**Национална програма “Обучение за ИТ кариера”**

# КУРСОВ ПРОЕКТ

**Тема:** Количка, контролирана чрез Bluetooth


Ученици:

Александър Милисов

Яна Братоева

Научни ръководители:

Димитър Николов

Росен Витанов



София

2019

# 1. Съдържание

# 2. Описание на проекта

Идеята на проекта е създаване на дистанционно управляема количка. Слобена от Lego и задвижвана от два мотора.
Управлението е реализирано на базата на танк, като всеки мотор отговаря за задвижването на едно от двете задни колела.
Завиването се осъществява като двата мотора се движат с различна скорост или в различни посоки.

За осъществяване на управлението на моторите е използван микроконтролер Arduino, програмиран на C.

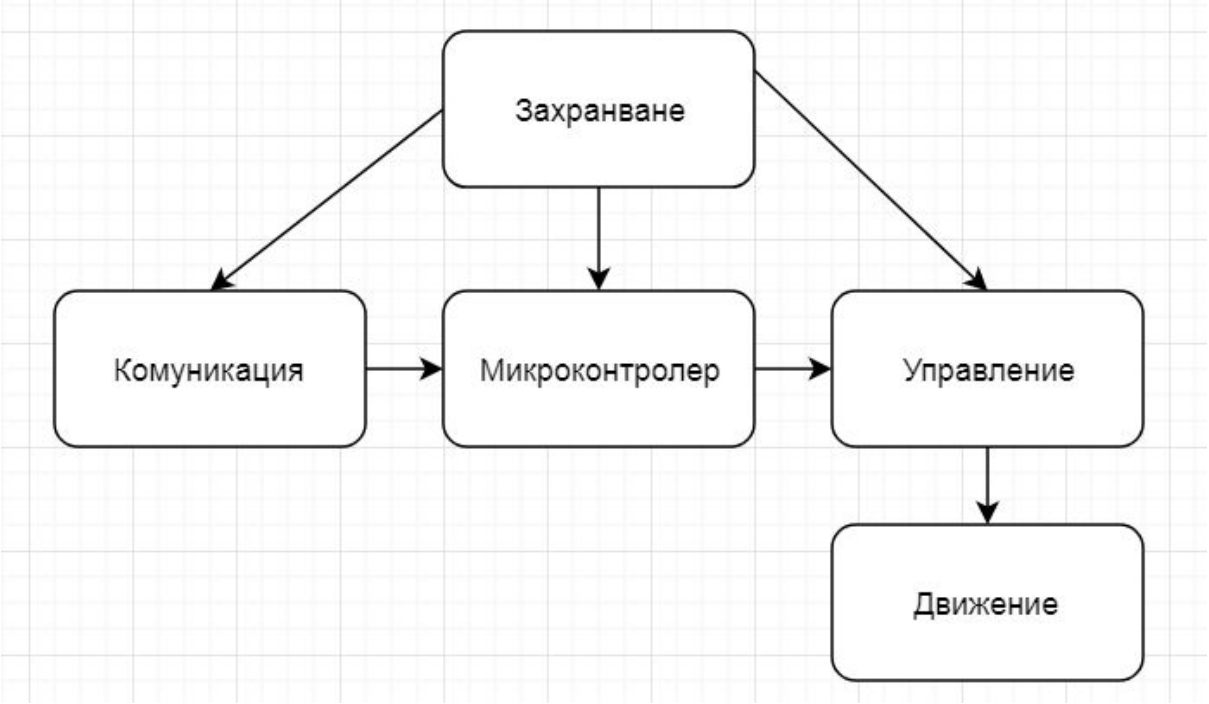
Написано е мобилно приложение за Android, написано на Kotlin, за дистанционно управление.

Използван е Bluetooth модул HC-05 за осъществяване на връзката с мобилното приложение.

Мобилното приложение осъществява изходна серийна комуникация на базата на RFCOMM чрез UUID на търсеното устройство.

# 3. Блокова схема

На фиг. 1 е показана блоковата схема на начина на работа на устройството.




*Фиг. 1.Блокова схема на работата на устройството*

# 4. Кратко описание на блоковете

## 4.1. Блок “Комуникация”

Блок “Комуникация” се използва за получаване на информация и изпращането и до блок “Микроконтролер”.

## 4.2. Блок “Микроконтролер”

Блок “Микроконтролер” получава команди от блок “Комуникация”, обработва ги и ги предава на Блок “Управление”.

## 4.3. Блок “Управление”

Блок “Управление” захранва и разрешава работата на блок “Движение”.

## 4.4. Блок “Движение”

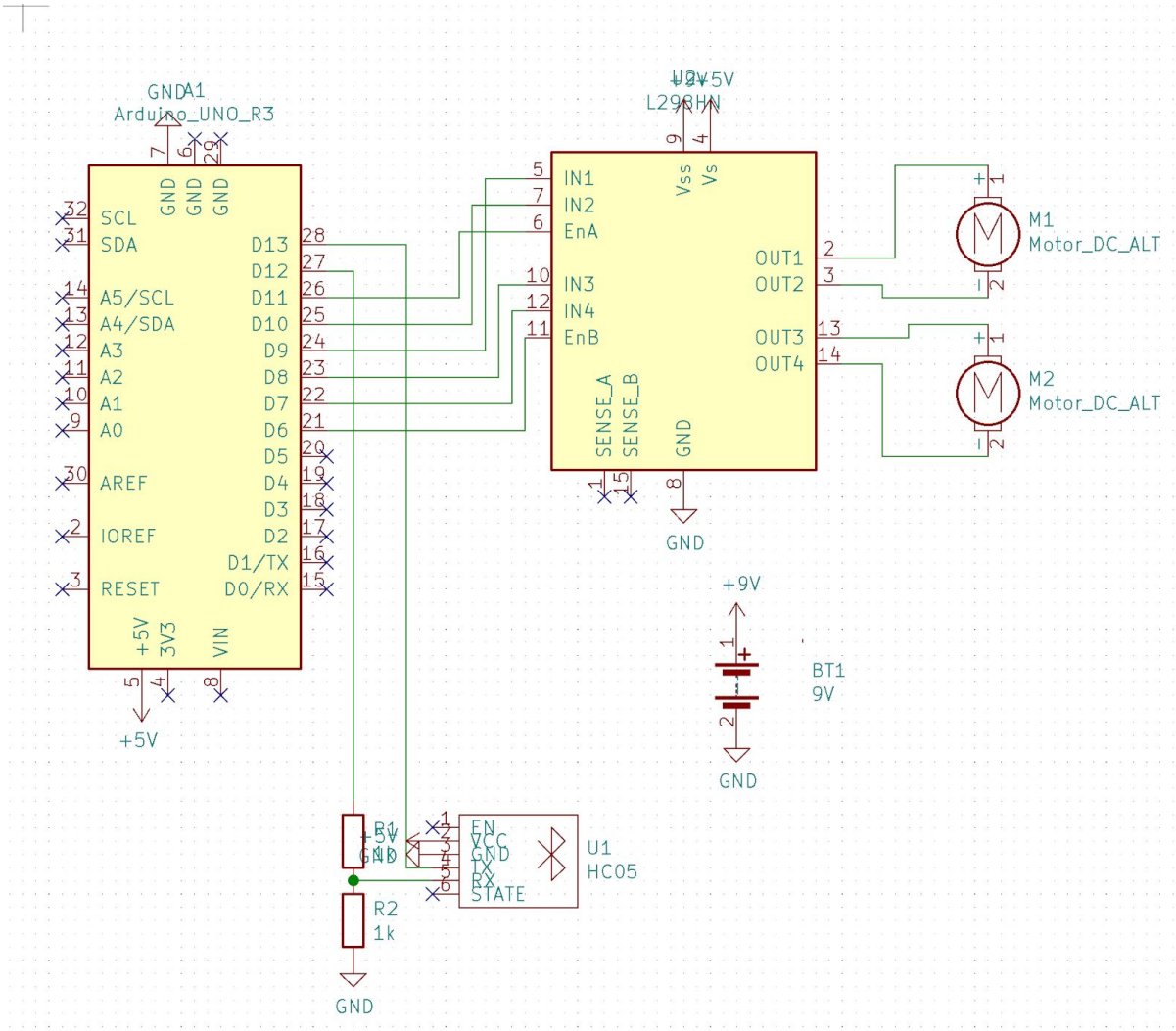
Блок “Движение” задвижва количката.

## 4.5. Блок “Захранване”

Блок “Захранване” служи за захранване на цялата електрическа схема.

# 5. Принципна електрическа схема

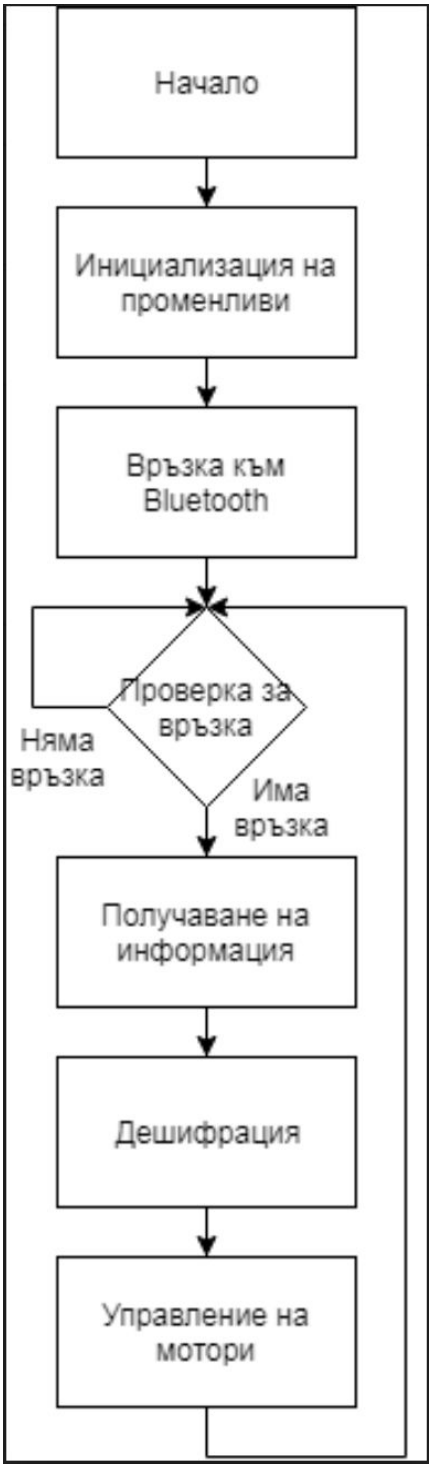
На фиг. 2 е показана принципната електрическа схема на устройството.




*Фиг. 2 Принципна електрическа схема на устройството, начертана на KiCad.*

# 6. Блокова схема на код за Ардуино

На фиг. 3 е показана блоковата схема на работата на кода за Ардуино.




*Фигура 3, изобразяваща начина на работа на кода от файл "Tank.ino".*

# 7. Код за Ардуино

На фиг. 4 и 5 е показан кодът от файл Tank.ino.

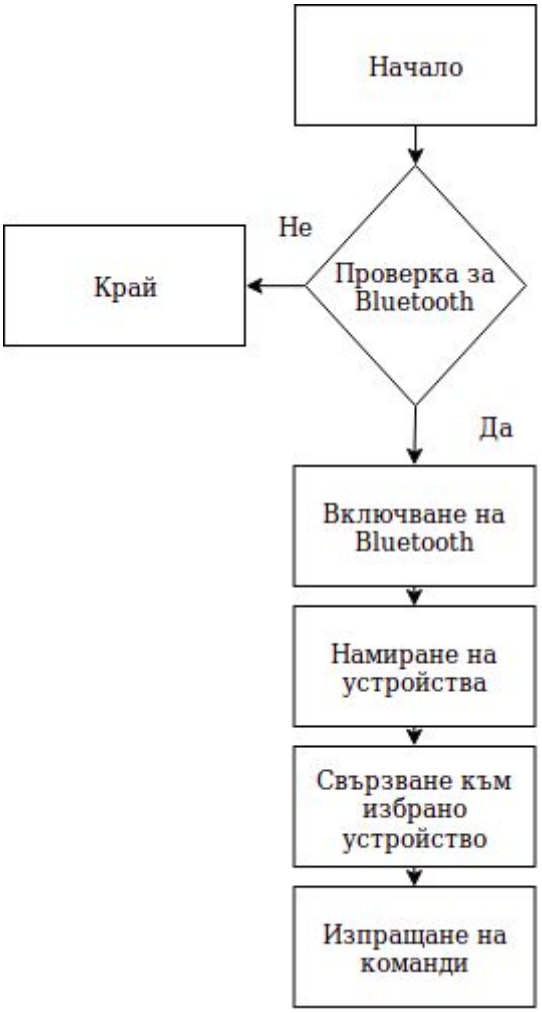
```
#include <SoftwareSerial.h>

byte remote_data;

SoftwareSerial Bluetooth(13, 12); // RX, TX on Arduino, connected to TX, RX on Bluetooth Module

struct Track {
  int pin1;
  int pin2;
  int enable;
};

void MoveTrack(struct Track, int, bool);

struct Track left_track;
struct Track right_track;


void setup() {

  Serial.begin(9600);
  Serial.println("var init");


  // left_track definition
  left_track.pin1 = 9;
  left_track.pin2 = 10;
  left_track.enable = 11;

  pinMode(left_track.pin1, OUTPUT);
  pinMode(left_track.pin2, OUTPUT);
  pinMode(left_track.enable, OUTPUT); // pwm

  //right_track definition
  right_track.pin1 = 8;
  right_track.pin2 = 7;
  right_track.enable = 6;

  pinMode(right_track.pin1, OUTPUT);
  pinMode(right_track.pin2, OUTPUT);
  pinMode(right_track.enable, OUTPUT); // pwm

  Bluetooth.begin(9600);
}
```

```
void loop() {

  if (Bluetooth.available()) {

    remote_data = Bluetooth.read();

    switch (remote_data) {
      case 'a':
        Serial.println("stop left track");
        MoveTrack(left_track, 0, false);
        break;
      case 'b':
        Serial.println("left track forward");
        MoveTrack(left_track, 255, true);
        break;
      case 'c':
        Serial.println("left track backward");
        MoveTrack(left_track, 255, false);
        break;
      case 'd':
        Serial.println("stop right track");
        MoveTrack(right_track, 0, false);
        break;
      case 'e':
        Serial.println("right track forward");
        MoveTrack(right_track, 255, true);
        break;
      case 'f':
        Serial.println("right track backward");
        MoveTrack(right_track, 255, false);
        break;
      default:
        Serial.println("uknown command received");
        break;
    }
  }
}

void MoveTrack(struct Track track, int speed_val, bool forward) {

  if (speed_val == 0) // no movement
  {
    digitalWrite(track.pin1, LOW);
    digitalWrite(track.pin2, LOW);
    return; // because analogWrite?
  }
  else if (forward) // move track forwards
  {
    digitalWrite(track.pin1, LOW);
    digitalWrite(track.pin2, HIGH);
  }
  else if (!forward) // move track backwards
  {
    digitalWrite(track.pin1, HIGH);
    digitalWrite(track.pin2, LOW);
  }

  analogWrite(track.enable, speed_val);
}
```

*Фигури 4 и 5, изобразяващи кода от файл "Tank.ino".*

# 8. Блокова схема на кода за мобилното приложение

На фиг. 6 е показана блоковата схема работата на мобилното приложение за дистанционно управление.




*Фигура 6, изобразяваща начина на работа на мобилното приложение.*

# 9. Код за мобилното приложение

На фиг. 7, 8 и 9 е показан кодът от файл ”MainActivity.kt”.

```kotlin
package com.example.phonetomodulebluetooth

import ...

class MainActivity : AppCompatActivity() {

    // private props
    private var m_bluetoothAdapter: BluetoothAdapter? = null
    private lateinit var m_pairedDevices: Set<BluetoothDevice> // list of paired devices to the Bluetooth adapter
    private val REQUEST_ENABLE_BLUETOOTH = 1 // bluetooth enable const

    // static properties
    companion object {
        val EXTRA_ADDRESS: String = "" // device addr
    }

    // UI + data init
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main) // Set app UI

        // get device's Bluetooth adapter
        m_bluetoothAdapter = BluetoothAdapter.getDefaultAdapter()

        // if device doesnt support BT
        if(m_bluetoothAdapter == null) {
            // spit out some err
            Log.i( tag: "bluetooth",  msg: "not supported")
            return
        }

        if(!m_bluetoothAdapter!!.isEnabled) {
            val enableBTIntent = Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE)
            startActivityForResult(enableBTIntent, REQUEST_ENABLE_BLUETOOTH)
//            Log.i("bluetooth", "device's bluetooth is now on")
        }

```

```kotlin

    // get/ refresh list of bluetooth devices
    refresh_button.setOnClickListener { it: View!
        pairedDeviceList()
    }

}

// get list of paired devices to the Bluetooth adapter
private fun pairedDeviceList() {
    m_pairedDevices = m_bluetoothAdapter!!.bondedDevices // get data set of paired devices

    val all_devices : ArrayList<BluetoothDevice> = ArrayList()

    if (!m_pairedDevices.isEmpty()) { // if there are paired devices, populate all_devices collection
        for (device: BluetoothDevice in m_pairedDevices) {
            all_devices.add(device)
            Log.i( tag: "bluetooth",  msg: "found: " +device.name + " - MAC -  " + device) // log name + MAC addr for every paired device found
        }
    } else { // if no paired devices found, log msg
        Log.i( tag: "bluetooth",  msg: "no paired devices found")
    }

    val adapter = ArrayAdapter( context: this, android.R.layout.simple_list_item_1, all_devices) // handle UI list
    list_all_devices.adapter = adapter

    list_all_devices.onItemClickListener = AdapterView.OnItemClickListener { _, _, position, _ ->
        val device: BluetoothDevice = all_devices[position]
        //val device_name : String = device.name
        val address: String = device.address

        // pass info to TankControlActivity
        val intent = Intent( packageContext: this, TankControlActivity::class.java)

        // put -> where, what
        intent.putExtra(Companion.EXTRA_ADDRESS, address)

        startActivity(intent)
    }
}
```

```
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)

        // if a request is sent
        if (requestCode == REQUEST_ENABLE_BLUETOOTH) {
            // if BT conn is ok
            if (resultCode == Activity.RESULT_OK) {

                if (m_bluetoothAdapter!!.isEnabled) {
                    // Bluetooth has been enabled
                    Log.i( tag: "bluetooth",  msg: "device's bluetooth is now on")
                } else {
                    // Bluetooth has been disabled
                    Log.i( tag: "bluetooth",  msg: "device's bluetooth is now off")
                }

            // if Bluetooth conn was cancelled for some reasong
            } else if (resultCode == Activity.RESULT_CANCELED) {
                Log.i( tag: "bluetooth",  msg: "Belutooth connecting was cancelled")
            }
        }
    }
}

```

*Фигури 7, 8 и 9, показващи кода от файл MainActivity.kt*

На фиг. 10, 11 и 12 е показан кодът от файл “TankControlActivity.kt”.

```
package com.example.phonetomodulebluetooth

import ...

class TankControlActivity : AppCompatActivity() {

    // static properties
    companion object {...}

    // UI + data init
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.tank_control_layout) // Set app UI

        // get MAC address of Tank bluetooth module from MainActivity
        m_address = intent.getStringExtra(MainActivity.EXTRA_ADDRESS)

        ConnectToDevice( c: this).execute() // this = Current context

        // left track stop
        left_track_stop.setOnClickListener {...}

        // left track forward
        left_track_forward.setOnClickListener {...}

        // left track backwards
        left_track_backwards.setOnClickListener {...}

        // right track stop
        right_track_stop.setOnClickListener {...}

        // right track foward
        right_track_forward.setOnClickListener {...}

        // right track backwards
        right_track_backwards.setOnClickListener {...}

```

```
80
81          // disconnect button, to close connection w/ curr device (Bluetooth module)
82          close_connection.setOnClickListener { it: View!
83              Log.i( tag: "connection",  msg: "closing connection. . . ")
84              disconnect()
85          }
86      }
87
88      // send commands via Bluetooth
89      private fun sendCommand(input: String) {
90          if (m_bluetoothSocket != null) {
91              try {
92                  m_bluetoothSocket!!.outputStream.write(input.toByteArray())
93              } catch (e: IOException) {
94                  e.printStackTrace()
95              }
96          }
97      }
98
99      // terminate Bluetooth connection
100     private fun disconnect() {
101         if (m_bluetoothSocket != null) {
102             try {
103                 m_bluetoothSocket!!.close()
104                 m_bluetoothSocket = null
105                 m_isConnected = false
106             } catch (e: IOException) {
107                 e.printStackTrace()
108             }
109         }
110         finish()
111     }
112
```

```
112
113     // connect to device via Bluetooth
114     private class ConnectToDevice(c: Context) : AsyncTask<Void, Void, String>() {
115         private var connectSuccess: Boolean = true
116         private val context: Context
117
118         init {
119             this.context = c
120         }
121
122         override fun onPreExecute() {
123             super.onPreExecute()
124         }
125
126         override fun doInBackground(vararg p0: Void?): String? {
127             try {
128
129                 // prep for connection, var assignment
130                 if (m_bluetoothSocket == null || !m_isConnected) {
131                     m_bluetoothAdapter = BluetoothAdapter.getDefaultAdapter()
132                     val device: BluetoothDevice = m_bluetoothAdapter.getRemoteDevice(m_address)
133                     m_bluetoothSocket = device.createInsecureRfcommSocketToServiceRecord(m_myUUID)
134                     BluetoothAdapter.getDefaultAdapter().cancelDiscovery()
135
136                     m_bluetoothSocket!!.connect() // attempt to connect to device
137
138                     // log after successful connection
139                     Log.i( tag: "connection",  msg: "connected to Bluetooth module")
140
141                 }
142             } catch (e: IOException) { // if connection fails
143                 connectSuccess = false
144                 e.printStackTrace()
145                 Log.i( tag: "connection",  msg: "connection failed")
146             }
147             return null
148         }
```

```
149
150         override fun onPostExecute(result: String?) {
151             super.onPostExecute(result)
152             if (!connectSuccess) {
153                 Log.i( tag: "connection",  msg: "couldn't connect")
154             } else {
155                 m_isConnected = true
156             }
157         }
158     }
159 }
160
```

*Фигури 10, 11 и 12, показващи кода от файл "TankControlActivity.kt".*

# 10. Заключение

Това е документацията на проекта ни за модул VIII на Националната програма “Обучение за ИТ кариера”.

За този проект беше реализирана електрическа схема и мобилно приложение за дистанционно контролиране чрез Bluetooth на постояннотоковите мотори, служещи за задвижване на количката.

Електическата схема се състои от микроконтролер Arduino, драйвер L298 и Bluetooth модул HC-05.

Мобилното приложение е написано за мобилната операционна система Android на език за програмиране Kotlin.

Известни проблеми:

- Недостатъчна мощност на моторите
- Кратък живот на батериите
- Понякога възникват неизвестни проблеми с връзката между микроконтролера и Bluetooth модула

# Приложение

За проекта са използвани следните компоненти:

- Arduino Uno R3
- L298 Motor Driver
- 3,3V DC Motor
- HC-05 Bluetooth module
- 2x 1kΩ resistors
- Lego

# 11. Използвана литература


- Документация на Arduino - https://www.arduino.cc/en/Reference/
- Документация на Kotlin - https://kotlinlang.org/docs/reference/
- Документация на Android - https://developer.android.com/guide
- Каталожна информация за L298 - https://cdn.instructables.com/ORIG/FCN/YABW/IHNTEND4/FCNYABWIHNTEND4.pdf
- Каталожна информация за HC-05 - http://www.electronicaestudio.com/docs/istd016A.pdf
- https://forum.arduino.cc/
- https://stackoverflow.com/