	Instytut Informatyki Politechniki Śląskiej Zespół Mikroinformatyki i Teorii Automatów Cyfrowych			
Rok akademicki:	**Rodzaj studiów*: SSI/NSI/NSM**	**Przedmiot (Języki Asemblerowe/SMiW):**	**Grupa**	**Sekcja**
2020/2021	**SSI**	**SMiW**	**3**	**1**
Imię:	**Mateusz**	**Prowadzący**: OA/JP/KT/GD/BSz/GB	**JP**	
Nazwisko:	**Płonka**			

Raport końcowy

Temat projektu:

Wzmacniacz alarmu

Data oddania: dd/mm/rrrr	**12/02/2021**

1. Opis założeń

Zadaniem projektu jest obserwacja diod sprzętów AGD przy pomocy fotorezystorów oraz reakcje na zmianę światła w postaci alarmu oraz powiadomień na konsoli Bluetooth.

Po dostarczeniu zasilania mikrokontroler wydaje dźwięk sygnalizujący poprawną konfigurację po czym rozpoczyna obserwację światła poprzez trzy podpięte i odpowiednio skonfigurowane programowo fotorezystory. Domyślnie na starcie aktywowany jest sensor nr 1, pozostałe odbierają światło, ale nie aktywują alarmu. Konfigurację można zmienić w czasie rzeczywistym przy pomocy konsoli Bluetooth, która umożliwia wyłączenie/włączenie odpowiedniego czujnika w zależności od potrzeb oraz wyświetlenie danych odbieranych przez system.

W przypadku spełnienia założeń co do poziomów światła określonych dla danych czujników (na przykład dioda na pralce przestanie świecić → pranie skończone), aktywowany zostaje alarm dźwiękowy oraz dostarczane jest powiadomienie na konsolę. Tryb alarmu można wyłączyć poprzez wciśnięcie przycisku znajdującego się na płytce lub używając terminala Bluetooth. Układ po wyjściu z alarmu, wraca do poprzedniego stanu.

2. Analiza zadania

Dobór elementów:

a) **Mikrokontroler – ESP32**
Wybrałem Esp-32 Wroom wyposażony w moduły Wi-FI oraz BT. Pomimo gabarytów oraz ceny uważam, że jest wart uwagi ze względu na rozwiązania, które umożliwia. Wbudowany moduł BT pozwoli na przystępną kontrolę, łączenie oraz zarządzanie z komponentami jak np. głośnik czy prosta aplikacja kontrolna na androida.
Napięcie zasilania 3,3V-5V, idealne dla wykorzystywanych komponentów w dodatku z wbudowanym konwerterem napięcia oraz portem micro-usb co pozwoli na uniwersalne i komfortowe wykorzystywanie urządzenia przez każdego i w każdym domu. Dodatkowo wbudowana pamięć flash o wartości 514kb idealnie posłuży do przechowywania krótkiego (2sek) pliku audio do odtworzenia w pętli po rozpoznaniu sygnału.

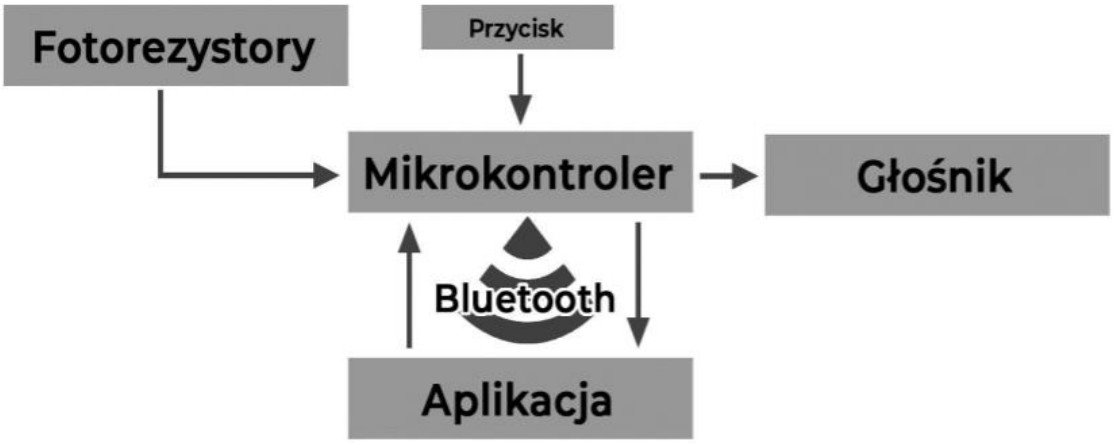
b) **Wzmacniacz audio**
Element jest wykorzystany do przetwarzania sygnału cyfrowego, nagranego na pamięć mikroprocesora sygnału dźwiękowego w postaci dzwonka, na dźwięk w postaci analogowej odtwarzany następnie przez głośnik podpięty do układu. Zdecydowałem się na MAX99357, pomimo dostawy z Chin, ze względu mniejszą objętość, dodatkowe elementy w zestawie oraz napięcie wejściowe 5V przy minimum 8V w konkurencyjnym elemencie.

c) **Fotorezystory i rezystory**
Trzy oddzielnie działające fotorezystory wykorzystane zostały w celu odczytywania zapalonej bądź zgaszonej diody sprzętu AGD. Możliwość wyłączania odpowiedniej komórki oraz reakcji na włączone lub wyłączone światło jest możliwa do zmiany software'owo w aplikacji. Rezystory 10kOhm wykorzystane zostały do utworzenia prostego dzielnika napięcia w celu wytworzenia sygnału cyfrowego.

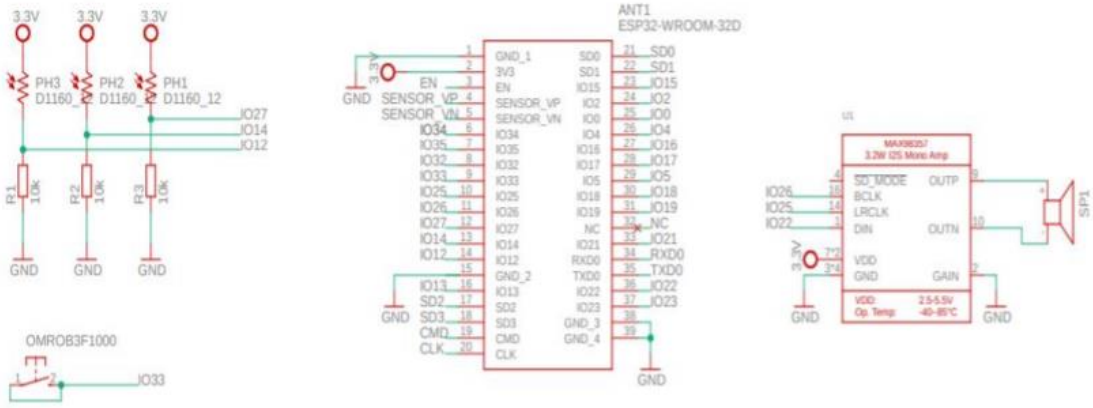
d) **Przycisk wyłączający alarm**
Dodatkowa opcja (poza aplikacją) manualnego wyłączenia sygnału dźwiękowego po rozpoznaniu końca pracy sprzętu AGD. Wykorzystany został przycisk **B3F-1000S.**

3. Specyfikacja wewnętrzna

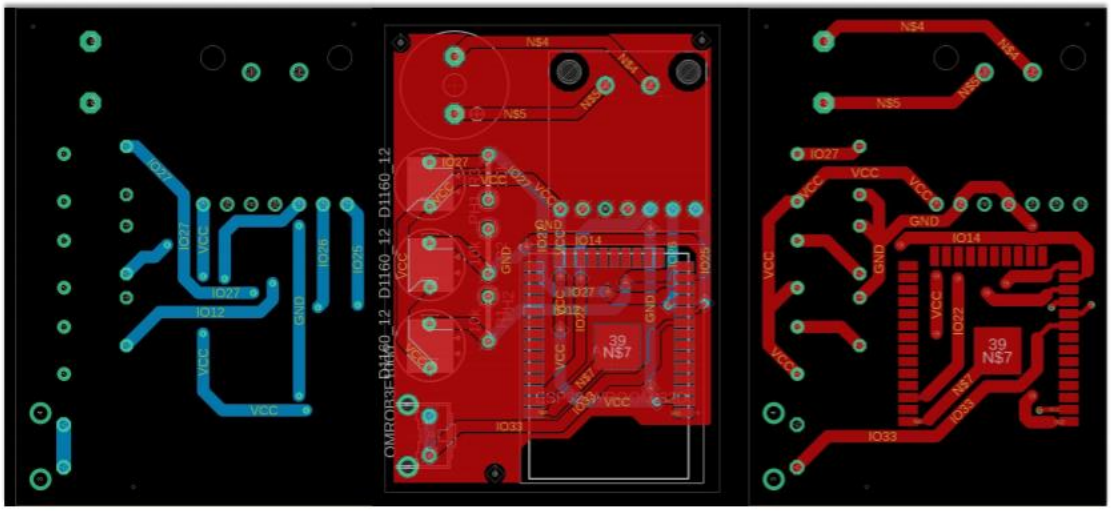
a) Schemat blokowy układu



b) Schemat ideowy układu



c) Schemat płytki PCB



d) Lista wykorzystanych elementów (wraz z kosztorysem)

Komponent	**Nazwa elementu**	**Cena [PLN]**
Mikroprocesor	ESP32-DEVKITC-32D (KIT)	50,68
Dekoder stereo	MAX98357A	24,90
Fotorezystory	PGM1200 THT	13,53
Przycisk	B3F1000S OMRON OCB THT	1,51
Rezystory	CFR0W4J0103A50 ROYAL 10kΩ	21,38
Płytka uniwersalna	PP-UM26	24,10
Dostawa		~10
		Suma: 146,10

e) Algorytm oprogramowania urządzenia

Część programowa projektu składa się z dwóch plików: **Photores.ino** zawierająca logikę działania programu oraz **sample.h,** w którym znajduje się zdekodowany do poziomu hex dźwięk alarmu w rozszerzeniu .flac.

Funkcja **void setup()** zawiera standardową inicjalizację oraz utworzenie obiektów zarządzających dźwiękiem.

Funkcja **void loop()** obsługuje kolejno: odczyt przycisku resetu, odczyt danych z konsoli bluetooth i reakcje na nie, obsługę trybu alarmu, odczyt danych z sensorów (kolejno 1,2,3) i reakcje na nie.

Photores.ino

```
#include <Arduino.h>
#include <AudioOutputI2S.h>
#include <AudioFileSourcePROGMEM.h>
#include <AudioGeneratorFLAC.h>

#include "sample.h"
#include "BluetoothSerial.h"

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

//Declaration of BT and audio export objects
BluetoothSerial SerialBT;
AudioOutputI2S *out;
AudioFileSourcePROGMEM *file;
AudioGeneratorFLAC *flac;

//Declaration of ports for sensors and reset pin
const int sens1Pin = 34;
const int sens2Pin = 35;
const int sens3Pin = 32;
const int resetPin = 19;
```

```
//Storage for sensors value
int sens1Value = 0;
int sens2Value = 0;
int sens3Value = 0;

String message = "";      //Stores message passed by BT
bool loundry = false;     //True if one of three sensors gets desired effect
bool isPlaying = false;   //For looping sound while alarm

//Disable/Enable sensors
bool input1 = true;
bool input2 = false;
bool input3 = false;
bool ShowData = false;  //True - Print sensors values on BT console


/*SETUP*/
void setup()
{
    Serial.begin(115200);
    pinMode(resetPin, INPUT_PULLUP);    // sets the digital pin 7 as input
    delay(100);

    //Audio objects configuration
    audioLogger = &Serial;
    file = new AudioFileSourcePROGMEM( sample_flac, sizeof(sample_flac) );
    out = new AudioOutputI2S();
    out->SetGain(0.2);
    flac = new AudioGeneratorFLAC();
    flac->begin(file, out);

    //Define BT name
    SerialBT.begin("Washing machine detector");
    Serial.println("Setup done");
}


/*LOOP*/
void loop()
{
   if(!flac->loop()){
    flac->stop();
    isPlaying = false;
   }


  //Reset button reaction
  if(!digitalRead(resetPin)) {

    //Recreate sound managment object for reset
    delete file;
    delete out;
    delete flac;
    file = new AudioFileSourcePROGMEM( sample_flac, sizeof(sample_flac) );
    out = new AudioOutputI2S();
    out->SetGain(0.2);
    flac = new AudioGeneratorFLAC();
    flac->begin(file, out);

    //Clear all bool variables
    input1 = false;
    input2 = false;
    input3 = false;
    isPlaying = false;
    loundry = false;
  }
```

```
  //Reading from Bluetooth (if avaible)
  if (SerialBT.available()){

    //Save input
    char incomingChar = SerialBT.read();

    //Convert to string variable if input exists
    if (incomingChar != '\n'){
      message += String(incomingChar);
    }
    else{
      message = "";
    }
    Serial.write(incomingChar);
  }


  // Check received message and control output accordingly
  //Turn of sensor 1 (/input1)
  if (message == "input1"){
      if(input1){
        input1 = false;
        SerialBT.println("Sensor 1 off!");
      }
      else{
        input1= true;
        SerialBT.println("Sensor 1 on!");
      }
  }

  //Turn of sensor 2 (/input2)
  else if(message == "input2"){
    if(input2){
      input2 = false;
      SerialBT.println("Sensor 2 off!");
    }
    else{
      input2= true;
      SerialBT.println("Sensor 2 on!");
    }
  }

  //Turn of sensor 3 (/input3)
  else if(message == "input3"){
    if(input3){
      input3 = false;
      SerialBT.println("Sensor 3 off!");
    }
    else{
      input3= true;
      SerialBT.println("Sensor 3 on!");
    }
  }

  //Respond on stop input when no alarm
  else if(message == "stop"){
    SerialBT.println("No alarm to finish!");
  }

  //Show current data of sensors (/ShowData)
  else if(message == "ShowData"){
      ShowData = true;
  }


  //Respond to alarm (loops sound)
  if(loundry){

    //If no sound executed, play again
```

```
    if(!isPlaying){
      //Set playing on true
      isPlaying = true;

      //Recreate sound object for reset (I could have place this in outer function)
      delete file;
      delete out;
      delete flac;
      file = new AudioFileSourcePROGMEM( sample_flac, sizeof(sample_flac) );
      out = new AudioOutputI2S();
      out->SetGain(0.2);
      flac = new AudioGeneratorFLAC();
      flac->begin(file, out);
    }

    //Reading from Bluetooth (if avaible)
    if (SerialBT.available()){

      //Save input
      char incomingChar = SerialBT.read();

      //Convert to string variable if input exists
      if (incomingChar != '\n'){
        message += String(incomingChar);
      }
      else{
        message = "";
      }
      Serial.write(incomingChar);
    }

    // Stop alarm and clear bool variables
    if (message =="stop"){
      SerialBT.println("Alarm turned off!");
      SerialBT.println("Sensor 1 off!");
      SerialBT.println("Sensor 2 off!");
      SerialBT.println("Sensor 3 off!");
      loundry = false;
      input1 = false;
      input2 = false;
      input3 = false;
    }
    delay(20);
  }


  /*SENSOR 1*/
  sens1Value = analogRead(sens1Pin);  //Get sensor data

  //Value over limit -> no sensor
  if(sens1Value > 4090){
    Serial.println("Sensor 1: No Sensor!!!");
    if(ShowData) {SerialBT.println("Sensor 1: No Sensor!!!");}
  }

  //Value under 1500 -> led has been turned on, react...
  else if(sens1Value < 1500){
        //printing info on console and bt console
        Serial.println("Sensor 1: "+String(sens1Value));
        if(ShowData) {SerialBT.println("Sensor 1: "+String(sens1Value));}

        //...but only if this sensor is avaible
        if(!loundry && input1)
        {
          loundry = true;
          SerialBT.println("Task done! (1)");
        }
  }
  //Rest of scenerios that aren't necessery right now
```

```
  else{
     Serial.println("Sensor 1: "+String(sens1Value));
     if(ShowData) {SerialBT.println("Sensor 1: "+String(sens1Value));}
  }

  //Print in Bt console if sensor is on or off
  if(ShowData){
   if(input1){
      SerialBT.println("Sensor 1: On");
    }
   else{
      SerialBT.println("Sensor 1: Off");
    }
  }


  /*SENSOR 2*/
  sens2Value = analogRead(sens2Pin);  //Get sensor data

  //Value over limit -> no sensor
  if(sens2Value > 4090){
      Serial.println("Sensor 2: No Sensor!!!");
      if(ShowData) {SerialBT.println("Sensor 2: No Sensor!!!");}
  }

  //Value between 4090 and 3000 to detect lack of light...
  else if(sens2Value > 3000){
        //printing info on console and bt console
        Serial.println("Sensor 2: "+String(sens2Value));
        if(ShowData) {SerialBT.println("Sensor 2: "+String(sens2Value));}

        //...but only if this sensor is avaible
        if(!loundry && input2)
        {
          loundry = true;
          SerialBT.println("Task done! (2)");
        }
  }

  //Rest of scenerios that aren't necessery right now
  else{
    Serial.println("Sensor 2: "+String(sens2Value));
    if(ShowData) {SerialBT.println("Sensor 2: "+String(sens2Value));}
  }

  //Print in Bt console if sensor is on or off
  if(ShowData){
   if(input2){
      SerialBT.println("Sensor 2: On");
    }
   else{
      SerialBT.println("Sensor 2: Off");
    }
  }


  /*SENSOR3 (not configured)*/
  sens3Value = analogRead(sens3Pin);   //Get sensor data

  //Value over limit -> no sensor
  if(sens3Value > 4090)
  {
    Serial.println("Sensor 3: No Sensor!!!");
    if(ShowData) {SerialBT.println("Sensor 3: No Sensor!!!");}
  }
  else
  {
    Serial.println("Sensor 3: "+String(sens3Value));
    if(ShowData) {SerialBT.println("Sensor 3: "+String(sens3Value));}
```

```
  }

  //Print in Bt console if sensor is on or off
  if(ShowData){
   if(input3){
      SerialBT.println("Sensor 3: On");
    }
   else{
      SerialBT.println("Sensor 3: Off");
    }
  }

  Serial.println("-------------------------");
  ShowData = false; //Clear show data variable after printing on Bt console
}
```

4. Specyfikacja zewnętrzna

a) **Opis funkcji wykorzystanych elementów**

1. **Mikrokontroler:** służy do sterowania całym układem, zarządzania jego logiką, przechowywania w pamięci programu oraz efektów dźwiękowych do odtworzenia.
2. **Fotorezystory:** odbierają natężenie światła i interpretują je w przedziale [0, 4090] gdzie 0 oznacza możliwie najjaśniejszy sygnał do odebrania.
3. **Głośnik:** Powiadamia użytkownika o zakończeniu zadania i konieczności wyciągnięcia prania.
4. **Przycisk:** służy do ręcznego wyłączenia układu z trybu alarmu.

b) **Reakcja oprogramowania na zdarzenie**

Gdy do układu nie jest doprowadzone zasilanie, żadna z jego funkcjonalności nie jest dostępna dla użytkownika. Jak wiemy układ zasilany działa o wiele lepiej. Po dostarczeniu zasilania poprzez post micro USB układ aktywuje się, co sygnalizuje pojedynczym dźwiękiem.

Fotorezystory odbierają światło z otoczenia i interpretują je w postaci liczby w przedziale [0, 4090] gdzie „0” oznacza możliwie najjaśniejszy możliwy sygnał do odebrania.

Do skonfigurowania urządzenia należy użyć telefonu/komputera z terminalem bluetooth i aktywować sensory których przeznaczenie jest nam aktualnie potrzebne (każda wszczyna alarm w różnym paśmie światła). Domyślnie aktywowany jest tylko sensor nr 1.

Gdy dany sensor wykryje odpowiednie światło, wywoła się tryb alarmu w którym z głośnika wydawany jest dźwięk oraz wysyłana jest wiadomość na konsolę użytkownika. Po resecie (zdalnie lub poprzez przycisk), układ wraca do stanu 0 z wyłączonymi wszystkimi sensorami.

c) **Instrukcja obsługi dla użytkownika**

1. Podłącz układ do prądu i zamontuj go w sprzęcie AGD wybierając odpowiedni fotorezystor:
    - nr 1 -> reakcja na pojawienie się światła
    - nr 2 -> reakcja na zniknięcie światła
    - nr 3 -> możliwy do własnej konfiguracji

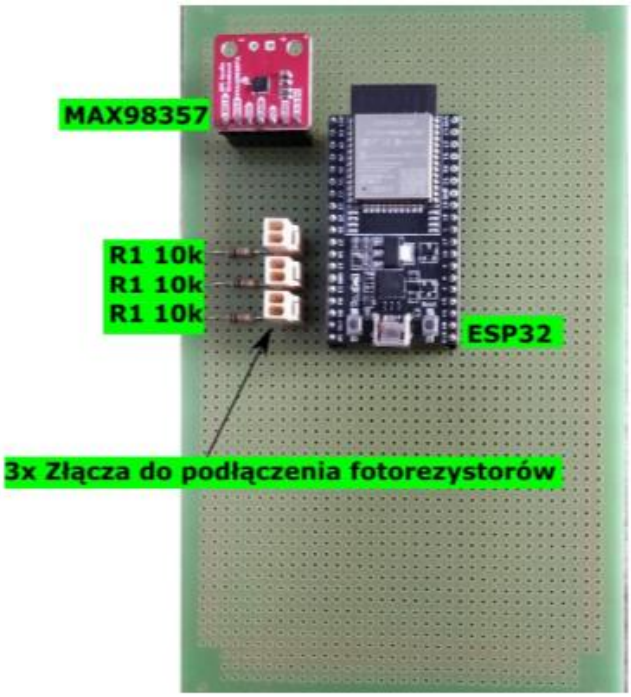
2. Połącz układ z innym urządzeniem poprzez bluetooth i skonfiguruj odpowiednie sensory dla poprawnego działania:
   Komendy
   - **stop –** zatrzymaj alarm
   - **input(1-3)** – aktywuje/dezaktywuje dany sensor
   - **ShowData** – wyświetl informacje na temat aktualnie odbieranych wartości oraz stanu danych sensorów

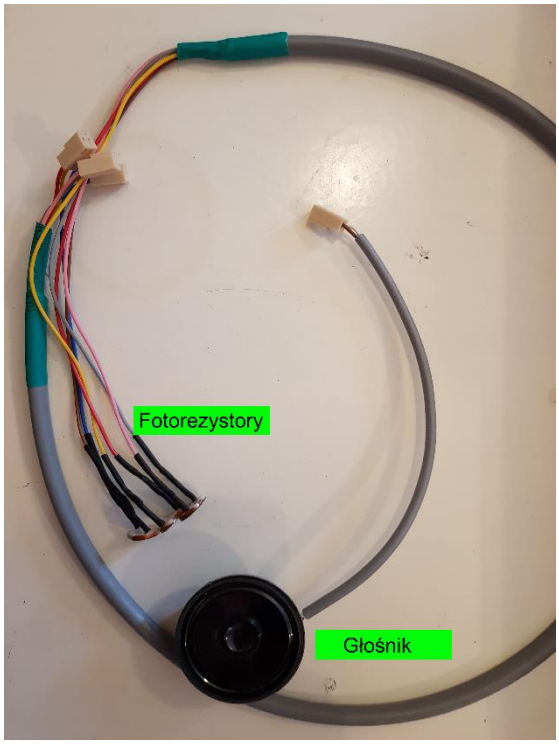
3. Gdy dojdzie do spełnienia określonych warunków układ powiadamia użytkownika o zakończeniu pracy sprzętu, na przykład pralki, bądź piekarnika za pomocą sygnału dźwiękowego oraz powiadomienia poprzez Bluetooth, gdy podpięte jest urządzenie zdalne.

4. Sygnał dźwiękowy wyłącza się przyciskiem lub komendą, po czym układ resetuje się i wraca do stanu „0” gdzie każdy sensor jest zdezaktywowany.

5. By ponownie skorzystać z urządzenia należy ponownie aktywować pożądane sensory

d) **Opis złączy i okablowania**





e) **Opis montażu układu**
Wszystkie elementy, zamówione na stronie tme.eu, zostały wlutowane w płytkę uniwersalną. Mikroprocesor, dekoder stereo oraz pomniejsze części jak rezystory czy złącza zostały wlutowane na górnej części płytki, pozostałe czyli głośnik i okablowane fotorezystory podpinane są do wspomnianych wcześniej portów.

f) **Sposób programowania układu**
Układ został zaprogramowany bez udziału programatora dzięki możliwości jaką udziela ESP32 czyli wprowadzenia danych bezpośrednio po micro USB. Projekt został napisany w języki C w środowisku Arduino. Do obsługi dźwięku zostały wykorzystane biblioteki **AudioOutputI2S.h, AudioFileSourcePROGMEM.h, AudioGeneratorFLAC.h** a do bluetootha **BluetoothSerial.h.**

g) **Uruchamianie oraz testowanie**
Ze względu na brak innych modułów jak Arduino, projekt był testowany już bezpośrednio na gotowym projekcie. Został on wielokrotnie sprawdzony pod względem wielu pokrętnych scenariuszy, które mogą się wydarzyć podczas obsługi sprzętu oraz przez osoby trzecie, nie mające nigdy kontaktu z mikroprocesorami, dla sprawdzenia przystępności obsługi układu.

5. Wnioski z przebiegu pracy

a) **Testowanie poprawności działania urządzenia**
Mimo prostej funkcjonalności układu przeprowadzone zostało wiele testów, w szczególności tych związanych z logiką działania programu i przystępnością obsługi niedzielnego użytkownika. Ze względu na brak innych modułów jak Arduino, projekt był testowany już bezpośrednio na gotowym projekcie. Układ był wielokrotnie poddawany różnym (często mało prawdopodobnym) scenariuszom by wykluczyć możliwość zdarzenia oraz został przez jeden dzień testowany przez osobę trzecią nie mającą nigdy kontaktu z mikroprocesorami czy IT w celu sprawdzenia czy jest przystępny w obsłudze.

b) **Napotkane problemy**
Za głównym problem uważam brak doświadczenia z konstrukcją i programowaniem układów tego typu. Jestem pewien że gdybym zabierał się za to teraz, zajęłoby mi to o wiele mniej czasu i wysiłku.

Pojawił się również problem związany z połączeniem odtwarzana dźwięku w pętli ze stałym połączeniem prze bluetooth. Wynikał on z niezgodności w użytych bibliotekach. Został on zażegnany programowo poprzez wprowadzenie odrobinę innej logiki polegającej na wyłączaniu/włączeniu sensorów gdy są potrzebne.

c) **Wyniki końcowe**
Projektowanie przedstawionego układu pozwoliło mi na poszerzenie wiedzy na temat tworzenia systemów mikroprocesorowych oraz z pewnością umożliwi mi zyskanie doświadczenia w procesie lutowania mikroelementów na płytce. Zarazem rozszerzając moją wiedzę związaną z realizacji takich układów w praktyce i skomplikowania tego procesu.

Proces tworzenia omówionego schematu będę z pewnością jeszcze długo dobrze wspominał, jednakże utwierdziło mnie to w przekonaniu że mikroprocesory są wymagającą gałęzią IT i nie czuje się na tyle kompetentny żeby wiązać z nią swoją przyszłość.

d) **Literatura**
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/
https://randomnerdtutorials.com/projects-esp32/
https://github.com/earlephilhower/ESP8266Audio
https://www.youtube.com/watch?v=RStncO3zb8g&feature=youtu.be&ab_channel=RuiSantos