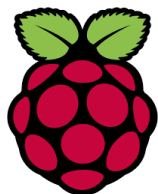


WLOラズパイ倶楽部

Sonic Piで音をプログラミング



2018年4月23日

- Sonic Piとは
- Raspberry PiにSonic Piをインストールしてつかえるようにする
 - インストール
 - 音声出力のセットアップ
- Sonic Piで音をならしてみる
 - Sonic Piの画面構成
 - 最初のスクリプト
- いろんな音をならしてみる
 - 単音
 - 和音
 - シンセサイザー
- 音をプログラムしてみる
 - ファイルを読み込んで音を鳴らす
 - 動的にならしてみる
 - 外部連携

・ケンブリッジ大学で開発された音を扱うソフトウェア

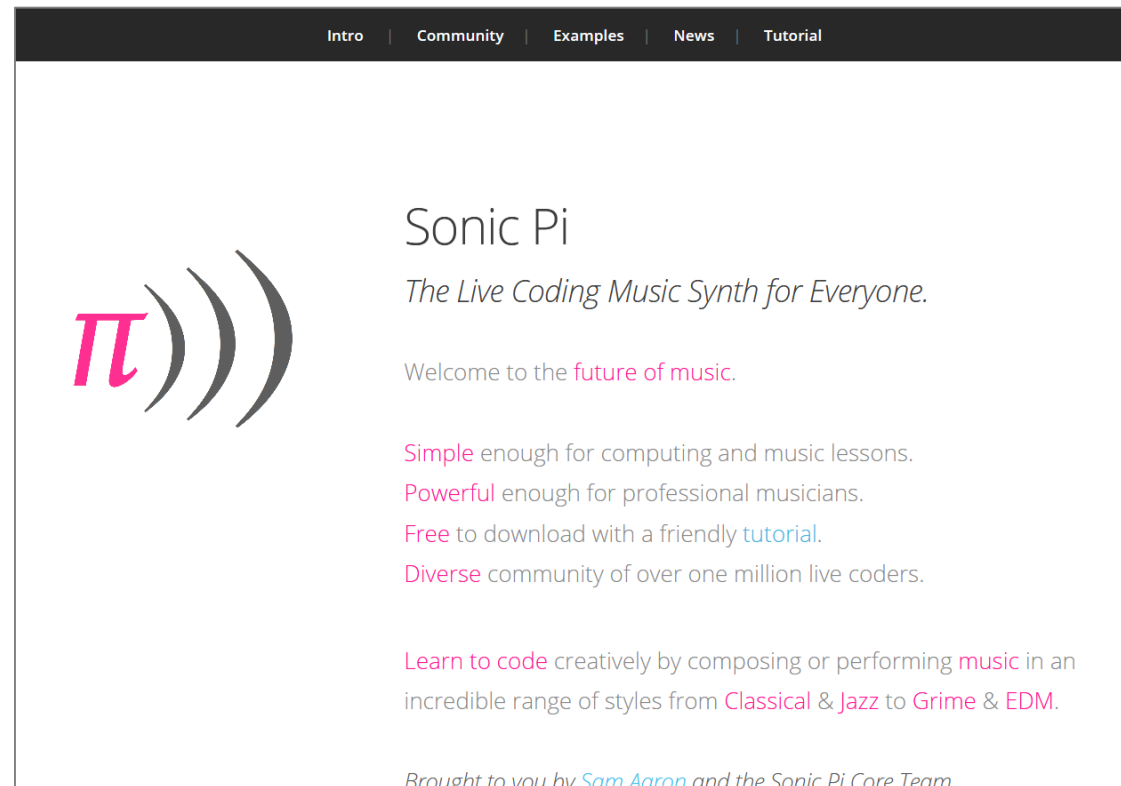
音を扱うプログラミング環境、MITライセンスのOSS(Open Source Software)

ライブコーディング可能なIDE(統合開発環境)

ruby/Qt/SuperColliderなどからできている

Raspberry Pi 財団の支援も受けている

<https://sonic-pi.net/>



<http://git.io/wlopi>

- Raspbian Stretchでは標準インストールされている
- 以前のRaspbianからアップグレードした場合など

```
$ sudo apt install sonic-pi
```

▪ Windows/MacOSの場合

Sonic Pi is available free for:

Windows

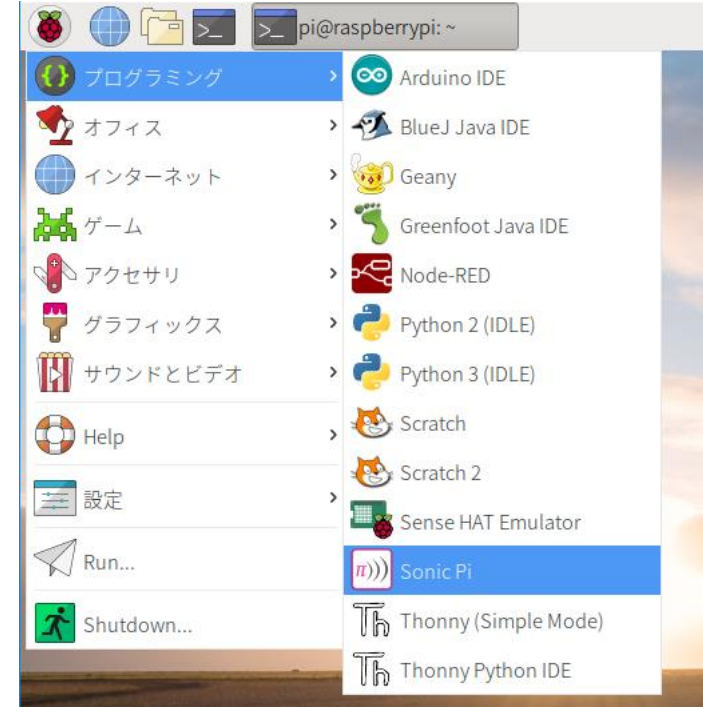
macOS

Raspberry Pi

Linux

<https://sonic-pi.net/#windows>

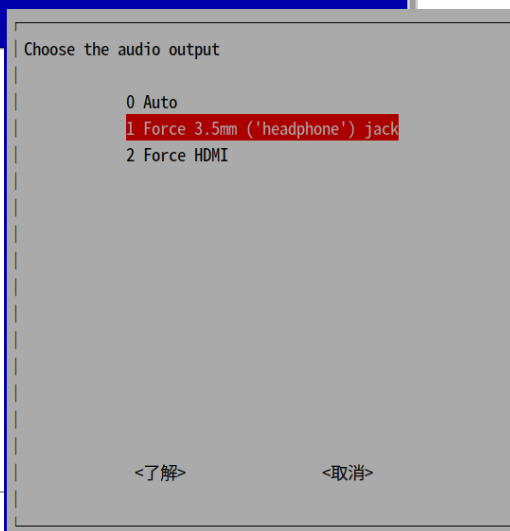
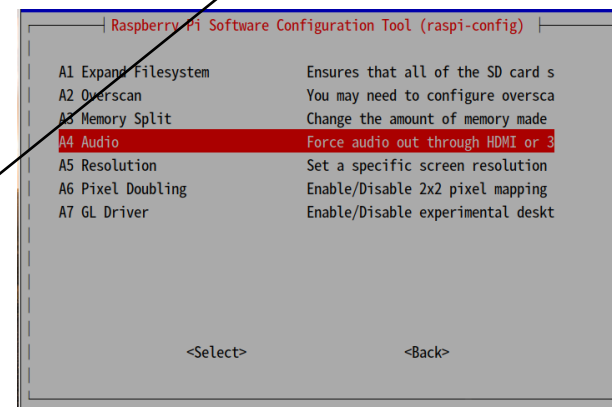
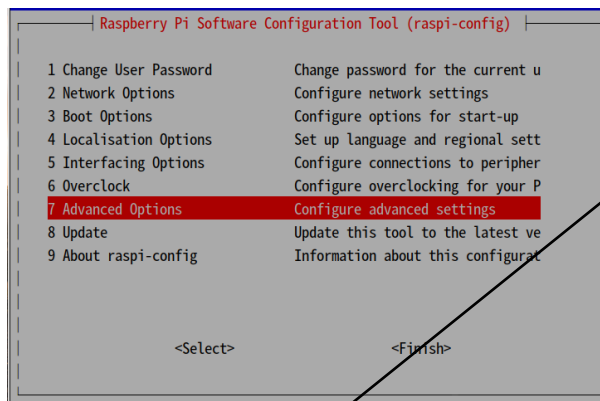
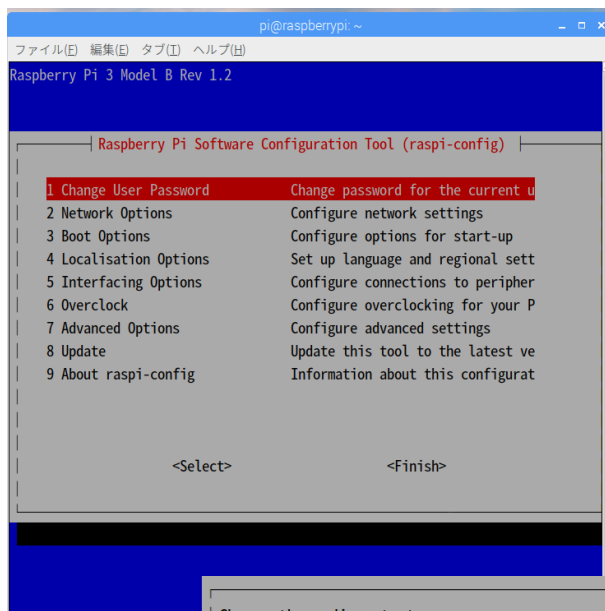
<https://sonic-pi.net/#mac>



<https://sonic-pi.net/>

・標準では、raspberrypi CLI設定(ターミナルから起動)でHDMIとオーディオジャックの切り替えができる

\$ sudo raspi-config → 7 Advanced Options → A4 Audio



1. Force 3.5mm ('headphone') jack

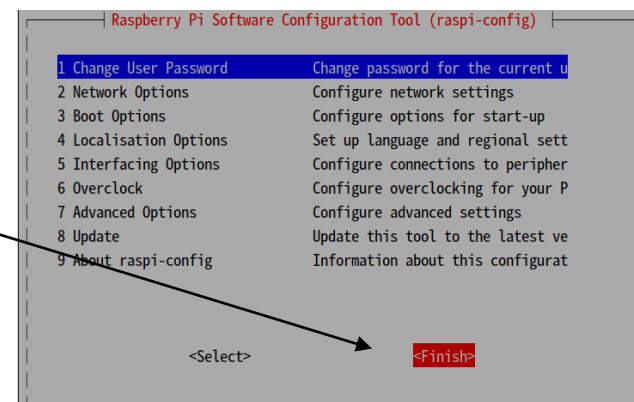
選択してEnter
(もしくは選択

→TAB

→<了解>でEnter

TAB -TAB

→<Finish>でEnter



• /boot/config.txtの設定

/boot/config.txt の `dtoverlay=audio=on` をコメントアウトすることで
USBその他の機器からオーディオ出力が出る

```
pi@raspberrypi:~$ tail /boot/config.txt

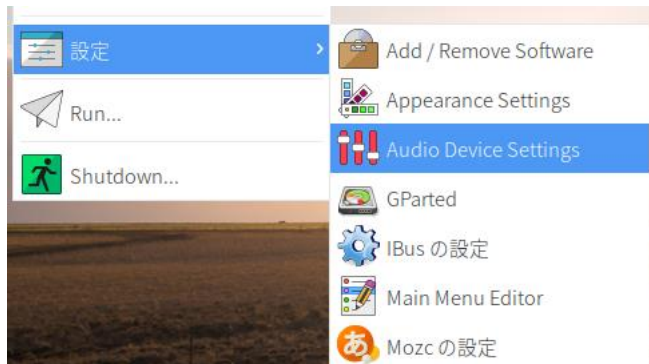
# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi
dtoverlay=lirc-rpi:gpio_out_pin=17,gpio_in_pin=18,gpio_in_pull=up

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
→ #dtoverlay=audio=on
enable_uart=0
```

• Audio Device Settings

設定メニューのAudio Device Settings

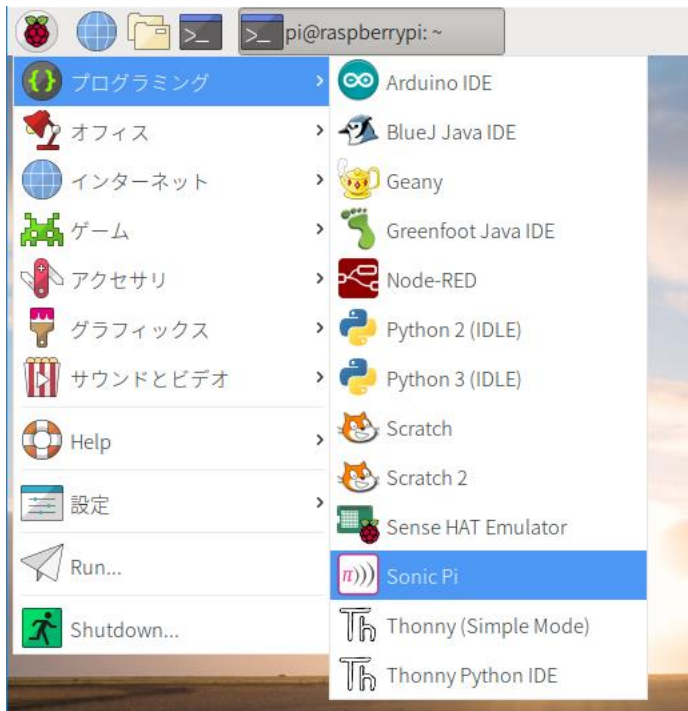


サウンドカードを選んで Make Defaultする



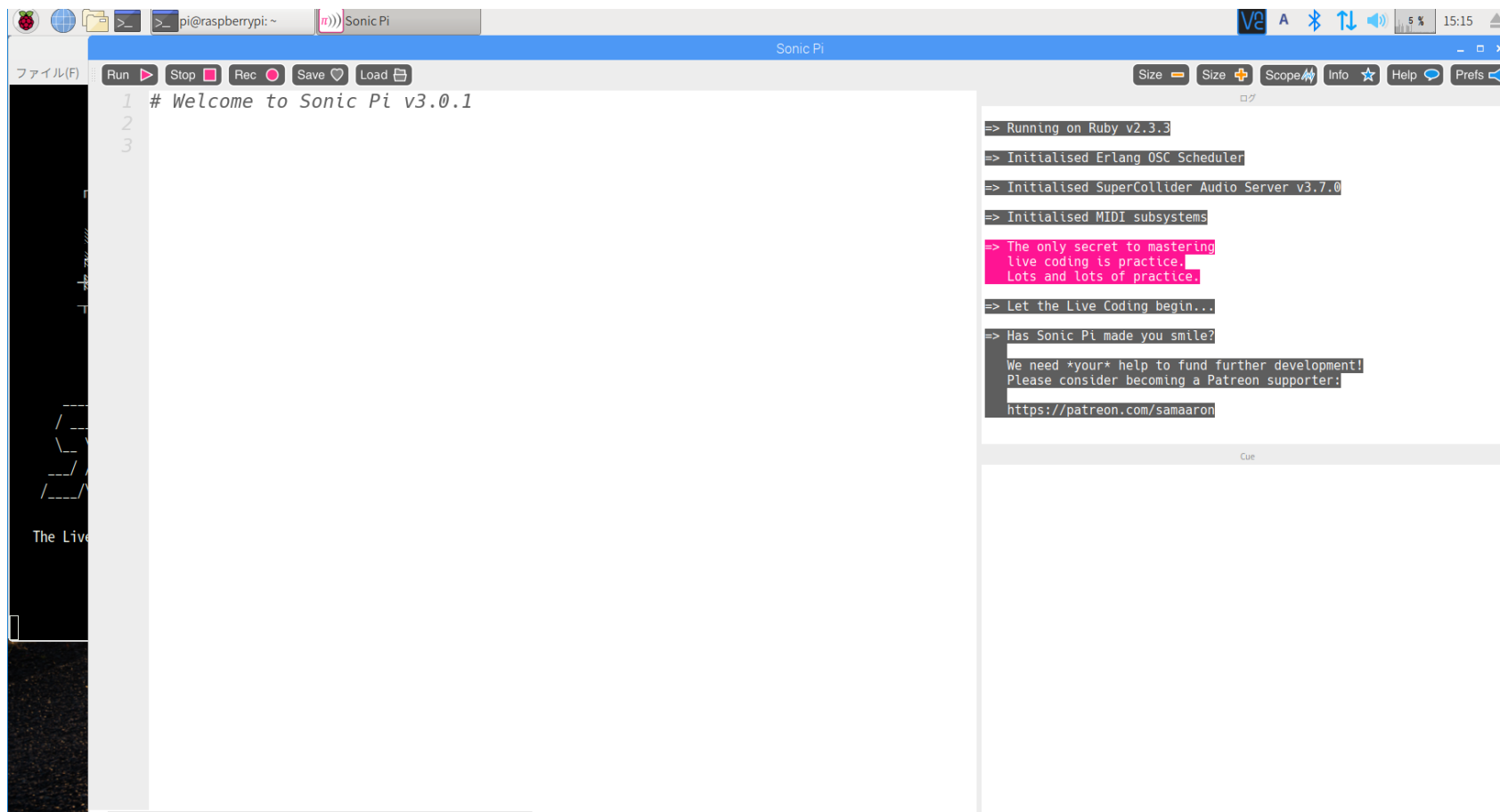
<http://git.io/wlopi>

- ・GUI起動の場合は「プログラミング」メニュー→Sonic Pi
- コンソール起動の場合は `$ sonic-pi`



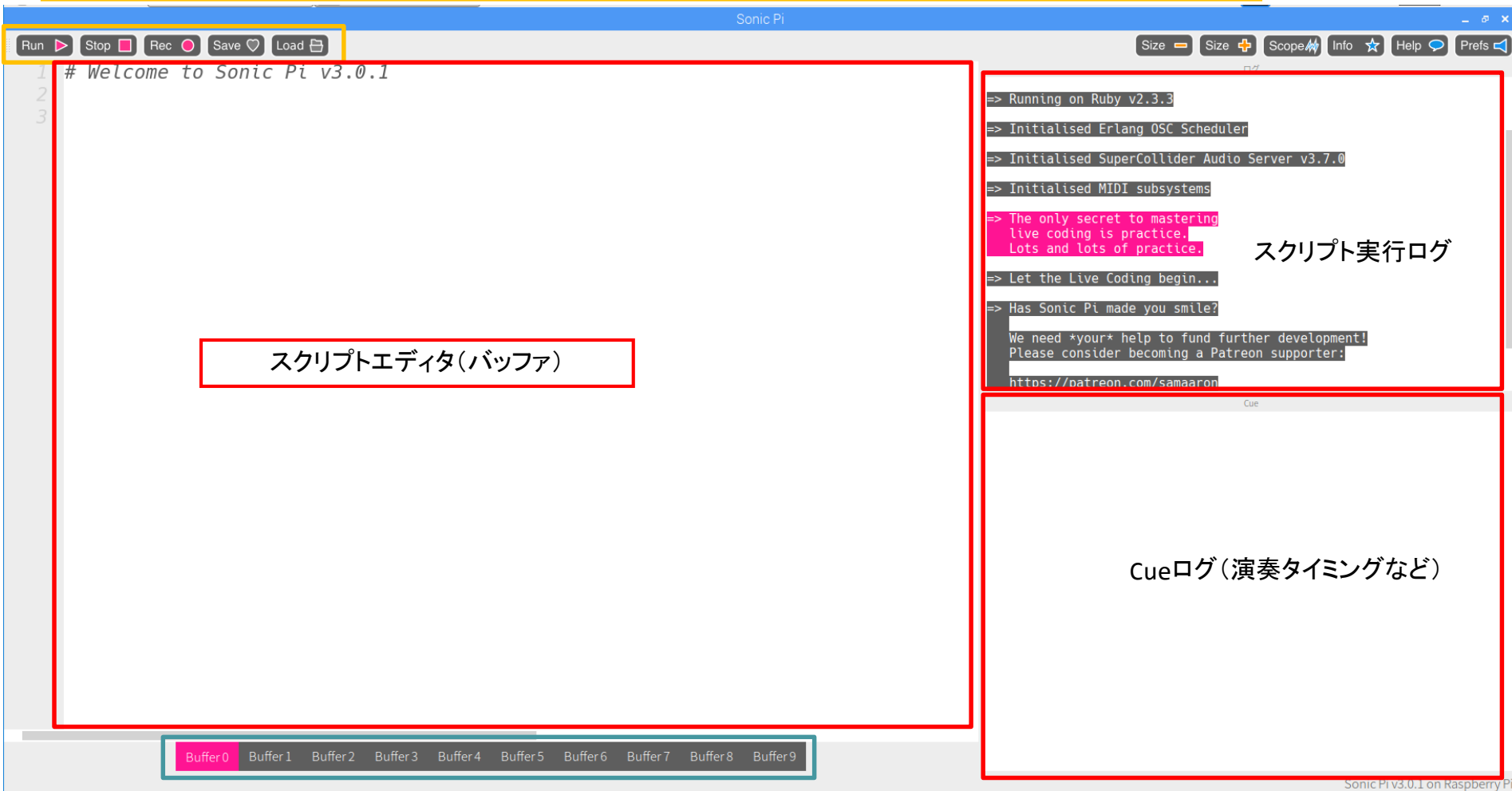
- ※起動には時間がかかる(30秒以上)
- ※Chromeなど大きなプログラムが起動していると起動時間がさらにかかることがある
- ※以前の設定があり起動しないことがある
その場合は、設定ディレクトリ
`/home/pi/.sonic-pi`
を削除する

- ・大きなウィンドウで表示(全画面が使いやすい)



プログラムボタン

Run: 実行 Stop: 停止 Rec: 実行中の音声を録音 Save: スクリプトの保存 Load: スクリプトの読み込み



バッファタブ切り替え

Scopeボタンで再生中の波形を表示できる

The screenshot displays the Sonic Pi application window. The top menu bar includes 'Run', 'Stop', 'Rec', 'Save', 'Load', 'Size', 'Scope', 'Info', 'Help', and 'Prefs'. The 'Scope' button is circled in red. Below the menu bar, the main area is divided into two panes. The left pane shows a code editor with the text '# Welcome to Sonic Pi v3.0.1'. The right pane is titled 'Mono' and contains a waveform plot with a y-axis ranging from -1 to 1. Below the plot, there is a text area showing the following output:

```
=> Running on Ruby v2.3.3  
=> Initialised Erlang OSC Scheduler  
=> Initialised SuperCollider Audio Server v3.7.0  
=> Initialised MIDI subsystems  
=> The only secret to mastering  
    live coding is practice.  
    Lots and lots of practice.  
=> Let the Live Coding begin...
```

At the bottom of the window, there is a buffer selection bar with buttons for 'Buffer0' through 'Buffer9'. The status bar at the bottom right indicates 'Sonic Pi v3.0.1 on Raspberry Pi'.



Run Stop Rec Save Load

Size Scope Info Help Prefs

1 # Welcome to Sonic Pi v3.0.1
2
3

Size Scope Info Help Prefs

Mono

1
0.5
0
-0.5
-1

=> Starting run 10
=> Completed run 10
=> All runs completed
=> Pausing SuperCollider Audio Server

Cue

Buffer0 Buffer1 Buffer2 Buffer3 Buffer4 Buffer5 Buffer6 Buffer7 Buffer8 Buffer9

ヘルプ

1 Sonic Pi へようこそ
1.1 ライブコーディング
1.2 インタフェースの探索
1.3 遊びを通じた学び
2 シンセ
2.1 初めての音
2.2 シンセのオプション
2.3 シンセの切り替え
2.4 エンベロープでのデューレーション
3 サンプル
3.1 サンプルを使う
3.2 サンプル・パラメータ
3.3 サンプルを引き延ばす
3.4 エンベロープ・サンプル
3.5 部分的なサンプル
チュートリアル 例 シンセ エフェクト サンプル 命令

ライブコーディング

Sonic Piの最もエキサイティングな側面のひとつは、まるでギターをライブで演奏するかのように、ライブでコードを書いて音楽を作ることができることです。つまり、ステージやコンサートでSonic Piが使えるということです！

心を解き放て

これからチュートリアルで、実際のSonic Piの詳しい使い方に入る前に、まず、ライブコーディングがどんなものか体験してみましょう。あまり（もしくは全然）わからなくても、心配ご無用！ そのまま席についたまま、楽しんでいきましょう。

ライブループ

さあ、はじめましょう！ 下のコードを上の方のBuffer（バッファ）にコピーしてみましょう。

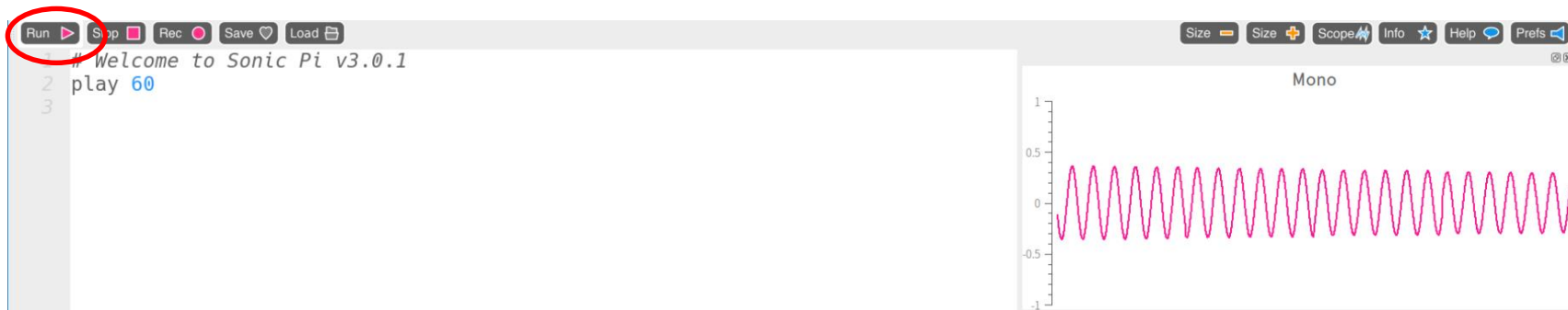
```
live_loop :flibble do
  sample :bd_haus, rate: 1
  sleep 0.5
end
```

左上のRun（再生）ボタンを押すと、いい感じの速さでバスドラムの音が聞こえてきます。Stop（停止）ボタンを押せば、いつでも音を止めることがで

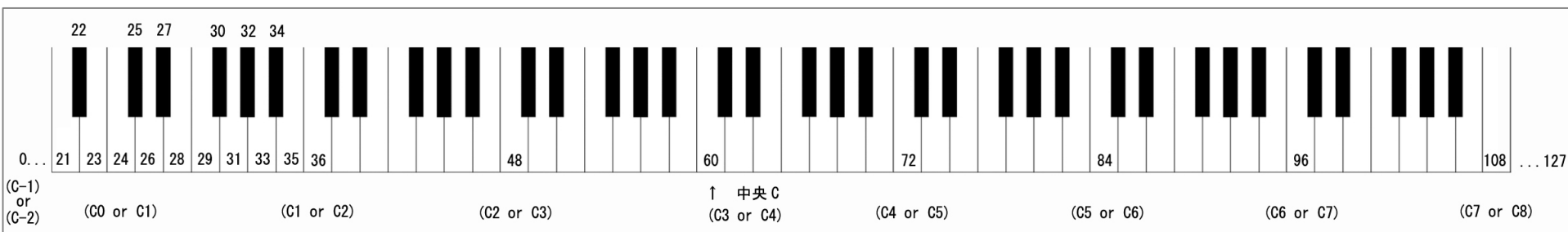
Sonic Pi v3.0.1 on Raspberry Pi

・最初のスクリプト

play 60



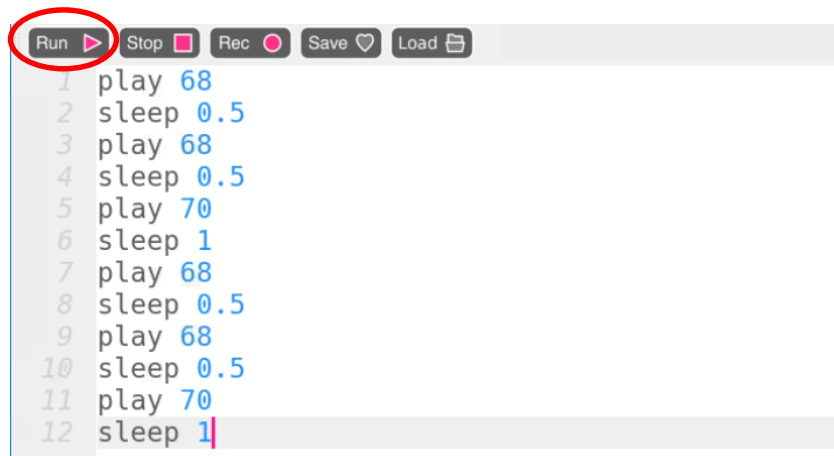
60は演奏する音程を表す。コードでの指定も可能。(:C4)
(音程はMIDIの指定と同じ。下記を参照)



https://qiita.com/santa_sukitoku/items/6585a84eeb01d2b8d3a3

• 順番に音を鳴らす

```
play 68  
sleep 0.5  
play 68  
sleep 0.5  
play 70  
sleep 1  
play 68  
sleep 0.5  
play 68  
sleep 0.5  
play 70  
sleep 0.5
```



次の行までの待ち時間をsleepで指定。単位は秒。

・ループ/繰り返し

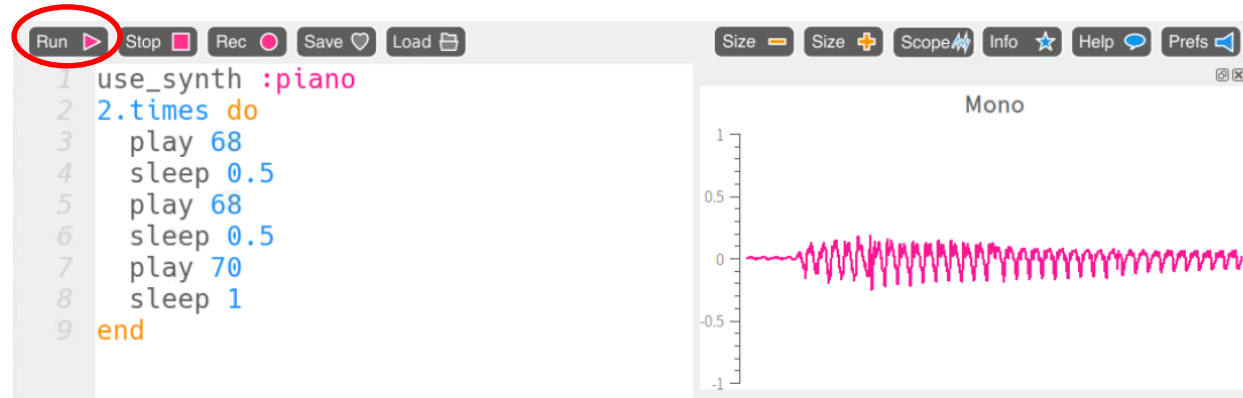
```
2.times do
  play 68
  sleep 0.5
  play 68
  sleep 0.5
  play 70
  sleep 1
end
```



do ~ end をtimesで指定した分だけ繰り返す

• ならす音源を変える

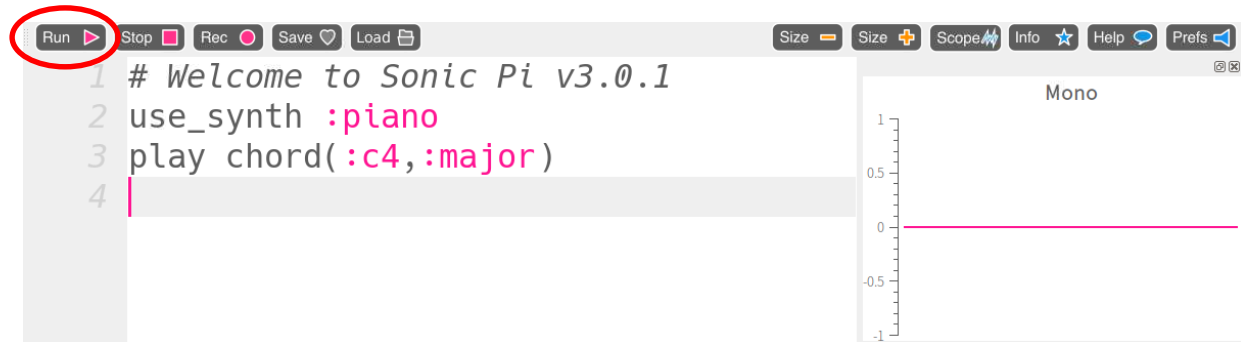
```
use_synth :piano
2.times do
  play 68
  sleep 0.5
  play 68
  sleep 0.5
  play 70
  sleep 1
end
```



use_synth 音源を指定

- 和音をならす

```
use_synth :piano  
play chord(:C4,:major)
```

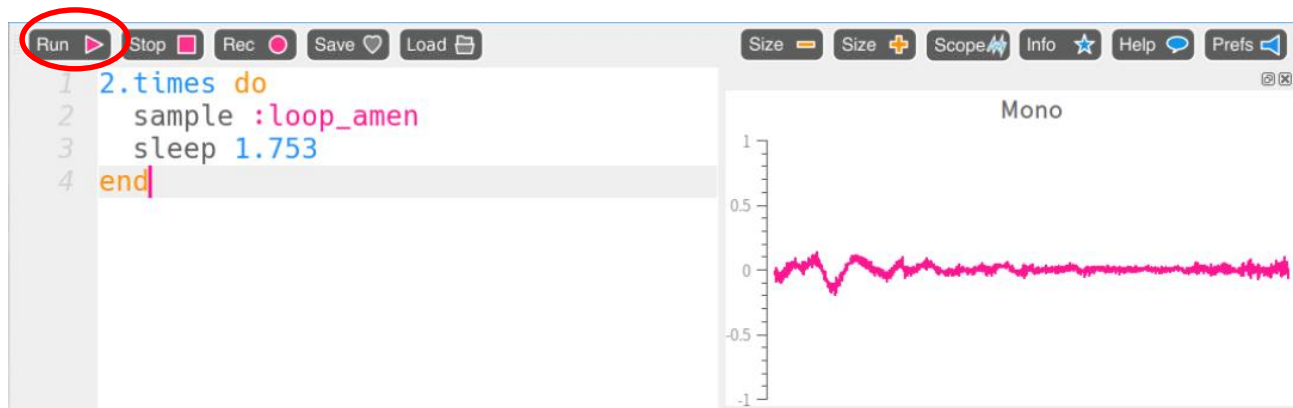


- パターンをならす

```
use_synth :piano  
play_pattern_timed chord(:C4,:major),0.25
```

・ サンプル音源をならす

```
2.times do
  sample :loop_amen
  sleep 1.753
end
```

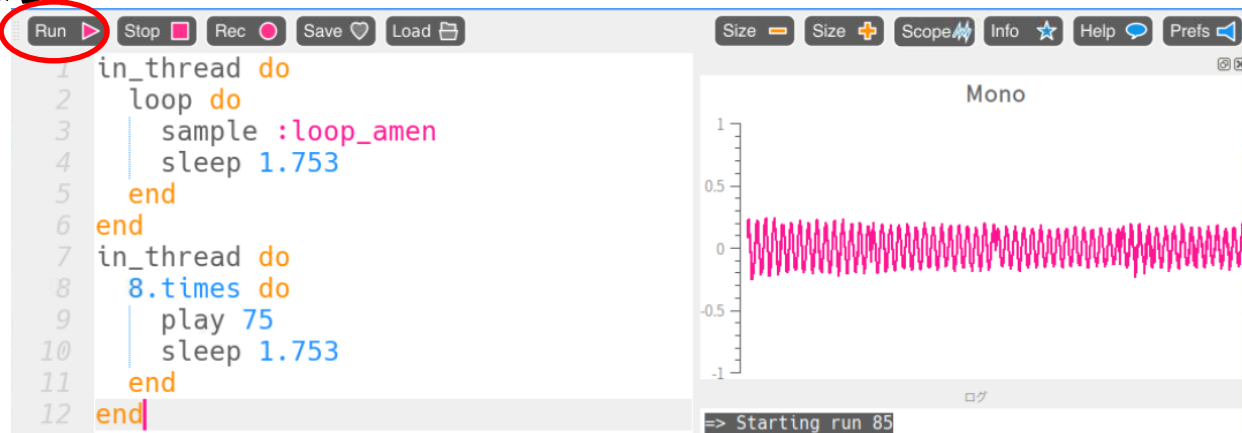


サンプル音源は `/opt/sonic-pi/etc/samples` にある

```
pi@raspberrypi:/opt/sonic-pi/etc/samples $ pwd
/opt/sonic-pi/etc/samples
pi@raspberrypi:/opt/sonic-pi/etc/samples $ ls loop*
loop_amen.flac      loop_compus.flac    loop_mika.flac
loop_amen_full.flac loop_garzul.flac     loop_safari.flac
loop_breakbeat.flac loop_industrial.flac loop_tabla.flac
pi@raspberrypi:/opt/sonic-pi/etc/samples $
```

・サンプル音源に重ねて音源をならす

```
in_thread do
  loop do
    sample :loop_amen
    sleep 1.753
  end
end
in_thread do
  8.times do
    play 75
    sleep 1.753
  end
end
```



`in_thread` で新しいスレッド作成して同時実行する

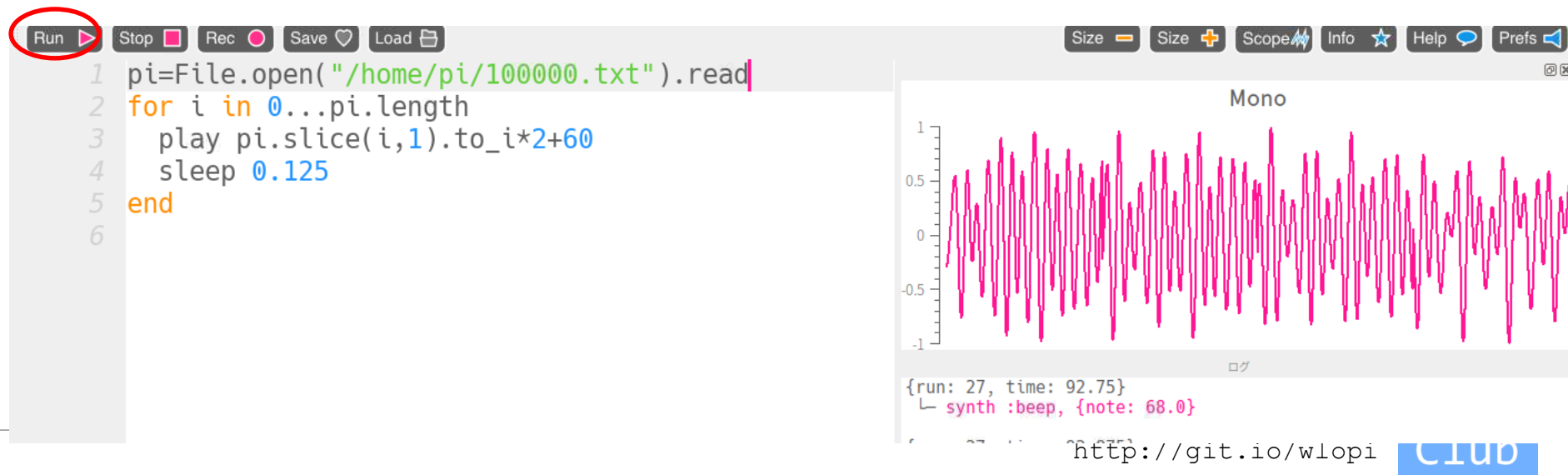
- ・ 円周率サウンド <http://www-b.uec.tmu.ac.jp/shakuhachi/SonicPi/#PiMusic>

<https://www.angio.net/pi/digits.html>

に円周率10/50/100/1000/10000/100000

のファイルがあるので、10万桁のファイルを/home/pi/100000.txt にダウンロード

```
pi=File.open("/home/pi/100000.txt").read
for i in 0...pi.length
  play pi.slice(i, 1).to_i*2+60
  sleep 0.125
end
```



The screenshot shows the Sonic Pi IDE interface. On the left, the code editor displays the following code:

```
1 pi=File.open("/home/pi/100000.txt").read
2 for i in 0...pi.length
3   play pi.slice(i,1).to_i*2+60
4   sleep 0.125
5 end
6
```

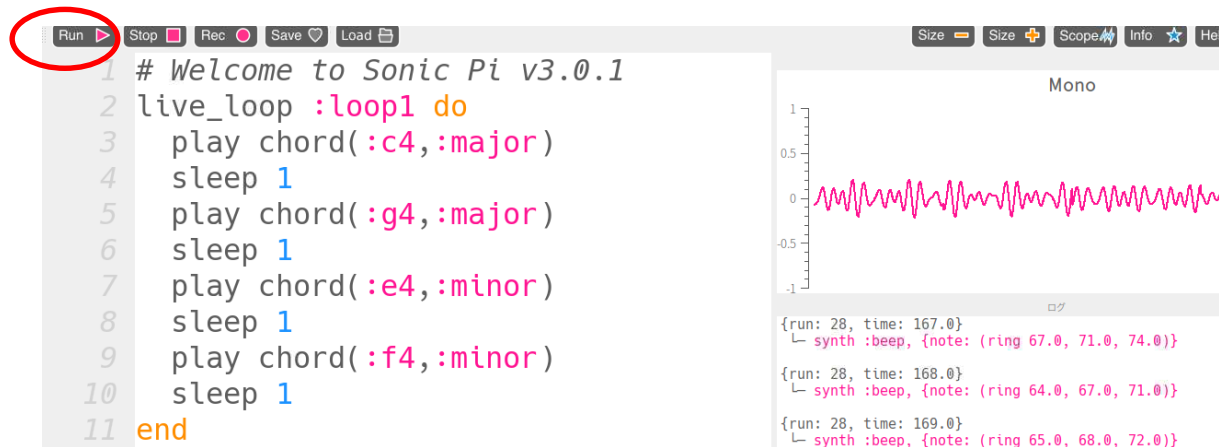
The "Run" button is circled in red. On the right, the "Scope" window shows a waveform plot titled "Mono". The plot displays a complex, oscillating signal. Below the plot, the log shows the following output:

```
{run: 27, time: 92.75}
└─ synth :beep, {note: 68.0}
```

At the bottom right, there is a URL <http://git.io/wlopi> and a blue button labeled "Club".

• Live Loop: 実行しながら変更

```
live_loop :loop1 do
  play chord(:c4,:major)
  sleep 1
  play chord(:g4,:major)
  sleep 1
  play chord(:a4,:minor)
  sleep 1
  play chord(:f4,:minor)
  sleep 1
end
```



エディタでスクリプトを編集し、Runを押すと
内容確認後、ループの次の回から更新される

• OSC (Open Sound Control) で外部からのコマンドを受け取る

コマンドライン送信ツールでテスト

```
$ sudo npm install -g osc-cli
```

```
pi@raspberrypi:~ $ sudo npm install -g osc-cli
/usr/bin/osc -> /usr/lib/node_modules/osc-cli/bin/osc
/usr/bin/osc-cli -> /usr/lib/node_modules/osc-cli/bin/osc
/usr/bin/osc-listen -> /usr/lib/node_modules/osc-cli/bin/osc-listen
/usr/lib
├─ osc-cli@0.2.0
│ └─ commander@2.0.0
│   └─ osc-min@1.1.1
│     └─ binpack@0.1.0
```

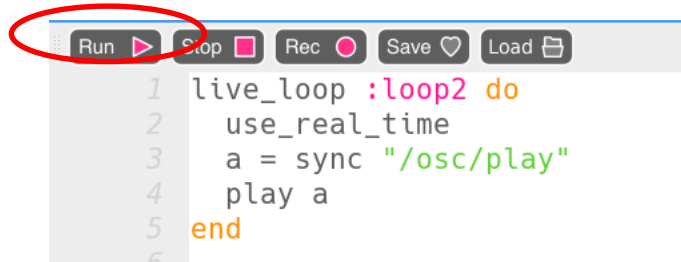
OSCの受信ポートは
Prefsの「入出力」で確認
(デフォルトは4559)

OSCメッセージをリモートから受信する
にチェックをいれると、同じLAN内からの
メッセージを受け取れる
チェックしないと
localhost (127.0.0.1)からしか
受信できない



• 外部からのパラメータをlive loopで変更

```
live_loop :loop2 do
  use_real_time
  a = sync "/osc/play"
  play a
end
```



コマンドラインから

```
$ osc-cli -host localhost:4559 /play 60
```

で、60の音が再生

```
pi@raspberrypi:~ $ osc-cli --host localhost:4559 /play 60
sent /play [60] to localhost:4559
```

• Node-REDから

node-red-contrib-osc 1.0.0

Open Sound Control (OSC) support for Node-RED

`npm install node-red-contrib-osc`

The `node-red-contrib-osc` module adds support to `Node-RED` for decoding and encoding OSC (Open Sound Control) messages.

This then makes it easy to bridge between other protocols, for example HTTP or MQTT.

Node-RED

OSC node

Info

Path

OSC Path

Metadata

Include metadata

Name

Name

Tip: leave path blank if you want to set using `msg.topic`.

Receiving

If the node receives a `Buffer` as `msg.payload` it will convert it and place the OSC address in `msg.topic` and the received arguments in `msg.payload`.

When "Include metadata" checkbox is checked, the OSC node will send an object with the type of the value along the value:

```
{
  "type": "f",
  "value": 33.4
}
```

Node Info

Version: 1.0.0

Updated 1 year, 7 months ago

License: Apache-2.0

Rating: ☆☆☆☆

View on npm

View on GitHub

Downloads

1 in the last day

18 in the last week

161 in the last month

Nodes

OSC

Keywords

OSC

Open Sound Control

node-red

<http://git.io/wlopi>

- ・公式サイト

公式サイト

<https://sonic-pi.net/>

チュートリアル

<https://sonic-pi.net/tutorial.html>

- ・Raspberry Pi 財団チュートリアル

<https://projects.raspberrypi.org/en/projects/getting-started-with-sonic-pi>

- ・デバイスプラス特集「IT女子のラズベリーパイ入門奮闘記」

http://deviceplus.jp/hobby/raspberrypi_entry_017/

- ・Sonic Piメモ

<http://www-b.uec.tmu.ac.jp/shakuhachi/SonicPi/>