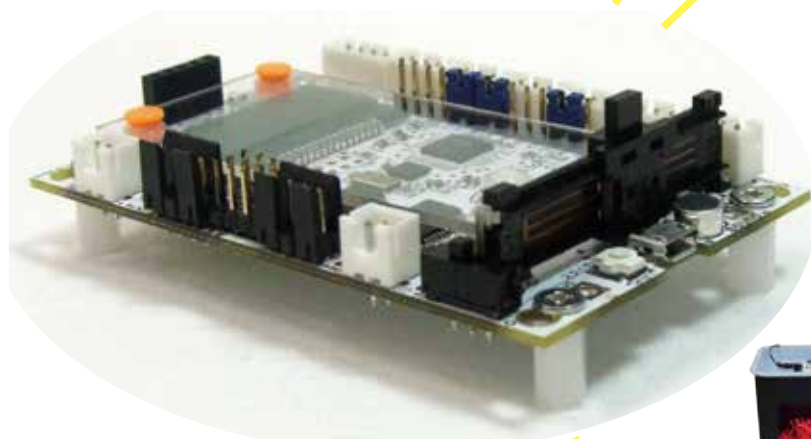
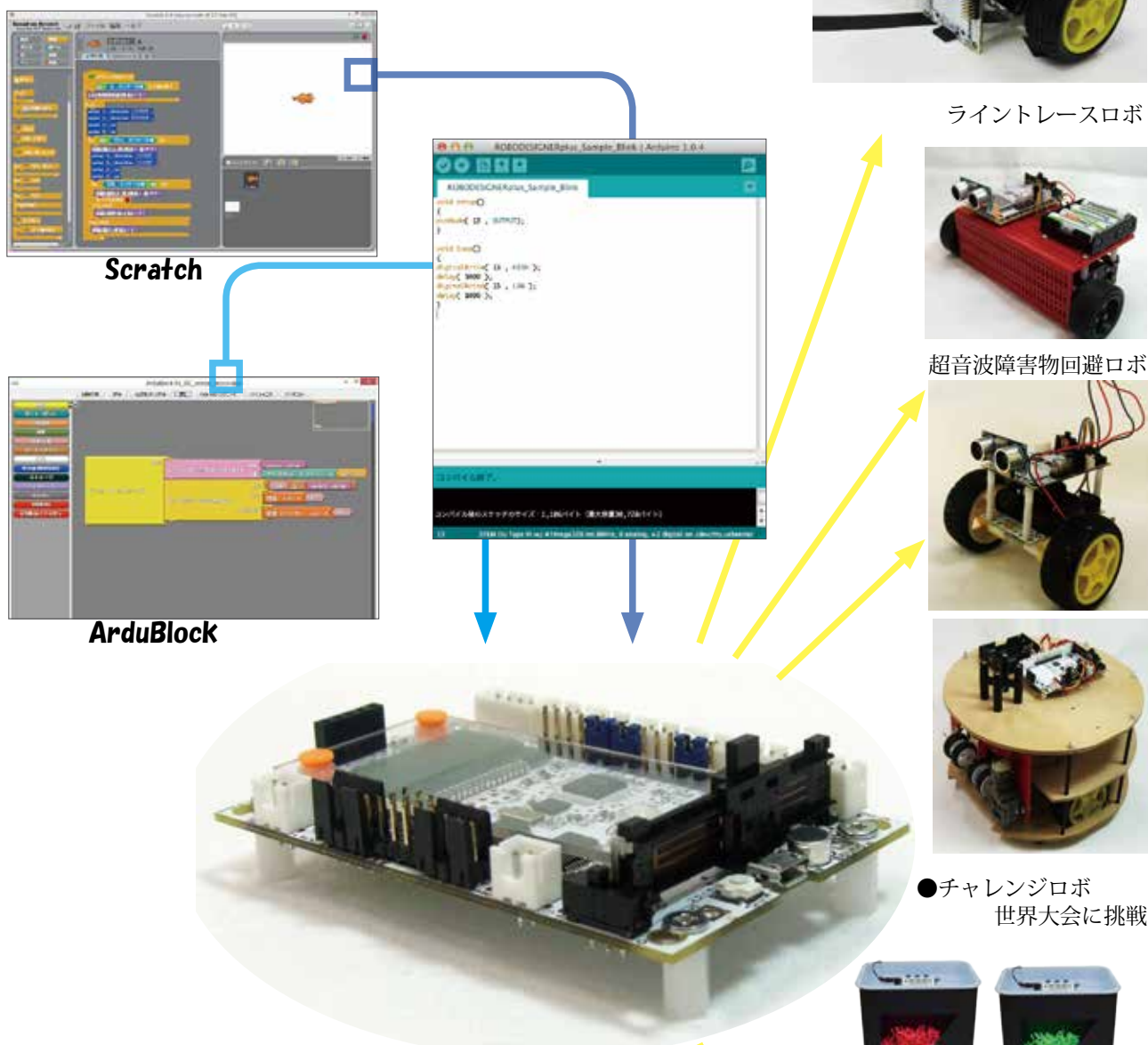


プログラムする、計測する、制御する programmed, measure and control

## 1-2. RoboDesignerPlusシステム相関図

System correlation chart



世界中で最も使われている  
コントローラ  
Arduino との互換性を持ち電子回路の初心者でも始められる手軽さと、これまでの RoboDesigner のセンサーもそのまま使える柔軟性拡張性を合わせ持つ

### RoboDesignerPlus



●自動ドアの仕組み  
ドアの動きを簡単制御



●エコで安全な扇風機  
人を感知して動作、近すぎると羽根停止





●栽培工場  
安全な食物育成を目指し、LED 照明を制御

# 3. コントローラボード

## 3-1. コントローラボード仕様

Table.3.1.1 Specification

Model No.					
		RDC-103 TYPE I	RDC-103 TYPE II	RDC-103 TYPE III	RDC-103 TYPE III +
機能概要	<ul style="list-style-type: none"> <li>Scratchを使って常にパソコンとUSBで接続して使うことを想定したモデル。</li> <li>Scratch上で、2つのモータまで動かすことが可能。</li> </ul>	<ul style="list-style-type: none"> <li>パソコンの無い環境でも、プログラミングからロボットの制御まで取り組むことができるオールインのポータブルモデル。</li> <li>2つまでのモータを使って自律型ロボットを作成可能</li> </ul>	<ul style="list-style-type: none"> <li>最大4個のモータを使用し、PCから独立して動く自律型ロボットを作成可能な本格モデル。</li> </ul>		
MPU	8 Bit AVR ATMEGA32U4 / 内蔵Flash 32kB / RAM 2.5kB / 動作クロック 8MHz				
プログラム環境	<ul style="list-style-type: none"> <li>Pico board(Scratch board)と互換性があり、Scratch1.4を使ってプログラムし動作可能。</li> <li>Arduino互換性があるためArduino-IDEを用いてプログラミングをすることも可能。</li> </ul>	<ul style="list-style-type: none"> <li>Arduino-IDE上で、視覚的プログラミング環境「ArduBlock」を用いて視覚的にプログラムを作成可能。(作成された視覚的プログラムはArduino-IDE上でC++言語に変換され、コンパイル後、コントローラへ転送。)</li> <li>Arduino互換性があるため、Arduino-IDEにてC++で直接記述することも可能。サンプルプログラム、技術資料はWeb上に多数公開されており、参考資料多数。</li> <li>オンボードの液晶表示モジュールとスイッチ、スライダ等のみを使って、パソコンなしに簡易プログラムの作成と実行も可能(サンプルスケッチを提供)。</li> <li>Scratch1.4を使ってプログラムし動作させることも可能。</li> </ul>			
プログラムダウンロード	<ul style="list-style-type: none"> <li>Scratchを使って動作させる場合は、常にPCとUSBで接続して使用。</li> <li>Arduino-IDEを用いる場合はPCのUSBポートからダウンロード可能</li> </ul>	<ul style="list-style-type: none"> <li>PCのUSBポートからダウンロード可能</li> </ul>	<ul style="list-style-type: none"> <li>PCのUSBポートからダウンロード可能</li> </ul>	<ul style="list-style-type: none"> <li>PCのUSBポートからダウンロード可能</li> </ul>	<ul style="list-style-type: none"> <li>PCのUSBポートからダウンロード可能</li> </ul>
DCモータ出力	<ul style="list-style-type: none"> <li>2個のDCブラシモータ制御が可能。 M1,M2</li> <li>正転/反転/停止 及び回転スピードコントロールが可能。</li> </ul>			<ul style="list-style-type: none"> <li>4個のDCブラシモータ制御が可能。 M1,M2,M3,M4</li> <li>正転/反転/停止 及び回転スピードコントロールが可能。</li> </ul>	
サーボ出力 デジタルピンと兼用	<ul style="list-style-type: none"> <li>8個のサーボモータ制御が可能</li> </ul>	<ul style="list-style-type: none"> <li>8個のサーボモータ制御が可能</li> </ul>		<ul style="list-style-type: none"> <li>4個のサーボモータ制御が可能</li> </ul>	
入力ポート	<ul style="list-style-type: none"> <li>アナログ入力コネクタ × 2個</li> </ul>	<ul style="list-style-type: none"> <li>アナログ入力コネクタ × 2個</li> </ul>		<ul style="list-style-type: none"> <li>アナログ入力コネクタ × 6個 (3個はボード上のセンサとジャンパで切替)</li> <li>デジタル入出力コネクタ × 8個、ピン × 6個 (デジタル端子はモータと兼用(モータ4個で12個使用))</li> </ul>	
みの虫クリップ用端子					
通信ポート	<ul style="list-style-type: none"> <li>基板上にUSBコネクタを搭載、プログラムのダウンロードや、取得データのアップロードが可能。</li> <li>I2C,UART通信可能</li> <li>USBコネクタより電源供給、机上の回路実験テストなどでは外部電源接続が不要。</li> </ul>				
搭載機能	<ul style="list-style-type: none"> <li>音センサ、光センサ、加速度センサ+ジャイロ (I2C)、スライダ、白色LEDをボード上に搭載</li> </ul>				
		<ul style="list-style-type: none"> <li>I2C接続各種センサ、超音波センサを追加可能</li> </ul>			
液晶/カバー	オプション	モノクロLCD/透明カバー	オプション	モノクロLCD/透明カバー	
電源	回路/ モータM1、M2	<ul style="list-style-type: none"> <li>USB/V1コネクタから供給 4.5V-6.0V</li> </ul>			<ul style="list-style-type: none"> <li>USB/V1コネクタから供給 4.5V-6.0V</li> </ul>
	モータ M3、M4	<ul style="list-style-type: none"> <li>V2コネクタから供給 モータに合わせた電圧 (M3,M4用3~12V)</li> </ul>			<ul style="list-style-type: none"> <li>V2コネクタから供給 モータに合わせた電圧 (M3,M4用3~12V)</li> </ul>
基板サイズ 重さ	56mm x 88mm 厚さ 25mm ・重さ 約 45.5 グラム				

## 3-2. コントローラボード概要

### 3.2.1. RDC - 103TYPE I

- Scratch を使って 2 つのモータまで動かすことができ、常にパソコンと接続して使用します。※ 1
- LED / 光センサー、音センサー、ジャイロ / 加速度センサ、スライダをボード上に搭載しており、これらを利用して各種制御を行います。
- 外部アナログセンサー 2 個まで接続可能。
- サーボモータ 8 個まで接続可能
- RDC-103TYPE I には、M3、M4、小型液晶モジュールは搭載していません。
- ※ 1 ・ USB 端子からの電源供給でモータ 1 個が動作可能です。モータ 2 個を動かすには、電池を接続して電源を供給してください。

## CONTROL BOARD RDC-103TYPE I OUTLINE

- It's even possible to move 2 motors using Scratch, and always it's connected with a PC and it's used. ※ 1.
- It's equipped with LED / Light sensor-, Sound sensor-and the Acceleration/a Gyro sensor and a Slider on the board, using these, you can control variously.
- It's even possible to connect 2 of outside analogue sensor (A1,A2).
- It's even possible to connect 8 servomotors.
- M3, M4 and a LCD module aren't loaded into RDC-103TYPE I.
- ※ 1) 1 motor can move by power supply from a USB terminal. Please connect a battery and supply me a power supply to move 2 motors.

仕様	The specification	DataSheet URL
マイコン / ATMEGA32U4、発信周波数 8MHz	MCU / ATMEGA32U4 Clock 8MHz	<a href="http://media.digikey.com/pdf/Data%20Sheets/Atmel%20PDFs/ATmega16U4,32U4.pdf">http://media.digikey.com/pdf/Data%20Sheets/Atmel%20PDFs/ATmega16U4,32U4.pdf</a>
加速度センサ / ジャイロ	Acceleration / Gyro sensor MPU-6050	<a href="http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/">http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/</a>
音センサ	Sound Sensor SPI XCM6035P	<a href="http://www.buzzer.com.hk">http://www.buzzer.com.hk</a>
スライダボリューム	SlidePotentiometers Alps RS30H121	<a href="http://www.alps.com/WebObjects/catalog.woa/J/HTML/Potentiometer/SlidePotentiometers/">http://www.alps.com/WebObjects/catalog.woa/J/HTML/Potentiometer/SlidePotentiometers/</a>
明るさセンサー	Light sensor Everlight PT12-21C	<a href="http://www.everlight.com/file/ProductFile/PT12-21C-TR8.pdf">http://www.everlight.com/file/ProductFile/PT12-21C-TR8.pdf</a>

RDC-103 には、いろいろな文字や記号が描かれていますが、大きく分けると、センサコネクター、モータコネクター、電源コネクター、USB コネクターの 4 つです。

### デジタル入出力 Digital in/out

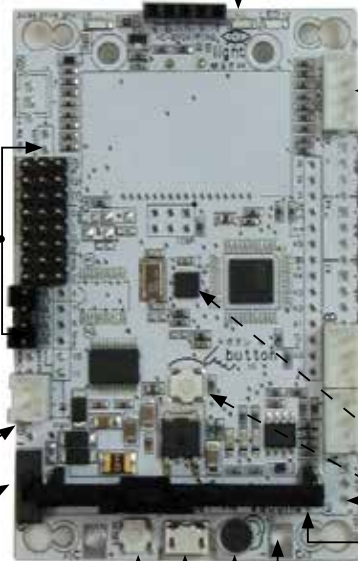
入出力端子を使用したい時はピンで接続します。

ピン番号	記号	解説
13		サーボ / 白色 LED / PWM 出力可能 R/C servo motor / White LED / PWM output
12		サーボ / ボタン R/C servo motor / Button
11		サーボ / 超音波 / 赤外線 LED R/C servo motor / UltraSonic / InfraRed
0		サーボ / ブザー / シリアル RX R/C servo motor / Buzzer / Serial RX
1		サーボ / LCD RS / シリアル TX R/C servo motor / LCD RS / Serial TX
10		サーボ / LCD CS / PWM 出力可能 R/C servo motor / LCD CS / PWM output
6		サーボ / M1 PWM 制御 / PWM 出力可能 R/C servo motor / M1 PWM control / PWM output
5	M1 (0.5A 程度 / 1 個)	サーボ / M1 制御 / PWM 出力可能 R/C servo motor / M1 control / PWM output
4		M1 制御 M1 control
7		M2 制御 M2 control
8	M2 (0.5A 程度 / 1 個)	M2 制御 M2 control
9		M2 PWM 制御 / PWM 出力可能 M2 PWM control / PWM output
<b>電源コネクター</b>		
V1		電源コネクター Power Connector
-		電源スイッチ Power Switch

増設可能  
★超音波センサ Ultrasonic sensor (別売品) 差し込んで使用します。



赤外線 LED  
Infrared Sensor



### 明るさセンサ Light sensor

発光 白色 LED / 受光 フォトトランジスタ

### I<sup>2</sup>C コネクター 3 SCL, 2 SDA

### アナログ入力 Analog input

A1 (A は Analog の A) と A2 の 2 ポートがあります。0 から電源電圧 (3.3V) までの入力電圧を 1024 段階で読み取ります。センサやボリュームなどを接続することができます。また、スケッチで設定を変更するとデジタル入出力ピンとして使うことができます。

ピン番号	記号	解説
A0	A0	音センサ Sound Sensor
A1	A1	アナログ入力コネクター Analog input
A2	A2	アナログ入力コネクター Analog input
A3	A3	みの虫クリップ Signal Clip terminal
A4	A4	明るさセンサ Light Sensor
A5	A5	スライダ Slider

接続コネクター JST connector XH3B

### LED

記号	解説
ON	青色 電源確認 Blue LED
RX	赤色 通信確認 Red LED
TX	緑色 通信確認 Green LED

### 加速度 / ジャイロ / 温度センサ (I2C)

Accelerometer/Gyroscope

ボタン button

スライダ (可変抵抗) Slider resistance

みの虫クリップ用端子 Terminal for clips

(抵抗等を測ります)

音センサ Sound sensor

USB コネクター USB connector

RESET スイッチ

3.2.2. RDC – 103TYPE II

- 2 個の DC モーターを使用し、PC から独立して動く自律型ロボットを作成可能。
- LED / 光センサー、音センサー、加速度 / ジャイロセンサー、スライダをボード上に搭載しており、これらを利用して各種制御を行えます。
- LCD モニターでセンサー値計測が行えます。
- 外部超音波センサー（別売）接続可能（センサソケット利用）I2C
- 外部アナログセンサー 2 個（A1,A2）まで接続可
- 通常はスケッチ（プログラム）に合わせて配線します。
- サーボモータ 8 個まで接続可能

Control board RDC-103TYPE II

- It's possible to make the autonomous robot which becomes independent of a PC using 2 DC motor-and moves.
- It's equipped with LED / Light sensor-, Sound sensor-and the Acceleration/a Gyro sensor and a Slider on the board, using these, you can control variously.
- The sensor value measurement with a LCD monitor can be performed.
- It's possible to connect Ultrasonic Sensor- (I2C sensor socket use), (separate sale part)
- It's even possible to connect 2 of outside analogue sensor (A1,A2).
- Usually wire according to the sketch (program).
- It's even possible to connect 8 servomotors.

仕様	The specification	DataSheet URL
マイコン / ATMEGA32U4、発信周波数 8MHz	MCU / ATMEGA32U4 Clock 8MHz	<a href="http://media.digikay.com/pdf/Data%20Sheets/Atmel%20PDFs/ATmega16U4_32U4.pdf">http://media.digikay.com/pdf/Data%20Sheets/Atmel%20PDFs/ATmega16U4_32U4.pdf</a>
加速度センサ / ジャイロ	Acceleration / Gyro sensor MPU-6050	<a href="http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/">http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/</a>
音センサ	Sound Sensor SPI XCM6035P	<a href="http://www.buzzer.com.hk">http://www.buzzer.com.hk</a>
スライダボリューム	SlidePotentiometers Alps RS30H121	<a href="http://www.alps.com/WebObjects/catalog_woa/J/HTML/Potentiometer/SlidePotentiometers/">http://www.alps.com/WebObjects/catalog_woa/J/HTML/Potentiometer/SlidePotentiometers/</a>
明るさセンサー	Light sensor Everlight PT12-21C	<a href="http://www.everlight.com/file/ProductFile/PT12-21C-TR8.pdf">http://www.everlight.com/file/ProductFile/PT12-21C-TR8.pdf</a>

- RDC-103 には、いろいろな文字や記号が描かれていますが、大きく分けると、センサコネクタ、モータコネクタ、電源コネクタ、USB コネクタの 4 つです。

デジタル入出力 Digital in/out

入出力端子を使用したい時はピンで接続します。

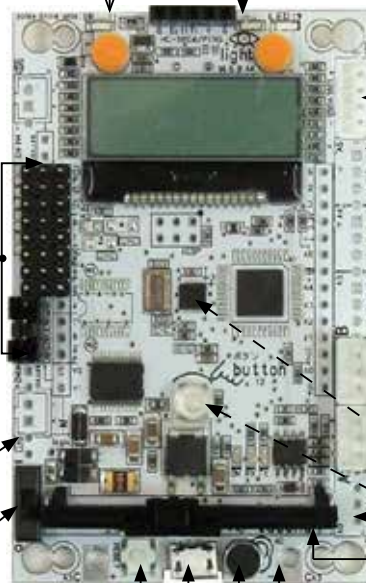
ピン番号	記号	解説
13		サーボ / 白色 LED / PWM 出力可能 R/C servo motor / White LED / PWM output
12		サーボ / ボタン R/C servo motor / Button
11		サーボ / 超音波 / 赤外線 LED R/C servo motor / UltraSonic / InfraRed
0		サーボ / ブザー / シリアル RX R/C servo motor / Buzzer / Serial RX
1		サーボ / LCD RS / シリアル TX R/C servo motor / LCD RS / Serial TX
10		サーボ / LCD CS / PWM 出力可能 R/C servo motor / LCD CS / PWM output
6	M1 (0.5A 程度 1個)	サーボ / M1 PWM 制御 / PWM 出力可能 R/C servo motor / M1 PWM control / PWM output
5		サーボ / M1 制御 / PWM 出力可能 R/C servo motor / M1 control / PWM output
4		M1 制御 M1 control
7	M2 (0.5A 程度 1個)	M2 制御 M2 control
8		M2 制御 M2 control
9		M2 PWM 制御 / PWM 出力可能 M2 PWM control / PWM output
<b>電源コネクタ</b>		
	V1	電源コネクタ Power Connector
	—	電源スイッチ Power Swith

RESET スイッチ

増設可能  
★超音波センサ Ultrasonic sensor (別売品) 差し込んで使用します。



赤外線 LED  
Infrared Sensor



明るさセンサ Light sensor

発光 白色 LED / 受光 フォトトランジスタ

I<sup>2</sup>C コネクタ 3 SCL, 2 SDA

アナログ入力 Analog input

A1 (A は Analog の A) と A2 の 2 ポートがあります。0 から電源電圧 (3.3V) までの入力電圧を 1024 段階で読み取ります。センサやボリュームなどを接続することができます。また、スケッチで設定を変更するとデジタル入出力ピンとして使うことができます。

ピン番号	記号	解説
A0	A0	音センサ Sound Sensor
A1	A1	アナログ入力コネクタ Analog input
A2	A2	アナログ入力コネクタ Analog input
A3	A3	みの虫クリップ Signal Clip terminal
A4	A4	明るさセンサ Light Sensor
A5	A5	スライダ Slider

接続コネクタ JST connector XH3B

LED

記号	解説
ON	青色 電源確認 Blue LED
RX	赤色 通信確認 Red LED
TX	緑色 通信確認 Green LED

加速度 / ジャイロ / 温度センサ (I2C)  
Accelerometer/Gyroscope

ボタン button

スライダ (可変抵抗) Slider resistance

みの虫クリップ用端子 (抵抗等を測ります)  
Terminal for clips (Resistance etc. are measured)

音センサ Sound sensor

USB コネクタ USB connector

### 3.2.3. RDC – 103TYPE III

- ・ 4 個の DC モーターを使用し、PC から独立して動く自律型ロボットを作成可能。
- ・ LED / 光センサー、音センサー、加速度 / ジャイロセンサー、スライダをボード上に搭載しており、これらを利用して各種制御を行えます。
- ・ 外部超音波センサー 増設可能・ I2C
- ・ 外部アナログセンサー 6 個まで接続可
- ・ 通常はスケッチ (プログラム) に合わせて配線します。
- ・ サーボモータ 4 個まで接続可能
- ・ 2 電源式 (M3,M4 モータ電源 最大 12V まで使用可能)
- ・ LCD モニター必要な場合は、RDC-103TYPE III + をご利用ください。

### Control board RDC-103TYPE III

- ・ It's possible to make the autonomous robot which becomes independent of a PC using 4DC motor-and moves.
- ・ It's equipped with LED / Light sensor-, Sound sensor-and the Acceleration / Gyro sensor and a Slider on the board, using these, you can control variously.
- ・ It's possible to connect Ultrasonic Sensor- (sensor socket use), I2C----(separate sale part)
- ・ It's even possible to connect 6 of outside analogue sensor (A0,A1,A2,A3,A4,A5).
- ・ It's even possible to connect 4 servomotors.
- ・ 2 power supply system (Even at most 12 V of motor power supply is practicable. M3,M4)

仕様	The specification	DataSheet URL
マイコン / ATMEGA32U4、発信周波数 8MHz	MCU / ATMEGA32U4 Clock 8MHz	<a href="http://media.digikey.com/pdf/Data%20Sheets/Atmel%20PDFs/ATmega16U4,32U4.pdf">http://media.digikey.com/pdf/Data%20Sheets/Atmel%20PDFs/ATmega16U4,32U4.pdf</a>
加速度センサ / ジャイロ	Acceleration / Gyro sensor MPU-6050	<a href="http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/">http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/</a>
音センサ	Sound Sensor SPI XCM6035P	<a href="http://www.buzzer.com.hk">http://www.buzzer.com.hk</a>
スライダボリューム	SlidePotentiometers Alps RS30H121	<a href="http://www.alps.com/WebObjects/catalog.woa/J/HTML/Potentiometer/SlidePotentiometers/">http://www.alps.com/WebObjects/catalog.woa/J/HTML/Potentiometer/SlidePotentiometers/</a>
明るさセンサー	Light sensor Everlight PT12-21C	<a href="http://www.everlight.com/file/ProductFile/PT12-21C-TR8.pdf">http://www.everlight.com/file/ProductFile/PT12-21C-TR8.pdf</a>

・ RDC-103 には、いろいろな文字や記号が描かれていますが、大きく分けると、センサコネクタ、モータコネクタ、電源コネクタ、USB コネクタの 4 つです。

#### デジタル入出力 Digital in/out.

入出力端子を使用したい時はピンで接続します。

ピン番号	記号	解説
13	M4 (0.5A 程度 / 1 個)	サーボ / 白色 LED / PWM 出力可能 R/C servo motor / White LED / PWM output
12		サーボ / ボタン R/C servo motor / Button
11	M3 (0.5A 程度 / 1 個)	サーボ / 超音波 / 赤外線 LED R/C servo motor / UltraSonic / InfraRed
0		サーボ / ブザー / シリアル RX R/C servo motor / Buzzer / Serial RX
1		サーボ / LCD RS / シリアル TX R/C servo motor / LCD RS / Serial TX
10		サーボ / LCD CS / PWM 出力可能 R/C servo motor / LCD CS / PWM output
6	M1 (0.5A 程度 / 1 個)	サーボ / M1 PWM 制御 / PWM 出力可能 R/C servo motor / M1 PWM control / PWM output
5		サーボ / M1 制御 / PWM 出力可能 R/C servo motor / M1 PWM control / PWM output
4		M1 制御 M1 control
7		M2 制御 M2 control
8	M2 (0.5A 程度 / 1 個)	M2 制御 M2 control
9		M2 PWM 制御 / PWM 出力可能 M2 PWM control / PWM output
<b>電源コネクタ</b>		
V2	M 3 , M 4 用電源供給端子 Power Connector for M3,M4	
V1	電源コネクタ Power Connector	
—	電源スイッチ Power Swith	

RESET スイッチ

増設可能  
★超音波センサ Ultrasonic sensor (別売品) 差し込んで使用します。



#### 明るさセンサ Light sensor

発光 白色 LED / 受光 フォトトランジスタ

#### I<sup>2</sup>C コネクタ 3 SCL, 2 SDA

#### アナログ入力 Analog input

A0 (A は Analog の A) から A5 までの計 6 ポートがあります。0 から電源電圧 (3.3V) までの入力電圧を 1024 段階で読み取ります。センサやボリュームなどを接続することができます。また、スケッチで設定を変更するとデジタル入力ピンとして使うことができます。

ピン番号	記号	解説
A0	A0	音センサ / アナログ入力コネクタ Sound Sensor / Analog input
A1	A1	アナログ入力コネクタ Analog input
A2	A2	アナログ入力コネクタ Analog input
A3	A3	みの虫クリップ Signal Clip terminal
A4	A4	明るさセンサ / アナログ入力コネクタ Light Sensor / Analog input
A5	A5	スライダ / アナログ入力コネクタ Slider / Analog input

接続コネクタ JST connector XH3B

#### LED

記号	解説
ON	青色 電源確認 Blue LED
RX	赤色 通信確認 Red LED
TX	緑色 通信確認 Green LED

#### 加速度 / ジャイロ / 温度センサ (I2C) Accelerometer/Gyroscope

#### ボタン button

#### スライダ (可変抵抗) Slider resistance

#### みの虫クリップ用端子 Terminal for clips (抵抗等を測ります)

#### 音センサ Sound sensor

#### USB コネクタ USB connector

### 3.2.4. RDC - 103TYPE III + (LCD 搭載)

- 4 個の DC モーターを使用し、PC から独立して動く自律型ロボットを作成可能。
- LED / 光センサー、音センサー、加速度 / ジャイロセンサー、スライダをボード上に搭載しており、これらを利用して各種制御を行えます。
- LCD モニターでセンサー値計測が行えます。
- 外部超音波センサー 増設可能・I2C
- 外部アナログセンサー 6 個まで接続可
- 通常はスケッチ(プログラム)に合わせて配線します。
- サーボモータ 4 個まで接続可能
- 2 電源式 (M3,M4 モータ電源 最大 12V まで使用可能)

### Control board RDC-103TYPE III +

- It's possible to make the autonomous robot which becomes independent of a PC using 4DC motor-and moves.
- It's equipped with LED / Light sensor, Sound sensor-and the Acceleration / Gyro sensor and a Slider on the board, using these, you can control variously.
- The sensor value measurement with a LCD monitor can be performed.
- It's possible to connect Ultrasonic Sensor- (sensor socket use), I2C----(separate sale part)
- It's even possible to connect 6 of outside analogue sensor (AO,A1,A2,A3,A4,A5).
- It's even possible to connect 4 servomotors.
- 2 power supply system (Even at most 12 V of motor power supply is practicable. M3,M4)

仕様	The specification	DataSheet URL
マイコン / ATMEGA32U4、発信周波数 8MHz	MCU / ATMEGA32U4 Clock 8MHz	<a href="http://media.digikey.com/pdf/Data%20Sheets/Atmel%20PDFs/ATmega16U4,32U4.pdf">http://media.digikey.com/pdf/Data%20Sheets/Atmel%20PDFs/ATmega16U4,32U4.pdf</a>
加速度センサ / ジャイロ	Acceleration / Gyro sensor MPU-6050	<a href="http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/">http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/</a>
音センサ	Sound Sensor SPI XCM6035P	<a href="http://www.buzzer.com.hk">http://www.buzzer.com.hk</a>
スライダポリューム	SlidePotentiometers Alps RS30H121	<a href="http://www.alps.com/WebObjects/catalog_woa/J/HTML/Potentiometer/SlidePotentiometers/">http://www.alps.com/WebObjects/catalog_woa/J/HTML/Potentiometer/SlidePotentiometers/</a>
明るさセンサー	Light sensor Everlight PT12-21C	<a href="http://www.everlight.com/file/ProductFile/PT12-21C-TR8.pdf">http://www.everlight.com/file/ProductFile/PT12-21C-TR8.pdf</a>

• RDC-103 には、いろいろな文字や記号が描かれていますが、大きく分けると、センサコネクタ、モータコネクタ、電源コネクタ、USB コネクタの 4 つです。

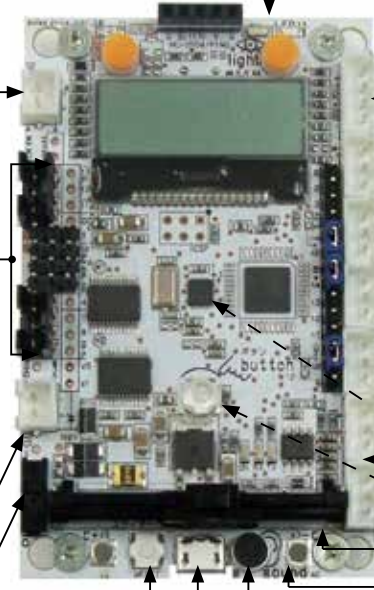
#### デジタル入出力 Digital in/out

入出力端子を使用したい時はピンで接続します。

ピン番号	記号	解説
13	M4 (0.5A 程度 / 1 個)	サーボ / 白色 LED / PWM 出力可能 R/C servo motor / White LED / PWM output
12		サーボ / ボタン R/C servo motor / Button
11	M3 (0.5A 程度 / 1 個)	サーボ / 超音波 / 赤外線 LED R/C servo motor / UltraSonic / InfraRed
0		サーボ / ブザー / シリアル RX R/C servo motor / Buzzer / Serial RX
1		サーボ / LCD RS / シリアル TX R/C servo motor / LCD RS / Serial TX
10		サーボ / LCD CS / PWM 出力可能 R/C servo motor / LCD CS / PWM output
6	M1 (0.5A 程度 / 1 個)	サーボ / M1 PWM 制御 / PWM 出力可能 R/C servo motor / M1 PWM control / PWM output
5		サーボ / M1 制御 / PWM 出力可能 R/C servo motor / M1 PWM control / PWM output
4		M1 制御 M1 control
7	M2 (0.5A 程度 / 1 個)	M2 制御 M2 control
8		M2 制御 M2 control
9		M2 PWM 制御 / PWM 出力可能 M2 PWM control / PWM output
<b>電源コネクタ</b>		
V2		M 3, M 4 用電源供給端子 Power Connector for M3,M4
V1		電源コネクタ Power Connector
—		電源スイッチ Power Switch

RESET スイッチ

増設可能  
★超音波センサ Ultrasonic sensor (別売品) 差し込んで使用します。



- 明るさセンサ Light sensor 発光 LED 白色 / 受光 LED
- I<sup>2</sup>C コネクタ 3 SCL, 2 SDA
- アナログ入力 Analog input

A0 (A は Analog の A) から A5 までの計 6 ポートがあります。0 から電源電圧 (3.3V) までの入力電圧を 1024 段階で読み取ります。センサやポリュームなどを接続することができます。また、スケッチで設定を変更するとデジタル入出力ピンとして使うことができます。

ピン番号	記号	解説
A0	A0	音センサ / アナログ入力コネクタ Sound Sensor / Analog input
A1	A1	アナログ入力コネクタ Analog input
A2	A2	アナログ入力コネクタ Analog input
A3	A3	みの虫クリップ Signal Clip terminal
A4	A4	明るさセンサ / アナログ入力コネクタ Light Sensor / Analog input
A5	A5	スライダ / アナログ入力コネクタ Slider / Analog input

接続コネクタ JST connector XH3B

#### LED

記号	解説
ON	青色 電源確認 Blue LED
RX	赤色 通信確認 Red LED
TX	緑色 通信確認 Green LED

- 加速度 / ジャイロ / 温度センサ (I2C) Accelerometer/Gyroscope
- ボタン button
- スライダ (可変抵抗) Slider resistance
- みの虫クリップ用端子 Terminal for clips (抵抗等を測ります)
- 音センサ Sound sensor
- USB コネクタ USB connector

### 3-3. 搭載機能概要(搭載センサの使い方)

RDCは、Arduino/SensorBoard互換コントローラです。以下、搭載センサの使い方例です。

#### 3.3.1. 音センサ



音センサの反応に応じてモータの回転方向を変えるプログラムです。条件分岐のしきい値を変更すると、反応する音の大きさが変わります。

使用コネクタ A0/ Motor1/ Motor2

変数

A0 搭載音センサの出力値を格納します。  
 ・センサ/モータは、車体後ろから見て小さい番号が左側になるように配置するように作成してあります。  
 ※作成したプログラムは、「名前をつけて保存」します。  
 作成完了後、「Arduinoへアップロード」して、コントローラへ書き込みます。

動作手順

- ・周囲の音がしきい値 (200) ≤ なら前進します。
- ・でなければ後進 (バック) します。

ソースコードは、PC/MyDocuments/Arduino/へ配置したサンプルフォルダ [ArduBlock Examples] に、ファイル名【01\_02\_sensor\_motor.abp】で格納されています。

#### Outline of functions

(Equipped sensor how to use)

RDC is Arduino/SensorBoard compatible controller, hereinafter a how to use example of a equipped sensor.

#### Sound sensor

The program example which reacts to the sound around RDC.

Use connector A0/ Motor1/ Motor2  
 Variable A0 An output value of a loading sound sensor is stocked.

\* A sensor/a motor is made as it is seen from the body rear, and it may be arranged as the small number will be the left side.

\*Name the program you made and save in a folder.

After making completion, it "is uploaded to Arduino", it's done and it's written in a controller.

Source code is stocked in the sample folder arranged to PC/MyDocuments/Arduino/ [ArduBlock Examples] by the file name [01\_02\_sensor\_motor.abp].

#### 3.3.2. 明るさセンサ



明るさセンサの値に応じてブザーを鳴らしたり止めたりします。条件分岐のしきい値を変えると反応する明るさが変わります。

使用コネクタ 明るさセンサ (A4) / buzzer (D0)

変数

A4 搭載光センサの出力値を格納します。  
 ※ RDC-103 のブザーは D0 に接続されています。ブロックで D0 が設定できない場合は、コンパイル後に C ソースコードの下記の関数の引数を以下のように「0」に修正してください。

```
tone(0, 440);
```

※作成したプログラムは、「名前をつけて保存」します。  
 作成完了後、「Arduinoへアップロード」して、コントローラへ書き込みます。

動作手順

- ・周囲の明るさがしきい値 (200) ≤ なら ブザーが鳴ります。
- ・でなければ ブザーが止まります。

ソースコードは、PC/MyDocuments/Arduino/へ配置したサンプルフォルダ [ArduBlock Examples] に、ファイル名【01\_03\_sensor\_buzzer.abp】で格納されています。

#### Light sensor

The program example which reacts to the brightness around RDC using a equipped light sensor and moves.

Use connector Brightness sensor (A4) / buzzer (D8)

Variable  
 An output value of a light sensor with A4 is stocked.

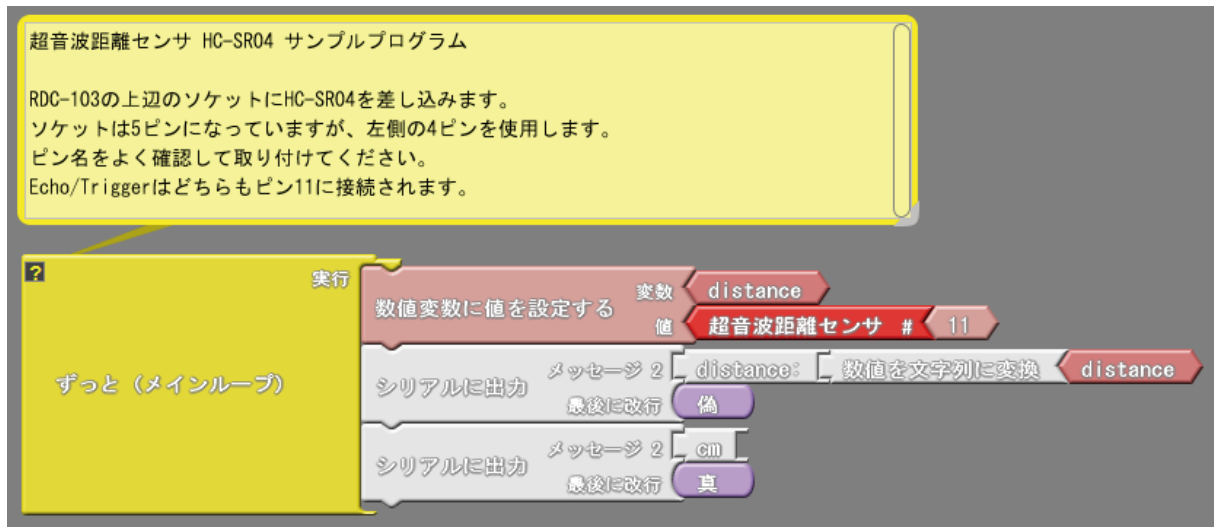
\*Name the program you made and save in a folder.

After making completion, it "is uploaded to Arduino", it's done and it's written in a controller.

Source code is stocked in the sample folder arranged to PC/MyDocuments/Arduino/ [ArduBlock Examples] by the file name [01\_03\_sensor\_buzzer.abp].

### 3.3.3. 超音波距離センサ

### Ultrasonic distance sensor.



超音波距離センサを使用して、対象物との距離を測るプログラム例です。超音波距離センサ HCSR04 の接続が必要です。

使用コネクタ            超音波センサ (D11) /  
変数  
A4                      超音波距離センサの出力値をリアルに出力します。

#### 動作手順

- ・超音波センサの出力値をリアルに出力します。

ソースコードは、PC/MyDocuments/Arduino/へ配置したサンプルフォルダ [ArduBlock Examples] に、ファイル名【01\_04\_ultrasonic\_HCSR04.abp】で格納されています。

It's the program example which measures the distance with the target thing using an ultrasonic sensor.

Use connector Ultrasonic sensor (D11)/ Variable  
An output value of a A4 ultrasonic sensor is output realistically.  
\*We make a made program name and preserve it ".  
After making completion, it "is uploaded to Arduino", it's done and it's written in a controller.

### 3.3.4. 加速度/ジャイロ/温度センサ



- 1). サンプルのソースコードは、PC/MyDocuments/Arduino/へ配置したサンプルフォルダ [RDS-X Examples\_C] に、ファイル名【MPU6050\_test.ino】で格納されています。ソースはC言語で記述されています。
- 2). Arduino を起動後、[ファイル] → [スケッチブック] → [RDS-X Examples\_C] → [MPU6050\_test.ino] で開くと確認できます。

Source code is stocked in the sample folder arranged to PC/MyDocuments/Arduino/ [ArduBlock Examples] by the file name [01\_04\_ultrasonic\_HCSR04.abp].

#### The acceleration/gyro/temperature sensor

- 1). Source code of a sample is stocked in the sample folder arranged to PC/MyDocuments/Arduino/ [RDS-X Examples\_C] by the file name [MPU6050\_test.ino]. A source is being described by a C language.
- 2). After starting Arudino, [file].-> [sketchbook] can confirm that-> [RDS-X Examples\_C]-> opens by [MPU6050\_test.ino].

※以下は、サンプルコードの内容です。

[\[Program source code.\]](#)

薄い色の文字はコメント文です。

```
// MPU-6050 Short Example Sketch
// By Arduino User JohnChi
// August 17, 2014
// Public Domain
#include<Wire.h>
const int MPU=0x68; // I2C address of the MPU-6050
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
void setup() {
```

\* Below is the contents of a sample source code.

The character of the light color is comment.

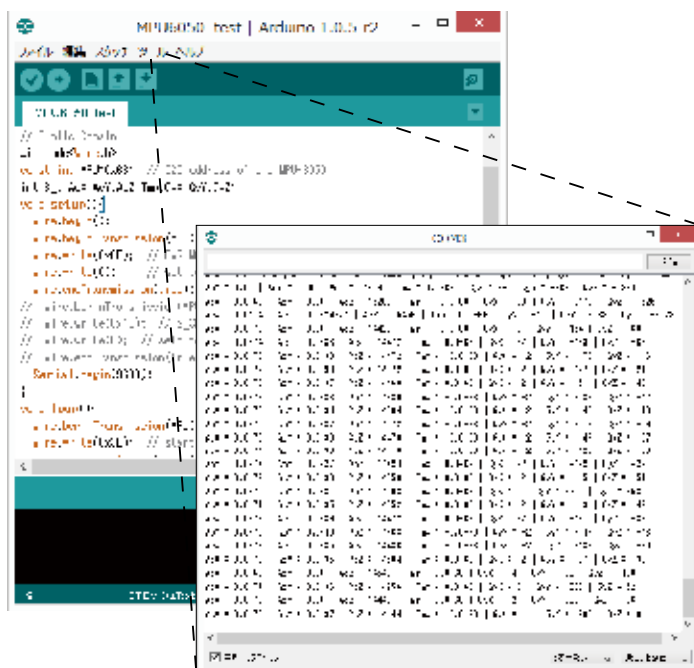


```

Wire.begin();
Wire.beginTransmission(MPU);
Wire.write(0x6B); // PWR_MGMT_1 register
Wire.write(0); // set to zero (wakes up the MPU-6050)
Wire.endTransmission(true);
// Wire.beginTransmission(MPU);
// Wire.write(0x1B); //FS_SEL register selects the full scale range of the gyroscope outputs
// Wire.write(0); // set to +250deg/s 00011000 +-2000deg/s
// Wire.endTransmission(true);
Serial.begin(9600);
}
void loop() {
Wire.beginTransmission(MPU);
Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
Wire.endTransmission(false);
Wire.requestFrom(MPU, 14, true); // request a total of 14 registers
AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
Tmp=Wire.read()<<8|Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
Serial.print("AcX = "); Serial.print(AcX / 1638.4);
Serial.print(" | AcY = "); Serial.print(AcY*0.0054);
Serial.print(" | AcZ = "); Serial.print(AcZ);
Serial.print(" | Tmp = "); Serial.print(Tmp/340.00+36.53); //equation for temperature in
degrees C from datasheet
Serial.print(" | GyX = "); Serial.print(GyX / 131);
Serial.print(" | GyY = "); Serial.print(GyY);
Serial.print(" | GyZ = "); Serial.println(GyZ);
delay(333);
}

```

### 3). センサーデータ計測



1. 【MPU6050\_test.ino】を書き込んだマイコンボードのプログラムが実行されている状態の時に、Arduinoの[シリアルモニター]を使ってセンサの値を調べることができます。(シリアルモニターできるようにsampleプログラムを作成しています)
2. Arduinoの[ツール]→[シリアルモニター]をクリックするとシリアルモニター画面が立ち上がり、リアルタイムでセンサ値が表示されます。
3. シリアルモニターでセンサ値を確認しながらデータ収集を行います。(USBケーブルは接続のまま)
4. USBケーブル接続を外して、データ計測を停止します。シリアルモニターウィンドウの計測値表示が停止しますので、確認が容易になります。
5. シリアルモニターから表計算ソフトなどにコピー (Ctrl+C)、ペースト (Ctrl+V) して数値をグラフ化処理すると分かり易くなります。
6. USBケーブルを抜いて、ロギングを停止してからコピーしてください。

測定データの、1例です。

AcX = 0.0-75 | AcY = 0.0-37 | AcZ = 14572 | Tmp = -0.0-68 | GyX = -2 | GyY = -119 | GyZ = -59  
 加速度データ X軸 | Y軸 | Z軸 | 温度データ | ジャイロデータ | X軸 | Y軸 | Z軸

### 3.3.5. LCD センサーテスター

1. RDC-103Type II / III + では、製品出荷時検査プログラム LCD センサーテスターを入れており、使用時に電源を入れ動作させると、下図のように搭載センサの計測センサ値が LCD に表示されます。
2. 写真の表示値は、コントローラボードの搭載センサが計測した値です。  

Sound	= 0
Light	= 652
Slider	= 153
Button	= 0
3. コントロールボード周辺の明るさ、周辺の音、スライダーを変化させると、表示センサ値も変化しますので、実験確認ください。
4. ご自分のプログラムや、サンプルプログラムなどをコントローラへアップロードされると、上書きされて、出荷時の検査プログラムは、消されます。
5. LCD にセンサー情報を表示して使用したい時は、下記の LCD モニターサンプルコードを参考してください。



#### 6. [参考] LCD\_テスター (製品検査時プログラム)

[Program source code.]

```
#include "U8glib.h"
#include <STEMDu.h>

U8GLIB_AQM1248A_2X u8g(10,1); // Hw SPI
CS=10, AO=1

/* Initialize STEM Du board */
STEMDu robot = STEMDu();

String lineBufferString;
char lineBuffer[16];

int soundValue; // Sound sensor
int lightValue; // Light sensor
int sliderValue; // Slider sensor
int pushValue; // Push button

void draw(void) {
  // graphic commands to redraw the complete screen
  should be placed here
  u8g.setFont(u8g_font_unifont);
  //u8g.setFont(u8g_font_osb21);
  lineBufferString = String("");
  lineBufferString += "Sound = ";
  lineBufferString += soundValue;
  lineBufferString.
  toCharArray(lineBuffer,16);
  u8g.drawStr(0, 11, lineBuffer);
  lineBufferString = String("");
  lineBufferString += "Light = ";
  lineBufferString += lightValue;
  lineBufferString.
  toCharArray(lineBuffer,16);
```

```
u8g.drawStr(0, 23, lineBuffer);
lineBufferString = String("");
lineBufferString += "Slider = ";
lineBufferString += sliderValue;
lineBufferString.
toCharArray(lineBuffer,16);
u8g.drawStr(0, 35, lineBuffer);
lineBufferString = String("");
lineBufferString += "Button = ";
lineBufferString += pushValue;
lineBufferString.
toCharArray(lineBuffer,16);
u8g.drawStr(0, 47, lineBuffer);
}

void setup(void) {
  // flip screen, if required
  //u8g.setRot180();

  // set SPI backup if required
  //u8g.setHardwareBackup(u8g_backup_avr_spi);
}

void loop(void) {
  // read the input on analog port
  soundValue = robot.readSound(); //
  Sound sensor
  lightValue = robot.readLight(); // Light
  sensor
  sliderValue = robot.readSlider(); //
  Slider sensor
  pushValue = robot.readPush(); // Push
  button

  // picture loop
  u8g.firstPage();
  do {
    draw();
  } while( u8g.nextPage() );

  // rebuild the picture after some delay
  delay(50);
}
```

※以上は、サンプルコードの内容です。  
 薄い色の文字はコメント文です。  
 ※ソースコードサンプルは、PC/MyDocuments/Arduino/へ配置したサンプルフォルダ [スケッチの例] に、ファイル名 [STEM-Du] → [TYPE II] → 【SPIGLCD\_SensorTest】で格納されています。  
 ソースはC言語で記述されています。  
 ※ Arduino を起動後、[ファイル] → [スケッチの例] → [STEM-Du] → [TYPE II] で開くと確認できます。

サンプルコードを呼出し後、3行目の  
 U8GLIB\_AQM1248A\_2X u8g(10,1); // Hw SPI  
 CS=10, AO=~~11~~を、  
 U8GLIB\_AQM1248A\_2X u8g(10,1); // Hw SPI CS=10,  
 AO=1  
 と2か所の11を1に変更してください。

\* Program source code of [SensorCheker] is stocked in the sample folder arranged to PC/MyDocuments/Arduino/RDS-X\_Example\_C by the file name [SPIGLCD\_Tester].

### 3.3.6. SensorValues Checker



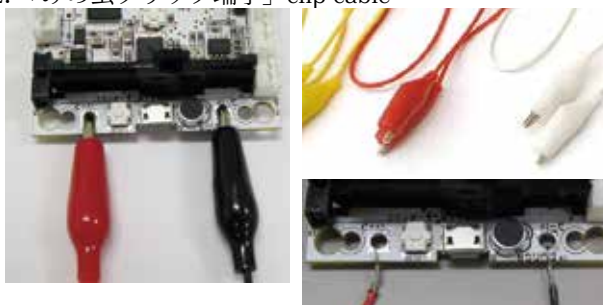
1. RDC-103Type II / III+では、アナログ入力のセンサ値を、LCD モニターに表示することができます。コントローラボード搭載センサ、もしくは、A0～A5 に接続している外部センサーの計測値を LCD モニター表示できますので、ロボットを動作させる環境でのセンサー値をその場で計測可能です。
  2. 作成 / 調整するプログラムの「しきい値」調整にお役立てください。
- ※. 「SensorValues」のプログラムソースコードは、PC/MyDocuments/Arduino/RDS-X\_Example\_C へ配置したサンプルフォルダに、ファイル名【SPIGLCD\_SensorTest\_Connector】で格納されています。

1. Analog sensor data is shown to a LCD monitor in RDC-103Type II / III+ .
  - A LCD monitor indicates measured data of a sensor with a controller board or connected outside sensor.
  - It's possible to measure a sensor data at a movement place.
2. Please use it for making/"threshold value" coordination of an adjusted program.
  - \* Program source code of [SensorValues] is stocked in the sample folder arranged to PC/MyDocuments/Arduino/RDS-X\_Example\_C by the file name [SPIGLCD\_SensorTest\_Connector].

### 3.3.7. BatteryChecker — ResistanceChecker



1. RDC-103Type II / III+ は、基板下部の「みの虫クリップ端子」を使って、乾電池のチェックや、抵抗などの計測ができます。(電池計測時に、電池をショートさせないように注意すること)  
計測可能範囲  乾電池：1 本  抵抗値：1 k～100k オーム
2. 「みの虫クリップ端子」clip cable



みの虫クリップ例

- 「BatteryChecker」のプログラムソースコードは、PC/MyDocuments/Arduino/RDS-X\_Example\_C へ配置したサンプルフォルダに、ファイル名【SPIGLCD\_Tester】で格納されています。

1. RDC-103Type II / III+ can measure a check of a dry battery and resistance using "clip terminal" for the substrate lower part. (Be careful so as not to make a battery short-circuit at the time of battery measurement.)  
Measurement possible area  
 Dry battery: 1pcs  
 Resistance value :1k-100k ohm
2. みの虫クリップコードは必要に応じて準備ください。  
\* You prepare clip cable as the need arises, please.

- コードの絶縁被覆をはがしてクリップ端子の穴に銅線を通し結びつけることでも使用できます。  
\* It can be used even to peel off insulation covering of a cable and tie to a hole of a clip terminal through a copper wire.

- \* Program source code of [BatteryChecker] is stocked in the sample folder arranged to PC/MyDocuments/Arduino/RDS-X\_Example\_C by the file name [SPIGLCD\_Tester].

### 3.3.8. 使用頻度が多い Example Code

実験 1. : Analog Read Serial

- ・アナログ入力をチェックする時に、何かと便利なサンプルコードです。調べたい端子を指定して、入力電圧の変化をシリアルモニターで確認できます。

PC/MyDocuments/Arduino/へ配置したサンプルフォルダ[スケッチの例][01.Basics]に、ファイル名【AnalogReadSerial】で格納されています。C言語で記述されています。

※ Arduino を 起 動 後、[フ ァ イ ル]→[ス ケ ッ チ の 例]→[01.Basics]→[AnalogReadSerial]で開くと確認できます。

```

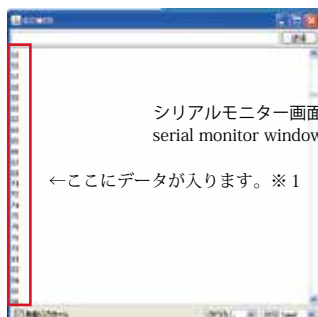
/*
 AnalogReadSerial
 Reads an analog input on pin 0, prints the result
 to the serial monitor.
 Attach the center pin of a potentiometer to pin
 A0, and the outside pins to +5V and ground.

 This example code is in the public domain.
 */

// the setup routine runs once when you press
reset:
void setup() {
 // initialize serial communication at 9600 bits
per second:
 Serial.begin(9600);
}

// the loop routine runs over and over again
forever:
void loop() {
 // read the input on analog pin 0:
 int sensorValue = analogRead(A0); // (A4) に書き変更
 // print out the value you read: 入力ポート端子番号
 Serial.println(sensorValue);
 delay(1); // delay in between reads for
stability
}

```



コントローラ搭載の「明るさセンサー」を調べたい時は、入力端子番号を (A4) に書き変更して、アップロードします。プログラム実行後に、Arduino-IDE の [ツール]→[シリアルモニター]をクリックすると、モニターウィンドウが出現し、指定した端子に入力している信号をモニター表示します。

### ExampleCode with a lot of use frequencies

#### Experimental 1 ; Analog Read Serial

\* When checking analog input, it's a convenient sample code. You can designate the terminal I'd like to check and confirm the change in input voltage by a serial monitor.

It's stocked in the sample folder arranged to PC/MyDocuments/Arduino/ [example of a sketch] [01.Basics] by the file name [AnalogReadSerial]. It's being described by a C language.

※ After starting Arudino, [file],-> can confirm that-> [Sample Code of Sketch]-> opens by [01.Basics].

When you check "brightness sensor-" of controller loading, an address changes and uploads the input terminal number in (A4).

When [tool]-> [serial monitor] of Arduino -IDE is clicked after program execution, a monitor indicates the signal a monitor window is inputting to the terminal which appeared and designated it.

### 3.3.9. (別売の) サーボモータ追加例



デジタルピン 5 番に追加し実験  
※サーボモータコードとコン  
トローラの接続極性を注意  
して接続。



別売品 SG92R

#### 実験例 1 : Sweep

※ソースコードは、PC/MyDocuments/Arduino/ のサンプル  
フォルダ [Servo] に、ファイル名【Sweep】で格納されています。  
ソースは C 言語で記述されています。

```

Sweep | Arduino 1.0.5 r2
// Sweep
// by BARRAGAN <http://barraganstudio.com>
// This example code is in the public domain.

#include <Servo.h>

Servo myservo;
    // create servo object to control a servo
    // a maximum of eight servo objects can be created
int pos = 0;
    // variable to store the servo position
void setup()
{
  myservo.attach(9);
    // この位置のピン番号9をサーボ接続した5番に変更して使用します。
    // attaches the servo on pin 9 to the servo object
}

void loop()
{
  for(pos = 0; pos < 180; pos += 1)
    // goes from 0 degrees to 180 degrees
    {
      // in steps of 1 degree
      myservo.write(pos);
      // tell servo to go to position in
variable 'pos'
      delay(15);
      delay(15);
    }
}
    
```

```

    // waits 15ms for the servo to reach the position
  }
  for(pos = 180; pos>=1; pos-=1)
    // goes from 180 degrees to 0 degrees
  {
    myservo.write(pos);
    // tell servo to go to position in variable 'pos'
    delay(15);
    // waits 15ms for the servo to reach the position
  }
}
    
```

必要事項を変更後、アップロードしてお使いください。

#### 実験例 2 : Knob ドアノブを回すような動きをします。

※ソースコードは、PC/MyDocuments/Arduino/ のサンプル  
フォルダ [Servo] に、ファイル名【Knob】で格納されています。  
ソースは C 言語で記述されています。

```

Knob | Arduino 1.0.5 r2
//Knob
// Controlling a servo position using a potentiometer (variable resistor)
// by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>

#include <Servo.h>

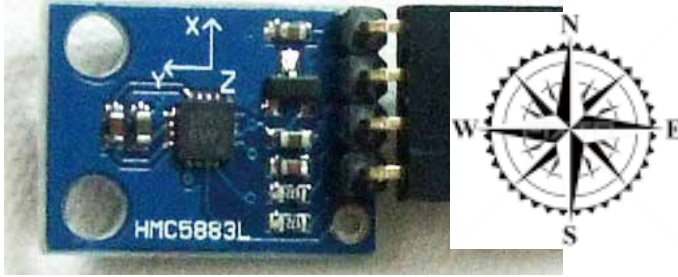
Servo myservo;
    // create servo object to control a servo

int potpin = 0;
    // analog pin used to connect the potentiometer
    // この位置の Pin 番号を A5 に変更すると、スライダで制
    御量を変化させることができます。
int val;
    // variable to read the value from the analog pin
void setup()
{
  myservo.attach(9);
    // attaches the servo on pin 9 to the servo object
    // この位置のピン番号9をサーボ接続した5番に変更して使用します。
}

void loop()
{
  val = analogRead(potpin);
    // reads the value of the potentiometer (value
    between 0 and 1023)
  val = map(val, 0, 1023, 0, 179);
    // scale it to use it with the servo (value
    between 0 and 180)
  myservo.write(val);
    // sets the servo position according to the
    scaled value
  delay(15);
    // waits for the servo to get there
}
    
```

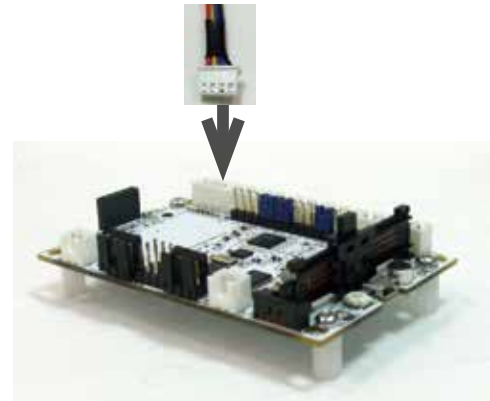
必要事項を変更後、アップロードしてお使いください。

3.3.10. (別売の) I<sup>2</sup>C コンパスセンサ追加例



接続は専用の I<sup>2</sup>C ケーブルを使用してください。

コントローラボード	コンパスセンサ
⊕	VCC
3 SCL	SCL
2 SDA	SDA
⊖	GND



X 方位	出力値	分解能
北 N	0	1 度単位で出力
東 E	90	
南 S	180	
西 W	270	

・使用例



コンパスセンサ RDI-5883L のサンプルプログラムです。

RDC-103 の I<sup>2</sup>C コネクタに RDI-5883L を接続します。  
1 度単位で角度を取得できます。

プログラムの動作手順

- ・変数「direction」に角度の値を格納
- ・シリアルモニタに出力
- ・角度が 180 度以上になったら LED を点灯、それ以下の場合は消灯

・利用例

ロボットによる「サッカー競技」など、守備 / 攻撃などの方向がある競技に利用します。ロボットの方向によって、守備行動、攻撃行動を変化させます。OWNゴールを防止する目的などに利用します。



サッカー競技に利用…「challenge\_range\_kick\_sample.abp」に HMC5883L コンパスセンサを追加したサンプルプログラム…

ソースコードは、PC/MyDocuments/Arduino/へ配置したサンプルフォルダ [ArduBlock Examples] に、ファイル名 [24\_04\_challenge\_range\_kick + sample.abp] で格納されています。Arduino/ArduBlock で、PC/MyDocuments/Arduino/ArduBlock Examples の中に配置したサンプルを開くと確認できます。

### 3-4. パーツアクセサリ

RoboDesignerPlus RDC-102,103 使用時に、充実拡張させるパーツアクセサリの紹介です。  
詳細は JAPAN ROBOTECH のホームページでご確認ください。

#### 1. コネクタ付 6V 電池ケース RDP-8093x4P



¥300\_(税抜き)  
単 3 x 4 本電池ケース、コード長さ 30cm、端末 JST 型 2 ピンコネクタ付

#### 2. コネクタ付 4.5V 電池ケース RDP-8093x3P



¥300\_(税抜き)  
単 3 x 3 本電池ケース、コード長さ 30cm、端末 JST 型 2 ピンコネクタ付

#### 3. コネクタ付 3V 電池ケース RDP-8093x2P



¥300\_(税抜き)  
単 3 x 2 本電池ケース、コード長さ 30cm、端末 JST 型 2 ピンコネクタ付

#### 4. USB 単 3 x 4本電池ケース RDP-8093x4USB



¥900\_(税抜き)  
単 3 x 4 本電池ケース、USBコネクタ付き、

#### 5. マイクロ USB 接続コード RDP-824 ¥400\_(税抜き)



USBケーブル A オス~マイクロ B オス 1.5 m A-m i c r o B

#### 6. 電池ケース コネクタ変換コード RDP-825



2 本組 ¥400\_(税抜き)  
バラ線コード~ JST 型 2 ピンコネクタ変換ケーブルです。長さ10cm 加工用熱収縮チューブ付

#### 7. DC ジャック -2 ピン変換ケーブル RDP-826



¥400\_(税抜き)  
2.1mm 標準 DC ジャック~ JST型 2 ピンコネクタ変換ケーブル、長さ 11cm AC アダプター接続時に使用

#### 8. 基板式電源スイッチ RDP-828



¥350\_(税抜き)  
電池ケースなどにスイッチを追加加工するスイッチ基板です。3 ピンコネクタ一接続式

#### 9. センサケーブル

20cm RDP-831 ¥400\_(税抜き)  
30cm RDP-832 ¥400\_(税抜き)



両端XH3極コネクタ付

#### 10. モーターケーブル

20cm RDP-833 ¥400\_(税抜き)  
30cm RDP-834 ¥400\_(税抜き)



XH2極~QL2極コネクタ付

#### 11. マルチメディアカードスロット MMC-DM3AT



¥300\_(税抜き)  
RDC-103データ記録に使用。  
取付加工は、基板裏面にハンダ付け自律行動時のセンサデータ回収に使用します。

#### 12. エンコーダケーブル

RDP-835 ¥500\_(税抜き)



RDC-103\_TYPE 3 にエンコーダを接続する拡張用ケーブル  
RDO-502EN用エンコーダケーブル 30cm  
XH3極~QL1極 x 3 コネクタ付  
拡張用 4 ピンヘッダーピン付属  
RDO-502EN付属エンコーダケーブルで長さが足りない時にもご利用ください。

#### 13. ホイール付ギアードモータ RDO-502 ¥1,800\_(税抜き)



130 型モータ内臓で、みの虫クリップ/コネクタでの接続可能です。電源電圧 DC4.5V 付属ケーブル20cm

#### 14. エンコーダ付ギアードモータ RDO-502EN



¥3,000\_(税抜き)  
単相出力のエンコーダ付で、制御の学習にご利用いただけます。付属ケーブル20cmXH3P-QL3P 電源電圧 DC4.5V 付属モーターケーブル 20cm  
エンコーダ: 1 周 8 パルス 単相モータ回転軸上にエンコーダ装備

#### 15. エンコーダ付モータ RDO-29BMA ¥3,700\_(税抜き)



2 相出力の12パルスエンコーダ付で、フィードバック制御の学習などにご利用いただけます。  
電源電圧 DC12.0V 付属ケーブル30cm 先バラ

16. エンコーダ・ギアヘッド付DC モータ RDO-29B50G27A, 54A  
 ¥7,800\_(税抜き)




エンコーダ付モータRDO-29BMA にギアヘッドを付けたモータです。減速比は 1/27、1/54の 2 種から選べます。  
 電源電圧 DC12.0V 付属ケーブル30cm 先バラ

17. ギアヘッド付 DC モータ RDO-29B36G10A  
 ¥3,280\_(税抜き)




ギアヘッド付モータです。減速比 1/10 回転数 370rpm 電源電圧 DC12.0V 電源ラグ端子付き

18. タッチセンサー JES-7022 ¥1,200\_(税抜き)



スプリングを利用した組み立て式で、接触式センサの仕組みがわかりやすくなっています。付属ケーブル30cm

19. アナログ赤外線センサー JES-7023 ¥1,500\_(税抜き)



アナログ出力の小型でシンプルな赤外線センサです。赤外線LED 付き、アクティブ/パッシブ切り替えスイッチ付き。30cmケーブル付

20. 変調赤外線センサー RDI-203JR ¥1,200\_(税抜き)



テレビなどの赤外線リモコンの変調信号(38kHz 搬送波)を受信するためのセンサです。ロボカップジュニアサッカー公式ボール(パルスモード)に対応しています。30cmケーブル付

21. 照度センサー RDI-204 ¥3,000\_(税抜き)



照度(環境の明るさ)を測るセンサです。太陽光～室内明るさまで測れるレンジ切替式です。30cmケーブル付

22. 測距センサー RDI-209 ¥1,800\_(税抜き)



赤外線を照射し、その反射量(距離)に応じたアナログ電圧を出力するセンサです。測定可能範囲は、10～80cm。30cmケーブル付

23. 超音波センサー RDI-HCSR04 ¥1,200\_(税抜き)



超音波を照射し、その反射量(距離)に応じたアナログ電圧を出力するセンサです。分解能:0.3cm 測距範囲:2～450cm

24. I<sup>2</sup>C コンパスセンサ RDI-5883L ¥1,800\_(税抜き)



地磁気を計測、分解能360、出力1度単位。I<sup>2</sup>C ケーブル付属です。ケーブル長 20cm

25. 赤外線反射センサー RDI-211 ¥1,200\_(税抜き)



対象物に赤外線を照射し、反射強度をアナログ出力します。小型・薄型・近距離計測用 焦点距離 1mm ケーブル長 30cm

26. R/G/B LED ライトスタンド RDS-LEDST404 ¥6,800\_(税抜き)



R/G/B LED ライトRDO-404 ボックス型スタンドセットです。12cm 角程度の水耕栽培プランターなどでそのまま実験などに利用することができます。



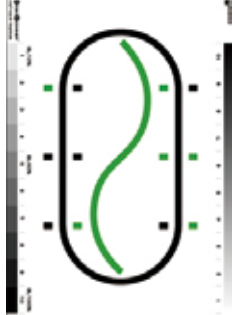
育成光の色の違いで、植物の味が変化します。RDC-103と接続、制御基板でプログラムにより、R/G/B調整が可能です。

27. R/G/B LED ボード RDO-404 ¥5,800\_(税抜き)



水耕栽培実験などに利用できるLED照明ボードです。切替スイッチで赤/緑/青の3色とその組み合わせで発光できます。

28. ライトレース&グレイスケールシート RDP-971 2枚組 ¥2,380\_(税抜き)



赤外線センサを使った明るさを測る実験や、ライトレースプログラムの実験、調整ができるシートです。シートサイズ:594x841mm(A1版)



## 4. プログラム環境の使い方

### 4-1. プログラム開発環境使用事前準備

(プログラム環境 Scratch,Arduino,ArduBlook, 及び RDC-102 動作環境とデバイスドライバーのインストールが完了していない場合は、[2. プログラム開発環境の準備] を参照して、先にインストールを完了させて下さい)



#### [1]. パソコン (PC) と基板を接続します。

1. PC と基板を、USB 接続コードで接続します。
2. 接続すると PC が基板を感知し、PC 側の「COM ポート」が自動設定されますので、次の手順に従い、COM 番号を調べてください。

#### [2]. COMポートの確認

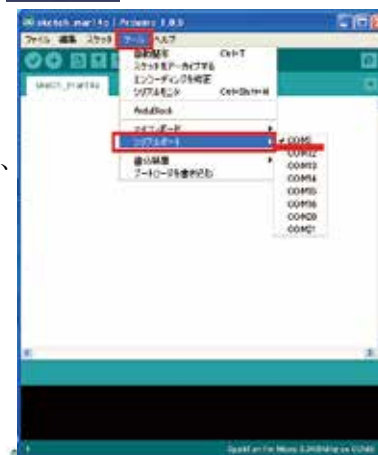
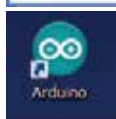
【Windows10 の場合】

1. [コントロールパネル] を選択します。
2. コントロールパネルの [ハードウェアとサウンド] を選択します。
3. [ハードウェアとサウンド] 内の [デバイスマネージャー] を選択します。
4. ポート (COM と LPT) をクリックし『STEM Du RDC-102(COMxx)』の番号を確認し、記録します。



#### [3]. Arduino-IDE の通信ポート設定

1. Arduino-IDE のアイコンをクリックして開発環境をスタートさせます。
2. 起動した Arduino-IDE の [ツール] ▶ [シリアルポート] の COM 番号を、調べた番号に設定します。出現するリストの該当 COM 番号を選択・クリック指定を行います。リスト左端に ☑ 印が付きます。
3. Arduino-IDE の [ツール] ▶ [マイコンボード] をクリックし、出現するマイコンボードリストで、[STEM Du/RoboDesigner+RDC-102w/ATmega32U4 3.3V 8MHz] を選択・クリック指定を行います。リスト左端に ● 印が付きます。



How to use the program environment

### Program development environment use preliminary preparations

(Program environment Scratch,Arduino,ArduBlook and RDC-102 When working environment and installation of a device driver aren't complete, please refer to [2. Preparations of a program development environment] and complete installation.)

#### [1]. PC and a circuit board are connected.

1. PC and a circuit board are connected by a USB connecting cable.
2. When it's connected, a PC senses a circuit board, and "communication port" on the PC side is established automatically, so please check the COM number with the next procedure.

#### [2]. Confirmation of communication port [In case of Windows8]

1. A charm is indicated on the desktop screen, and [setting] is chosen.
2. [Control Panel] is chosen.
3. [Hardware and Sound] of a Control Panel are chosen.
4. [Device manager] of [Hardware and Sound] is chosen.
5. A port (COM and LPT) is clicked, the number of "STEM Du RDC-102 (COMxx)" is confirmed and it's recorded.

#### (3).Communication port setting of Arduino-IDE

1. You click an icon of Arduino-IDE, and make a development environment start.
2. Which is started Arduino-IDE [tool] ▶ [serial port] is opened. The COM number is set as the checked number.
3. [STEMDu/RoboDesigner+RDC-102w/ATmega32U4 3.3V 8MHz] is chosen and click designation is performed by the microcomputer board list which clicks Arduino-IDE [tool] ▶ [microcomputer board] and appears. A ● mark sticks to the list left end.



4.2.2. Scratch を起動する。

1). windows の場合：

[WinScratch1.4-stemdu01] > [Scratch] の中の、[Scratch4STEMDu.image] を、[Scratch.exe](猫顔マーク) へ、ドラッグ&ドロップして起動します。

2). Mac OS X の場合：

配置した [Scratch\_14\_for\_STEM\_Du\_01] の中の、[Scratch4STEMDu.image] を、ダブルクリックして起動します。

3). まず、動かしてみましょう。

1. Scratch はおもちゃのブロックを組み立てるような感覚で手軽にプログラムを作成できるソフトウェアです。必要なスクリプト（プログラム言語の一種）が「ブロック」として用意され、ブロックを組み合わせるだけでプログラムを作ることができます。

2. Scratch の画面は左右に分かれており、右側には「スプライト」と呼ばれる画像が、左側にはスクリプトの編集画面が表示されます。

3. 編集画面には「10歩動かす」「15度右に回す」のように平易な言葉で書かれた小さなブロックが一覧表示されています。動かしたいスプライト画像を指定し、「見た目」や「動き」、「音」などで分けられたブロックを編集画面にドラッグ&ドロップします。

4. ブロックをクリックすると、ブロックに記載された内容に応じて画面右側の画像が動き出します。  
\* まずは画像を左から右に移動させたり、吹き出し文字の台詞を表示させたりするといった、簡単な動作をさせながら操作を覚えていきます。

5. 操作に慣れてきたら「調べる」にあるブロックを使って、イベントに応じて異なる動きをするよう設定してみましょう。  
・例えば複数の画像が重なったときに「ガチャ」と音を鳴らしてぶつかった様子を演出したり、画像の上でクリックしたときに向きを変えたりするなど、条件によって異なる動作をさせることが可能です。  
・上手に組み合わせることでちょっとしたゲームを作ることも夢ではありません。

6. 完成したアニメーションは「発表モード」に切り替えることで、全画面表示で見ることが出来ます。

Scratch is started.

1). In case of windows:

[WinScratch1.4-stemdu01] > drag and drop does and starts [Scratch4STEMDu.image] in [Scratch] to [Scratch.exe] (cat facial mark).

2). When it's Mac OS X:

[Scratch4STEMDu.image] in arranged [Scratch\_14\_for\_STEM\_Du\_01] is double-clicked and started.

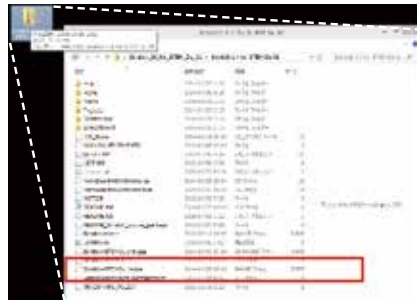
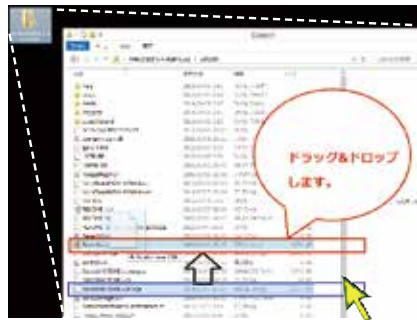
3). First, we'll move that.

1. Scratch is the software which can make a program easily by the sense to put a block of a toy together. A necessary script is prepared as "block", and it's possible just to combine a block and make a program.

2. A screen of Scratch is divided into left and right, and the picture called "Sprite" is shown to the right side and an edit display of a script is shown to the left side.

3. The small block written by simple words is showing a glance to an edit display, "It's moved 10 steps." "It's turned to the right 15 times." like.  
\* You designate the Sprite picture you'd like to change, and drag and drop makes the block divided among "the appearance", "movement" and "sound" etc. an edit display.

↓ Scratch which starts



4. When a block is clicked, a picture on the screen right side begins to move according to the contents indicated on a block.

\* We'll make them move easily first, and remember operation.

You move a picture from the left to the right, please, of the balloon character, a word, please make them indicate.

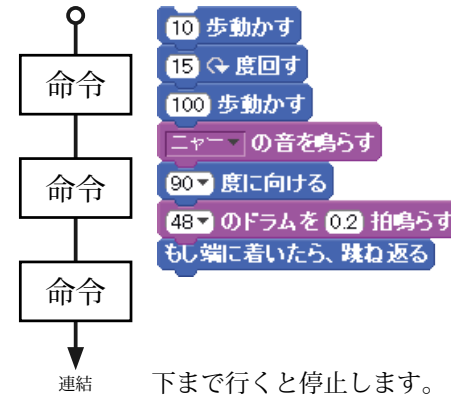
5. If you're being experienced in operation, it "is checked", you'll establish it so that a movement different according to the event may be done using some blocks.

\*For example such as changing the direction ringing "moth" and sound and producing the crashed state, and when more than one picture was piled, when clicking on the picture, it's possible to make them do movement different depending on the conditions.

\*It isn't also a dream to make a form of game with combining well.

6. It's possible to make the cartoon film made by a mode change full display.

4.2.3. 命令ブロックの実行規則



1). スクリプトの実行プロセスは、「連結」が基本で、並んだ命令ブロックが上から順に処理されます。また、「繰り返し」と「条件分岐」という処理手順も使えます。

2). 連結  
1. 先ず、最も基本的で簡単な「連結」の例です。くっつけた「命令ブロック」は、上から順に実行されます。

Execution regulation in an order block

1). "Connection" is a basis for Executing Process of a script, and the order block you lined is disposed of in turn from the top. You can also use a process as "repeat" and "conditional branch".

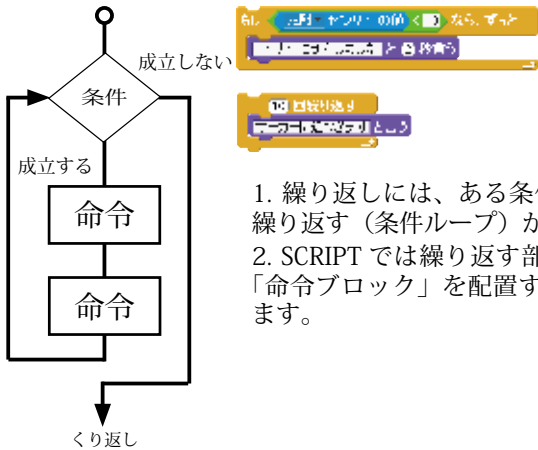
2). Connection

1. An easy example of "connection". Attached "order block" is executed in turn from the top. When go to the bottom, a program stops.

3). Repetition (condition loop)

1. Only while the condition to have that is met to repeat it, there is a repeated (condition loop).  
2. A repeated part is written in the □ in SCRIPT. When "order block" is arranged, the size of the □ changes.

3). くり返し (条件ループ)

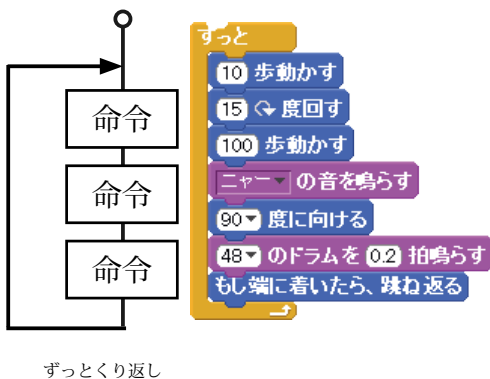


1. 繰り返すには、ある条件が満たされている間だけ繰り返す (条件ループ) があります。  
2. SCRIPT では繰り返す部分を □ の中に書きます。「命令ブロック」を配置すると□の大きさが変わります。

4). Repetition (infinite loop)

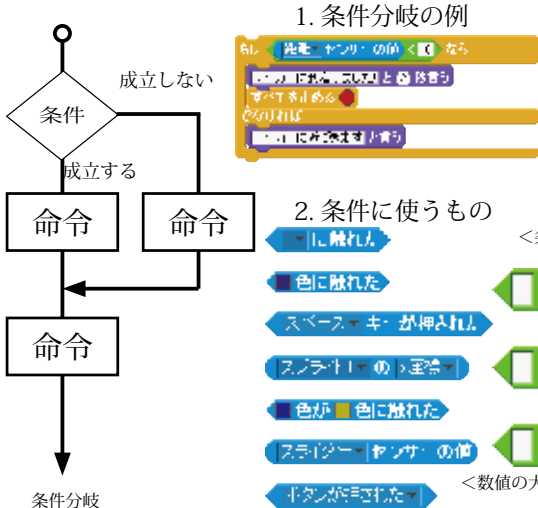
1. There is also a (infinite loop) repeated eternally.

4). ずっとくり返し (無限ループ)



1. ずっと繰り返す (無限ループ) も、あります。

5). 条件分岐 (1)



1. 条件分岐の例

2. 条件に使うもの

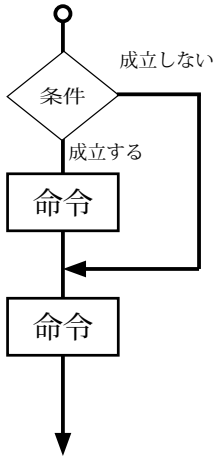
- <条件>
- 色に触れた < >
- スペースキーが押された < >
- 色が色に触れた < >
- スライダの位置が < >
- ボタンが押された < >
- かつ < >
- または < >
- ではない < >
- <数値の大小比較>
- <論理的条件>

5). Conditional branch (1)

1. An example of a conditional branch

2. Something to use for the condition

6). 条件分岐 (2)



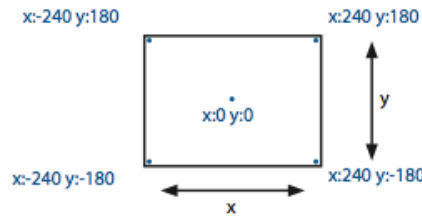
7). スタート命令ブロック緑の旗をクリックした時あるキーが押された時など、条件に応じて、スクリプトの実行をスタートします。

8), 停止命令



9), 画面の座標

舞台 (ステージ) 上の x、y 座標は次の通りです。画面中央が原点 (0,0) です。注意してください。



6). Conditional branch (2)

7). Start order block

When the time when a green flag was clicked and some keys were pushed, execution of a script is started according to the condition.

8).Order of the stop

9).Coordinate of a screen

The coordinate on the stage is (x,y) as follows.

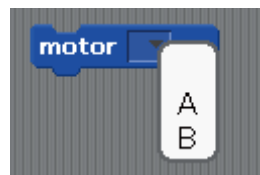
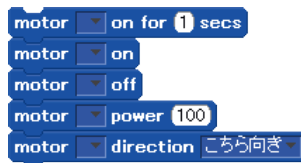
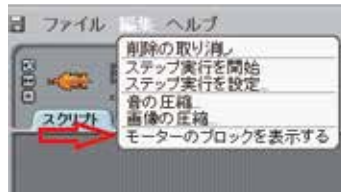
The screen center is the starting point (0,0), so please be careful.

4.2.4. ロボットで多く使うスクリプト

(1). モーター

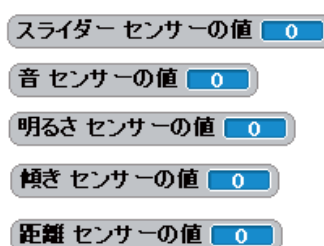
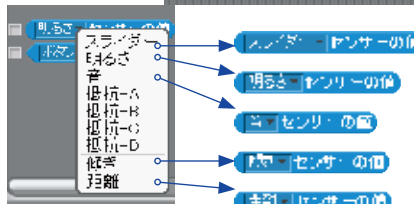
1. モータのブロックは、[編集]▷[モーターのブロックを表示する]をクリックします。
2. ブロックパレットの「動き」リストに追加されます。
3. モータは A,B の 2 種類を使えます。
4. Scratch と RDC のモータ端子

Scratch	RDC
motorA	M1
motorB	M2



(2). センサー

1. センサーは下記の 5 種類を選ぶことができます。
2. センサ-ブロック左側の□にチェック入れると、センサー値を調べるマークが、右側のステージに配置されます。
  - この計測値を、「しきい値」条件の参考とします。



The script used much by a robot

(1). Motor

1. [Edit] clicks ▷ [A block of a motor is indicated.] and uses a motor block.
2. It's added to the "moving" list of block palettes.
3. A motor can use 2 kinds, A and B.
4. Scratch and motor terminal for RDC.

(2). Sensor

1. A sensor can choose the following 5 kinds.
2. When  on the sensor - block left side can be made a check on.
  - Mark who checks the sensor value is arranged in a right stage.
  - \* This value is made reference of the "threshold value" condition.

### 4-3. Scratchの動作実験

#### 4.3.1. Sensor-Board の起動方法

(1).【パソコン (PC) での準備】

1. マイコンボードをPCに接続します。
2. [PC]▷[ハードウェア]▷[デバイスマネージャ]で、COM番号を確認します。



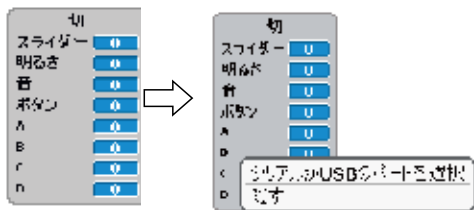
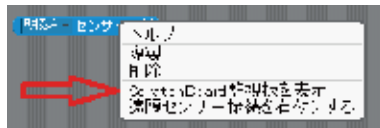
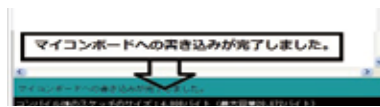
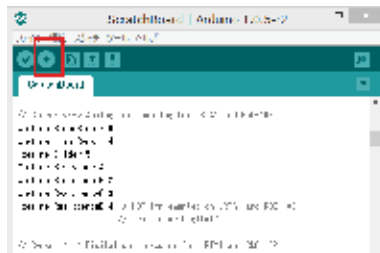
(2).【Sensor-Board に設定】 Scratch のプログラムをコントローラで実行できるようにするために RDC ヘスケッチを書き込み、Sensor-Board に設定します。

1. [Arduino-IDE]▷[スクラッチの例]▷[STEMDu]▷[Type\_1]の[ScratchBoard.ino]を開きます。
2. [Arduino-IDE]▷[ツール]▷[シリアルポート]にて調べたCOM番号に設定します。
3. Arduino-IDEの(→)をクリックし、マイコンボード(RDC)に書き込みます。
4. 書き込みに成功するとメッセージが表示されます。



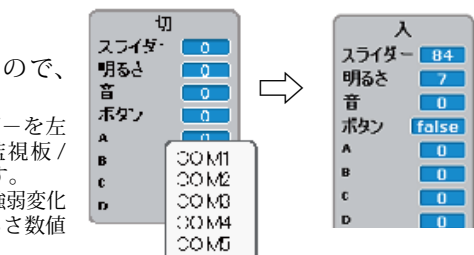
(3).【Scratch での準備】 通信設定を行います。

1. センサ-ブロックを選択し、右クリックすると、メニューが現れます。「ScratchBoard 監視板を表示」を選択し、クリック実行します。
2. ステージに「ScratchBoard 監視板」が出現します。
3. 「ScratchBoard 監視板」を右クリックし、「シリアルか USB のポート」を選択すると、通信ポート (COM ポート) リストが現れますので、調べておいた COM 番号に設定します。  
 ・通信設定完了すると、「切」→「入」へ変化し、ボード搭載センサのデータ値が表示されます。



4. 使用準備ができましたので、確認実験です。

- \* マイコンボードのスライダーを左右へ動かしてみます。→監視板/スライダー数値が変化します。
- \*\* ライトセンサへ光を当てて強弱変化させてみます→監視板/明るさ数値が変化します。
- \*\*\* これで、マイコンボードは ScratchBoard として動作していますので、モータ制御などを含むスケッチを実行させると、接続しているモータが動き始めますので、机の上から落としたりしないように注意します。



#### Movement experiment of Scratch Initiation method of Sensor - Board

(1). [Preparations by a PC]

1. A microcomputer board is connected to a PC.
2. [PC]▷[Hardware]▷[Device manager], the COM number is confirmed.

(2). [It's set as Sensor - Board.] a sketch is written in RDC and it's set as Sensor - Board because I'll can execute a program of Scratch by a controller.

1. [Arduino-IDE]▷[Example of a scratch]▷[STEMDu] the one of the [Type\_1]▷[ScratchBoard.ino] is opened.

2. It's set as the COM number checked in [serial port] in [Arduino-IDE] [tool].

3. (→) of Arduino-IDE is clicked and it's written in a microcomputer board (RDC).

4. When I succeed in writing in, a message is indicated.

(3).[Preparations in Scratch] communication setting is performed.

1. When a sensor - a block is chosen and right-clicked, the menu appears. "Of a ScratchBoard watcher, indication" is chosen and a click is executed.

2. "ScratchBoard Watcher" appears in a stage.

3. When "ScratchBoard watcher" is right-clicked and "of a serial or a port in USB, choice" is chosen, a communication port list shows, so it's set as the checked COM number.

\* When communication setting is completed, a data value of a sensor with a board changes into "on" "off", and is indicated.

4. Use preparations are done, so it's a confirmation experiment.

\* Slider-of a microcomputer board will be changed to left and right. -> A watch board/a slider - a figure changes.

\*\* You apply light to a light sensor, and the watcher/ brightness numerical value I'll make do strength transition of-> changes.

\*\*\* A microcomputer board is moving as ScratchBoard with this, so when I make them carry out the sketch which includes motor control, a connected motor begins to move, so I pay attention so as not to drop it from the top of the desk.

### 4.3.2. スクリプト例を使用して動作実験

(1). ScratchProject の Example4-1 を開いてみます。

1. ○○センサの値の3か所を「スライダー」に選択変更します。
2. このスクリプトではスタートが「がクリックされたとき」ですので、Scratch 画面右肩に配置されている「緑色の旗」をクリックして、スクリプトを実行します。
3. 実行中は、周囲が白枠で囲われます。
4. マイコンボードのスライダーを動かして監視板の数値を観察してください。
5. マイコンボードの motor1、motor2 に接続しているモータの動きが「しきい値」を境にして、プログラム通りに回転を変化させながら動くことが確認できます。



### 4.3.3. Scratch では、接続した状態のまま、マイコンボードが動作します。

1. Scratch では、常にパソコンとマイコンボードを USB で接続して使います。
2. マイコンボードの PC 接続を外して、自律型動作をさせるときは、ArduBlock をご利用ください。



### Experiment on movement using a script example.

(1). Example4-1 of ScratchProject is read.

1. Choice changes 3 points of a ,  sensor value to "slider".
2. A start is A by this script, so the "green flag" arranged by a Scratch screen right shoulder is clicked and a script is executed.
3. During carrying out, the environment is surrounded with a white frame.
4. Please change a slider of a microcomputer board and observe the numerical value of Watcher.
5. A movement of the motor connected to motor1 and motor2 of a microcomputer board can confirm the thing which moves while changing a revolution into a program street on reaching "threshold value".

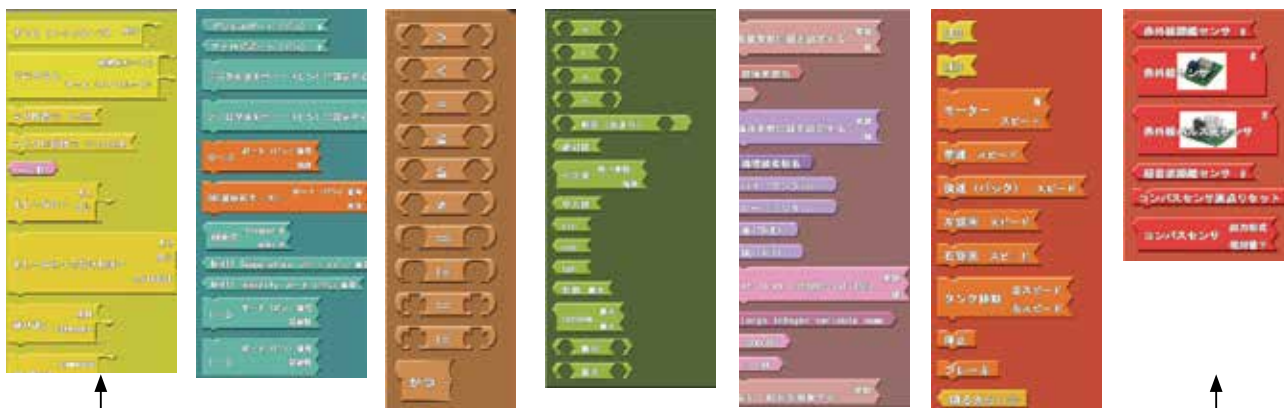
### A condition and a microcomputer board of the connected position move in Scratch.

1. Always it's connected with a PC and it's used in Scratch.
2. When removing a PC connection of a microcomputer board and making them do autonomous movement, please use ArduBlock.

## 4-4. ArduBlock

### 4.4.1. ArduBlock構成

- [制御] [ポート(ピン)] [判断条件] [演算] [変数(定数)] [STEMDu] [STEMDu Accessories]



**アイコンパレット icon palet**  
アイコンを格納しています。



**新規作成 New Sketch**  
新規スケッチを作成します。

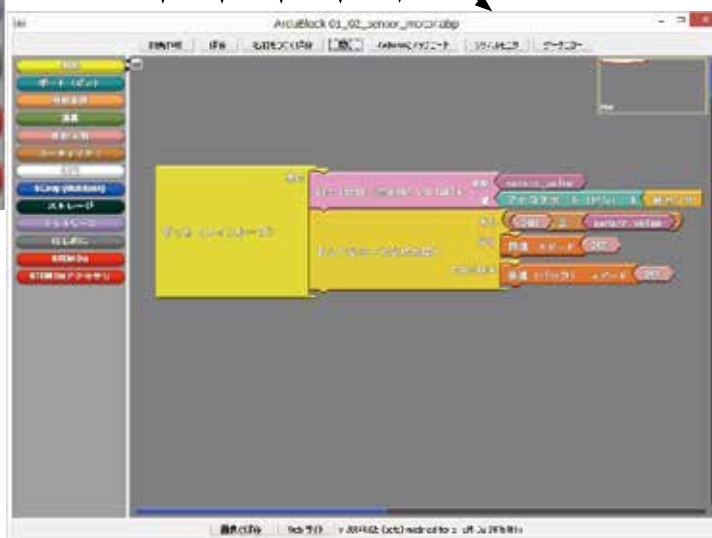
**保存 Save**  
作成したスケッチを保存します。

**名前をつけて保存 Save As**  
作成したスケッチに名前をつけて保存します。

**開く Open**  
保存しているスケッチを開きます。

**Arduino にアップロード upload**  
作成したスケッチを Arduino-IDE へ送ります。

**シリアルモニタ Serial monitor**  
センサーデータをシリアルモニターします。





## 4.4.2. ArduBlock の使い方

### (1) .ArduBlock を起動します。

1. Arduino-IDE を起動し、IDE の [ ツール ] ▶ [ ArduBlock ] を選択し、クリックします。



右の画面が、立ち上がった ArduBlock です。

### (2) .LED で光実験をする。

1. デジタル 13 番ピンにつながった LED を 1 秒毎に点滅させるためのスケッチを書いてみます。

2. まずは、左列アイコンパレットの一番上にある { 制御 } をクリックした時に出現するサブウィンドウから [ ループ ] を選択し中央のフィールドにドラッグ & ドロップします。

\* プログラムには、[ ループ ] が必ず必要です。

\* [メインプログラムのループ]

[初期化ルーティンを伴うループ]  
のいずれかのループでプログラム作成します。



3. [ ループ ] をドロップした画面に、次を加えます。

左列アイコンパレットの { ポート (ピン) } をクリックした時に出現するサブウィンドウから [ デジタル値をポート (ピン) に設定する ] を選択し中央フィールドにドラッグ & ドロップします。

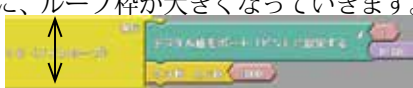
4. ループの中のパーツの位置に収まるようにドロップすると「カチッ」と音がして、ブロックが、ループにはまり込み結合します。



5. 次にアイコンパレットの { 制御 } をクリックしたときに現れるサブウィンドウから \* [ ミリ秒待つ ミリ秒 ] をループ枠の中へドラッグ & ドロップします。



6. ブロックをドロップするたびに、ループ枠が大きくなっていきます。プログラムの大きさに合わせて変化します。



\* [ミリ秒待つ ミリ秒] は、結合アイコンの動作をその設定時間継続動作するという意味です。

## How to use ArduBlock

### (1). ArduBlock is started.

1.Arduino-IDE is started, which is IDE [tool] ▶ [ArduBlock] is chosen and it's clicked.

An upper screen is ArduBlock.

### (2). A light experiment is done by an LED.

1. The sketch for number 13 of digital to make the LED which connected with a pin flash on and off every 1 second will be written.

2. When clicking the {control} on number one of left side, you choose [loop] from the right sub-window from which you emerge, and do drag and drop in a central field.

3. [Loop], the next is added to the screen which dropped.

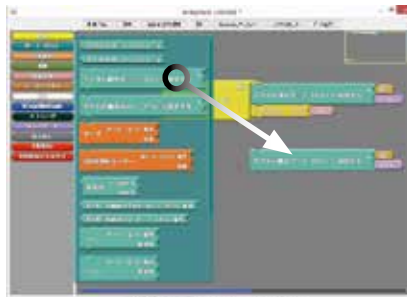
•When clicking the left side {port (pin)}, ▶ [the digital value set as a port (pin)] from the sub-window from which emerge, you choose and do drag and drop in the center field.

4. That it drops so that it may fit into the location of the parts in the loop, it makes a noise with "Cachi", and a block telescopes in a loop and combines.

5. When clicking {control}, next drag and drop does [the millisecond indicated] to the inside of a loop frame from the sub-window which shows.

6. A block, every time it drops, a loop frame is becoming big. It changes according to the size of the program.

7. もう一度、左列 { **ポート (ピン)** } をクリックし、出現するサブウィンドウから [デジタル値をポート (ピン) に設定する] を選択し中央フィールドにドラッグ&ドロップします。



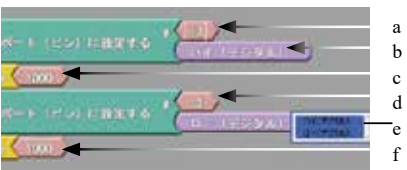
8. プログラムがさらに大きくなりました。

9. さらに、もう一度 { **制御** } のサブウィンドウから [ミリ秒待つ ミリ秒] をループ枠の中へドラッグ&ドロップします。



10. ドロップ配置したブロックの設定を行います。

- a).LED 配置番号にあわせてポートピン番号を 13 に設定する。
- b).HIGH は ON、LOW は OFF
- c).ミリ秒単位で記入、1000 ミリ秒は 1 秒
- d).LED 配置番号にあわせてポート (ピン) 番号を 13 に設定する。
- e).HIGH は ON、LOW は OFF
- f).ミリ秒単位で記入、1000 ミリ秒は 1 秒



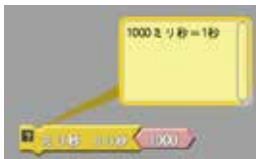
7. When clicking the left side { **port (pin)** } again, ▽ [the digital value set as a port (pin)] from the sub-window from which emerge, you choose and do drag and drop in the center field.

8. A program became bigger.

9. Drag and drop does more [milliseconds] to the inside of a loop frame from the sub-window of { **control** } again.

10. The block where a drop was arranged is set.

- a).The port pin number is set as 13 according to the LED arrangement number.
  - b).HIGH is on and LOW is off.
  - c).By the millisecond unit, entry and 1000 milliseconds are 1 second.
  - d).The port (pin) number is set as 13 according to the LED arrangement number.
  - e).HIGH is on and LOW is off.
  - f).By the millisecond unit, 1000 milliseconds are 1 second.
- \*a) It was set as 13.  
\*b)It was set as HIGH.  
\*c) It was set as 1000.  
\*d) It was set as 13.  
\*e) Please set it as LOW.  
\*f)It was set as 1000.



- \* a).13 に設定しました。
- \* b).HIGH に設定しました。
- \* c).1000 に設定しました。
- \* d).13 に設定しました。
- \* e).LOW に設定ください。
- \* f).1000 に設定しました。

・ 以上で、コントローラ RDC-103 の 13 番ポートに配置されている LED を ON (点灯) して 1000 ミリ秒 (1 秒) 経過後に、13 番ポート LED を OFF (消灯) するプログラムを、ずっと繰り返すプログラムが完成しました。

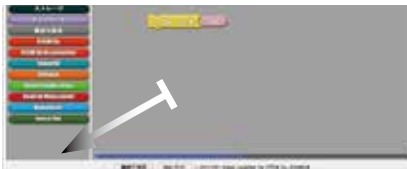


11. 操作の説明です。

ArduBlock で作成するプログラムは、基本「ループ」ブロックの中に作ります。左列一番上にある { **制御** } をクリックした時に出現するサブウィンドウから [ループ] を選択し中央のフィールドにドラッグ & ドロップします。

- \* プログラムには、[ループ] が必ず必要です。
- \* [メインプログラムのループ] [初期化ルーティンを伴うループ]

のいずれかのループでプログラム作成します。

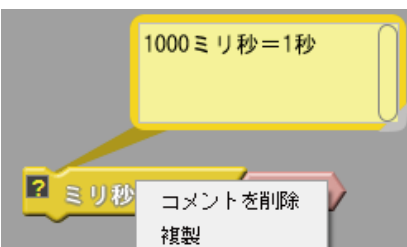


Delete : 配置済みブロックを消したいときは、左枠内へドラッグ & ドロップする。

Comment : コメントを作成すると、プログラムメモに便利です。ブロックにカーソルを合わせ右クリックで出現します。

プログラム中の Block 画像の [?] マークは、コメントがあることを知らせるアナウンスです。

- コメントを追加 (Add Comment) :
- 複写 (Copy) :
- コメントを削除 (Delete Comment) :
- 複製 (Clone) :



Delete : It's already arranged, a block, when you'd like to put it out, you do drag and drop inside the left limits.

Comment : When a comment is made, it's convenient for a program memo. You appear by a right click together with a cursor in a block.

この手続きを、必ずしてください。  
しないとプログラムのアップロードができません。エラーになります。

スケッチを Arduino へアップロードする前に下記の確認を必ず行います。

- ・ PC とマイコンボードの接続をします。
  - ・ Arduino-IDE の [ツール] ▶ 「マイコンボード」設定の確認、「シリアルポート」の接続 COM 番号設定をします。
12. [保存]、[名前をつけて保存] いずれかで、作成したスケッチを保存します。
- ・ 保存先 [PC] ▶ [ドキュメント] ▶ [Arduino] ▶ [ArduBlock Examples] ファイルフォルダーに保存します。
  - ・ 保存したプログラムは、いつでも読み込むことが可能です。
13. [Arduino にアップロード] をクリックし、作成したスケッチを、Arduino へアップロードします。
- ・ アップロードしたスケッチは、Arduino-IDE のスケッチに「C 言語」で表示されて、コンパイルされます。
14. コンパイル後は、すぐにマイコンボードへの書き込みが始まります。
15. 書き込み完了メッセージ



書き込みが失敗すると赤色表示でエラーを知らせます。  
**エラーメッセージ** Couldn't find a Leonardo on the selected port. Check

that you have the correct port selected. If it is correct, try pressing the board's reset button after initiating the upload.

- (1). マイコンボードを USB ポートへ接続していなかったり、(\*a) 通信ポート COM (No) 設定が間違っていたり、(\*\*b) ボードの動作環境が合っていないかたりすると、(\*\*c) 通信エラーが発生して、マイコンボードへの書き込みが失敗します。
- \*a). Arduino メニューバーの [ツール] ▶ [マイコンボード] メニューから接続したい Arduino ボードの名前を選びクリック指定します。(RDC-103 は、STEM Du/RoboDesigner+ RDC-102 w/ ATmega32U4 - 3.3V 8MHz)
  - \*\*b). Arduino メニューバーの [ツール] ▶ [シリアルポート] メニューから接続したいポート番号を選びクリック指定します。Windows の場合は COM3 といったような名前になっていて、数は 3 以上の場合もあります。
  - \*\*c). 「2-4. ドライバーインストール」の項を参照して適切なドライバーをセットください。

- (2). エラーメッセージを確認して、対策を施して、問題を解決した後で、再度、マイコンへの書き込みを行います。



Please be sure to do this procedure. When it isn't done, a program can't be uploaded. It'll be an error.

Before uploading a sketch to Arduino, the following confirmation is performed certainly.

- \* A PC and a microcomputer board are connected.
- \* Confirmation of the → "microcomputer board" setting which is Arduino-IDE [tool] and connection COM number setting of "serial port" are done.

12. A made sketch is preserved by one of [Save] and [Save as].

13. [To Arduino, upload] is clicked and a made sketch is uploaded to Arduino.

An uploaded sketch is shown to a sketch of Arduino-IDE by "C language", and is compiled.

14. After compiling, writing in to a microcomputer board will start immediately.

**\* Error message**

Couldn't find a Leonardo on the selected port. Check that you have the correct port selected. If it is correct, try pressing the board's reset button after initiating the upload.

(1) A microcomputer board isn't connected to a USB port.

(\*a) communication port COM (No) The setting is wrong.

(\*\*b) working environment of a board isn't right.

(\*\*c) then a communication error occurs, and writing in to a microcomputer board is failed.

\*a).The name of the Arduino board I'd like to connect is chosen from the [microcomputer board] menu which is Arduino menu bar [tool]. (For RDC-103, STEM Du/RoboDesigner+ RDC-102 w/ ATmega32U4 - 3.3V 8MHz)

\*\*b). You choose the portnumber you'd like to connect from whole menu of the subwindow in which is a Arduino menu bar [tool] ▶ [serial port].

・ It's COM3 and the name I needed in case of Windows.

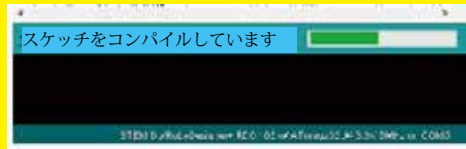
\*\*c). Please refer to an item of "2-4. Driver installation" and set an appropriate driver.

(2). After you confirmed the error message and did a measure, and settled a problem, you write notes in a microcomputer once again.

(3). マイコンボードへの書き込み失敗が続く場合



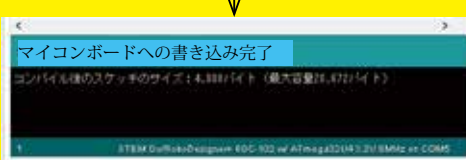
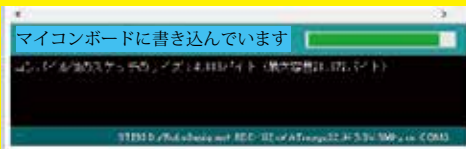
・アップロードがうまくいかない場合は、RDC-103 コントローラの RESET スイッチを「ダブルクリック」してください。  
 \* When upload doesn't work, you double-click the RESET switch of RDC-103 controller.



プログラムコンパイルの後、マイコンボードに書き込みが始まる前に、マイコンボードリセットボタンをダブルクリックします。・・・このタイミングです。



↑ RESET



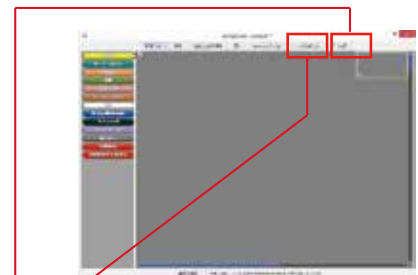
4.4.3. ArduBlock 仕様でのお断りとお願い

(1). この開発環境は、ArduBlock をベースに、「STEM Du」ブロックを追加したものです。弊社では、他社のセンサなどを使用するためのブロック群についての技術仕様、接続方法などのご相談・質問等の対応ができませんので、あらかじめ、ご了承ください。

(2). 下記の部分は、開発中の内容（予告のお知らせでメニューが見えています。）今のところ、使用できませんので、ご案内いたします。開発完了後は、弊社 H/P にてご案内申し上げます。



サポートできません。



◇ 開発中 □ ハードウェアコンフィギュレータ  
 今のところ：Pin 番号にて、I/O 設定をお願いします。  
 開発中 □ データロガー  
 今のところ：シリアルモニターをご利用ください。  
 (サンプルプログラムを準備しています)

(1). This development environment added a "STEM Du" block based on ArduBlock. The correspondence which are consultation and a question, etc. of the technological specification about the block group and the connection method to use a sensor of an other company in our company can't be done, so beforehand, please accept it.

#### 4.4.4. ArduBlock サンプルプログラム

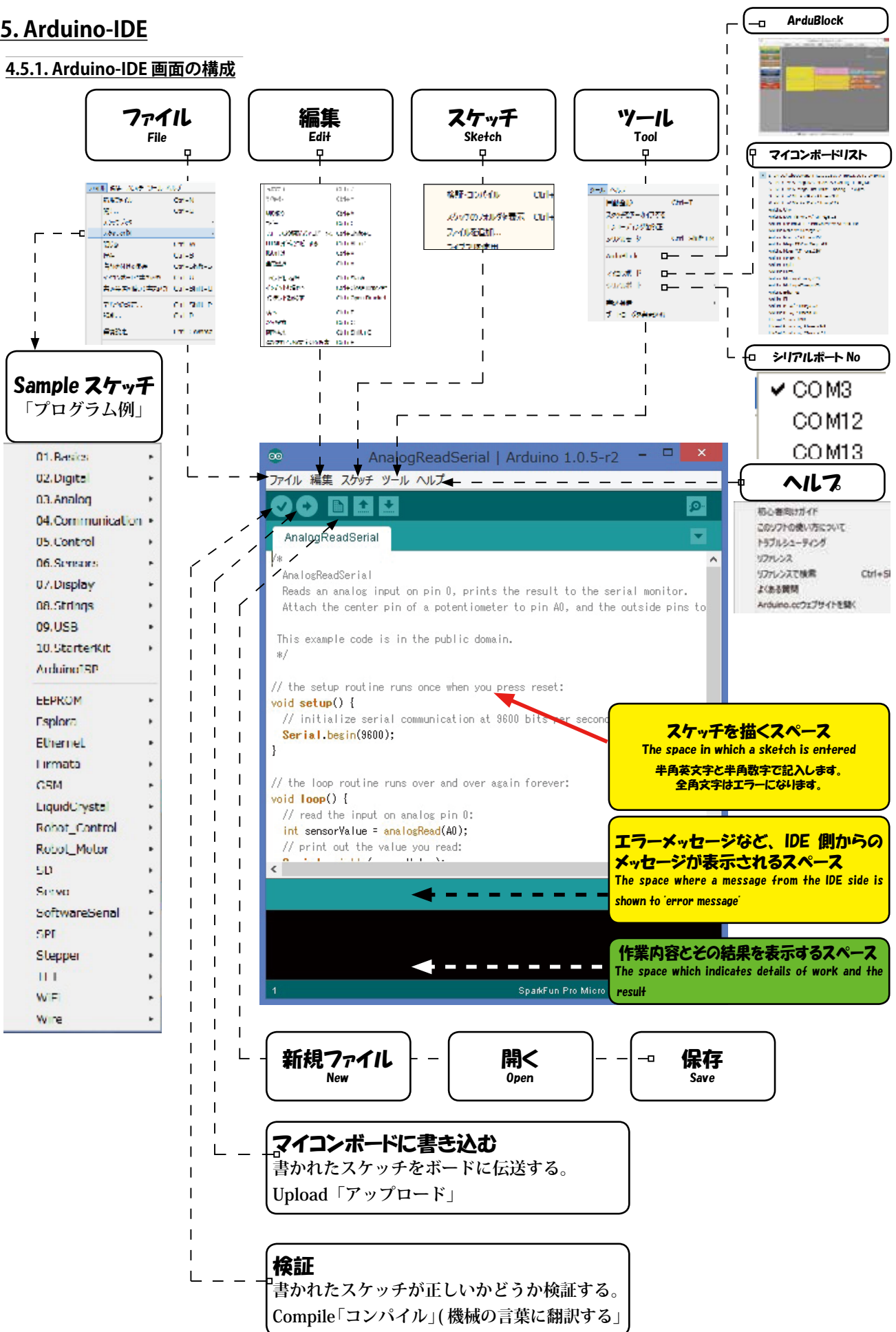
※サンプルは、PC/MyDocument/Arduino へ配置した ArduBlock Examples に格納されています、必要なプログラム名を指定して開きます。



ロボットモデル	サンプルプログラム名	プログラム概要
Robotics experiment	01_01_motor_slider.abp	スライダーを動かすと、出力される値に応じてモータの回転数が変化します。
Robotics experiment	01_02_sensor_motor.abp	音センサの入力に応じて、モータの前進、後進が変化します。
Robotics experiment	01_03_sensor_buzzer.abp	明るさセンサの入力に応じて、搭載ブザーが鳴ります。
Robotics experiment	01_04_ultrasonic_HCSR04.abp	超音波距離センサHCSR04 サンプルプログラムです。対象物との距離を計測します。
Robotics experiment	01_05_ultrasonic_PING.abp	超音波距離センサ PING サンプルプログラム Parallax のPING 用のサンプルです。
Robotics experiment	01_06_analog_checker.abp	アナログセンサの値をシリアルモニタで確認するArduBlock サンプルプログラムです。反射センサや変調赤外線センサなどの値をチェックして、しきい値設定の参考にします。
Robotics experiment	01_07_button.abp	押ボタンを押されると動作プログラムをスタートします。
Robotics experiment	01_11_I2Ccompass_HMC5883L.abp	ハネウェルのコンパスセンサHMC5883L のサンプルプログラムです。HMC5883L 用のブロックです。1 度単位で角度を取得できます。
Robotics experiment	01_12_I2Ccompass_HMC6352.abp	ハネウェルのコンパスセンサHMC6352 のサンプルプログラムです。HMC6352 は2 バイトで0.1 度単位の方位の値を返します。
RDS-X21	21_01_clash_avoidance_HCSR04.abp	超音波障害物回避ロボットのサンプルプログラムです。
RDS-X21	21_02_Line_board-lightsensor.abp	ライントレースサンプルプログラムです。搭載センサ使用
RDS-X23/24	24_01_base_floor_X24_sample.abp	赤外線反射センサ RDI-211 を使用して床の明暗に反応して動きます。
RDS-X23/24	24_02_base_ball_X24_sample.abp	変調赤外線センサ RDI-203JR を使用して赤外線ボールを探します。
RDS-X23/24	23_03_base_floor_ball_sample.abp	「floor_sample」と「ball_sample」を組み合わせたプログラムです。
RDS-X23/24	23_04_base_floor_ball_button_sample.abp	「base_floor_ball_sample」に、ボタンで動作プログラムをスタートする機能を加えてあります。
RDS-X23/24	23_05_base_stop_sample.abp	白いラインがあったら停止します。
RDS-X24	14_01_challenge_range_sample.abp	「floor_ball_sample」に測距センサを使ったドリブル動作を加えたプログラムです。
RDS-X24	24_03_challenge_range_kick_sample.abp	「challenge_range_sample」にキック動作を加えたプログラム
RDS-X24	24_04_challenge_range_kick+_sample.abp	「challenge_range_kick_sample」にアナログセンサを追加したプログラムです。
RDS-X24	24_05_challenge_range_kick+_button_sample.abp	「challenge_range_kick+_sample.abp」に、ボタンで動作プログラムをスタートする機能を加えてあります。
RDS-X24	14_05_challenge_compass_sample.abp	コンパスセンサの値から、ロボットが敵ゴールの方を向いている場合のみ前進します。
RDS-X24	14_06_challenge_compass+_sample.abp	「challenge_compass_sample.abp」に、ボタンで動作プログラムをスタートする機能を加えてあります。
RDS-E20	20_01_LED_RGB.abp	栽培工場のRGB サンプルプログラムです。
RDS-E20	20_03_LED_RGB_sequence.abp	時間に応じてRGB のon/off を制御して、色の組み合わせを変化させます。
RDS-E20	20_04_LED_RGB_colorChange.abp	スライダー(A5) の入力値に応じてRGB のon/off を制御して、色の組み合わせを変化させます。
RDS-E20	20_05_LED_RGB_color&brightnessChange.abp	明るさセンサ(A 4) の入力値に応じてRGB への出力を制御して、色の明るさを変化させます。暗いほど明るくなります。明るさが一定以上の場合、消灯します。
RDS-E20	20_06_LED_RGB_color&brightnessChange&timer.abp	ボタン(12)を押すと、一定時間消灯します。

## 4-5. Arduino-IDE

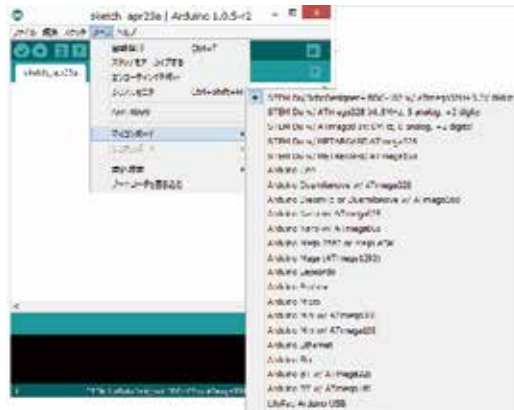
### 4.5.1. Arduino-IDE 画面の構成



### 4.5.2. Arduino-IDE の使い方

次の手順で Arduino ヘスケッチ (プログラム) をアップロードできます。

1. 開発環境でボードの種類を選びます。・メニューバーの [ ツール ] ▶ [ マイコンボード ] メニューから接続したい Arduino ボードの名前を選びクリック指定します。(RDC-103 は、STEM Du/RoboDesigner+ RDC-102 w/ATmega32U4 - 3.3V 8MHz)



2. シリアルポートを選ぶ手順
  - ・メニューバーの [ ツール ] ▶ [ シリアルポート ] メニューから接続したいポート番号を選びクリック指定します。Windows の場合は COM3 といったような名前になっていて、数は 3 以上の場合もあります。Mac の場合は /dev/tty.usbserial といったような名前になっています。

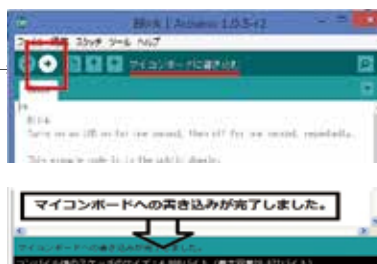
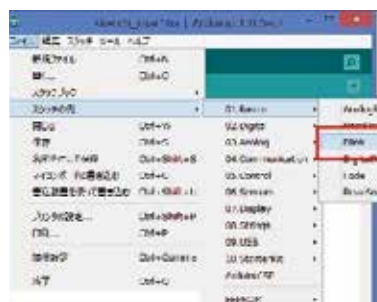


3. スケッチをアップロードする手順
  - メニューバーから [ ファイル ] ▶ [ スケッチの例 ] ▶ [ 01.Basics ] ▶ [ Blink ] を選んで、例題スケッチの「Blink」を開きます。

「Blink」を開いたエディタのアップロードボタンを押すだけで Arduino ヘプログラムが書き込まれます。数秒待つとボード上の RX と TX の LED が瞬くのが見えます。

アップロードボタン  
upload button

アップロードがうまく行けば、ステータスバーに「マイコンボードへの書き込みが完了しました。」と表示されます



アップロードが終わった数秒後に、ボード上の pin 13 LED が点滅を始めます。

pin 13 LED

こうなれば成功です!! これで Arduino にプログラムを書き込んで動かすことができるようになりました。



### How to use Arduino-IDE

A sketch (program) can be uploaded to Arduino by the next procedure.

1. The kind of boards is chosen by a development environment.

\* The name of the Arduino board I'd like to connect is chosen from the [a microcomputer, board] menu in the [tool] of a menu bar. (For RDC-103, STEM Du/RoboDesigner+ RDC-102 w/ATmega32U4 - 3.3V 8MHz)

2. The procedure from which a serial port is chosen

\* The portnumber I'd like to connect is chosen from the [serial port] menu in the [tool] of a menu bar.

・It's the name like COM3 in case of Windows, and the number is sometimes more than 3.

・It's the name like /dev/tty.usbserial in case of Mac.

3. The procedure which uploads a sketch.

From a menu bar [File] ▶ [Example of a sketch] ▶ [01.Basics] ▶ [Blink] is chosen and "Blink" of an exercise sketch is opened.

A program just presses an upload button of the editor which held "Blink", and is written in Arduino. When you wait for several seconds, I'd see RX on the board and an LED of TX twinkling.

When upload works, you indicate "Writing in to a microcomputer board has been completed." in a status bar.

pin13 LED on the board will begin to blink several seconds later when upload has ended.

When it's so, it's success! You could write a program in Arduino by this and make now them move.

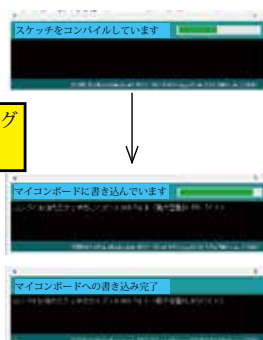
・アップロードがうまくいかない場合は、コンパイルの後、マイコンボードに書き込みが始まる前に、RDC-103 コントローラの RESET スイッチを「ダブルクリック」してください。



このタイミングです。



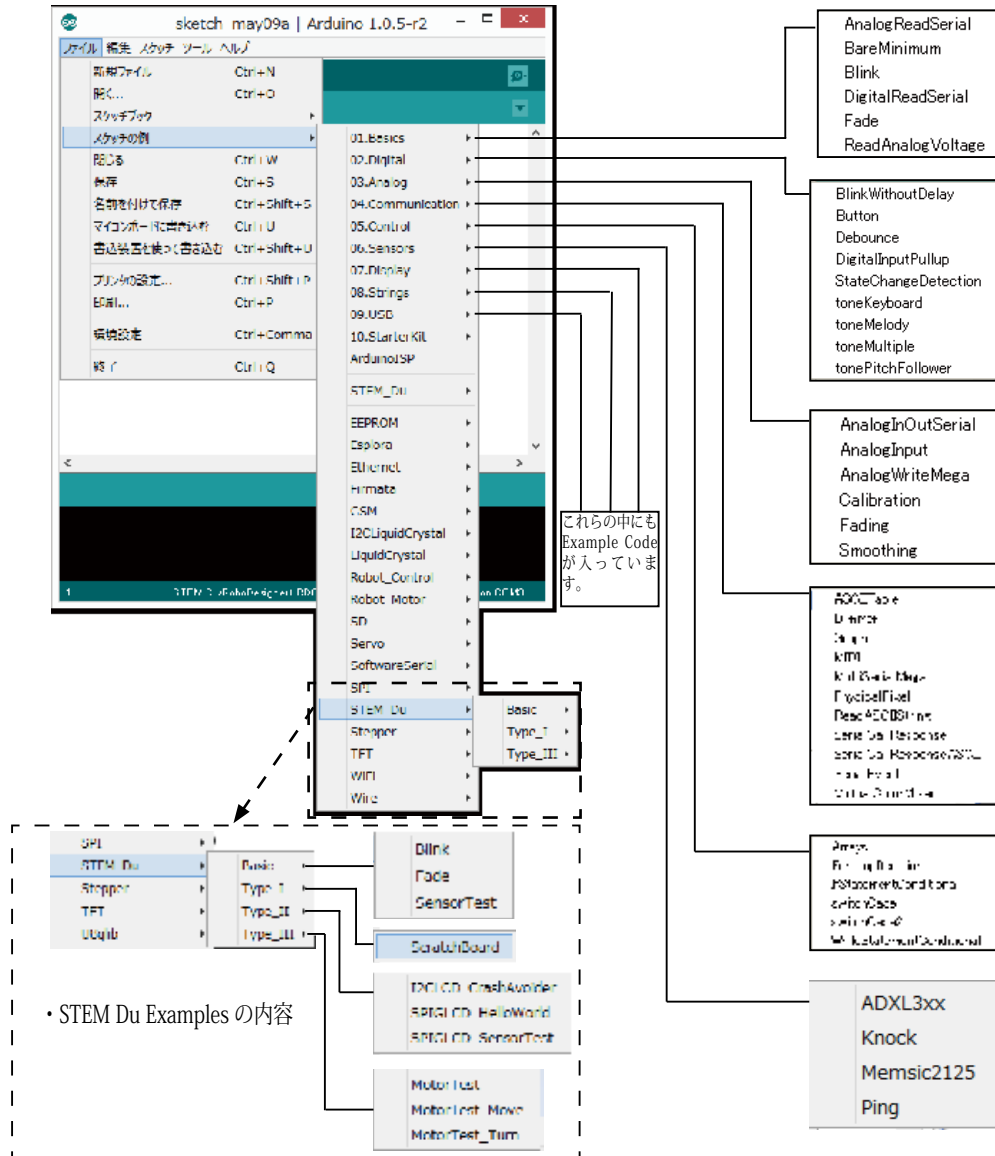
↑ RESET



\* When upload doesn't work, you double-click the RESET switch of RDC-103 controller.

### 4.5.3. Arduino Example Code (スケッチの例)

(1). [ファイル] > [スケッチの例] の中には、数多くのサンプルコードが準備されています。



• Examples の内容

- <http://arduino.cc/en/Tutorial/HomePage> [Arduino] > [Learning] > [Examples] で Example Code 個別の内容確認が可能です。

1. Basics

- **BareMinimum**: The bare minimum of code needed to start an Arduino sketch.
- **Blink**: Turn an LED on and off.
- **DigitalReadSerial**: Read a switch, print the state out to the Arduino Serial Monitor.
- **AnalogReadSerial**: Read a potentiometer, print its state out to the Arduino Serial Monitor.
- **Fade**: Demonstrates the use of analog output to fade an LED.
- **ReadAnalogVoltage**: Reads an analog input and prints the voltage to the serial monitor

2. Digital

- **Blink Without Delay**: blinking an LED without using the delay() function.
- **Button**: use a pushbutton to control an LED.
- **Debounce**: read a pushbutton, filtering noise.
- **Button State Change**: counting the number of button pushes.
- **Input Pullup Serial**: Demonstrates the use of INPUT\_PULLUP with pinMode().
- **Tone**: play a melody with a Piezo speaker.
- **Pitch follower**: play a pitch on a piezo speaker depending on an analog input.
- **Simple Keyboard**: a three-key musical keyboard using force sensors and a piezo speaker.
- **Tone4**: play tones on multiple speakers sequentially using the tone() command.

3. Analog

- **AnalogInOutSerial**: Read an analog input pin, map the result, and then use that data to dim or brighten an LED.
- **Analog Input**: Use a potentiometer to control the blinking of an LED.
- **AnalogWriteMega**: Fade 12 LEDs on and off, one by one, using an Arduino Mega board.
- **Calibration**: Define a maximum and minimum for expected analog sensor values.
- **Fading**: Use an analog output (PWM pin) to fade an LED.
- **Smoothing**: Smooth multiple readings of an analog input.

ここをクリックすると、リンクしている Code の個別説明に入ります。  
 • 使用方法・回路概要・Pin 説明・コード等必要事項がすべて公開されています。



#### 4.5.4. 使用例 Read Analog Voltage

##### (1). センサデータを取得します。

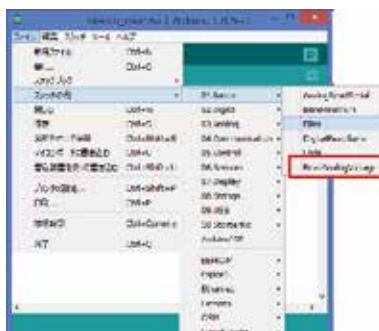
###### 1. Basic -ReadAnalogVoltage :

Reads an analog input and prints the voltage to the serial monitor  
を使用して、センサのデータを調べてみます。

[ファイル]▷[スケッチの例]▷[1. Basic]▷[-ReadAnalogVoltage]をArduinoで開きます。

###### 2. Example コードを変更しないでそ

のまま記載し、変更部分は青色コメントで追加していますので、各自で変更してください。変更したスケッチは別名で[名前を付けて保存]します。



#### Use example, Read Analog Voltage

##### (1). Sensor data is acquired.

1. Data of a sensor will be checked using "Reads an analog input and prints the voltage to the serial monitor".

• [File] ▷ [example of a sketch] ▷ [1.Basic] ▷ [-ReadAnalogVoltage] is opened in Arduino.

2. Example code isn't changed and it's mentioned just as it is, and a change part is adding a blue comment, so be respectively and please change it. Please name a changed sketch and preserve it.

```

Arduino IDE - ReadAnalogVoltage | Arduino 1.0.5 r2
ファイル 編集 スケッチ ツール ヘルプ
Program source code.
/*
  ReadAnalogVoltage
  Reads an analog input on pin 0, converts it
  to voltage, and prints the result to the serial
  monitor.
  Attach the center pin of a potentiometer to pin
  A0, and the outside pins to +5V and ground.

  This example code is in the public domain.
  */

// the setup routine runs once when you press
// reset:
void setup() {
  // initialize serial communication at 9600 bits
  // per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again
// forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  //(A0)を取得したいセンサーのPin番号に変更します。
  //A0 is changed to the Pin number of the sensor I'd like to acquire.

  // Convert the analog reading (which goes from 0
  // - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.println(voltage);
  //(voltage)を(sensorValue)に変更します。
  //voltage is changed to sensorValue.

}
1 STEM Du/RoboDesigner+ RDC-102 w/ ATmega32U4 3.3V 8MHz on COM3

```

3. The code of [-ReadAnalogVoltage] is changed according to its use destination.

A change point is the following 2 kinds.

\* The Pin number with which the sensor you'd like to acquire is connected

Sound sensor : A0 of controller loading

Light sensor : A4 of controller loading

Slider volume value : A5 of controller loading

The Pin number which was connected to connected optional sensor:- A1, A2, A3

\* Acquired data classification

Same data classification :sensorValue as "threshold value" to make a program reflected, which doesn't have to calculate substitution (Sensor Value is expressed by reduced property of 5V/1,023.)

4. After confirming [microcomputer board] and [serial port] by [tool], a made sketch is written in a microcomputer board.

3. 自分の目的に合わせて、  
[ReadAnalogVoltage] のコードを  
変更します。

変更点は下記の 2 種類です。

- 取得したいセンサーを接続して  
いる Pin 番号  
コントローラ搭載の音センサー: A0  
コントローラ搭載の光センサー: A4  
コントローラ搭載のスライダーボリューム値: A5  
接続した任意のセンサー: A1, A2, A3 等接続した Pin 番号
- 取得するデータ種別  
プログラムに反映させるため、置換計算し  
なくて済む「しきい値」と同じデータ種別:  
sensorValue (Sensor Value は、5V/1,023 の  
換算値で表示されます)

4. [ツール] で [マイコンボード]、[シ  
リアルポート] を確認し、通信可  
能に設定のうえ、作成したスケッ  
チを、マイコンボードに書き込み  
ます。

\*Arduino IDE のアップロードボ  
タン(→)アイコンを選択クリッ  
クすると、マイコンボードへの  
書き込みを開始します。

5.Arduino メッセージ「マイコン  
ボードへの書き込み完了しまし  
た。」がでると、書き込み成功  
です。

6. [ツール] ▶ [シリアルモニタ] をクリックし、実行します。

7. 下図画面の COM ウィンドウが表示され送られてくるデータ内容が表  
示されます。

※シリアルモニターを停止する場合には、マイコンボードから USB  
ケーブルを抜きます。抜いても COM ウィンドウ No 表示は消えま  
せんので、測定を停止後に、データの確認を行うことが  
できます。

8. データは早いスピードでカウントされます。10 秒で  
40,000 超のデータカウント数に及びます。データを個  
別で見ると変化を読み取ることが難しいので…→ 全部  
データを選択 [Ctrl]+[A]、あるいは、必要なデータ範囲を  
選択し、[Ctrl]+[C] キーを使用してコピー、Excel など表  
計算ソフトにペースト [Ctrl]+[V] して、グラフ化機能  
を使い、整理すると、見やすいデータとして使い勝手が良  
いでしょう。

\*USB ケーブルを抜いて、ロギングを停止してからコピー  
してください。

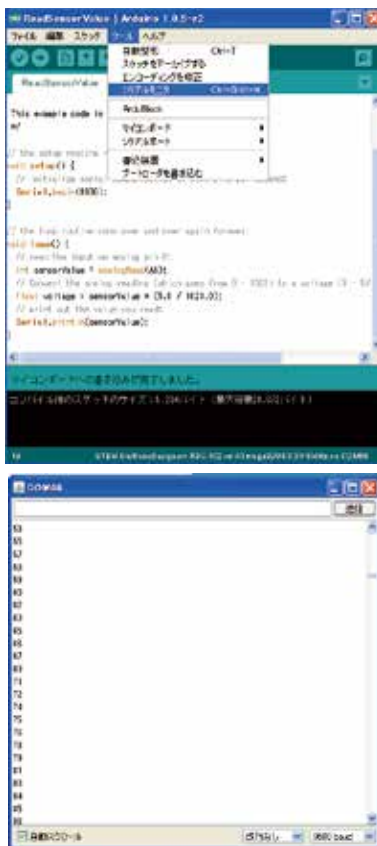
\*\* 右は、床にひかれた白線を測定したデータを、グラフ化した図で  
す。どれほどの大きさか一目で理解でき、「しきい値」の検討など  
に役に立てることが出来ます。

9. この機能を使用して、「音センサー」のデータを取得してみてください。



マイコンボード搭載の「音センサー」が、周囲の音を計測して  
いることが分かります。「話し声」や「手をたたく音」などを計  
測してみると、なるほど面白く実験できます。

シリアルモニターを長い時間継続すると、取得データがオーバーフ  
ローし、PC がフリーズすることがあります。  
このような場合、データ取得を中止し、コントローラのリセット、  
Arduino の再立ち上げを行ってください。  
お使いの P/C によっては、P/C の再起動が必要な場合もあります。



5. When a Arduino message "Writing in to  
a microcomputer board has been completed."  
goes out, it's writing in success.6. [Tool] ▶  
[serial monitor] is clicked and executed.

7. A COM window of the following figure  
screen is indicated, and the data contents  
which are being sent are indicated.

\* When stopping a serial monitor, a USB  
cable is removed from a microcomputer  
board.

\*Even if it's removed, a COM window  
doesn't go off, so after suspending  
measurement, it's possible to confirm the  
data.

8. Data is counted by the early speed.  
That comes to the number of data counts  
in which 40,000 per 10 seconds is exceeded.  
When data is seen separately, it's difficult  
to read a change.

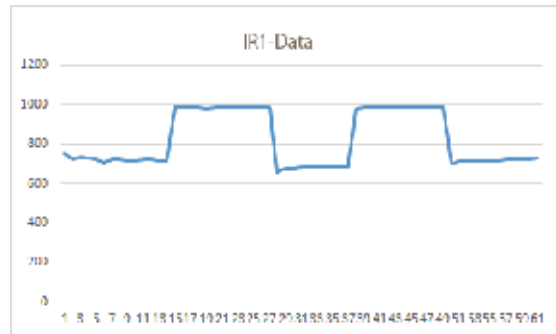
You'd be easy to use as the data you tend  
to think Excel chooses a necessary data  
area [Ctrl] + [C], and makes the copy paste  
spreadsheet software using a key, and to put  
it in order using the graphing function.

\* Remove a USB cable, and please copy after a  
log is suspended.

\* \*Below is a figure which graphs the  
data with which the white line attracted by  
a floor was gauged. It can be understood  
and is it possible to be able to be useful for  
consideration of "threshold value" by how  
much size or look?

9. Please acquire data of "sound sensor-"  
using this function.

You find out that "sound sensor" of  
microcomputer board loading is measuring



the surrounding sound.

That "voice" and "the sound with which a  
hand is hit" etc. will be measured, I see, you  
can experiment interestingly.

### PC freezes

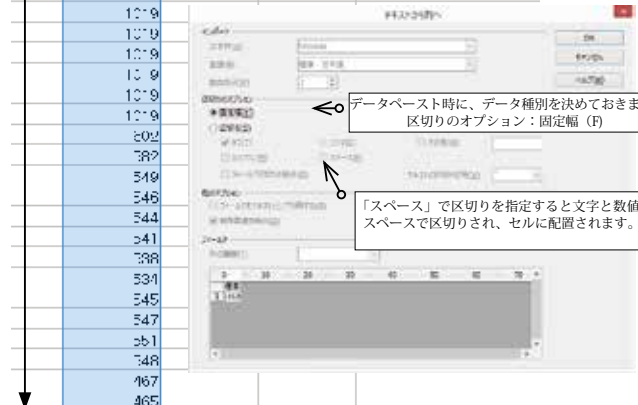
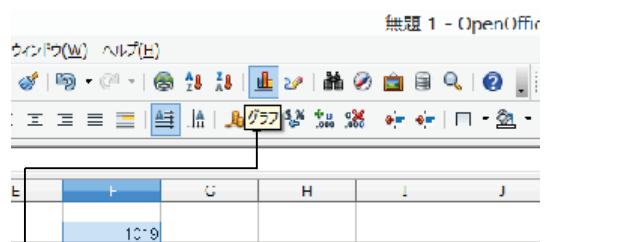
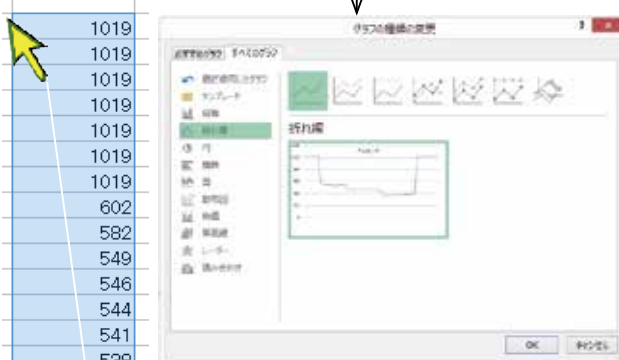
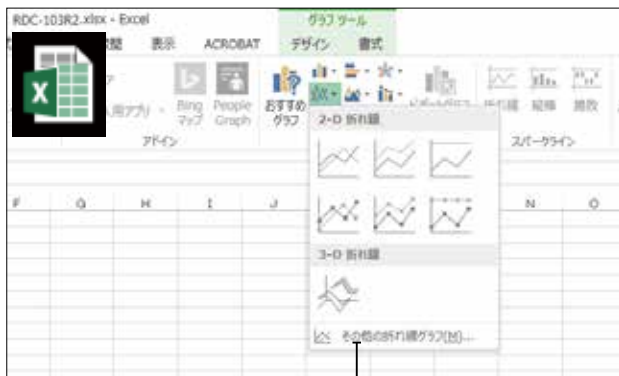
When a serial monitor is continued long  
time, acquisition data overflows, and a PC  
freezes.

Please cancel data acquisition in such  
case and restart a reset of a controller and  
Arduino.

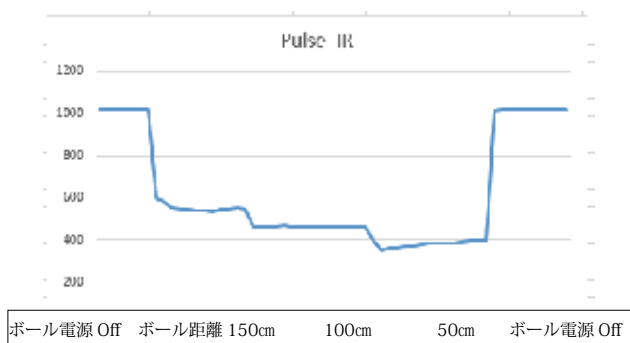
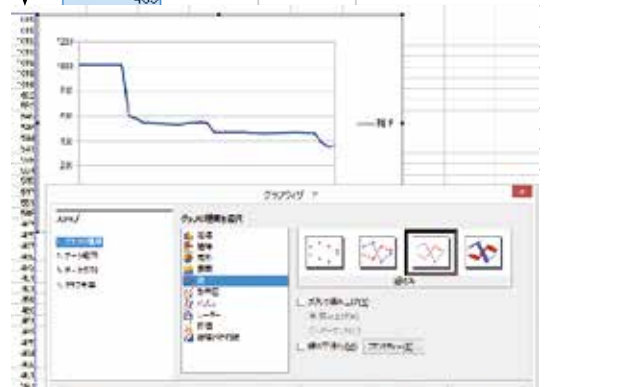
A restart of a computer is sometimes  
needed by the model you use.

### 4.5.5 表計算ソフト

1. シリアルモニタのデータは、1秒で4000個超のカウント数になります。プログラムの分岐条件に使用するしきい値は取得データを「表計算ソフト」などを利用してグラフ化し、分岐点を考察します。
  - 代表的な表計算ソフトとして **EXCEL**(有料ソフトウェア) があります。その他無料でインターネットから入手できるソフトウェアの例として **OpenOffice** もあります。



1. 以下の全部データか、一部データかの2通りのいずれかの方法でグラフ化します。
  - 測定したデータすべてを選ぶ場合はシリアルモニターウィンドウをクリックしアクティブにして [Ctrl+A] を押して、コピーします [Ctrl+C]。
  - 一部データを範囲指定する場合は、変化をグラフ化したい始点で左クリック、終点で右クリックし計測データを範囲指定して、コピーします [Ctrl+C]。
2. 貼り付けたい表計算ソフトの位置 (先頭のセル) を指定して、ペーストします。 [Ctrl+V]
3. グラフ化したいデータを範囲指定して、グラフ化処理をします。
4. グラフの形などは自分が分かりやすい形を選びます。



※シリアルモニタの出力を表計算ソフトで処理する場合は、変数の前のテキストをカンマやスペースに置き換えてください。OpenOffice の場合は、区切りのオプション「固定幅(F)」を選択指定してデータをペーストしてください。ペースト時に出現するダイアログのメニューで、「スペースで区切り」を指定すると文字と数値がスペースで区切りられセルに配置されます。

## 5. ロボットの仕組み

プログラムする、計測する、制御する    programmed, measure and control

### 5-1. RoboDesignerの構成

- \* ロボットの製作に入る前に、知っておかなければならないことがあります。まずは、どのような仕組みでロボットが動くかを学習しましょう。
- \*\*RoboDesigner はコントローラボードが中心となって構成されています。つまり、コントローラボードからの指令で各パーツが動くわけですが、どうやって指令の内容を決めればいいのでしょうか？まずは、それを理解しましょう。
- \*\*\*RoboDesigner には、プログラム開発環境 Arduino 及び ArduBlock 並びに Scratch が付属しています。まずは、それらのプログラム開発環境を用いて、指令の内容(プログラムのこと)を作成します。それを、コントローラボードに転送します。

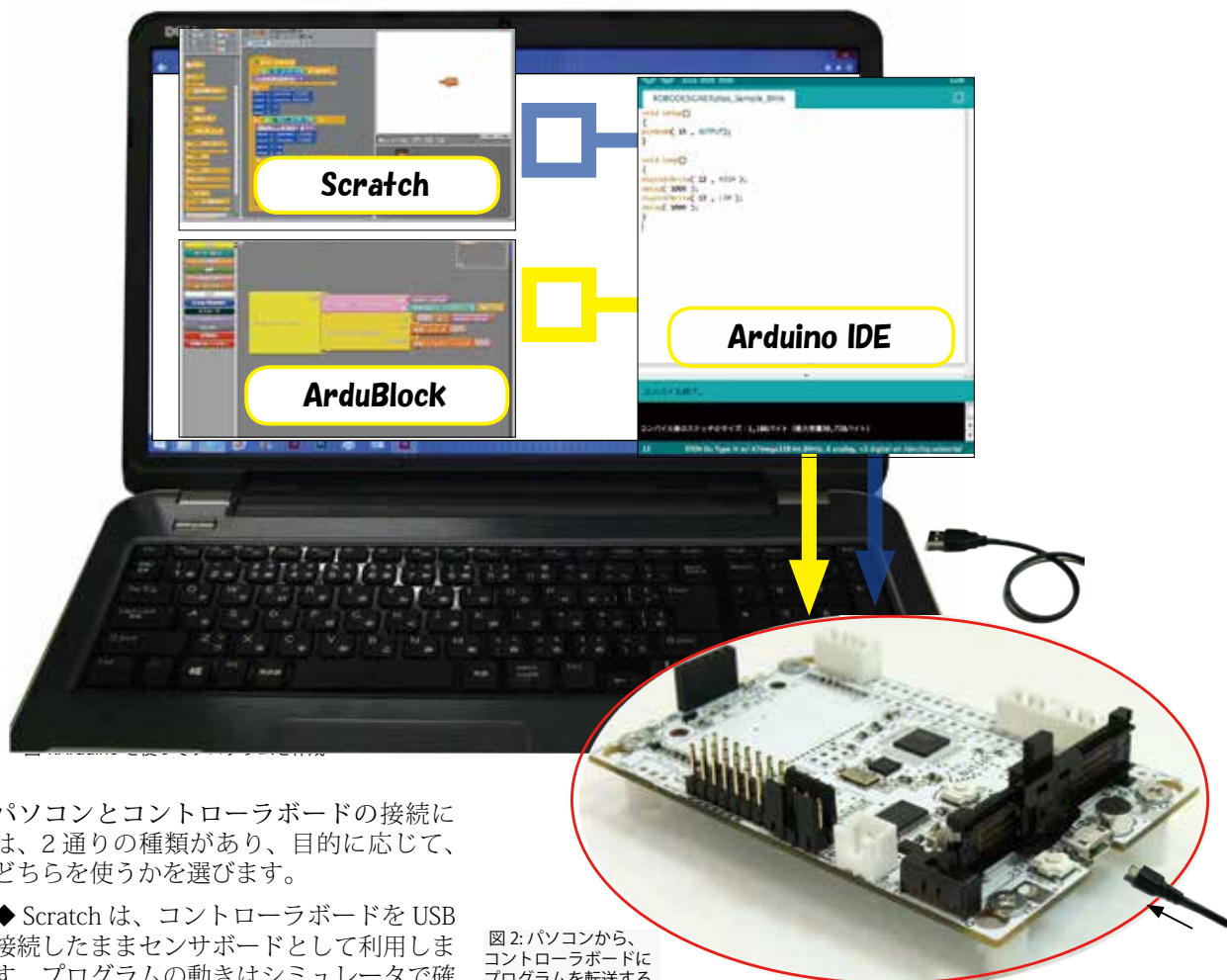
### The construction of RoboDesigner

\*Before entering making of a robot, We have to know. We'll learn by what kind of mechanism a robot moves first.

\*\*A controller board takes the leading part, and RoboDesigner consists of it. In other words, each parts are the reason which moves by an order from a controller board, but how should the contents of an order be decided? First, We'll understand that.

\*\*\*Program development environment Arduino, ArduBlock and Scratch attach to RoboDesigner.

First, the contents of an order (Program.) are made using those program development environments. That's forwarded to a controller board.



パソコンとコントローラボードの接続には、2通りの種類があり、目的に応じて、どちらを使うかを選びます。

◆ Scratch は、コントローラボードを USB 接続したままセンサボードとして利用し、プログラムの動きはシミュレータで確認できます。

◆ ArduBlock は、パソコンとコントローラボードの接続を外して、ロボットは自律して動けます。

このように、コントローラボードは独立で、プログラムにしたがって指令を出すこととなります。実際は、転送の際に、パソコンとコントローラボードとの間に USB ケーブルが中継することになりますし、コントローラには、センサからの信号も入ることとなりますので、少し違うのですが、自律型とはどのようなことなのかわかりましたか？

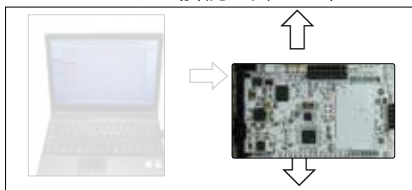


図3: コントローラボードから各パーツに指令が出る

図2: パソコンから、コントローラボードにプログラムを転送する

There are 2 kinds in a connection of a PC and a controller board, and it's chosen which to use according to the destination.

◆ Scratch can confirm the movement of the program used as a sensor board by a simulator while connecting a controller board.

◆ ArduBlock removes a connection of a PC and a controller board, and a robot does self-control, and can move.

Thus as a controller board is programed independently, instructions will be given.

A USB cable will report live between the PC and the controller board in case of transmission actually and a signal from a sensor will also enter a controller, so it's a little different, did you know what kind of thing an autonomous type was?

## 5-2. 実際の動作例

- 前項の説明は、概略的なもので、良くわからないでしょう。実際に、ロボットの構成を考えてみます。
- 例として、最初は前進を続け、接触式センサが反応したら後進するロボットを考えてみます。前進をするのですから、ギアボックス(+タイヤ)が2個、接触式のタッチセンサが1個必要になります。また、プログラムを転送するのですから、USB転送ケーブルも必要になります(実際に動かすときは外します)。
- モータへの電源として電池ボックスも必要です。  
\* コントローラボードだけであればUSBケーブルを通してパソコンから電源が供給されますので、基板は電気が入りませんが、大きな電力を使うモータを動かすときは電池からコントローラへの電源供給が必要です。その構成はこのようになります。
- では、信号の流れを順番に追っていきましょう。青矢印線が信号の流れです。まず、パソコンで作成したプログラムをコントローラボードに転送します。
- 転送が終了したら、USBケーブルは、コントローラボードから外してしまいます。転送時以外にケーブルがつながっていると、ロボットが動くときにケーブルが邪魔になり、ロボットの行動に制限がかかります。
- これは、パソコンとコントローラボードをつないでいるUSBケーブルを外すだけでOKです。では、コントローラボードに転送されたプログラムを実行します。
- プログラムが実行されると、まず、コントローラからギアボックスに前進の信号が送られて、ギア(タイヤ)が前進方向に回転します。これで、ロボットが前進します。
- ロボットがずっと前進を続けていくと、壁にぶつかって、タッチセンサのスイッチが入り、信号がコントローラボードに送られます。
- 壁にぶつかりタッチセンサから信号が入ったので、前後進が切り替わります。今度は、コントローラからギアボックスに後進の信号が送られて、先ほどとは逆の方向に車輪が回転します。
- このように、センサからの信号にあわせてロボットを自在に動作させることができます。ロボット製作には機体製作、センサ感度調整、プログラミング等の総合的バランスが必要です。トライ&エラーという言葉もあるように、繰り返し調整をしましょう。

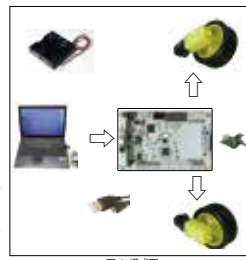


図4: 構成図

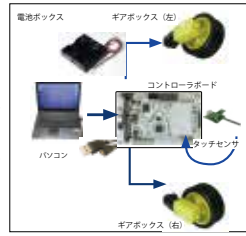


図5: コントローラボードにプログラムを転送

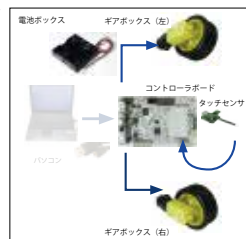


図6: USBケーブルを切り離す

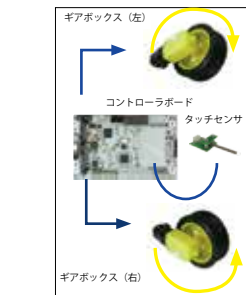


図7: ギアボックスに前進の信号を送る

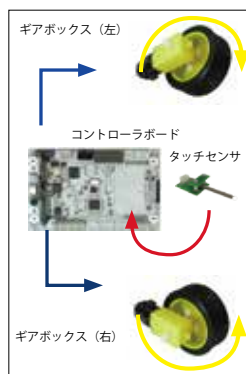


図8: タッチセンサからの信号の入力

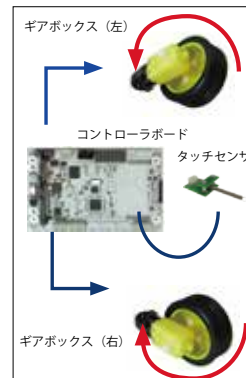


図9: ギアボックスに後進の信号を送る

### An actual movement example.

1. The explanation of the preceding clause is summarizing, and it wouldn't be understood well. Then, actually, the person who considered the construction of the robot may be easy to understand.

2. The advance is continued as an example, and if a contact type sensor reacts, the robot the younger generation does is made. Because a robot moves ahead, I need 1 of touch sensor by which gearboxes (+ tire) are 2 and a contact type. I also need a USB transmission cable for you to upload a program in a robot. (When moving it separately.)

3. A battery housing is also necessary as a power supply to a motor.

\*When it's only a controller board, a power supply is supplied from a PC through a USB cable, so electricity enters a substrate, but when moving a motor using the big electric power, power supply to a controller is needed from a battery. Its construction will be such feeling.

4. Then, we'll follow the signal trend in turn. A blue arrow line is a signal flow. First the program made by a PC is upload to a controller board.

5. If transmission ends, please remove a USB cable from a controller board. When a robot moves when a cable is connected besides the transmission time, a cable is annoying, and it takes restriction for behavior of a robot.

6. This just removes the USB cable with which a PC and a controller board are being connected, and is OK. Then, we'll execute the program uploaded to a controller board.

7. The signal which is an advance first in a gearbox from a controller when a program is executed, is sent, and a gear (tire) revolves in the advance direction.

A robot moves forward with this, doesn't it?

8. That a robot is keeping moving forward all the while, the switch of the touch sensor runs against the wall, and enters, and a signal is sent to the controller board.

9. A signal ran against the wall and entered at touch sensor, so the previous younger generation switches over. A developing signal is sent to the gearbox from a controller this time, and a gear revolves in the direction just now though.

10. Could almost all movement be understood? Thus you can make a robot move freely according to the signal from a sensor. There would be a lot of ones of not moving as I thought easily of course.

The overall balance which are airframe manufacturing, a sensor sensitivity adjustment and a programming, etc. in robot making is because it's necessary.

A challenge and failure are repeated as there is also a word as a try and an error.

We'll adjust it repeatedly.

## 6. 実際に作ってみよう(実験)

では、実際に作ってみます。しかし、いくら最も簡単にロボットが製作でき動かせるキットで、既に半分できていると言っても、初めての人には、

とても難しい!

初めての人は、何をしたらいいのかほとんどわからないことでしょう。

理由はいろいろありますが、

(1) プログラム作成ソフトのArduino に英語表記がある。

(2) 部品も英語表記がある。

などが挙げられます。もちろん、製品付属説明書にかかれている、「ロボットをつくらう」の通りに作れば動くロボットは製作できますが、理屈がわかっていなければ、それ以上の発展は望めないでしょう。

この章では、RoboDesigner の機能を順番に学習して、ロボットを自在に動かすとはどういうことかを勉強していきます。

6 章では、実験をしながら、少しずつ、ロボットの組み立て方や、使用方法を学習します。順番にやっていけば、必ずわかるようになりますので、がんばってやっていきましょう。

### 6-1. パーツの組み立てとプログラム転送

いきなり、ロボットを作っても、学習することが多すぎて、よくわからないでしょう。そこで、まず、一部の部品を使って実験装置を製作し、パーツの組み立て方を学習します。それから、ロボットを動かすプログラムを作成し、パソコンからプログラムを転送することを学習しましょう。これは、もっとも基礎的で、最も重要なことです。必ず最後までやりましょう。

実験目的: パーツの組み立て方、プログラムの作成、転送を習得する

実験部品: パソコン(Arduino-IDE がインストールされたパソコン)

コントローラボード(RDC-103)	× 1
ギアードモータ(RDO-502)	× 1
ユニバーサルシャーシプレート	× 1
電池ボックス(RDP-8093x4P)	× 1
マイクロUSB ケーブル	× 1
単3 電池	× 4
ビス(M3 × 6)	× 8
ビス(M3 × 10 で頭が皿)	× 2
ナット(M3)	× 6
樹脂スペーサ (M3x10)	× 4
(RDC-103 に取り付けてあるものです)	

工具:  
ドライバー  
ナット回し  
ラジオペンチ



### Then, We'll make. (Experiment)

Then, We'll make actually. But for the person how much is who for the first time even if a robot can be manufactured most easily, and he says that half is made of the kit which can be moved already

Very difficult!

The case that a first person knows what to do almost no. It's well-founded variously.

It can span product accessory instructions of course, "We'll make a robot.", when making a street, the robot which moves can be manufactured, but when not understanding a logic, We wouldn't have the chance of any more development.

It'll be studied whether it's what's thing to learn the function of RoboDesigner in turn and move a robot freely by the reason which says so while supplementing the part lack of the explanation on instructions by this text.

### Assembly of parts and program upload.

Even if a robot is made suddenly, it's often learned too much and we aren't known well.

So first We'll manufacture laboratory equipment using a certain part and learn how to put together parts, and We'll learn to make the program to which a robot is moved and forward a program from a PC.

This is most basic and is the most important thing. We'll do until the end certainly.



### 6.1.1. 組み立て時の注意点

RoboDesigner のパーツを使って、実験装置を組み立てます。組み立てるときにいくつかの注意点があるので、まずはそれから説明します。ここで何より最初に覚えて欲しいのは、

#### 1. 締めすぎたら、壊れる

ということです。ロボットのパーツは、薄いプラスチックでできていますから、きつく締めこんだら壊れてしまいます。ようするに、力加減が必要だということです。ゆっくりでいいので、力を加減して作ってください。壊れるくらいなら、締め込み不足でバラバラになった方がましです( また組み立てればいいことですからね)。これは常に注意しておいてください。また、電子部品ですので、

#### 2. 水でぬれた手で触るのも厳禁

配線がショートして、壊れる場合があります。汗かきの人は注意が必要です。ジュースを飲みながら作業するなどのもっての外です。もし、濡らしてしまって、動かなくなったら、急いで拭いてください。

拭いたらドライヤーの送風か、扇風機の風で乾かしてください。温風だと、パーツが壊れてしまいます。ちなみに、乾かしても動かなくなってしまうときは、最後の手段として、電池を外してから

きれいな水で洗う

水でぬれると壊れるといっても、実際にぬれた瞬間に壊れるのは、50%以下の確率です(ぬれた物にもよりますが)。水が配線の間に入ってショートしたり、さびたりするのが原因です。乾かしても、配線の中に水に含まれているゴミが残ってしまうので、結局ショートして動かなくなってしまうのです。そこで、きれいな水でよく洗ってから乾かすと、ゴミがとれるので動くようになることがあります。

ただし、最後の手段なので、覚悟を決めてやってください。

それから、人と季節によるのですが、

#### 3. 静電気にも注意

市販のICを使っていますので、それなりに静電気に対する耐性がありますが、あまりにひどいと壊れるときがあるので注意してください。

作業を行うときは、静電気を溜め込まないようにスリッパなどは履かずに、作業開始前に作業台を触って静電気を放出しておきましょう。

### 6.1.2. 実験装置の組み立て

まず、ユニバーサルシャーシプレートにそれぞれのパーツを設置します。どのパーツから付けていっても良いのですが、最初にモータから組み立てておきましょう。モータは部品が入っていますので、使用できるように部品の組み立てをまず行います。



組立てに必要なネジ一式が袋に入っています。

使用部品

- ギアードモータ 2 個
- タイヤホイール 2 個
- コネクタ付モータケーブル 2 本
- マウント金具 2 個
- 長ネジ M3x30mm 4 本
- ナット M3 4 個



組み立て完成図

### Careful point at the time of a system.

Laboratory equipment is put together using parts of RoboDesigner.

#### 1. If We finish too much, it breaks.

Because parts of a robot are made of light plastic, if it's tightened up tight, it breaks. In short it's said that they need the power degree.

It's slowly and good, so please moderate and make me the power. It's lack of tightening up and would rather be dispersively (It's because it's that it should be put together again.) rather than it breaks. Please be always careful about this.

#### 2. It's an electronic component, so touching by hand wet with water is also prohibited.

Wiring short-circuits and sometimes breaks. A sweating person needs attention. Such as working while drinking juice, it's outrageous. If it's wetted, and it doesn't move any more, please wipe it up quickly. If it's wiped up, please dry by the style of the ventilation of a hair dryer or the breeze from an electric fan, because a hair dryer is ventilation, please be careful. If it's warm breeze, parts break.

By the way, when having not moved any more even if it's dried, a battery is washed with clean water as the last means after I take off.

Even if I say that an electronic circuit breaks when I get wet with water, the moment I got wet actually, it's less than 50 % of probability that a controller board breaks.

Water enters during wiring, and it's done because of short-circuiting and rusting.

Even if it's dried, the trash included in water during wiring is left, so it short-circuits after all and doesn't move any more. So when it's dried after it's often washed with clean water, trash comes, so it starts to move. But, they're the last means, so please be ready.

#### 3. Please be also careful about static electricity.

Over-the-counter IC is used, so there is tolerance to static electricity to some degree, but when it's too terrible, there is time which breaks, so please be careful. Without putting on slippers so as not to amass static electricity when working, before beginning to work, I'll finger a workbench and release static electricity.

### Assembly of laboratory equipment.

### 6.1.2.1. ギアボックス組立て

ギアボックスを左右対象で作成します。

#### (1) マウンター取付

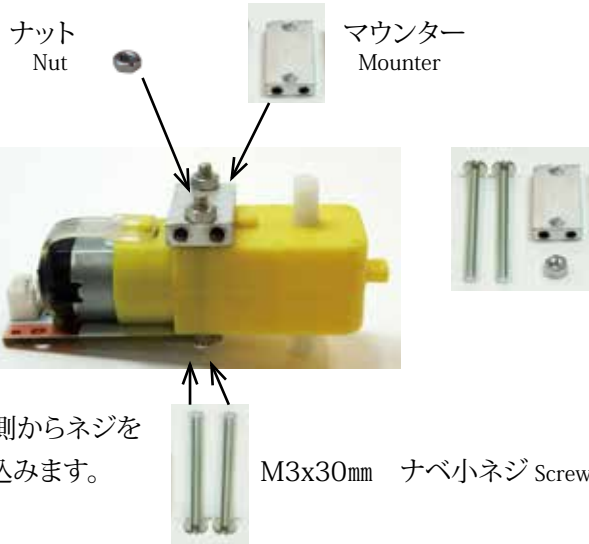
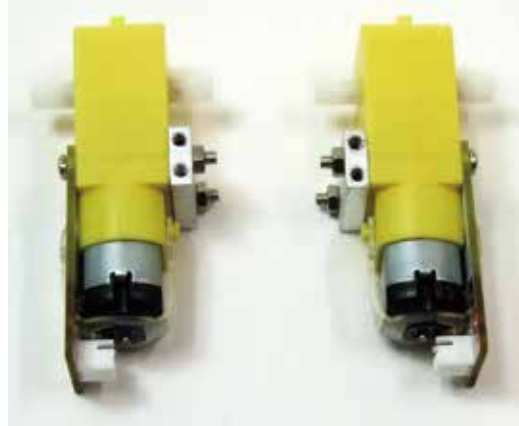
[Assemble of a Mounter]

ギアードモータにマウンターを取り付けます。

(検査のため取り付けている場合もありますが、その場合ナットを外して、マウンターの方向性を図に合わせ入れて替えてください。)

(図ではネジ穴が上を向いています)

マウンター金具には方向性があります。右図のネジ穴の方向を確認して取り付けてください。左右対称になるように作ります。



Down view



Side view

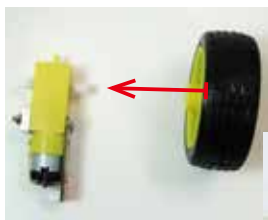


完成品: 2 個作ります。

Finished goods: 2 are made.

#### (2) タイヤホイールの組み込み

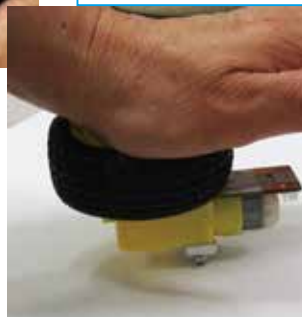
[Assemble of a Tire Wheel]



①モータ出力軸とホイールの差込口の長円形  
の方向を合わせます。

②回転軸を支えてホイールを差し込みます。

③最後に、図のように、出力軸の反対側を  
台に当てて、タイヤを上から手の平で押  
さえて差し込みます。



ギアボックス内部ギアに無理な力を加えないように注意ください。

#### 組み込んだホイールを取り外したい時



ギアボックスとホイールの間にラジオペンチの先を差し込み、ラジオペンチをゆっくりとこねてホイールを真上へ抜きます。

※手で無理やり、タイヤを曲げるとシャフトが折れて壊れます。

ホイールの組み込み図 Fig Wheel assembly



### 6.1.2.2. コントローラボード確認

コントローラボードRDC-103の裏を触ってください。 チクチクする  
 と思います。基板に部品が半田付けされたものは、部品の足が裏側に少し  
 はみ出しています。ここで重要なのは、  
 そのままユニバーサルプレートにつけると壊れることがある  
 ということです。無理やり締めこむと、チクチクの部分(基板裏面の突起)  
 が取付るプレートに押し付けられて基板が破損してしまいます。  
 そこで、チクチクの部分だけ(基板裏面突起の高さ) 隙間を空けるよう  
 に取り付けます。隙間の部分にスペーサを挟み込みます(青い矢印の部  
 分)。こうすると、スペーサが隙間を作ってくれるので、基板が壊れる  
 ことはありません(と言っても、締め込みすぎたら壊れてしまいます)。写  
 真では10mmスペーサをはさんでいますが、裏側についている部品など  
 で隙間が不足する場合は、スペーサを長くしてください。



図2: 使用工具 ⊕ドライバー ○ナットドライバー

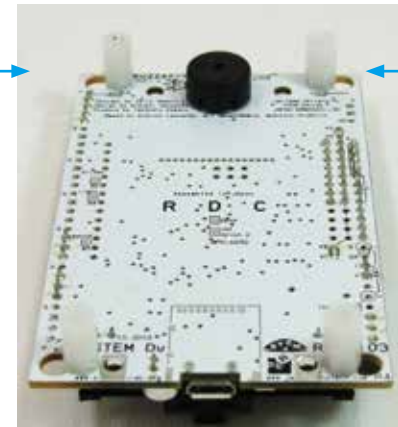
では、取り付けていきます。ネジの取り付けは、⊕ドライバーで行いま  
 す。スペーサがすべる場合は、ナットドライバー、又は、ラジオペンチで挟  
 むと便利です。

この要領で、写真の位置にある4箇所の穴にネジM3x6mmで樹脂スペー  
 サM3x10mmを止めてしまいます。緩んでいると、あとで外れたりしま  
 すが、あまりきつく締めこんでしまうと、基板が割れてしまうので、ほ  
 どほどにしてください。

4つの穴全部に取り付けたら終了です。



ネジ/スペーサ 取り付け



スペーサ取り付け例


### 6.1.2.3. 実験機組み立て

モータ回転実験時に実験機が移動しないように、タイヤを宙に浮かせた形  
 の構造物を作ります。

右の図を参考に組み立ててください。

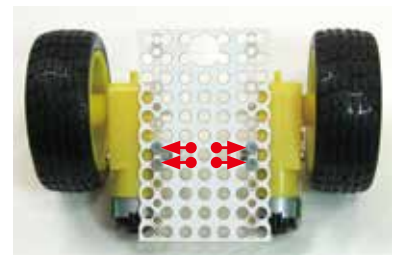
シャーシプレートへギアードモータ取付

1. 右図の赤矢線のシャーシプレート位置に、ギアードモータを超低頭ネジ  
 で取り付けます。(片方2か所、左右両方で4か所)

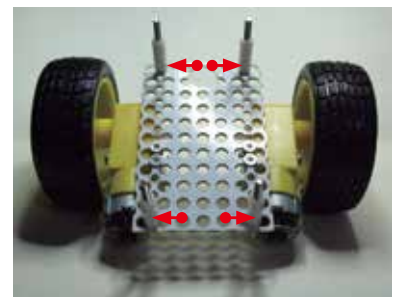
	使用部品	使用数
	超低頭ネジ M3x4mm Low Head Machine Screw	4本
*超低頭ネジは、シャーシプレートに同梱しています。 ※とても小さいので見逃さないようにしてください。		

2. 四隅に支柱を取付けます。

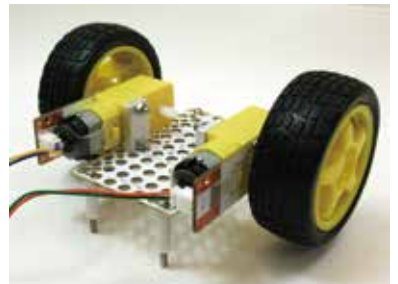
	使用部品	使用数
	樹脂スペーサー M3x20mm Resin spacer	4本
	ナベ長ネジ M3x30mm Long screw	4本



1. ギアードモータ取付位置



2. 支柱取付位置



実験は、支柱を床に向けて使い、ホイールは接地せず浮いています。

全体をレイアウトしてみましょう。電池ボックスは浮かせる必要が無いので、そのまま置いてみます。

### 6.1.2.4. 実験装置の配線

組み立てが完了したら、次は、実験装置の配線を行います。今回の実験装置で配線するのは3 箇所です。それに、パソコンからの配線も必要になりますので、合計で4 箇所の配線が必要になります。

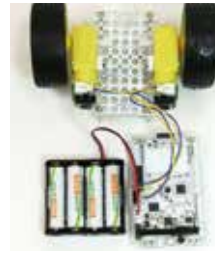
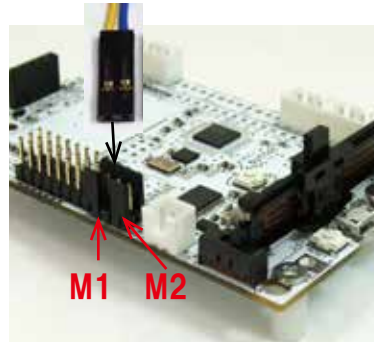


図11: 接続の概略図

線種	つなぐ場所	実際につなぐ線
橙緑線	左ギアードモータ ⇄ M1 コネクタ	ギアードモータに接続されているコード(2 本) 先端コネクタ黒
青黄線	右ギアードモータ ⇄ M2 コネクタ	ギアードモータに接続されているコード(2 本) 先端コネクタ黒
赤黒線	電池ボックス ⇄ V1 コネクタ	電池ボックスに接続されているコード(2 本) 先端コネクタ白
黒矢線	パソコン ⇄ コントローラボード	マイクロUSB ケーブル

#### モータコードの取り付け

表の上から順番に配線を行きましょう。まずは、左モータとコントローラボードのM1、右モータとコントローラボードのM2 コネクタを接続します。このモータは、逆につないでも逆に回るだけです。今回の実験には極性を気にせずに接続します。接続の方法は、ソケットにコネクタを奥まで差し込んで、固定します。



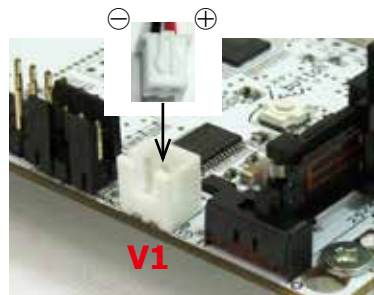
#### 電池ボックスからのコードの取り付け

次は、電池ボックスと、コントローラボードのV1 を接続します。ここで注意です。電源には⊕と⊖があるので注意してください。ソケット根元の基板に印刷しています。

電池ボックスの配線は、赤と黒です。赤をソケットの⊕に、黒をソケットの⊖に接続します。電源のコードを逆につなぐと、一発で壊れることがあるので十分な注意が必要です(このキットは逆につなげないようにコネクタ一式にして壊れにくいにはできていますが注意が必要なことに変わりはありません。)

逆につないだまま電池を装着し、長時間放置すると電源がOFF のままでも電池が発熱しますので注意して下さい。ひどい場合は電池の発熱により電池ボックスの樹脂が変形したり、電池が発煙することもあります。

逆につないだらどうなるのだろう?と、コネクタ部分を分解し、逆接続の実験などなさらないようにしてください。



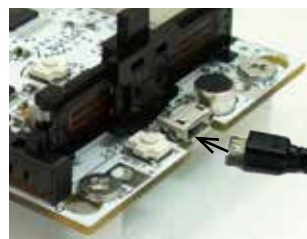
#### マイクロUSB ケーブル接続

パソコンとコントローラをマイクロUSB ケーブルで接続します。

パソコン側:ドライバーソフトのインストールを行った差込口にUSB ケーブルを接続します。

\* Windows の場合、USB ポートごとに、ドライバーソフトのインストールが必要です。常時使用するUSB 差し込み口にはシールなどで目印を貼り、その差込口に対してドライバーソフトのインストールを行っておきます。他の差込口を使うときには違うデバイスとしてWindows が認識し、さらにその差込口でのドライバーソフトのインストールが必要となりますので、注意してください。

以上で、実験装置の配線が終了しました。配線に間違いが無いのか、もう1 度確認してください。



#### Wiring of laboratory equipment

After your assembly has been completed, please wire laboratory equipment. You're this laboratory equipment and it's 2 to wire. And we also need wiring from a PC, so we need 3 points of wiring in total.

Installation of a cable from a battery housing

Next, V1 of a battery housing and a controller board is connected. It's attention here.

A power supply has ⊕Plus and ⊖Minus , so please be careful.

When a cable of a power supply is connected reversely, it breaks by blow, so enough attention is needed.

(We decide not to link this kit reversely in the connector system, and, it doesn't break, seem for, it's done, but a change doesn't give an instruction to a necessary thing.)

When a battery is loaded while connecting conversely, and it's left long, a battery is also exothermic by the condition by which a power supply is off, so please be careful. When being terrible, resin of a battery housing is transformed by fever of a battery.)

Wiring of a battery housing is red and black.

Red has been connected to ⊕ of a connector and black has been connected to ⊖ of a connector.

I suggest you connect conversely, will it be? Please hold a question, take a connector part apart and experiment on a reverse connection.

It breaks.

#### Wiring

If wiring ends, please confirm the color accurately.

Above, wiring of laboratory equipment has ended. In the making by which wiring of MicroUSB cable with which a PC and laboratory equipment are connected is a program, after We initiate, We'll do.

Please confirm whether it's without mistakes in wiring again.

### 6.1.4. プログラムの作成

では、次に、ロボットを動かすためのプログラムを作成しましょう。

RoboDesignerのコントローラボードは、小型のコンピュータを内蔵していて、そのコンピュータ用のプログラムを作成するわけですが、ここで問題が1つあります。そもそも、プログラムとは何でしょう？

1. プログラムとかプログラミングとかはよく耳にする言葉ですが、実際はどのようなものか説明されることは、少ないと思います。まず、プログラムを考える前に、1つ知っておかなければならないことがあります。そもそも、コンピュータとは、**魔法の箱ではない**
2. 大学関係者は、コンピュータではなく「**計算機**」と呼んでいます。なぜなら、コンピュータとは、与えられた式に従って、単純計算を超高速で行う機械に他ならないからです。すごく難しい計算を、あっという間に計算してしましますが、それは計算式を誰かが与えることが前提です。要するに、「○○○な計算を1億回繰り返せ」ということが簡単に行えるだけで、○○○の部分、誰かが考えないといけないのです。その、計算式のことをプログラムと呼び、プログラムを作ることをプログラミングといいます。簡単に言えば、「コンピュータにどうやって計算すればよいかを書いた内容」がプログラムなわけです。このプログラムは、大変重要で、はっきり言えば、機械の部分は多少いい加減でも動きます(動きや耐久性は悪いでしょうが)。しかし、プログラムを間違えると、まったく動きません。また、ロボットサッカーなどに出た場合、勝負を分けるのは、90%がプログラムです。どんなに高速で動けても、味方を攻撃するようでは、どうしようもないでしょう？ですから、プログラムの内容を良く理解し、何度も調整を行うことは大変重要なことなのです。
3. では、早速プログラムの作成、つまりプログラミングを行いましょ。RoboDesignerの場合、プログラムは専用の作成ソフトArduino-IDE, ArduBlock, Scratch の3種を使って作成します。これがあれば、難しいコンピュータ言語(プログラムを記述するための言葉)を覚える必要がありません。それだけでも、かなり簡単にプログラミングを行えます。まず、3種類のプログラム開発環境をパソコンにインストールしましょう。その手順は、IDE-DISKの中の「インストールガイド」の手順PDFを参照してください。インストールが終了したら、さっそく作成に移りましょう。

### Making of a program

Then, next We'll make the program to move a robot. A controller board of RoboDesigner has a small computer built in and is the reason which makes a program for the computers, but there is 1 problem with here.

What is a program?

1. A word as a programming is the word heard well, but We think it's little to explain what actual condition is. First before considering a program, We have to know 1. After all a computer isn't a box of magic.

2. The university person concerned is calling "calculated machine", not a computer.

A computer is because there are no other ones in the machine calculated at super-speed simply with the system to which it was given. Very difficult calculation is calculated suddenly, but it's presupposing that someone gives that an arithmetic expression.

In short someone can just do to say "○○○ repeat calculation 100,000,000 times." easily, and has to consider a part of ○○○ . The arithmetic expression is called a program. It's called a programming to make a program.

Briefly speaking, "the contents which described how to calculate in a computer" but the reason which is a program.

The part which is a machine this program is very important, and when saying clearly, a little, it's irresponsible, it moves. But when We make a mistake in a program, it doesn't move at all.

When having gone out to robot soccer,

90% is a program for ending in a draw.

Soit 's that it's very important to understand the contents of a program well and adjust it any times.

3. Then, We'll do making of a program in other words a programming right away. In case of RoboDesigner, a program is made using 3 kinds of exclusive making soft Arduino-IDE, ArduBlock, Scratch.

When there is this, it isn't necessary to remember difficult computer language (the word to describe a program). Only that can be programmed quite easily.

First We'll install 3 kinds of program development environment in a PC.

Please refer to procedure PDF of "installation guide" in accessory DISK for the procedure.

#### 6.2.1. Scratch is started.

1. In case of windows:

[WinScratch1.4-stemdu01] > drag and drop does and starts [Scratch4STEMDu. image] in [Scratch] to [Scratch.exe] (cat facial mark).

2. When it's Mac OS X:

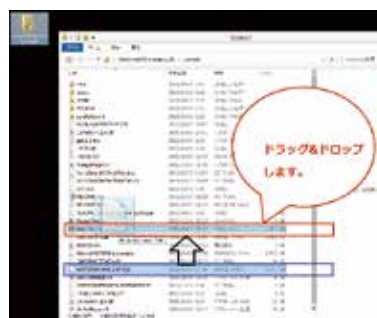
[Scratch4STEMDu.image] in arranged [Scratch\_14\_for\_STEM\_Du\_01] is double-clicked and started.

## 6-2. Scratchでプログラム実験

### 6.2.1. Scratch を起動する。

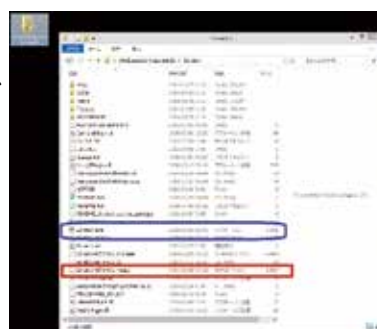
1. windows の場合:

[WinScratch1.4-stemdu01] --> [Scratch] の中の、[Scratch4STEMDu.image] を、[Scratch.exe] (猫顔マーク) へ、ドラッグ&ドロップして起動します。




2. Mac OS X の場合:

配置した [Scratch\_14\_for\_STEM\_Du\_01] の中の、[Scratch4STEMDu.image] を、ダブルクリックして起動します。



3. まずは、使用言語の設定をします。Scratch の画面構成を確認し、上部のメニューバーの言語設定のメニューで設定ください。

 のアイコンをクリックすると、[言語を設定する(Set language)]のサブウィンドウが現れます。50 種ほどの言語が準備されています。

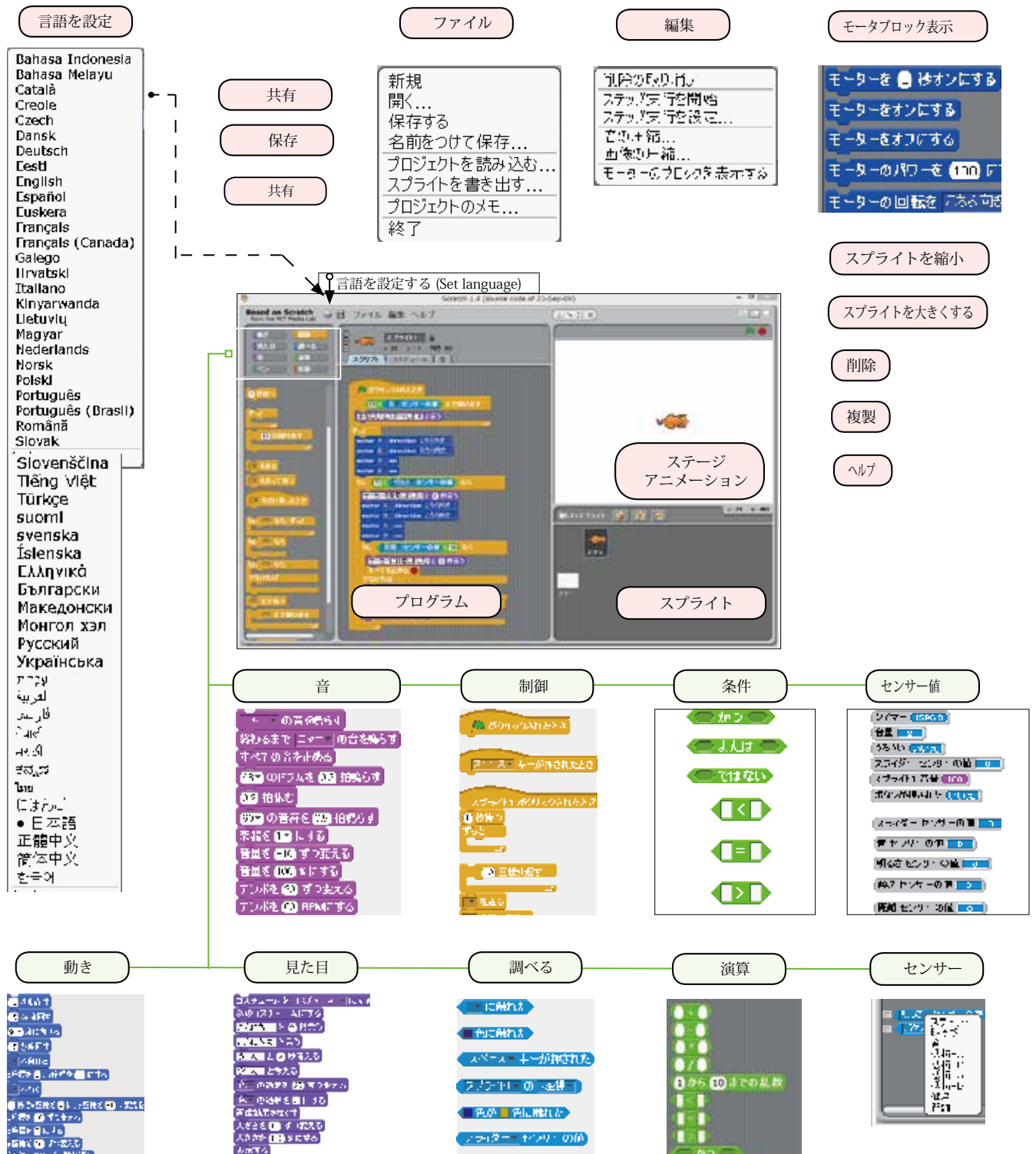
(日本語の場合、ひらがなだけの「にほんご」と、漢字を含む「日本語」の2通りの表記が選べます。)

3. First, a use language is established. Please confirm the screen structure of Scratch of the next page and establish it by the menu bar in the upper part of the linguistic setting of a menu bar in the upper part.

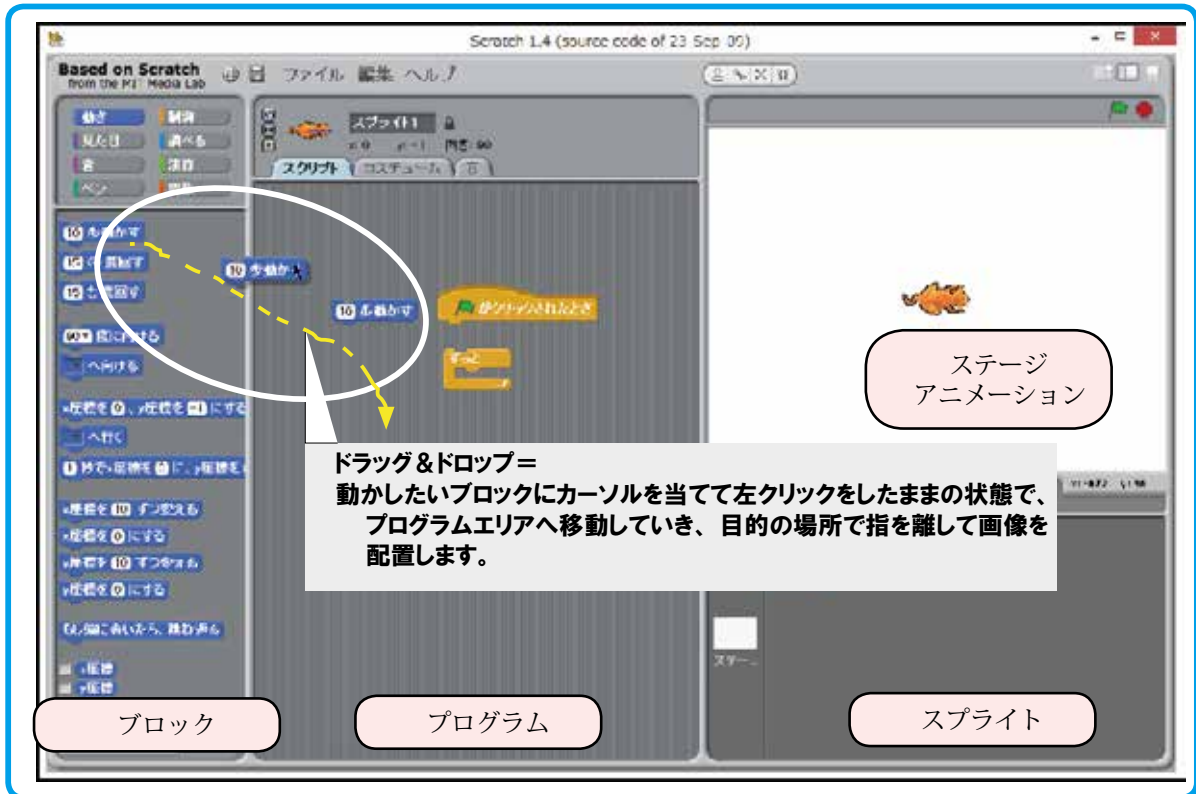
When an icon is clicked, a sub-window of [ (Set language) which establishes a language] shows. (In case of Japanese, transcription of 2 ways, “NIHON GO” only of hira-gana and “Japanese” including a kanji can be chosen.

### 6.2.2. Scratch 画面の構成

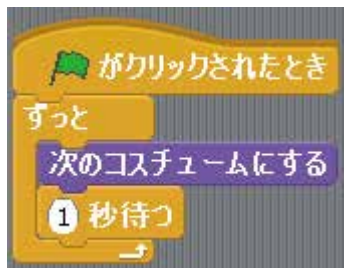
1. 準備されている各種のメニューと、スクリプト(プログラム言語の一種で「ブロック」と呼びます。)を確認してください。



6.2.3. プログラム作成



1. まずは、サンプルスケッチを開いてみます。[ファイル]▷[開く]▷[プロジェクトを開く]▷[例]をクリックし、出現するサブウィンドウの[Animation]の中に8種類のアニメーションが準備されていますので、どれかを選び、OK ボタンで確定します。
2. 選んだサンプルスケッチのプログラムが読み込まれてプログラムエリアに配置されます。
3. ブロック中の がプログラム実行アイコンです。



4. ステージの右上にある の緑旗も同じくプログラム実行アイコンです。となりの赤丸はプログラム停止アイコンです。
5. [実行][停止]を利用しプログラムを繰り返して、アニメーションを実行してみてください。
6. プログラムブロックの中の[1秒待つ]の数値を変更してみてください。マウスでポインタを移動し窓の白い部分の数字を指定しクリックすると、数字が変更できるようになります。数字を上書きしたら[Enter]キーを押して確定します。たとえば、[3秒待つ]などに上書き変更して実行するとアニメーションに変化が起ることが確認できます。
7. 他のサンプルスケッチも開いてみます。「ミャー」と声を出すスケッチもあります。
8. サンプルに手を加えて、独自の動きをするスケッチを作成してみましょう。完成したアニメーションは「発表モード」で全面表示で見ることが可能です。

プログラム作成にもう慣れましたね。

1. A sample sketching will be held. [File]-> [It opens.]->[A project is held.]-> 8 kinds of cartoon film is prepared in [Animation] of a subwindow which clicks [example] and appears, so something is chosen. OK, We fix by a button.

2. A program of a chosen sample sketch is read and arranged by a program area.

3. Blocked "green flag" is a program execution icon.

4. That there is a stage in the upper right, green flag is also a program execution icon. The next red circle mark is a program stop icon.

5. Please use [execution] and [stop], repeat a program and carry out a cartoon film.

6. In the program block is [Wait for 1 second.], please change the numerical value. When We move a pointer by a mouse and the partial number with a white window is designated and clicked, the number can be changed now. If a number is overwritten, We press a [Enter] key and fix.

7. Other sample sketches will also open. There are also "Mya-" and a sketch which raises a cry.

8. We'll make an improvement on a sample and make the sketch which does an original movement. It's possible to judge a completed cartoon film from full display by "announcement mode".

**It has been already accustomed to a program creation, has not it?**

### 6.2.4. コントローラボードと接続実験

Scratch では、コントローラボードを Sensor-Board として扱います。



#### 6-2-4-1. Sensor-Board の起動方法

##### (1). 【パソコンでの準備】

1. コントローラボード RDC をパソコン (PC) に接続します。
2. [PC] > [ハードウェア] > [デバイスマネージャー] > [ポート] で、COM 番号を確認します。



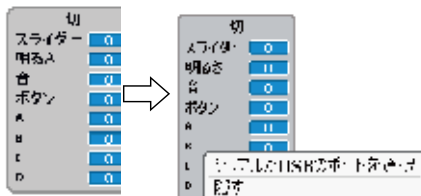
##### (2). 【Sensor-Board に設定】 Scratch のプログラムをコントローラで実行できるようにするために RDC ヘスケッチを書き込み、Sensor-Board に設定します。

1. [Arduino-IDE] > [スケッチの例] > [STEMDu] > [Type\_1] の [ScratchBoard.ino] を開きます。
2. [Arduino-IDE] > [ツール] > [シリアルポート] にて調べた COM 番号に設定します。
3. Arduino-IDE のアップロードボタン(→) をクリックし、マイコンボード (RDC) に書き込みます。
4. 書き込みに成功するとメッセージが表示されます。



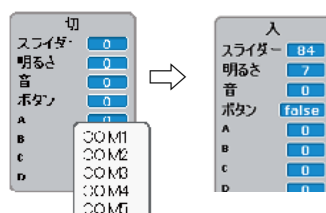
##### (3). 【Scratch での準備】通信設定を行います。

1. センサ-ブロックを選択し、右クリックすると、メニューが現れます。「ScratchBoard 監視板を表示」を選択し、クリック実行します。
2. ステージに「ScratchBoard 監視板」が出現します。



3. 「ScratchBoard 監視板」を右クリックし、「シリアルか USB のポートを選択」を選択すると、通信ポート (COM ポート) リストが現れますので、調べておいた COM 番号に設定します。

• 通信設定完了すると、「切」→「入」へ変化し、ボード搭載センサのデータ値が表示されます。



4. 使用準備ができましたので、確認実験です。
  - RDC のスライダーを左右へ動かしてみます。→監視板 / スライダー数値が変化します。

### Controller board and connection experiment.

A controller board is used as Sensor - Board in Scratch.

#### Initiation method of Sensor - Board

##### (1). [Preparations by a PC]

1. Controller board RDC is connected to a PC (PC).
2. [PC]---> [hardware], the COM number is confirmed by [device manager].

(2). [It's set as Sensor - Board.] Please write a sketch in RDC and set it as Sensor - Board because I'll can execute a program of Scratch by a controller.

- 1.[Arduino-IDE]--> [example of a scratch]--> [STEMDu]--> "the one of the Type\_1] please open [ScratchBoard.ino].
2. [Arduino-IDE], please set as the COM number which checked [tool] in [serial port].
- 3.Arduino-IDE upload key is clicked and it's written in a microcomputer board (RDC).
4. When You succeed in writing in, a message is indicated.

##### (3). [Preparations in Scratch] communication setting is performed.

1. When a sensor block is chosen and right-clicked, the menu appears. "Of a ScratchBoard watch board, indication" is chosen and a click is executed.

2. "ScratchBoard watch board" appears in a stage.

3.When "ScratchBoard watch board" is right-clicked and "of a serial or a port in USB, choice" is chosen, a communication port list shows, so please set it as the checked COM number.

\* When communication setting is completed, "off"--> changes into "on", and the data value of the sensor with a board is indicated.

4. Use preparations are done, so it's a confirmation experiment.

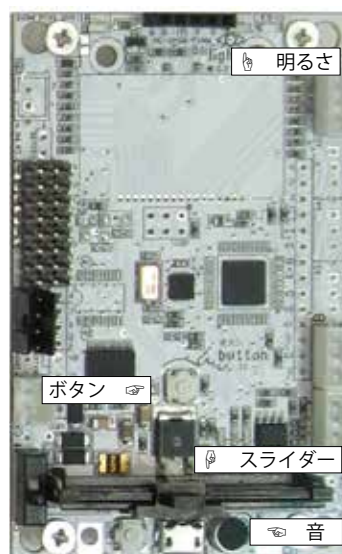
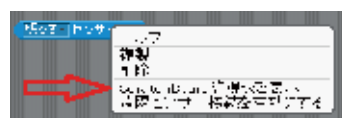
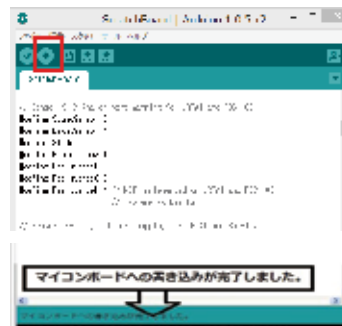
\* Please change slider-of RDC to left and right.

-> A watch board/a slider - a figure changes.  
\* Please apply light to a light sensor and change the strength.

-> A watch board/a brightness figure changes.

\* RDC is moving as ScratchBoard with this.

The sketch which includes motor control, Excuted, make, when, a connected motor begins to move, so please be careful so as not to drop it from the top of the desk.



- ・ライトセンサへ光を当てて強弱変化させてみます→監視板 / 明るさ数値が変化します。
- ・これで、RDCはScratchBoardとして動作していますので、モータ制御などを含むスケッチを実行させると、接続しているモータが動き始めますので、机の上から落としたりしないように注意します。

**6.2.5. スクリプト例を使用して動作実験**

- (1). ScratchProject の Example4-1 を開いてみます。
  1. ○○センサの値の3か所を「スライダ」に選択変更します。
  2. このスクリプトではスタートが「**緑の旗がクリックされたとき**」です、Scratch 画面ステージ右肩に配置されている「緑色の旗」をクリックして、スクリプトを実行します。
  3. 実行中は、周囲が白枠で囲まれます。
  4. RDC のスライダを動かして監視板の数値を観察してください。
  5. RDC の motor1 に接続しているモータの動きが「しきい値」を境にして、プログラム通りに回転を変化させながら動くことが確認できます。



**6.2.6. Scratch では、接続した状態のまま、マイコンボード RDC が動作します。**

1. Scratch では、常にパソコンと RDC を接続して使用します。
2. RDC のパソコン接続を外して、自律型動作をさせるときは、ArduBlock をご利用ください。

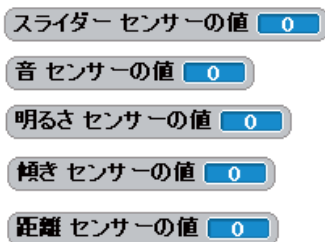
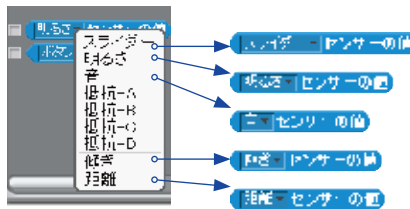
**6.2.7. Scratch 使用時のロボットで多く使うスクリプト**

- (1). モーター
  1. モータのブロックは、[編集]▷[モータのブロックを表示する]をクリックします。
  2. ブロックパレットの「動き」リストに追加されます。
  3. モータは A,B の2種類を使えます。
  4. Scratch と RDC のモータ端子

Scratch	RDC
motorA	M1
motorB	M2



- (2). センサー
  1. センサーは下記の5種類を選ぶことができます。
  2. センサ - ブロック左側の□にチェック入ると、センサー値を調べるマークが、右側のステージに配置されます。



3. 接続している RDC の計測値が表示されます。
  - ・この値を、「しきい値」分岐条件の参考とします。

**Experiment on movement using a script example.**

- (1). Example4-1 of ScratchProject is read. During carrying out, the environment is surrounded with a white frame. Please change a slider of RDC and observe the numerical value of the watch board. A movement of the motor connected to motor1 of RDC can confirm the thing which moves while changing a revolution into a program street on reaching "threshold value".

**You can experiment on movement using a script example.**

- (1). Example4-1 of ScratchProject is read.
  1. Choice change 3 points of sensor value to "slider", please.
  2. A start is a green flag by this script, so the "green flag" arranged by a Scratch screen stage right shoulder is clicked and a script is carried out.
  3. During carrying out, the environment is surrounded with a white frame.
  4. Please change a slider of RDC and observe the numerical value of the watch board.
  5. A movement of the motor connected to motor1 of RDC can confirm the thing which moves while changing a revolution into a program street on reaching "threshold value".

**A condition of the connected position and microcomputer board RDC move in Scratch.**

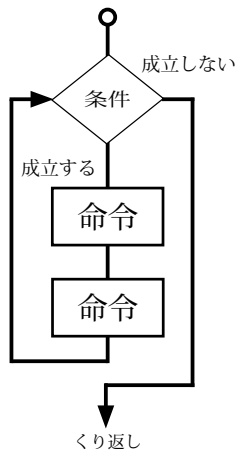
1. A PC and RDC are always connected and used in Scratch.
2. When removing a PC connection of RDC and making them do autonomous movement, please use ArduBlock.

**The script which is used much by the robot which is at the time of Scratch use**

- (1). Motor
  1. [Edit] clicks ▷ [A block of a motor is indicated.] in a block of a motor.
  2. It's added to the "moving" list of block palettes.
  3. A motor can use 2 kinds, A and B.
  4. Motor terminal for Scratch and RDC
- (2). Sensor
  1. A sensor can choose the following 5 kinds.
  2. When □ on the sensor - block left side can be made a check on, Mark who checks the sensor value is arranged in a right stage.
  3. A measured value of connected RDC is indicated.
    - \* This numerical value is made reference of the "threshold value" condition.

6.2.8. 例題

(1) 周囲が明るくなったら、動いて回るアニメーションを作りなさい。



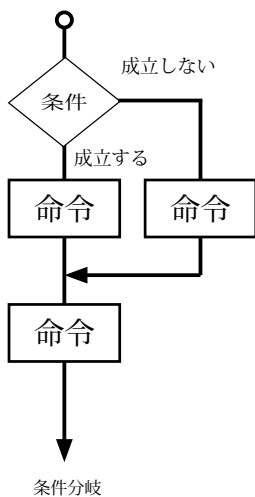
1. 全体として、「動いて回る」という設問ですから「ずっとくり返し」で構成します。
2. 「明るくなったら」の条件設定ですので、「明るさセンサー」を利用します。
3. 次に、「明るくなったら」の条件を満たすために「もし〇〇ならこうする、でなければあ～する」プログラム作成を考えてみます。
4. 分岐プログラム周囲の明るさが分岐条件ですので、コントローラボード搭載している「明るさセンサー」を利用します。
5. では、以上のヒントで考えてください。

Exercise

(1) if entourage become cheerful, make the cartoon film which moves and goes around.

1. Because it's the question to which you say "It moves and goes around." as the whole, compose by "repetition".
2. "If it becomes light." it's condition setting, so use "brightness sensor".
3. Next "If it becomes light." in the purpose which meets the condition, "If it's A, 1 is done, or, 2, it's done.", please consider a program creation.
4. The brightness around the branching program is a branch condition, so use "brightness sensor" equipped with a controller board.
5. Then please think by the above mentioned hint.

(2) コントローラ上のスライダの位置を変更すると、行動に変化を起こすアニメーションを作りなさい。



1. スライダーの位置により、行動に変化を起こすプログラムですので、「スライダー」を利用します。
2. 「スライダーの変化値」を Scratch Board 監視盤を使って計測します。
3. 計測できた値を「表計算ソフト」などを使い、変化量をグラフ化します。
4. 作成したグラフに基づき、変化を起こさせたい位置を、「プログラムのしきい値」として決めます。
5. 「しきい値」を条件値としてプログラム化し、動きが変化するプログラムを作成します。

(2) when the location of the slider on the controller is changed, make the cartoon film which brings about changes in behavior.

1. It's the program which brings about changes in behavior every the place by the slider, so "slider" is used.
2. Please measure "the change value of the slider" using Scratch Board watch board.
3. The numerical value which could be measured, please graph the change amount using "spreadsheet software" etc..
4. Please make the point of view you'd like to make bring about changes "threshold value of a program" based on a made chart.
5. Program "threshold value" as the condition value and make the program from which a movement changes.

(3) では、同じ条件時に、モータの動きが逆回転するプログラムを作成してみましょう。

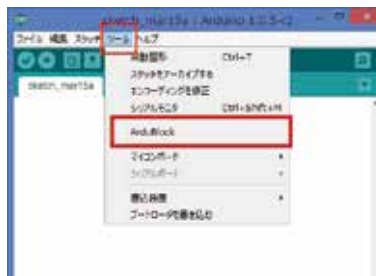
(3) Then, please make the program by which a movement of a motor backlashes at the time of the same condition.



## 6-3. ArduBlockでプログラム実験

### 6.3.2. ArduBlock を起動する。

1. Arduino-IDE をインストールした PC を準備します。
2. 使用 PC と実験機のコントローラボード RDC をマイクロ USB ケーブルで接続します。
3. 使用 PC のデスクトップに、[arduino-exe ショートカット] が配置されているはずですので、それをクリックすると、Arduino が起動します。
4. IDE の [ ツール ] → [ ArduBlock ] を選択し、クリックします。



右の画面が、立ち上がった ArduBlock です。

### ArduBlock is started.

1. Please prepare the PC in which Arduino-IDE was installed.
2. Connect controller board RDC of a use PC and an experimental unit by my black USB cable.
3. [arduino-exe short cut] should be arranged by a desktop of a use PC, so when that's clicked, Arduino starts.
4. [Tool] of Arduino-IDE-> Please choose and click [ArduBlock].

The left screen is ArduBlock which stood up.

### 6.3.3. ArduBlock プログラム作成 [1] モータを回転させてみる。

ブロックを配置してみましよう。今回、はじめに作成するのは、永遠にモータを正回転方向方向に低速で回転させるプログラムです。その配置は次のようになります。

1. 左列アイコンパレットの一番上にある { **制御** } をクリックした時に出現するサブウィンドウから [ **ずっと (メインループ)** ] を選択し中央のプログラムフィールドにドラッグ&ドロップします。
2. 左列アイコンパレット { **STEM Du** } をクリックした時に出現するサブウィンドウから [ **モータ** ] を選択し中央のプログラムフィールドにドラッグ&ドロップします。
3. ドロップしたばかりの [ **モータ** ] ブロックは M (モータ) 1、スピード 255 のパラメータになっています。



- モータは実験機で接続した M1 を使いますので、1 のままです。
- スピードは、最大に速いモータスピードが PWM 値 255 ですから、半分くらいの回転スピードで実験するために 130 と書き換えてみましょう。数字を範囲指定して上書きし ENTER キーで確定します。



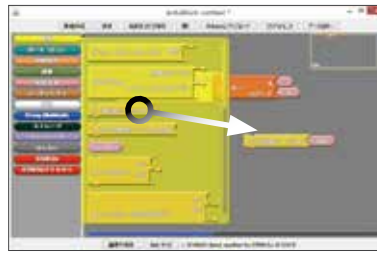
### We'll make a ArduBlock program creation [1] motor revolve.

We'll arrange a block. To make it this time and first?

The program which makes a motor revolve at low speed in the positive direction of rotation direction eternally. The arrangement starts to be the next.

1. When clicking { **Control** } of the left side icon palette, you designate [ **repeated (main loop)** ] from the subwindow from which you emerge, and please do drag and drop in a central program field.
2. When clicking the left side icon palette, { **STEM Du** }, you choose a motor from the subwindow from which you emerge, and do drag and drop in a central program field.
3. The [ **motor** ] block which has just dropped is a parameter of motor 1, speed 255.
  - \* M1 connected by an experimental unit is used, so a motor is a condition of 1.
  - \* The speed is biggest, because the fast motor speed is PWM value 255, please rewrite with 130 to experiment with the revolving speed which is about half.
 Please specify the range of a number, overwrite and fix by an ENTER key.

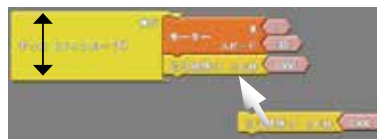
4. 1で配置した「ずっと(メインループ)」ブロックのパーツ位置に収まるようにドロップすると「カチッ」と音がして、タイルが、ループにはまり込み結合します。



5. 次にアイコンパレット { **制御** } をクリックしたときに現れるサブウィンドウから「ミリ秒待つ」を中央のプログラムフィールドにドラッグ&ドロップします。



6. 「ミリ秒待つ」を、3で「モータ」を結合させた「ずっと(メインループ)」ブロックのパーツ位置に追加で収まるようにドロップするとループ枠が広がり「カチッ」と音がして、追加ブロックが、ループにはまり込み結合します。



\* ブロックをドロップするたびに、ループ枠が大きくなっていきます。ループ枠はプログラムの大きさに合わせて変化します。

7. 今、{「モータ 1 をスピード 130 で 1000 ミリ秒回す」ことを、「ずっと繰り返す」} プログラムが完成しました。

**プログラムは意外と簡単ですね。**

\* モータは何番を使うか、スピードはどの速さで回すか、繰り返す時間は何ミリ秒とするかなどのパラメータ設定を調整し、自分の思い通りに動くように調整することが重要な要素です。

4. That it drops so that it may be satisfied with 1 in the location of parts of the arranged [repeated (main loop)] block, I hear sound, and a tile fits into a loop and is crowded, and combines noise with "Cachi".

5. From the subwindow which shows when you clicked an icon palette **Control** in the next, [A millisecond, wait.] Please do drag and drop in a central program field.

6. [A millisecond, wait.] when it drops so that it may fit into the location of parts of the block which made [motor] combine [repeated (main loop)] additionally, a loop frame spreads, and 3 makes a noise with "Cachi", and an additional block telescopes in a loop and combines.

\* A block, every time it drops, a loop frame is becoming big. A loop frame changes according to the size of the program.

7. {It's repeated to dial motor 1 for 1000 milliseconds in speed 130.} a program has been completed.

**That the program is unexpected, it's easy, isn't it?**

\* It's an important element to adjust it as a motor adjusts parameter setting of how many milliseconds to set time to repeat by which speed you turn the speed to to what number to use, and may move to its concerned street.

### 6.3.4. プログラムアップロード前準備

#### 1. 出力軸旗立

・モータを回す実験をしますが、モータだけの回転では回っている状態の確認が難しいので、実験機ギアボックスの出力軸(タイヤ側面)にラベルなどを利用して印を付けておきます。(少し色付きのラベルを利用した方が回転が見えやすいかもしれません)



#### 2. 通信ポート確認

・Arduino-IDE の [ツール]▷「マイコンボード」設定の確認、「シリアルポート」の接続 COM 番号設定をします。

・この段階で、「マイコンボード」に●マーク、「接続 COM 番号」に✓マークがついていることを確認してください。ついていないと通信ができませんので、マウスを使って該当箇所をクリック指定してマークを付けてください。

・この手続きを、必ずしてください。しないとプログラムのアップロードができません。エラーになります。



#### 3. 作成プログラム保存

・[保存]か[名前をつけて保存]のいずれかで、作成したスケッチ(プログラム)を保存します。

※保存したプログラムは、いつでも読み込むことが可能です。



Preparations before program upload

#### 1. A flag is put on the output shaft(wheel).

\* The experiment to which a motor is turned is done, but confirmation of the state that only a motor is running by a revolution is difficult, so in an output shaft of an experimental unit gearbox, using a color tape, a flag is put.

#### 2. The communication port is checked.

\* Confirmation of [tool]--> "a microcomputer, board" setting of Arduino-IDE and COM number setting of "serial port" are done.

\* At this stage.

● mark "microcomputer board".

☑mark "the connection COM number".

Please confirm that there is a mark. When there isn't a mark, a communication line doesn't connect.

A click designate a relevant part using a mouse, and please mark.

**\* Please be sure to do this procedure. When it isn't done, you can't upload a program. It'll be an error.**

#### 3. Making program preservation

\* A made sketch (program) is preserved by one of [Save] or [Save As].

### 6-3-5. プログラムアップロード

1. ArduBlock の [Arduino に アップロード] をクリックし、作成したスケッチ (プログラム) を、Arduino へアップロードします。

・アップロードしたスケッチは、Arduino-IDE に「C++言語」で表示されて、コンパイルされます。

2. コンパイル後は、すぐにマイコンボードへの書き込みが始まります。

3. コントローラボード側がアップロードの受信をしていることを確認する方法は、アップロードの最中にコントローラボードの電源表示 (青色 ON) 以外の 2つの LED (RX 赤色 / TX 黄色) が点滅することで、確認できます。

4. アップロード受信が終了したら RX 赤色 / TX 黄色 LED の点滅が止まり、電源表示 (青色 ON) だけが点灯しますので、アップロード時はコントローラボードの LED を見て確認してください。

5. 成功すると「マイコンボードへの書き込みが完了しました。」と、下段のメッセージ欄に表示されます。

6. 通信不成功の場合は**エラーメッセージ**を赤色表示

Couldn't find a Leonardo on the selected port. Check that you have the correct port selected. If it is correct, try pressing the board's reset button after initiating the upload.

- ・マイコンボードを USB ポートへ接続していなかったり、(a) 通信ポート COM (No) 設定が間違っていたり、(b) ボードの動作環境が合っていないと、**通信エラー**が発生して、マイコンボードへの書き込みが失敗します。

[ツール]▷[マイコンボード]に●マーク「接続COM番号」に△マークがついていることを確認してください。

- A). Arduino メニューバーの [ツール]▷[マイコンボード] メニューから接続したい Arduino ボードの名前を選びます。(RDC-103 は、STEM Du/RoboDesigner+ RDC-102 w/ATmega32U4 - 3.3V 8MHz)
- B). Arduino メニューバーの [ツール]▷[シリアルポート] メニューから接続したいポート番号を選びます。Windows の場合は COM3 といったような名前になっていて、数は 3 以上の場合もあります。
- C). 「2-4. ドライバインストール」の項を参照して適切なドライバーをセットください。

7. プログラムタイピングミスの場合など、Arduino の該当行が**黄色ハイライト**表示で警告されます。

・プログラムで使用できる文字は「半角英文字」と「半角数字」のみです。全角文字は、プログラム文ではエラーとなり、該当行が黄色マークで警告されます。

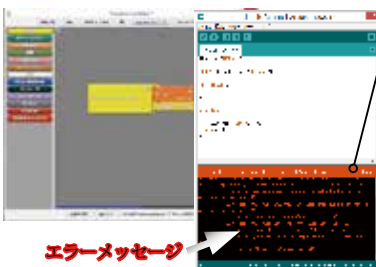
- ◆エラーメッセージを確認して、対策を施して、問題を解決した後で、再度、マイコンへの書き込みを行います。

8. 書き込み完了後、すぐにマイコンボードはプログラムが実行されます。

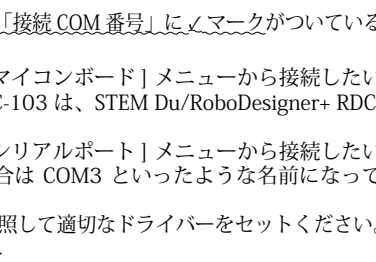
- ・今回のプログラムは、「モータ 1 をスピード 130 で 1000 ミリ秒回す」ことを、「ずっと繰り返す」プログラムでしたので、すぐに M1 に接続している実験機モータが回転を始めます。
- ・接続した実験機のギアボックスは回転しましたか？
- ・先ほど、タイヤ側面にラベルなどを利用し印を付けるように準備しましたので、回転方向も含めて回転している状態が確認できると思います。

9. USB ケーブル接続時には、電源供給は USB ケーブルを介して PC から行われていますので、停止させたい時には USB ケーブル接続を外します。

・電池で動作させている場合は、電源スイッチを OFF にします。



エラーメッセージ



### Program upload

1. Please click [to Arduino, upload] and upload a sketch (program) to Arduino.

\* An uploaded sketch is shown to Arduino-IDE by "C language", and is compiled.

2. After compiling, writing in to a microcomputer board will start immediately.

3. The way to confirm that the controller board side is receiving upload is that 2 LEDs besides the power supply indication of a controller board (blue on) (RX red/TX yellow) flash on and off during upload, and it can be checked.

4. If upload reception ends, a flash of a RX red /TX yellow LED stops, and only power supply indication (blue on) lights up. When uploading it, please see and check the LED of a controller board.

5. When it succeeds, you indicate "Writing in to a microcomputer board has been completed." in a message space in a lower berth.

### 6. Error message

Couldn't find a Leonardo on the selected port. Check that you have the correct port selected. If it is correct, try pressing the board's reset button after initiating the upload.

1) A microcomputer board isn't connected to a USB port.

(a) communication port COM (No) The setting is wrong.

(b) working environment of a board isn't right.

(c) then a communication error occurs, and writing in to a microcomputer board is failed.

A). Choose the name of the Arduino board you'd like to connect from the menu of the Arduino menu bar, [tool]▷[microcomputer board]. (For RDC-103, STEM Du/RoboDesigner+ RDC-102 w/ATmega32U4 - 3.3V 8MHz)

B). You choose the port number you'd like to connect from whole menu of the subwindow in which is a Arduino menu bar [tool]▷[serial port].

・It's COM3 and the name I needed in case of Windows.

C). Please refer to an item of "2-4. Driver installation" and set an appropriate driver.

2). After you confirmed the error message and did a measure, and settled a problem, you write notes in a microcomputer once again.

7. After writing in completion, a program will be executed by a microcomputer board immediately.

\* This program, [Of "Motor 1 is dialed for 1000 milliseconds in speed 130.", "it's repeated."], it was a program, so the experimental unit motor which will be connected to M1 immediately begins to revolve.

\* Did a gearbox of a connected experimental unit revolve?

\* I prepared as a tape was used for a gearbox output shaft and a flag was put a short while ago, so I think the state that I'm circulating including the direction of rotation can be confirmed.

### 6.3.6. 実験機でプログラム実験

1. 前項で、{「モータ 1 をスピード 130 で 1000 ミリ秒回す」ことを、「ずっと繰り返す」} プログラムを作りました。



タイルで作成したプログラムは、アップロードするとC言語に変換されます。

2. 今度は、モータを逆に回転させるプログラムに挑戦してみます。左列アイコンパレット { **STEMDu** } をクリックし、出現するサブウィンドウから [後進 (バック)] を選択し中央のプログラムフィールドにドラッグ & ドロップします。



3. [後進(バック)] を、前項で [モータ] を結合させた [ずっと (メインループ)] ブロックのパーツ位置に追加で収まるようにドロップするとループ枠が広がり「カチッ」と音がして、追加タイルが、ループにはまり込み結合します。



\* タイルをドロップするたびに、ループ枠が大きくなっていきます。ループ枠はプログラムの大きさに合わせて変化します。



「後進 (モータ逆回転) スピード 255」へ変更し、アップロード

4. 今、{「モータ 1 をスピード 130 で、後進 (バック) することを「ずっと繰り返す」} プログラムが完成しました。

5. ArduBlock の [Arduino にアップロード] をクリックし、作成したスケッチ (プログラム) を、Arduino へアップロードします。
  - ・アップロードしたスケッチは、Arduino-IDE に「C ++ 言語」で表示されて、コンパイルされます。上の 2 種類のプログラム比較用にコードを右に示しますので、後進 (バックワード) を加えたことによる C プログラムの変化も参考にします。

6. 先ほどは、モータスピードを 130 で設定しましたが、今回の逆回転時にはスピードは最大の 255 設定のままですので、モータの回転方向が逆になりギアボックスの回転スピードが変化したことに気が付きます。

7. Arduino-IDE にコンパイルされたコードを直接上書きしてプログラム変更することも可能です。実験的に数種類のモータスピードに変化させながら、プログラムを実験機にアップロードし、モータ回転の変化を確認しましょう。

STEMDU\_robot.motor(1,130) のアンダーライン部分がモータスピードです。0 ~ 255 の範囲での PWM 設定が可能です。(Pulse Width Modulation)

前進、後進を繰り返すプログラム例



・プログラムで使用できる文字は「半角英文字」と「半角数字」のみです。

8. A power supply is supplied the time of USB cable junction with from a PC through a USB cable. When I'd like to make them stop, USB cable junction is removed.

アップロードしたコード

```

Program source code.
// {「モータ 1 をスピード 130 で 1000 ミリ秒回す」
// ことを、「ずっと繰り返す」} プログラム
#include <STEMDu.h>
STEMDu_STEMDU_robot = STEMDu();

void setup()
{
}

void loop()
{
  _STEMDU_robot.motor(1,130);
  delay( 1000 );
}

// コンパイルしたソースコード
    
```

```

Program source code.
// {「モータ 1 をスピード 255」で、後進 (バック)
// することを「ずっと繰り返す」} プログラム
#include <STEMDu.h>
STEMDu_STEMDU_robot = STEMDu();

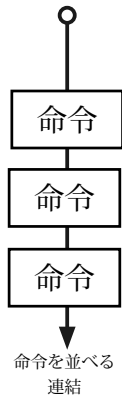
void setup()
{
}

void loop()
{
  _STEMDU_robot.motor(1,255);
  _STEMDU_robot.backwardM1M2(255);
}
    
```

◆ 実験機で動きの変化をよく確認しましょう。ロボットをうまく動かすコツがここにあります。

## 6-4. 制御文(ループ)

### 6.4.1 制御文とは



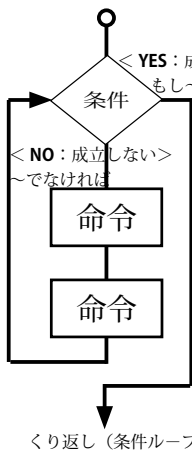
前節で学習したプログラムは、命令を並べていくもので、スタートから、終了まで一直線でした。最初のうちは、これでもよいのですが、同じことを 100 回繰り返すときは、100 個の動作を書くのでしょうか？100 個なら書いても、10000 回とか、永久(スイッチを切るか、電池がなくなるまで)に繰り返すとなると、とても無理です。また、「こういう場合はこう動くが、このような時は別の動きがしたい」という時も、一直線のプログラムでは不便です。そこで、単にプログラムを上から下に行うのではなく、順番を変えて実行したり、状況に応じて動きを変えて実行する(場合分け)ための命令を制御文(制御タイル)といいます。ここでは、その手始めとして、同じ動作を繰り返す、ループについて学習します。



ロボットのプログラミングを行う際に、多く用いられる書式として、**繰り返し(ループ)**があります。繰り返しは制御文と呼ばれる命令のひとつですが、制御文とはなんでしょうか？ある決まった動作を 1 回だけ行う場合は必要ありません。あるプログラムを繰り返し行う場合に必要となる構文で、自律型ロボットのプログラミングだけでなく、ほとんどのプログラミングにおいて必要不可欠なものです。



ArduBlock で使う制御文 初期化を伴うメインループ



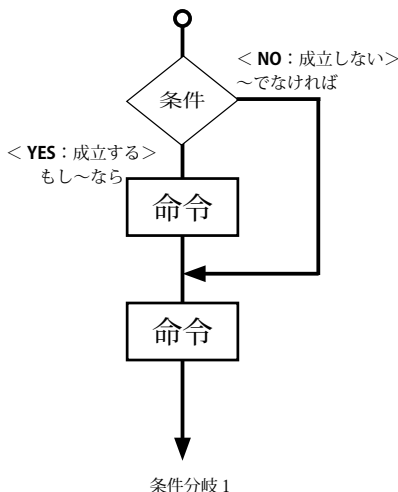
状況に応じて動きを変えて実行する(場合分け)ための命令を**条件分岐**といいます。**[もし~なら~]**で動きを変えていきます。条件が満たされない場合もありますので、**[もし~なら~、でなければ~]**も利用します。



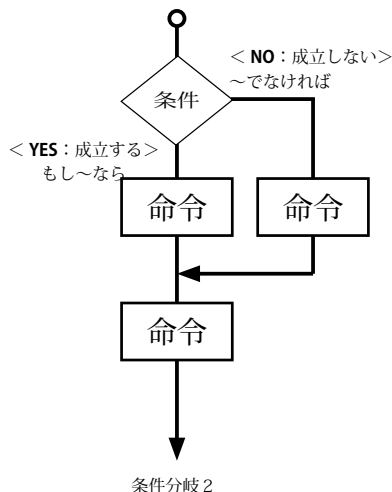
**[ずっと(メインループ)]**に入れて使います。



くり返し(条件ループ)



条件分岐 1



条件分岐 2

### Control statement (loop)

The program learned by the preceding section was lining up an order, and was straight from a start to an end.

This is also fine for first us, when repeating the same thing 100 times, is 100 of movement written?

When 10000 times are repeated eternally even if 100 can be written, it's very unreasonable.

When saying "Such case moves so, I'd like to do a different movement at such time.", it's inconvenient by a straight program.

So not just to execute a program in the bottom from the top, but order is changed and carried out, and a movement is changed according to the situation and an executed order for (case separation) is called a control statement (control tile).

The same movement is learned about a repeated loop as the beginning here.

When programing a robot, there are (loops) repeatedly as a used form.

A repeat is one of the order called a control statement, but what is a control statement? When doing the decided movement which is here only once, it isn't necessary. Need is an indispensable one in most programmings as well as a programming of an autonomous robot by the construction which it's needed when doing some programs repeatedly.

A movement is changed according to the situation and an executed order for (case separation) is called a conditional branch.

[If, ~ if, ~] please, a movement will be changed. The condition isn't sometimes met, so [if, ~ if, ~ or, ~] it's used.

It's put in [much repeatedly, (main loop)] and it's used.



6.4.2. 実験：くり返し（ループ）プログラム作成

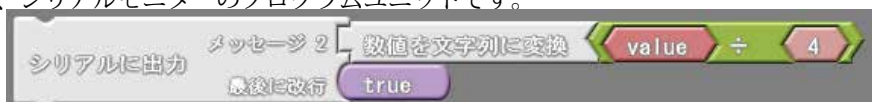
…スライダーの位置で、動きが変化するプログラムを作成してみます。

タイトルを配置してみましよう。今回、作成するのは、スライダーの位置によりモータ回転に変化を与えるプログラムです。

使用アイコンタイル表を参考に、左列の格納パレットよりドラック&ドロップでプログラムエリアに揃えます。

	左列のアイコン格納パレット	取り出す命令アイコンタイル	変数 / 定数の編集など
①	制御	ずっと (メインループ) 実行 メインプログラムのループ	そのまま使用
②	変数 / 定数	数値変数に値を設定する 変数 integer.variable.name 値 1	1 行目: 作成する変数 [Value] を入れ替える 2 行目: スライダーと入れ替える
③	変数 / 定数	整数の変数名	"value" に変更 ("整数の編集名" にマウスのカーソルをあててクリック選択し、"Value" と上書きする)
④	変数 / 定数	1	定数を 4 に変更 (1 の数値を、選択し、4 に上書きする)
⑤	STEM Du	スライダー	そのまま使用
⑥	通信	メッセージ 2 シリアルに出力 最後に改行 true	1 行目: [数値を文字列に変換] を入れ替える 2 行目: そのまま使用
⑦	通信	数値を文字列に変換	そのまま使用
⑧	演算	÷ "2つの整数の商"	左枠に作成した [Value] を入れる。 スライダーからの入力値は 0~1023、モータの PWM 値は 0~255 なので、変数「value」を 4 で割ります。 右枠に作成した [定数4] を入れる。
⑨	STEM Du	モーター M 1 スピード 255	[スピード] の位置の PWM 値 255 を外して 作成した [Value ÷ 4] のブロックパーツを入れる。

・この列が、シリアルモニターのプログラムユニットです。



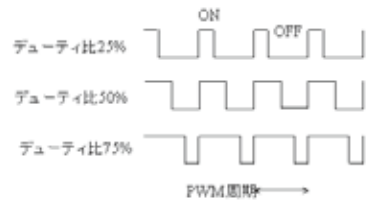
- ①. 左列アイコンパレット一番上にある **制御** をクリックし、出現するサブウィンドウから **ずっと(メインループ)** を選択し中央のプログラムフィールドにドラッグ&ドロップします。
- ②. 左列アイコンパレット **変数/定数** をクリックし、出現するサブウィンドウから **数値変数に値を設定する** を選択し中央のプログラムフィールドにドラッグ&ドロップします。
  - 1 行目: Value に変更
  - 2 行目: [スライダー] と入れ替えます。
- ③. 左列アイコンパレット **変数/定数** をクリックし、出現するサブウィンドウから **整数の変数名** を選択し中央のプログラムフィールドにドラッグ&ドロップします。
  - 1 行目: Value に変更
- ④. 左列アイコンパレット **変数/定数** をクリックし、出現するサブウィンドウから **定数1** のブロックを選択し中央のプログラムフィールドにドラッグ&ドロップします。
  - 定数: 4 に変更
- ⑤. 左列アイコンパレット **STEM Du** をクリックし、出現するサブウィンドウから **スライダー** を選択し中央のプログラムフィールドにドラッグ&ドロップします。
  - 変更を加えず、そのまま使用します。
- ⑥. 左列アイコンパレット **通信** をクリックし、出現するサブウィンドウから **シリアルに出力して改行** を選択し中央のプログラムフィールドにドラッグ&ドロップします。
  - 変更を加えず、そのまま使用します。
- ⑦. 左列アイコンパレット **通信** をクリックし、出現するサブウィンドウから **数値文字列に変換** を選択し中央のプログラムフィールドにドラッグ&ドロップします。
  - 変更を加えず、そのまま使用します。
- ⑧. 左列アイコンパレット **演算** をクリックし、出現するサブウィンドウから **数値文字列に変換** を選択し中央のプログラムフィールドにドラッグ&ドロップします。
  - 左枠に作成した [Value] を入れます。
  - 右枠に作成した [定数 4]
- ⑨. 左列アイコンパレット **STEM Du** をクリックし、出現するサブウィンドウから **モータ** を選択し中央のプログラムフィールドにドラッグ&ドロップします。
  - ドロップしたばかりの **モータ** ブロックは M(モータ) 1、スピード 255 のパラメータになっています。
    - ・モータは実験機で接続した M1 を使いますので、1 のままです。
    - ・スピードの位置の PWM 値 255 を外して作成した **Value ÷ 4** のブロックパーツを入れる。

PWM はパルス幅変調 (Pulse Width Modulation) と言って、デジタル信号の H と L の長さを変化させて、指令値を作る方法です。

例えば、LED を高速に点滅させて、点滅が人間の目にわからないようにした状態で H の時間と L の時間の比を変化させると LED の明るさが変化したように見えます。

モータの駆動時において、高速にスイッチの ON-OFF を繰り返し、ON になっている時間と OFF になっている時間の比を変更することによって、見かけ上、モータにかかる電圧を変更する駆動も可能になります。

次の図は PWM 信号の例です。周期的な ON-OFF の信号で、その周期は PWM 周期と呼ばれます。また (ON になっている時間) ÷ (PWM 周期) のことはデューティ比と呼ばれます。



### 6.4.3. プログラムアップロード

1. ArduBlock の **Arduino にアップロード** をクリックし、作成したスケッチ (プログラム) を、Arduino へアップロードします。
  - ・アップロードしたスケッチは、Arduino-IDE のスケッチに「C 言語」で表示されて、コンパイルされます。
2. コンパイル後は、すぐにマイコンボードへの書き込みが始まります。



## 6-5. 制御文(分岐)と入力

### 6.5.1. 実験：条件分岐プログラム作成…明るさセンサーの反応量で、動きが変化するプログラムを作成。

タイルを配置してみましょう。今回、作成するのは、明るさセンサーにより、検出する情報（センサー出力電圧）によりモータ回転に変化を与えるプログラムです。 使用センサー：明るさセンサー

使用タイル表を参考に、左列のアイコンパレットよりドラック&ドロップでプログラムエリアに揃えます。

	左列のアイコンパレット	取り出す命令アイコンタイル	変数 / 定数の編集など
①	制御	ずっと（メインループ） 実行	そのまま使用
②	変数 / 定数	数値変数に値を設定する 変数 integer variable name 値 0	1 行目 . 作成する変数 [light] と入れ替える 2 行目 . [明るさセンサ] と入れ替える  <small>プログラム中の Block 画像の [?] マークは、コメントがあることを知らせるアナウンスです。</small>
③	通信	シリアルに出力 ブレークアウト MESSAGE true	そのまま使用
④	通信	数値を文字列に変換	そのまま使用
⑤	制御	もし～なら～でなければ～	そのまま使用
⑥	変数 / 定数	整数の変数名	"light" に変更 (" 整数の編集名" にマウスのカーソルをあててクリック選択し、"light" と上書きします)
⑦	STEM Du	明るさセンサ	そのまま使用
⑧	変数 / 定数	1	しきい値 [50] に上書きし、左側に入れる。 右に作成した変数 light を入れる。  <small>プログラム中の Block 画像の [?] マークは、コメントがあることを知らせるアナウンスです。</small>
⑨	STEM Du	前進 スピード 255	そのまま使用
⑩	STEM Du	後進 (バック) スピード 255	そのまま使用 <small>プログラム中の Block 画像の [?] マークは、コメントがあることを知らせるアナウンスです。</small>



- ・コントローラの明るさ (light) センサが検出した情報により、接続したモータの回転が変化することが分かります。
- ・コントローラを手で覆うなどして、明るさを変更すると、モータ回転が変化することが確認できます。
- ・しきい値は、いったん 50 と仮定しプログラムしましたが、シリアルモニターを使用して、明るさセンサが検出している情報（センサー出力値）をモニターして、コントローラを設置した環境での明るさに合わせたしきい値に変更し実験してください。