ウェアラブル会話記録デバイス 開発仕様書 (Draft)

これまでの議論（システム構成、ハードウェア選定、開発ロードマップ）をまとめた**技術仕様書（ドラフト版）**を作成しました。このドキュメントをベースに開発を進めたり、部品購入リストとしてご活用ください。最後に、動作検証用のサンプルコードを添付しています。

1. プロジェクト概要

- **目的:** 人の会話を記録し続けるアクセサリーデバイスの開発
- **形状:** 直径5cm程度の首かけデバイス
- **主要機能:**
    - 内蔵マイクによる音声取得
    - SDカードへのデータ保存（フェーズ1）
    - 省電力駆動による長時間稼働

2. システム構成図

[Device] ウェアラブルデバイス (直径5cm)

1. **Power (USB Type-C 入力)**
    - (給電・充電) → **PMIC (充電回路 / XIAO内蔵)**
2. **PMIC**
    - (充放電) → **Battery (LiPoバッテリー 3.7V)**
3. **Battery**
    - (電力供給) → **MCU (Seeed XIAO nRF52840 Sense)**
4. **MIC (内蔵PDMマイク)**
    - (音声信号) → **MCU**
5. **MCU**
    - (SPI通信) → **SD (microSDカード)**

[UserInterface]

1. **PC / スマホ**
    - (SD読み込み) → **AudioData (音声データ .wav)**

3. ハードウェア選定

3.1 メインコントローラ (MCU)

- **製品名:** Seeed XIAO nRF52840 Sense
- **選定理由:**
    1. **Nordic nRF52840搭載:** ESP32と比較して圧倒的な低消費電力（数mA vs 数百mA）。
    2. **マイク内蔵:** PDMデジタルマイクが基板上にあり、外付け不要。
    3. **充電回路内蔵:** リチウムポリマー電池の充電管理ICを搭載済み。
    4. **サイズ:** 21mm x 17.5mm と極小。

3.2 ストレージ

- **媒体:** microSDカード
- **理由:** 開発段階でのデバッグ容易性を優先（PCで即座にデータ確認が可能）。
- **将来性:** 量産時は基板実装型のSPI Flashメモリへの置き換えを検討。

3.3 電源・バッテリー

- **バッテリー:** リチウムポリマー電池 (LiPo) 1セル (3.7V)
- **容量目安:** 300mAh ～ 500mAh
- **充電方式:** マイコンボードのUSB Type-C端子経由で充電。

4. 開発ロードマップと構成

Phase 1: プロトタイプ開発 (PoC)

ハンダ付けを行わず、機能検証を行うフェーズ。

- **構成:**
  - Seeed XIAO nRF52840 Sense
  - Seeed Studio XIAO Expansion Board (SDスロット、OLED、バッテリーコネクタ搭載)
  - LiPoバッテリー (JST-PH2.0コネクタ接続)
- **検証項目:**
  - マイクからの録音品質確認
  - SDカードへの書き込み動作
  - 実際のバッテリー稼働時間の計測

Phase 2: 小型化・実装

直径5cmの筐体に収めるための最小構成フェーズ。

- **構成:**
  - Seeed XIAO nRF52840 Sense
  - 小型microSDカードスロット (DIP化モジュール等を直接配線)
  - LiPoバッテリー (XIAO裏面のパッドに直接ハンダ付け)
  - 電源用スライドスイッチ
  - **筐体:** 3Dプリンタによる作成

5. 必要な部材リスト (BOM)

部品名	備考
Seeed XIAO nRF52840 Sense	※必ず"Sense"版を購入すること
Seeed Studio XIAO Expansion Board	プロトタイプ用拡張ボード
リチウムポリマー電池 (3.7V)	300~500mAh推奨。コネクタ極性に注意
microSDカード	32GB以下 (FAT32フォーマット)
microSDカードスロットモジュール	Phase 2用 (小型SPI接続タイプ)
スライドスイッチ	Phase 2用

6. ソフトウェア要件とサンプルコード

- **開発環境:** Arduino IDE
- **ボード定義:** Seeed nRF52 Boards
- **使用ライブラリ:**
  - `PDM` (内蔵マイク制御)
  - `SD` (SDカード制御)

動作検証用ソースコード (Arduino/C++)

このコードは、**「電源を入れると録音を開始し、WAVファイルとしてSDカードに保存し続ける」**シンプルなサンプルです。

```
/*
  XIAO nRF52840 Sense Audio Recorder (Simple WAV)
  機能: 内蔵PDMマイクから音声を拾い、SDカードにWAV形式で保存します。

  注意:
  - Expansion Boardを使用する場合、CSピンは D2 です。
  - バッファがあふれないよう、SD書き込み速度に注意が必要です。
*/

#include <PDM.h>
#include <SD.h>
#include <SPI.h>

// --- 設定 ---
#define SD_CS_PIN D2       // Expansion BoardのCSピン
#define FILE_NAME "REC001.WAV"

// 音声設定 (nRF52840 PDM)
// 16kHz, モノラル, 16bit が安定動作の基本ライン
#define SAMPLE_RATE 16000
#define CHANNELS    1

// バッファ設定
// PDMからのデータを受け取る短期バッファ
short sampleBuffer[256];
volatile int samplesRead; // 読み込んだサンプル数

File audioFile;

// WAVヘッダを書き込む関数
// ファイルサイズは後で更新するため、プレースホルダーとして書き込みます
void writeWavHeader(File &file) {
  file.print("RIFF");
  uint8_t temp[4] = {0, 0, 0, 0}; // ファイルサイズ (後で埋める)
  file.write(temp, 4);
  file.print("WAVE");
  file.print("fmt ");

  uint32_t subChunk1Size = 16; // PCM
```

```
  uint16_t audioFormat = 1;    // PCM
  uint16_t numChannels = CHANNELS;
  uint32_t sampleRate = SAMPLE_RATE;
  uint32_t byteRate = SAMPLE_RATE * CHANNELS * 2; // 16bit = 2bytes
  uint16_t blockAlign = CHANNELS * 2;
  uint16_t bitsPerSample = 16;

  file.write((uint8_t*)&subChunk1Size, 4);
  file.write((uint8_t*)&audioFormat, 2);
  file.write((uint8_t*)&numChannels, 2);
  file.write((uint8_t*)&sampleRate, 4);
  file.write((uint8_t*)&byteRate, 4);
  file.write((uint8_t*)&blockAlign, 2);
  file.write((uint8_t*)&bitsPerSample, 2);

  file.print("data");
  file.write(temp, 4); // データサイズ（後で埋める）
}

// ファイルサイズを更新する関数（録音終了時や定期保存時に使用）
void updateWavHeader(File &file) {
  unsigned long fileSize = file.size();
  unsigned long dataSize = fileSize - 44; // ヘッダ44バイトを除く

  file.seek(4);
  uint32_t riffSize = fileSize - 8;
  file.write((uint8_t*)&riffSize, 4);

  file.seek(40);
  file.write((uint8_t*)&dataSize, 4);

  // ファイルの末尾に戻る
  file.seek(fileSize);
}

// PDMデータ受信時のコールバック関数（割り込み処理）
void onPDMdata() {
  int bytesAvailable = PDM.available();
  PDM.read(sampleBuffer, bytesAvailable);
  samplesRead = bytesAvailable / 2; // 16bitなのでバイト数/2がサンプル数
}

void setup() {
  Serial.begin(115200);
  // while (!Serial); // PCレスで動かす場合はコメントアウトする

  // LED設定（録音中表示用）
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  digitalWrite(LED_RED, HIGH); // 消灯（Low Activeの場合あり）

  Serial.println("Initializing SD card...");
  if (!SD.begin(SD_CS_PIN)) {
    Serial.println("SD initialization failed!");
```

```
    // エラー時は赤点滅などで通知する処理を入れると良い
    while (1);
  }

  // 既存ファイルがあれば削除（テスト用）
  if (SD.exists(FILE_NAME)) {
    SD.remove(FILE_NAME);
  }

  audioFile = SD.open(FILE_NAME, FILE_WRITE);
  if (!audioFile) {
    Serial.println("Failed to open file for writing");
    while (1);
  }

  // WAVヘッダの書き込み
  writeWavHeader(audioFile);

  // PDMマイク初期化
  PDM.onReceive(onPDMdata);
  if (!PDM.begin(CHANNELS, SAMPLE_RATE)) {
    Serial.println("Failed to start PDM!");
    while (1);
  }

  // ゲイン設定（必要に応じて調整: 0-80, デフォルト20）
  PDM.setGain(30);

  Serial.println("Recording started...");
  digitalWrite(LED_GREEN, LOW); // 緑点灯（録音中）
}

void loop() {
  // マイクからデータがバッファに来ていればSDに書き込む
  if (samplesRead) {
    // 割り込み中にデータが変わらないよう一時的に停止するのは推奨だが、
    // ここではシンプルに書き込む

    // SDへの書き込み（バイナリデータとして）
    audioFile.write((uint8_t *)sampleBuffer, samplesRead * 2);

    samplesRead = 0; // フラグクリア

    // 定期的にSDカードへ保存を確定させる（データ損失防止）
    // 頻繁すぎると遅くなるので注意。ここでは簡易的に毎回flushしない。

    static int saveCounter = 0;
    saveCounter++;
    if (saveCounter > 100) { // ある程度溜まったらHeader更新など
        // 実際の運用ではここでflushやHeader更新を行う
        saveCounter = 0;
    }
  }
}
```

今後の実装課題（Next Step）

VAD (音声区間検出) の実装: 上記コードは無音でも録音し続けます。バッテリー節約のため、「音がした時だけSDに書く」ロジックが必要です。

リングバッファの実装: SDカードの書き込み遅延で音が途切れないよう、ダブルバッファリング等の工夫が必要になります。

WAVヘッダの適切なクローズ: 電源が急に切れるとWAVファイルが壊れるため、一定時間ごとに保存を確定する処理が必要です。