vide (IDE), kas ir paredzēta galvenokārt Python programmām. Šī IDE ir vienkārša, intuitīva un viegli lietojama, tādēļ lielākoties produkta izstrādei tika izmantota šī izstrādes vide. Šeit ir programmatūras izvēles pamatojuma punkti:

* Vienkārša un tīra lietotāja saskarne, kas darba procesā izstrādātajam sniedza vieglu sadarbību ar rīku un darbstaciju.
* Iebūvēts interpretatora atbalsts ir liels iemesls kādēļ tika izmantota tieši šī programmatūra, tas ļāva izstrādātājam vieglu maiņu starp micropython programmaparatūrām.
* Kļūdu atklāšana un labošana nodrošināja izstrādātājam efektīvi un ātri izprast izstrādes strupceļus, jo programmatūra sniedza atšifrētus rīka kļūmes.
* Programmatūras atbalsts vairākām operētājsistēmām deva izstrādātājam ērtu programmatūras izmantošanu ar Windows un Linux operētājsistēmām, jo izstrādes laikā izstrādātājam strupceļu gadījumos bija nepieciešams izmantot vairākas operētājsistēmas.

**PuTTY** - bezmaksas un atvērtā koda termināla emulators un tālvadības pieteikums, kas galvenokārt tiek izmantots SSH, Telnet un seriālo savienojumu veidošanai un pārvaldīšanai ar citiem datoriem vai tīkla ierīcēm. Šeit ir programmatūras izvēles pamatojuma punkti:

* Seriālais savienojums ar rīku un darbstaciju izstrādātājam atviegloja informācijas izvadi no rīka testēšanas etapa.
* Portabilitāte un viegla instalācija starp rīka USB savienojuma un darbstacijas sniedza izstrādātājam ērti izveidot seriālo savienojumu bez liekiem strupceļiem.

**Arduino** **IDE** - integrētā izstrādes vide, kas ir paredzēta Arduino mikrokontrollieru programmēšanai un projektu izstrādei. Izstrādātāja pamatojums programmatūras izmantošanai bija strupceļa gadījumā izmantojot Arduino mikrokontrollieri to atrisināšanai.

**Git** - izplatīts versiju kontroles sistēma, kas izmantojama programmēšanas projektu vadīšanai un kodu sadarbībai starp dažādiem izstrādātājiem. Tā ir atvērtā koda un bezmaksas, un tās izmantošana ir kļuvusi par standarta praksi programmēšanas nozarē. Šeit ir programmatūras izvēles pamatojuma punkti:

* Distribuēta versiju kontrole izstrādātājam ļāva versionēt produkta izstrādes gaitu un strupceļa gadījumos atgriezt atpakaļ iepriekšējo produkta versiju.
* Atzaru un apvienošanas modelis izstrādātājam atviegloja produkta koda daļu apvienošanu ar dažādiem izstrādes zariem.
* Versiju viegla pārvaldīšana izstrādātājam sniedza vispārējo pārlūku starp iepriekšējo produkta versiju izmaiņām.

## Izstrādes darbstacija

Izstrādes darbstacija nodrošina visu nepieciešamo produkta izstrādē. Šeit minētās ierīces ir izstrādātāja darbstacijas komplekts :

**Lenovo ThinkPad X230** – portatīvais dators, kuru izstrādātājs izmantoja gatavā produkta izstrādei. Komplektācijā arī portatīvā datora lādētājs. Ierīci nodrošināja prakses vieta projekta realizācijai.

**Logitech M90 datora pele** - ierīce, kas ļāva izstrādātājam daudz ērtāk un ergonomiskāk veikt programmatūras izstrādi. Ierīci nodrošināja prakses vieta.

**Arduino Nano ESP32** – mikrokontrolliera ierīce ar kuru ir iespējams veidot dažādus programmēšanas projektus. Izstrādātājs izmantoja šo ierīci, lai atrisinātu izstrādes ceļā izveidojušo strupceļu saistībā ar vairāku mikrokontrollieru savstarpējo sazināšanos.

**USB-A uz Micro-USB kabelis** – galvenā izstrādātāja aparatūra, kas izveido savienojumu starp portatīvo datoru un rīku. Kabelis arī darbojas kā savienotājs, nodrošinot strāvu rīkam un TTunit mikročipam.

## Produkta komponentes un ierīces

Produkta komponentes un ierīces ir vispārējais aparatūras komplekts, kas izveido pašu rīku. Šis ir aparatūras komplekts :

**Rīka mikrokontrolliera ierīce** – izstrādātāja prakses vietas “SIA InPass” pašu izveidotā ierīce ar RP2040 mikrokontrolliera komponenti. Ierīcē tiek uzstādīta programmatūra, radot gatavu produktu (skatīt 16.attēlu).



**16.attēls. Rīka mikrokontrolliera ierīce**

**ILI9341 2.8 collu displejs** – komponente, kas nodrošina rīka saskarni un mikročipu pārbaudes informācijas izvadi. Komponentes priekšrocība ir tās slaidais izmērs un kvalitāte, kura vispiemērotāk atbilst produkta prasībām.

## Programmatūras programmēšanas valodas un bibliotēkas

Precīzai programmatūras izpildei un funkcionalitātei liela būtība ir tās izmantotai programmēšanas valodai un bibliotēkām, kas nodrošina funkcionālu un ātru programmatūras izpildi. Šeit tiek minētas izmantotās izstrādes programmatūras valodas un bibliotēkas :

**Python** – programmēšanas valoda ar kuru izstrādātājs izgatavoja rīka programmatūru. Programmēšanas valodas izvēle ir pasūtītāja prasība, izrietot no šīs programmēšanas valodas izvēles ir nosacījums izmantot Python programmēšanas valodas balstīto Micropython programmaparatūru.

**Micropython** – programmaparatūra ar kuras palīdzību tiek palaista izstrādātā programmatūra. Pati programmaparatūra tiek uzstādīta uz rīka mikrokontrolliera. Programmaparatūra tiek lietota balstoties uz pasūtītāja prasībām.

**Wiegand** – protokola bibliotēka, kuru izmanto karšu lasītāja pārbaudes rīka darbības procesā. Bibliotēka tiek uzstādīta rīka mikrokontrolliera ierīcē un to nodrošina prakses vietas izveidotā bibliotēka. Šī bibliotēka ir vispiemērotākā programmatūrai, jo tiek pārbaudītas “SIA InPass” izmantotie karšu lasītāji.

**EEPROM** – bibliotēka, kas ar tās funkcijām ātri un efektīvi ļauj izmantot EEPROM komponenti. Bibliotēkas bezmaksas piekļuves izmantošana un efektīvās iebūvētās funkcijas attaisno tās pielietojumu.

**ILI934xnew –** bibliotēka ar kuru tiek izveidota displeja saskarne. Bibliotēkas iebūvētās funkcijas izstrādātājs veiksmīgi pielietoja, izstrādājot saskarni, kura nodrošina ātru renderēšanu.

**Fonta izveides bibliotēkas** – bibliotēku kopums ar kuras palīdzību tiek renderēts teksts uz displeja un efektīvi tiek mainīti šie teksta fonta izmēri, krāsas, novietojums.

**Machine bibliotēka** – dažādu funkciju kopums, kas iebūvēts Micropython programmaparatūrā, ar kuras palīdzību programmatūra spēj izmantot digitālās tapas uz rīka mikrokontrolliera ierīces, radot seriālo komunikāciju starp citiem mikrokontrollieriem. Bibliotēkas efektivitāte padara to par vispiemērotāko programmatūras konkrētajām vajadzībām.

## Iespējamo risinājuma līdzekļu un valodu apraksts

Programmatūras izstrāde ir plašs un daudzveidīgs process, kurā iespējams izmantot vairākus līdzekļus, komponentes, ierīces, programmēšanas valodas un bibliotēkas. Tomēr, projekta izstrādes iespējas ir ierobežotas no pasūtītāja noteiktajām prasībām. Potenciāli var tikt izmantotas citas efektīvākas bibliotēkas, kas uzlabotu programmatūras kvalitāti un veiktspēju. Vienlaikus, var uzstādīt jaunākās Micropython programaparatūras versijas mikrokontrollierī, kas nodrošinātu ātrāku un efektīvāku veiktspēju, kā arī izmantot citus programmēšanas algoritmus, paņēmienus un datu struktūras, kas atļautu programmatūrai būt efektīvākai.

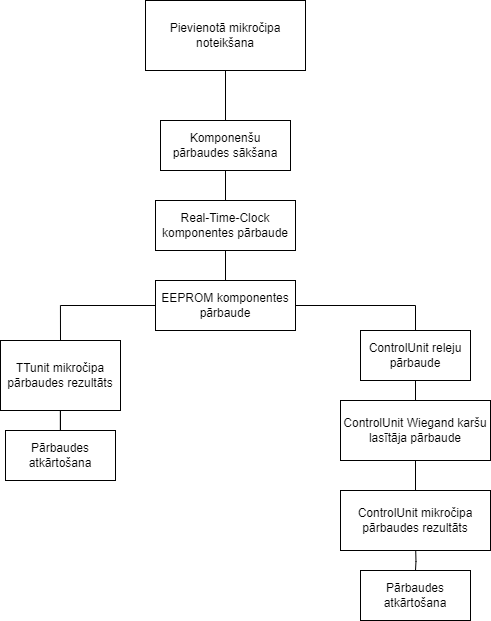
Pasūtītāja prasības neierobežo darbstaciju, tādēļ, izstrādes procesā izstrādātājs potenciāli varēja izmantot alternatīvas darbstacijas ierīces un izstrādes vides, piemēram, jaunākus un efektīvākus programmēšanas rīkus vai modernāku datoru aprīkojumu. Šāda izvēle būtu rezultējusi uz īsāku izstrādes laiku un ērtāku, efektīvāku izstrādes procesu.

# Sistēmas modelēšana un projektēšana

Šajā nodaļā aprakstīta sistēmas modelēšana un projektēšana, precīzāk, sistēmas struktūras modeļi, klašu diagrammas un aktivitāšu diagramma.

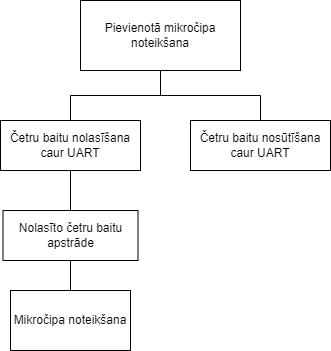
## Sistēmas struktūras modelis

Apakšnodaļā vizualizēts vispārējais sistēmas struktūras modelis un katrs redzamais bloka elements (skatīt 17. attēlu) tiek sīkāk sadalīts citos bloka elementos.



**17. attēls. Sistēmas struktūra**

## Pievienota mikročipa noteikšana



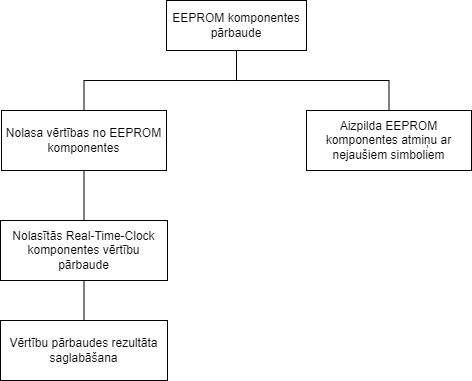
**18. attēls. Pievienota mikročipa noteikšana**

## Real-Time-Clock komponentes pārbaude



**19. attēls. Real-Time-Clock komponentes pārbaude**

## EEPROM komponentes pārbaude



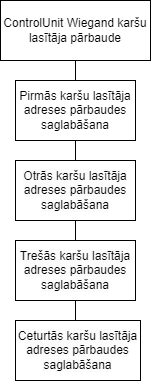
**20.attēls. EEPROM komponentes pārbaude**

## ControlUnit releju pārbaude



**21. attēls. ControlUnit releju pārbaude**

## ControlUnit Wiegand karšu lasītāja pārbaude



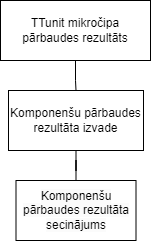
**22. attēls. ControlUnit Wiegand karšu lasītāja pārbaude**

## ControlUnit mikročipa pārbaudes rezultāts



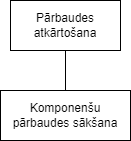
**23. attēls. ControlUnit mikročipa pārbaudes rezultāts**

## TTunit mikročipa pārbaudes rezultāts



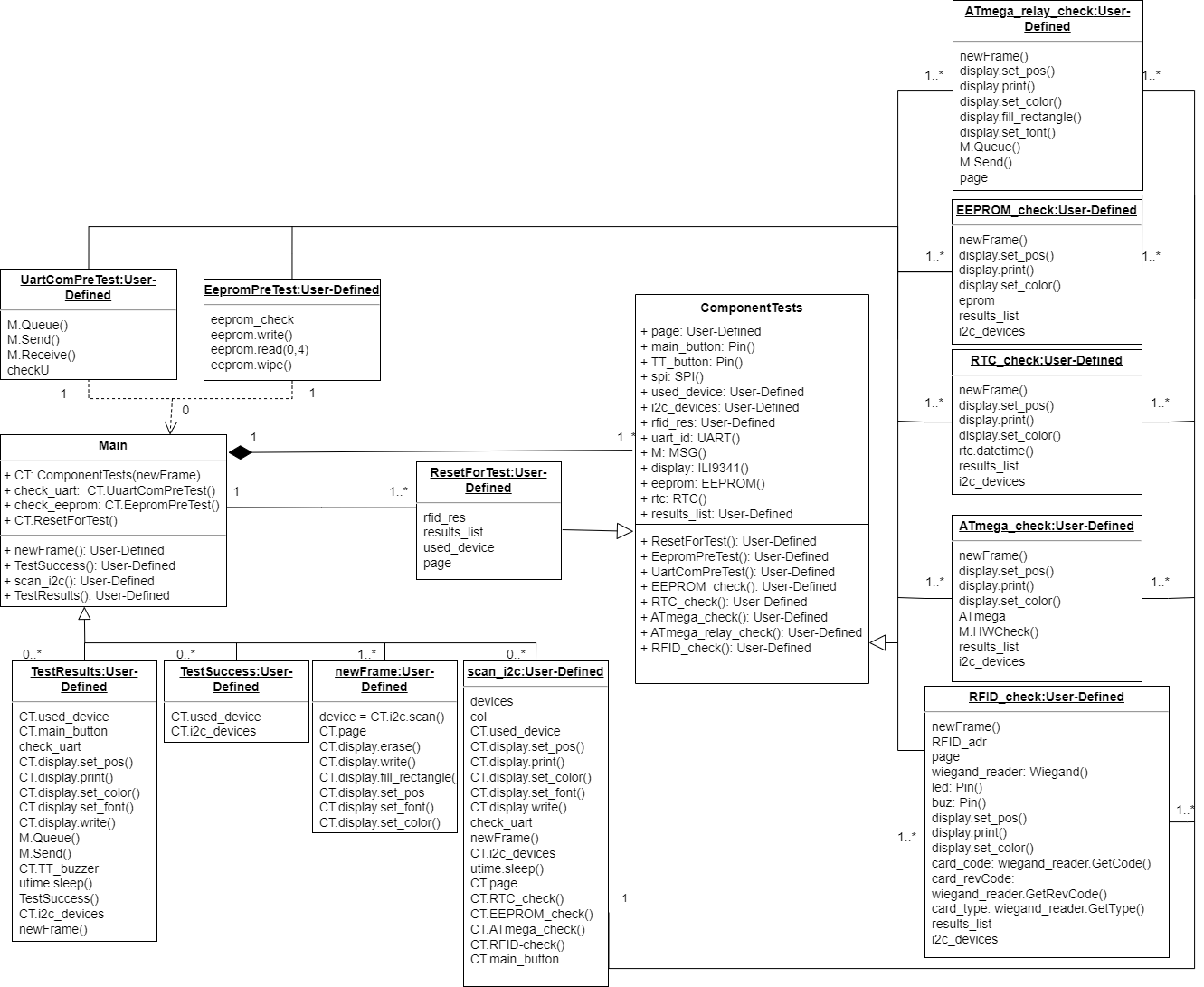
**24. attēls. TTunit mikročipa pārbaudes rezultāts**

## Pārbaudes atkārtošana

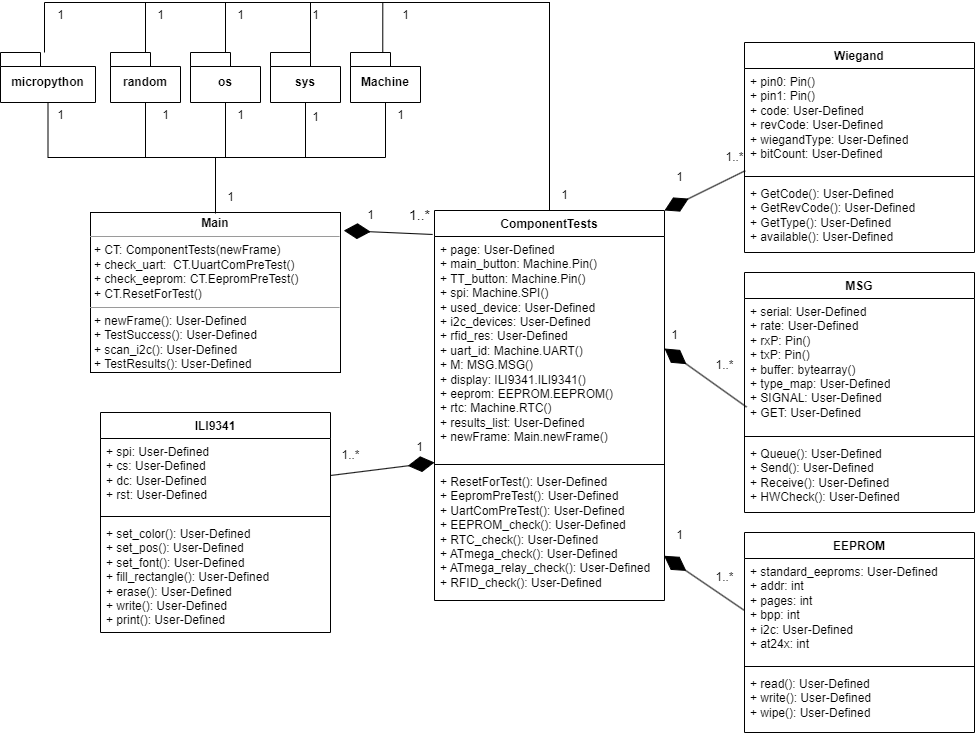


**25. attēls. Pārbaudes atkārtošana**

## Klašu diagramma / ER diagramma



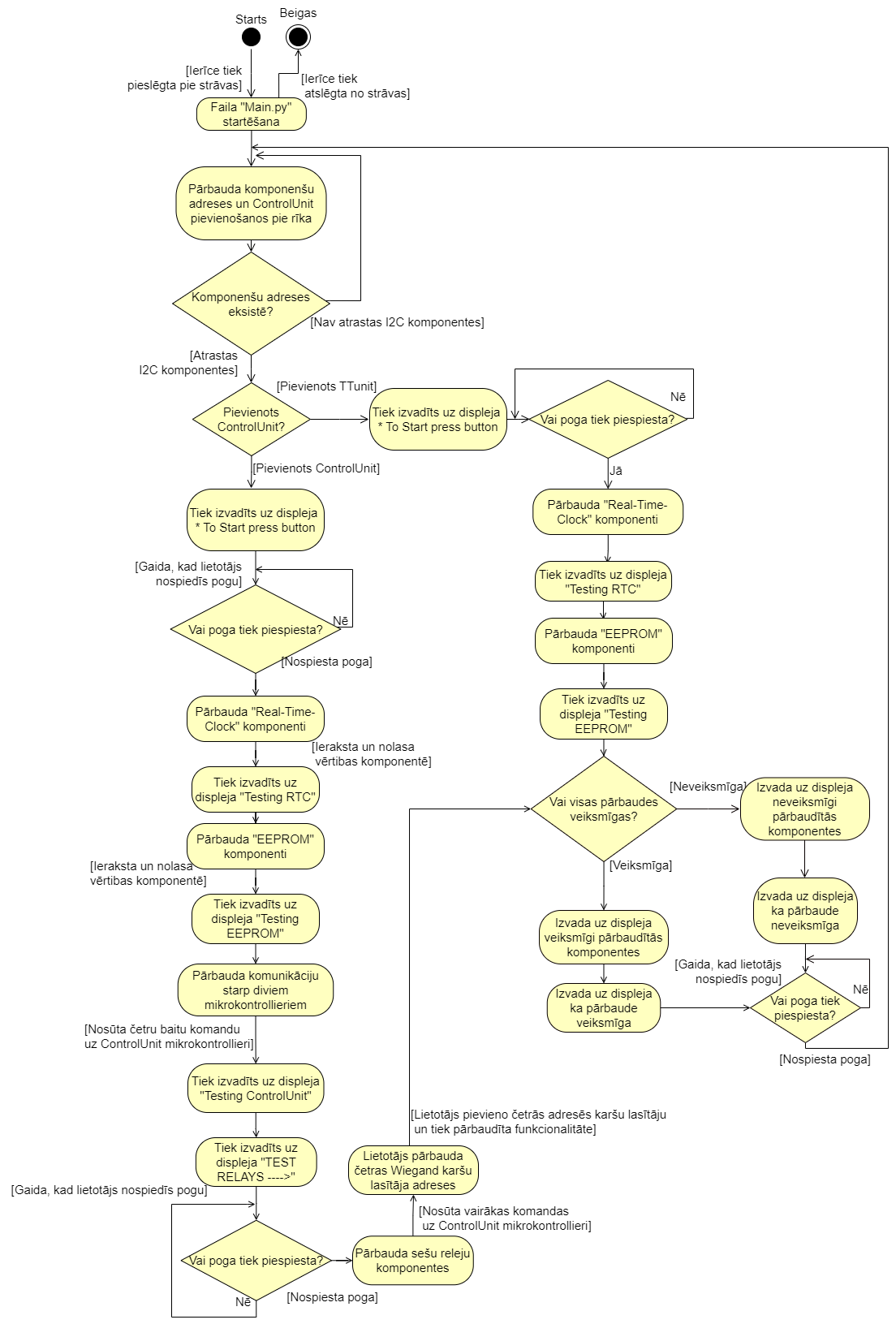
**26. attēls. Galveno programmatūras klašu diagramma**

****

**27. attēls. Bibliotēku un palīg-klašu klašu diagramma**

## Funkcionālais un dinamiskais sistēmas modelis

## Stāvokļu diagramma



**28. attēls. Stāvokļu diagramma**

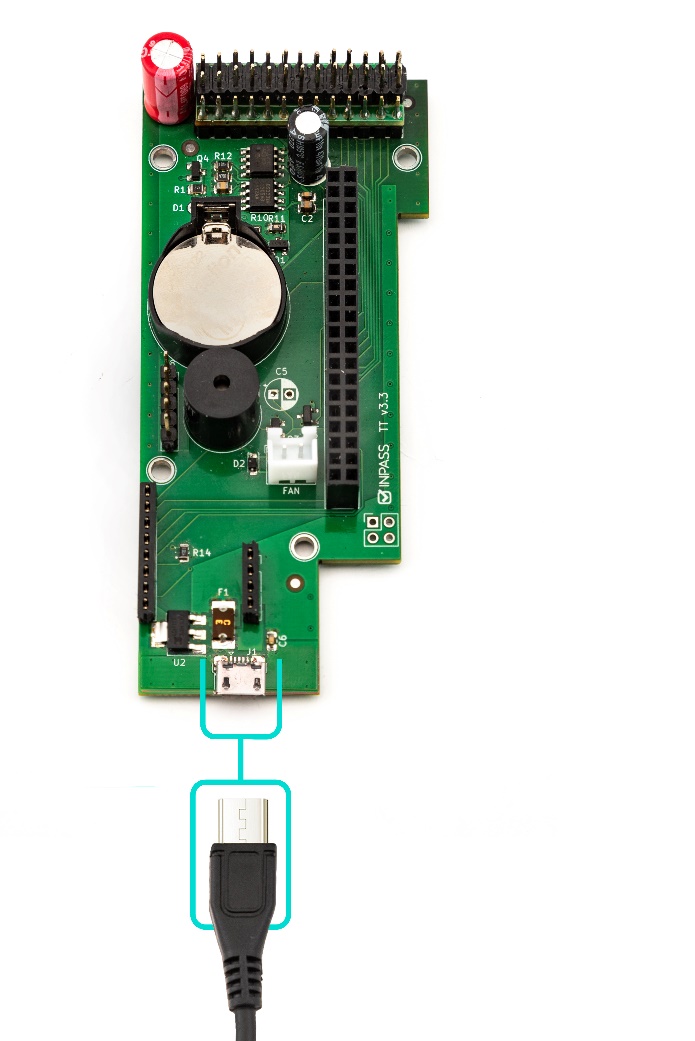
# Lietotāju ceļvedis

Šajā nodaļā detalizēti aprakstīts ceļvedis, kurš lietotājam nodrošinās nepieciešamos norādījumus un procedūras, lai veiksmīgi izmantotu produktu un veiktu pareizu TTunit un ControlUnit mikročipu pārbaudi.

## TTunit mikročipa pārbaudes sagataves process

TTunit mikročipa pārbaudes sagataves process sastāv no sekojošiem soļiem:

1. Pieslēgt pārbaudei vēlamo TTunit mikročipu pie strāvas ar micro-USB vadu (skatīt 29. attēlu).



**29. attēls. TTunit pieslēgšana pie strāvas**

1. Pieslēgt rīku pie TTunit mikročipa 2x20 ligzdas (skatīt 30. attēlu).



**30. attēls. Rīka pieslēgšana pie TTunit**

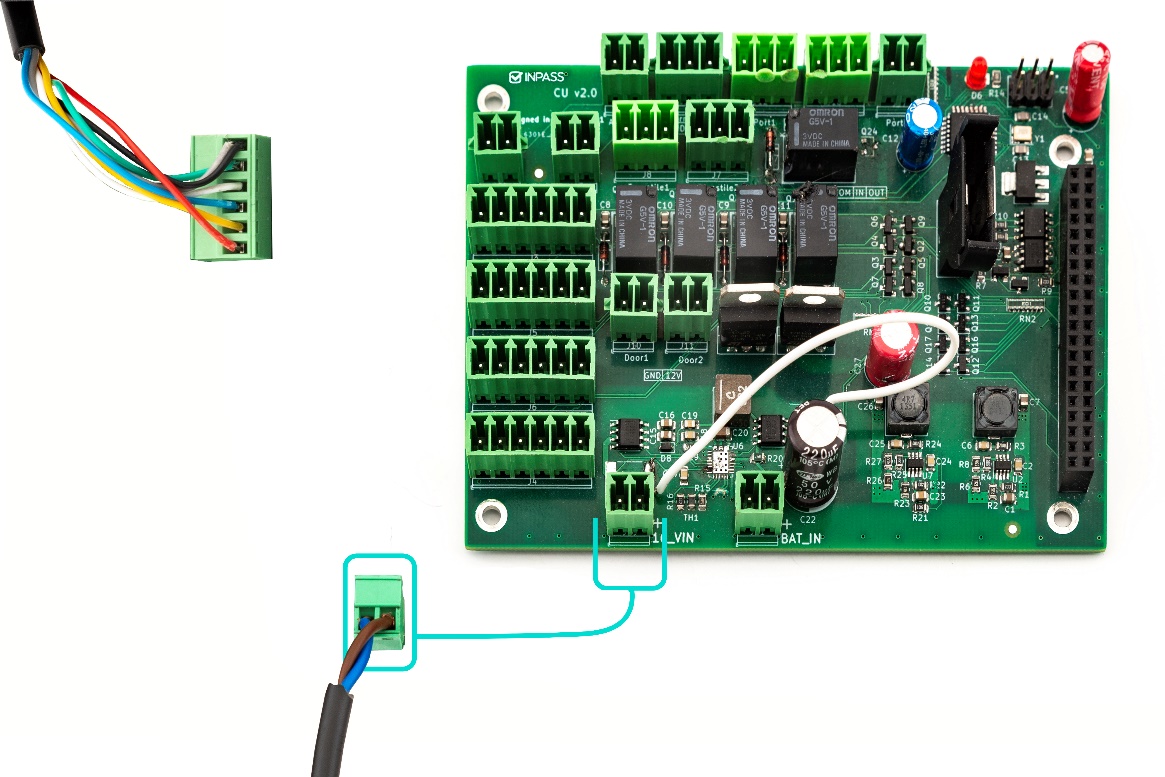
1. Gaidīt līdz rīka displejs ieslēdzas un tiek izvadīts sākuma kadrs (skatīt 5.attēlu).

Lietotājam veicot šos trīs soļus būs iespēja pārbaudīt pievienoto TTunit mikročipu un iegūt mikročipa funkcionalitātes stāvokli.

## ControlUnit mikročipa pārbaudes sagataves process

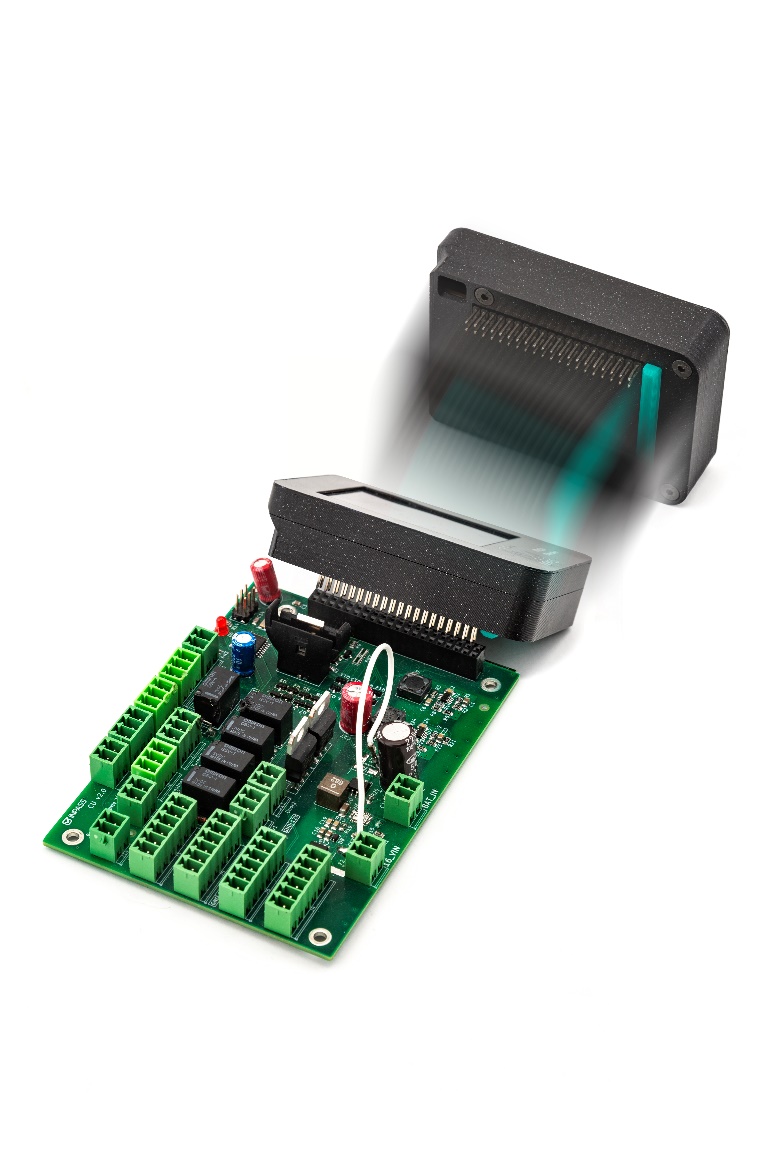
ControlUnit mikročipa pārbaudes sagataves process sastāv no sekojošiem soļiem:

1. Pieslēgt pārbaudei vēlamo ControlUnit mikročipu pie strāvas ar 16V vadu pie 16\_VIN savienojuma ligzdas (skatīt 31. attēlu).



**31. attēls. ControlUnit pieslēgšana pie strāvas**

1. Pievienot rīku pie ControlUnit mikročipa 2x20 ligzdas (skatīt 32. attēlu).



**32. attēls. Rīkā pievienošana pie ControlUnit**

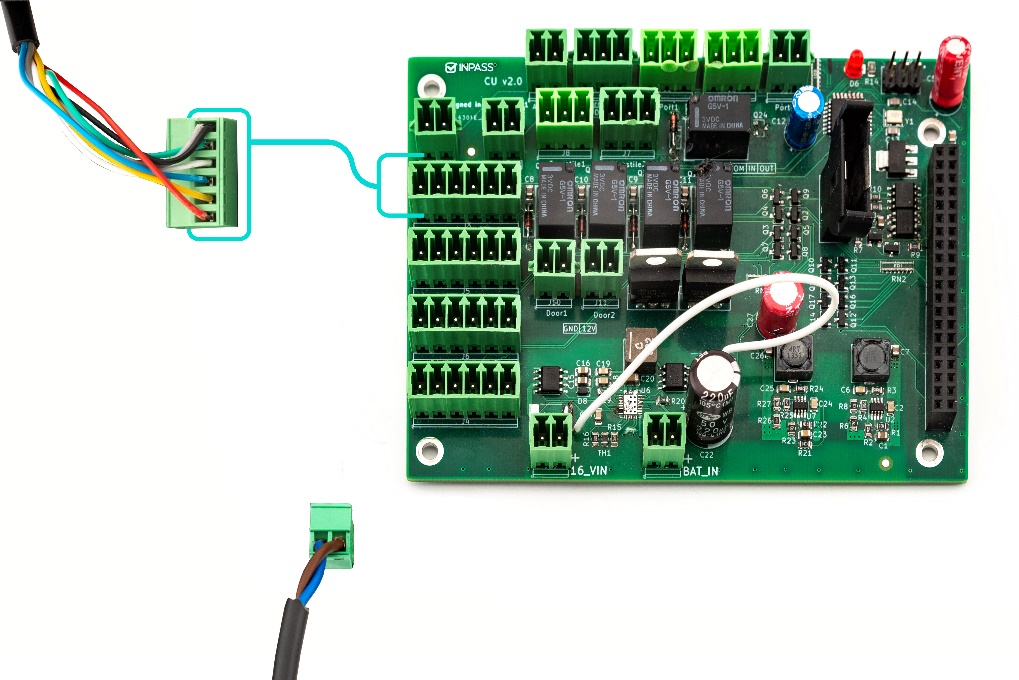
1. Gaidīt līdz displejs ieslēdzas un tiek izvadīts sākuma kadrs ar ControlUnit.

Veicot šos trīs soļus būs iespēja pārbaudīt pievienotā ControlUnit funkcionalitātes stāvokli un citu komponenšu stāvokli.

## ControlUnit mikročipa Wiegand karšu lasītāja pārbaudes process

Pievienojot rīku pie ControlUnit un progresējot cauri mikročipa pārbaudei līdz desmitai pārbaudei, jeb “10. test”, lietotājam ir iespēja pārbaudīt četras ControlUnit Wiegand protokola karšu lasītāju adreses. Lietotājam jāveic karšu lasītāja pievienošana attiecīgi pēc izvadītās informācijas uz displeja (skatīt 12. attēlu). Lietotājam jāseko sekojošiem soļiem:

1. Gadījuma, ja uz displeja tiek izvadīts “1. Wiegand device address testing”, tad lietotājam jāpievieno Wiegand karšu lasītājs pie pirmās ControlUnit adreses (skatīt 33. attēlu).



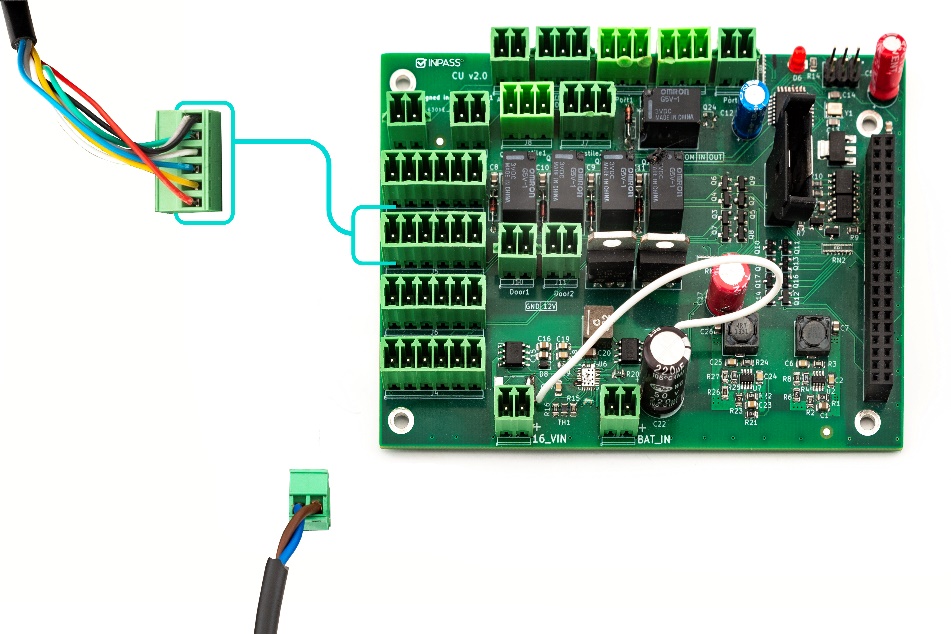
**33. attēls. Karšu lasītāja pievienošana pie pirmās ControlUnit adreses**

1. Lietotājam pievienojot karšu lasītāju ir jānoskenē RFID kartiņa uzliekot to uz karšu lasītāja (skatīt 34. attēlu).



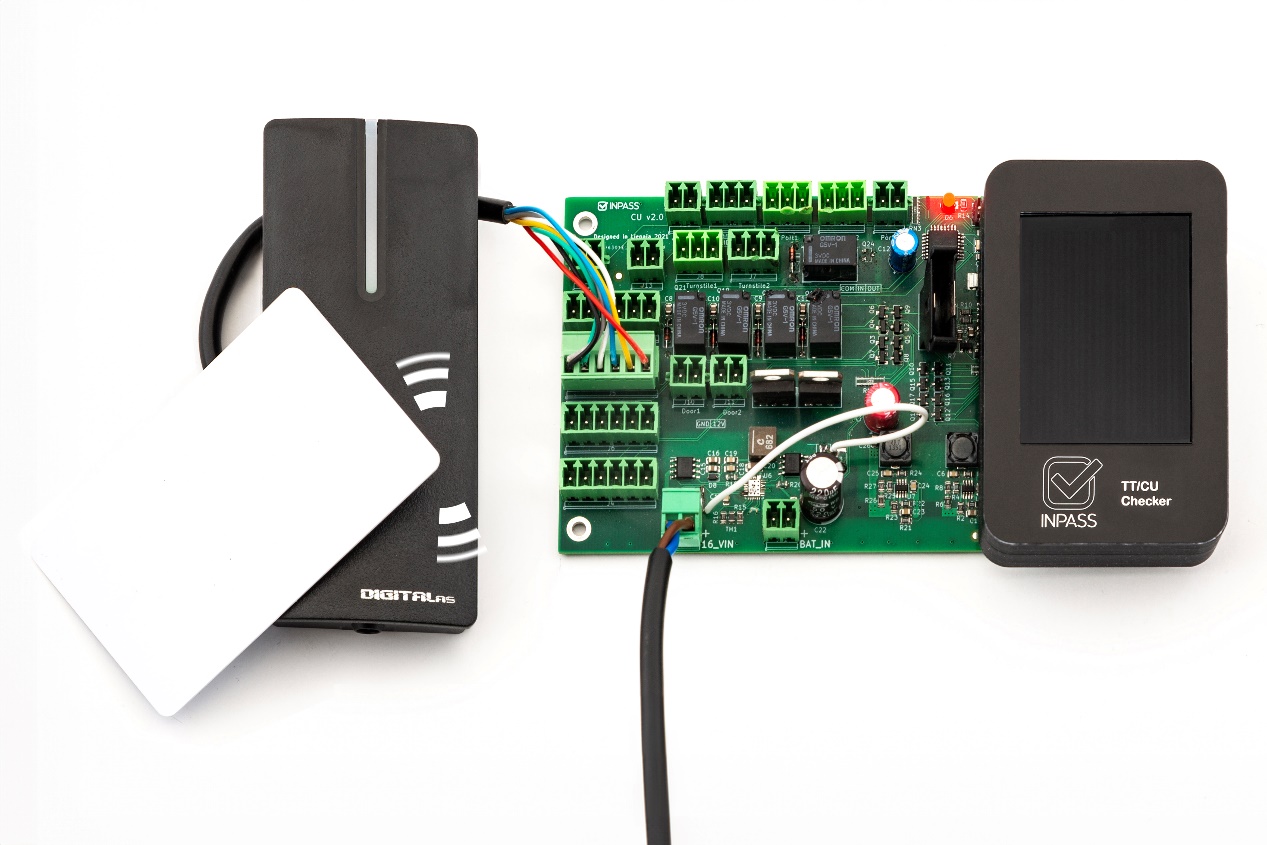
**34. attēls. RFID kartiņas noskenēšana uz pirmās adreses**

1. Noskenējot kartiņu tiek izvadīts kartiņas numurs uz displeja (skatīt 13. attēlu).
2. Gadījuma, ja uz displeja tiek izvadīts “2. Wiegand device address testing”, tad lietotājam jāpievieno Wiegand karšu lasītājs pie otrās ControlUnit adreses (skatīt 35. attēlu).



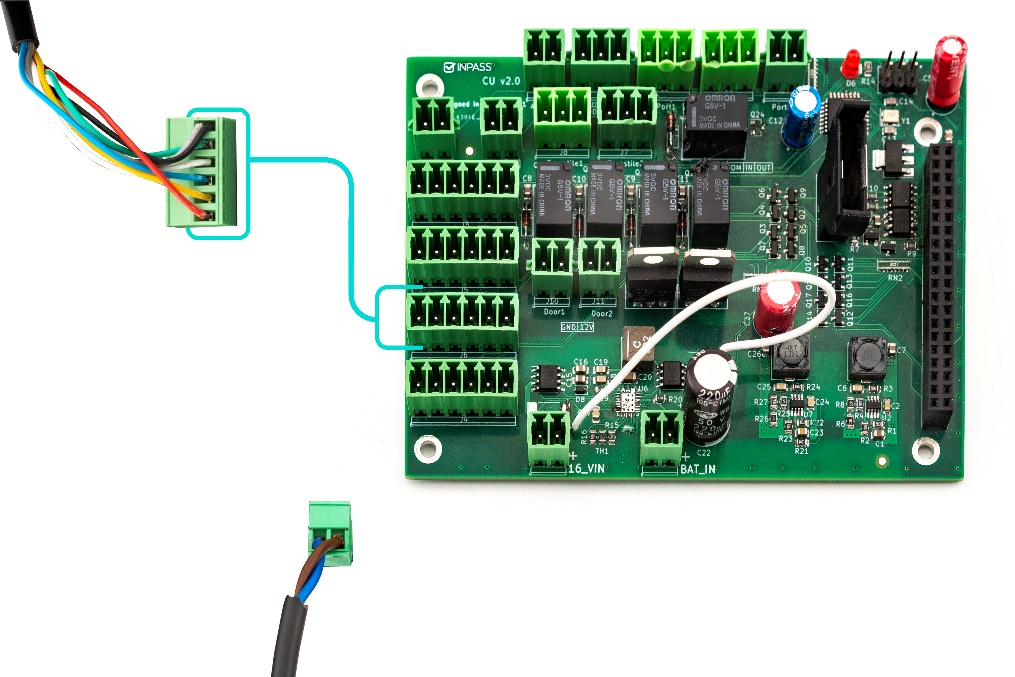
**35. attēls. Karšu lasītāja pievienošana pie otrās ControlUnit adreses**

1. Lietotājam pievienojot karšu lasītāju ir jānoskenē RFID kartiņa uzliekot to uz karšu lasītāja (skatīt 36. attēlu).



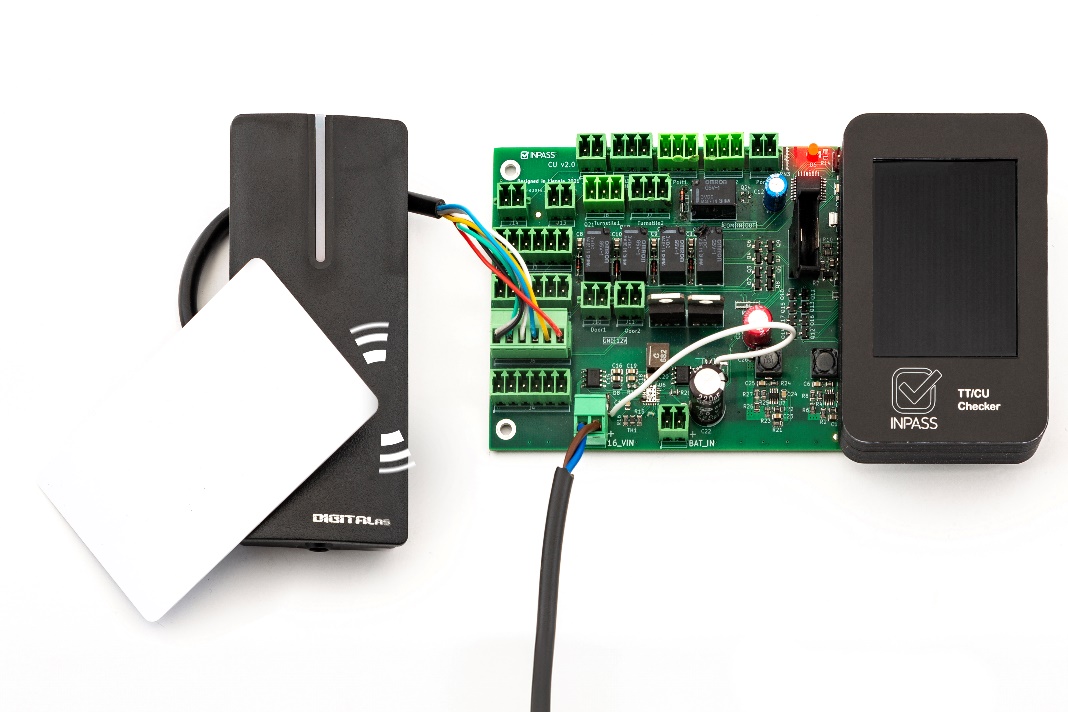
**36. attēls. RFID kartiņas noskenēšana uz otrās adreses**

1. Noskenējot kartiņu tiek izvadīts kartiņas numurs uz displeja (skatīt 13. attēlu).
2. Gadījuma, ja uz displeja tiek izvadīts “3. Wiegand device address testing”, tad lietotājam jāpievieno Wiegand karšu lasītājs pie trešas ControlUnit adreses (skatīt 37. attēlu).



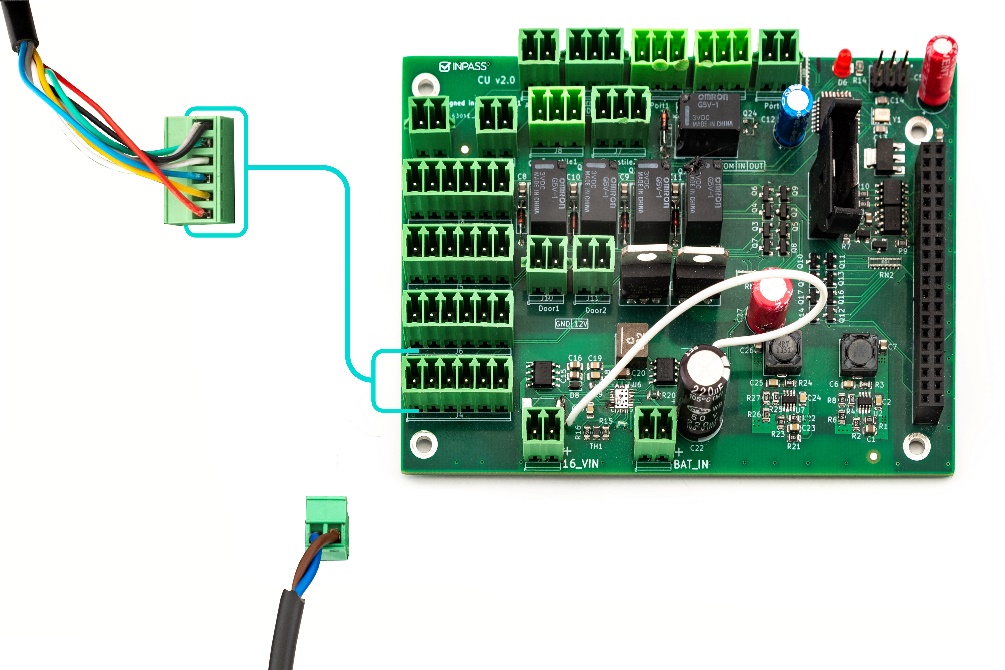
**37. attēls. Karšu lasītāja pievienošana pie trešas ControlUnit adreses**

1. Lietotājam pievienojot karšu lasītāju ir jānoskenē RFID kartiņa uzliekot to uz karšu lasītāja (skatīt 38. attēlu).



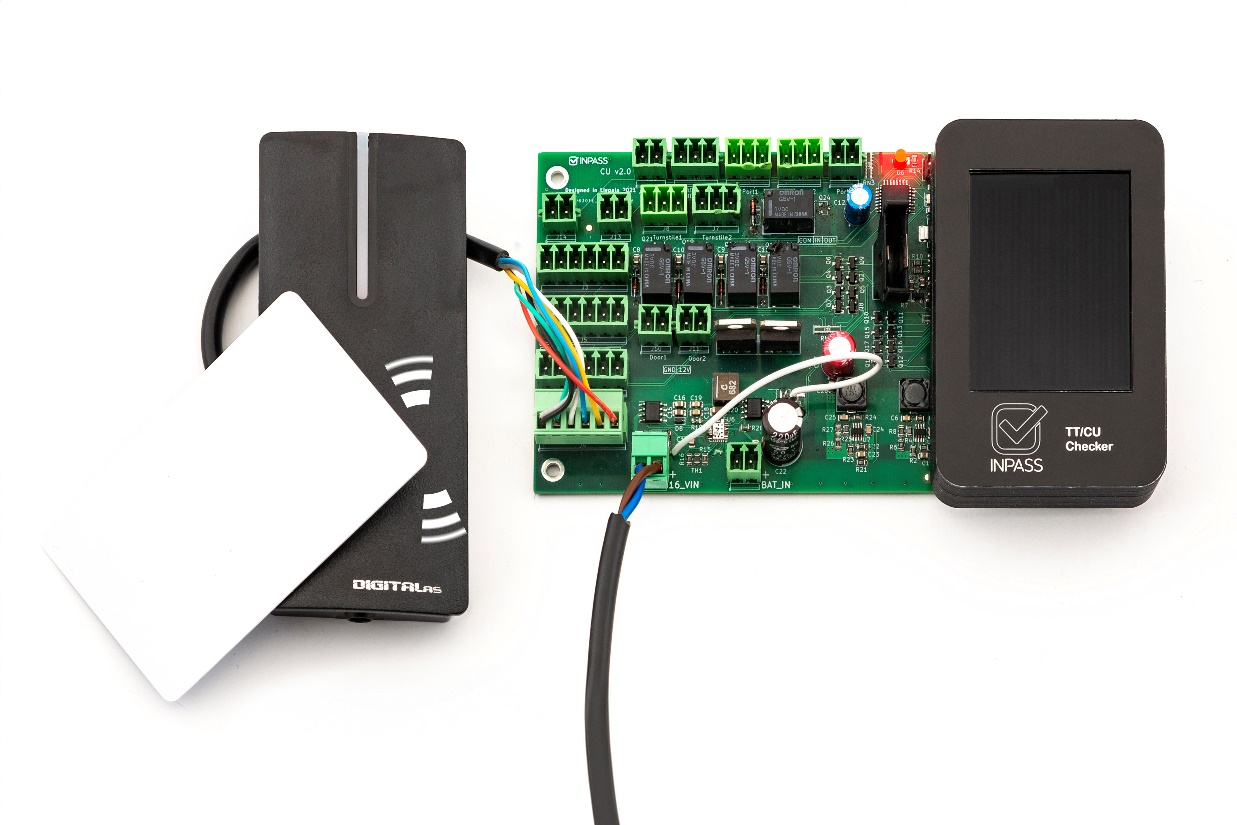
**38. attēls. RFID kartiņas noskenēšana uz trešās adreses**

1. Noskenējot kartiņu tiek izvadīts kartiņas numurs uz displeja (skatīt 13. attēlu).
2. Gadījuma, ja uz displeja tiek izvadīts “4. Wiegand device address testing”, tad lietotājam jāpievieno Wiegand karšu lasītājs pie ceturtās ControlUnit adreses (skatīt 39. attēlu).



**39. attēls. Karšu lasītāja pievienošana pie ceturtās ControlUnit adreses**

1. Lietotājam pievienojot karšu lasītāju ir jānoskenē RFID kartiņa uzliekot to uz karšu lasītāja (skatīt 40. attēlu).



**40. attēls. RFID kartiņas noskenēšana uz ceturtās adreses**

1. Noskenējot kartiņu tiek izvadīts kartiņas numurs uz displeja (skatīt 13. attēlu).

## Saskarnes pamācība

Saskarne ir būtisks aspekts programmatūras lietojamībā, kur uzsvars tiek likts uz fiziskās pogas izmantošanu dažādos darbības scenārijos. Lietotājiem tiek sniegti precīzi norādījumi par to, kā un kad izmantot fizisko pogu, lai nodrošinātu ne tikai pareizu programmatūras darbību, bet arī optimālu lietošanas pieredzi. Šie norādījumi ietver gan praktiskus piemērus, gan nākošo procesu aprakstus, kas palīdz lietotājiem saprast nākamo soli un veikt attiecīgās darbības bez problēmām vai neskaidrībām. Tajā pašā laikā veido vienkāršu un intuitīvu lietojamības pieredzi lietotājam.

Scenārijos, kad uz displeja labajā augšējā stūrī tiek parādīts signāls “PRESS---->”, “BUZZ---->” vai “RESTART-->”, līdzīgi redzams (5. attēlā), lietotājam ir jāveic fiziskās pogas nospiešana (skatīt 4. attēlu), lai virzītos uz priekšu un turpinātu programmatūras darbību. Šāda saskarne starp lietotāju un programmatūru nodrošina nepieciešamo līdzsvaru starp digitālo un fizisko vidi, piedāvājot lietotājam konkrētu darbības veikšanu, lai panāktu vēlamo rezultātu.

Attiecīgajā scenārijā, kad tiek veikta ControlUnit releju pārbaude, uz displeja izvadās signāls "Listen for relay!" Šis norādījums informē lietotāju par nepieciešamību klausīties releju aktivizēšanās skaņu, lai efektīvi izmantotu programmatūras funkciju. Lietotājs, uzklausot šo skaņu, var novērtēt releja darbību un secināt, vai attiecīgais relejs ir darbspējīgs vai nē. Šāda pieeja nodrošina lietotājam konkrētu rīcību, lai noteiktu mikročipa stāvokli un veiktu nepieciešamos pasākumus. Tas savukārt uzlabo programmatūras lietojamību, piedāvājot lietotājam skaidras norādes un iespēju aktīvi iesaistīties darbību veikšanā, tādējādi veicinot efektīvu problēmu risināšanu un lietošanas pieredzi.

# Testēšanas dokumentācija

Šajā nodaļā detalizēti aprakstīts un vizualizēts programmatūras testēšanas process, iekļaujot izvēlētās metodes un rīkus, kā arī demonstrēti testpiemēri.

## Izvēlētās testēšanas metodes, rīku apraksts un pamatojums

Programmatūras testēšanas posmā izstrādātājs ir izvēlējies divas nozīmīgas testēšanas metodes - melnās kastes testēšanu un baltās kastes testēšanu. Šīs divas metodes nodrošina pietiekamu un rūpīgu pārbaudi, kas aptver gan programmatūras funkcionalitāti no lietotāja perspektīvas (melnās kastes testēšana), gan arī tās iekšējo struktūru un loģiku (baltās kastes testēšana). Šāda daudzpusīgā pieeja palīdzēs nodrošināt augstu kvalitāti un uzticamību visā programmatūras testēšanas procesā.

## Melnās kastes testēšanas rīku apraksts un pamatojums

Izstrādātājam, lai sasniegtu augstas kvalitātes programmatūras melnās kastes testus, ir jāizmanto un jānodrošina vairāki rīki:

* Lielākā daļa programmatūras funkcionalitātes balstās uz fizisku mikročipu pārbaudi. Izstrādātājam jānodrošina visaptveroša pārbaude, aptverot visus iespējamos scenārijus. Piemēram, var būt nepieciešams pārbaudīt, vai EEPROM komponente darbojas pareizi uz TTunit mikročipa, vai arī jāpārbauda, vai mikrokontrollieru pareizi mijiedarbojas starp produktu un ControlUnit. Šo rīku pamatojums slēpjas tajā, ka programmatūras lietošanas scenāriji var būt ļoti dažādi, tādēļ, lai nodrošinātu uzticamu programmatūras testēšanu, nepieciešams iegūt pilnīgu un detalizētu priekšstatu par visiem iespējamajiem scenārijiem. Šāda pieeja ļauj izstrādātājam veiksmīgi identificēt un novērst iespējamās problēmas pirms tās nonāk lietotāju rokās.
* Produkta konfigurācijas aplikācija ir papildus sistēmas elements, kas balstās uz dažādiem ievades laukiem un fizisku komunikāciju starp produktu un darbstaciju, nodrošinot precīzu un efektīvu programmatūras darbību. Lai sasniegtu augstu kvalitāti un uzticamu produktu, izstrādātājam ir jāveic rūpīga ievades lauku manipulācija, lai nodrošinātu veiksmīgu testēšanas procesu un uzticamus testus. Tas ietver ne tikai ievades lauku vērtību dažādo variantu testēšanu, bet arī jāveic komunikācijas traucējumu simulācijas, lai novērtētu sistēmas izturību un atgūstīšanas spējas neparedzētos apstākļos. Tādējādi izstrādātājs ir atbildīgs par visu šo procesu plānošanu, izpildi un dokumentēšanu, lai nodrošinātu pilnīgu un precīzu produktu konfigurācijas aplikācijas darbību un savietojamību ar dažādām darbības vides situācijām. Šāda pieeja ļauj veiksmīgi identificēt un novērst potenciālās problēmas jau izstrādes stadijā, nodrošinot augstu programmatūras kvalitāti un lietotāja apmierinātību.

## Baltās kastes testēšanas rīku apraksts un pamatojums

Izstrādātājam, lai sasniegtu augstas kvalitātes programmatūras baltās kastes testus, izstrādātājam ir jāizmanto un jānodrošina vairākas testēšanas metodes, kas sniedz padziļinātu ieskatu programmēšanas kodā un nodrošina pārliecību par programmatūras darbību:

* Izvades validācijas testēšanas metodes ļauj izstrādātājam veiksmīgi veikt baltās kastes testus, pārbaudot programmas izvades rezultātus un to atbilstību paredzētajiem kritērijiem. Piemēram, ir nepieciešams pārbaudīt, vai konkrēta vērtība, ko programma izdrukā, atbilst paredzētajam rezultātam noteiktā situācijā. Šī pieeja nodrošina uzticamu programmatūras funkcionalitātes pārbaudi dažādos darbības scenārijos un apstākļos, nodrošinot to precizitāti un atbilstību specifikācijām.
* Cikla testēšanas metode ir būtiska izstrādājamās programmatūras testēšanas procesā, jo lielākā daļa programmatūras balstās uz atkārtojošiem cikliem un pastāvīgu darbību. Cikla testēšana nodrošina pārliecību par programmatūras stabilitāti un pareizu darbību, izmantojot gan programmatūras, gan iekšējos ciklus. Tas palīdz identificēt un novērst iespējamās kļūdas, kas var rasties bezgalīgā darbības ciklā, un nodrošina augstu programmatūras kvalitāti un uzticamību.

Šīs baltās kastes testēšanas metodes ir būtiskas, lai nodrošinātu pilnīgu un precīzu ieskatu programmēšanas kodā un garantētu augstu programmatūras kvalitāti, atbilstību specifikācijām un izcilu lietotāju pieredzi.

## Testpiemēru kopa

## Testēšanas žurnāls

# Secinājumi

Šajā nodaļā aprakstīts paša izstrādātāja secinājumi par izstrādāto programmatūru, tā darba apjomu, problēmām un izaicinājumiem.

## Rezultāta novērtējums

Šis projekts ir bijis ne tikai darbs vai uzdevums, bet gan ceļojums, kas man ir sniedzis īpašības un zināšanas profesionālajā nozarē. Strādājot pie šīs programmatūras, esmu ieguvis plašu klāstu jaunu prasmju un zināšanu, kas ne tikai ir paplašinājušas manas zināšanas par programmatūras izstrādi, bet arī stiprinājušas manu pašpārliecību un mācījušas man risināt sarežģītas problēmas.

Esmu gandarīts par iespēju iepazīties ar jaunām programmēšanas jomām un izmēģināt dažādas programmēšanas metodes un rīkus. Katrs izaicinājums un problēma, ar kuru saskāros, ir bijusi iespēja attīstīt savas prasmes un spējas.

Process man nav tikai solis uz priekšu karjeras ceļā, bet gan liels pamats manai profesionālajai identitātei un nākotnes izaicinājumiem. Tas ir apliecinājums tam, ka man ir spēks un potenciāls sasniegt savus mērķus un sapņus, kā profesionālam programmētājam.

## Uzdevumu sasniegšanas analīze

Pats varu secināt, ka esmu sasniedzis un izstrādājis visas pasūtītāja prasības, kā arī esmu izstrādājis programmatūru ko pasūtītājs šobrīd aktīvi izmanto ikdiena uzņēmumā “SIA InPass”, tādēļ izstrādātā programmatūra ir veiksmīgi izstrādāta.

Es esmu pārliecināts, ka esmu pilnībā sasniedzis un izstrādājis visas pasūtītāja prasības, nodrošinot radošu un efektīvu risinājumu. Turklāt esmu radījis programmatūru, kas pašlaik tiek aktīvi izmantota uzņēmuma "SIA InPass" ikdienas darbā. Šis fakts liecina par to, ka izstrādātā programmatūra ir ne tikai sekmīgi izveidota, bet arī efektīvi integrēta uzņēmuma operatīvajos procesos, atbilstot tā darbības vajadzībām un prasībām. Šis rezultāts ir rūpīgas darba un uzticības attiecību izveides, kurām esmu veltījis savu laiku un pūles, un tas liecina par veiksmīgu sadarbību un kopīgu panākumu.

## Darba apjoms

Tā kā programmatūras funkcionālais process ir diezgan sarežģīts un komplekss darba apjoms, manā uztverē, ir liels. Paša rīka produkta programmatūras apjoms:

* Failu skaits ir 15, lielākoties, “.py” jeb python faili.
* Funkciju un metožu skaits ir aptuveni 40.
* Aptuvenais kopējais komandrindu skaits ir 3375 (neieskaitot tukšumus vai komentārus).

# Lietoto terminu un saīsinājumu skaidrojumi

**4.tabula**

**Terminu un saīsinājumu skaidrojumi**

|  |  |
| --- | --- |
| **Termins vai saīsinājums** | **Skaidrojums** |
| TT/CU skeneris | Nosaukums izstrādātajam produktam. |
| Mikročips | Pusvadītāju materiāla plāksne ar pievienotām komponentēm . |
| TTunit | Mikročips ko veido un izmanto uzņēmums “SIA InPass” priekš laika uzskaites. |
| ControlUnit | Mikročips ko veido un izmanto uzņēmums “SIA InPass” priekš dažādu mehatronisku elementu darbībām. |
| Mikrokontrollieris | Vadības ierīce jeb komponente, kas ietver mikroprocesoru. |
| Real-Time-Clock | Digitāla pulksteņa komponente ar primāro funkciju, lai precīzi sekotu laikam pat tad, ja strāvas padeve ir izslēgta. |
| RTC | Saīsinājums terminam Real-Time-Clock. |
| EEPROM | Rakstāma un lasāma atmiņa, kuras saturu var izdzēst un pārprogrammēt, izmantojot impulsa spriegumu. |
| EEPROM lapa | Atmiņas bloks EEPROM komponentē, ko izmanto neatkarīgām lasīšanas un rakstīšanas darbībām. |
| Zumers | Elektriskā signāla komponente, kas rada dūkojošu skaņu. |
| RFID | Tehnoloģija, kas izmanto radioviļņus, lai bezvadu režīmā identificētu objektus. |
| RFID karšu lasītājs | Ierīce, kas nolasa RFID radioviļņus un sniedz datus uz procesora. |
| UART | Universālais asinhronais uztvērējs-raidītājs ir mikrokontrollieru sakaru protokols, ko izmanto seriālai saziņai starp ierīcēm. |
| ControlUnit adreses | Ligzdas uz ControlUnit mikročipa, kas sniedz datus no citām ierīcēm uz mikrokontrolliera. |
| Wiegand | Sakaru protokols, ko parasti izmanto piekļuves kontroles sistēmās, lai pārsūtītu datus starp karšu lasītājiem un vadības paneļiem. |
| IDE | Integrētā izstrādes vide. |
| SSH | Kriptogrāfijas tīkla protokols, kas nodrošina drošu saziņu neaizsargātā tīklā, kas ļauj lietotājiem droši piekļūt un pārvaldīt attālās sistēmas un ierīces. |
| Micro-USB | Standarta savienotājs, ko plaši izmanto mobilo ierīču savienošanai uzlādes nolūkos. |
| 2x20 ligzda | Ligzdas uz ControlUnit un TTunit mikročipa, kas sniedz strāvu citiem mikročipiem, kā arī datu plūsmu starp mikrokontrollieriem. |
| 16\_VIN | Galvenā strāvas ligzda uz ControlUnit mikročipa. |

# Literatūras un informācijas avotu saraksts

Nodaļā tiek aprakstīts izstrādes laikā izmantotā literatūra un informācija, kas sekmēja izstrādes procesu izstrādātājam.

Micropyhon programmaparatūras dokumentācija.

Avots: <https://docs.micropython.org/en/latest/>

# Pielikumi