“ゼロ”から学び，プログラミングを楽しもう

# インターフェースZERO No.05

見本

キーボード/モニタ/DVD/カメラをつないでLet'sプログラミング

スクリプト言語Pythonでサクサク動かす

# ラズベリー・パイで作る手のひらLinuxパソコン

中村 文隆　著

## 付属DVD-ROM内容

■Raspberry Pi用Linuxインストール関連パッケージ
Raspbian OSディスク・イメージ
NOOBSパッケージ
Windowsイメージ書き込みツール

■本書解説で使用したサンプル・ファイル
気圧センサTCPサーバ・スクリプト
温度記録スクリプト
ログHTML生成スクリプト
I²C読み出しサンプル・スクリプト　ほか

■本書解説で使用したツール
Teraterm+TTSSH
tightVNC 32ビットWindowsクライアント
tightVNC 64ビットWindowsクライアント
WinSCP(FTPクライアント)

CQ出版社

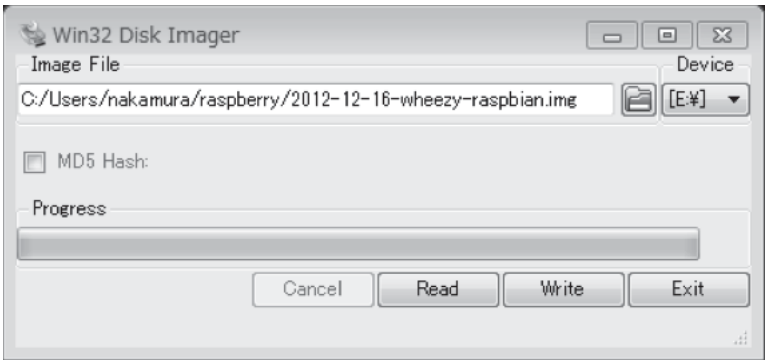



**図 1-8**
**Win32 Disk Imager の画面**

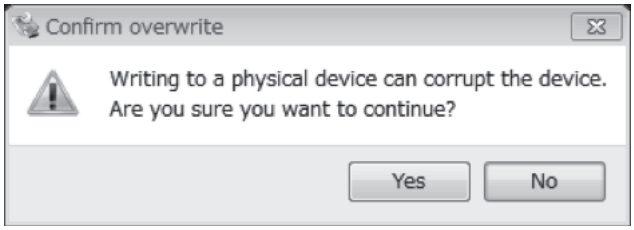



**図 1-9**
**ディスク・イメージ書き込み時の警告**

Write のボタンを押すと書き込みを開始しようとしますが，その際，**図 1-9** のような警告が出てきます．これは，「物理デバイスに書き込むとデバイスを壊してしまいますが，本当に続けますか」ということを言っています．対象ドライブに確証があれば，Yes を押して継続すると，SD メモリ・カードにシステムが書き込まれます．

解凍したイメージ・ファイルは 1.8GB ほどあり，SD メモリ・カードの Class6 の読み書きの速度は，それほど高速とはいえないデバイスなので，書き込みが完了するまでには何分かかかります．高速とされる Class10 でも多少短縮されますが同様に数分かかります．

書き込みが成功すると，Write Successful のダイアログが出ます．念のため，SD メモリ・カード・ドライブに対して右クリックから「取り出し」を選択し，安全に取り出すことができる，というメッセージが表示されてから SD メモリ・カードを抜いて準備完了です．

### ▶ Mac OS X の場合

アプリケーションからユーティリティを開き，ターミナル・アプリケーションを起動します．ターミナル内で，diskutil list とタイプして Enter を押すと，ディスクの一覧が出てくるので，SD メモリ・カードを挿して，サイズから SD メモリ・カードのデバイス名を探します．デバイスの名前は /dev/disk ○，のようになっていて，○の部分にはアルファベット 1 文字が入ります．例えば，8GB の SD メモリ・カードを使うのであれば，8GB 程度のサイズのドライブを見つけて，上記の○にあたる文字が何になっているかを diskutil list の結果から見つけてメモしておきます．

SD メモリ・カードのデバイス名がわかったら，書き込みを行うために SD メモリ・カードのマウントを解除します．ターミナル内で，diskutil unmountdisk /dev/disk ○とタイプして Enter することで，SD メモリ・カードのマウントが解除され，イメージの書き込みができるようになります．その後，

```
dd if=(イメージ・ファイル) of=/dev/disk○ bs=2m
```

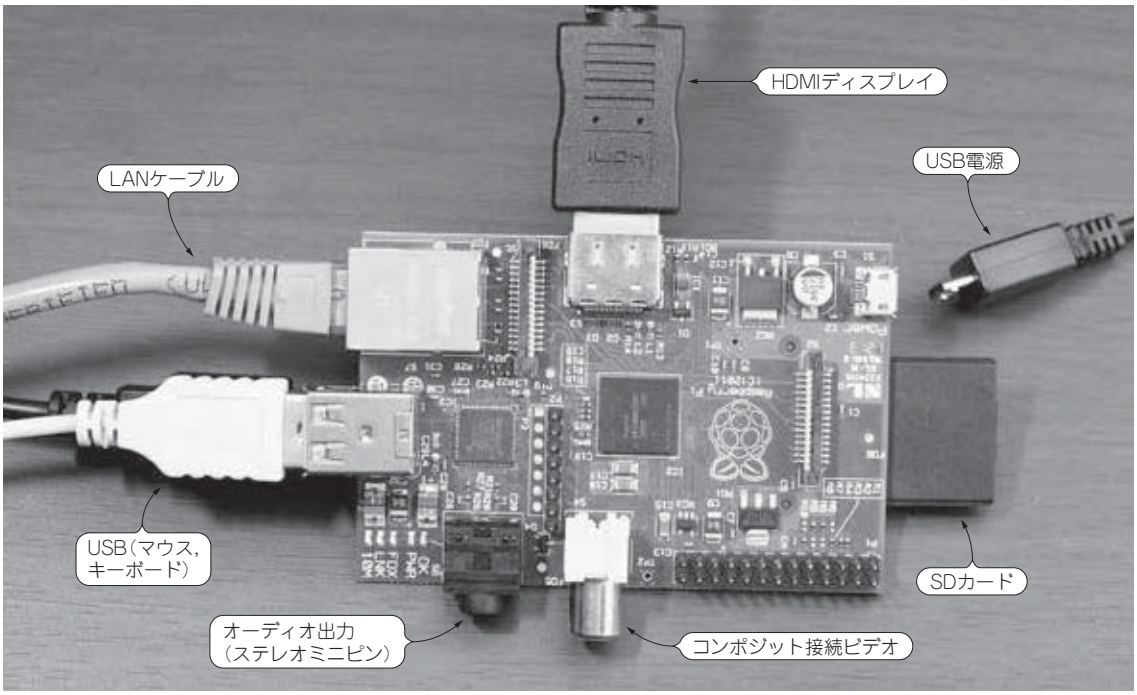



**図 1-10　ラズベリー・パイに主なケーブルを接続したようす**

とタイプして Enter すると，SD メモリ・カードにディスク・イメージが書き込まれます．ここで，(イメージ・ファイル) は，ダウンロードして解凍しておいたディスク・イメージ・ファイルの名前で，あらかじめ解凍したファイルが置いてあるフォルダに cd コマンドで移動しておくか，解凍ファイルへのパスを省略せずに指定する必要があります．

以上の作業で，Mac OS X での SD メモリ・カードへのディスク・イメージの書き込みが行われます．

**▶ Linux の場合**

diskutil list の代わりに fdisk -l というコマンドを使うことと，SD メモリ・カードのアンマウトに umount コマンドを使う点を除けば，手順は Mac OS X の場合とほぼ同じです．なお，Mac OS X，Linux ともに，SD メモリ・カードをパソコンに挿入した際に，OS が自動的にマウントするように設定されている場合のみ，diskutil umount，もしくは umount コマンドの実行が必要になります．設定によっては，SD メモリ・カードを挿入しても，明示的にマウント操作をしなければマウントされないようになっていることがあります．その場合にアンマウント操作をするとエラーになりますが，ブート・ディスクを作るうえでは特に問題になりません．

### 1-4-3　起動と初期設定

**▶ ケーブルと SD メモリ・カードの接続**

ここまでの説明で，ラズベリー・パイの標準的な構成でブートするために必要なものは揃いました．電源は最後に接続するので，電源以外のものをすべてラズベリー・パイに接続します．ラズベリー・パイに電源スイッチは用意されておらず，電源を供給する USB コネクタを接続すると ON 状態になり，すぐにブートが始まります．

HDMI でディスプレイを接続した場合のケーブル接続は，**図 1-10** のようになります．コネクタの形

# 第3章 ラズベリー・パイの外部入出力端子

外部入出力端子に接続したデバイスを制御したり，接続したセンサからデータを取り込んでデータ処理することができます．外部入出力端子として備えてある，UART(RS-232C)，GPIO，SPI，I²Cの概要，デバイスやセンサの接続方法とデータ処理方法を解説します．

## 3-1 外部入出力端子の概要

ラズベリー・パイは，外部入出力のための端子をいくつか備えています．これらの端子にセンサやLEDなどの電子部品を接続することによって，ラズベリー・パイから制御したり，ラズベリー・パイにデータを取り込んだりすることができます．ここでは，いくつかの簡単な例を通して，ラズベリー・パイの外部入出力の使い方を見ていきます．

ラズベリー・パイには，**図3-1**のように左上に26本のピン・ヘッダが2列に並んだI/O端子が実装されています．ここで，左や上というのは，ラズベリー・パイの表面に印刷されているラズベリー模様が正立して"Raspberry Pi"の文字が読める向きに置いたときの方向で言っています．

この26本のピン・ヘッダは，**図3-2**のように，ラズベリー・パイの基板上，P1の文字が印刷されている位置のピンがP1-01，その上がP1-02という順に，P1-26まで番号を付けて区別されています．それぞれのピンは，いくつかの異なった目的に用いられるようになっていて，大別すると以下のようになります．

- +5Vの電源
- +3.3Vの電源
- 0V（GND）
- UART（RS-232C）
- GPIO（ディジタルのIN/OUT）
- SPI通信
- I²C通信

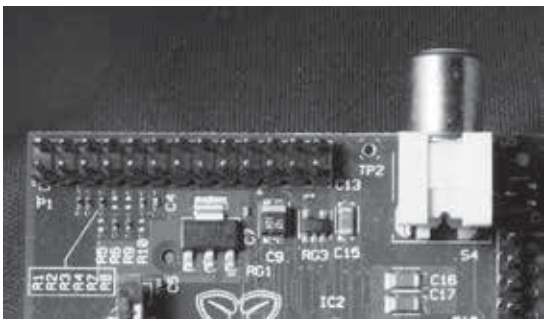



図3-1 ラズベリー・パイのI/O端子

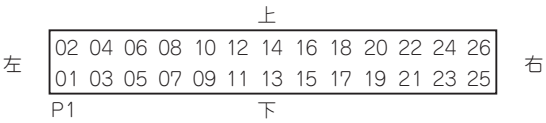



図3-2 I/O端子のピン番号

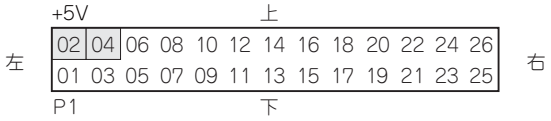
これらのうち，+5V の電源の扱いには特に注意してください．ラズベリー・パイの I/O 端子は，入力電圧の上限が +3.3V で，保護回路が内蔵されていないため，+5V の電源端子が，I/O の他の端子とショートすると，ラズベリー・パイが壊れる恐れがあります．この +5V の電源端子は，外部に接続するパーツが +5V の電源電圧を必要としている場合に使われることがありますが，その場合でも，外部に接続したパーツからラズベリー・パイへの入力電圧が +3.3V を超えないようにする必要があります．

### 3-1-1　+5V 電源

+5V 電源は，ラズベリー・パイ自体に電力を供給している USB の 5V 電源をそのまま出しています．このため，+5V 電源から取り出すことのできる最大電流は，

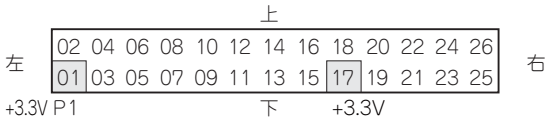
（+5V 電源からの電流の総和）=（USB 電源の最大供給電流）− 700mA

となります．USB 電源に余裕のない場合は注意してください．また，いま述べたように，+5V 電源の外部パーツからの入力を I/O 端子で受ける場合は，入力電圧が +3.3V を超えないよう，自前の保護回路を付ける必要があります．+5V 電源は，次の 2 か所から取り出すことができます（P1-02，P1-04）．



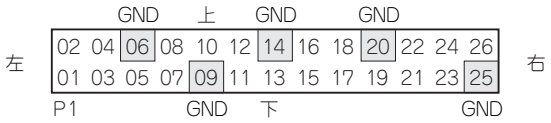

### 3-1-2　+3.3V 電源

+3.3V 電源は P1-01 と P1-17 で，この二つを合わせて最大 50mA までの電流を流すことができます[53]．+3.3V の電源は以下の位置にあります．




### 3-1-3　0V（GND）

GND は 0V の基準電位となるピンで，P1-06，P1-09，P1-14，P1-20，P1-25 の 5 か所があります．GND は基準電位を与えるラインになるため，例えば +5V 電源から大きめの電流を流すような場合は，GND のピンを多めに使うようにすると気分的に良いかもしれません．

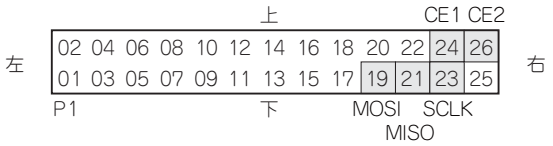



### 3-1-4　UART（RS-232C）

UART は，RS-232C の通信に使われるピンで，P1-08 が送信，P1-10 が受信のピンになります．RTS などのフロー制御や DTR などの接続状況を確認するピンは出ていないため，接続は TXD，RXD と GND の 3 本のみの簡略版となります．また，このピンも 3.3V のロジック・レベルで動作するので，そ

[53] 通常は使わない，P2 の + 3.3V を使う場合はそれも含めて最大 50mA となるので注意．

場合は，二つのスレーブ IC までの接続を行うことができます．

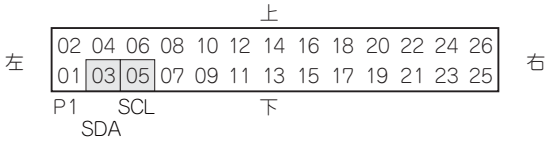



### 3-1-7　I²C

I²C は Inter-Integrated Circuit の頭文字をとったもので，I²C と 2 を上付きとして 2 乗を意味する表記とするのが正式ですが，上付き文字が使えない環境もあるため，しばしば I2C と表記されています．I²C では信号線は 2 本で，SDA（Serial Data）でデータ伝送を行い，SCL（Serial Clock）でクロックを伝送します．

接続はパーティ・ラインと呼ばれる方式で，接続する IC の数が増えても信号線は 2 本のままで済み，各 IC に割り当てられているアドレス（デバイスの ID のようなもの）で通信相手を選択するようになっています．アドレスは 7 ビットで，128 個のうち，I²C の規格上で予約されていて割り当てることができない 16 個を除いた 112 個が，理論上接続できる IC の最大数となります．

ラズベリー・パイでは，P1-03 が SDA，P1-05 が SCL に割り当てられています．




以上が，ラズベリー・パイの外部入出力端子の種別と概要となります．続いて，センサや LCD など，具体的な外部デバイスの接続を例にとって，実際の使い方を説明します．

### 3-1-8　準備するもの

最初に，外部にパーツを接続して実験するために共通して揃えておきたいものを説明します．

センサや LCD などのパーツをラズベリー・パイに接続するためには，**図 3-3** のようなジャンパ・ケーブル（ジャンプ・ワイヤ）が便利です．**図 3-3** には，ケーブルの両端がオス-オス，オス-メス，メス-メス[54] の 3 種類があります，これは，接続する対象についている接続端子の形状の組合せに応じて，適切な組みあわせのケーブルを用意することになります．

新規にケーブルを購入する際には，可能であればオス-メスのタイプを購入するとよいでしょう．ラズベリー・パイ側の P1 にはピン・ヘッダが並んでいるため，ラズベリー・パイ側はメス端子を使うと接続が楽になります．対向側は，この先ブレッドボードのようにオス端子を差し込むパーツを使うことも多いため，オス-メスのケーブルがあるとなにかと便利です．

また，実験によっては簡単な回路を組むことが必要になりますが，その際には**図 3-4** のようなブレッドボードを使うことになります．ブレッドボードは，ジャンパ・ケーブルや抵抗，コンデンサ，センサにはんだ付けするピン・ヘッダなどを差し込むことができるようになっていて，差し込んだ部品間をジャンパ・ケーブルで接続することで回路を組むことができるようになっています．本書で行う実験は小規模なものなので，ブレッドボードとしては**図 3-4** 程度の小さなものでも十分です．

センサをブレッドボードに差し込んだり，ジャンパ・ケーブルでラズベリー・パイと接続したりするときには，**図 3-5** のようなピン・ヘッダをセンサ基板にとり付けることがよくあります．このピン・

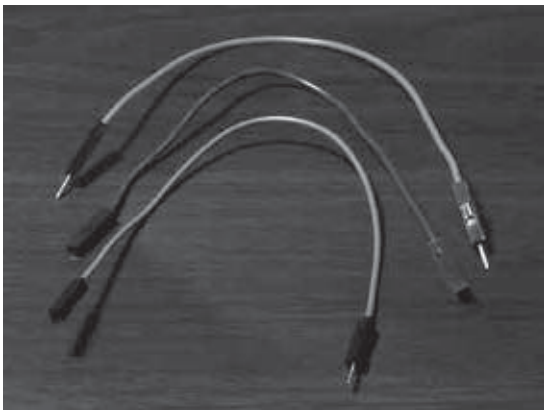


図 3-3　ジャンパ・ケーブル

図 3-4　ブレッドボード

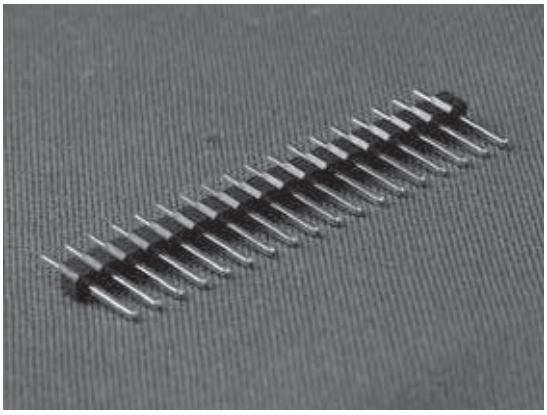


図 3-5　ピン・ヘッダ

図 3-6　工具類

ヘッダは，センサを購入したときにおまけで付いてくることもありますが，付いていないものも珍しくはありません．ピン・ヘッダはニッパで必要な数にカットすることができるので，適当なピン数のものを手持ちで用意しておくと安心です．

このほか，センサ・パーツにピン・ヘッダを取り付ける程度のちょっとしたハンダ付けが必要になります．ハンダごてと糸ハンダ，ハンダごて台は，ぜひ揃えておきましょう．ハンダごては，電力の高いものは避け，20 ～ 30W 程度のこて先の細いものを用意します．**図 3-6** のハンダごては 23W のものです．糸ハンダは，電子部品用と記載のある径の細いハンダが使いやすいでしょう．ハンダごて台は，スポンジなどに水を含ませることができるものがお勧めです．こて先にハンダが残った場合の拭取りや，こて先が熱くなり過ぎた場合にこて先の温度調節に使います．

**図 3-6** にはニッパとラジオ・ペンチもありますが，とりあえずニッパは用意しておいたほうがよいでしょう．ラジオ・ペンチは，なにかのときにあると便利ですが，必須というほどはありません．

図に出ていないものでは，テスタがあるとハンダ付けのチェックなどに使うこともできますが，これがなくても実験に支障はありません．ニッパやペンチ，テスタはホームセンタや 100 円ショップなどで売られている安価なものでも十分です．

続いて，センサや LCD など，具体的な外部デバイスの接続を例にとって，実際の使い方を説明します．

0 はヨーロッパなどで使われている PAL 方式です．日本は 1 の NTSC なので，動画をキャプチャして保存する場合は 1 にしておきます．ストリーミングだけであれば関係ありません．

### 4-1-7　動画と静止画の保存

この節で motion を導入した際に，最初に動画と静止画の記録を off にしましたが，motion はライブ映像の定期的な記録と，動体検知を行って検知があったときのスナップ・ショットを保存する機能を持っています．

以下に，関係する設定項目を列挙していきます．

| | |
|---|---|
| snapshot_interval | ライブ映像のスナップ・ショット間隔．0 にすると記録しない |
| ffmpeg_cap_new | 動体検知時に動画を記録するかどうか |
| output_normal | 動体検知時に静止画を出力するかどうか |

なお，ffmpeg_cap_motion や output_motion は， それぞれ ffmepg_cap_new や output_normal と似ていますが，生の画像ではなく検出時の演算画像に近いものを保存します．通常は，normal のついている項目を設定したほうがよいでしょう．

上記で動画や静止画を記録する場合，フォーマットやファイル名についての設定も必要に応じて行っておきます．これに関しては設定項目が多いので，ここではキーワードだけを挙げておきます．

動画に関しては，ffmpeg で始まる設定項目で，video_codec を mpeg4 にする，bps や variable_bitrate を必要に応じて設定する，などがあります．静止画に関しては，quality の項目が jpeg の保存クオリティの設定なので，適度にサイズを抑えるように設定したりするとよいでしょう．保存されるファイル名の設定は，最後が _filename で終わる設定項目になります．

ライブ・カメラは便利で楽しいものですが，プライバシなどの問題が生じることもありますし，自動生成される動画や静止画ファイルでハード・ディスクがいっぱいになってしまうトラブルもあり得ます．運用する場合は放置せず，運用状態をよく把握しながら行うようにしましょう．

## 4-2　ピコボード＋スクラッチ

### 4-2-1　スクラッチ

スクラッチは，MIT メディアラボで作成された教育用のプログラム環境で， ラズベリー・パイの Wheezy では，デスクトップ環境に最初からインストールされています．

**図 4-7** がスクラッチを起動したときの初期画面で，真ん中の「スクリプト」の中に，左側のプログラム・パーツを配置していくことで，右側のスプライト（ここでは猫の画像）を制御します．

**図 4-7** の左上は**図 4-8** のようになっていて，八つの分類項目の中に「調べる」という項目があります．これは英語環境では "Sensing" となっていて，この中に，この後で導入するピコボードというセンサ・ボードからの入力を受け取るパーツが**図 4-9** のように入っています．

**図 4-9** ではスライダ・センサの値となっていますが，▼をクリックすると**図 4-10** のように，スライダ，明るさ，音，抵抗- A ～ D，傾き，距離の項目があります．これらのうち，ピコボードではスライダ，明るさ，音，抵抗- A ～ D をそれぞれ取得して，スクラッチのスクリプトの中で使用することができます．

### 4-2-2　ピコボード

ピコボードは Spark Fun から販売されているセンサ・ボードで，国内ではスイッチサイエンスや千

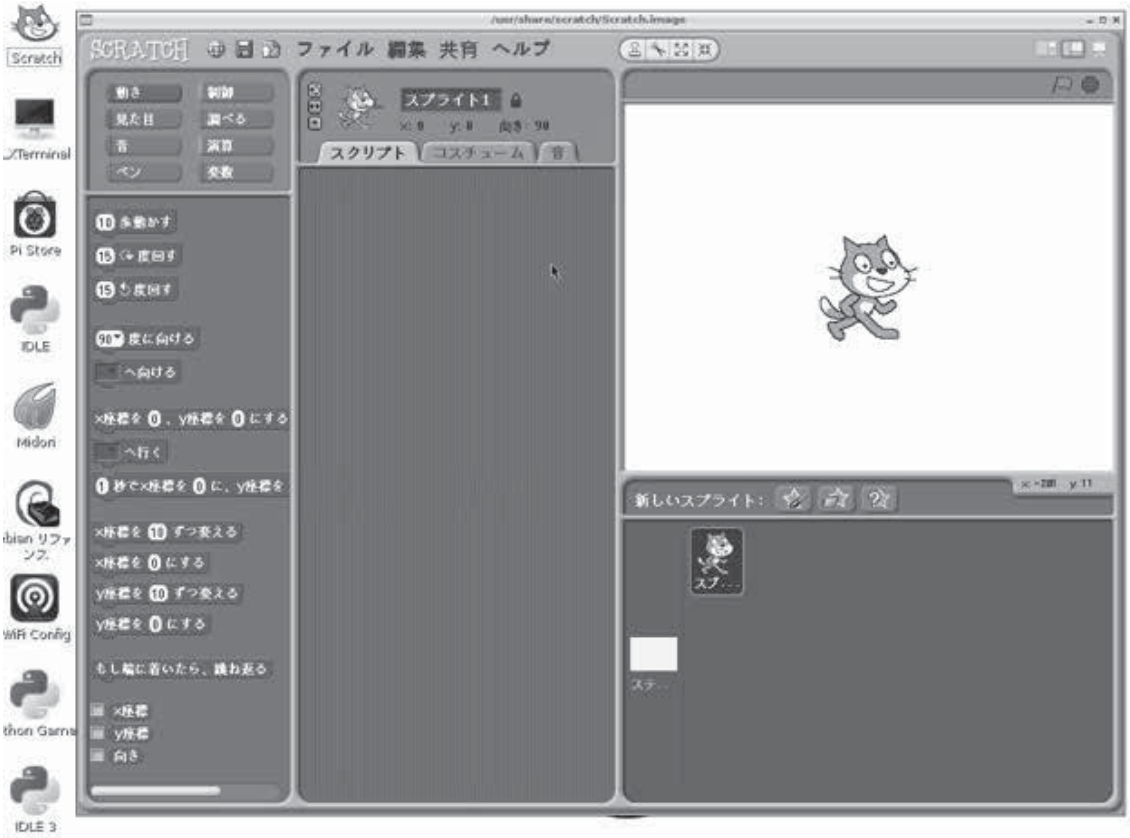



図 4-7　スクラッチの起動画面


図 4-8
スクラッチのプログラム・パーツ項目


図 4-9
スクラッチの「調べる」内にあるセンサ入力パーツ

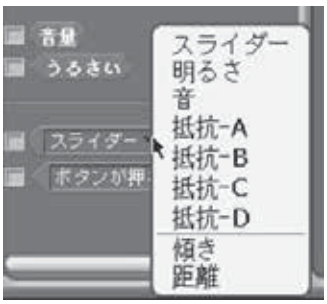



図 4-10　センサの種類

石電商などから数千円程度で入手することができます．ピコボード本体は**図 4-11** のようなもので，奥側にあるツマミがスライド抵抗，右寄りに目のような記号が描かれた位置に光センサ[97]，左下の耳のような記号が描かれた位置に音センサ[98]，左側の A ～ D の $\phi$2.5 ピン・ジャックに +5V に 10kΩ を介してプルアップされた A - D 入力[99] 4 系統が，それぞれ配置されています．

A ～ D には，**図 4-12** 左側のような，ワニ口クリップ ×2 から 2.5$\phi$ ピンに配線されたケーブルを

97　フォト・トランジスタ TEMT6000.
98　コンデンサ・マイク MIC9.7MM.
99　ATMega328P の A-D 入力で，10bit A-D.


**図 4-18**
**発表モード**

```
with sprite2
  while  true
    if  lightsensor.value < 20 then
      .show
    else
      .hide
    endif
  end while
end with
```

という雰囲気になるでしょう．暗くなったら幽霊が出る，という設定です．

このスクリプトを**図 4-17** のように実行状態にして部屋を明るくし，ピコボードの光センサに手をかざして光を遮ると幽霊が現れ，手をどけて光があたるようにすると幽霊が消えることを確かめることができます．

スプライト画面には，「調査する（Sensing）」のパーツの値をリアルタイムに表示することもできるので，**図 4-18** のようにセンサの値を表示させながら，スクラッチ左上の「発表モード」を選ぶと，用意したスプライトがすべて実行状態になり，全画面表示で全体の動作をみることができます．

スクラッチは直感的で学びやすく，ピコボードの使い勝手も悪くありません．ここではスライド抵抗と光センサだけを使ってみましたが，音センサや 4 系統の A-D も同様に簡単に使うことができます．

# 4-3 Node.js

ここでは，前章で作成した気圧センサの値を TCP/IP 経由で送信するサーバと，node.js という仕組みを使ってサーバから値を読み出すクライアントを作成してみます．

## 4-3-1 フレームワーク

想定するフレームワークは**図 4-19** のようなものです．Python スクリプトで SPI，I²C，GPIO を経由してセンサの値を読み取り，TCP/IP で任意のプログラムからアクセスできるようにする，というシン


**図 4-19　Python プログラムを介して，センサ値を TCP/IP 経由で読み出す**

プルなフレームワークです．

Python は，SPI などのセンサへのアクセスと，TCP/IP のサーバのどちらも比較的簡素な記述で実現することができます．外部のアプリケーションで TCP/IP を使った通信がサポートされていれば，このフレームワークを使うことでセンサの値をアプリケーションで利用することができるようになります．

### 4-3-2　Python スクリプト

Python スクリプトは，**図 4-20** のようになります．

```
def getPressure():
```

の行から，

```
import socket
```

までの範囲は，大気圧センサのところで作成した，読み出しと補正の処理を関数として定義したものです．getPressure () として呼び出すことで，気圧を hPa 単位で表した数値を返すようになっています．

```
import socket
```

以降の記述がサーバ部分で，クライアントが，

```
p を送信　→　気圧値を返す
q を送信　→　接続を終了する
それ以外　→　Unknown command を返す
```

というシンプルなものです．

なお，**図 4-20** のスクリプトは，一つのクライアントとのみ通信を行うもので，クライアントとの通信が終了するとサーバ・スクリプトも終了します．複数のクライアントとの多重化通信を行うスクリプトについては，Appendix を参照してください．

**図 4-20** では，

```
myaddrport = (socket.gethostbyname('localhost'),3000)
```

と指定しているので，localhost のポート 3000 で [102] サーバが待ち受け状態になります．

**図 4-20** のスクリプトを入力し，例えば prs_srv.py のような名前で保存します．

[102] localhost の IP アドレスは 127.0.0.1 となる．

見本