**JELGAVAS TEHNIKUMS**

**Aleks Markuss Balodis**

**Datu ieguve un apmaiņa izmantojot mikro kontrolierus**

**Kvalifikācijas darbs**

**kvalifikācijas ieguvei**

**programmēšanas tehniķis**

**Darba izpildītājs:  410.gr.izgl. Aleks Markuss Balodis**

***paraksts, datums***

**Jelgava**

**2022**

ANOTĀCIJA

Balodis Aleks Markuss. Datu ieguve un apmaiņa izmantojot mikro kontrolierus: kvalifikācijas darbs. Jelgava: JT, 2022. 17 lpp., 0 att., 0 tab., 0 bibl. nos., 0 pielikumi..

Mikro kontrolieri plaši tiek izmantoti mūsdienās dažādās ierīcēs, lai veiktu vienkāršas funkcijas, kur nav nepieciešama liela veiktspēja. Darba mērķis ir ievākt datus par laikapstākļiem ar mikro kontrolieriem un izmantojot bezvadu komunikāciju nosūtīt tos uz serveri, kas tos saglabās datubāzē un attēlos lietotājam viegli pārredzamā veidā mājaslapā.

Mikro kontrolieri paši nespēj izmērīt temperatūru, gaismas daudzumu vai nosūtīt informāciju bez vadiem, bet vien apstrādāt loģiku, tāpēc tiek pievienoti dažādi moduļi vai komponentes. Komponentes kā *LED* tiek pievienotas izmantojot *GPIO* kontaktus, bet moduļi kā, piemēram nRF24L01 bezvadu modulis, izmanto komunikācijas protokolu SPI.

Darba veikšanai nepieciešams būs izmantot dažādas programmēšanas valodas kā *C*. Koda rediģēšanai tiks izmantots *Visual Studio Code* un *Arduino IDE*. Versiju kontroles sistēmai tiks izmantots *GIT*.

ANNOTATION

Balodis Aleks Markuss. Data collection and exchange using microcontrollers – qualification work. Jelgava: JT, 2022, 17 pages, 0 images, 0 tables, 0 used information sources, 0 attachments.

Microcontrollers are widely used today in various devices to perform basic functions which do not require high performance. The purpose of my final year project is to harvest weather data using microcontrollers and then using wireless communication send the harvested data to a server, which would store the data in a database.

Microcontrollers by themselves cannot measure temperature, light or send information wirelessly. They can only process logic, that is why extra modules or components are added to them. Components like LEDs are added using GPIO pins, but modules like the nRF24L01 wireless module use a communication protocol, in this case SPI.

For this project I will need to use a programming language like C. To edit the code, I will be using Visual Studio Code and the Arduino IDE. For version control I will be using GIT.

SATURS

[IEVADS 1](#_Toc105547881)

[1. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA 2](#_Toc105547882)

[1.1 Ievads 2](#_Toc105547883)

[1.1.1 Dokumentācijas mērķis 2](#_Toc105547884)

[1.1.2 Produkta sfēra 2](#_Toc105547885)

[1.1.3 Dokumentācijas pārskats 2](#_Toc105547886)

[1.2 Vispārīgs apraksts 3](#_Toc105547887)

[1.2.1 Produkta perspektīva 3](#_Toc105547888)

[1.2.2 Produkta funkcijas 3](#_Toc105547889)

[1.2.3 Programmatūras gaita 3](#_Toc105547890)

[1.2.4 Mērķauditorija 3](#_Toc105547891)

[1.3 Funkcionālās prasības 3](#_Toc105547892)

[1.4 Nefunkcionālās prasības 5](#_Toc105547893)

[2. IZSTRĀDEI IZMANTOTĀS TEHNOLOĢIJAS UN PROGRAMMATŪRA 6](#_Toc105547894)

[2.1 Programmatūras izstrādes vide 6](#_Toc105547895)

[2.2 Visual Studio Code izmantotie paplašinājumi 6](#_Toc105547896)

[2.2.1 PlatformIO 7](#_Toc105547897)

[2.2.2 C / C++ Syntax 7](#_Toc105547898)

[3. PRODUKTA KOMPONENŠU APRAKSTS 8](#_Toc105547899)

[3.1 Mikro kontroliera pirmkoda komponentes 8](#_Toc105547900)

[3.1.1 Bezvadu moduļa nRF24L01 bibliotēkas funkcijas 8](#_Toc105547901)

[4. NOBEIGUMS 13](#_Toc105547902)

[4.1 Kopsavilkums 13](#_Toc105547903)

[4.2 Secinājumi 13](#_Toc105547904)

# **IEVADS**

Darba tēma “Datu ieguve un apmaiņa izmantojot mikro kontrolierus” tika izvēlēta, lai uzlabotu savas prasmes zema līmeņa programmēšanā. Mūsdienās programmētāji arvien vairāk izvēlas pilnveidot savas zināšanas WEB vai aplikāciju programmēšanas, bet arvien retāk zema līmeņa programmēšanā – BIOS programmēšanā, mikro kontrolieru programmēšanā un cita veida *embedded* programmēšanā.

Darba laikā tika izmantota C++ programmēšanas valoda un netika izmantotas gatavas bibliotēkas, lai izprastu kā ir strādāt bez abstrakcijas.

Darba mērķis: Izveidot no nulles mikro kontroliera programmatūru, kas var ievākt datus par laika apstākļiem un tos nosūtīt uz citu mikro kontrolierī.

Darba uzdevumi:

Izveidot bibliotēkas nRF24L01 bezvadu modulim, DHT11 temperatūras/mitruma sensoram un gaismas sensoram.

Izveidot pārbaudi vai saņemtie dati ir neskarti.

# **PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA**

## Ievads

### Dokumentācijas mērķis

Programmatūras prasību specifikācijas nolūks ir aprakstīt prasības uzdevumu plānošanas informācijas sistēmā.

Dokuments paredzēts izstrādātājiem un programmatūras lietotājiem, lai varētu izprast programmatūras funkcijas, darbību un specifiku. Dokumentācija palīdzēs tās lietotājiem integrēt programmatūru veiksmīgi savā projektā.

### Produkta sfēra

Programmas produkta nosaukums ir “MCU Weather” – C++ pirmkods, ko var izmantot savos projektos, lai ievāktu datus par laikapstākļiem un nosūtīt tos izmantojot bezvadu savienojumu.

Programmu var pielietot dažādos veidos, dažas aplikācijas ir termometrs, apkures sistēmas un gaisa mitrinātājs.

### Dokumentācijas pārskats

## Vispārīgs apraksts

### Produkta perspektīva

Izstrādātā programma (pirmkods) ir pieejams vietnē GitHub.

Šo pirmkodu var izmantot jebkurš, piemēram, hobijisti, lai integrētu, pielāgotu un izmantotu savām vajadzībām. Apskatot šo kodu arī rodas priekštats par to kā programmēt zemā līmenī.

### Produkta funkcijas

Izstrādātai programmatūrai ir vairākas funkcijas. Pirmā pamata funkcija ir iespēja vienkāršoti strādāt ar nRF24L01 bezvadu moduli – to konfigurēt savām vajadzībām, sūtīt un saņemt datus. Otrā funkcija ir iespēja veikt temperatūras un gaisa mitruma mērījumus ar DHT11 sensoru. Trešā funkcija ir veikt apgaismojuma mērījumus ar foto-pretestību. Ceturtā funkcija ir nosūtīt datus uz citu mikro kontrolieri kurš tos apstrādā un pārraida datoram/serverim.

### Programmatūras gaita

Mikro kontrolierim uzsākot darbu tiks inicializēti visi nepieciešamie moduļi – nRF24L01, DHT-11, Gaismas sensors. Tālāk tiks periodiski ievākti dati – gaismas līmenis, gaisa mitrums, temperatūra un tiks nosūtīti centrālajam kontrolierim. Centrālais kontrolieris savukārt būs pievienots datoram un nosūtīs visus saņemtos datus tam. Pēc tam dators saņemtos datus ievietos datubāzē.

### Mērķauditorija

Programmatūra ir paredzēta hobijistiem kuri vēlas sava projekta ietvaros ievākt datus par laika apstākļiem un tos nosūtīt izmantojot bezvadu savienojumu uz citu mikro kontrolieri.

Projekta pirmkods ir brīvi pieejams ikvienam GitHub un to var rediģēt, lai pielāgotu savām vajadzībām

## Funkcionālās prasības

Programmatūras funkcionālās prasības ietilpst šāda funkcionalitāte un sistēmas darbības principi:

Klienta mikro kontrolieris ievāc sekojošos datus par apkārtni: temperatūru, gaisa mitrumu un apgaismojuma spilgtumu.

Klienta mikro kontrolieris šos ievāktos datus pārraida ar bezvadu savienojuma palīdzību caur nRF24L01 moduli.

Dati tiek nosūtīti periodiski.

Centrālais mikro kontrolieris saņem datus no klienta.

Centrālajam mikro kontrolieris ievieto šos datus datubāzē.

Programmatūra ir saderīga ar Arduino.

## Nefunkcionālās prasības

# **IZSTRĀDEI IZMANTOTĀS TEHNOLOĢIJAS UN PROGRAMMATŪRA**

## Programmatūras izstrādes vide

Lai izstrādātu programmatūru ATmega328P (*Arduino Uno*) mikro kontrolierim esmu izvēlējies izmantot *Visual Studio Code (VSC)* ar paplašinājumiem *PlatformIO* un *C / C++ Syntax*. VSC izvēlējos, jo šis IDE pataisa kodu pārskatāmu un tam arī ir *IntelliSense*, kas atvieglo koda rakstīšanu. Paplašinājums *PlatformIO* pielāgo *VSC* darbam ar mikro kontrolieriem ļaujot augšupielādēt kodu uz tiem un izmantot dažas pamata bibliotēkas, kas atvieglo darbu.

Par cik rakstot kodu uz mikro kontrolieriem tiek izmantota *C / C++* valoda paplašinājums *C / C++ Syntax* ļauj *VSC* saprast *C / C++* sintaksi un izcelt to.

*Visual Studio Code (VSC)* ir Microsoft veidota teksta rediģēšanas programma, kurai ir iespējams uzstādīt dažādus paplašinājumus, kas nodrošina lietotājam nepieciešamo funkcionalitāti. VSC pamatā nodrošina šīs un citas funkcijas:

* integrētu komandas rindu;
* sintakses pārbaudītāju;
* *IntelliSense*, kas ļauj automātiski pabeigt rakstīto, dod informāciju par funkciju, tās parametriem un tipu un daudz ko citu.

## Visual Studio Code izmantotie paplašinājumi

Lai izstrādātu programmatūru esmu izvēlējies izmantot *Visual Studio Code* ar sekojošiem paplašinājumiem. Šie paplašinājumi padara programmēšanu vieglāku integrējot *VSC* tādas funkcijas kā:

* Koda augšupielāde mikro kontrolierī
* Seriālās konsoles integrācija *VSC*
* *C / C++* sintakses izgaismošana vieglākai koda pārlasāmībai

### PlatformIO

### C / C++ Syntax

# **PRODUKTA KOMPONENŠU APRAKSTS**

## Mikro kontroliera pirmkoda komponentes

### Bezvadu moduļa nRF24L01 bibliotēkas funkcijas

**void CS\_Select(void);**

Funkcija iestata Chip Select signālu zemajā līmenī SPI interfeisā starp MCU un nRF24L01. Šī funkcija neatgriež neko un nepieprasa parametrus.

**void CS\_UnSelect(void);**

Funkcija iestata Chip Select signālu augstajā līmenī SPI interfeisā starp MCU un nRF24L01. Šī funkcija neatgriež neko un nepieprasa parametrus.

**void CE\_Enable(void);**

Funkcija iestata Chip Select Not (Chip Enable) signālu augstajā līmenī SPI interfeisā starp MCU un nRF24L01. Šī funkcija neatgriež neko un nepieprasa parametrus.

**void CE\_Disable(void);**

Funkcija iestata Chip Select Not (Chip Enable) signālu zemajā līmenī SPI interfeisā starp MCU un nRF24L01. Šī funkcija neatgriež neko un nepieprasa parametrus.

**void WriteRegister(uint8\_t reg, uint8\_t data);**

Funkcija ļauj rakstīt nRF24L01 reģistros caur SPI interfeisu. Kā parametrus funkcija pieņem reģistra adresi un datus, kas tiks ierakstīti konkrētajā reģistrā. Funkcija neatgriež vērtību.

**void WriteRegisterMulti(uint8\_t reg, uint8\_t \* data, int size);**

Funkcija ļauj rakstīt konkrētā reģistrā vairākas vērtības. Funkcija neatgriež neko, bet kā parametrus pieņem reģistra adresi, rādītāju uz vērtību masīvu un masīva lielumu.

**uint8\_t ReadRegister(uint8\_t reg);**

Funkcija ļauj nolasīt nRF24L01 reģistra vērtību caur SPI interfeisu. Kā parametru funkcija pieņem nolasāmā reģistra adresi. Kā vērtību funkcija atgriež veselu skaitli, kas atbilst reģistra vērtībai.

**void ReadRegisterMulti(uint8\_t reg, uint8\_t \* data, int size);**

Funkcija ļauj nolasīt vairākas vērtības no konkrēta reģistra caur SPI interfeisu. Funkcija neatgriež neko, bet kā parametrus pieņem nolasāmo reģistru, rādītāju uz masīvu, kur ievietot nolasītās vērtības un masīva lielumu.

**void SendCommand(uint8\_t cmd);**

Funkcija caur SPI interfeisu ļauj nosūtīt komandu uz nRF24L01. Funkcija vērtību neatgriež, bet kā parametru pieprasa nosūtāmo komandu.

**void Init(void);**

Funkcija sagatavo nRF24L01 bezvadu moduli darbībai. Vispirms konfigurācija tiek nodzēsta iestatot visas vērtības uz 0. Pēc tam tiek izslēgta Auto-Acknowledgment funkcija, izslēgtas visas datu saņemšanas līnijas, saņemšanas/sūtīšanas līniju platums tiek iestatīts uz 5 baitiem, Auto-Retransmit tiek iestatīts uz 250 mikrosekundēm, moduļa frekvence tiek iestatīta uz 2400 MHz un ātrums uz 2Mbps. Funkcija neatgriež vērtību un nepieprasa parametrus.

**void TX\_Mode(uint8\_t \*address, uint8\_t ch);**

Funkcija iestata nRF24L01 bezvadu moduli sūtītāja režīmā. Kā parametrus funkcija pieprasa rādītāju uz sūtīšanas adreses masīvu un moduļa frekvenci. (Frekvence = 2400 + ch).

**uint8\_t Transmit(uint8\_t \*dati);**

Funkcija pārraida datus, kas atrodas datu sūtīšanas buferī. Atgriež 1, ja dati ir veiksmīgi nosūtīti – citādi 0. Kā parametru funkcija pieņem rādītāju uz sūtāmo datu masīvu. Šim masīva lielumam ir jābūt 32.

**void RX\_Mode(uint8\_t \*address, uint8\_t ch);**

Funkcija iestata nRF24L01 bezvadu moduli saņēmēja režīmā. Kā parametrus funkcija pieprasa rādītāju uz saņemšanas adreses masīvu un moduļa frekvenci. (Frekvence = 2400 + ch).

**uint8\_t IsDataAvailable(int pipenr);**

Funkcija pārbauda vai ir dati konkrētajā reģistrā. Ja ir dati konkrētā reģistrā funkcija atgriež 1, citādi 0. Kā parametru pieņem saņemšanas reģistra numuru.

**void Receive(uint8\_t \*data);**

Funkcija ļauj saņemt datus. Neatgriež vērtību. Kā parametru pieņem rādītāju uz masīvu, kur glabāt saņemtos datus. Šim masīva garumam ir jābūt 32.

# **NOBEIGUMS**

## Kopsavilkums

## Secinājumi