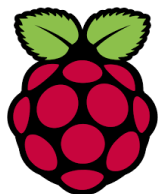


WLOラズパイ倶楽部+α

# Chirpで音声通信に挑戦



2020年1月9日  
(1/10 一部修正版)

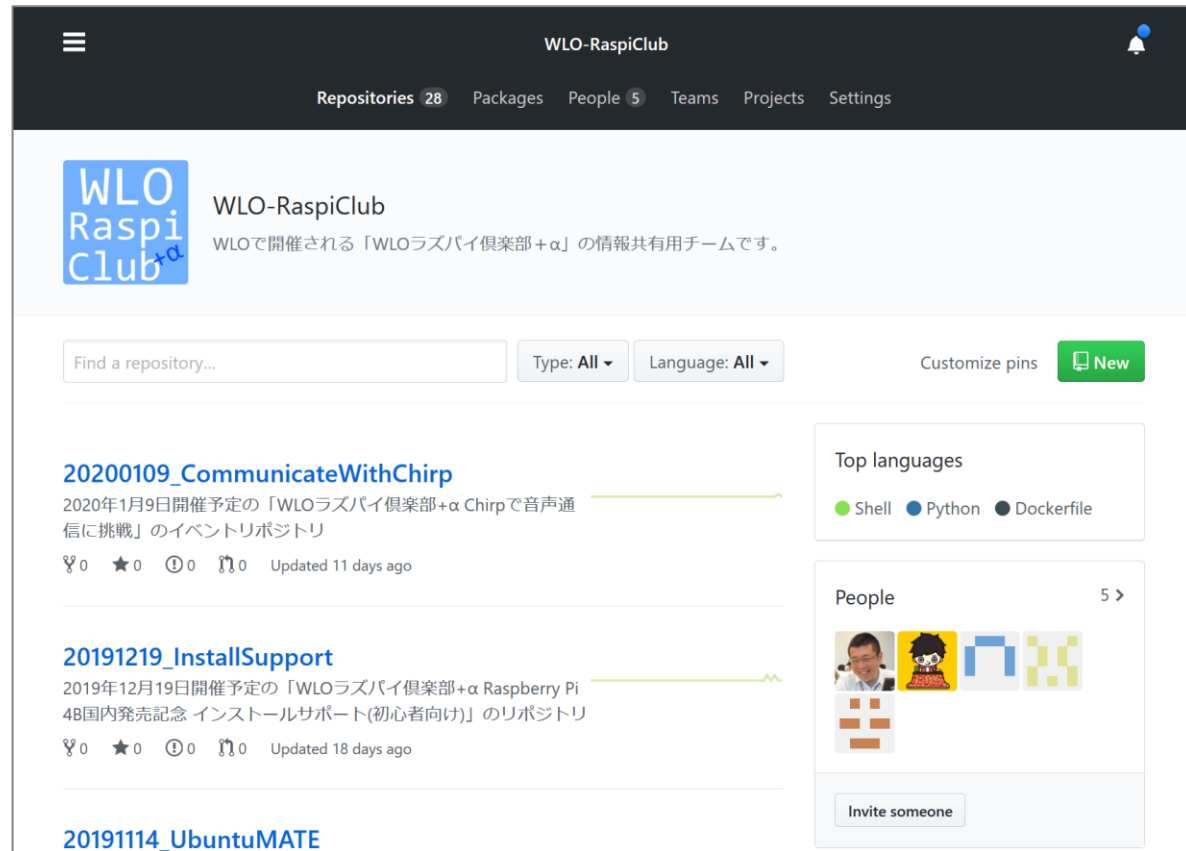
過去の

「WLOラズパイ倶楽部」の  
資料、スクリプトが  
格納されています。

今後のイベントでも  
ここに資料を格納するので  
予習・復習にご活用ください。

<https://github.com/WLO-RaspiClub/>

<https://git.io/wlopi>



The screenshot shows the GitHub profile page for WLO-RaspiClub. The header includes the repository count (28), packages, people (5), teams, projects, and settings. The main content area displays the repository name, a description, and a list of repositories. The first repository is '20200109\_CommunicateWithChirp', which is a repository for a Chirp audio message challenge event. The second repository is '20191219\_InstallSupport', which is a repository for a Raspberry Pi installation support event. The third repository is '20191114\_UbuntuMATE'. On the right side, there are sections for 'Top languages' (Shell, Python, Dockerfile) and 'People' (5 members).

WLO-RaspiClub

WLOで開催される「WLOラズパイ倶楽部+α」の情報共有チームです。

Find a repository... Type: All Language: All Customize pins New

**20200109\_CommunicateWithChirp**  
2020年1月9日開催予定の「WLOラズパイ倶楽部+α Chirpで音声通信に挑戦」のイベントリポジトリ  
👤 0 ★ 0 ⓘ 0 📄 0 Updated 11 days ago

**20191219\_InstallSupport**  
2019年12月19日開催予定の「WLOラズパイ倶楽部+α Raspberry Pi 4B国内発売記念 インストールサポート(初心者向け)」のリポジトリ  
👤 0 ★ 0 ⓘ 0 📄 0 Updated 18 days ago

**20191114\_UbuntuMATE**

Top languages  
● Shell ● Python ● Dockerfile

People 5 >  
[Profile pictures of team members]  
Invite someone

<http://git.io/wlopi>



## ・ Raspberry Pi 4Bな年末年始

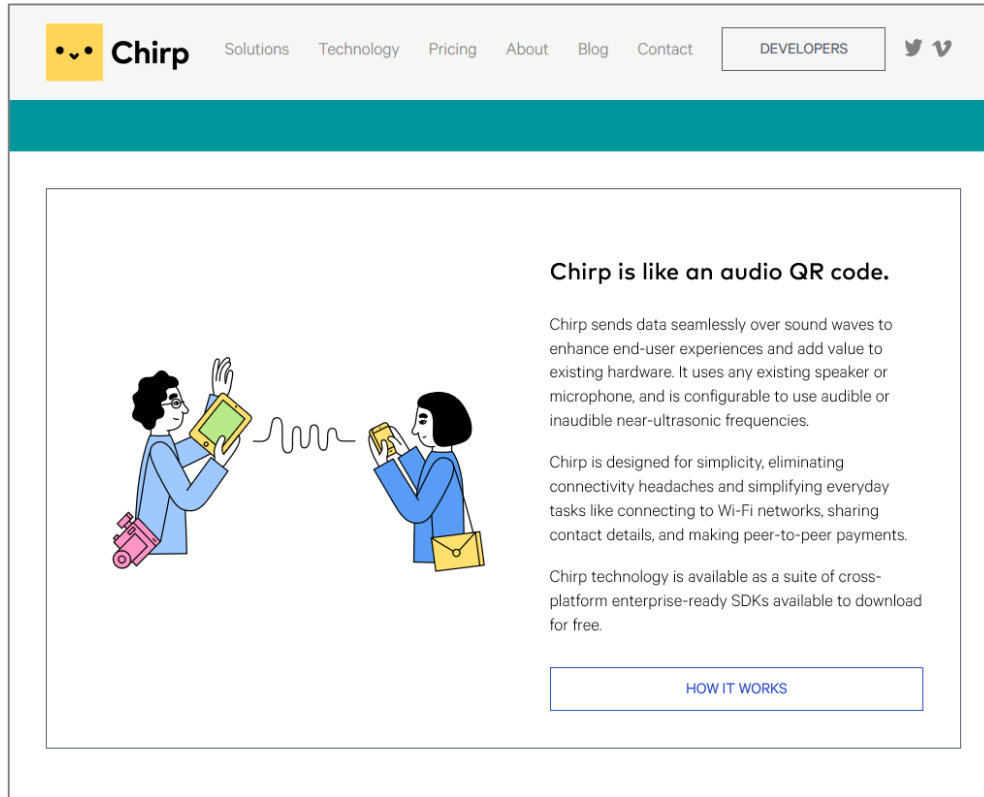
過去商品の国内発売時と比較して、  
Raspberry Pi 4Bの国内初期流動が多く入手がしやすい。

- ・ 英国発売時から時間がたって製造がこなれていた？  
（技適版は新生産されたので在庫量には影響しないはず...）
- ・ 周辺機器（ケース、ファン、電源等）も最初から充実  
（こちらは先行国版と共通なので発売遅延がメリット）  
（熱対策の情報等も先行して
- ・ 3B+の在庫が12月上旬までなかった  
（12月中旬以降、徐々に流通が戻りつつあります）



- ・ 音声通信ライブラリ Chirp を理解する
- ・ Raspberry Piと周辺機器で音声通信を体験する
- ・ 自分のプロジェクトに取り入れる方法を学ぶ

## ■ 音声通信ライブラリ Chirp



<https://chirp.io/>

### 特徴：

2009年にロンドン大学で設立  
音のQRコード

近接通信に特化

速度は16～80bps

最大90Bytes (条件あり)

4種類のプロトコル(不可聴域版有)  
でPC/スマホ/組み込みに対応

Swift/ObjectiveC/Kotlin/Java  
/.NET/JavaScript/WebAssembly  
/Python/WebAPIのSDKが利用可能

放送・電話などでも送受信可能  
(メディア帯域によって  
プロトコル制限あり)

教育、非営利、個人利用  
月1万MAUまでの営利利用は無料

<http://git.io/wlopi>

## ■ Chirpのユースケース

### 近接通信：

デバイス間、デバイス～サービスの省リソースな通信

### プロビジョニング：

デバイスの設定（WiFi設定などの初期設定、ネット非接続機器の設定）

### 認証通信：

双方向通信（マイク：スピーカー）による認証認可

## ■ Chirpのメリット

強固な誤り訂正、1対n通信、多彩な対応プラットフォーム、OpenなSDKの提供

## ■ Chirpが向いてない分野

リアルタイム通信、高速通信、n対n通信

## ■ Chirpのプロトコルとペイロード、認証

プロトコル：4種類（有償契約でいくつか追加される）

standard: 可聴域、ステレオ、サンプリング周波数44.1kHz～、最大32bytes

ultrasonic: 不可聴域、最大8bytes

16kHz: 組み込み用にサンプリング周波数を16kHzに、**最大90bytes**

16kHz-mono: さらに軽量化にするために16kHzモノラルに、最大32bytes

SDKによって利用可能なプロトコルの制限がある。

（Arduino/WebAssemblyは16khz-monoのみ、Javascriptは送信のみ、など）

ペイロード：バイト配列/文字列/HEXバイナリ/数字

認証：アカウント登録を行い、発行されたkey/secret/configuration tokenを

SDKに設定することが必要

（WebAPIではkey/secretでBASIC認証する）

## ■スマートフォン ストアアプリ版 Chirp Messenger

Android版  
※iOS版は  
公開停止

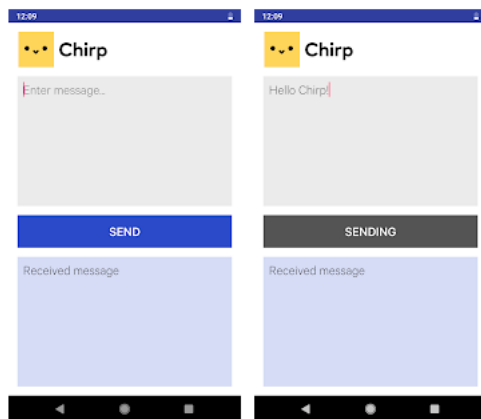


Chirp Messenger

Asio Ltd 通信

3+

📌 ほしいもののリストに追加



## ■Webアプリ版 Chirp Messenger

<https://messenger.chirp.io/>



iOSでも  
Androidでも  
利用可能  
※Raspbianの  
Chromiumでは  
マイク設定がないと  
動作しない

### Chirp Messenger

Enter your message below.

SEND

Made by [Chirp](#).  
Want to create your own web apps using data-over-sound?  
Start building at the [Chirp developer hub](#).

<http://git.io/wlopi>



## ■ Raspberry Piで使うためには

- 1 : オーディオインターフェイスの設定  
マイク・スピーカー、マイク入力ゲイン
- 2 : Chromiumの chirp messengerを試す
- 3 : Chirp.io に登録してクレデンシャル(APP\_KEY/APP\_SECRET/APP\_CONFIG)を取得
- 4 : コマンドライン版の導入  
→ターミナルで使える、音声ファイルの生成ができる
- 5 : Python / C / Java のSDKをダウンロード  
→自分のプロジェクトで使える

- 音声出力：標準でオーディオジャック or HDMI、オプションでUSB audio

※Raspberry Pi Zeroにはオーディオジャックはない

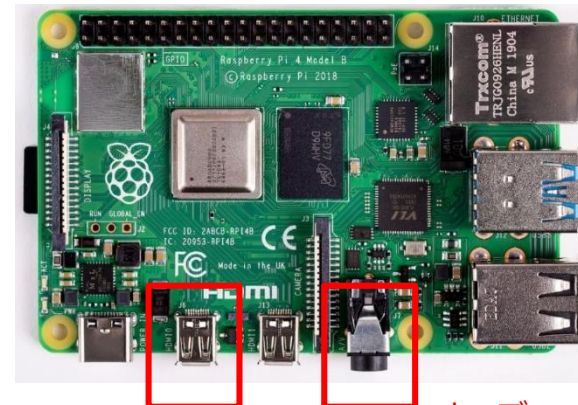
デフォルトではオーディオジャック/HDMIが自動切換

※HDMIディスプレイに音声出力ある場合はHDMI優先

`sudo raspi-config`

で強制的に出力先を指定できる (P14)

USB audioを使う場合は、P15の設定を実施



HDMI

オーディオ  
ジャック

- 音声入力：標準では手段なし、オプションでUSB audio/Bluetooth

出力先はALSADEV 環境変数で設定する (P11)

※注意：USB audio インターフェイスを複数つかうと設定が複雑になるので、  
音声入力だけをUSB audioにするか、入出力を持つUSB audioがおすすめ

## ■ USBマイク or オーディオインターフェースを接続する

- lsusb コマンドで認識を確認

```
pi@raspberrypi:~ $ lsusb
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 04f2:0918 Chicony Electronics Co., Ltd
Bus 001 Device 003: ID 8086:0808 Intel Corp.
Bus 001 Device 005: ID 0d8c:013c C-Media Electronics, Inc. CM108 Audio Controller
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

USB  
マイク

複数のUSB audioを  
取り付けていて  
どれが特定のデバイス  
か不明な場合は  
デバイスを抜き差し  
して確認

USBaudioコンバータ

- arecord -l で対象の音声入力デバイス認識を確認

```
pi@raspberrypi:~ $ arecord -l
**** ハードウェアデバイス CAPTURE のリスト ****
カード 1: Device [USB PnP Sound Device], デバイス 0: USB Audio [USB Audio]
サブデバイス: 1/1
サブデバイス #0: subdevice #0
カード 2: Device_1 [USB PnP Sound Device], デバイス 0: USB Audio [USB Audio]
サブデバイス: 1/1
サブデバイス #0: subdevice #0
```

複数のUSB audioを  
取り付けてる場合  
カード番号は  
lsusbのBus:Deviceの  
若い順に認識される

カード番号（上記だと2）とデバイス番号（上記だと0）を確認しておく

※複数の入力デバイスを持つUSBインターフェースの場合は、 とりあえず一番若い番号を試す

- ALSADEV 環境変数を設定する

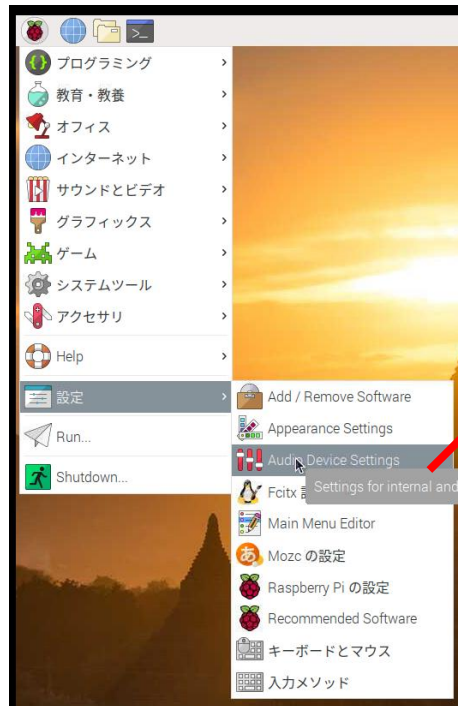
カード番号2 デバイス番号0 の場合、 `export ALSADEV="plughw:2,0"`

```
pi@raspberrypi:~ $ arecord -l
**** ハードウェアデバイス CAPTURE のリスト ****
カード 1: Device [USB PnP Sound Device], デバイス 0: USB Audio [USB Audio]
サブデバイス: 1/1
サブデバイス #0: subdevice #0
pi@raspberrypi:~ $ export ALSADEV="plughw:1,0"
```

- ALSADEV 環境変数はログイン毎、仮想端末(LXTerminal)毎に設定が必要  
自動で設定するには、.bashrcに記述する

## ■ USBマイク or オーディオインターフェースを接続する(承前)

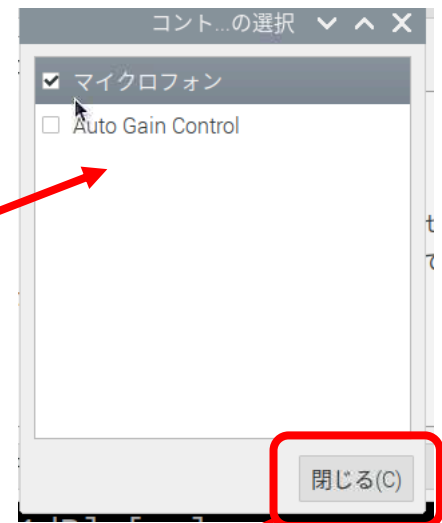
### ・ 録音ゲインの設定 (GUIでの設定)



### サウンドカードの切換え(USB PnP Sound Device)



### マイクロフォンをチェック

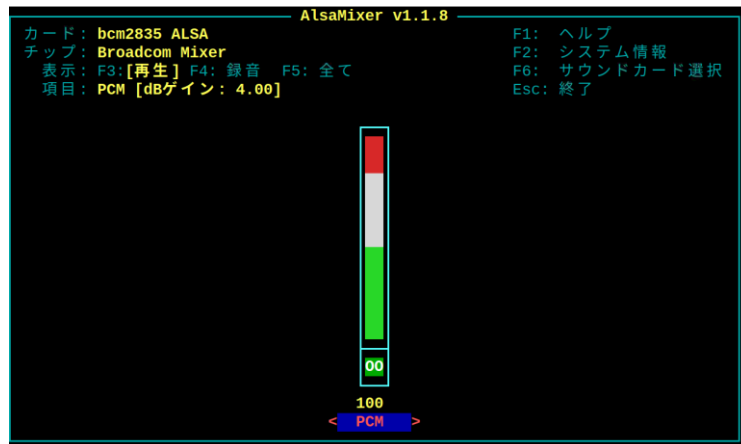


ボリュームを最大にして  
OKで閉じる

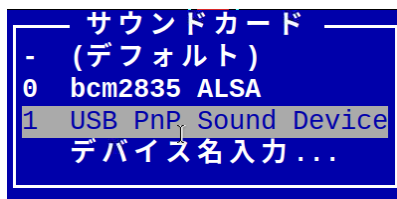


## ■ USBマイク or オーディオインターフェースを接続する(承前)

- ・ 録音ボリュームの設定 (CLIの設定: alsamixer コマンド)

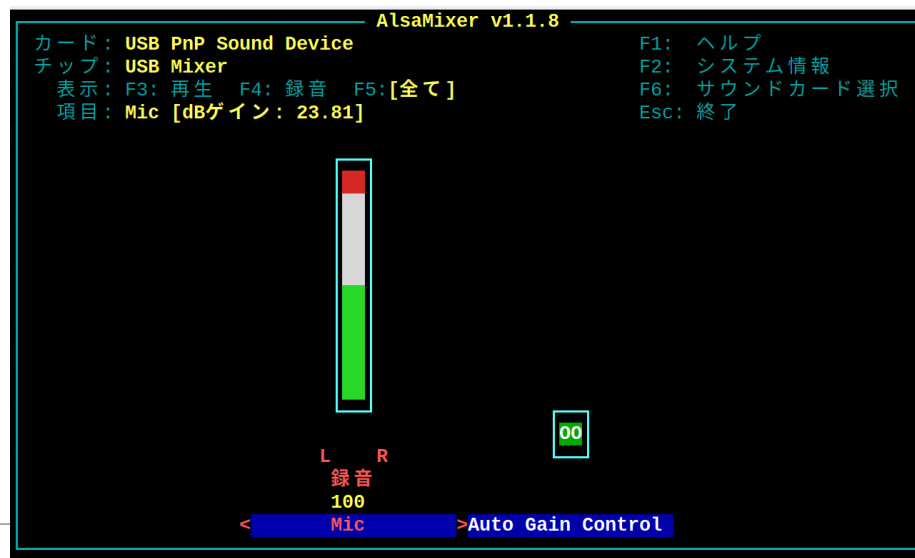


- ・ サウンドカード選択(F6)  
→USB PnP Sound Deviceを選択(Enter)



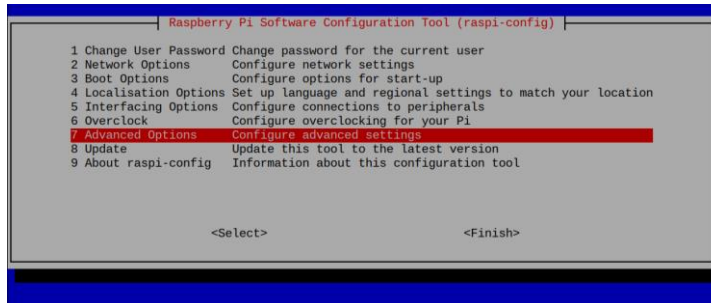
- ・ 録音デバイス選択(F4)  
→カーソルキーで100%にする

- ・ ESCで終了

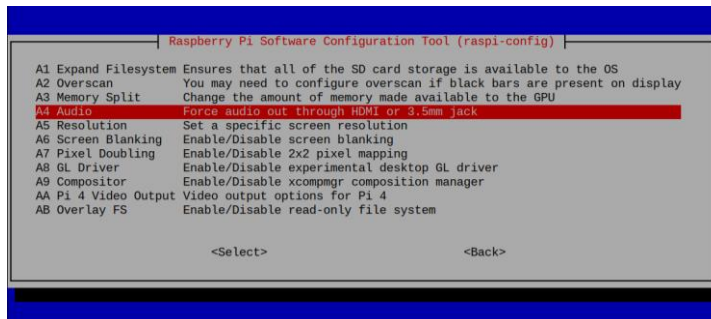


## ■内蔵オーディオインターフェースの設定

- ・ raspi-config で設定する コンソールから「sudo raspi-config」



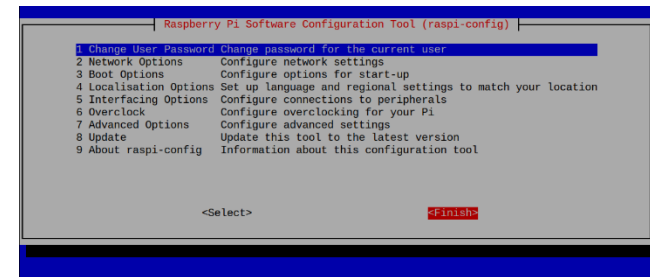
カーソルキーで  
7 Advanced Options  
を選択してEnter キー



カーソルキーで  
A4 Audio  
を選択してEnter キー



カーソルキーで  
HDMIか3.5mmを  
選択して  
TABで了解に移動して  
Enter キー



TAB 2回でFinishに移動して  
Enter で終了

## ■ USBオーディオインターフェースを接続する

- lsusb コマンドで認識を確認

```
pi@raspberrypi:~ $ lsusb
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 04f2:0918 Chicony Electronics Co., Ltd
Bus 001 Device 003: ID 8086:0808 Intel Corp.
Bus 001 Device 005: ID 0d8c:013c C-Media Electronics, Inc. CM108 Audio Controller
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

不明な場合は  
デバイスを抜き差し  
して確認

- aplay -l で対象の音声入力デバイス認識を確認

```
pi@raspberrypi:~ $ aplay -l
**** ハードウェアデバイス PLAYBACK のリスト ****
カード 0: ALSA [bcm2835 ALSA], デバイス 0: bcm2835 ALSA [bcm2835 ALSA]
  サブデバイス: 7/7
    サブデバイス #0: subdevice #0
    サブデバイス #1: subdevice #1
    サブデバイス #2: subdevice #2
    サブデバイス #3: subdevice #3
    サブデバイス #4: subdevice #4
    サブデバイス #5: subdevice #5
    サブデバイス #6: subdevice #6
カード 0: ALSA [bcm2835 ALSA], デバイス 1: bcm2835 IEC958/HDMI [bcm2835 IEC958/HDMI]
  サブデバイス: 1/1
    サブデバイス #0: subdevice #0
カード 0: ALSA [bcm2835 ALSA], デバイス 2: bcm2835 IEC958/HDMI1 [bcm2835 IEC958/HDMI1]
  サブデバイス: 1/1
    サブデバイス #0: subdevice #0
カード 2: Device_1 [USB PnP Sound Device], デバイス 0: USB Audio [USB Audio]
  サブデバイス: 1/1
    サブデバイス #0: subdevice #0
```

bcm2835  
が標準インターフェイス

USB Audio  
はここ

カード番号（上記だと2）とデバイス番号（上記だと0）を確認しておく  
※複数の入力デバイスを持つUSBインターフェイスの場合は、  
とりあえず一番若い番号を試す



## ■ USBオーディオインターフェースを接続する（承前）

- ALSADEV 環境変数を設定する

カード番号1デバイス番号0 の場合、 `export ALSADEV="plughw:1,0"`

```
pi@raspberrypi:~ $ arecord -l
**** ハードウェアデバイス CAPTURE のリスト ****
カード 1: Device [USB PnP Sound Device], デバイス 0: USB Audio [USB Audio]
  サブデバイス: 1/1
  サブデバイス #0: subdevice #0
pi@raspberrypi:~ $ export ALSADEV="plughw:1,0"
```

- ALSADEV 環境変数はログイン毎、仮想端末(LXTerminal)毎に設定が必要  
自動で設定するには、.bashrcに記述する



## ■ブラウザのオーディオ設定

- Chromium標準設定では、カード番号0のオーディオインターフェイスを使う  
→通常、カード番号0にはオーディオ入力がないため、WebAudioが起動しない

## ■ コマンドラインから、オーディオデバイスを指定してChromiumを起動する

引数：

入力デバイスの指定: `--alsa-input-device='plughw:2,0'`

出力デバイスの指定: `--alsa-output-device='plughw:1,0'`

(P11/P16参照)

起動サンプル：

```
$ chromium-browser --alsa-input-device='plughw:2,0' --alsa-output-device='plughw:1,0'
```

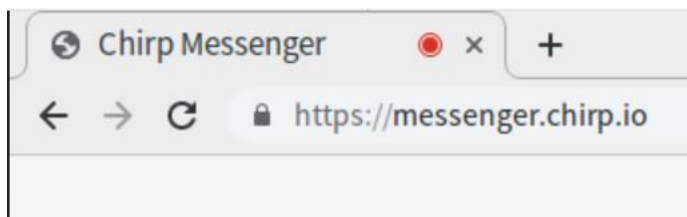
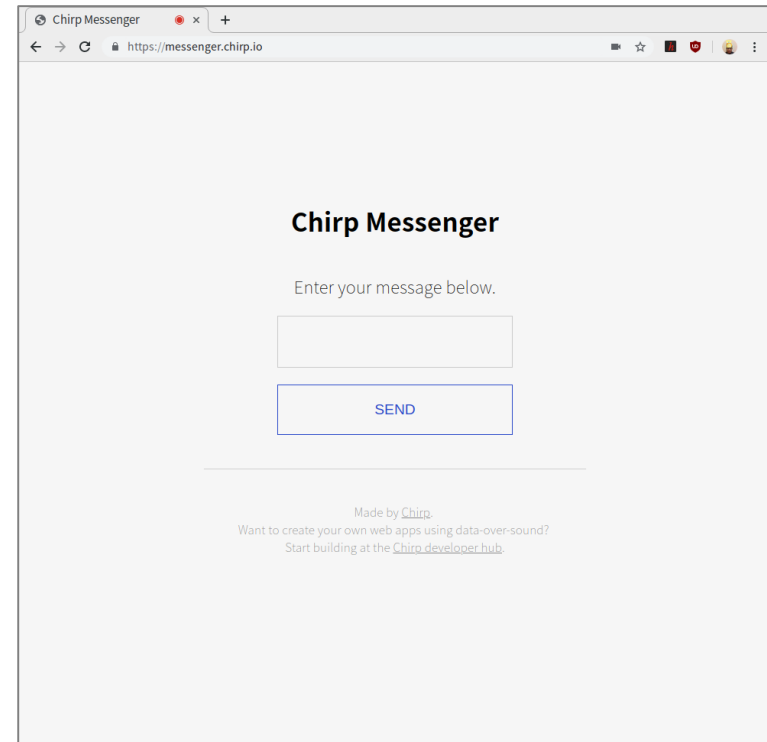
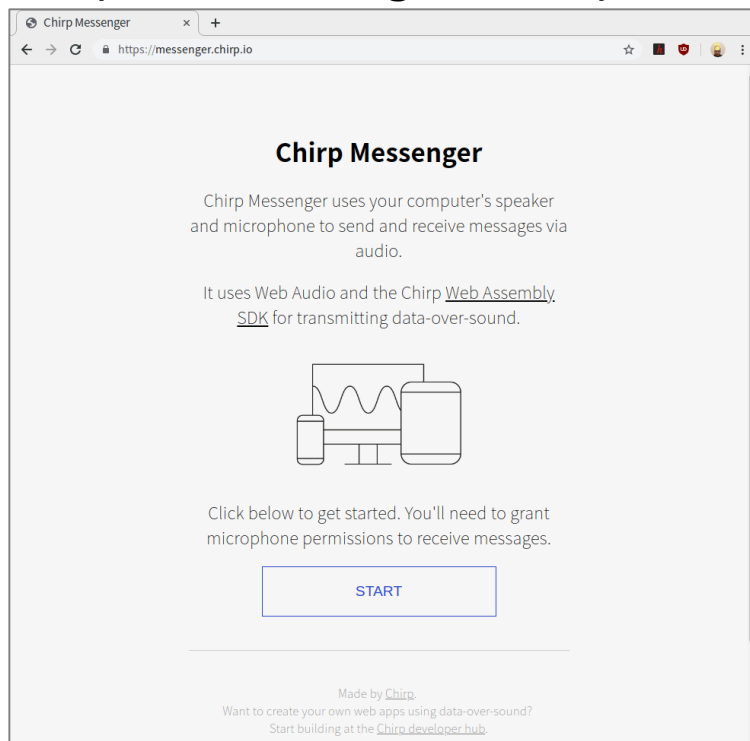
※Chromiumが起動しない、起動してもマイク使えない場合は、  
起動ターミナルに表示されるエラーメッセージを参照

## 2. ブラウザのchirp messengerで試す

18

- 開いたブラウザでchirp-messengerサイトにアクセス

<https://messenger.chirp.io/>



マイク入力の設定できていたら  
タブに赤丸が表示されます。

<http://git.io/wlopi>

## ■ アカウント登録

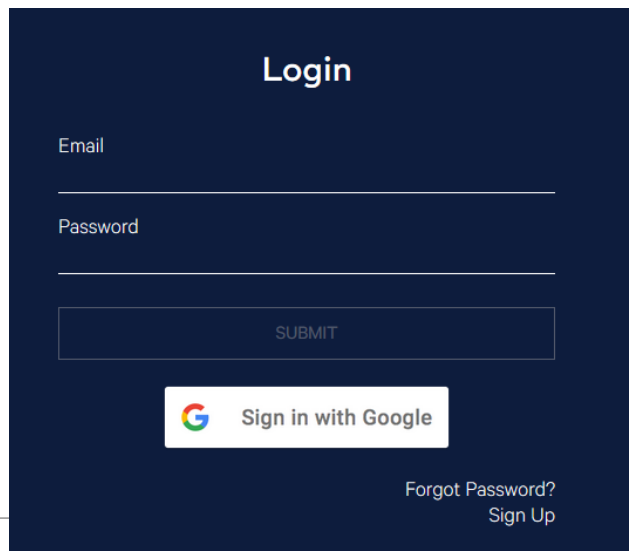
- Chirp-SDKを使うためには、無料利用でもアカウント作成を行い、クレデンシャルの発行入手が必要です。

<https://developers.chirp.io/sign-up>

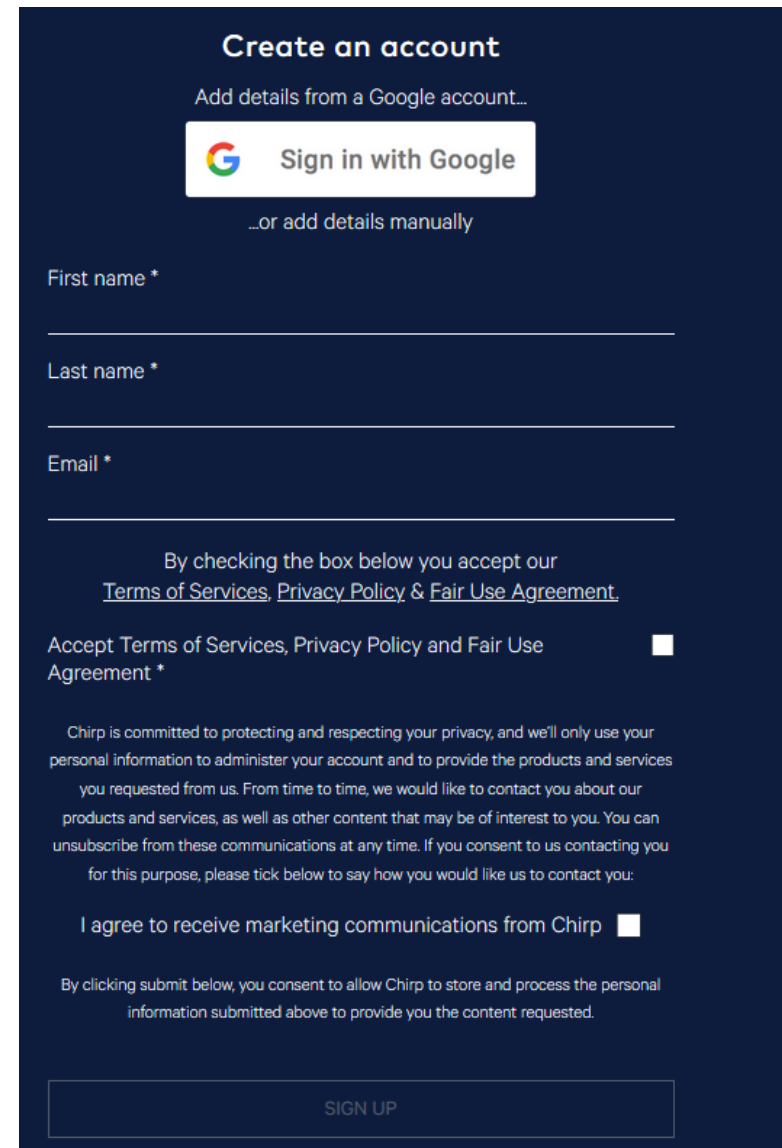
- Google Signinでの登録が簡単

## ■ アカウントができたならログイン

<https://developers.chirp.io/login>



The login form is titled "Login" and is set against a dark blue background. It features two input fields: "Email" and "Password". Below these fields is a "SUBMIT" button. At the bottom of the form, there is a "Sign in with Google" button and a link for "Forgot Password? Sign Up".



The "Create an account" form is also on a dark blue background. It starts with the heading "Create an account" and a subtext "Add details from a Google account...". There is a "Sign in with Google" button, followed by a link "...or add details manually". The form includes input fields for "First name \*", "Last name \*", and "Email \*". Below these is a section for terms and conditions, starting with "By checking the box below you accept our" followed by links to "Terms of Services", "Privacy Policy", and "Fair Use Agreement". There is a checkbox for "Accept Terms of Services, Privacy Policy and Fair Use Agreement \*". A paragraph of text explains Chirp's privacy policy. Another checkbox is for "I agree to receive marketing communications from Chirp". A final paragraph states that by clicking submit, the user consents to allow Chirp to store and process personal information. At the bottom is a "SIGN UP" button.

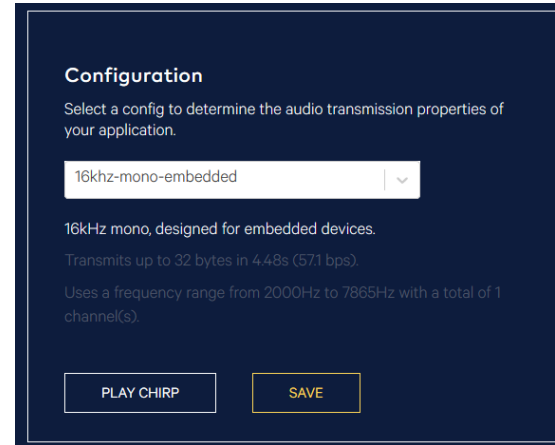
# 3. クレデンシャルの導入

20

## ■ クレデンシャル取得

- ・ クレデンシャルの発行  
Configurationで  
16kHz-mono-embeddedを選択  
→SAVEをクリック

<https://developers.chirp.io/applications>



**Configuration**

Select a config to determine the audio transmission properties of your application.

16khz-mono-embedded

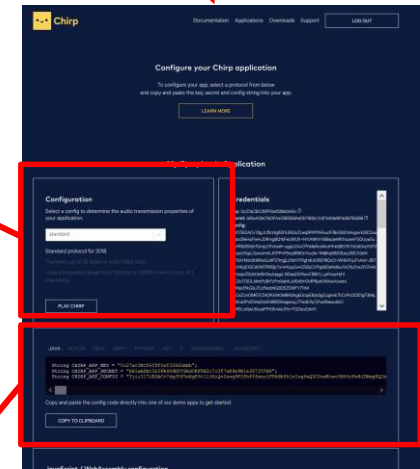
16kHz mono, designed for embedded devices.

Transmits up to 32 bytes in 4.48s (571 bps).

Uses a frequency range from 2000Hz to 7865Hz with a total of 1 channel(s).

PLAY CHIRP SAVE

全景



**Configuration**

Select a config to determine the audio transmission properties of your application.

16khz-mono-embedded

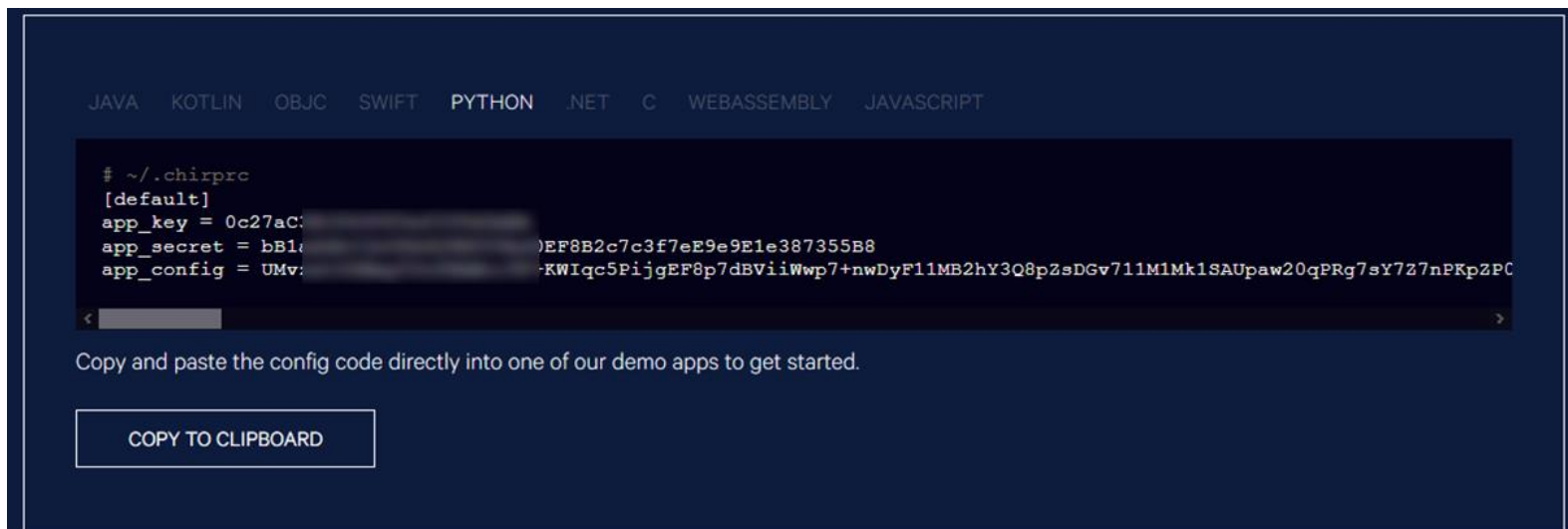
16kHz mono, designed for embedded devices.

Transmits up to 32 bytes in 4.48s (571 bps).

Uses a frequency range from 2000Hz to 7865Hz with a total of 1 channel(s).

PLAY CHIRP SAVE

言語タブからPYTHONを  
選んでCOPY TO CLIPBOARD  
をクリック



JAVA KOTLIN OBJC SWIFT PYTHON .NET C WEBASSEMBLY JAVASCRIPT

```
# ~/.chirp
[default]
app_key = 0c27ac...EF8B2c7c3f7eE9e9E1e387355B8
app_secret = b1...KWiqc5PiJgEF8p7dBViiWwp7+nwDyF11MB2hY3Q8pZsDGv711Mk1SAUpaw20qPRg7sY7Z7nPKpZPC
app_config = UMv...
```

Copy and paste the config code directly into one of our demo apps to get started.

COPY TO CLIPBOARD

<http://git.io/wlopi>

WLO  
Raspi  
Club

## ■ クレデンシャルを保存

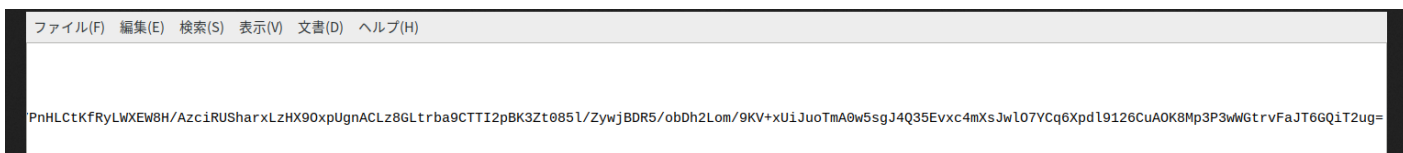
/home/pi/.chirprc にクレデンシャルを保存する

※使い慣れたエディタがある方はそちらで。以下はRaspbian PIXELのText Editorの方法

- Text Editorを開く  
(メニュー→アクセサリ→Text Editor)



- クレデンシャルをペースト  
(編集→貼り付け)

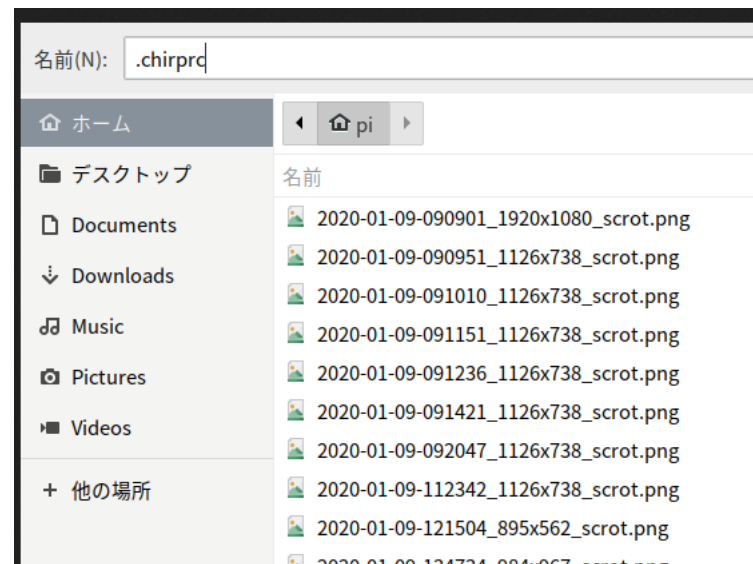


## ■ クレデンシャルを保存（承前）

- ・ 名前をつけて保存  
（ファイル→名前をつけて保存）

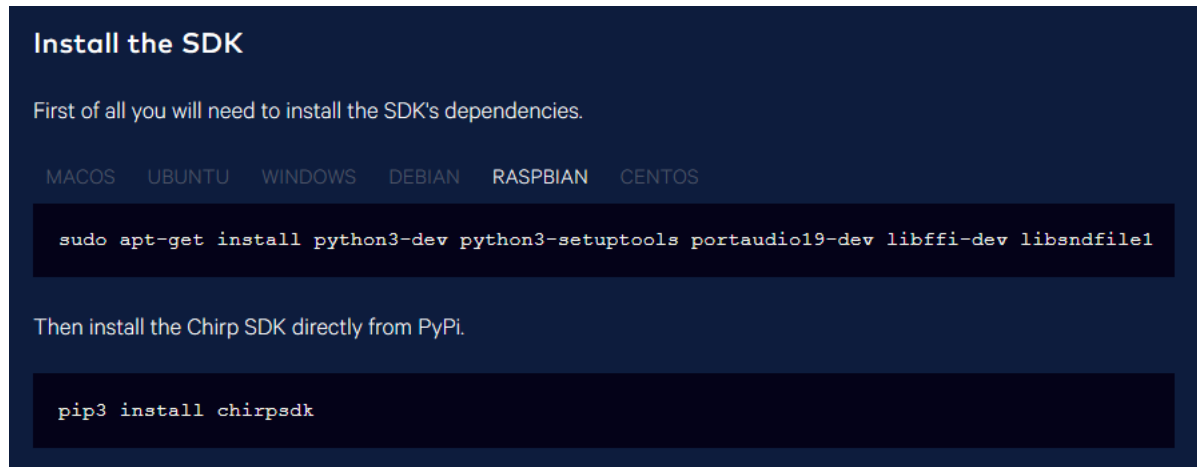


- ・ ファイル名指定  
（.chirprc）



## ■ コマンドラインツールを導入（python3で書かれています）

<https://developers.chirp.io/docs/getting-started/python>



### ・ pip3で導入する

```
$ sudo apt install python3-dev python3-setuptools portaudio19-dev libffi-dev libsndfile1
$ sudo pip3 install chirpsdk
```

※Raspbian Buster 2019-09-26

であれば、先頭のapt installはなくてもそのまま導入できるはず

### ■ コマンドラインツールの種類

chirp-send      オーディオデバイスから送信  
chirp-recieve   オーディオデバイスから受信  
chirp-audio-write WAVファイルを生成  
chirp-audio-read   WAVファイルからデータ生成

### ■ Chirpを受信してみる

\$ chirp-receive -u

```
pi@raspberrypi:~ $ chirp-receive -u -i 3
< 0 bcm2835 ALSA: IEC958/HDMI (hw:0,1), ALSA (0 in, 8 out)
  1 bcm2835 ALSA: IEC958/HDMI1 (hw:0,2), ALSA (0 in, 8 out)
> 2 USB PnP Sound Device: Audio (hw:1,0), ALSA (1 in, 2 out)
  3 USB PnP Sound Device: Audio (hw:2,0), ALSA (1 in, 0 out)
  4 dmix, ALSA (0 in, 2 out)
Chirp SDK 3.5.0 [3.3.1 1403] initialised.
Protocol: 16khz-mono-embedded [v1]
Receiving data [ch0]
Received: テスト [ch0]
```

認識している  
オーディオインターフェイス  
がリスト表示されている

\$ chirp-receive -u -i デバイス番号  
で、指定したデバイスから読み込むことができる



## ■ Chirpを送信してみる

\$ chirp-send -u 文字列

```
pi@raspberrypi:~ $ chirp-send -u 今日は  
< 0 bcm2835 ALSA: IEC958/HDMI (hw:0,1), ALSA (0 in, 8 out)  
  1 bcm2835 ALSA: IEC958/HDMI1 (hw:0,2), ALSA (0 in, 8 out)  
> 2 USB PnP Sound Device: Audio (hw:1,0), ALSA (1 in, 2 out)  
  3 USB PnP Sound Device: Audio (hw:2,0), ALSA (1 in, 0 out)  
  4 dmix, ALSA (0 in, 2 out)  
Chirp SDK 3.5.0 [3.3.1 1403] initialised.  
Protocol: 16khz-mono-embedded [v1]
```

認識している  
オーディオインターフェイス  
がリスト表示されている

\$ chirp-send -u -i デバイス番号 文字列  
で、指定したデバイスから送信できる

## ■ Chirpで音声ファイルを生成してみる

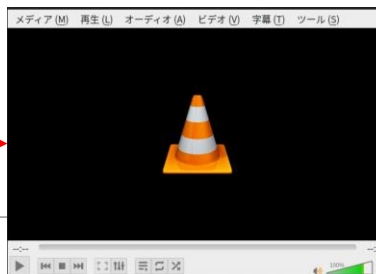
\$ chirp-audio-write -u 文字列 ファイル名

```
pi@raspberrypi:~ $ chirp-audio-write -u テスト 001.wav  
Chirp SDK 3.5.0 [3.3.1 1403] initialised.  
Protocol: 16khz-mono-embedded [v1]  
Wrote audio to output: 001.wav
```

生成した音声ファイルは  
音楽プレイヤーで  
再生することで  
Chirpを送信できる  
(P28参照)



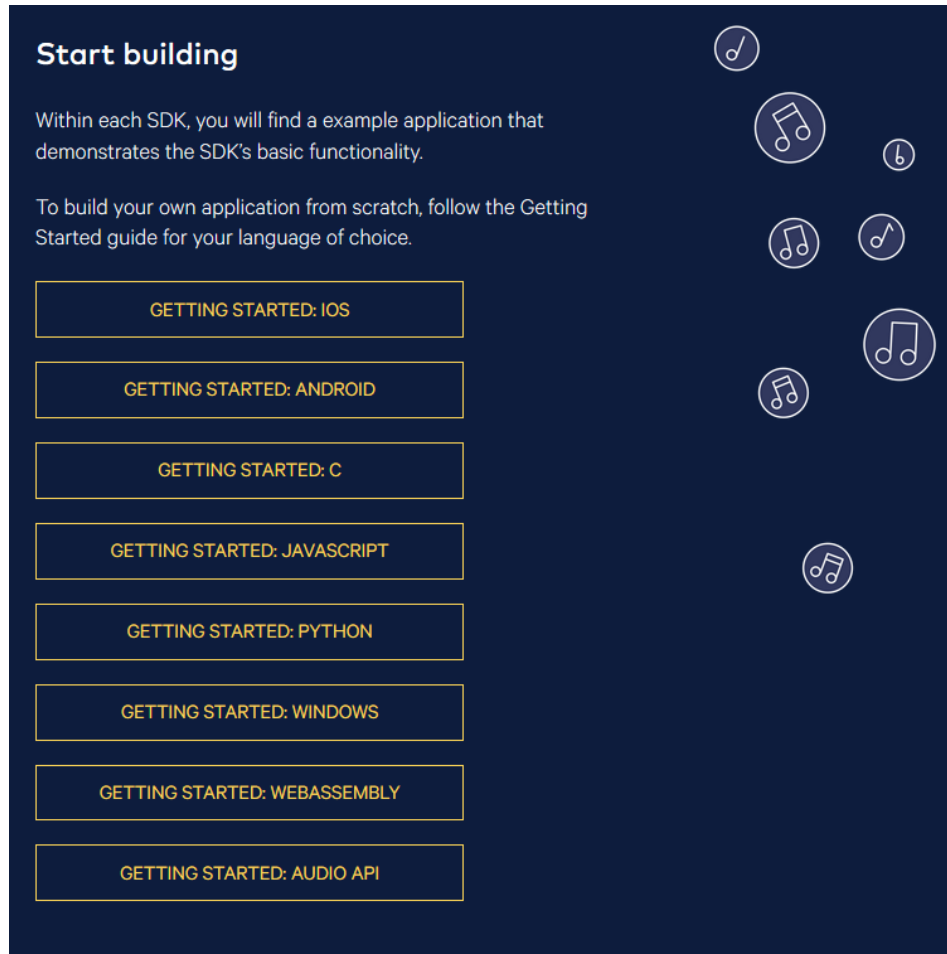
ダブルクリック  
→VLC Playerで再生



<http://git.io/wlopi>

## ■ ChirpSDKの情報

<https://developers.chirp.io/docs>



## ■ ESP32 / M5Stack / M5StickC など、Arduinoで使う

<https://developers.chirp.io/docs/getting-started/arduino>

GETTING STARTED

### Arduino

The Chirp Arduino SDK brings the capabilities of data-over-sound to the Arduino maker community.

Most boards will require an external I2S microphone and/or I2S amplifier and speaker attached. However, the new [Nano 33 BLE Sense](#) board comes with an on board microphone. You can read tutorials on how to interface with these components at the [Arduino Project Hub](#) and the [Chirp blog](#).

To see the complete set of supported platforms, head over to the [supported platforms](#) page.

#### ⚠ 16khz-mono-embedded only

The Arduino SDK is designed to work with the `16khz-mono-embedded` protocol, so you will need to ensure you select this configuration, and also use this with other SDKs for interoperability.

To configure the SDK you will need to copy your `app_key`, `app_secret`, and `app_config` for the `16khz-mono-embedded` protocol from the [applications](#) page.

#### APPLICATIONS

#### 📘 Example projects

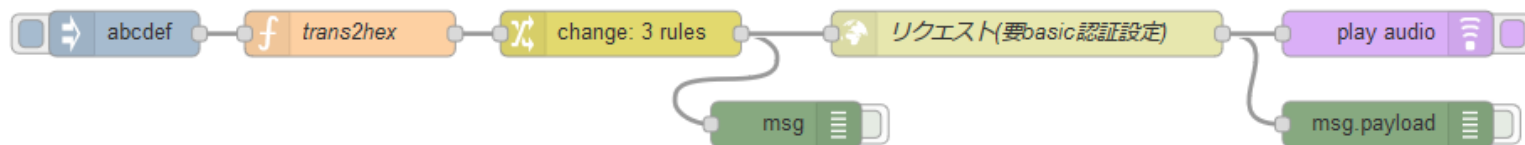
Once you have installed the SDK, you can access the example programs for each board from the menu

```
File > Examples > ChirpSDK > Example
```

<http://git.io/wlopi>

## ■ WebAPIで使う

Chirp社が提供しているWebAPIを使って、データをChirpに変換できる



[https://github.com/WLO-RaspiClub/20200109\\_CommunicateWithChirp/blob/master/NodeRED\\_chirpWebAPI.json](https://github.com/WLO-RaspiClub/20200109_CommunicateWithChirp/blob/master/NodeRED_chirpWebAPI.json)

にNode-REDのサンプルフローを置いています。

(http-request ノードにBASIC認証設定が必要です。

userにはCHIRP\_APP\_KEY、passwordにはCHIRP\_APP\_SECRETを設定します。)

## ■ chirp-audio-writeで生成したwavファイルは音楽再生ソフトで再生して送信可能

iTunes / iPodなどのmp3プレイヤーで送信できる

Google Play Music等に置いておいてスマートスピーカーから送信できる