Vol.2015-CE-131 No.3
2015/10/10

ラズベリーパイを用いたドリトルでのデータ計測とデータ蓄積サーバの提案

林 康平[1,a)]　西川 弘恭[1,b)]　小林 史弥[1,c)]　間辺 広樹[2,d)]　大村 基将[1,e)]　兼宗 進[1,f)]

概要：小型のコンピュータである Raspberry Pi 上でドリトル言語を実行し，GPIO や I2C によって外部接続された温度や気圧などのセンサ値を読み込んで利用する仕組みを作成した．Raspberry Pi は Linux で動作しているため，入出力はデバイスファイルにアクセスすることで行い，対応する命令をドリトルに用意した．計測した値はドリトルのグラフィックス機能を使い画面上でグラフ表示を行うほか，計測値を記録するログサーバをインターネット上に用意し，ドリトルから HTTP 通信でデータをリアルタイムに送信する仕組みを作成した．試作した環境を高校の授業で利用してもらい，高校生が理解して使用できることを確認した．

キーワード：ドリトル, Raspberry Pi, データ計測

Hayashi Kohei[1,a)]　Nishikawa Hiroyuki[1,b)]　Kobayashi Fumiya[1,c)]　Manabe Hiroki[2,d)]
Omura Motomasa[1,e)]　Kanemune Susumu[1,f)]

1. はじめに

ドリトルの Raspberry Pi 計測・制御用ライブラリおよびデータ蓄積サーバの実装と提案を行う．

現在，モノのインターネット（Internet of Things : IoT）という言葉に代表されるように，あらゆるモノがネットワークと接続し，あらゆる時間，あらゆる場所で，情報と接続することができる社会となってきた．私達自身も，スマートフォンなどに代表されるネットワークと接続可能なコンピュータが組み込まれた小型端末を日常的に利用している．また，このような端末では，音声認識による操作に対応したり，腕時計型で脈拍を測り体調などを管理するしたりする機能を備えるものが登場するなど，環境の変化や状況を計測し情報として利用することが行われ始めている．しかしながら，従来から行われているコンピュータ単体でのプログラミング教育では，必ずしも情報社会の現状を反映しているとは言えない．このような状況を踏まえると，これからの情報教育においてはセンサや LED などとの入出力を用いた制御に加え，ネットワーク通信を体験的に学習することができる教材が求められていると考えた．そこで本研究では，GPIO を持ち OS が利用可能なシングルボード PC である Raspberry Pi[5] を，初等・中等教育の現場において利用されているプログラミング言語「ドリトル」[1] により制御し，接続したセンサによる計測や制御する

1 大阪電気通信大学
Osaka Electro-Communication University, Shijonawate, Osaka 575–0063, Japan
2 柏陽高等学校
Hakuyo High School
a) ht13a072@oecu.jp
b) ht13a065@oecu.jp
c) ht13a036@oecu.jp
d) manaty2005@mh.scn-net.ne.jp
e) esetanuki@gmail.com
f) kanemune@acm.org

図 1 Raspberry Pi2 本体

ことを考えた．ドリトルは，初等中等教育における実践から，プログラミング経験のない生徒がプログラムを作成できることが知られている [2]．本稿では，Raspberry Pi の IO に接続されたセンサ等の電子部品をドリトル上から制御する方法を検討する．続いて，接続されたセンサ等から得られた計測値を Web 上に設けたデータ蓄積サーバに転送する方法を検討する．

2. Raspberry Pi とプログラミング環境

2.1 ラスベリーパイ

Raspberry Pi は，英国 Raspberry Pi 財団が開発し，若年層のコンピュータ技術への関心やスキルを高めるなどの教育目的から開発された安価なシングルボードコンピュータである．用途に合わせて利用できるように，性能・価格等が異なるいくつかのエディションが存在する．本研究で利用した Raspberry Pi 2 Model B では，カードサイズの基板上に CPU として ARM プロセッサと，1GB のメモリ，HDMI や LAN，USB を搭載している．ARM Linux 等の OS を動作させることも可能となっており，パーソナルコンピュータとして運用することも可能である．また，デジタル入出力や I^2C，SPI といった各種ペリフェラルを持ち，46 ピンの汎用入出力（GPIO）に電子部品を接続することで，電子回路の制御やセンサから環境情報を取得することも可能である．Raspberry Pi2 本体の画像を図 1 に示す．

Raspberry Pi は様々な OS を動作させることが可能であるが [6]，今回は Raspberry Pi の標準イメージ環境である Raspbian を採用した．Raspbian は Debian をベースとした Linux 系の OS であり [7]，2014 年 9 月 9 日版より Java SE Development Kit 8 が標準搭載されている [8]．そのため，Java で作成されたドリトルのバイナリを追加のソフトウェアのインストールを行わずに動作させることが可能である．旧モデルと比べ性能が上昇している [9]Raspberry Pi 2 Model B においては，ドリトルが実用的な速度で動作することを確認した．

表 1 Raspberry Pi のポート一覧

機能	番号		機能
3.3V	1	2	5V
GPIO2(I2C_SDA)	3	4	5V
GPIO3(I2C_SCL)	5	6	GND
GPIO4(GPCLK0)	7	8	GPIO14(UART_TXD)
GND	9	10	GPIO15(UART_RXD)
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10(SPI_MOSI)	19	20	GND
GPIO9(SPI_MISO)	21	22	GPIO25
GPIO11(SPI_SCLK)	23	24	GPIO8(SPI_CE0)
GND	25	26	GPIO7(SPI_CE1)
ID_SD	27	28	ID_SC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

2.2 GPIO の配置と制御

Raspberry Pi2 の GPIO はデジタル入出力としての機能だけではなく I^2C 等のペリフェラル利用のための端子としても利用することができる．GPIO の端子割り当てと配置を表 1 に示す．なお，本研究で取り扱う LED やセンサなどの電子部品はすべてこの GPIO ピンに接続する．Raspbian 等の OS では GPIO やセンサを扱う機能を提供しており，制御にあたっては，デバイスファイルへ読み書きを行うことで，ポートの初期化や値の取得，セットを行うことができる．

3. ドリトルによる Raspberry Pi 制御の目的と課題

ドリトルは兼宗らが開発したオブジェクト指向の教育用プログラミング言語である [1][2]．日本語による構文を特徴としており，プログラミング初学者への教育活動などに利用されている．ドリトルは Java VM 上で動作し，Raspberry Pi においても Raspbian を OS とすることで，基本機能を利用可能である．そこで，ユーザによるドリトル単独での GPIO に接続した電子回路やセンサの制御や計測を実現するとともに，センサ等の計測結果などをインターネットを利用してサーバ上に蓄積することで，自身あるいは友人が作成した計測機からの計測結果を PC やスマートフォン等で自由に確認したり，計測データを自由に収集し加工できるプログラミング環境の実現を目的とする．

このとき，ドリトル上から GPIO などの Raspberry Pi 固有の各種機能を制御を行うためにドリトルの言語命令の拡張が必要となる．その際には，従来のドリトルで行われ

表 2 選定したセンサ・IC 一覧

種類	センサ・IC 名
温湿度センサ	HDC1000
照度センサ	TSL2561
気圧センサ	LPS25H
モータドライバ	TA7291

る学習の延長線上として活動できることが望ましく，言語命令の拡張は通常のドリトルの構文や命令に準拠して作成することが期待される．また，教育現場に合わせてセンサ等を随時追加する必要等も考えられるため，Raspberry Pi 固有の機能の利用方式はできる限り単純にして機能追加等が容易であることが望ましい．インターネット上のデータ転送と蓄積については，学校現場のネットワーク環境がポート制限等の多くの制約を持つケースがあることを考慮して，HTTP 等の標準的なプロトコルにて通信を実現する必要がある．これらの要素を踏まえた言語拡張方法を検討することが課題となる．

なお，ドリトルでの制御に対応したセンサ，IC を表 2 を示す．対応にあたっては，コストを含む入手性や，複雑な配線を必要とせずブレッドボード等を用いて生徒にも容易に回路を作成できるといった点からデータ線 2 本と電源を配線すれば良い I^2C に対応したセンサを優先的に対応した．対応センサについては，随時追加を行うことを計画している．

3.1 Raspberry Pi 固有機能のための言語拡張

言語拡張にあたっては，ドリトルの依存機能を提供するオブジェクトをライブラリとして構築する方法を選択した．ライブラリを含むシステム構成を図 2 に示す．ユーザはライブラリで定義されたメソッドを用いてプログラミングを行うことで，Raspberry Pi 固有の機能を利用することができる．ライブラリは，直接ユーザが利用するオブジェクト宣言部と，GPIO やペリフェラル（センサ）制御などの Raspberry Pi 固有の機能を実現する制御部からなる．前者はドリトルにより記述し，後者は主としてシェルスクリプトと Python により記述した．Raspberry Pi に接続されたセンサや LED の制御については新たに Raspberry Pi の固有機能を提供するラズパイオブジェクトをユーザに提供することで固有機能の利用を実現する．

プログラム言語の拡張にあたっては，従来のドリトル構文に準拠する形で行う．最も基本的なドリトルの構文例は以下となる．

かめた！ １０度 右回り。

ドリトルは後置的な表現が特徴であり，上記の例では，「かめた」がオブジェクト名，「右回り」がメソッド名となり，「１０度」はメソッドの引数となる．C 言語的な記述に置き換えると「かめた.歩く(10);」と等価である．

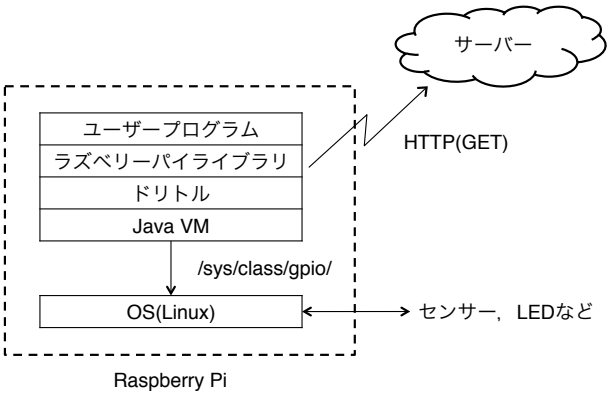

図 2 システム構成

なお，ドリトルにおいては英数字の全角・半角は同じ文字として扱われ，また，数字の後ろにつく文字列は無視される．

今回の言語拡張においてもこの構文を遵守する観点から，ラズパイオブジェクトから，さらにデジタル入出力やセンサ等を扱うためのオブジェクトを動的に生成し，生成したオブジェクトのメソッド呼び出しにより制御を実現できるように配慮した．

現時点において以下のような機能に対応している．

- デジタル入出力
- 温湿度センサからの値の取得
- 照度センサからの値の取得
- 気圧センサからの値の取得
- モータドライバの制御

なお，ユーザがライブラリの利用を行う場合は，以下のライブラリの利用宣言をプログラムの先頭で行う．

システム！ `"raspberry"` 使う。

以降，デジタル入出力，センサからの値取得，モータドライバの制御について詳細を報告する．

3.2 デジタル入出力

GPIO を用いてデジタル入出力を行う場合，以下の 3 つの手順を記述する．

(1) ポートの使用宣言
(2) ポートへの値の読み書き
(3) 後始末

ポートの使用宣言では，GPIO 初期化を行い，システム上に GPIO ポートを扱うためのデバイスファイルが作成される．ドリトル上では GPIO ポートを扱うオブジェクトが返される．以後の操作はこのオブジェクトを通して入出力を行う．

次の例では，GPIO4 番ポートを出力として扱うように使用宣言を行っている．

`LED` ＝ ラズパイ！ 4 出力ポート。

上記の例では，変数 LED 内に GPIO4 番ポートの出力

```
システム！ "raspberry"　使う。
LED ＝ラズパイ！ 4 出力ポート。
「
    LED ！ 1　書く。
    ラズパイ！　2 秒　待つ。
    LED ！ 0　書く。
    ラズパイ！　2 秒　待つ。
」！ 10 回　繰り返す。
LED ！後始末。
```

図 3　LED を 10 回点滅させるドリトルのプログラム例

表 3　GPIO ポート使用メソッドの一覧

メソッド	引数	機能
入力ポート	GPIO 番号	デジタル入力ポートを宣言する
出力ポート	GPIO 番号	デジタル出力ポートを宣言する

を扱うためのメソッドが定義されたオブジェクトが代入される．

ポートの初期化の完了後，ポートの値の読み書きが可能となる．ポートへの値の読み書きでは，GPIO ポートの出力のオン・オフの制御や，ポートに入力されている値の取得を行うことができる．GPIO ポートを扱うオブジェクトには，出力を操作する「書く」メソッドと値を読み取る「読む」メソッドが実装されている．これを利用して接続された LED を点灯させるには，次の命令を実行する．

```
LED ！ 1　書く。
```

GPIO の利用終了後は，初期化を行った GPIO ポートの開放を行う．生成したオブジェクトの「後始末」メソッドを実行することで GPIO ポートの開放が行われる．記述例を以下に示す．

```
LED ！後始末。
```

上記の例を用いて，図 3 に GPIO4 番ポートに接続された LED を 10 回点滅させるサンプルプログラムを示す．図 4 はサンプルプログラムを実行するための実体配線図である．

次のプログラムでは，最初に本対応のライブラリを読み込んでいる．そして，GPIO4 ポートをデジタル出力ポートとして設定し，オブジェクトを変数 LED に代入する．変数 LED に代入されたオブジェクトには，GPIO4 ポートを出力として制御するためのメソッドが定義されている．その後，GPIO4 ポートに対して ON，2 秒のスリープ，OFF，2 秒のスリープを 10 回繰り返し，最後に LED の後始末メソッドを実行することでポートの開放を行っている．

GPIO のポートを入力用/出力用として使用するためのメソッドを表 3 に示す．これらのメソッドを実行すると GPIO ポートの初期化が行われ，ポートに対応したオブジェクトが返される．

また，GPIO を扱うためのオブジェクトのメソッドは入力ポート，出力ポートをそれぞれ表 4，表 5 に示す．

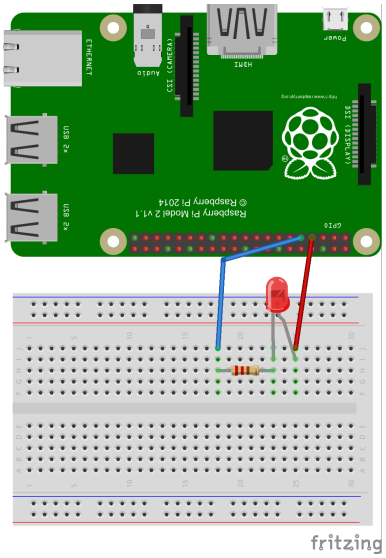

図 4　LED を点滅させる回路

表 4　入力ポートメソッドの一覧

メソッド	引数	機能
プルアップ	なし	ポートのプルアップ抵抗を有効にする
プルダウン	なし	ポートのプルダウン抵抗を有効にする
読む	なし	ポートの値を読み取る
後始末	なし	ポートの開放を行う

表 5　出力ポートメソッドの一覧

メソッド	引数	機能
書く	1/0	ポートに値を出力する
読む	なし	ポートの値を読み取る
後始末	なし	ポートの開放を行う

3.3　センサ

センサを用いる場合，以下の 2 つの手順を記述することにより実現する．

(1) 利用の宣言
(2) 値の読み取り

センサ利用の宣言ではラズパイオブジェクトから該当するセンサオブジェクトを取得する．温湿度センサを用いる場合の命令は以下となる．

```
温湿度センサー ＝ ラズパイ！ HDC1000。
```

センサから値を読み取りについては，値取得メソッドの戻り値として提供する．温湿度センサでは温度と湿度を読み取ることができるが，それぞれ対応した命令を呼び出すことで実現する．

```
気温 ＝ 温湿度センサー！温度？。
湿度 ＝ 温湿度センサー！湿度？。
```

センサの初期化を行う際は，ラズパイ！センサ名。と実行し，オブジェクトを再取得することで行える．また，各センサーが持つメソッドの一覧をそれぞれ表 6，表 7，表 8 に示す．

表 6 温湿度センサ HDC1000 のメソッドの一覧

メソッド	引数	機能
温度？	なし	温度を読み取る
湿度？	なし	湿度を読み取る

表 7 照度センサ TSL2561 のメソッドの一覧

メソッド	引数	機能
照度？	なし	照度を読み取る

表 8 気圧センサ LPS25H のメソッドの一覧

メソッド	引数	機能
気圧？	なし	気圧を読み取る
気温？	なし	気温を読み取る

3.4 モータドライバ

モータドライバを扱う場合は GPIO の手順と同様の 3 つの手順で行われる．

初めに，以下の通りの命令にて，モータドライバ TA7291 のオブジェクトを作成する．

モータードライバ ＝ ラズパイ！ TA7291。

次に，ポートの初期化を行う．モータドライバは 2 つの信号を用いてモータの回転方向を制御する．初期化にはモータドライバの制御端子と接続した GPIO ポート番号を引数として次の命令を実行する．

モータードライバ！ ３ ４ 作る。

初期化完了以降，モータの駆動が可能になる．モータ制御命令は回転方向に応じて用意されており，モータを正転させるには，次の命令を実行する．

モータードライバ！正転。

モータ制御修了時は，GPIO 同様に後始末を行う．後始末には次の命令を実行する．

モータードライバ！後始末。

4. Raspberry Pi 対応ライブラリの実装

4.1 GPIO へのアクセス

Raspberry Pi の GPIO ピンへアクセスは，ARM のペリフェラルを制御するレジスタを直接制御する方法や，OS 上のデバイスファイルにアクセスする方法が考えられるが，ここでは，後者の制御方法にて実装を行った．Raspberry Pi は，以下の手順に沿ってファイルアクセスを行うことで GPIO ピンの入出力を制御できる．（これらのファイルは/sys/class/gpio フォルダに存在する）

(1) `export` に制御したいポート番号を書き込む
(2) `gpio`（番号）`/direction` に入出力の方向を書き込む
(3) `gpio`（番号）`/value` に値を書き込み，読み込みを行う
(4) `unexport` にポート番号を書き込みポートを開放する

この手順をそれぞれシェルスクリプトとして実装した．そして，ライブラリ上の各手順のメソッドが実行された際にドリトルから外部のシェルコマンドなどを実行するシステム命令を用いてシェルスクリプトを実行することにより GPIO ポートの制御を行っている．また，実行中のプログラムが何らかのエラーのために途中で停止した場合，GPIO ポートの開放が行われない．そのため，初期化の段階で一度ポートの開放を実行するようにしている．

```
#!/bin/sh
if [ $# -ne 1 ]; then
    exit 1
fi
echo $1 > /sys/class/gpio/export
```

図 5 GPIO 初期化のシェルスクリプト

作成した GPIO 制御用スクリプトの例として，GPIO の初期化を行うシェルスクリプトを図 5 に示す．

4.2 I²C へのアクセス

ドリトルから Raspberry Pi の I²C へ接続されたセンサが接続されているかの確認および値の取得には Raspberry Pi に内蔵されているコマンドを用いる方法，もしくは，他のスクリプト言語のライブラリを用いて取得する方法がある．例えば，センサが接続されているかを確認するには `i2cdetect` というコマンドを用いる．このコマンドは I²C バス上に接続された機器のアドレスを取得するコマンドである．I²C バスに接続するセンサにはそれぞれアドレスが割り当てられている．このアドレスはセンサの種類毎に固有，もしくは数種類からの選択となっているため，設定したアドレスの数値で `grep` を行うことでセンサが接続されているかを確認することが可能である．例えば，温湿度センサである HDC1000（アドレス番号 0x40）が接続されているかを確認する場合，以下のコマンドを実行する．

```
i2cdetect | grep 40
```

本ライブラリでは，センサ毎のデータ取得にはセンサ毎のライブラリが充実している Python 言語を用いて実装を行った．こちらも，GPIO と同じくライブラリ上からドリトルのシステム命令を用いて Python スクリプトを実行して値を取得している．

5. データ蓄積サーバ

データ蓄積サーバとは，センサなどで計測したデータを Raspberry Pi などのインターネットに接続できる機器を用いることで，Web 上のサーバにデータを蓄積するサーバである．このサーバに蓄積されたデータは CSV 形式のデータとしてダウンロードを行うことができ，ローカルで Excel 等を使い詳細な分析を行うことが可能である．さらには，ブラウザー上でグラフとして表示したり，他人のデータや複数の地点で計測したデータとの比較を行える様になる予定である．

表 9 データ蓄積サーバ　パラメータ

パラメータ	概要	詳細
user	ユーザ名	
filename	ファイル名	
systemtime	システム時間	計測時．UNIX 秒を想定
dataN	データ	N はデータ番号（1,2,...）

```
2015-9-10,06:04:09,1441121386,24.5241,54.235
```

図 6 サーバからダウンロードしたデータ

5.1 仕様

データ蓄積サーバは，PHP で書かれた Web 上のアプリケーションとして実装されている．サーバ上にはユーザ毎のディレクトリが作られ，送信したデータをユーザの任意のファイル名で保存することができる．また，データは任意の数同時に送信することが可能であり，データの他にデータを送信した時刻（UNIX 秒），サーバ上の日付と時刻が保存される．

現在の実装では，存在していないユーザやファイル名を指定すると自動的にディレクトリやファイルが作成され，存在しているユーザやファイル名を指定すると追記されるようになっている．

5.2 利用方法

データ蓄積サーバにデータを送信する場合，Web 上の API に対して表 9 のパラメータを URL の末尾に指定して以下の URL に GET リクエストを送信する．

```
http://サーバー URL/api.php
```

例えば，ユーザ名を TestUser，ファイル名を test.csv，システム時間を 1441121386 に指定して 2 つのデータを送信する場合，以下のようなパラメータとなる．

```
?user=TestUser&filename=test.csv
&systemtime=1441121386&data1=24.5241&data2=54.235
```

ここで，データの送信と保存に成功した場合には送信されたデータと送信元 IP アドレスが返される．失敗した場合は error と返される．

データ蓄積サーバに保存したデータをダウンロードする場合，次のような URL にアクセスすることでカンマ区切り（CSV）形式でデータをダウンロードすることができる．

```
http://サーバー URL/[ユーザ名]/[ファイル名]
```

例えば，上記の送信例で用いたパラメータでのデータをダウンロードする場合には以下の様になる．

```
http://サーバー URL/TestUser/test.csv
```

上記の送信例で用いたパラメータからデータをダウンロードした場合，図 6 の様なデータがダウンロードできる．

```
システム！"raspberry" 使う。
システム！"RecordingStorage" 使う。

// --------- 記録の準備 ---------
計測データ＝記録ストレージ！"usr1" "file1.csv"　作る。
計測データ！"http://サーバー URL/api.php"　保存先選択。

// --------- 気圧温度センサの準備 ---------
気圧センサー＝ラズパイ！ LPS25H。

// --------- 気圧温度センサの計測結果表示 ---------
「
    現在の気圧＝気圧センサー！気圧？。
    現在の温度＝気圧センサー！温度？。

    計測データ！（現在の気圧）　（現在の温度）　記録。
    ラズパイ！ 60　待つ。
」！ 60　繰り返す。
```

図 7 気圧センサの値をサーバに記録するサンプル

5.3 ドリトルからの利用方法

本ライブラリには，データ蓄積サーバにデータを送信するための命令が実装されている．データを送信するには以下の手順で命令を実行する．

(1) ライブラリの読み込み
(2) ファイル名とユーザ名を指定する
(3) サーバの URL を指定する
(4) データを送信する

ライブラリの読み込みでは，命令が定義されている RecordingStorage ライブラリを読み込む．

```
システム！"RecordingStorage"　使う。
```

続いてデータ蓄積サーバに保存する際のユーザ名とファイル名を指定する．

```
計測データ＝記録ストレージ！"usr1" "a.csv" 作る。
```

サーバの URL の指定では，サーバの WebAPI が存在する URL を指定する．

```
計測データ！"http://サーバー URL/api.php"
保存先選択。
```

データの送信では，サーバにデータを送信する．

```
計測データ！（気温）（湿度）　記録。
```

以上の手順を元に，気圧センサから読み取った気圧と気温を 1 分ごとに 1 時間の間記録していくサンプルを図 7 に示す．

6. 高校での実践

神奈川県立柏陽高等学校において，本ライブラリを用いた実践授業が合計 10 コマ行われた．この授業は，Raspberry Pi の基本的な使い方と，ドリトルと上記ライブラリ，LED やセンサを用いてデータの計測を行う実習を中心とした授

業であった．

まず初めに「L チカ」と呼ばれる LED を点滅させる簡単な回路をブレッドボード上に組み，この回路を動作させるサンプルプログラムを活用しながらドリトルの基本的な構文とライブラリの使い方の学習を行った．その後，センサやデータ蓄積サーバへのデータ保存などの学習を行った．LED を点滅させるサンプルプログラムからは，「プログラミングを使って，LED を自由に光らせることが出来て，面白かった．「書く」や「後始末」といった命令で LED が光ったり消えたりするなんて最初は信じられなかったが，実際に動くのを見て，すごいなと感心するとともにいろいろな命令があって興味深いと感じた．」や，「第 1 回を休んでいて，使い方がわからなくてはじめは大変だったけど，使い方がわかってからはドリトルと組み合わせることで LED が光ることに感動しました．」との感想から，生徒の興味を惹くのには十分であったと思われる．

しかし，本ライブラリを利用するにあたって Raspberry Pi 上にセットアップする際には OS 自体のアップデートや，ドリトル以外にも追加ソフトウェアのインストールが必要となる．教育機関のネットワークはプロキシサーバやフィルタリング等のセキュリティが強く掛かっているためにこれらのソフトウェアを `apt-get` コマンド等を用いてインストールを行うことが難しい場合があると考えられる．そのため，あらかじめセットアップが完了した OS のディスクイメージの配布や，セットアップの完了した SD カードを貸し出すなどの対応が必要である．

データ蓄積サーバはデータをサーバに保存し，必ず追記を行うため，データを誤って削除する，上書きする，といった些細なミスによるデータ消失が防げる．また，Raspberry Pi から Excel 等がインストールされているコンピュータにデータを移すことが容易に行える，と言った利点があると考える．さらに，ブラウザー上で蓄積したデータをグラフでプレビューできる機能への対応が求められる．この機能に対応することで，データの変化をブラウザでリアルタイムに確認できると考えられる．

7. おわりに

教育用プログラミング言語ドリトルの処理系から，Raspberry Pi の GPIO を扱うことができるライブラリを実装した．ドリトルを利用することで，センサからの値を読み出したり，LED やボタンを使った電子回路を制御するということが容易に行うことが可能になった．

さらに，センサから読みだした値をサーバに蓄積して CSV 形式でダウンロードできるようにすることで，集積したデータを生徒が容易に分析することが可能となった．

参考文献

[1] 教育用プログラミング言語「ドリトル」. `http://dolittle.eplang.jp`
[2] 兼宗進, 中谷多哉子, 御手洗理恵, 福井眞吾, 久野靖: 初中等教育におけるオブジェクト指向プログラミングの実践と評価. 情報処理学会論文誌プログラミング Vol.44, No.13, pp.58–71 (2003).
[3] 紅林秀治, 兼宗進: 制御と計測を取り入れた情報教育の提案. 情報処理学会研究報告. コンピュータと教育研究会報告 Vol.2004, No.100, pp.41–48 (2004).
[4] 佐藤和浩, 紅林秀治, 兼宗進: 小学校におけるプログラミング活用の現状と課題. 情報処理学会研究報告. コンピュータと教育研究会報告 Vol.2005, No.15, pp.57–63 (2005).
[5] Raspberry Pi – Teach, Learn, and Make with Raspberry Pi. `https://www.raspberrypi.org/`
[6] Raspberry Pi Downloads - Software for the Raspberry Pi. `https://www.raspberrypi.org/downloads/`
[7] FrontPage - Raspbian. `http://www.raspbian.org/`
[8] Raspbian Release Notes. `https://downloads.raspberrypi.org/raspbian/release_notes.txt`
[9] Raspberry Pi 2 on sale now at $35. `https://www.raspberrypi.org/blog/raspberry-pi-2-on-sale/`
[10] 文部科学省: 中学校学習指導要領 (2008).
[11] 文部科学省: 高等学校学習指導要領解説情報編 (2010).