Car Bluetooth Adapater - ImperiumBT

Introducere

ImperiumBT este un adaptor Bluetooth audio creat pentru casetofoanele din gama VAG (ex: **Volkswagen Gamma 5**, **Audi Concert**, **Skoda Symphony**, etc) care nu dispun de intrare auxiliara.

Proiectul are ca scop modernizarea acestor sisteme audio fara a le modifica structura, prin emularea unui CD changer prin interfata SPI a unui Arduino Uno si prin transmiterea semnalului audio de la un ESP32 cu Bluetooth integrat.

Simulare pentru prezentare

In lipsa unui casetofon real in cadrul demonstratie, se va realiza o **simulare** a comportamentului acestuia utilizand:

- un amplificator audio in clasa D, alimentat de la o sursa de 24v
- un difuzor JBL de 5inch/ 4Ω
- conexiune analog de la ESP32 catre amplificator

Voi include mai jos si o demnstratie cu functionalitatea reala a proiectului, dar prin simularea anterioara voi demonstra polivalenta adapter-ului, care poate fi folosit in mai multe contexte.

Functionalitati implementate

- **Bluetooth Audio (A2DP)** – conectare la telefon pentru redare muzica
- **Emulare CD changer** prin comunicare SPI intre Arduino si casetofon
- **Iesire audio analog** de la ESP32 la amplificator
- **Ecran OLED** care afiseaza melodia curenta, conexiunea activa, etc.
- **Butoane fizice**: Play/Pause, Next, Previous

Functionalitati suplimentare

- **Hands-Free**:

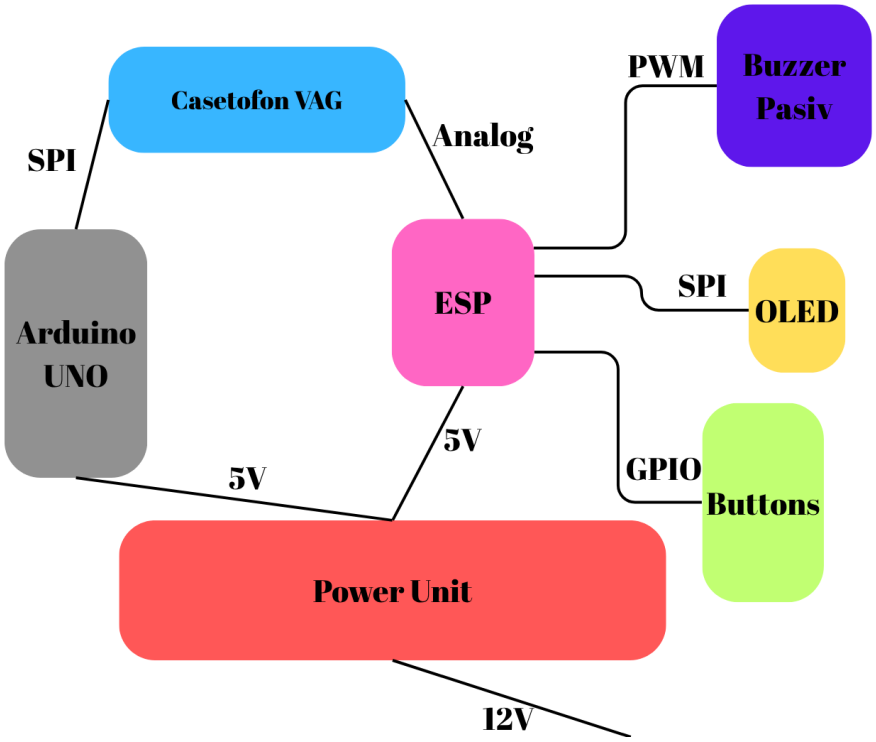
  - microfon atasat la ESP32 pentru apeluri
  - butoane pentru **preluare** / **respingere apel**
  - posibil suport HFP (viitor)
- **Extensibilitate**:
  - cititor **microSD** pentru redare locala de fisiere audio
  - conector **jack auxiliar** pentru surse audio prin cablu

Hardware Design

Lista de piese

Nr. Crt.	Denumire Componenta	Cantitate	Descriere / Specificatii	Link + DataSheet
1	ESP32 DevKit v1	1	Microcontroller cu WiFi + Bluetooth, interfete SPI/I2C/I2S	https://www.electronicmarket.ro/ro/product/esp32-wroom-32-placa-de-dezvoltare-38-pini?gad_campaignid=21513542058 https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
2	Arduino Uno R3	1	Pentru comunicarea SPI cu casetofonul (emulare CD changer)	https://ardushop.ro/ro/plci-de-dezvoltare/2282-placa-de-dezvoltare-uno-r3-compatibil-arduino-6427854027122.html?gad_campaignid=22058879462 https://ww1.microchip.com/downloads/en/devicedoc/doc8161.pdf
3	Modul microfon I2S (INMP441)	1	Microfon digital pentru transmitere voce (handsfree)	https://www.emag.ro/microfon-digital-electroweb-omnidirectional-mems-inmp441-3-f-060/pd/DL3NPBYBM/ https://www.invensense.com/wp-content/uploads/2015/02/INMP441.pdf
4	Modul cititor microSD pe SPI	1	Pentru redare muzica de pe card	
5	Modul ecran TFT 1.44” SPI	1	Afisare melodie curenta, status conexiune, etc.	https://www.optimusdigital.ro/en/lcds/2167-144-lcd-for-stc-stm32-and-arduino-boards.html?gad_campaignid=19615979487 https://cdn-shop.adafruit.com/datasheets/ST7735.pdf
6	Butoane tactile (tip pushbutton)	5	Play/Pause, Next, Prev, Accept Call, Reject Call	
7	Amplificator audio clasa D (TPA3118)	1	Suporta pana la 60W pe canal, alimentare 12–24V	https://sigmanortec.ro/modul-amplificator-audio-1-canal-60w-4-8ohm-tpa3118-8-24vdc-digital?SubmitCurrency=1&id_currency=2&gad_campaignid=22174019478 https://www.ti.com/lit/ds/symlink/tpa3118d2.pdf
8	Difuzor 5 inch, 4 Ohmi	1	Pentru testarea audio (simulare casetofon)	
9	Sursa de alimentare 24V	1	Pentru alimentarea amplificatorului si difuzorului	https://www.a2t.ro/default-category/sursa-de-alimentare-24v-10-in-comutatie-stabilizata?gad_campaignid=18574987832
10	Sursa step-down (LM2596)	1	Pentru a obtine 5V/3.3V pentru ESP32 si periferice	https://www.emag.ro/modul-coborare-tensiune-lm2596-tri252/pd/D9XL5VBBM/?ref=history-shopping_423138763_34366_1 https://www.ti.com/lit/ds/symlink/lm2596.pdf
11	Modul DAC audio CS4344	1	Pentru a genera semnal analogic	https://ardushop.ro/ro/module/729-modul-dac-audio-cs4344-cu-filtru-trece-jos-si-port-i2s-compatibil-cu-arduino-6427854009272.html https://www.cirrus.com/products/cs4344/
12	Breadboard + jumper wires	1 set	Pentru prototipare si conectare module	

Diagrama Bloc pentru casetofon



Schema electrica

Software Design

CDC Emulator

Pentru a putea trimite orice fel de semnal audio catre casetofonul auto, vom folosi un Arduino Uno, iar cu ajutorul interfetei SPI vom trimite o secvente de date catre casetofon. In lipsa acestei secvente, casetofonul in momentul in care este aleasa optiunea de redare de pe CD, va genera o eroare de tipul **NO CD CHANGER** chiar daca pe inputul audio este trimis semnal.

Folosim urmatoarele define-uri pentru a indica fiecare comanda suportata de casetofon:

```
#define CDC_PREFIX1 0x53
#define CDC_PREFIX2 0x2C
#define CDC_END_CMD 0x14
#define CDC_PLAY 0xE4
#define CDC_STOP 0x10
#define CDC_NEXT 0xF8
#define CDC_PREV 0x78
#define CDC_SEEK_FWD 0xD8
```

```
#define CDC_SEEK_RWD 0x58
#define CDC_CD1 0x0C
#define CDC_CD2 0x8C
#define CDC_CD3 0x4C
#define CDC_CD4 0xCC
#define CDC_CD5 0x2C
#define CDC_CD6 0xAC
#define CDC_SCAN 0xA0
#define CDC_SFL 0x60
#define CDC_PLAY_NORMAL 0x08
#define MODE_PLAY 0xFF
#define MODE_SHFFL 0x55
#define MODE_SCAN 0x00
uint8_t cd, tr, mode;
unsigned long prevMillis = 0;
```

Functia urmatoare **send_package()** trimite un pachet de cate 8 octeti pe SPI, cu intarziere pe fiecare transfer:

```
void send_package(uint8_t c0, uint8_t c1, uint8_t c2, uint8_t c3, uint8_t c4,
uint8_t c5, uint8_t c6, uint8_t c7) {
  uint8_t data[8] = {c0, c1, c2, c3, c4, c5, c6, c7};
  for (int i = 0; i < 8; i++) {
    SPDR = data[i];
    while (!(SPSR & (1 << SPIF)));
    delayMicroseconds(874);
  }
}
```

Initializam SPI, dupa cum urmeaza, astfel incat sa ruleze cu o viteza de transfer de 62.5kHz

```
void spi_init() {
  DDRB |= (1 << PB3) | (1 << PB5);
  DDRB &= ~(1 << PB4);
  SPCR = (1 << SPE) | (1 << MSTR) | (1 << SPR1) | (1 << SPR0);
  SPSR &= ~(1 << SPI2X);
}
```

In functia **setup()** initializam valorile initiale pentru disc, mod de redare, piesa, precum si interfata SPI. Dupa care trimitem o secventa de comenzi pentru *load* si *idle*

```
void setup() {
  cd = 1;
  tr = 1;
  mode = MODE_PLAY;
  #ifdef DEBUG
  Serial.begin(9600);
  #endif
  delay(1000);
  spi_init();
```

```
  send_package(0x74, 0xBE, 0xFE, 0xFF, 0xFF, 0xFF, 0x8F, 0x7C); // idle
  delayMicroseconds(10000);
  send_package(0x34, 0xFF, 0xFE, 0xFE, 0xFE, 0xFF, 0xFA, 0x3C); // load disc
  delayMicroseconds(100000);
  send_package(0x74, 0xBE, 0xFE, 0xFF, 0xFF, 0xFF, 0x8F, 0x7C); // idle
  delayMicroseconds(10000);
  #ifdef DEBUG
  Serial.println("Sent idle/load/idle commands");
  #endif
}
```

In continuare tot pe Arduino, in functia **loop()** vom trimite la fiecare 50ms cate un pachet care contine discul, piesa si modul de redare pentru a mentine conexiunea activa cu casetofonul auto ales:

```
void loop() {
  if ((millis() - prevMillis) > 50) {
    send_package(0x34, 0xBF ^ cd, 0xFF ^ tr, 0xFF, 0xFF, mode, 0xCF, 0x3C);
    prevMillis = millis();
```

```
    #ifdef DEBUG
    Serial.println("Sent packet");
    #endif
  }
}
```

Bluetooth Receiver

In urmatoarele bucati de cod voi descrie cum am folosit un **ESP32** pentru a deveni un receiver audio prin interfata Bluetooth a acestuia si sa redirectionez semnalul audio prin interfata I2S catre DAC extern (e.g. CS4344)

Definim in continuare pinii folositi pentru interfata I2S, butoanele de play/pause pe care le vom folosi, pinii SPI folositi pentru ecranul TFT, care va folosi biblioteca Adafruit_ST7735, precum si variabilele folosite pentru debounce:

```
#define I2S_BCLK    26
#define I2S_LRCK    25
#define I2S_DATA    22
#define I2S_MCLK    3  // sau I2S_PIN_NO_CHANGE
```

```
// === Butoane ===
#define BTN_PLAY    12
#define BTN_PAUSE   14  // redenumit din NEXT
```

```
// === Ecran ST7735 ===
#define TFT_CS     5
#define TFT_RST    4
```

```
#define TFT_DC     16
```

```
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);
BluetoothA2DPSink a2dp_sink;
```

```
String current_song = "Fara melodie";
String last_displayed_song = "";
```

```
volatile bool playPressed = false;
volatile bool pausePressed = false;
```

```
unsigned long lastDebouncePlay = 0;
unsigned long lastDebouncePause = 0;
const unsigned long debounceDelay = 150;
```

```
unsigned long lastPlayPressTime = 0;
bool waitingForSecondClick = false;
```

Functia urmatoare **metadata_callback()** va fi folosita pentru trimiterea de text pe ecranul TFT:

```
void metadata_callback(uint8_t id, const uint8_t *text) {
  if (id == 0x01) {
    current_song = String((char*)text);
  }
}
```

In functia **update_display()** vom scrie momentan numele melodiei curente:

```
void update_display() {
  if (current_song != last_displayed_song) {
    tft.fillScreen(ST77XX_BLACK);
    tft.setTextColor(ST77XX_WHITE);
    tft.setTextSize(1);
    tft.setCursor(2, 10);
    tft.println("Redare Bluetooth:");
    tft.setCursor(2, 30);
    tft.setTextWrap(true);
    tft.setTextSize(1);
    tft.println(current_song);
    tft.setCursor(2, 60);
    tft.setTextSize(1);
    tft.println("Play=D12 | Pause=D14");
    last_displayed_song = current_song;
  }
}
```

Definim urmatoarele 2 functii care vor declansa intreruperile corespunzatoare cand unul din cele 2 butoane este selectat:

```
void IRAM_ATTR isr_play() {
```

```
  if ((millis() - lastDebouncePlay) > debounceDelay) {
    playPressed = true;
    lastDebouncePlay = millis();
  }
}
```

```
void IRAM_ATTR isr_pause() {
  if ((millis() - lastDebouncePause) > debounceDelay) {
    pausePressed = true;
    lastDebouncePause = millis();
  }
}
```

In functia **setup()** initializam interfata I2S cu un sample rate de 44.1 Mhz si un MCLK de 11.2896 Mhz, precum si pinii pe care ii folosim pentr DAC-ul CS4344. Mai apoi pornim receiver-ul Bluetooth A2DP si initializam o conexiune 'ESP32_Speaker'. In final setam butoanele D12 si D14 ca intrari cu rezistenta de pull-up interna, pe care atasam 2 intreruperi.

```
void setup() {
  Serial.begin(115200);
  i2s_config_t i2s_config = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_TX),
    .sample_rate = 44100,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT,
    .channel_format = I2S_CHANNEL_FMT_RIGHT_LEFT,
    .communication_format = I2S_COMM_FORMAT_STAND_I2S,
    .intr_alloc_flags = 0,
    .dma_buf_count = 8,
    .dma_buf_len = 64,
    .use_apll = true,
    .tx_desc_auto_clear = true,
    .fixed_mclk = 11289600
  };
  i2s_pin_config_t pin_config = {
    .mck_io_num = I2S_MCLK,
    .bck_io_num = 26,
    .ws_io_num = 25,
    .data_out_num = 22,
    .data_in_num = I2S_PIN_NO_CHANGE
  };
  a2dp_sink.set_pin_config(pin_config);
  a2dp_sink.set_avrc_metadata_callback(metadata_callback);
  a2dp_sink.start("ESP32_Speaker");
  tft.initR(INITR_BLACKTAB);
  tft.setRotation(1);
  update_display();
  pinMode(BTN_PLAY, INPUT_PULLUP);
  pinMode(BTN_PAUSE, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(BTN_PLAY), isr_play, FALLING);
  attachInterrupt(digitalPinToInterrupt(BTN_PAUSE), isr_pause, FALLING);
```

```
}
```

In functia **loop()** se executa continuu si verifica daca butoanele D12 si D14 sunt apasate, precum actualizeaza si ecranul TFT:

```
void loop() {
  unsigned long now = millis();
  if (playPressed) {
    playPressed = false;
    if (waitingForSecondClick && (now - lastPlayPressTime <
doubleClickDelay)) {
      a2dp_sink.next();
      waitingForSecondClick = false;
    } else {
      lastPlayPressTime = now;
      waitingForSecondClick = true;
    }
  }
  if (waitingForSecondClick && (now - lastPlayPressTime > doubleClickDelay))
{
    a2dp_sink.play();
    waitingForSecondClick = false;
  }
  if (pausePressed) {
    pausePressed = false;
    a2dp_sink.pause();
  }
  update_display();
  delay(50);
}
```

Rezultate Obţinute

Rezultatele obtinute au fost destul de satisfacatoare, am intampinat cateva probleme pe care nu am putut sa le rezolv din cauza DAC-ului ales, deoarece acesta are nevoie de un master clock extern care sa fie sincron cu celelalte semnale de pe ele, iar ESP-ul nu este capabil de a oferi corect un astfel de semnal, generand destul de mult zgomot (cred ca un MAX98357 ar fi rezolvat problema zgomotului). Rezultatul final poate fi observat in urmatorul link:
https://drive.google.com/file/d/1erp1HS01shDhwGrpYLM5Av1lJeFgSRbM/view?usp=sharing

Bibliografie/Resurse

https://github.com/NullString1/VWCDC
https://schuett.io/2013/09/avr-raspberry-pi-vw-beta-vag-cdc-faker/

https://itohi.com/acoustics/esp32-as-bluetooth-audio/