Proiect Informatică Aplicată

LED control using Bluetooth Classic

Studenți:
Asan Aura-Ingrid
Teodorescu Tania-Maria-Gabriela
Grupa 411A

Cuprins

1. Introducere

Programarea este dispunerea cronologică a unor mișcări, operații, acțiuni sau activități astfel încât în finalul perioadei să se realizeze o stare posibilă a unui sistem. Programarea este cunoscută ca activitate umană, dar există semnalmente că pot exista forme de programare naturale, fără intervenția omului cum sunt dispunerile proceselor genetice sau comportamentele dirijate instinctual la animale.

Programarea calculatorului este o activitate informatică de elaborare a produselor-program, a programelor (software) necesare activităților realizate cu ajutorul calculatorului. Programarea informatică conține următoarele subactivități: specificarea, proiectarea, implementarea, documentarea și întreținerea produsului program.

Informatica cuprinde domeniile legate de proiectarea, construcţia, evaluarea, utilizarea şi întreţinerea sistemelor de prelucrare automată a datelor, incluzând componentele hardware, software, elementele organizaţionale şi umane cu impactul lor în industrie, administraţie, comerţ etc.

Termenul informatică desemnează știința procesării sistematice a informației, în special a procesării cu ajutorul calculatoarelor. Termenul informatică provine din alăturarea cuvintelor „informație” și „matematică”. Alte surse susțin că provine din combinația „informație” și „automatică”.

Informatica se divide în următoarele domenii fundamentale:

- informatică teoretică
- informatică aplicată
- informatică tehnică

Pe lângă aceste trei domenii principale, mai există:

- inteligență artificială, considerată drept interdisciplinară, într-o anumită măsură de sine stătătoare.

În lucrarea de față, vom folosi o ramură a informaticii și anume, informatică aplicată. Acest proiect presupune un program ce controlează LED-uri prin Bluetooth Classic conform acțiunilor solicitate de aplicația mobilă, ProiectIA.

2. Considerente teoretice

- Modelul ESP32

ESP32 este o serie de sisteme cu costuri reduse, cu putere redusă, pe microcontrolere cu cip, cu WiFi integrat și Bluetooth Classic și Low-Energy. Seria ESP32 folosește fie un microprocesor Tensilica Xtensa LX6 în variante dual-core și single-core, microprocesor Xtensa LX7 dual-core sau un microprocesor RISC-V cu un singur nucleu și include comutatoare de antenă incorporate, amplificator de putere, amplificator de recepție cu zgomot redus, filtre și module de gestionare a puterii. ESP32 este creat și dezvoltat de Espressif Systems, o companie chineză cu sediul în Shanghai, și este fabricat de TSMC folosind procesul lor de 40 nm.

Caracteristicile ESP32 includ următoarele:

• Procesoare:
  - CPU: microprocesor Xtensa dual-core (sau single-core) LX6 pe 32 de biți, care funcționează la 160 sau 240 MHz și funcționează la până la 600 DMIPS
  - coprocesor de putere ultra joasă (ULP).

• Memorie: 320 KiB RAM, 448 KiB ROM

• Conectivitate wireless:
  - WiFi: 802.11 b/g/n
  - Bluetooth: v4.2 BR/EDR și BLE (partajează radioul cu Wi-Fi)

• Interfete periferice:
  - 34 pini programabili GPIO (General Purpose Input/Output)
  - 18 canale de conversie analog-digitală (ADC) cu rezoluție de 12 biți
  - 2 canale de conversie digital-analogică (DAC)cu rezoluție pe 8 biți
  - 16 canale de ieșire PWM (Pulse Width Modulation)
  - 10 senzori tactili intern capacitivi
  - 3 interfețe SPI (Serial Peripheral Interface)
  - 3 interfețe UART (Universal Asynchronous Receiver-Transmitter)
  - 2 interfețe I^2C (Inter-Integrated Circuit)
  - 2 interfețe I^2S (Inter-IC Circuit)
  - 1 interfață CAN 2.0 (Controller Area Network)
  - controler pentru conectarea dispozitivelor de stocare (carduri de memorie)
  - controler gazdă SD / SDIO / CE-ATA / MMC / eMMC
  - controler slave SDIO/SPI
  - interfață Ethernet MAC cu DMA dedicat și suport planificat IEEE 1588 Precision Time Protocol
  - CAN bus 2.0

- telecomandă cu infraroșu (TX/RX, până la 8 canale)
- motor PWM
- LED PWM (până la 16 canale)
- senzor cu efect Hall
- preamplificator analog de putere ultra joasă

• Securitate:

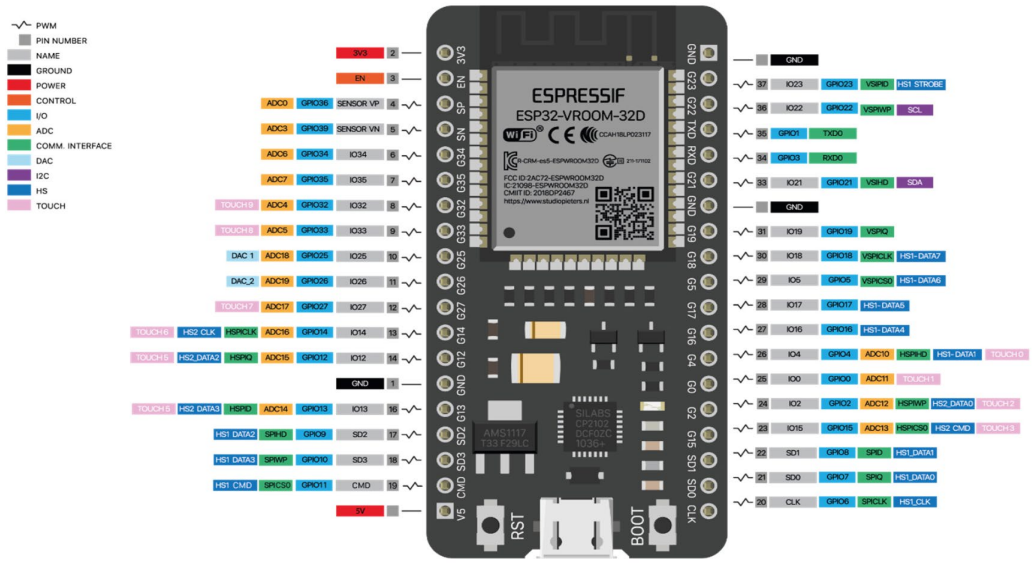
- toate funcțiile de securitate standard IEEE 802.11 sunt acceptate, inclusiv WPA, WPA2, WPA3 (în funcție de versiune) și infrastructura de autentificare și confidențialitate WLAN (WAPI)
- încărcare sigură
- criptare flash
- OTP de 1024 de biți, până la 768 de biți pentru clienți
- accelerare hardware criptografică: AES , SHA-2 , RSA , criptografie cu curbă eliptică (ECC), generator de numere aleatorii (RNG)

• Gestionare a energiei:

- regulator intern cu pierderi reduse
- domeniu de putere individual pentru RTC
- 5 µA curent de somn profund
- trezire de la întrerupere GPIO, temporizator, măsurători ADC, întrerupere senzor tactil capacitiv

Există mai multe familii de ESP32, cum ar fi: ESP32, ESP32-S2, ESP32-C3, ESP32-S3, ESP32-C6, ESP32-H2 etc.

Modelul ESP32 poate fi integrat în diferite plăci de dezvoltare care, în funcție de producător și tip, pot expune toți pinii/interfețele modulului sau doar o parte din ele. Pentru proiectul nostru am utilizat modulul ESP32-WROOM-32D.



- Protocoalele și metodele de comunicație utilizate

Protocoale de comunicație

Un protocol de comunicare este un sistem de reguli care permite două sau mai multe entități ale unui sistem de comunicații să transmită informații prin orice variație a unei cantități fizice. Acesta reprezintă un standard sau o convenție asupra modului de desfășurare a unui anumit lucru – în cazul rețelelor, protocoalele permit calculatoarelor să comunice între ele printr-un limbaj comun. Protocolul definește regulile, sintaxa, semantica și sincronizarea comunicării și posibilele metode de recuperare a erorilor. Protocoalele pot fi implementate prin hardware, software sau o combinație.

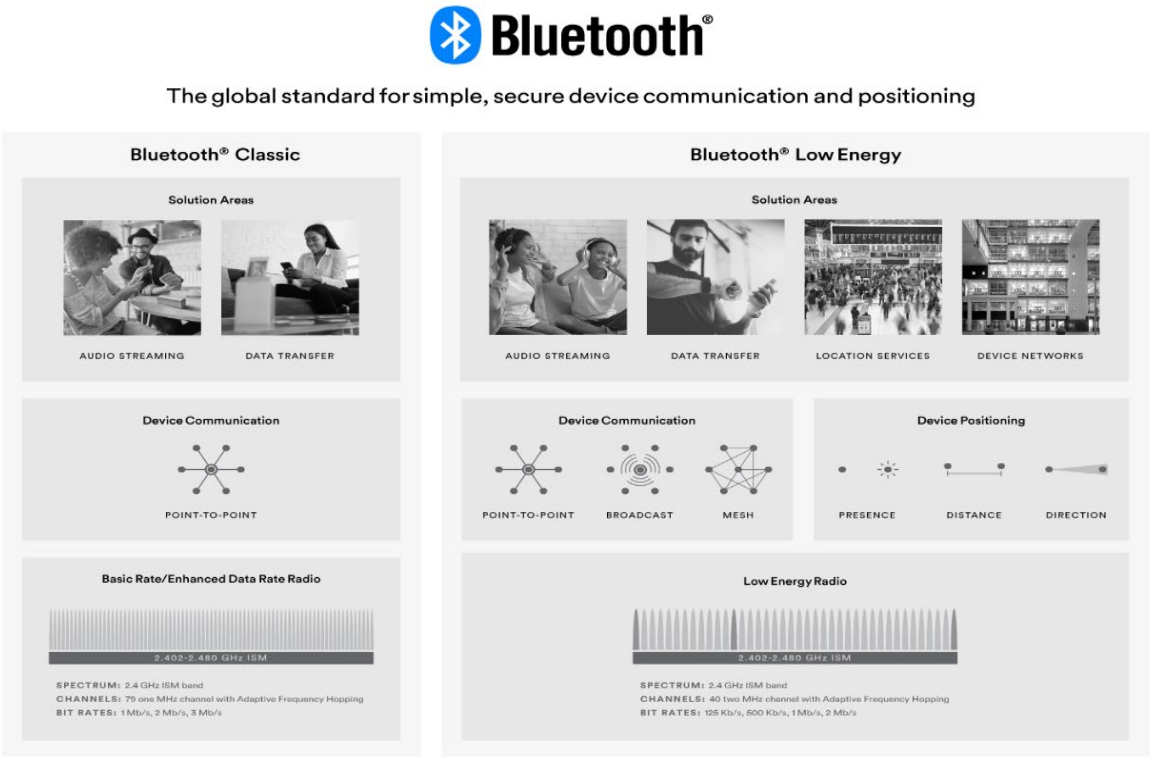
Sistemele de comunicare folosesc formate bine definite pentru schimbul de mesaje diverse. Fiecare mesaj are o semnificație exactă menită să obțină un răspuns dintr-o serie de răspunsuri posibile predeterminate pentru acea situație particulară. Comportamentul specificat este de obicei independent de modul în care urmează să fie implementat. Protocoalele de comunicare trebuie convenite de către părțile implicate. Pentru a ajunge la un acord, un protocol poate fi dezvoltat într-un standard tehnic. Un limbaj de programare descrie același lucru pentru calcule, așa că există o analogie strânsă între protocoale și limbajele de programare: protocoalele sunt pentru comunicare ceea ce limbajele de programare sunt pentru calcule. O formulare alternativă afirmă că protocoalele sunt pentru comunicare ceea ce algoritmii sunt pentru calcul.

Tehnologie Bluetooth

Bluetooth este o tehnologie de comunicație fără fir care permite transferul de date între dispozitive într-un mod convenabil și eficient. A fost dezvoltată pentru a facilita conectarea și interacțiunea între dispozitivele electronice personale, cum ar fi telefoane mobile, tablete, căști, tastaturi, imprimante, difuzoare și multe altele.

Câteva aspecte cheie legate de tehnologia Bluetooth:

1. Conectivitate fără fir: Bluetooth utilizează radiofrecvențe pentru a crea o conexiune directă între dispozitivele compatibile, eliminând necesitatea utilizării cablurilor.
2. Raza de acțiune: Raza de acțiune a conexiunii Bluetooth poate varia în funcție de versiunea Bluetooth și de mediul înconjurător. În general, raza de acțiune este de aproximativ 10 metri, dar tehnologia Bluetooth 5.x poate permite o acoperire mai mare de până la 100 de metri.
3. Lățime de bandă și viteză de transfer: Bluetooth utilizează o lățime de bandă de 2,4 GHz și oferă o viteză de transfer de date variabilă în funcție de versiunea Bluetooth. În funcție de implementare, vitezele tipice variază de la 1 Mbps (Bluetooth 1.x) până la 3 Mbps (Bluetooth 2.x și 3.x), sau chiar până la 24 Mbps (Bluetooth 3.0 + HS). Bluetooth Low Energy (BLE), introdus în Bluetooth 4.0, utilizează o lățime de bandă mai mică și consumă mai puțină energie, permițând conexiuni de lungă durată cu dispozitive alimentate cu baterie.



Bluetooth Classic

Radioul Bluetooth Classic, denumit și Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR), este un radio de putere redusă care transmite date pe 79 de canale în banda de frecvență industrială, științifică și medicală (ISM) fără licență de 2,4 GHz.

Caracteristici principale:

- Conectivitate fără fir: Bluetooth Classic permite comunicarea între dispozitive prin intermediul unei conexiuni radio pe o bandă de frecvență comună de 2,4 GHz.
- Raza de acțiune: În general, Bluetooth Classic poate asigura o acoperire într-un perimetru de aproximativ 10 metri, dar acest lucru poate varia în funcție de dispozitiv și de mediul înconjurător.
- Lățime de bandă: Bluetooth Classic utilizează o lățime de bandă de aproximativ 1 MHz, oferind o viteză de transfer de date de până la 3 Mbps.

	Bluetooth Classic
Frequency Band	2.4GHz ISM Band (2.402 – 2.480 GHz Utilized)
Channels	79 channels with 1 MHz spacing
Channel Usage	Frequency-Hopping Spread Spectrum (FHSS)
Modulation	GFSK, π/4 DQPSK, 8DPSK
Data Rate	EDR PHY (8DPSK): 3 Mb/s EDR PHY (π/4 DQPSK): 2 Mb/s BR PHY (GFSK): 1 Mb/s
Tx Power*	≤ 100 mW (+20 dBm)
Rx Sensitivity	≤-70 dBm
Data Transports	Asynchronous Connection-oriented Synchronous Connection-oriented
Communication Topologies	Point-to-Point (including piconet)
Positioning Features	None

JSON

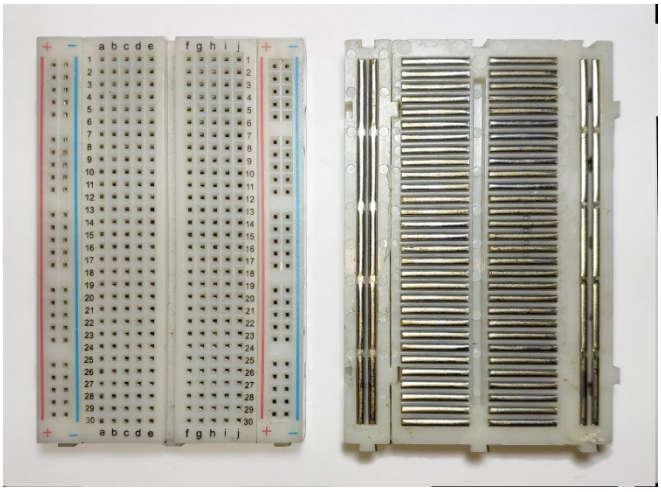
JSON este un acronim în limba engleză pentru *JavaScript Object Notation*, și este un format de reprezentare și interschimb de date între aplicații informatice. Este un format text, inteligibil pentru oameni, utilizat pentru reprezentarea obiectelor și a altor structuri de date și este folosit în special pentru a transmite date structurate prin rețea, procesul purtând numele de serializare.

Caracteristici si aspecte:

1. Sintaxa: JSON utilizează o sintaxă simplă și ușor de înțeles, care se bazează pe două structuri de bază: obiecte și tablouri.
    - Obiecte: Sunt reprezentate între acolade {} și constau din perechi cheie-valoare, unde cheile sunt de tip string și valoare poate fi de orice tip de date JSON valid;
    - Tablouri: Sunt reprezentate între paranteze [] și conțin o listă ordonată de valori JSON separate prin virgule.
2. Tipuri de date suportate: JSON suportă următoarele tipuri de date:
    - Șiruri de caractere (string): Secvență de caractere închisă între ghilimele duble;
    - Numere: Reprezentate sub forma valorilor numerice;
    - Boolean: Poate fi reprezentat ca true sau false;
    - Obiecte: Structuri complexe ce conțin perechi cheie-valoare, definite în interiorul acoladelor {};
    - Tablouri: Liste ordonate de valori, definite în interiorul parantezelor [];
    - Null: Reprezentat ca null, pentru a indica absența valorii.
3. Utilizare: JSON este utilizat pe scară largă în dezvoltarea aplicațiilor web și API-urilor pentru transmiterea și stocarea datelor. JSON este frecvent utilizat în comunicarea între client și server prin intermediul cererilor și răspunsurilor de tip JSON.
4. Avantaje:
    - Ușor de înțeles și de scris: Sintaxa JSON este intuitivă și ușor de citit și scris atât pentru oameni, cât și pentru mașini.
    - Compatibilitate: JSON este suportat de majoritatea limbajelor de programare și este independent de platformă.
5. Limitări:
    - JSON este limitat la reprezentarea structurilor de date și nu are suport nativ pentru alte concepte, cum ar fi datele binare sau referințele circulare.
    - Este necesară atenție în procesul de serializare și deserializare a datelor JSON pentru a evita vulnerabilități de securitate, precum atacurile de tipul "JSON injection".

- Componente utilizate

Breadboard



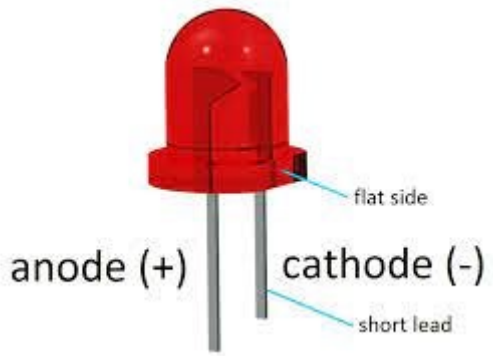
Breadboard-ul (placa de prototipare) este folosită pentru realizarea provizorie, rapidă și cu ușurință a circuitelor electrice. Toate componente folosite în acest proiect se conectează pe aceasta. Breadboard-ul constă dintr-o bază din plastic cu orificii mici. Orificiile sunt dispuse în benzi verticale și orizontale. Benzi verticale sunt de obicei conectate electric între ele, în timp ce benzi orizontale nu sunt conectate între ele.

Conexiunile dintre componente le-am realizat cu ajutorul cablurilor Dupont tată-tată, iar pentru a realiza conectarea dintre placa de dezvoltare și computer am folosit un cablu USB tip A - microUSB.

LED-uri

Un LED (*light-emitting diode*) este o diodă semiconductoare ce emite lumină la polarizarea directă a joncțiunii p-n. Efectul este o formă de electroluminescență.

Aspecte importante:



1. LED-urile sunt compuse dintr-un material semiconductor care este dopat cu impurități pentru a crea o zonă p și o zonă n. Când un curent electric trece prin LED, electronii și găurile se recombină în zona activă, eliberând fotoni de lumină.
2. LED-urile sunt disponibile într-o gamă largă de culori, de la roșu, portocaliu și galben la verde, albastru și violet. Culoarea luminii emise depinde de compoziția și de starea materialului semiconductor folosit, și poate fi în spectrul infraroșu, vizibil sau ultraviolet.

3. LED-urile au câteva limitări, cum ar fi sensibilitatea la tensiuni și curenti electrici mari, care pot duce la deteriorarea LED-ului.

Rezistențe

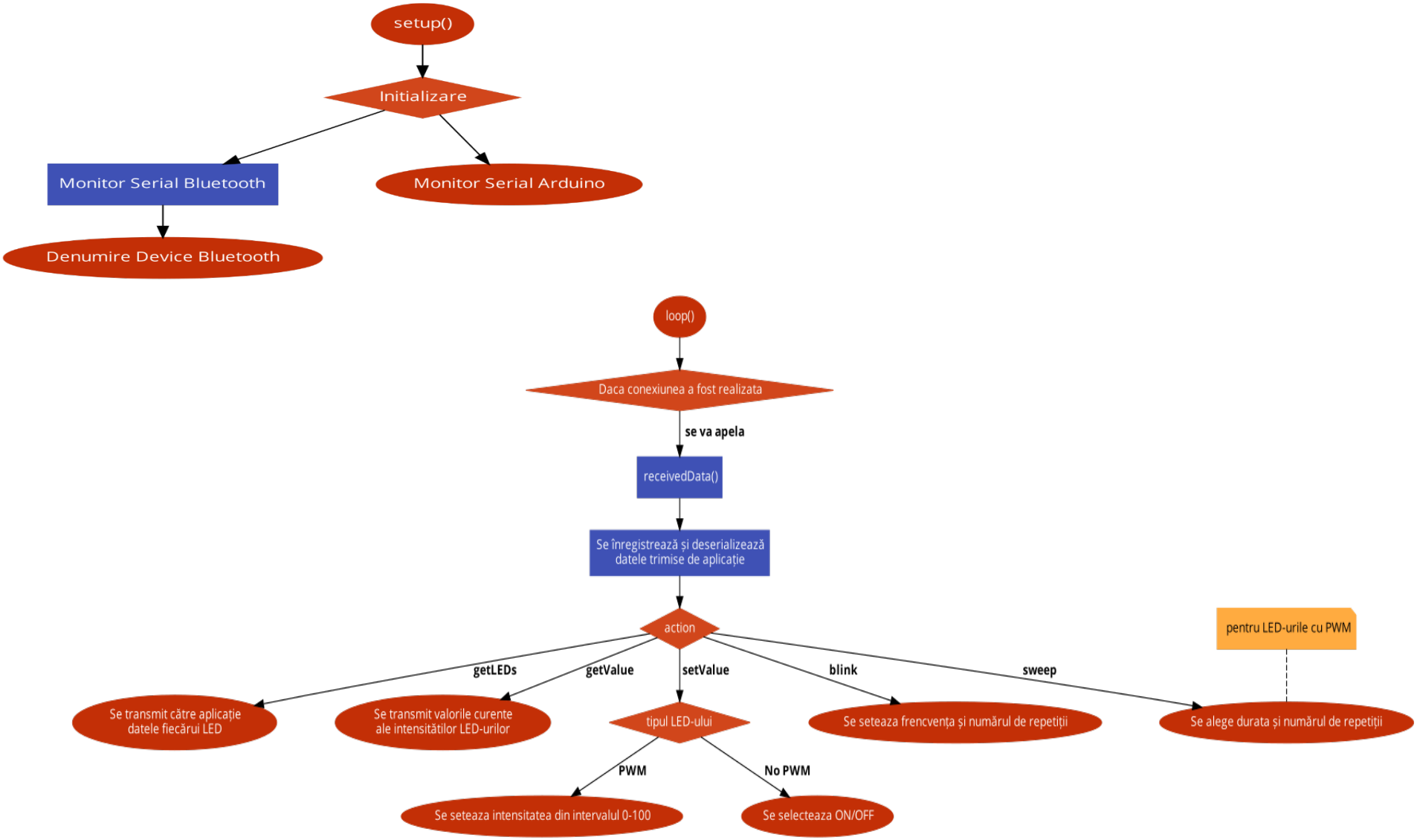
Rezistorii sunt componente electrice care limitează sau regulează fluxul de curent electric printr-un circuit. Valoarea rezistenței unui rezistor este definită ca raportul dintre tensiunea pe porțiunea respectivă de circuit U și curentul electric I ce parcurge acea porțiune de circuit. Pentru proiectul nostru am utilizat rezistențe de 330Ω ca să nu ardem LED-urile.

Caracteristici:

1. Structură și materiale: Rezistențele clasice sunt construite în jurul unei tije cilindrice sau unui substrat plat. Materialele utilizate pentru a crea rezistențele pot varia, cum ar fi carbonul, metale precum nichelul sau cuprul, sau compuși ceramici.

2. Pentru a identifica valoarea rezistenței, rezistențele clasice folosesc un cod de culoare. Culorile benziilor pe rezistență reprezintă cifre și factori de multiplicare, oferind o valoare numerică și o toleranță.
3. Rezistențele au o putere nominală specificată, exprimată în wați (W). Aceasta indică câtă putere electrică poate disipa rezistența în mod sigur fără a se supraîncălzi. Este important să se aleagă o rezistență cu o putere nominală adecvată pentru a evita deteriorarea sau arderea componentei.

4. Implementare



Funcția setup():

```
void setup()
{
  Serial.begin(115200);

  // Initialize BTC
  SerialBT.begin(btcServerName); // Bluetooth device name

  // Initialize the LED pins

  pinMode(LED4_PIN, OUTPUT);
  pinMode(LED5_PIN, OUTPUT);
  pinMode(LED6_PIN, OUTPUT);
  pinMode(LED7_PIN, OUTPUT);

  // Initialize the LED channels for PWM control
  ledcSetup(LED1_CHANNEL, 5000, 8);
  ledcSetup(LED2_CHANNEL, 5000, 8);
  ledcSetup(LED3_CHANNEL, 5000, 8);

  ledcAttachPin(LED1_PIN, LED1_CHANNEL);
  ledcAttachPin(LED2_PIN, LED2_CHANNEL);
  ledcAttachPin(LED3_PIN, LED3_CHANNEL);

  Serial.println("\nThe device started, now you can pair it with bluetooth!\n");
}
```

Se inițializează monitoarele seriale pentru Arduino și Bluetooth, alături de inițializarea pinilor pentru LED-urile cu PWM control și cele fără.

Funcția loop():

```
void loop()
{
  // Check available Bluetooth data and perform read from the app
  if (SerialBT.available())
  {
    Serial.println("\nConnected\n");
    receivedData();
  }
}
```

Când se conecteaza la Bluetooth va apela funcția receivedData().

Funcția receivedData():

```
void receivedData()
{
  if (SerialBT.available())
  {
    inputData = SerialBT.readStringUntil('\n');
  }
  Serial.println(inputData);
  StaticJsonDocument<1500> readDoc;
  DeserializationError err = deserializeJson(readDoc, inputData);
  if (err)
  {
    Serial.print(F("deserializeJson() failed with code "));
    Serial.println(err.f_str());
  }
  else
  {
    LED_Project(readDoc);
  }
  inputData = "";
}
```

În această funcție se va memora ceea ce se primește de la aplicație, se va deserializa și în cazul în care nu exista erori, se va apela funcția LED_Project().

Funcția LED_Project():

```
void LED_Project(StaticJsonDocument<1500> readDoc)
{
  String action = readDoc["action"];
  if (action == "getLEDs")
  {
    getLEDS(readDoc);
  }
  if (action == "getValue")
  {
    getValue(readDoc);
  }
  if (action == "setValue")
  {
    setValue(readDoc);
  }
  if (action == "blink")
  {
    blink(readDoc);
  }
  if (action == "sweep")
  {
    sweep(readDoc);
  }
}
```

Se verifică ce acțiune a fost selectată, apelându-se pentru fiecare funcția ei caracteristică.

Funcția getLEDs():

```
void getLEDS(StaticJsonDocument<1500> readDoc)
{
  StaticJsonDocument<1500> printDoc;

  String team = readDoc["teamId"];
  teamId=team;
  String output = "";

  // led 1
  printDoc["color"] = "white";
  printDoc["pin"] = LED1_PIN;
  printDoc["pwm"] = true;
  printDoc["channel"] = LED1_CHANNEL;
  printDoc["id"] = 1;
  printDoc["teamId"] = teamId;

  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
```

La prima apelare a acestei funcții se și memorează ID-ul echipei pentru a fi folosit în următoarele etape.

Se creează un document de tip JSON cu datele fiecărui LED, pe care îl serializăm și îl trimitem înapoi aplicației, aceasta metoda

Funcția getValue():

```
void getValue(StaticJsonDocument<1500> readDoc)
{
  StaticJsonDocument<1500> printDoc;
  int id = readDoc["id"];
  if (id == 1)
  {
    int value = 0;
    value = ledcRead(LED1_CHANNEL);
    printDoc["id"] = 1;
    printDoc["value"] = value;
    printDoc["teamId"] = teamId;

    String output = "";
    serializeJson(printDoc, output);
    Serial.println(output);
    Serial.println();
    SerialBT.println(output);
  }
 if (id == 4)
 {
   int value = 0;
   value = digitalRead(LED4_PIN);
   printDoc["id"] = 4;
   printDoc["value"] = value;
   printDoc["teamId"] = teamId;

   String output = "";
   serializeJson(printDoc, output);
   Serial.println(output);
   Serial.println();
   SerialBT.println(output);
 }
```

Se extrage id-ul LED-ului căruia dorim să îi aflam intensitatea, iar în funcție de tipul acestuia (PWM sau No PWM) vom folosi ledcRead(), respectiv digitalRead() pentru a afla intensitatea acestuia.

Funcția setValue():

```
void setValue(StaticJsonDocument<1500> readDoc)
{
  StaticJsonDocument<1500> printDoc;
  int id = readDoc["id"];
  if (id == 1)
  {
    float value = readDoc["value"];
    setLedPWM(LED1_CHANNEL, value);
    printDoc["id"] = 1;
    printDoc["value"] = value;
    printDoc["teamId"] = teamId;

    String output = "";
    serializeJson(printDoc, output);
    Serial.println(output);
    Serial.println();
    SerialBT.println(output);
  }
if (id == 4)
{
  int value = readDoc["value"];
  setLedNoPWM(LED4_PIN, value);
  printDoc["id"] = 4;
  printDoc["value"] = value;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
}
```

Se extrage id-ul LED-ului căruia dorim să îi setăm intensitatea, iar în funcție de tipul acestuia se vor apela funcțiile setLedPWM(), respectiv setLedNoPWM().

Funcția setLedPWM():

```
void setLedPWM(int channel_id, float duty_cycle)
{
  ledcWrite(channel_id, duty_cycle * 2.55);
}
```

Acestei funcții i se transmite canalul LED-ului și duty cycle-ul pe care îl dorim. O soluție pentru a converti valoarea din intervalul 0-100 pe care o introducem din aplicație în intervalul 0-255 este să înmulțim valoarea duty cycle-ului cu 2.55

Funcția setLedNoPWM():

```
void setLedNoPWM(int id, int value)
{
  digitalWrite(id, value);
}
```

Acestei funcții i se transmite pinul și valoarea intensității (care va fi 0 sau 1) pe care o dorim.

Funcția blink():

```
void blink(StaticJsonDocument<1500> readDoc)
{
  StaticJsonDocument<1500> printDoc;
  int id = readDoc["id"];
  if (id == 1)
  {
    float frequency = readDoc["frequency"];
    int number = readDoc["number"];
    blinkLedPWM(LED1_CHANNEL, frequency, number);
    printDoc["id"] = id;
    printDoc["value"] = 0;
    printDoc["blink_number"] = number;
    printDoc["teamId"] = teamId;

    String output = "";
    serializeJson(printDoc, output);
    Serial.println(output);
    Serial.println();
    SerialBT.println(output);
  }
if (id == 4)
{
  float frequency = readDoc["frequency"];
  int number = readDoc["number"];
  blinkLedNoPWM(LED4_PIN, frequency, number);
  printDoc["id"] = id;
  printDoc["value"] = 0;
  printDoc["blink_number"] = number;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
}
```

Se extrage id-ul LED-ului, iar în funcție de tipul acestuia se vor apela funcțiile blinkLedPWM(), respectiv blinkLedNoPWM().

Funcția blinkLedPWM():

```
void blinkLedPWM(int id, float frequency, int number)
{
  float time = 1.0 / frequency * 1000.0;
  for (int blink_number = 0; blink_number < number; ++blink_number)
  {
    setLedPWM(id, 100);
    delay((int)(time / 2));
    setLedPWM(id, LOW);
    delay((int)(time / 2));
  }
}
```

Acesteia i se transmite canalul LED-ului, frecvența și numărul de repetiții, folosindu-se de funcția setLedPWM() pentru a porni și stinge LED-ul.

Funcția blinkLedNoPWM():

```
void blinkLedNoPWM(int id, float frequency, int number)
{
  float time = 1.0 / frequency * 1000.0;
  for (int blink_number = 0; blink_number < number; ++blink_number)
  {
    setLedNoPWM(id, HIGH);
    delay((int)(time / 2));
    setLedNoPWM(id, LOW);
    delay((int)(time / 2));
  }
}
```

Acesteia i se transmite pinul LED-ului, frecvența și numărul de repetiții, folosindu-se de funcția setLedNoPWM() pentru a porni și stinge LED-ul.

Funcția sweep():

```
void sweep(StaticJsonDocument<1500> readDoc)
{
  StaticJsonDocument<1500> printDoc;
  int id = readDoc["id"];
  if (id == 1)
  {
    int duration = readDoc["duration"];
    int number = readDoc["number"];
    sweepLed(LED1_CHANNEL, duration, number);
    printDoc["id"] = 1;
    printDoc["value"] = 0;
    printDoc["sweep_number"] = number;
    printDoc["teamId"] = teamId;

    String output = "";
    serializeJson(printDoc, output);
    Serial.println(output);
    Serial.println();
    SerialBT.println(output);
  }
```

Aceasta este doar pentru LED-urile cu PWM.
Se extrage id-ul LED-ului, iar apoi se apelează funcția sweepLed() pentru fiecare caz.

Funcția sweepLed():

```
void sweepLed(int id, int duration, int number)
{
  int time = duration * 5;
  for (int k = 0; k < number; ++k)
  {
    for (int i = 1; i <= 100; ++i)
    {
      setLedPWM(id, i);
      delay(time);
    }
    for (int i = 99; i >= 0; --i)
    {
      setLedPWM(id, i);
      delay(time);
    }
  }
}
```

Primește canalul LED-ului, durata și numărul de repetiții.

Deoarece durata pe care o introducem este in secunde, dar funcția delay() utilizeaza milisecunde, variabila time (care este folosită pentru a face ca intentisatea LED-ului să rămână aceeași pentru un interval dat de timp) va fi egala cu valoarea duratei înmulțită cu 5. În cazul în care durata este de 1 secundă, îi va lua 5 × 100 (de iterații) = 500 de milisencunde să ajungă la intensitate maximă și 5 × 100 (de iterații) = 500 de milisecunde să ajungă înapoi la minim, luându-i 1000 de milisecunde, adică 1 secundă, să realizeze ciclul.

5. Concluzii

În final, acest proiect are multe aplicabilități în viața de zi cu zi, modelându-se perfect pe următoarele situații:

- Smart Home (controlarea LED-urilor din casă prin intermediul tehnologiei Bluetooth)
- Banda LED controlată prin Bluetooth.

În timpul realizării proiectului ne-am avântat în fascinanta lume JSON și am descoperit câte lucruri uimitoare se pot realiza cu ajutorul acestuia. JSON este esențial pentru un viitor inginer, studentul putând realiza mici fragmente dintr-un proiect mai complex.

Concret, despre proiectul nostru, cu ajutorul modulului ESP32 am reușit să implementăm un program care să controleze LED-urile prin intermediul aplicației mobile, folosind Bluetooth Classic.

6. Bibliografie

- https://ro.wikipedia.org/wiki/Programare
- https://en.wikipedia.org/wiki/Computer_programming
- https://ro.wikipedia.org/wiki/Informatic%C4%83
- http://retele.elth.ucv.ro/Duinea%20Adelaida/Informatica%20Aplicata/INFORMATICA%20APLICATA%20-%20suport%20de%20curs.pdf
- https://en.wikipedia.org/wiki/ESP32#QFN_packaged_chip_and_module
- Documentații PIA
- https://en.wikipedia.org/wiki/Communication_protocol
- https://www.slideshare.net/CarolinaStochitoiu/6-protocoale-de-retea
- https://en.wikipedia.org/wiki/Bluetooth
- https://www.bluetooth.com/learn-about-bluetooth/tech-overview/
- https://ro.wikipedia.org/wiki/JSON
- https://en.wikipedia.org/wiki/Breadboard
- https://ro.wikipedia.org/wiki/LED
- https://en.wikipedia.org/wiki/Electrical_resistance_and_conductance
- https://app.code2flow.com/

7. Anexe (codul complet)

```
#include <Arduino.h>
#include <BluetoothSerial.h>
#include <ArduinoJson.h>

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

#define btcServerName "ESP32_TANING"

// Define the LED pins
const int LED1_PIN = 15; // led1
const int LED2_PIN = 0;  // led2
const int LED3_PIN = 5;  // led3
const int LED4_PIN = 2;  // led4
const int LED5_PIN = 4;  // led5
const int LED6_PIN = 16; // led6
const int LED7_PIN = 17; // led7

// Define the LED channels for PWM control
const int LED1_CHANNEL = 0;  // led1
const int LED2_CHANNEL = 1;  // led2
const int LED3_CHANNEL = 2;  // led3
const int LED4_CHANNEL = -1; // led4
const int LED5_CHANNEL = -1; // led5
const int LED6_CHANNEL = -1; // led6
const int LED7_CHANNEL = -1; // led7

// Define the BluetoothSerial
BluetoothSerial SerialBT;

// Received data
String inputData = "";

// The receivedData function is called when data is available on Bluetooth (see loop function)
String teamId = "";

void setLedPWM(int channel_id, float duty_cycle)
{
  ledcWrite(channel_id, duty_cycle * 2.55);
}

void setLedNoPWM(int id, int value)
{
  digitalWrite(id, value);
}
```

```
void blinkLedNoPWM(int id, float frequency, int number)
{
 float time = 1.0 / frequency * 1000.0;
 for (int blink_number = 0; blink_number < number; ++blink_number)
 {
  setLedNoPWM(id, HIGH);
  delay((int)(time / 2));
  setLedNoPWM(id, LOW);
  delay((int)(time / 2));
 }
}

void blinkLedPWM(int id, float frequency, int number)
{
 float time = 1.0 / frequency * 1000.0;
 for (int blink_number = 0; blink_number < number; ++blink_number)
 {
  setLedPWM(id, 100);
  delay((int)(time / 2));
  setLedPWM(id, LOW);
  delay((int)(time / 2));
 }
}

void sweepLed(int id, int duration, int number)
{
 int time = duration * 5;
 for (int k = 0; k < number; ++k)
 {
  for (int i = 1; i <= 100; ++i)
  {
   setLedPWM(id, i);
   delay(time);
  }
  for (int i = 99; i >= 0; --i)
  {
   setLedPWM(id, i);
   delay(time);
  }
 }
}

void getLEDS(StaticJsonDocument<1500> readDoc)
{
 StaticJsonDocument<1500> printDoc;

 String team = readDoc["teamId"];
 teamId=team;
```

```
String output = "";

// led 1
printDoc["color"] = "white";
printDoc["pin"] = LED1_PIN;
printDoc["pwm"] = true;
printDoc["channel"] = LED1_CHANNEL;
printDoc["id"] = 1;
printDoc["teamId"] = teamId;

serializeJson(printDoc, output);
Serial.println(output);
Serial.println();
SerialBT.println(output);

// led2
printDoc["color"] = "blue";
printDoc["pin"] = LED2_PIN;
printDoc["pwm"] = true;
printDoc["channel"] = LED2_CHANNEL;
printDoc["id"] = 2;
printDoc["teamId"] = teamId;

output = "";
serializeJson(printDoc, output);
Serial.println(output);
Serial.println();
SerialBT.println(output);

// led3
printDoc["color"] = "yellow";
printDoc["pin"] = LED3_PIN;
printDoc["pwm"] = true;
printDoc["channel"] = LED3_CHANNEL;
printDoc["id"] = 3;
printDoc["teamId"] = teamId;

output = "";
serializeJson(printDoc, output);
Serial.println(output);
Serial.println();
SerialBT.println(output);

// led4
printDoc["color"] = "red";
printDoc["pin"] = LED4_PIN;
printDoc["pwm"] = false;
printDoc["channel"] = LED4_CHANNEL;
```

```
printDoc["id"] = 4;
printDoc["teamId"] = teamId;

output = "";
serializeJson(printDoc, output);
Serial.println(output);
Serial.println();
SerialBT.println(output);

// led5
printDoc["color"] = "yellow";
printDoc["pin"] = LED5_PIN;
printDoc["pwm"] = false;
printDoc["channel"] = LED5_CHANNEL;
printDoc["id"] = 5;
printDoc["teamId"] = teamId;

output = "";
serializeJson(printDoc, output);
Serial.println(output);
Serial.println();
SerialBT.println(output);

// led6
printDoc["color"] = "blue";
printDoc["pin"] = LED6_PIN;
printDoc["pwm"] = false;
printDoc["channel"] = LED6_CHANNEL;
printDoc["id"] = 6;
printDoc["teamId"] = teamId;

output = "";
serializeJson(printDoc, output);
Serial.println(output);
Serial.println();
SerialBT.println(output);

// led 7
printDoc["color"] = "white";
printDoc["pin"] = LED7_PIN;
printDoc["pwm"] = false;
printDoc["channel"] = LED7_CHANNEL;
printDoc["id"] = 7;
printDoc["teamId"] = teamId;

output = "";
serializeJson(printDoc, output);
Serial.println(output);
```

```
  Serial.println();
  SerialBT.println(output);
}

void getValue(StaticJsonDocument<1500> readDoc)
{
  StaticJsonDocument<1500> printDoc;
  int id = readDoc["id"];
  if (id == 1)
  {
    int value = 0;
    value = ledcRead(LED1_CHANNEL);
    printDoc["id"] = 1;
    printDoc["value"] = value;
    printDoc["teamId"] = teamId;

    String output = "";
    serializeJson(printDoc, output);
    Serial.println(output);
    Serial.println();
    SerialBT.println(output);
  }
  if (id == 2)
  {
    int value = 0;
    value = ledcRead(LED2_CHANNEL);
    printDoc["id"] = 2;
    printDoc["value"] = value;
    printDoc["teamId"] = teamId;

    String output = "";
    serializeJson(printDoc, output);
    Serial.println(output);
    Serial.println();
    SerialBT.println(output);
  }
  if (id == 3)
  {
    int value = 0;
    value = ledcRead(LED3_CHANNEL);
    printDoc["id"] = 3;
    printDoc["value"] = value;
    printDoc["teamId"] = teamId;

    String output = "";
    serializeJson(printDoc, output);
    Serial.println(output);
    Serial.println();
```

```
  SerialBT.println(output);
}
if (id == 4)
{
  int value = 0;
  value = digitalRead(LED4_PIN);
  printDoc["id"] = 4;
  printDoc["value"] = value;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
}
if (id == 5)
{
  int value = 0;
  value = digitalRead(LED5_PIN);
  printDoc["id"] = 5;
  printDoc["value"] = value;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
}
if (id == 6)
{
  int value = 0;
  value = digitalRead(LED6_PIN);
  printDoc["id"] = 6;
  printDoc["value"] = value;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
}
if (id == 7)
{
  int value = 0;
  value = digitalRead(LED7_PIN);
```

```
    printDoc["id"] = 7;
    printDoc["value"] = value;
    printDoc["teamId"] = teamId;

    String output = "";
    serializeJson(printDoc, output);
    Serial.println(output);
    Serial.println();
    SerialBT.println(output);
  }
}

void setValue(StaticJsonDocument<1500> readDoc)
{
  StaticJsonDocument<1500> printDoc;
  int id = readDoc["id"];
  if (id == 1)
  {
    float value = readDoc["value"];
    setLedPWM(LED1_CHANNEL, value);
    printDoc["id"] = 1;
    printDoc["value"] = value;
    printDoc["teamId"] = teamId;

    String output = "";
    serializeJson(printDoc, output);
    Serial.println(output);
    Serial.println();
    SerialBT.println(output);
  }
  if (id == 2)
  {
    float value = readDoc["value"];
    setLedPWM(LED2_CHANNEL, value);
    printDoc["id"] = 2;
    printDoc["value"] = value;
    printDoc["teamId"] = teamId;

    String output = "";
    serializeJson(printDoc, output);
    Serial.println(output);
    Serial.println();
    SerialBT.println(output);
  }
  if (id == 3)
  {
    float value = readDoc["value"];
    setLedPWM(LED3_CHANNEL, value);
```

```
  printDoc["id"] = 3;
  printDoc["value"] = value;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
 }
 if (id == 4)
 {
  int value = readDoc["value"];
  setLedNoPWM(LED4_PIN, value);
  printDoc["id"] = 4;
  printDoc["value"] = value;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
 }
 if (id == 5)
 {
  int value = readDoc["value"];
  setLedNoPWM(LED5_PIN, value);
  printDoc["id"] = 5;
  printDoc["value"] = value;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
 }
 if (id == 6)
 {
  int value = readDoc["value"];
  setLedNoPWM(LED6_PIN, value);
  printDoc["id"] = 6;
  printDoc["value"] = value;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
```

```
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
 }
 if (id == 7)
 {
  int value = readDoc["value"];
  setLedNoPWM(LED7_PIN, value);
  printDoc["id"] = 7;
  printDoc["value"] = value;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
 }
}

void blink(StaticJsonDocument<1500> readDoc)
{
 StaticJsonDocument<1500> printDoc;
 int id = readDoc["id"];
 if (id == 1)
 {
  float frequency = readDoc["frequency"];
  int number = readDoc["number"];
  blinkLedPWM(LED1_CHANNEL, frequency, number);
  printDoc["id"] = id;
  printDoc["value"] = 0;
  printDoc["blink_number"] = number;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
 }
 if (id == 2)
 {
  float frequency = readDoc["frequency"];
  int number = readDoc["number"];
  blinkLedPWM(LED2_CHANNEL, frequency, number);
  printDoc["id"] = id;
  printDoc["value"] = 0;
  printDoc["blink_number"] = number;
```

```
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
 }
 if (id == 3)
 {
  float frequency = readDoc["frequency"];
  int number = readDoc["number"];
  blinkLedPWM(LED3_CHANNEL, frequency, number);
  printDoc["id"] = id;
  printDoc["value"] = 0;
  printDoc["blink_number"] = number;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
 }
 if (id == 4)
 {
  float frequency = readDoc["frequency"];
  int number = readDoc["number"];
  blinkLedNoPWM(LED4_PIN, frequency, number);
  printDoc["id"] = id;
  printDoc["value"] = 0;
  printDoc["blink_number"] = number;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
 }
 if (id == 5)
 {
  float frequency = readDoc["frequency"];
  int number = readDoc["number"];
  blinkLedNoPWM(LED5_PIN, frequency, number);
  printDoc["id"] = id;
  printDoc["value"] = 0;
  printDoc["blink_number"] = number;
```

```
    printDoc["teamId"] = teamId;

    String output = "";
    serializeJson(printDoc, output);
    Serial.println(output);
    Serial.println();
    SerialBT.println(output);
   }
   if (id == 6)
   {
    float frequency = readDoc["frequency"];
    int number = readDoc["number"];
    blinkLedNoPWM(LED6_PIN, frequency, number);
    printDoc["id"] = id;
    printDoc["value"] = 0;
    printDoc["blink_number"] = number;
    printDoc["teamId"] = teamId;

    String output = "";
    serializeJson(printDoc, output);
    Serial.println(output);
    Serial.println();
    SerialBT.println(output);
   }
   if (id == 7)
   {
    float frequency = readDoc["frequency"];
    int number = readDoc["number"];
    blinkLedNoPWM(LED7_PIN, frequency, number);
    printDoc["id"] = id;
    printDoc["value"] = 0;
    printDoc["blink_number"] = number;
    printDoc["teamId"] = teamId;

    String output = "";
    serializeJson(printDoc, output);
    Serial.println(output);
    Serial.println();
    SerialBT.println(output);
   }
}

void sweep(StaticJsonDocument<1500> readDoc)
{
  StaticJsonDocument<1500> printDoc;
  int id = readDoc["id"];
  if (id == 1)
  {
```

```
  int duration = readDoc["duration"];
  int number = readDoc["number"];
  sweepLed(LED1_CHANNEL, duration, number);
  printDoc["id"] = 1;
  printDoc["value"] = 0;
  printDoc["sweep_number"] = number;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
 }
 if (id == 2)
 {
  int duration = readDoc["duration"];
  int number = readDoc["number"];
  sweepLed(LED2_CHANNEL, duration, number);
  printDoc["id"] = 2;
  printDoc["value"] = 0;
  printDoc["sweep_number"] = number;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
 }
 if (id == 3)
 {
  int duration = readDoc["duration"];
  int number = readDoc["number"];
  sweepLed(LED3_CHANNEL, duration, number);
  printDoc["id"] = 3;
  printDoc["value"] = 0;
  printDoc["sweep_number"] = number;
  printDoc["teamId"] = teamId;

  String output = "";
  serializeJson(printDoc, output);
  Serial.println(output);
  Serial.println();
  SerialBT.println(output);
 }
}
```

```
void LED_Project(StaticJsonDocument<1500> readDoc)
{
 String action = readDoc["action"];
 if (action == "getLEDs")
 {
  getLEDS(readDoc);
 }
 if (action == "getValue")
 {
  getValue(readDoc);
 }
 if (action == "setValue")
 {
  setValue(readDoc);
 }
 if (action == "blink")
 {
  blink(readDoc);
 }
 if (action == "sweep")
 {
  sweep(readDoc);
 }
}

void receivedData()
{
 if (SerialBT.available())
 {
  inputData = SerialBT.readStringUntil('\n');
 }
 Serial.println(inputData);
 StaticJsonDocument<1500> readDoc;
 DeserializationError err = deserializeJson(readDoc, inputData);
 if (err)
 {
  Serial.print(F("deserializeJson() failed with code "));
  Serial.println(err.f_str());
 }
 else
 {
  LED_Project(readDoc);
 }
 inputData = "";
}

void setup()
{
```

```
  Serial.begin(115200);

  // Initialize BTC
  SerialBT.begin(btcServerName); // Bluetooth device name

  // Initialize the LED pins

  pinMode(LED4_PIN, OUTPUT);
  pinMode(LED5_PIN, OUTPUT);
  pinMode(LED6_PIN, OUTPUT);
  pinMode(LED7_PIN, OUTPUT);

  // Initialize the LED channels for PWM control
  ledcSetup(LED1_CHANNEL, 5000, 8);
  ledcSetup(LED2_CHANNEL, 5000, 8);
  ledcSetup(LED3_CHANNEL, 5000, 8);

  ledcAttachPin(LED1_PIN, LED1_CHANNEL);
  ledcAttachPin(LED2_PIN, LED2_CHANNEL);
  ledcAttachPin(LED3_PIN, LED3_CHANNEL);

  Serial.println("\nThe device started, now you can pair it with bluetooth!\n");
}

void loop()
{
  // Check available Bluetooth data and perform read from the app
  if (SerialBT.available())
  {
    Serial.println("\nConnected\n");
    receivedData();
  }
}
```