Rīgas Tehniskā universitāte

Elektronikas un Telekomunikāciju fakultāte

Elektronisko vadības sistēmu projektēšana

Kursa darbs

Bluetooth Low Energy atslēga

Studenta vārds, uzvārds: Krišjānis Noviks

Fakultāte, grupa: ETF REBM01

Studenta apliecības numurs: 201REB605

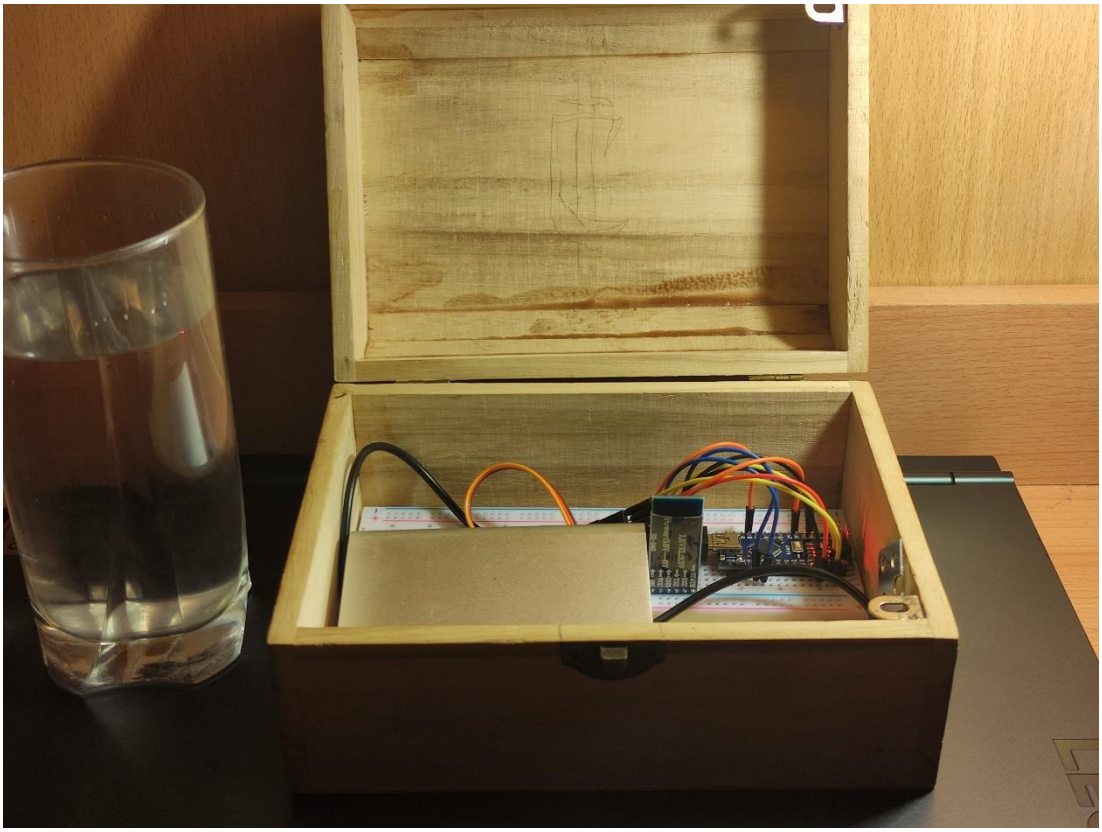
1. Projekta apraksts

Šajā darbā es izveidoju kasti, kuru iespējams atslēgt un aizslēgt, ievadot paroli no telefona.

Lai atslēgtu kasti, ir nepieciešama jebkāda aplikācija, kura spēj komunicēt ar *Bluetooth Low Energy(BLE)* ierīcēm un sūtīt *serial* datus. Kad caur *serial* terminālu tiek iesūtīta parole, tad kaste atslēgsies, ja tā ir aizslēgta, vai aizslēgsies, ja tā ir atslēgta. Arī, kā signals par veiksmīgu kastes slēgšanu ir pīkstulis.

Projektā es izmantoju:

- Arduino Nano
- 1 Pīkstuli
- BLE moduli HM-10
- 1 Servo motoru
- 5V Bateriju



1. Attēls. Aizslēdzamās kastes izskats

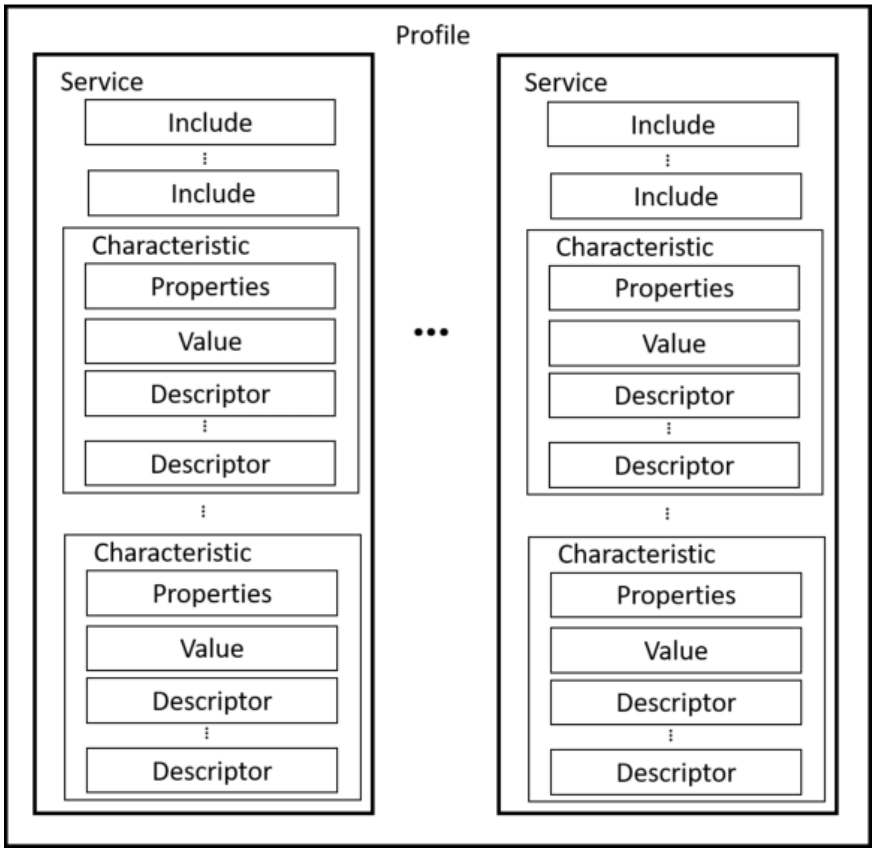
Es izmantoju Servo motoru, jo to var precīzi novietot un pagriezt pa noteiktu skaitu grādiem, un tas aizņēma mazāk vietas kā Stepper motors ar vadības slēgumu.

Lai atslēgtu un aizslēgtu izmantoju aplikāciju **BLE Scanner**, taču var izmantot jebkuru programmu, kas ļauj sūtīt serial datus BLE ierīcēm (parasti kāds BLE terminmāls)

2. Bluetooth Low Energy (BLE)

Publikai pieejams kļuvis 2010 gadā kopā ar Bluetooth 4.0, BLE ir pavisam citāds protokols, kas atšķiras no Bluetooth, domāts priekš dažādām ierīcēm, kurās nepieciešams novērot vienu vai mazu skaitu ar mainīgajiem, piemēram, baterijas procentus vai sirdspukstus minūtē. Galvenās atšķirības ir tas, ka tā vietā lai nepārtraukti novērotu savienotās ierīces, BLE sūta informāciju tikai tad, kad tai ir ko sūtīt, kad ir kāda izmaiņa norādītajos parametros, un tas, ka nav nepieciešama pārošana ar ierīcēm. Izmanto General Attributes sistēmu (Gatt), lai meklētu kā veidot programmu ar Bluetooth Low Energy jāveido Gatt informāciju pārstrādojoša programma.

Manā projektā šis ir noderīgi, jo šis ievērojami samazina patērēto enerģiju.



2.Attēls. Gatt datu struktūra

No šīs struktūras, lai atslēgtu kasti, es aizūtu noteiktu vērtību (*Value*) noteiktai īpašībai (*Characteristic*), šajā gadījumā vienkārša ieejas/izejas teksta īpašība un aizsūtu es burtu virkni *string*.

3. Android programmēšana

Beigu projektā es nolēmu neizmantot sevis veidotu android aplikāciju, jo man neizdevās izveidot programmu, kas savienotu manu telefonu ar Bluetooth moduli un spētu sūtīt datus uz to. Manai aplikācijai būtu:

1. Jāpārbauda parole.
2. Jāmeklē BLE ierīces
3. Jāatbild ar visām pieejamajām ierīcēm
4. Jāsavienojas ar manu ierīci.
5. Jāaizsūta signāls, lai vērtu slēdzeni.

Ar savām zināšanām un laiku, man izdevās izveidot aplikāciju, kura veiksmīgi pārbauda paroli, taču man neizdevās pārējās ar BLE saistītās funkcijas, es mēģināju izmantot dažādas bibliotēkas, kā **Android BLE Library, NeatLE, RxAndroidBLE** un citas, taču visās sastapos ar dažādām problēmām. Man bija problēmas galvenokārt bibliotēku nesakritību, *plug-in* versiju nesakritību, jo es iepriekš neesmu izmantojis Android Studio un nezinu kā pielāgot pareizi šos complicētos kodus savām vajadzībām.

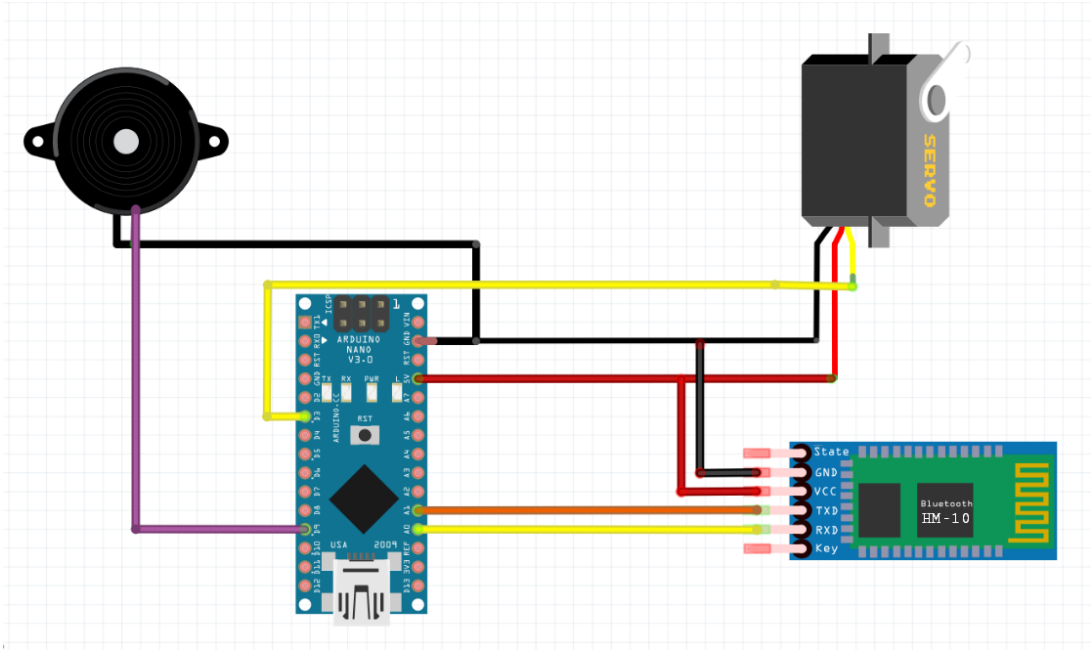
Neskatoties uz to, es no šī iemācījos par Android aplikāciju veidošanu, struktūru Koltin un Javas programmēšanas valodām.

3.Attēls Manis izveidotā Paroles aplikācija

4. Arduino programmēšana un savienošana.

Ar šo problēmu bija daudz mazāk, taču varu minēt, ka neizmantoju Arduino bibliotēku **SoftwareSerial**, lai sazinātos ar Bluetooth moduli un Arduino, bet gan trešās partijas bibliotēku **NeoSWSerial,** tādēļ, ka manam modulim bija problēmas ar to, ka reizēm tas datus saņēma pārāk ātri, vai pārāk novēloti un arī, ja pārāk garu tekstu vai specifiskus simbolus ievadīja Arduino serial input, tad visa programma nobrūk un nav iespējams vairs ne aizslēgt, ne atslēgt slēdzeni, šī bibliotēka novērš visas šīs problēmas un padara programmu stabilāku un drošāku. Lai kontrolētu motoru izmantoju **Servo** bibliotēku.

Paroli var uzstādīt kādu vēlas un uzlikt līdz 64 simbolu garumam nomainot mainīgo *pass* kodā, kas pašlaik ir uzstādīts uz “parole123”

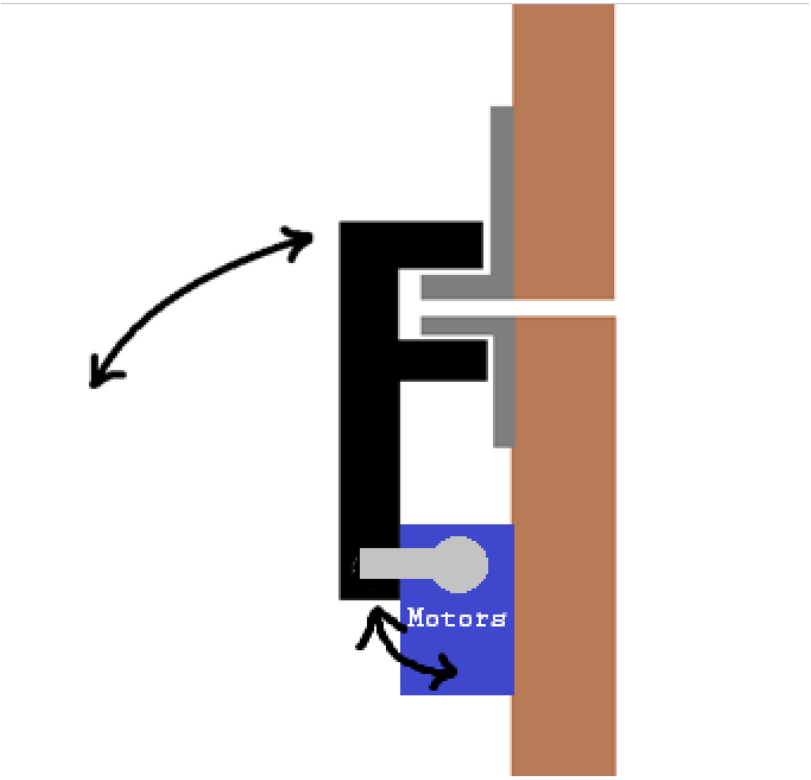


4.Attēls Arduino savienojuma shēma

(Shēmas izveidošanai izmantota programma Fritzing)

Jāņem vērā, ka HM-10 modulim RX un TX pini darbojas 3.3V līmenī, tāpēc, lai sūtītu informāciju no Arduino uz Moduli jāizmanto sprieguma dalītājs, taču Arduino spēj nolasīt arī 3.3V informāciju bez kaitējumiem iekārtai.

5. Slēgšanas mehānisms



5.Attēls Slēdzenes mehānisms

Ja bildēs nav pietiekami skaidri redzams, šāda ir mana slēdzene, tā nav visdrošākā, taču veidota šādi, lai nebojātu motoru, mēģinot ar varu atlauzt to, protams, nopietnos pielietojumos dizainu būtu jāpamaina. Man bija arī vairāki Stepper motori, pat ar skrūvi un sliedi no vecas diskmašīnas, kas būtu drošāks variants, lai aizslēgtu kastīti, taču nebūtu parocīgs priekš tik mazas kastes. Konstrukcija ir pietiekami izturīga, lai paceltu kastes smagumu pat ar vēl priekšmetiem iekšā turot kasti tikai aiz vāka.

Secinājumi.

Lai gan man neizdevās uzprogrammēt strādājošu Bluetooth aplikāciju priekš Android, man izdevās izveidot ar paroli atslēdzamu slēdzeni caur Bluetooth, šim variantam ir lielāka pielietojamība, jo atslēgt to var ar jebkādu ierīci, kas spēj sūtīt serial datus uz BLE ierīcēm, piemēram, portatīvo datoru vai citu Arduino ar Bluetooth moduli (jāizmanto tāds, kas spēj sūtīt datus Master mode, HM-10 ir šāda iespēja). Arī iemācījos daudz par Android aplikāciju būvēšanu un Bluetooth tehnoloģijām, taču lai uztaisītu iecerēto aplikāciju, man vispirms būtu jāuztaisa vismaz 5 daudz vienkāršākas, lai pieķertos klāt BLE.

Pielikumi.

Arduino kods.

```
#include <NeoSWSerial.h>
#include <Servo.h>

String ReceivedString = "";
Servo myServo;
NeoSWSerial mySerial( 15, 14 );  //TX, RX pini bluetooth modulim
char open = 0;

// IESTATĪT PAROLI
String pass = "parole123";

void setup() {
  // put your setup code here, to run once:
  mySerial.begin(9600);
  Serial.begin(9600);
  myServo.attach(3); //kontroles pin priekš servo motora
}

void loop() {
  //Lai notiritu buffer
  while (Serial.available() > 0) {
      Serial.read();
  }

  String ReceivedString = mySerial.readString();
  //Serial.println(ReceivedString);
  noTone(9);

  while(ReceivedString==pass){
      tone(9,1600); //pīkstēšana

      if (open==1) {
        myServo.write(80);
        open = 0 ;
      } else {
        myServo.write(115);
        open = 1 ;
      }
      ReceivedString = ""; //notīra saņemto tekstu
  }
}
```

Kotlin kods Android aplikācijai

Parolei

```
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //setContentView(R.layout.activity_main)
        binding = ActivityMainBinding.inflate(layoutInflater)

        setContentView(binding.root)

        binding.openbutton.text="Atvērt"
        binding.closebutton.text="Aizvērt"

        val answer = "0102"


        binding.openbutton.setOnClickListener(
            View.OnClickListener {
                var editText = binding.editTextNumberPassword.text.toString()
                if (editText.equals(answer)) {
                    binding.title.text = "Pareizi!"
                    Intent intent = new
Intent(MainActivity.this,MainActivity2.class)
                    startActivity(intent)
                } else binding.title.text = "Nepareizi, Ievadi Paroli:"

            }
            )

    }
}
```

Bluetooth Savienošanai (nestrādā)

```
class MainActivity2 : AppCompatActivity() {

    private val callback = object : ScanCallback() {
        override fun onBatchScanResults(results: MutableList<ScanResult>?) {
            results?.forEach { result ->
                deviceFound(result.device)
            }
        }

        override fun onScanResult(callbackType: Int, result: ScanResult?) {
            result?.let { deviceFound(result.device) }
        }

        override fun onScanFailed(errorCode: Int) {
            handleError(errorCode)
        }
    }

    private fun deviceFound(device: BluetoothDevice) {
        device.connectGatt(context, true, gattCallback)
    }


    fun scanForDevices(context: Context, serviceUUID: UUID) {
        val adapter = BluetoothAdapter.getDefaultAdapter()

        if (!adapter.isEnabled) {
            //handle error
        }

        val uuid = ParcelUuid(serviceUUID)
        val filter = ScanFilter.Builder().setServiceUuid(uuid).build()
        val filters = listOf(filter)

        val settings = ScanSettings
            .Builder()
            .setScanMode(ScanSettings.SCAN_MODE_LOW_LATENCY)
            .build()

        context.runWithPermissions(Manifest.permission.ACCESS_COARSE_LOCATION) {
            adapter.bluetoothLeScanner.startScan(filters, settings, callback)
        }
    }
    private val gattCallback = object : BluetoothGattCallback() {
        override fun onConnectionStateChange(gatt: BluetoothGatt?, status: Int, newState: Int) {
            if (newState == BluetoothGatt.STATE_CONNECTED) {
                gatt?.requestMtu(256)
                gatt?.discoverServices()
            }
        }

        override fun onServicesDiscovered(gatt: BluetoothGatt?, status: Int) {
            val characteristic = gatt?.getService(expandUuid(0x180F)) // Battery service
                ?.getCharacteristic(expandUuid(0x2A19)) // Battery life
            gatt?.readCharacteristic(characteristic)
            gatt?.setCharacteristicNotification(characteristic, true)
            characteristic?.value = byteArrayOf(50)
            gatt?.writeCharacteristic(characteristic)
        }

        override fun onCharacteristicRead(
            gatt: BluetoothGatt?, characteristic: BluetoothGattCharacteristic?, status: Int
        ) { /* ... */ }

        override fun onCharacteristicWrite(
            gatt: BluetoothGatt?, characteristic: BluetoothGattCharacteristic?, status: Int
        ) { /* ... */}

        override fun onCharacteristicChanged(
            gatt: BluetoothGatt?, characteristic: BluetoothGattCharacteristic?
        ) {
            characteristic?.let {
                val batteryLife = characteristic.value[0].toInt()
                Log.d(TAG, "Battery life is: $batteryLife")
            }
        }
    }
}
```

XML formatēšanas kods paroles programmai.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://sche-
mas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">



    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Ievadi paroli:"
        android:textSize="34sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.515"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.227" />

    <EditText
        android:id="@+id/editTextNumberPassword"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@android:color/background_light"
        android:backgroundTint="#5C5C5C"
        android:ems="10"
        android:inputType="numberPassword"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.527"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/title"
        app:layout_constraintVertical_bias="0.373" />

    <Button
        android:id="@+id/openbutton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="32dp"
        app:layout_constraintBottom_toTopOf="@+id/editTextNumberPassword"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.253"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/closebutton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="4dp"
        android:layout_marginBottom="32dp"
        app:layout_constraintBottom_toTopOf="@+id/editTextNumberPassword"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/openbutton" />

</androidx.constraintlayout.widget.ConstraintLayout>
```