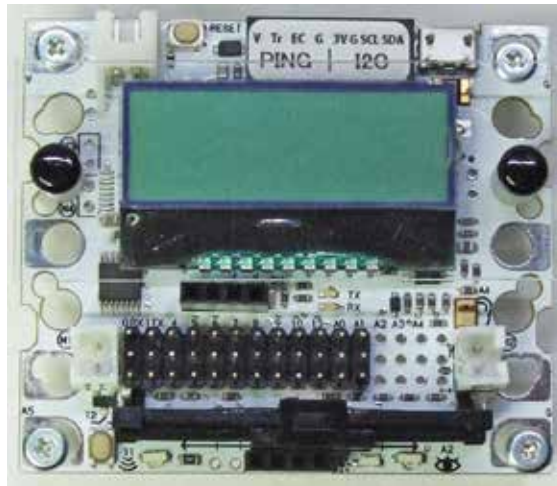




ROBO**D**ESIGNER™

User's Guide




ver4.01






1.はじめに	3	6.3.2. ArduBlock を起動する。	70
1-1. ご使用上の注意及び警告	3	6.3.3. ArduBlock プログラム作成 [1] モータを回転させてみる。	70
1-2. RoboDesigner システム相関図	4	6.3.4. プログラムアップロード前準備	71
2.インストールガイド	別添 PDF 参照ください。	6-3-5. プログラムアップロード	72
3.コントローラボード	5	6.3.6. 実験機でプログラム実験	73
3-1-1. マイコンボード RDC-104 仕様	5	6-4. 制御文 (ループ)	74
3.1.2. ピンアサイン	6	6.4.1 制御文とは	74
3-2. マイコンボード概要	7	6.4.2. 実験：くり返し (ループ) プログラム作成	75
3.2.1. RDC - 104TYPE I 仕様	7	6.4.3. プログラムアップロード	76
3.2.5. RDC - 104TYPE II 仕様	8	6-5. 制御文 (分岐) と入力	77
3.2.5. RDC - 104TYPE III 仕様	9	6.5.1. 実験：条件分岐プログラム作成	77
3.2.5. RDC - 104TYPE III+ (プラス) 仕様	10	7. ライントレース / 超音波障害物回避ロボ RDS-TEC31	79
3.3. マイコンボードは、センサーボードです。	11	7-1. RoboDesigner RDS-TEC31 構成部品	80
3-4. マイコンボード サンプルプログラム例	12	7-1-1. パーツリスト	80
ボタン・・・ボタンを押すと LED が点灯する	12	7.1.2. 部品の見方、使い方	81
アナログセンサ・・・アナログ入力をシリアルモニタする	13	7-2. マイコンボード概要	82
超音波距離センサ・・・超音波距離センサで測距	14	7.2.1. RDC - 104TYPE I 仕様	82
内臓赤外線距離センサ・・・赤外線 LED と明るさセンサで測距	15	7-3. 走行台車組み立て [Assemble of a Vehicles]	83
モータ・・・モータを回転する / 前進	19	7.3.1 ギアードモータ準備	83
サーボ・・・サーボを回転する	21	7.3.2. スタビライザーで左右を連結	83
サーボ II・・・スライダーの位置に応じてサーボが回転	22	7.3.3 マイコンボード取付	84
I2CLCD (RDC-104 用)・・・液晶に文字表示する	24	7.3.4. 電池ボックス取付	84
加速度 / ジャイロセンサ・・・MPU-6050 シリアルモニタ	26	7.3.5. キャスタ制作	85
I2C コンパスセンサ・・・1度単位で角度を取得できます。	28	7.3.6 キャスター取付	85
IoT・・・[拡張] Wi-Fi でシリアルモニターをする。	30	7-4. 配線	86
ブザー・・・ブザーでメロディを出力する	33	7-5. ライントレースロボ機体	87
3-5. パーツアクセサリ	35	7-6. 超音波障害物回避ロボの組立	88
4. プログラム環境の使い方	37	7.6.1. 超音波距離センサ取付	88
4-1. プログラム開発環境使用前準備	37	7-7. ロボットの動作確認をする	89
4-2. Scratch	38	7.7.1. プログラム開発環境使用事前準備	89
4.2.1. Scratch 画面の構成	38	7.7.2. 動作テストプログラムを作成します。	90
4.2.2. Scratch を起動する。	39	7.7.3. はじめてロボットを動作スタート、各種動作点検	91
4.2.3. 命令ブロックの実行規則	40	7-8. ライントレースロボのプログラム	92
4.2.4. ロボットで多く使うスクリプト	41	7.8.1 条件分岐プログラム	92
4-3. Scratch の動作実験	42	7.8.2 ArduBlock プログラミング	93
4.3.1. Sensor-Board の起動方法	42	7.8.3. ArduBlock ライントレースプログラム作成	93
4.3.2. スクリプト例を使用して動作実験	43	7.8.4 プログラム調整ロボット作り込み	94
4.3.3. Scratch は、接続状態のまま、マイコンボードが動作。	43	7.8.5. Arduino C でプログラミング	97
4-4. ArduBlock	44	7-9. 超音波障害物回避ロボのプログラミング	103
4.4.1. ArduBlock 画面の構成	44	7.9.1. 超音波距離センサー HC-SR04	103
4.4.2 ArduBlock の使い方	45	7.9.2. 超音波距離センサ取付	103
4.4.3. ArduBlock 仕様でのお断りとお願ひ	49	7.9.3 プログラム (ArduBlock) を作成します。	104
4.4.4. サンプルプログラムリスト	50	7-10. 製作例① 障害物回避とライントレースをするロボ	107
4-5. Arduino-IDE	51	7-11. ライントレース競技例	108
4.5.1. Arduino-IDE 画面の構成	51	7.12. IoT ロボ RDS-TEC31-Wi-Fi	109
4.5.2. Arduino-IDE の使い方	52	7.12.1. 機体の拡張	109
4.5.3. Arduino Example Code (スケッチの例)	53	7.12.2. ブレッドボード Wi-Fi モジュール組み込み	110
4.5.4. 使用例 Read Analog Voltage	54	7.12.3. データ送信プログラム	112
4.5.5 表計算ソフト	56	7.12.4 使用に際しての割り込み処理	115
5. ロボットの仕組み	57	資料：テクニカルガイド	238
5-1. RoboDesigner の構成	57	12.1 マイコンボード	
5-2. 実際の動作例	58	12.1.2. RDC-103 回路図	240
6. 実際に作ってみよう (実験)	59	12.1.3. RDC-104 仕様	241
6-1. パーツの組み立てとプログラム転送	59	12.1.4. RDC-104 回路図	242
6.1.1. 組み立て時の注意点	60	12.1.5. ピンアサイン	243
6.1.2.1. マイコンボード確認	61	12.1.6. モータドライバ	243
6.1.2.2. 実験機組み立て	61	12.1.7 ハードウェア割り込みの処理	244
6.1.2.4. 実験装置の配線	62	12-2. タッチセンサ	245
6.1.4. プログラムの作成	63	12-3. 赤外線アナログセンサ - JES-7023VAD	246
6-2. Scratch でプログラム実験	63	12.3.3. センサ調整	247
6.2.1. Scratch を起動する。	63	12-4. 変調赤外線センサ RDI-203JR	249
6.2.2. Scratch 画面の構成	64	12.4.1. 変調赤外線センサ特性	249
6.2.3. プログラム作成	65	12.4.2. ロボカップジュニア：パルスボールについて	249
6.2.4. コントローラボードと接続実験	66	12-5. 測距センサ RDI-209	250
6-2-4-1. Sensor-Board の起動方法	66	12-6. 超音波距離センサ HC-SR04	251
6.2.5. スクリプト例を使用して動作実験	67	12.6.1. [一般的] 超音波センサとは	251
6.2.6. Scratch は、接続状態のまま、マイコンボードが動作。	67	12.6.2. 圧電セラミックとは	251
6.2.7. Scratch 使用時のロボットで多く使うスクリプト	67	12.6.3 超音波距離センサ HC-SR04	251
6.2.8. 例題	68	12-7. I ² C コンパスセンサ RDI-5883L QMC	252
6-3. ArduBlock でプログラム実験	69	12-8. モータ付ギアボックス RDO-500P	254
6.3.1. ArduBlock 画面の構成	69	12.8.2. ギアードモータ RDO-502-48 RDO-502-120	255
		12.8.3. エンコーダ付ギアードモータ RDO-502EN	256
		13. トラブルシューティング【troubleshooting】	259

1. はじめに (Warnings and Safety Precautions)





1-1. ご使用上の注意及び警告

記号の意味 (Definition of symbols)	
	←記号は、注意(気をつけること)を表します。 This symbol represents CAUTION (Careful forethought to avoid danger or harm).
	←記号は、禁止(してはいけないこと)を表します。 This symbol represents PROHIBITED (Not permitted).
	←記号は、義務(しなければならないこと)を表します。 This symbol represents MUST (Something that is absolutely required or indispensable).

警告：人が死亡または、重症を負う恐れのある内容を示します。
WARNING: This part introduces the matters related to potentially serious injuries, or death.

	工具及び工作機械の取り扱いには、十分注意してください。 ・ ・ ・ ケガや事故の原因となります。 Pay great attention to operate machine tools. Careless operation usually involves accidents or injuries.
	ネジやナットその他小さな部品は、口に入れたりしないよう十分注意、管理してください。 ・ ・ ・ 事故の原因となります。 Pay great attention to avoid small parts such as screws and nuts be put into the mouth otherwise, accidents or injuries may happen.
	制御基板(コントローラ・センサ等)には、説明書に指示ある電圧以上の電圧を入力しないでください。 発火する恐れがあります。 ・ ・ ・ 火災やヤケドの原因となります。 Do not input higher supply voltage of the circuit board (control board, sensors, etc.) than the threshold instructed in the manuals. Otherwise, it will result in fire accidents, and may lead to terrible burn injuries.

注意：人がケガをしたり、財産に損害を与える恐れのある内容を示します。
CAUTION: This part introduces the matters related to potential injuries, and property damages.

	部品を切り取ったり、カット加工する際、カットのしかたによっては、カット面がとがったり、鋭利になっている場合がありますので、ケガや事故のないよう十分注意してください。 ・ ・ ・ ケガや事故、器物を損壊する恐れがあります。 The cutting surfaces become possibly quite sharp during cutting processing, so you should pay great attention to avoid injuries or damages. ... Causes of injuries, accidents, component damages.
	コントローラボード等基板に電源を投入する際は、電源(電池ボックス及び乾電池)の極性に注意して下さい。 ・ ・ ・ 部品を破損する原因となります。 To confirm the polarity of power supply (dry batteries, battery housing, etc.) before connecting them to the circuit board such as controller. ... Causes of component damages.
	制御基板や CD-ROM、工作物には、重いものを載せたり、曲げたり、投げたり、落としたり、熱いものに近づけたりしないでください。 ・ ・ ・ 破損、ケガや事故の原因となります。 To prevent accident or injury on you, do not put heavy objects on the control board or CD-ROM. Also do not bend, throw, drop and do not put any heat source near the control board or CD-ROM. ... Causes of damages, injuries and accidents.
	制御基板など電子部品は、基板裏面に回路が露出していますので、電源を接続したまま、導通性があるアルミや鉄などの金属類の上に置かないようにしてください。 ・ ・ ・ 誤って電源が入ると、電子回路がショートして、焼損、破損の原因となります。 Don't put the control board and other electronic components on the electrically conductive metals such as aluminum and iron due to the exposed circuit on the bottom of the board. ... If the power is turned on accidentally, circuit may be shorted, causing burning out, and damage of circuit board.

	制御基板など電子部品を水に浸けたり、濡らしたり Soaked in water, be wet, or be touched by wet hands. ... Causes of damage.
	コネクター・ケーブル等配線材は無理に引っ張ったり極端に折り曲げたりしないでください。 ・ ・ ・ ケーブルが破損し、動作不良の原因となります。 Don't pull or bend wire cables and connectors excessively. ... Causes of damaged cables and malfunction.
	制御基板など電子部品は説明書に指示及び説明等がない限り不要に改造しないでください。 ・ ・ ・ 動作不良の原因となります。 Don't attempt to modify the control board and other electronic components unnecessarily without such explanations or instructions in the manual. ... Causes of malfunction.
	製品(電子回路基板・モータ等機構部品・シャーシ等筐体部品・パッケージ・ケース・CD-R)の一部には、鋭利な部分や先の尖った部分があります。 ケガや事故の原因となりますので、充分注意して下さい。 Parts of the product (electrical circuit boards・packages・cases・chassis parts・mechanical parts such as motor・CD-ROM) contain some sharp edges or sharp objects. ... Causes of injuries and accidents.
	著作権者の許諾なく、本製品の複製、賃貸、その他類する行為は、法律で禁じられています。 Replication rent of this product, and other similar acts without the permission of copyright holder are prohibited by law.
	組み立て後、長時間使用しない場合は、乾電池を必ず電池ボックスから外して下さい。 Be sure to remove batteries from the battery housing if you may not use it in a long term after assembly.
	組み立て後の不用となった端材やパッケージに使用しているダンボール等のゴミは、各市町村のゴミ処理方法に従って処分して下さい。 Dispose of waste (e.g., useless scraps, cardboard used for packaging) after assembly should obey the waste treatment rules of each municipality.

おことわり (DISCLAIMER)

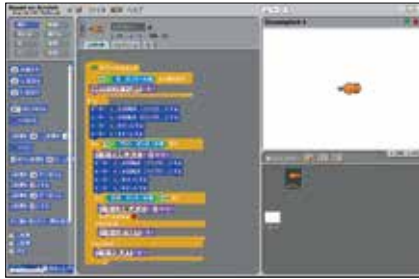
1.	この学習教材を正しくお使いいただくために、組み立て加工前に、必ずこの説明書をよくお読みください。 In order to properly use this learning material, be sure to carefully read this manual before starting the assembly process.
2.	当社は、この製品の使用の誤り、使用中に発生した故障、その他の不具合によって生じた損害については、法令上賠償責任が認められる場合を除き、その一切の責任を負いませんので、あらかじめご了承ください。 We assume no responsibility or liability for any loss, damage or injury as a result of misuse except for acknowledging the indemnity liability by regulations.
3.	本製品は、ご利用になる方の学習目的達成を支援するための教材です。本製品をご使用になられて得られた結果に関するいかなる保証もいたしかねますので、あらかじめご了承ください。 This product is an educational material to help users to achieve the learning goal. We do not guarantee any results by using this product.
4.	本製品の仕様及び、価格等については、予告なく変更する場合があります。 Specifications, prices and so on of this product are subject to changes without notice.

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

1-2. RoboDesigner システム相関図

Scratch

System correlation chart



50種の言語に対応したプログラミング環境
コントローラはPCと接続のまま センサーボードで使用します。

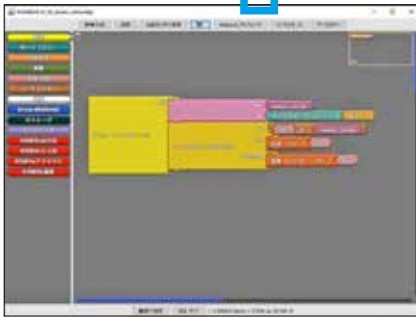
自動プログラム変換
C言語で展開



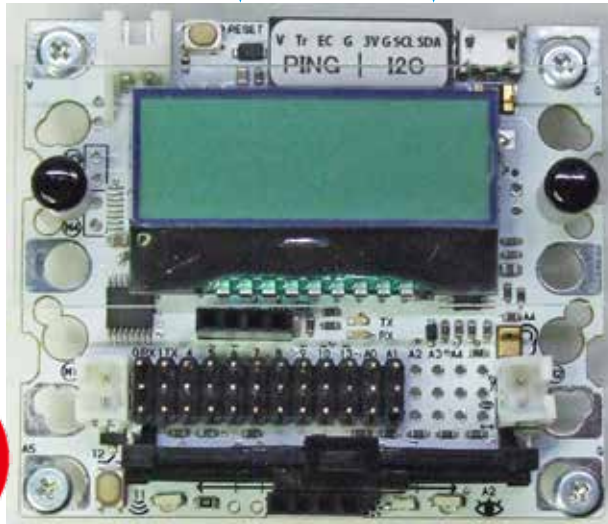
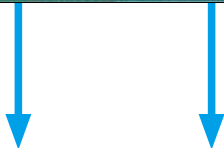
C++でコーディング可能な
Arduino-IDE

自動プログラム変換
C言語で展開

ArduBlock



自律型ロボットのための視覚的プログラミング環境
マイコンボードはPCとの接続をはずして、動作可能



ライトレースロボ



IoTロボWi-Fi/ライト
レース超音波障害物回避
ロボ



サッカーチャレンジロボ



PID制御 描画ロボ

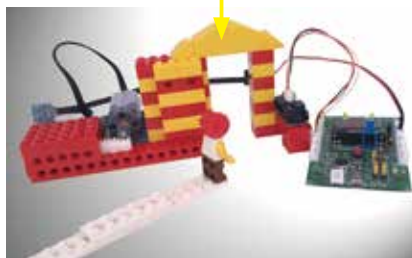


●栽培工場
安全な食物育成を目指し、LED 照明を制御

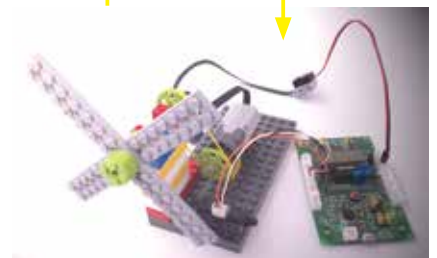


世界中で最も使われている
マイコンボード
Arduino との互換性を持ち電子回路の初心者でも始められる手軽さと、これまでのRoboDesignerのセンサーもそのまま使える柔軟性拡張性を合わせ持つ

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

3. マイコンボード

3-1-1. マイコンボード RDC-104 仕様

Table.3.1.2 Specification				
Model No.	 RDC-104 TYPE I	 RDC-104_TYPE II	 RDC-104_TYPE III	 RDC-104_TYPE III +
機能概要	<ul style="list-style-type: none"> 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。 Scratch上で、2つのモータまで動かすことが可能。 	<ul style="list-style-type: none"> 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。 2つまでのモータを使って自律型ロボットを作成可能 	<ul style="list-style-type: none"> 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。 	
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"> Scratch1.4を使ってプログラムし動作可能。 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。 	<ul style="list-style-type: none"> 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。) 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。 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)。 Scratch1.4を使ってプログラムし動作させることも可能。 		
プログラムダウンロード	<ul style="list-style-type: none"> 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。 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 	<ul style="list-style-type: none"> PCのUSBポートからダウンロード可能 	<ul style="list-style-type: none"> PCのUSBポートからダウンロード可能 	<ul style="list-style-type: none"> PCのUSBポートからダウンロード可能
DC モータ出力	<ul style="list-style-type: none"> 2個のDC ブラシモータ制御が可能。M1,M2 正転/反転/停止及び回転速度コントロールが可能。 		<ul style="list-style-type: none"> 4個のDC ブラシモータ制御が可能。M1,M2,M3,M4 正転/反転/停止及び回転速度コントロールが可能。 	
	※モータ1個あたり0.5A程度が上限です。接続するモータの定格電圧、電流をご確認ください。			
サーボ出力 デジタルピンと兼用	<ul style="list-style-type: none"> 10個のサーボモータ制御が可能 		<ul style="list-style-type: none"> 10個のサーボモータ制御が可能 	
入力ポート	<ul style="list-style-type: none"> アナログ入力コネクタ × 2個 		<ul style="list-style-type: none"> 5(ボード上のセンサとジャンパワイヤで接続可能) 	
みの虫クリップ端子	◎	◎	◎	◎
通信ポート	<ul style="list-style-type: none"> 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。 I²C,UART 通信可能 (加速度センサ+ ジャイロ(I²C)増設可能) 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。 			
搭載機能	<ul style="list-style-type: none"> 赤外線センサ、音センサ、光センサ、スライダ、白色LEDをボード上に搭載 I²C 接続各種センサ、超音波センサ、加速度センサ+ ジャイロ(I²C)を追加可能 			
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"> USB / V1 コネクタから供給 6.0V 		<ul style="list-style-type: none"> USB / V1 コネクタから供給 6.0V
	モータ M3、M4	<ul style="list-style-type: none"> V2 コネクタから供給 モータに合わせた電圧 (M3,M4用 3~12V) 		<ul style="list-style-type: none"> V2 コネクタから供給 モータに合わせた電圧 (M3,M4用 3~12V)
基板サイズ 重さ	66mm x 55mm 厚さ20mm ・重さ 約19.0グラム			

3.1.2. ピンアサイン

基板により、ピンアサインが異なります。プログラム時にご注意ください。

	RDC-102R4	RDC-103R4	RDC-104R4
light:明るさセンサ	A4	A4	A2
slider:スライダー	A5	A5	A3
sound:音センサ	A0	A0	A4
clip:ミノムシクリップ端子	-	A3	A5
button:ボタン	12	12	12
LED:白色LED	13	13	13
IR LED:赤外線LED	-	11	11
PING:超音波センサソケット	-	11	11
MPU 6050:加速度/ジャイロ	I2C 2SDA 3SCL	I2C 2SDA 3SCL	-
Buzzer:ブザー	0	0	-
M1:モータ1	4 5 6(PWM)	4 5 6(PWM)	4 9 6(PWM)
M2:モータ2	7 8 9(PWM)	7 8 9(PWM)	7 8 5(PWM)
M3:モータ3	0 1 10(PWM)	0 1 10(PWM)	0 1 11(PWM)
M4:モータ4	11 12 13(PWM)	11 12 13(PWM)	10 12 13(PWM)
LCD:液晶表示	SPI 10-CS1B/SS 11-RS	SPI 10-CS1B/SS 1-RS	I2C 2-SDA 3-SCL

3-2. マイコンボード概要

3.2.1. RDC – 104TYPE I仕様

- ・赤外線センサ、LED / 光センサ、音センサ、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。
- ・I²C機器接続可能、ジャイロ/加速度センサ接続可能
- ・サーボモータ10 個まで接続可能
- ・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

※ 1・本ボードをセンサーボードとして使用、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-104TYPE I OUTLINE

- ・It's equipped with LED / Light sensor, Sound 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 possible to connect Ultrasonic Sensor, Acceleration/a Gyro sensor - (I2C sensor socket use), (separate sale part)
 - ・It's even possible to connect 10 servomotors.
 - ・M3, M4 and a LCD module aren't loaded into RDC-104TYPE I.
 - ・It's even possible to move 2 motors using Scratch, and always it's connected with a PC and it's used.*
- *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	MCU / ATMEGA32U4 Clock 8MHz	http://media.digikey.com/pdf/Data%20Sheets/Atmel%20PDFs/ATmega16U4,32U4.pdf
音センサ	Sound Sensor SPU0410HR5H-1	https://media.digikey.com/pdf/Data%20Sheets/Knowles%20Acoustics%20PDFs/SPU0410HR5H-1.pdf
スライダーボリューム	SlidePotentiometers Alps RS30H121	http://www.alps.com/WebObjects/catalog.woa/J/HTML/Potentiometer/Slide-Potentiometers/
明るさセンサ	Light sensor Everlight PT12-21C	http://www.everlight.com/file/ProductFile/PT12-21C-TR8.pdf
赤外線センサ	Vishay Semiconductor VSMB10940	https://www.digikey.jp/product-detail/ja/VSMB10940/VSMB-10940TR-ND/3915205

デジタル入出力 Digital in/out

端子を使用したい時はピンで接続します。

ピン番号	ヘッダーピン	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			ボタン button
10			サーボ / 超音波 / 赤外線LED R/C servo motor / UltraSonic / InfraRed
0		RX	サーボ / シリアルRX R/C servo motor / Serial RX
1		TX	サーボ / シリアルTX R/C servo motor / Serial TX
11			PING /IRLED
6	M 1 (0.5A 程度)		サーボ / M1 PWM 制御 / PWM 出力可能 R/C servo motor / M1 PWM control / PWM output
9			サーボ / M1 制御 / PWM 出力可能 R/C servo motor / M1 control / PWM output
4	M 2 (0.5A 程度)		M1 制御 M1 control
7			M2 制御 M2 control
8			M2 制御 M2 control
5			M2 PWM 制御 / PWM 出力可能 M2 PWM control / PWM output
電源コネクタ			回路、M1.M2
		V1	電源コネクタ Power Conector

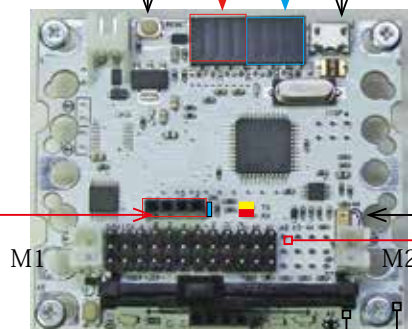
LED		
ON	青色	電源確認 Blue LED
RX	赤色	通信確認 Red LED
TX	緑色	通信確認 Green LED

増設可能

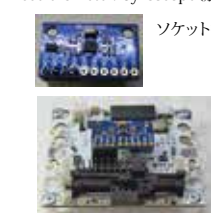
★超音波センサ Ultrasonic sensor
(差し込んで使用します。別売品)



リセットスイッチ Reset Button
電源コネクタ V1(回路電源、M1.M2)
PING 11 Trig Echo
I²C コネクタ SCL, SDA
USB コネクタ USB connector



12 ボタン button
赤外線LED 11 Infrared sensor
加速度/ジャイロ/温度センサ Accelerometer/Gyroscope (別売品) (I2C)
ソケット接続使用



加速度/ジャイロセンサ取付状態図

ピン番号	ヘッダーピン	記号	解説
P I N G	V	電源 Power	⊕ライン
	Tr	Trig	
	Ec	Echo	
	G	⊖ライン	
I ² C	3.3V	電源 Power	⊕ライン
	G	⊖ライン	
	SCL	SCL	
	SDA	SDA	
		A5	アナログ入力ポート Analog input
		A4	音センサ Sound Sensor / Analog input
		A3	スライダー:アナログ入力ポート Slider / Analog input
		A2	明るさセンサ/アナログ入力ポート Light / Analog input
		A1	アナログ入力ポート Analog input
		A0	アナログ入力ポート Analog input
		G	⊖ライン

音センサ Sound Sensor A4

アナログ入力 Analog input A0~A1 2ポート
ピンで接続します。A はAnalog のA) 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。

みの虫クリップ用端子 Terminal for clips (抵抗等測定) A5

スライダー (可変抵抗) Slider resistance A3

明るさセンサ Light sensor A2

発光 白色LED / 受光 フォトトランジスタ

3.2.5. RDC – 104TYPE II仕様

- ・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。
- ・赤外線センサ、LED / 光センサ、音センサ、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。
- ・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
- ・外部加速度センサ/ジャイロセンサ（別売）接続可能
- ・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10個まで接続可能

Control board RDC-104TYPE 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 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, Acceleration/a Gyro 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 10 servomotors.

仕様	The specification	DataSheet URL
マイコン / ATMEGA32U4	MCU / ATMEGA32U4 Clock 8MHz	http://media.digikey.com/pdf/Data%20Sheets/Atmel%20PDFs/AT-mega16U4,32U4.pdf
音センサ	Sound Sensor SPU0410HR5H-1	https://media.digikey.com/pdf/Data%20Sheets/Knowles%20Acoustics%20PDFs/SPU0410HR5H-1.pdf
スライダーボリューム	SlidePotentiometers Alps RS30H121	http://www.alps.com/WebObjects/catalog.woa/J/HTML/Potentiometer/Slide-Potentiometers/
明るさセンサ	Light sensor Everlight PT12-21C	http://www.everlight.com/file/ProductFile/PT12-21C-TR8.pdf
赤外線センサ	Vishay Semiconductor VSMB10940	https://www.digikey.jp/product-detail/ja/VSMB10940/VSMB-10940TR-ND/3915205

デジタル入出力 Digital in/out

端子を使用したい時はピンで接続します。

ピン番号	ヘッダーピン	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			ボタン Button
10			サーボ / 超音波 / 赤外線LED R/C servo motor / UltraSonic / InfraRed
0		RX	サーボ / シリアルRX R/C servo motor / Serial RX
1		TX	サーボ / シリアルTX R/C servo motor / Serial TX
11			PING / IR-LED
6	M1 (0.5A程度)		サーボ / M1 PWM 制御 / PWM 出力可能 R/C servo motor / M1 PWM control / PWM output
9			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)		M2 制御 M2 control
8			M2 制御 M2 control
5			M2 PWM 制御 / PWM 出力可能 M2 PWM control / PWM output
電源コネクタ			
	V1		電源コネクタ 回路、M1.M2 Power Conector

LED			
	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

増設可能

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

ピン番号	ヘッダーピン	記号	解説
P I N G		V	電源 ⊕ライン Power
		Tr	Trig
		Ec	Echo
		G	⊖ライン
I ² C		3.3V	電源 ⊕ライン Power
		G	⊖ライン
		SCL	SCL
		SDA	SDA
		A5	アナログ入力ポート Analog input
		A4	音センサ Sound Sensor / Analog input
		A3	スライダー:アナログ入力ポート Slider / Analog input
		A2	明るさセンサ/アナログ入力ポート Light / Analog input
		A1	アナログ入力ポート Analog input
		A0	アナログ入力ポート Analog input
		G	⊖ライン

- ・音センサ Sound Sensor A4
- ・アナログ入力 Analog input A0~A1 2ポート
ピンで接続します。A は Analog の A) 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。
- ・みの虫クリップ用端子 Terminal for clips (抵抗等測定) A5
- ・スライダー (可変抵抗) Slider resistance A3
- ・明るさセンサ Light sensor A2
発光 白色LED / 受光 フォトトランジスター

3.2.5. RDC – 104TYPE III仕様

- ・ 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。
- ・ 赤外線センサ、LED / 光センサ、音センサ、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えます。
- ・ 外部超音波センサー / 加速度・ジャイロセンサ増設可能 (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 10 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 モニターが必要な場合は、RDC-104TYPE III plus をご利用ください。

Control board RDC-104TYPE 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 a Slider on the board, using these, you can control variously.
- ・ It's possible to connect Ultrasonic Sensor, and the Acceleration / Gyro 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 10 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	MCU / ATMEGA32U4 Clock 8MHz	http://media.digikey.com/pdf/Data%20Sheets/Atmel%20PDFs/ATmega16U4,32U4.pdf
音センサ	Sound Sensor SPU0410HR5H-1	https://media.digikey.com/pdf/Data%20Sheets/Knowles%20Acoustics%20PDFs/SPU0410HR5H-1.pdf
スライダーボリューム	SlidePotentiometers Alps RS30H121	http://www.alps.com/WebObjects/catalog.woa/J/HTML/Potentiometer/SlidePotentiometers/
明るさセンサ	Light sensor Everlight PT12-21C	http://www.everlight.com/file/ProductFile/PT12-21C-TR8.pdf
赤外線センサ	Vishay Semiconductor VSMB10940	https://www.digikey.jp/product-detail/ja/VSMB10940/VSMB-10940TR-ND/3915205

デジタル入出力 Digital in/out

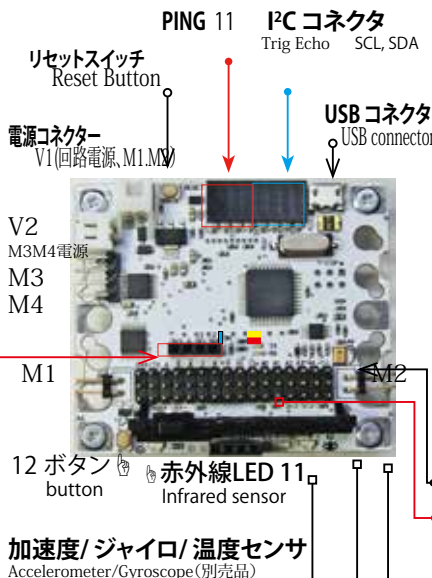
端子を使用したい時はピンで接続します。

ピン番号	ヘッダー	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	M4 (0.5A 程度)		ボタン Button
10			サーボ / 超音波 / 赤外線LED R/C servo motor / UltraSonic / InfraRed
0			サーボ / シリアルRX R/C servo motor / Serial RX
1	M3 (0.5A 程度)		サーボ / シリアルTX R/C servo motor / Serial TX
11			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
9	M1 (0.5A 程度)		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)		M2 制御 M2 control
5			M2 PWM 制御 / PWM 出力可能 M2 PWM control / PWM output
電源コネクタ			
		V1	電源コネクタ 回路、M1.M2 Power Conector
		V2	電源コネクタ M3,M4 Power Conector

LED			
	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

増設可能

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



加速度/ジャイロセンサ取付状態図

ピン番号	ヘッダー	記号	解説
P I N G		V	電源 ⊕ライン Power
		Tr	Trig
		Ec	Echo
		G	⊖ライン
I ² C		3.3V	電源 ⊕ライン Power
		G	⊖ライン
		SCL	SCL
		SDA	SDA
		A5	アナログ入力ポート Analog input
		A4	音センサ Sound Sensor / Analog input
		A3	スライダー:アナログ入力ポート Slider / Analog input
		A2	明るさセンサ/アナログ入力ポート Light / Analog input
		A1	アナログ入力ポート Analog input
		A0	アナログ入力ポート Analog input
		G	⊖ライン

- ・音センサ Sound Sensor A4
- ・アナログ入力 Analog input A0~A5 6ポート
ピンで接続します。A は Analog の A) 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。
- ・みの虫クリップ用端子 Terminal for clips (抵抗等測定) A5
- ・スライダー (可変抵抗) Slider resistance A3
- ・明るさセンサ Light sensor A2
発光 白色LED / 受光 フォトトランジスタ

3.2.5. RDC – 104TYPE III+ (ﾌﾞﾗｽ) 仕様

- ・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。
- ・赤外線センサ、LED / 光センサ、音センサ、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 / 加速度・ジャイロセンサ増設可能 (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10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モニター不要な場合は、RDC-104TYPEIIIをご利用ください。

Control board RDC-104TYPE 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 a Slider on the board, using these, you can control variously.
- ・It's possible to connect Ultrasonic Sensor, and the Acceleration / Gyro 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 10 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	MCU / ATMEGA32U4 Clock 8MHz	http://media.digikey.com/pdf/Data%20Sheets/Atmel%20PDFs/ATmega16U4,32U4.pdf
音センサ	Sound Sensor SPU0410HR5H-1	https://media.digikey.com/pdf/Data%20Sheets/Knowles%20Acoustics%20PDFs/SPU0410HR5H-1.pdf
スライダボリューム	SlidePotentiometers Alps RS30H121	http://www.alps.com/WebObjects/catalog.woa/J/HTML/Potentiometer/SlidePotentiometers/
明るさセンサ	Light sensor Everlight PT12-21C	http://www.everlight.com/file/ProductFile/PT12-21C-TR8.pdf
赤外線センサ	Vishay Semiconductor VSMB10940	https://www.digikey.jp/product-detail/ja/VSMB10940/VSMB-10940TR-ND/3915205

デジタル入出力 Digital in/out

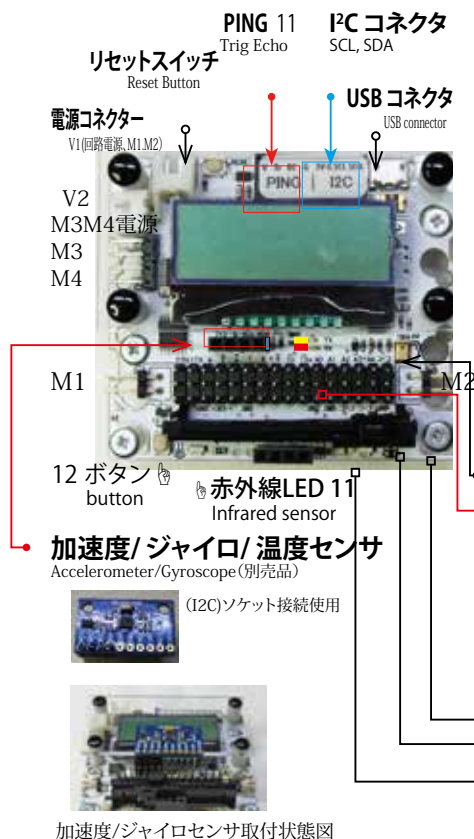
端子を使用したい時はピンで接続します。

ピン番号	ヘッダーピン	記号	解説
13	M4 (0.5A程度)		サーボ / 白色LED/PWM 出力可能 R/C servo motor / White LED / PWM output
12			ボタン Button
10	M3 (0.5A程度)		サーボ / 超音波 / 赤外線LED R/C servo motor / UltraSonic / InfraRed
0			サーボ / シリアルRX R/C servo motor / Serial RX
1	M1 (0.5A程度)		サーボ / シリアルTX R/C servo motor / Serial TX
11			サーボ / LCD CS / PWM 出力可能 R/C servo motor / LCD CS / PWM output
6	M1 (0.5A程度)		サーボ / M1 PWM 制御 / PWM 出力可能 R/C servo motor / M1 PWM control / PWM output
9			サーボ / M1 制御 / PWM 出力可能 R/C servo motor / M1 control / PWM output
4	M2 (0.5A程度)		M1 制御 M1 control
7			M2 制御 M2 control
8	M2 (0.5A程度)		M2 制御 M2 control
5			M2 PWM 制御 / PWM 出力可能 M2 PWM control / PWM output
電源コネクタ			
	V1		電源コネクタ 回路、M1.M2 Power Conector
	V2		電源コネクタ M3.M4 Power Conector

LED			
	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

増設可能

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



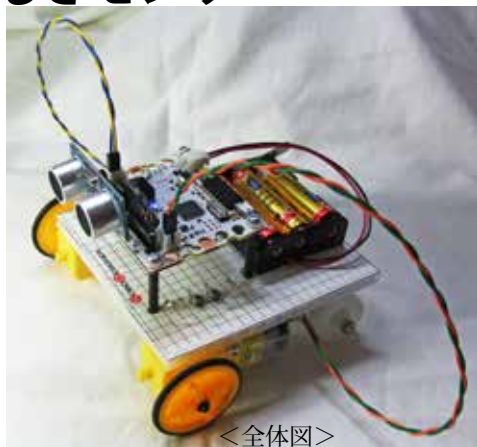
ピン番号	ヘッダーピン	記号	解説
P I N G		V	電源 ⊕ライン Power
		Tr	Trig
		Ec	Echo
		G	⊖ライン
I ² C		3.3V	電源 ⊕ライン Power
		G	⊖ライン
		SCL	SCL
		SDA	SDA
		A5	アナログ入力ポート Analog input
		A4	音センサ Sound Sensor / Analog input
		A3	スライダ:アナログ入力ポート Slider / Analog input
		A2	明るさセンサ/アナログ入力ポート Light / Analog input
		A1	アナログ入力ポート Analog input
		A0	アナログ入力ポート Analog input
		G	⊖ライン

- 音センサ Sound Sensor A4
- アナログ入力 Analog input A0~A5 6ポート
- ピンで接続します。A は Analog のA) 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。
- みの虫クリップ用端子 Terminal for clips (抵抗等測定) A5
- スライダ (可変抵抗) Slider resistance A3
- 明るさセンサ Light sensor A2
- 発光白色LED / 受光フォトトランジスタ

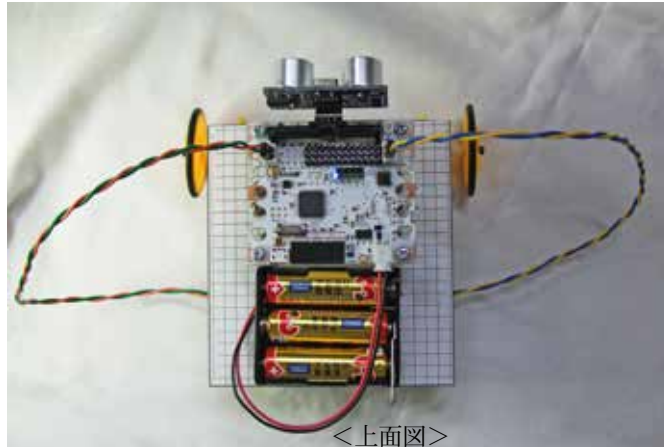
3.3. マイコンボードは、センサーボードです。

各種センサーを搭載していますので、いろいろな制御システムを作成できます。
センサ機能を使用した明るさセンサ / 障害物回避ロボの例です。

明るさセンサ …明るさセンサで動くとき、超音波センサは、不要です。

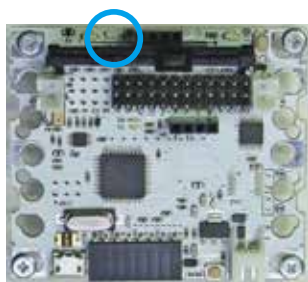


<全体図>



<上面図>

明るさセンサの値に応じてモータを低速前進したり、高速後進をしたりします。



RDC-104



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 RDC-104:(A2)/Motor1/ Motor2

変数 sensor_value 搭載光センサの出力値を格納します。

※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、マイコンボードへ書き込みます。

動作手順

- ・周囲の明るさがしきい値(100) ≤なら 180のスピードで前進します。
- ・でなければ 255のスピードで後進します。

■Arduino Cのサンプルプログラムは、[Arduino] > [ファイル-スケッチの例-01. Basics-AnalogReadSerial]を開いて、アナログセンサの指定をA2(明るさ)に変更してください。

障害物回避 次頁以降に記載の超音波センサのサンプルプログラムをご利用ください。

超音波センサを差し込む…障害物回避ロボへ変更



RDC-10 4

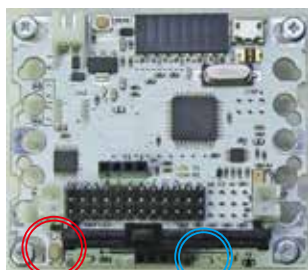
RDC は、Arduino / SensorBoard 互換マイコンボードです。以下、次頁サンプルプログラムの例です。

3-4. マイコンボード サンプルプログラム例

ボタン・・・ボタンを押すとLEDが点灯する

●ArduBlockサンプルプログラム

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



RDC-10 4



ボタン(12番ピン●)が押されたらLED(13番ピン●)を点灯します。
ボタンを離したら消灯します。

シリアルモニタにボタンの状態(押すと0、離すと1)を出力します。
ボタンはプルアップするので、離した状態が1 (HIGH)です。

■Arduino Cのサンプルプログラム

・Arduinoスケッチ例> 02.Digital > DigitalInputPullup(ボタンを押すとLEDが点灯する)を読み込み
pinMode(2, INPUT_PULLUP) と digitalWrite(2)のピン番号を2から12に修正する。

```
void setup() {
  //start serial connection
  Serial.begin(9600);
  //configure pin2 as an input and enable the internal pull-up resistor
  pinMode(2, INPUT_PULLUP);
  pinMode(13, OUTPUT);
}

void loop() {
  //read the pushbutton value into a variable
  int sensorVal = digitalRead(2);
  //print out the value of the pushbutton
  Serial.println(sensorVal);

  // Keep in mind the pullup means the pushbutton's
  // logic is inverted. It goes HIGH when it's open,
  // and LOW when it's pressed. Turn on pin 13 when the
  // button's pressed, and off when it's not:
  if (sensorVal == HIGH) {
    digitalWrite(13, LOW);
  } else {
    digitalWrite(13, HIGH);
  }
}
```

アナログセンサ・・・アナログ入力をシリアルモニタする

● ArduBlock サンプルプログラム

・Block ソースコードは、PC/MyDocuments/Arduino/へ配置したサンプルフォルダ [RDC_ArduBlockSamples] に、ファイル名【02a_RDC_AnalogRead.abp】で格納されています。



- アナログ入力の数値に合わせて LED の明るさを変えます。
 アナログ入力値は 0~1023、PWM の出力値は 0~255 なので、変数「value」を 4 で割ります。
- ・センサの値を変数 value に格納します。
 - ・シリアルに value を出力します（シリアルモニタで確認できます）。
 - ・LED に value を 1/4 にした PWM 出力値を設定します。
 - ・ずっと繰り返します。

■ Arduino C のサンプルプログラム

スケッチの例 > 01.Basics > AnalogReadSerial（アナログ入力をシリアルモニタに出力する）

AnalogRead(A0) のピン番号を各センサのピン番号に修正する。

(RDC-104：明るさ A2 / スライダー A3 / 音 A4 / 抵抗センサ A5 RDC-103：明るさ A4 / スライダー A5 / 音 A0 / 抵抗センサ A3)

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

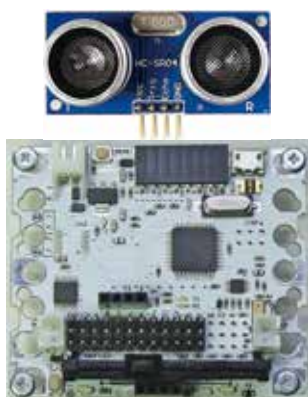
// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1); // delay in between reads for stability
}
```

使用するピン番号に修正する

超音波距離センサ . . . 超音波距離センサで測距

● ArduBlock サンプルプログラム

- Block ソースコードは、PC/MyDocuments/Arduino/ へ配置したサンプルフォルダ [RDC_ArduBlockSamples] に、ファイル名【03_RDC_Ping.abp】で格納されています。



RDC-104



超音波距離センサHC-SR04で距離を測ります。

ソケットに超音波距離センサを差し込みます。

ピン名をよく確認して取り付けてください。

(Echo/Triggerはどちらもピン11に接続されます)

距離をシリアルモニタに出力します。

■ Arduino C のサンプルプログラム

スケッチの例 > 06.Sensors > Ping (HC-SR04 超音波センサで測距する)

const int pingPin = 7 のピン番号を 7 から 11 に修正する。

```
// this constant won't change. It's the pin number
// of the sensor's output:
const int pingPin = 7;

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
}

void loop() {
  // establish variables for duration of the ping,
  // and the distance result in inches and centimeters:
  long duration, inches, cm;

  // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);

  // The same pin is used to read the signal from the PING))) : a HIGH
  // pulse whose duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);

  // convert the time into a distance
  inches = microsecondsToInches(duration);
  cm = microsecondsToCentimeters(duration);
```

ピン番号を 7 から 11 に修正する

```

Serial.print(inches);
Serial.print("in, ");
Serial.print(cm);
Serial.print("cm");
Serial.println();

delay(100);
}

long microsecondsToInches(long microseconds) {
  // According to Parallax's datasheet for the PING))) , there are
  // 73.746 microseconds per inch (i.e. sound travels at 1130 feet per
  // second). This gives the distance travelled by the ping, outbound
  // and return, so we divide by 2 to get the distance of the obstacle.
  // See: http://www.parallax.com/dl/docs/prod/acc/28015-PING-v1.3.pdf
  return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
  // The speed of sound is 340 m/s or 29 microseconds per centimeter.
  // The ping travels out and back, so to find the distance of the
  // object we take half of the distance travelled.
  return microseconds / 29 / 2;
}

```

内臓赤外線距離センサ …赤外線 LED と明るさセンサで測距

● ArduBlock サンプルプログラム

- ・ Block ソースコードは、PC/MyDocuments/Arduino/へ配置したサンプルフォルダ [RDC_ArduBlockSamples] に、ファイル名【04_RDC_IRdistance.abp】で格納されています。



RDC-104
内臓赤外線距離センサ



ボードの赤外線 LED と明るさセンサを使って対象物までの距離を測ります。

10cm 程度までの距離を計測できます。

数値（単位はありません）をシリアルモニタに出力します。

対象物が黒いなど、赤外線を反射しないと計測できません。

■ Arduino C のサンプルプログラム

スケッチブック >RDC_samples>RDC_IR_distance（赤外線 LED と明るさセンサで測距する）

```

// Definitions.
const int BUFFER_SIZE = 45;    // use an odd number

// Global variables.
double buf[BUFFER_SIZE];    // Analog readings at 100khz & stored
    here
double out[BUFFER_SIZE];    // output of filter stored here.
int buffer_index;          // Interrupt increments buffer
boolean buffer_full;       // Flag for when complete.

double a0,a1,a2,b1,b2;      // filter kernel poles
double f,bw;                // frequency cutoff and bandwidth
double r,k;                 // filter coefficients

int LEDonoff;

// the setup routine runs once when you press reset:
void setup() {
    for( int i = 0; i < BUFFER_SIZE; i++ ) {
        buf[i] = 0;
        out[i] = 0;
    }
    buffer_index = 0;
    buffer_full = false;

    // initialize serial communication at 9600 bits per second:
    Serial.begin(57600);
    while(!Serial){          // For STEM Du RDC

    }
    pinMode(11,OUTPUT);
    pinMode(12,INPUT_PULLUP);

    // Lets sort out the filter variables before we end setup.

    // Cut-off frequency.
    // We are looking for a 38khz IR TV remote.
    // F is fraction of the sample frequency
    // It has to be between 0 and 0.5.  Therefore, the interrupt
    // needs to be at least *double* the bandpass frequency.
    // I picked 100khz as a nice number to scale from.
    // So, f = (100khz * 0.38) = 38Khz
    //f = 0.38;
    f = 0.4;
    // Bandwidth (allowance) of bandpass filter.

```



```

// Same principle as above (fraction of 100khz).
// We are using this filter to get rid of ambient environment
// noise. 20khz seems like a big band, but I wouldn't expect
// there to be much in the khz. You can fine tune downwards.
//bw = 0.2;
bw = 0.2;

// Maths. Read the book. Does the trick.
r = 1 - ( 3 * bw );
k = 1 - ( 2 * r * cos( 2 * PI * f ) ) + ( r * r );
k = k / ( 2 - ( 2 * cos( 2 * PI * f ) ) );

a0 = 1 - k;
a1 = ( 2 * ( k - r ) ) * ( cos( 2 * PI * f ) );
a2 = ( r * r ) - k;
b1 = 2 * r * cos( 2 * PI * f );
b2 = 0 - ( r * r );

LEDonoff = 0;
}

// the loop routine runs over and over again forever:
void loop() {
  float output;

  // read the input on analog pin A2 of Ph.T:
  //int sensorValue = analogRead(A2);

  if( buffer_index >= BUFFER_SIZE ) {
    buffer_index = 0;
    buffer_full = true;
  }
  else if ( buffer_full == false ){
    buf[ buffer_index ] = (double)analogRead(A2); // For STEM Du
RDC-104. A4 for RDC_103
    buffer_index++;
  }

  // print out the value you read:
  //Serial.println(sensorValue);

  if( buffer_full == true ) {

    // Run the input buffer through the filter
    output = doFilter();
  }
}

```

```

// We are going to transmit as an integer
// Move up the decimal place
//output *= 1000;
//output -= 50;
output *= 1000;
output -= 600; // Offset tuning
//Serial.print( (int)output );
//Serial.print(",");
output = 12000/sqrt(output); // Amplify
Serial.println( (int)output );

// Reset our buffer and interupt routine
buffer_index = 0;
buffer_full = false;
}

if(digitalRead(12)<1){
  analogWrite(11,0);
}
else{
  int slider = analogRead(A3); // For STEM Du RDC-104. A5 for
RDC_103
  analogWrite(11,map(slider,0,1023,0,255));
}
}

// This filter looks at the previous elements in the
// input stream and output stream to compound a pre-set
// amplification. The amplification is set by a0,a1,a2,
// b1,b2. Please see the linked book, above.
double doFilter() {
  int i;
  double sum;

  // Convolute the input buffer with the filter kernel
  // We work from 2 because we read back by 2 elements.
  // out[0] and out[1] are never set, so we clear them.
  out[0] = out[1] = 0;

  for( i = 2; i < BUFFER_SIZE; i++ ) {
    out[i] = a0 * buf[i];
    out[i] += a1 * buf[i-1];
    out[i] += a2 * buf[i-2];
    out[i] += b1 * out[i-1];
  }
}

```

```

    out[i] += b2 * out[i-2];
}

// Bring all the output values above zero
// To get a well reinforced average reading.
for( i = 2; i < BUFFER_SIZE; i++ ) {
    if( out[i] < 0 ) out[i] *= -1;
    sum += out[i];
}
sum /= BUFFER_SIZE -2;

return sum;
}

```

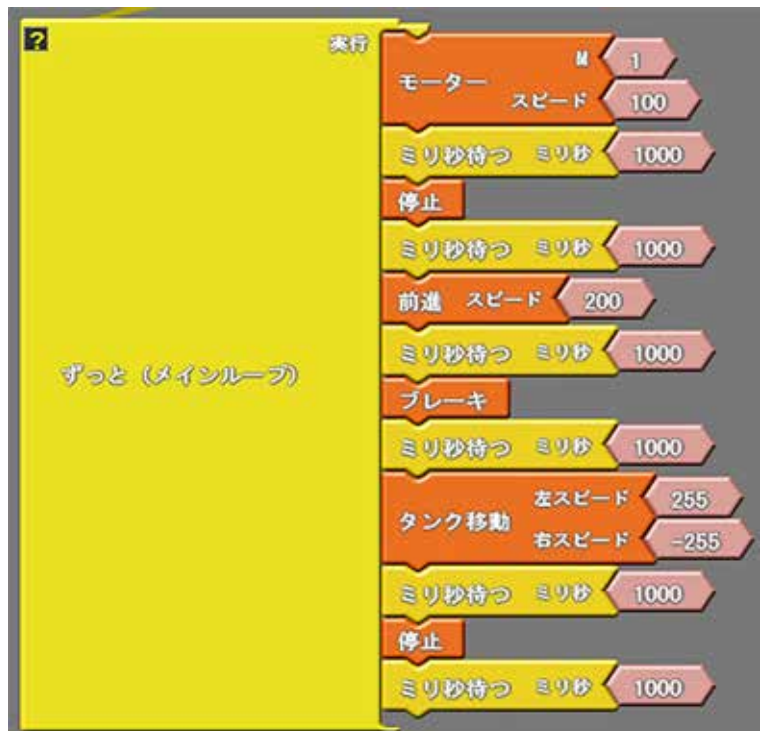
モータ・・・モータを回転する / 前進

● ArduBlock サンプルプログラム

・Block ソースコードは、PC/MyDocuments/Arduino/へ配置したサンプルフォルダ [RDC_ArduBlockSamples] に、ファイル名【O5_RDC_Motor.abp】で格納されています。



STEM Du 出力系のブロックでモータを動かします。
 スピードの PWM 値は -255 ~ 255 です。



■ Arduino C のサンプルプログラム

スケッチブック >RDC_samples>RDC_Motor (モータを回転する / 前進)
 M1,M2 コネクタに接続したモータを回転させます。

```

/*
DC motor control sample for RDC.
6.2018 Atsushi Hasegawa

Connect the motor to the M1/M2 socket.

```

```

Motor driver setting H:HIGH L:LOW
IN1 IN2 PWM
H H H/L short brake
L H H CCW(counterclockwise) /PWM = L short brake
H L H CW(clockwise) /PWM = L short brake
L L H stop

```

```
*/
```

```
// Motor driver pin assign for RDC-104 M1/M2.
```

```
int M1_1 = 4;
int M1_2 = 9;
int M1_PWM = 6;
int M2_1 = 7;
int M2_2 = 8;
int M2_PWM = 5;
```

```
/*
```

```
// Motor driver pin assign for RDC-103 M1/M2.
```

```
int M1_1 = 4;
int M1_2 = 5;
int M1_PWM = 6;
int M2_1 = 7;
int M2_2 = 8;
int M2_PWM = 9;
```

```
*/
```

```
// the setup routine runs once when you press reset:
```

```
void setup() {
```

```
    // initialize the digital pin as an output.
```

```
    pinMode(M1_1, OUTPUT);
    pinMode(M1_2, OUTPUT);
    pinMode(M1_PWM, OUTPUT);
    pinMode(M2_1, OUTPUT);
    pinMode(M2_2, OUTPUT);
    pinMode(M2_PWM, OUTPUT);
```

```
}
```

```
// the loop routine runs over and over again forever:
```

```
void loop() {
```

```
    digitalWrite(M1_1, HIGH);
    digitalWrite(M1_2, LOW);
    analogWrite(M1_PWM, 200); // PWM 0~255
```

```
    digitalWrite(M2_1, HIGH);
    digitalWrite(M2_2, LOW);
    analogWrite(M2_PWM, 200); // PWM 0~255
```

```

}

/*
// Motor driver pin assign for RDC-104 M3/M4.
int M3_1 = 0;
int M3_2 = 1;
int M3_PWM = 11;
int M4_1 = 10;
int M4_2 = 12;
int M4_PWM = 13;
// Motor driver pin assign for RDC-103 M3/M4.
int M3_1 = 0;
int M3_2 = 1;
int M3_PWM = 10;
int M4_1 = 11;
int M4_2 = 12;
int M4_PWM = 13;
*/

```

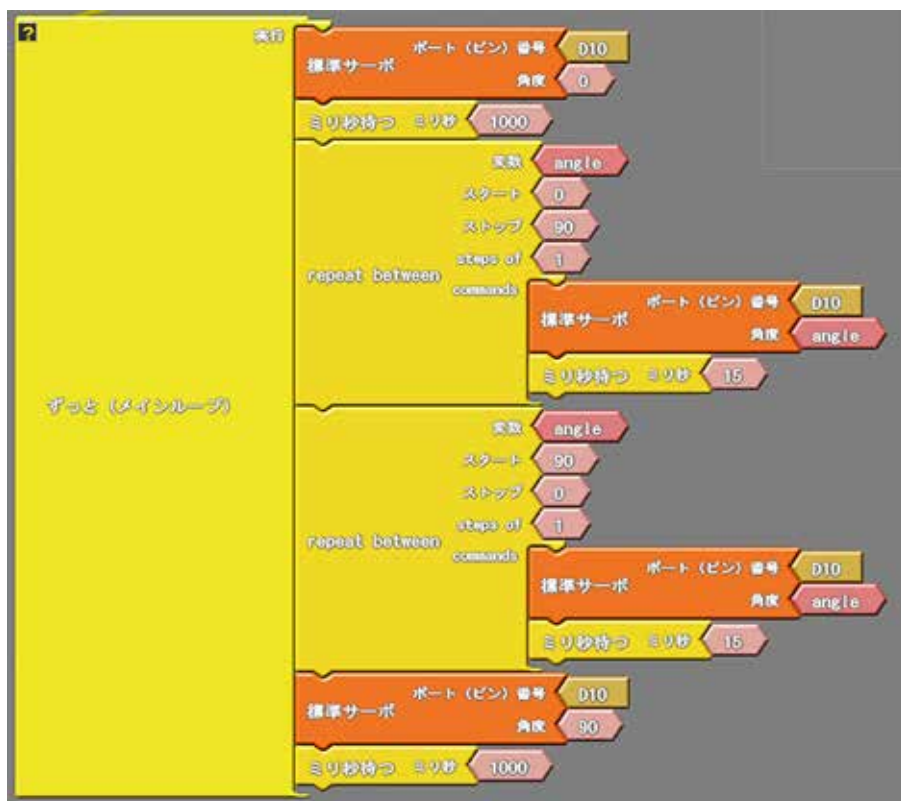
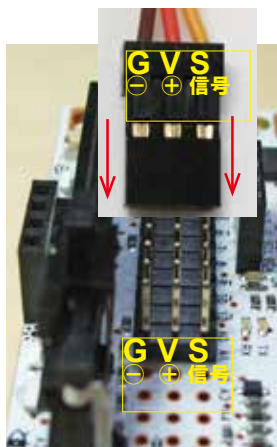
サーボ . . . サーボを回転する

● ArduBlock サンプルプログラム

・Block ソースコードは、PC/MyDocuments/Arduino/へ配置したサンプルフォルダ [RDC_ArduBlockSamples] に、ファイル名【06a_RDC_Servo.abp】で格納されています。



RDC-104



STEM Du 出力系の標準サーボブロックでサーボを動かします。単に待ち時間を入れるとサーボ固有の速度で指定位置まで動きます。リピートを使って細かいステップごとに待ち時間を入れるとゆっくり動かすことができます。

■ Arduino Cのサンプルプログラム

スケッチの例 > Servo > Sweep (サーボが回転する)

myservo.attach(9) のピン番号をサーボを接続したピンの番号に修正する。

```
#include <Servo.h>

Servo myservo;           // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;             // variable to store the servo position

void setup()
{
    myservo.attach(9);   // 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);                   // waits 15ms for the servo to reach the position
    }
    for (pos = 180; pos >= 0; 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
    }
}
```

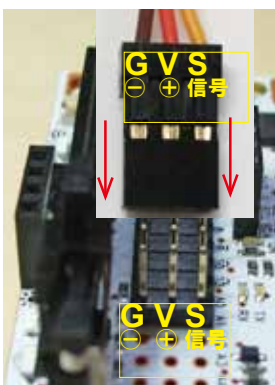
ピン番号を9から接続したピン番号に修正する

サーボ II … スライダーの位置に応じてサーボが回転

● ArduBlock サンプルプログラム



RDC-104



・Block ソースコードは、PC/MyDocuments/Arduino/ へ配置したサンプルフォルダ [RDC_ArduBlockSamples] に、ファイル名【06b_RDC_Servo_Slider.abp】で格納されています。



STEM Du 出力系の標準サーボブロックでサーボを動かします。スライダーの値をサーボの駆動範囲にマッピングして出力します。

■ Arduino C のサンプルプログラム

スケッチの例 > Servo > Knob (スライダの位置に応じてサーボが回転する)

myservo.attach(9) のピン番号をサーボを接続したピンの番号に修正する。

potpin = 0 のピン番号を 0 から A3 に修正する。



```
#include <Servo.h>

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

int potpin = A3; // analog pin used to connect the potentiometer
                (ピン番号をスライダピン番号 A3 に修正する)

int val; // variable to read the value from the analog pin

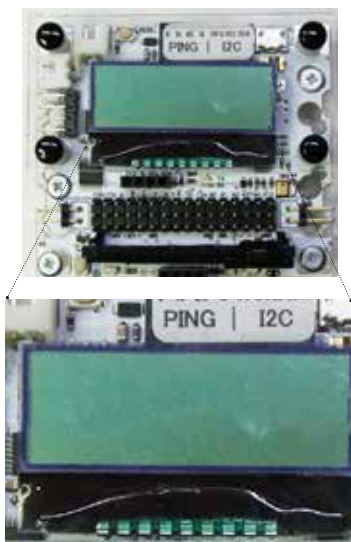
void setup()
{
  myservo.attach(13); // attaches the servo on pin 9 to the servo object
                    (ピン番号をサーボ接続したピン番号に修正する)
}

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
}
```

I2CLCD (RDC-104 用) ・ ・ ・ 液晶に文字表示する

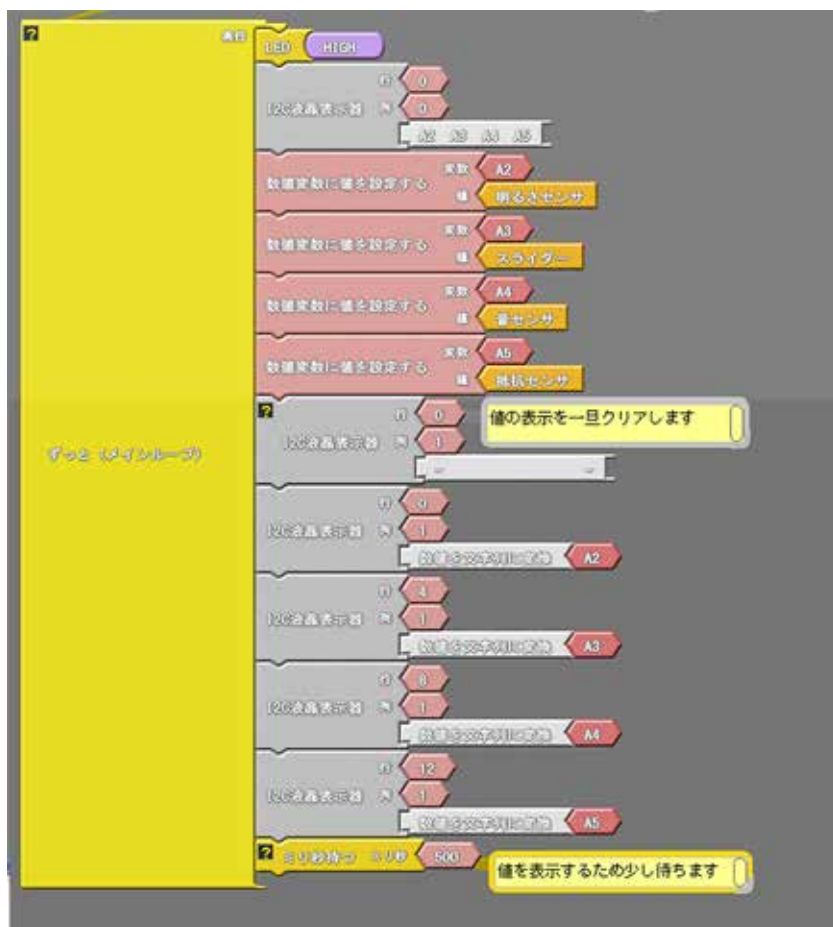
● ArduBlock サンプルプログラム

・ Block ソースコードは、PC/MyDocuments/Arduino/へ配置したサンプルフォルダ [RDC_ArduBlockSamples] に、ファイル名【07_RDC_I2CLCD_Sensors.abp】で格納されています。



RDC-104TYPE2, TYPE3+(plus)には、液晶が搭載されています。

RDC-104 の I2C 接続の液晶に文字を表示します。アナログ入力 of 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。



！ I2C 液晶表示器ブロックは製作中です。
！ 現状は行と列の指定が逆になっていますのでご注意ください。

■ Arduino C のサンプルプログラム

スケッチブック > RDC_samples > RDC_I2CLCD_Sensors (アナログセンサモニタ)

```
// include the library code:
#include <I2CLiquidCrystal.h>
#include <Wire.h>

// initialize the library
// uncomment next line if you are using a LCD from Strawberry Linux
I2CLiquidCrystal lcd(40, false);
// | +--- set true if the power suply is 5V, false if it is 3.3V
// +----- contrast (0-63)
// uncomment next line if you are using a LCD from Akizuki denshi
// I2CLiquidCrystal lcd;

void setup() {
```



```
// set up the LCD' s number of columns and rows:
lcd.begin(16, 2);
// Print pin numbers
lcd.print("  A2  A3  A4  A5");

pinMode(0, INPUT_PULLUP);
pinMode(1, OUTPUT);
pinMode(12, INPUT_PULLUP);

pinMode(13, OUTPUT);
digitalWrite(13,HIGH);
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);

  for (int i=A2; i<=A5; i++) {
    int a = analogRead(i);
    if (a<1000)
      lcd.print(' ');
    if (a<100)
      lcd.print(' ');
    if (a<10)
      lcd.print(' ');
    lcd.print(a);
  }
  delay(100);
}
```

加速度 / ジャイロセンサ ・ ・ ・ MPU-6050 シリアルモニタ

● ArduBlock サンプルプログラム

・Block ソースコードは、PC/MyDocuments/Arduino/へ配置したサンプルフォルダ [RDC_ArduBlockSamples] に、ファイル名【09_RDC_MPU6050.abp】で格納されています。



RDC-104



RDI-9250 4本のヘッダーピンで取り付け使用します。



VCC, GND, SCL, SDA コントローラソケットの記号と合わせて差し込み接続をして使用します。



接続状態図

STEM Du 入力系のブロックを使います。加速度 / ジャイロセンサ MPU6050 からシリアルモニタに出力します。

！加速度センサブロックは、現状プログラムの中で一つの値しか出力することができません。

！複数入れるとコンパイルでエラーになります。

！複数のパラメータを使用する場合はC言語で記述してください。

■ Arduino C のサンプルプログラム

スケッチブック >RDC_samples>RDC_MPU6050_test

```
// 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() {
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0); // set to zero (wakes 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。

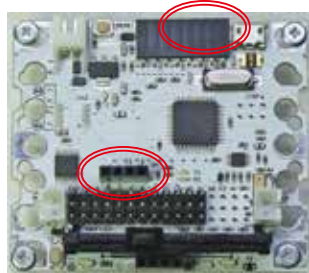
図は、ArduinoCのサンプルMPU6050testでのシリアルモニターです。

I2C コンパスセンサ・・・1 度単位で角度を取得できます。

● ArduBlock サンプルプログラム

電子コンパスモジュール 3 軸コンパス 磁気センサー

- ・ Block ソースコードは、PC/MyDocuments/Arduino/ へ配置したサンプルフォルダ [RDC_ArduBlockSamples] に、ファイル名【08_RDC_QMC5883.abp】で格納されています。



RDC-104

コントローラボード	コンパスセンサ
3V3	VCC
3SCL	SCL
2SDA	SDA
G	GND

コンパスセンサはピンの表示を合わせて I2C ソケットに挿します。

！コンパスセンサブロックは製作中です。(エラーが出てコンパイルできません)

！C 言語のサンプルを使用してください。

X 方位	出力値	分解能
北 N	0	360
東 E	90	1 度単位で
南 S	180	出力
西 W	270	



STEM Du アクセサリのコンパスセンサブロックを使います。
QMC5883 コンパスセンサの値をシリアルモニタに出力します。

Serial print例

```
Heading = 3.76 Degree = 215.19
Heading = 3.02 Degree = 172.90
Heading = 0.81 Degree = 46.65
```

■ Arduino C のサンプルプログラム

スケッチの例 >[DFRobot_QMC5883]>【QMC5883_compass】で格納されています。

■同じ場所に 3 軸を計測するサンプルソースコード【QMC5883_readRaw】が格納されています。

```
/*!
 * @file QMC5883_compass.cpp
 * @brief The program shows how to realize the function compass. When
 the program runs, please spin QMC5883 freely to accomplish calibration.
 * @n 3-Axis Digital Compass IC
 *
 * @copyright [DFRobot] (http://www.dfrobot.com), 2017
 * @copyright GNU Lesser General Public License
 *
 * @author [dexian.huang] (952838602@qq.com)
 * @version V1.0
 * @date 2017-7-3
 */
```

```

#include <Wire.h>
#include <DFRobot_QMC5883.h>

DFRobot_QMC5883 compass;

void setup()
{
  Serial.begin(115200);
  while (!compass.begin())
  {
    Serial.println("Could not find a valid QMC5883 sensor, check wiring!");
    delay(500);
  }

  if(compass.isHMC()){
    Serial.println("Initialize HMC5883");
    compass.setRange(HMC5883L_RANGE_1_3GA);
    compass.setMeasurementMode(HMC5883L_CONTINUOUS);
    compass.setDataRate(HMC5883L_DATARATE_15HZ);
    compass.setSamples(HMC5883L_SAMPLES_8);
  }
  else if(compass.isQMC()){
    Serial.println("Initialize QMC5883");
    compass.setRange(QMC5883_RANGE_2GA);
    compass.setMeasurementMode(QMC5883_CONTINUOUS);
    compass.setDataRate(QMC5883_DATARATE_50HZ);
    compass.setSamples(QMC5883_SAMPLES_8);
  }
}

void loop()
{
  Vector norm = compass.readNormalize();

  // Calculate heading
  float heading = atan2(norm.YAxis, norm.XAxis);

  // Set declination angle on your location and fix heading
  // You can find your declination on: http://magnetic-declination.com/
  // (+) Positive or (-) for negative
  // For Bytom / Poland declination angle is 4'26E (positive)
  // Formula: (deg + (min / 60.0)) / (180 / M_PI);
  float declinationAngle = (4.0 + (26.0 / 60.0)) / (180 / PI);
  heading += declinationAngle;

  // Correct for heading < 0deg and heading > 360deg
  if (heading < 0){
    heading += 2 * PI;
  }
}

```

```

if (heading > 2 * PI) {
  heading -= 2 * PI;
}

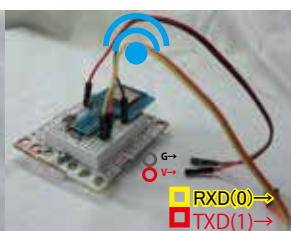
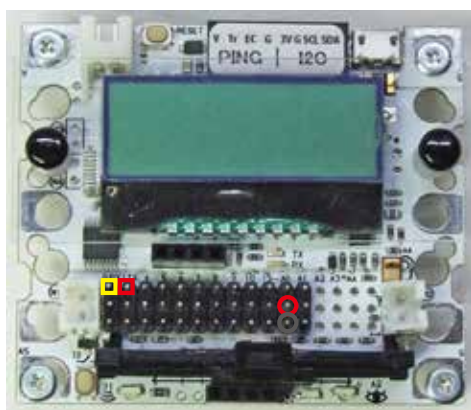
// Convert to degrees
float headingDegrees = heading * 180/M_PI;

// Output
Serial.print(" Heading = ");
Serial.print(heading);
Serial.print(" Degress = ");
Serial.print(headingDegrees);
Serial.println();

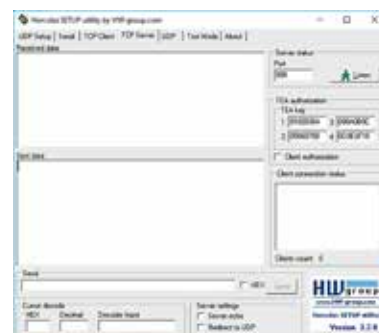
delay(100);
}

```

IoT・・・[拡張]Wi-Fiでシリアルモニターをする。



拡張 WiFi モジュール
RDP-TEC31-WiFi



windows 側は「Hercules SETUP utility」というソフトを使います。<https://www.hw-group.com/software/hercules-setup-utility> からソフトをダウンロードします。

添付のプログラムに使用する無線ルータの SSID、パスワードと、使用する PC の IPv4 アドレスを入力します。

入力したプログラムをマイコンボードに書き込んで、シリアルモニタを開くとプログラムが実行されます。

シリアルモニタに AT コマンドの接続の状態が表示されて透過モードで待機状態になります。arduino 側、Hercules SETUP utility 側でそれぞれ文字入力すると相互に表示されます。プログラムの loop の中のコメントアウトされているところのコメントを外して書き込むとデータを連続して送ります。

ソースコードは、PC/MyDocuments/Arduino/ スケッチブック >RDC_samples> に、ファイル名【10_RDC_WiFi_Serial】で格納されています。Arduino で、RDC_samples の中に配置したサンプルを開くと確認できます。

マイコンボード側プログラム

```

/*
multiple serial test for RDC

Receives from the USB serial, sends to hardware serial.
Receives from hardware serial, sends to USB serial.

```

```

The circuit:
* RX is digital pin 0 (connect to TX of other device)
* TX is digital pin 1 (connect to RX of other device)
*/

void setup()
{
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while(!Serial); // for 32U4 シリアルモニタを起動するまで待つ

  Serial.println("Goodnight moon!");

  // set the data rate for the Serial1 port
  Serial1.begin(9600);
  Serial1.println("AT");
  for(int i = 0; i < 5000; i++) {
    serialLoop();
  }

  // delay で待つとシリアル処理ができないので、for でループして待つ 待ち時間は機材条件によって調整する
  Serial1.println("AT+CWMODE=1");
  for(int i = 0; i < 5000; i++) {
    serialLoop();
  }
  Serial1.println("AT+CWJAP_CUR=\"無線ルータ SSID\", \"パスワード\"");
  for(int i = 0; i < 80; i++) {
    for(int ii = 0; ii < 10000; ii++) {
      serialLoop();
    }
  }
  Serial1.println("AT+CIPSTART=\"TCP\", \"windows の IPv4 アドレス\", 999");
  for(int i = 0; i < 10000; i++) {
    serialLoop();
  }
  Serial1.println("AT+CIPMODE=1");
  for(int i = 0; i < 5000; i++) {
    serialLoop();
  }
  Serial1.println("AT+CIPSEND");
  for(int i = 0; i < 5000; i++) {
    serialLoop();
  }
}

void loop() // run over and over
{
  Serial.println("send message and data");
  serialLoop();
}

```

```
void serialLoop()
{
  if (Serial1.available())
    Serial.write(Serial1.read());
  if (Serial.available())
    Serial1.write(Serial.read());
}
```

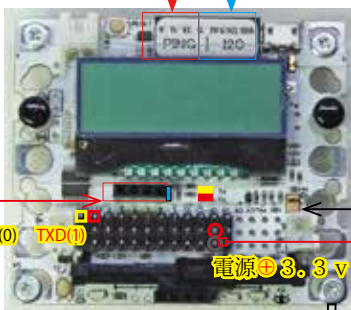
マイコンボードのセンサ情報を、Wi-Fi でリアルモニターします。

⇒電源 ⊕ 3.3V ⊖ G

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



PING 11 Trig Echo I2Cコネクタ SCL, SDA



12 ボタン button 赤外線LED 11 Infrared sensor
加速/度/ ジャイロ/ 温度センサ Accelerometer/Gyroscope (別売品) (I2C) ソケット接続使用



加速/度/ ジャイロセンサ取付状態図

・RDC104：内蔵赤外線距離センサ / 明るさセンサ / 音センサを、内蔵しています、その他の計測の場合、必要なセンサをマイコンボードに接続してください。アナログセンサ 2 系統入力 (A0,A1) 接続可能。加速度センサ / ジャイロセンサ / 温度センサも I2C 接続可能です。

・音センサ
アナログセンサ入力ピン端子
今回 WiFi モジュールの 3V3,GND を A0,A1 端子 ⊕ 3.3V、⊖ G に接続

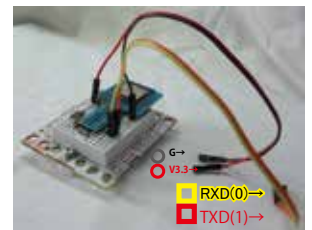
・みのむしクリップ端子

・明るさセンサ



拡張
Wi-Fi モジュールキット
RDP-TEC31-WiFi

✓ RDC104 RXD(0) TXD(1)

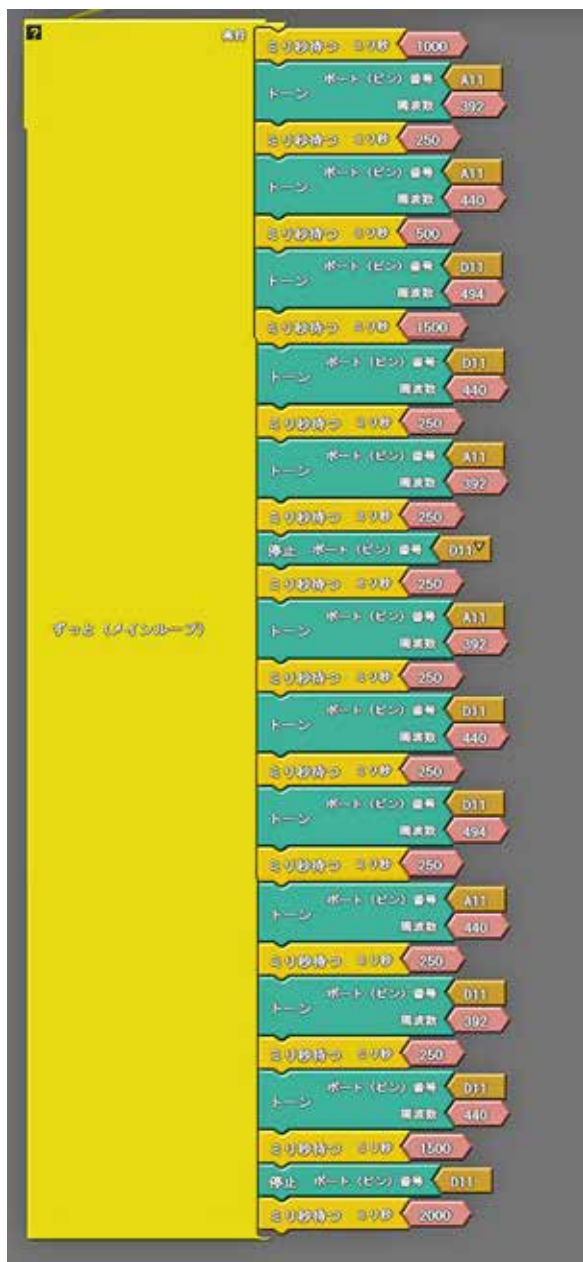


搭載ロボット例：RDS-TEC31WiFi

ブザー・・・ブザーでメロディを出力する

● ArduBlock サンプルプログラム

・Block ソースコードは、PC/MyDocuments/Arduino/へ配置したサンプルフォルダ [RDC_ArduBlockSamples] に、ファイル名【10_RDC_tone.abp】で格納されています。



トーンブロックを使ってブザーの音を出します。

RDC-104では超音波(PING)のソケットのTr(11番ピン)とGに圧電ブザーを接続します。



RDC-103では11番ピンに圧電ブザーが付いています。



■ Arduino Cのサンプルプログラム

スケッチの例 > 02.Digital > toneMelody(ブザーでメロディを出力する)

tone(8, melody[thisNote], noteDuration) と noTone(8) のピン番号を 8 から 11 に修正する。

```
#include "pitches.h"

// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};
```

```
// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(8);
  }
}

void loop() {
  // no need to repeat the melody.
}
```

ピン番号 8 を 11 に修正する

ピン番号 8 を 11 に修正する

3-5. パーツアクセサリ

RoboDesignerPlus RDC-102,103,104 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 ROBOTTECH のホームページでご確認ください。

1. レバースイッチ / コネクタ付 6V 電池ケース



RDP-8093x4LSWP ¥300_ (税抜)

単 3 x 4 本電池ケース、コード長さ 15 cm、端末 XH 型 2 ピンコネクタ付、レバースイッチ付

2. レバースイッチ / コネクタ付 4.5V 電池ケース



RDP-8093x3LSWP ¥300_ (税抜)

単 3 x 3 本電池ケース、コード長さ 15 cm、端末 XH 型 2 ピンコネクタ付、レバースイッチ付

3. USB 単 3 x 4 本電池ケース RDP-8093x4USB



¥900_ (税抜)

単 3 x 4 本電池ケース、USB コネクタ付き、スイッチ付

4. マイクロ USB 接続コード RDP-824 ¥400_ (税抜)



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

5. 電池ケース コネクタ変換コード RDP-825



2 本組 ¥400_ (税抜)

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 10 cm 加工用熱収縮チューブ付

6. DC ジャック -2 ピン変換ケーブル RDP-826



¥400_ (税抜)

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 11 cm AC アダプター接続時に使用

7. センサケーブル 20 cm RDP-831 ¥400_ (税抜)



30 cm RDP-832 ¥400_ (税抜)

両端 XH3 極コネクタ付

8. センサ延長ケーブル RDP-829 ¥400_ (税抜)



4 芯 QL オスマス端子 20 cm

9. モーターケーブル 20 cm RDP-833 ¥400_ (税抜)



30 cm RDP-834 ¥400_ (税抜)

XH2 極 - QL2 極コネクタ付

10. ジャンプケーブル RDP-836 ¥400_ (税抜)



20 cm ケーブル
両端 QL メスプラグ付
10 本セット

11. マルチメディアカードスロット MMC-DM3AT ¥300_ (税抜)



RDC-103 データ記録に使用。
取付は、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)



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 30 cm
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-502-48



¥1,800_ (税抜)

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。
モータ回転数 6,720rpm 減速比
1/48 出力軸回転数 140rpm
電源電圧 DC4.5V 付属ケーブル
20 cm

14. ホイール付ギアードモータ RDO-502-120



¥1,800_ (税抜)

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。
モータ回転数 6,720rpm 減速比
1/120 出力軸回転数 56rpm
電源電圧 DC4.5V 付属ケーブル 20
cm

15. エンコーダ付ギアードモータ RDO-502EN-48



¥3,000_ (税抜)

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 20 cm XH3P-QL3P 電源
電圧 DC4.5V 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
モータ回転数 6,720rpm 減速比
1/48 出力軸回転数 140rpm

16. エンコーダ付モータ RDO-29BMA ¥3,700_ (税抜)



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 30 cm 先バラ

17. エンコーダ・ギアヘッド付 DC モータ RDO-29B50G27A RDO-29B50G 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 30 cm 先バラ



18. 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



19. タッチセンサー JES-7022

¥1,200_ (税抜)

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 30 cm



20. アナログ赤外線センサー JES-7023 VAD

¥1,500 (税抜)

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。垂直取付可 30 cm ケーブル付



21. 変調赤外線センサー RDI-203JR

¥1,200_ (税抜)

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。30 cm ケーブル付



22. 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。30 cm ケーブル付



23. 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。

30 cm ケーブル付



24. 超音波センサー RDI-HCSR04

¥1,200_ (税抜)

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: 0.3 cm 測距範囲: 2 ~ 450 cm



25. I²C コンパスセンサ RDI-5883L_QMC

¥1,800_ (税抜き)

地磁気を計測、分解能 360、出力 1 度単位。I2C ケーブル付属です。ケーブル長 20 cm



26. ジャイロセンサ RDI-9250

RDI-9250

¥1,800_ (税抜き)

9 軸 加速度 ジャイロ コンパス 磁気センサ

4 ピンヘッダ取り付け済み。

予備のピンヘッダと、オスメスの 20cm ワイヤが付属します。



27. 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 1 mm ケーブル長 30 cm



28. R/G/B LED ボード RDO-404

¥5,800_ (税抜き)

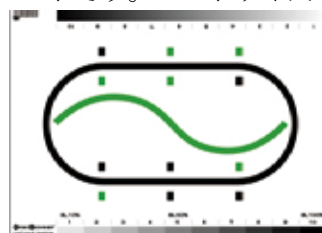
水耕栽培実験などに利用できる LED 照明ボードです。切替スイッチで赤/緑/青の 3 色とその組み合わせで発光できます。



29. ライントレース&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、ライン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 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)



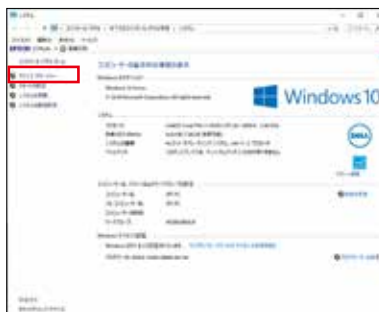
How to use the program environment

Program development environment use preliminary preparations

(Program environment Scratch,Arduino,ArduBlook and RDC-104 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]. 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。

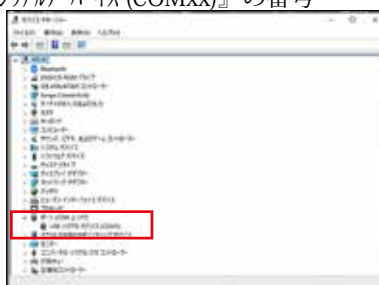


[1]. PC and a circuit board are connected.

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]. COMポートの確認【Windows10の場合】

1. PCの[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内の [デバイスマネージャー] を選択します。
3. ポート (COM と LPT) をクリックし『USB シリアルデバイス (COMxx)』の番号を確認し、記録します。



[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-104 (COMxx)" is confirmed and it's recorded.

[3]. プログラム環境を起動

1. デスクトップに配置した [Arduino-1.6.10-win-stemdu 16]IDE ファイルの中の [arduino.exe] をクリックして開発環境をスタートさせます。
2. ARDUINO Genuine が起動し、プログラム開発環境画面が出現します。



(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.

[4]. Arduino-IDE のボードマネージャ設定

1. 起動した Arduino-IDE の [ツール] > [ボードマネージャー] をクリックし、出現するマイコンボードリスト、Arduino AVR ボードで、使用するマイコンボードに合わせて [STEM Du/RoboDesigner+RDC 102w/ ATmega32U4 3.3V 8MHz] [STEM Du/RoboDesigner+RDC103R4w/ ATmega32U4 3.3V 8MHz] [STEM Du/RoboDesigner+RDC 104w/ ATmega32U4 3.3V 8MHz] のいずれかをクリックし選択します。リスト左端に●印が付きます。



3. [STEMDu/RoboDesigner+RDC-104w/ ATmega32U4 3.3V 8MHz] is chosen and click designation is performed by the microcomputer board list which clicks is Arduino-IDE [tool] > [microcomputer board] and appears. A ● mark sticks to the list left end.

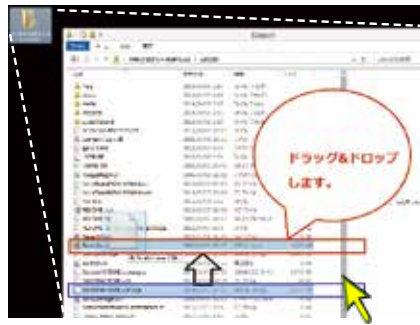
シリアルポート設定 [ツール] > [シリアルポート] のCOM番号を、調べた番号に設定します。



4.2.2. Scratch を起動する。

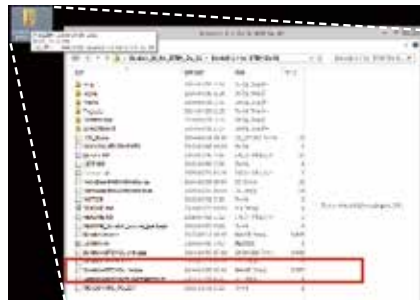
1). windows の場合：

[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、動かしてみましよう。

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よう。

・例えば複数の画像が重なったときに「ガチャン」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.

↓ Scratch which starts

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.

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.

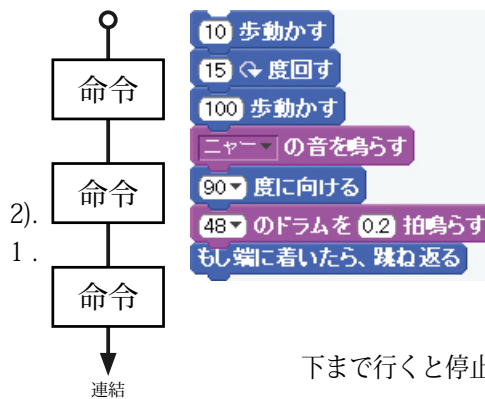
5. If you're being experienced in operation, it "is checked", you'll establish it so that a movement different according to the event may be done using some blocks.

*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.

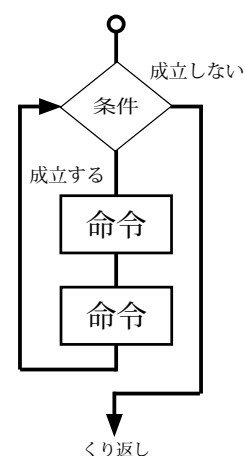
4.2.3. 命令ブロックの実行規則



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。

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。

下まで行くと停止します。



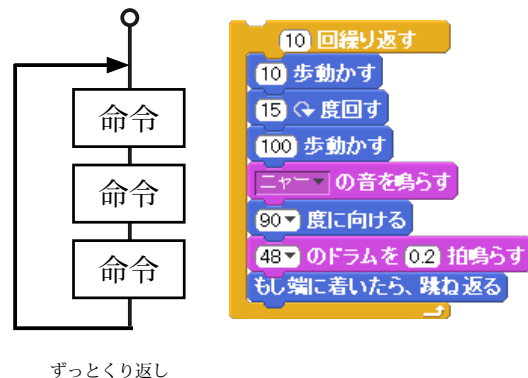
3). くり返し（条件ループ）

1. 繰り返しには、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。

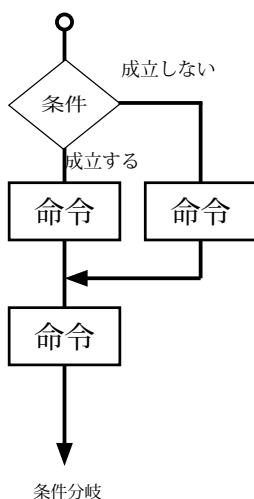
4). ずっとくり返し（無限ループ）

1. ずっと繰り返す（無限ループ）も、あります。



ずっとくり返し

5). 条件分岐（1）

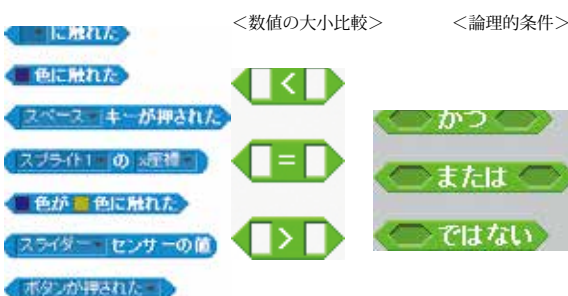


1. 条件分岐の例



<条件>

2. 条件に使うもの



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.

4). Repetition (infinite loop)

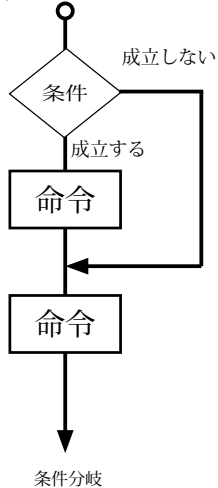
1. There is also a (infinite loop) repeated eternally.

5). Conditional branch (1)

1. An example of a conditional branch

2. Something to use for the condition

6). 条件分岐 (2)



実行をスタートします。

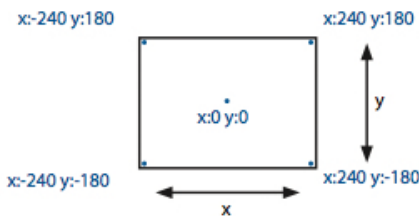
7). スタート命令ブロック緑の旗をクリックした時あるキーが押された時など、条件に応じて、スクリプトの

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) ですので、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.

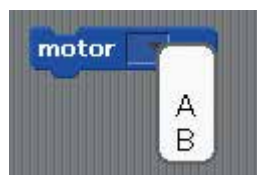
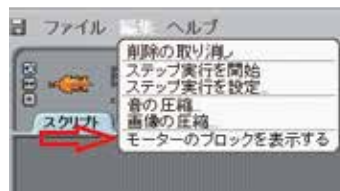
The screen center is the starting point (0,0), so please be careful.

4.2.4. ロボットで多く使うスクリプト

(1). モーター

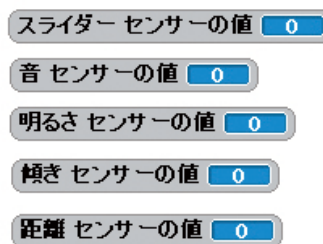
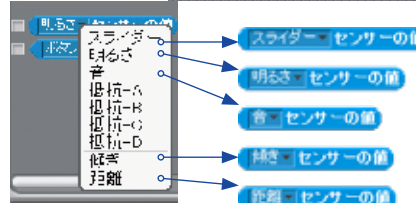
1. モータのブロックは、[編集]>[モーターのブロックを表示する]をクリックします。
2. ブロックパレットの「動き」リストに追加されます。
3. モータは A,B の 2 種類を使えます。
4. Scratch と RDC のモータ端子

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. StemDu 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.

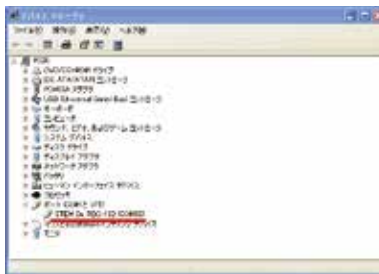
4-3. Scratch の動作実験

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 マネージャー] 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_104.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」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。



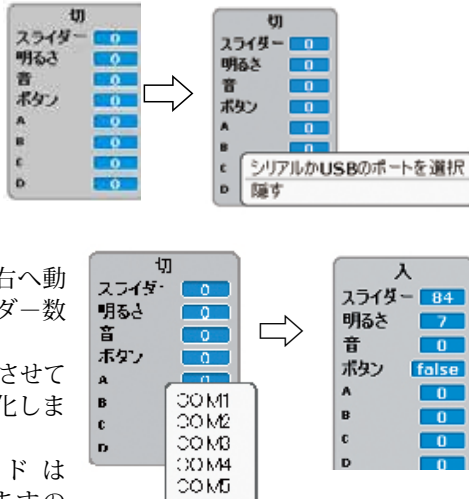
- 通信設定完了すると、「切」→「入」へ変化し、ボード搭載センサのデータ値が表示されます。

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「」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、周囲が白枠で囲まれます。
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。



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.

4.3.3. Scratch は、接続状態のまま、マイコンボードが動作。

1. Scratch では、常にパソコンとマイコンボードを USB で接続して使用します。

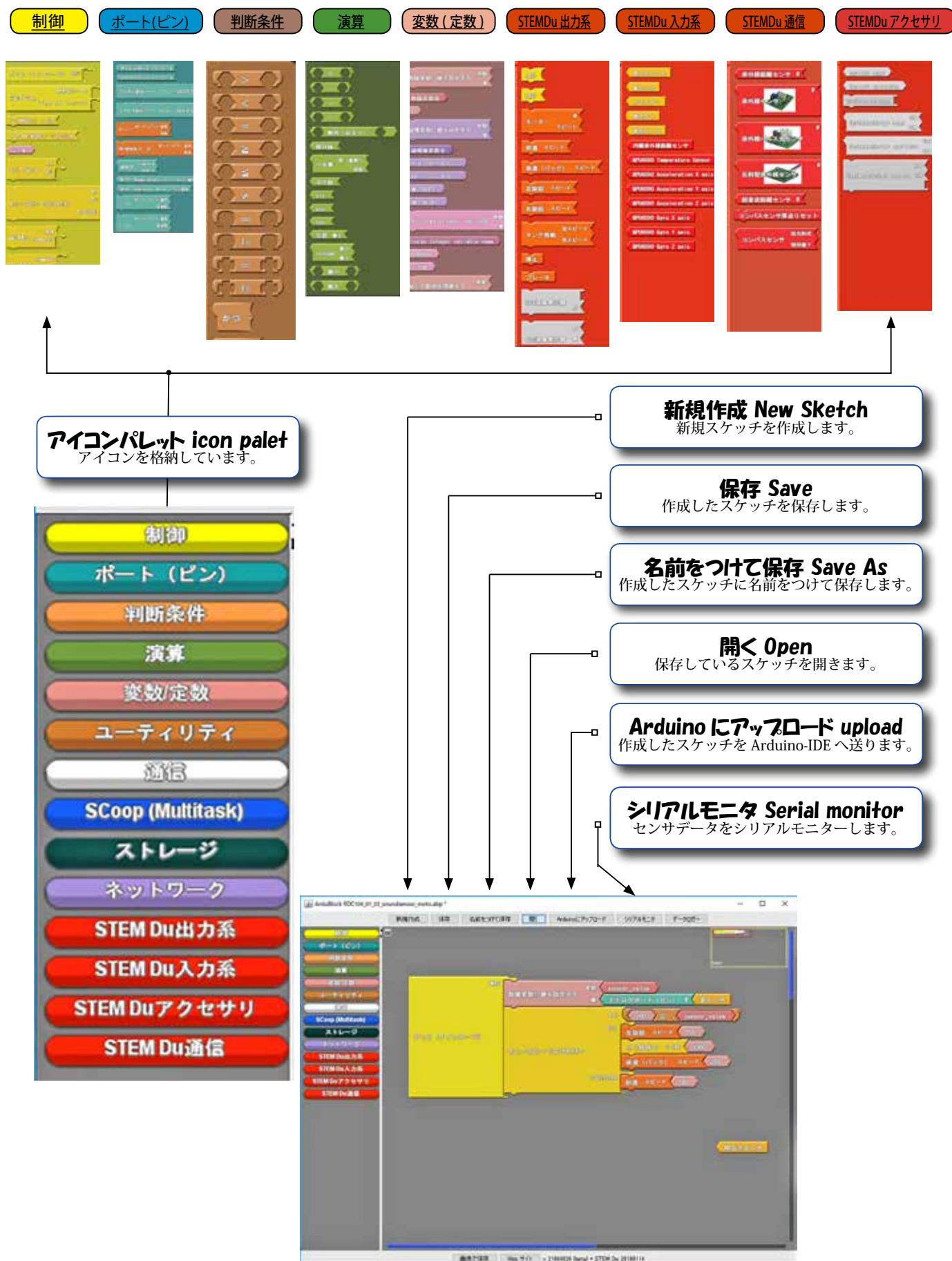


2. マイコンボードの PC 接続を外して、自律型動作をさせるときは、ArduBlock をご利用ください。



4-4. ArduBlock

4.4.1. ArduBlock 画面の構成



4.4.2 ArduBlock の使い方

[1]. Arduino プログラム環境を起動

1. デスクトップに配置した [Arduino-1.6.10-win-stemdu16]IDE ファイルの中の [arduino.exe] をクリックして開発環境をスタートさせます。



2. ARDUINO Genuino が起動し、プログラム開発環境ウインドウが出現します。



[2]. Arduino-IDE のボードマネージャ設定

1. 起動した Arduino-IDE の [ツール] >

[ボードマネージャ] をクリックし、

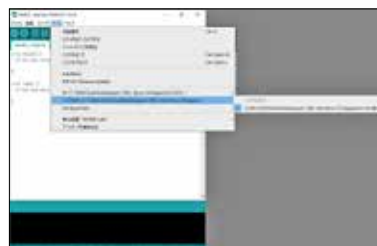
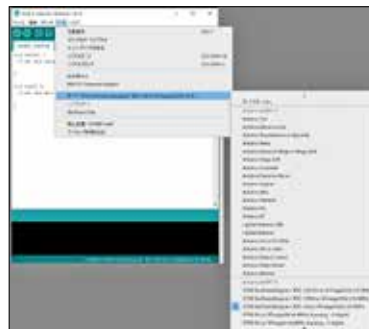
出現するマイコンボードリスト、Arduino AVR ボードで、
[STEM Du/RoboDesigner+RDC102w/ATmega32U4 3.3V 8MHz]

[STEM Du/RoboDesigner+RDC103R4w/ATmega32U4 3.3V 8MHz]

[STEM Du/RoboDesigner+RDC104w/ATmega32U4 3.3V 8MHz]

のいずれかを使用するマイコンボードに合わせてクリックし選択設定します。

リスト左端に ●印が付きます。



[3]. シリアルポート設定

[ツール] > [シリアルポート] に出現している COM 番号を確認し、調べた番号でクリック設定します。 リスト左端に ☑印が付きます。

シリアルポート

COM5 [STEM Du/RoboDesigner+RDC104w/ATmega32U4 3.3V 8MHz]

など

この「マイコンボード」「通信ポート」の2つの設定をしないと、PCのマイコンボードCOM認識が外れて通信ができなくなるなどの誤動作が発生します。

[4]. ArduBlock を起動します。

(1). Arduino-IDE を起動し、[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]
 [初期化ルーチンを伴うループ]
 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。

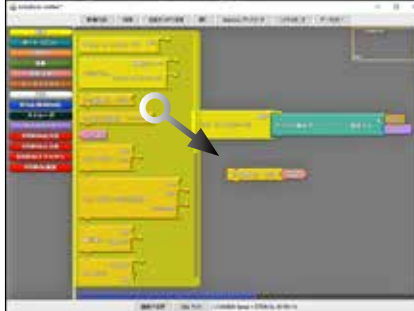


4. ループの中のパーツの位置に収まるようにドロップすると「カチッ」と音がして、ブロックが、ループにはまり込み結合します。



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。

* [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。



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。

(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.

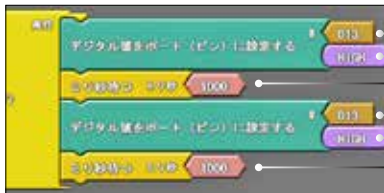
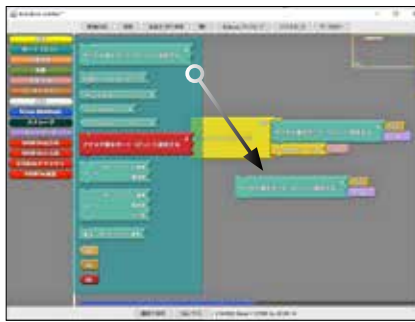
• When clicking the left side {port (pin)}, > [the digital value set as a port (pin)] from the sub-window from which emerge, you choose and do drag and drop in the center field.

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.

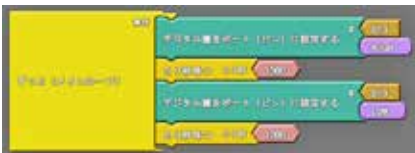
6. A block, every time it drops, a loop frame is becoming big. It changes according to the size of the program.

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



- * 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。

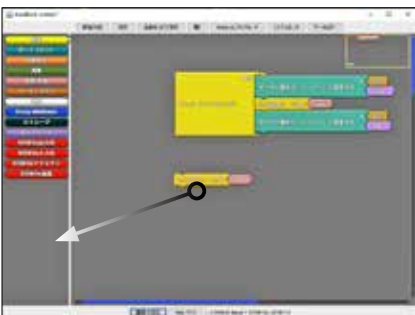
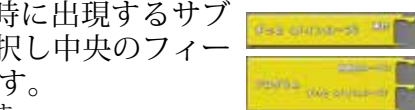
* プログラムには、[ループ] が必ず必要です。
 *[メインプログラムのループ] [初期化ルーティンを伴うループ] のいずれかのループでプログラム作成します。

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) :
- 複製 (Copy) :
- コメントを削除 (Delete Comment) :
- 複製 (Clone) :



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.

11. The operational explanation.

The program made in ArduBlock is made in the basic [loop] block.

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.

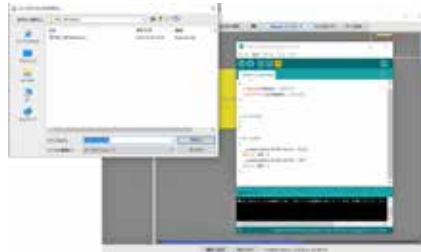
この手続きを、必ずしてください。

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。



12. 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。



13. [Arduinoへアップロード]をクリックすると、スケッチホルダーの保存ウィザードが出ますので、作成したスケッチを名前をつけて保存します。

- 保存先[PC] > [ドキュメント] > [Arduino] > [ArduBlockExamples] フォルダに保存します。
- 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。



• 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。

最大28,672バイトのフラッシュメモリのうち、スケッチが4,564バイト(15%)を使っています。
 最大2,560バイトのRAMのうち、グローバル変数が149バイト(5%)を使っていて、ローカル変数で2,411バイト使うことができます。
 指定されたポートには、ボードが接続されていません。正しいポートを選んである事を確認してください。もしも正しいポートを選んである場合には、書き込みを開始した直後にボードのリセットボタンを押してみてください。



エラーメッセージ

(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ったりすると、
- (***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] > [ボードマネージャー]メニューから接続したいマイコンボードの名前を選びクリック指定します。

AVRボードで、

- RDC102: [STEM Du/RoboDesigner+RDC102w/ATmega32U4 3.3V 8MHz]
- RDC103: [STEM Du/RoboDesigner+RDC103R4w/ATmega32U4 3.3V 8MHz]
- RDC104: [STEM Du/RoboDesigner+RDC104w/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。

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-104, [STEM Du/RoboDesigner+RDC104w/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.

(3). マイコンボードへの書き込み失敗が続く場合



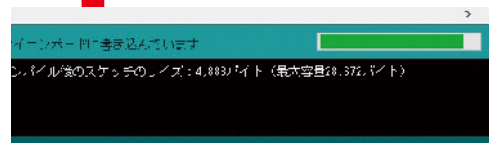
エラーメッセージ Couldn't find a Leonardo on the selected port. Check that you have the correct port selected. If it is correct, try pressing the board's reset button after initiating the upload.

出現時には、通信エラーが発生し、マイコンボードへの書き込みが失敗しています。

- 1)USB ケーブル接続確認
- 2)Arduino > [ツール] > 「ボードマネージャー」で、使用しているコントローラ RDC に●マーク、「シリアルポート」接続COM 番号に☑マークを確認ください。



※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中に、コントローラの RESET スイッチを「ダブルクリック」し、**強制アップロード**してください。



このタイミングです。



4.4.3. ArduBlock 仕様でのお断りとお断り

(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。

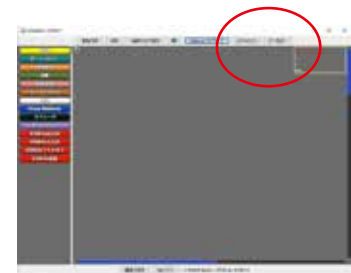
サポート対象



サポートできません。

(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.

(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。



開発中□・シリアルモニタ・データロガー
今のところ：Arduinoのツール内で、シリアルモニタ、シリアルプロッ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）

4.4.4. サンプルプログラムリスト

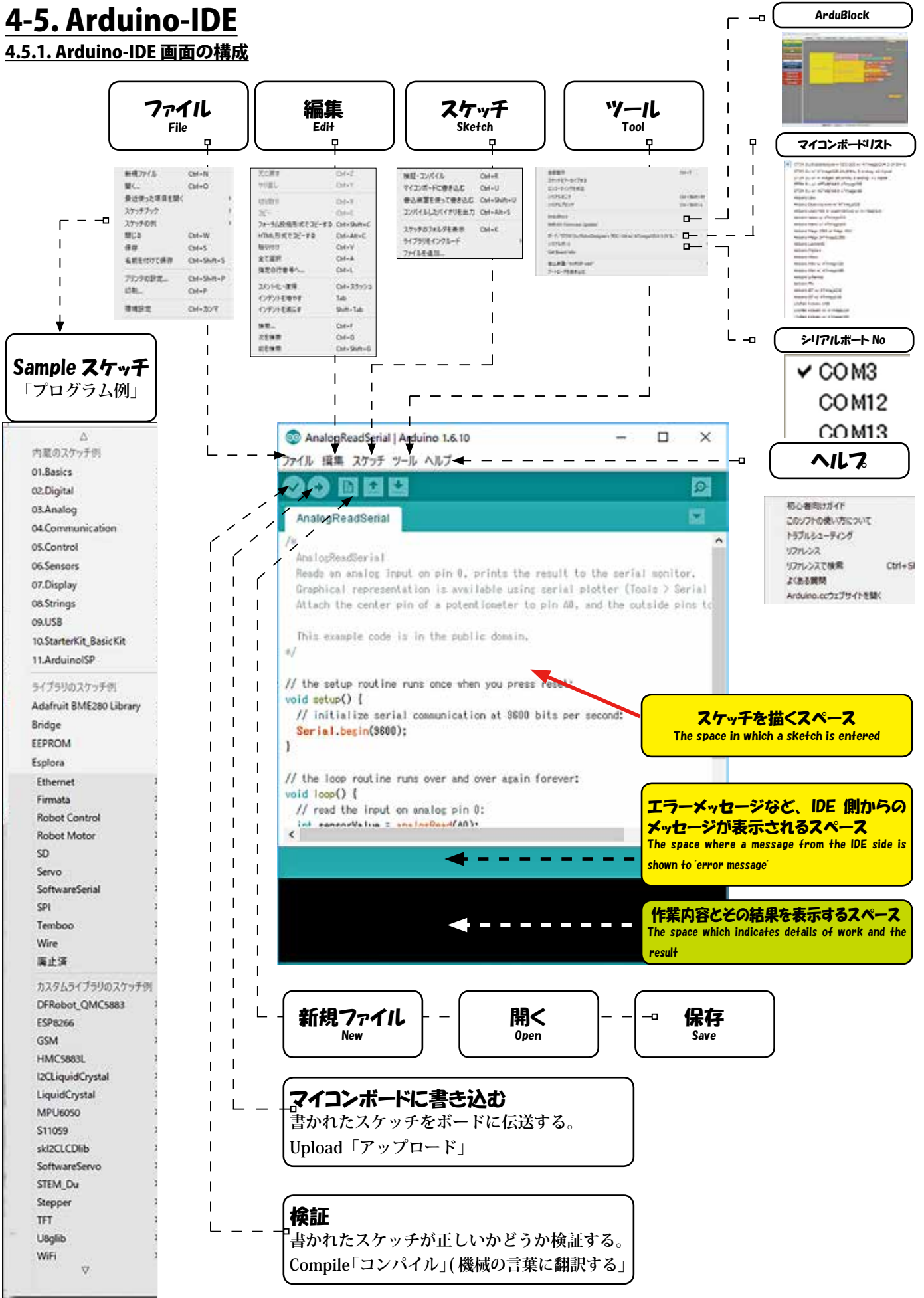
※Blockサンプルは、PC/MyDocument/Arduino へ配置したRDC_ArdublockSamples に格納されています、必要なプログラム名を指定して開きます。



ロボットモデル	Block サンプル	C サンプル	プログラム概要
Robotics experiment	RDC_ArdublockSamples > 01_RDC_Button_LED.abp	スケッチの例 > 02.Digital > DigitalInputPullup	ボタンを押すとLEDが点灯する。
Robotics experiment	RDC_ArdublockSamples > 02a_RDC_AnalogRead.abp	スケッチの例 > 01.Basics > AnalogReadSerial	センサーの値をシリアルモニターで確認する。
Robotics experiment	RDC_ArdublockSamples > 02b_RDC_AnalogReadSensors.abp	スケッチブック > RDC_samples > 02_RDC_AnalogReadSensors	A0～A5まで全てのアナログセンサの値をシリアルモニターで確認します。
Robotics experiment	RDC_ArdublockSamples > 03_RDC_PING.abp	スケッチの例 > 06.Sensors > Ping	超音波距離センサHC-SR04で距離を測ります。
Robotics experiment	RDC_ArdublockSamples > 04_RDC_IRdistance.abp	スケッチブック > RDC_samples > 04_RDC_IRdistance	ボードの赤外線LEDと明るさセンサを使って対象物までの距離を測ります。
Robotics experiment	RDC_ArdublockSamples > 05_RDC_motor.abp	スケッチブック > RDC_samples > 05_RDC_motor	STEM Du出力系のブロックでモーターを動かします。
Robotics experiment	RDC_ArdublockSamples > 06a_RDC_Servo.abp	スケッチの例 > Servo > Sweep	STEM Du出力系の標準サーボブロックでサーボを動かします。
Robotics experiment	RDC_ArdublockSamples > 06b_RDC_Servo_Slider.abp	スケッチの例 > Servo > Knob	スライダーの位置に応じてサーボが回転する。
Robotics experiment	RDC_ArdublockSamples > 07_RDC_IC2LCD_Sensors.abp	スケッチブック > RDC_samples > 07a_RDC_IC2LCD_Sensors	RDC-104のI2C接続のLCDにアナログセンサの値を表示します。
Robotics experiment		スケッチブック > RDC_samples > 07b_RDC_I2CLCD_Tester	テスター、電池チェッカー+抵抗チェッカー。
Robotics experiment	RDC_ArdublockSamples > 08_RDC_QMC5883.abp	スケッチブック > RDC_samples > IC2センサQMC5883	STEM Duアクセサリのコンパスセンサブロックを使います。QMC5883コンパスセンサの値をシリアルモニターに出力します。
Robotics experiment	RDC_ArdublockSamples > 09_RDC_MPU6050.abp	スケッチブック > RDC_samples > 09a_RDC_MPU6050	STEM Du入力系のブロックを使います。加速度/ジャイロセンサMPU6050からシリアルモニターに出力します。
Robotics experiment	RDC_ArdublockSamples > 10_RDC_tone.abp	スケッチの例 > 02.Digital > toneMelody	トーンブロックを使ってブザーの音を出します。
Robotics experiment		スケッチブック > RDC_samples > RDC_Serial	シリアル通信でWifi基板など他のデバイスとシリアル通信する。
Robotics experiment		スケッチブック > RDC_samples > RDC_MPU6050_2	ライブラリーを使わないスケッチ
Robotics experiment	RDC_ArdublockSamples > RDC_SPIGLCD_Sensors	スケッチブック > RDC_samples > RDC_SPIGLCD_Sensors	RDC-103用 アナログセンサモニタ
Robotics experiment	RDC_ArdublockSamples > RDC_SPIGLCD_Tester	スケッチブック > RDC_samples > RDC_SPIGLCD_Tester	RDC-103用 テスター、電池チェッカー+抵抗チェッカー
RDC-TEC31	RDC_ArdublockSamples > 11_clash_avoidance-HCSR04.abp		STEM Du入力/出力系のブロックを使います。超音波距離センサを使い、障害物回避します。
RDC-TEC31	RDC_ArdublockSamples > 12_Line_board-lightsensor_sample.abp		STEM Du入力/出力系のブロックを使います。明るさセンサでライントレースします。
RDC-TEC31		スケッチブック > RDC_samples > 10_RDC_WiFi_Serial	シリアルモニターをWiFiでデータ送信します。
RDC-TEC31		スケッチブック > RDC_samples > RDC_linetrace_1 sensor_LCD	LCDにデータ表示、スライダーでしきい値
RDC-TEC31		スケッチブック > RDC_samples > RDC_linetrace_1 sensor	明るさセンサ1個でライントレース
RDC-TEC31		スケッチブック > RDC_samples > RDC_linetrace_2sensor	反射赤外線センサ2個でライントレースをする

4-5. Arduino-IDE

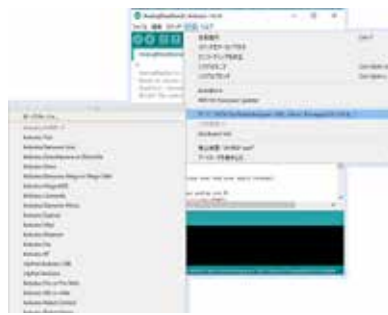
4.5.1. Arduino-IDE 画面の構成



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] > [ボードマネージャ] メニューから接続したい Arduino ボードの名前を選びます。(RDC-104 は、STEM Du/RoboDesigner+ RDC-104 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。

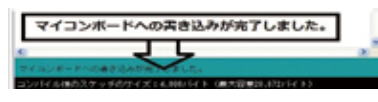


アップロードボタン →
upload button

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



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。



pin 13 LED

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。

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-104, STEM Du/RoboDesigner+ RDC-104 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.

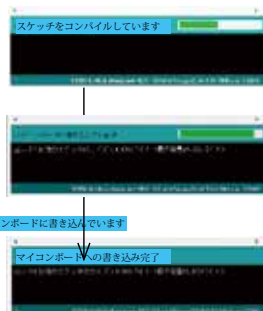
pin13 LED on the board will begin to blink several seconds later when upload has ended.

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中に、RDC-103 コントローラのRESET スイッチを「ダブルクリック」し強制アップロードしてください。



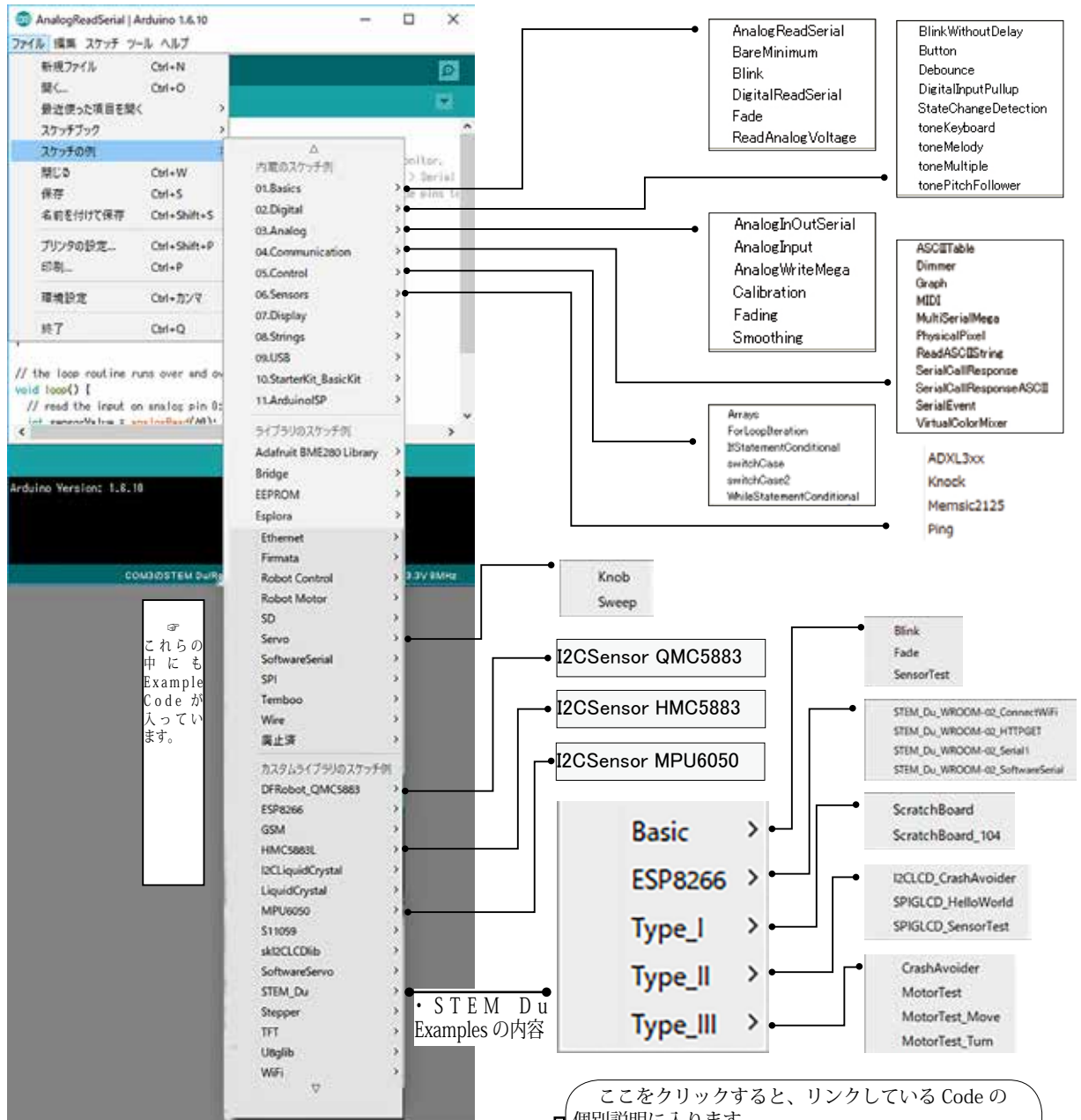
このタイミングです。



* 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.

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角丸枠囲み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.

```

ReadAnalogVoltage | Arduino 1.6.10
ファイル 編集 スケッチ ツール ヘルプ

ReadAnalogVoltage
/*
  ReadAnalogVoltage
  Reads an analog input on pin 0, converts it to voltage, and prints the result to the serial monitor.
  Graphical representation is available using serial plotter (Tools > Serial Plotter).
  Attach the center pin of a potentiometer to pin A0, and the outside pins to ground and +5V.

  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);

  // 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);
}

```

//(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.

//((voltage) を (sensorValue) に変更します。
//voltage is changed to sensorValue.

COM3のSTEM Du/RoboDesigner+ RDC-104 w/ ATmega32U4 3.3V 8MHz

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: A4
コントローラ搭載の光センサ: A2
コントローラ搭載のスライダボリューム値: A3
接続した任意のセンサ: A0, A1 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 の換算値で表現されます)

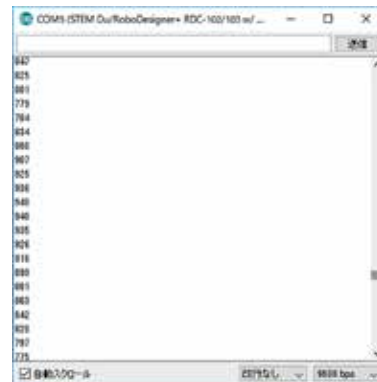


4. [ツール] で [ボードマネージャー]、[シリアルポート] を確認し、通信可能に設定のうえ、作成したスケッチを、マイコンボードに書き込みます。

*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「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。COM ウィンドウが表示され、計測データが表示されます。



7. [ツール] > [シリアルプロッタ] をクリックすると、測定中のデータが描画されます。リアルタイムにプロットされますので、イメージを把握するのに利用します。

8. データを全体傾向として把握し、しきい値などの検討に利用するのは、データ全体を収集し、グラフ化などをして分析します。

※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。

9. 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…→ 必要なデータ範囲を選択 [Ctrl]+[A] 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、電気スタンドに近づけたり、離したりした明るさセンサで測定したデータを、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。
- ***Arduino [ツール] > [プロッター] を利用して、測定中にリアルタイムにグラフ化することもできます。

10. 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.

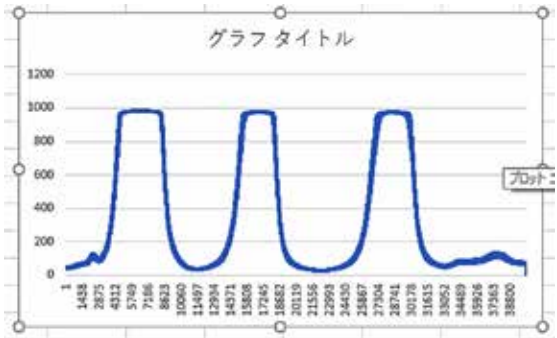
8. You'll collect the whole data, graph it and analyze.

9. 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。



1019
1019
1019
1019
1019
1019
1019
602
582
549
546
544
541
538
534
545
547
551
548
467
465
467
468
473
461
461

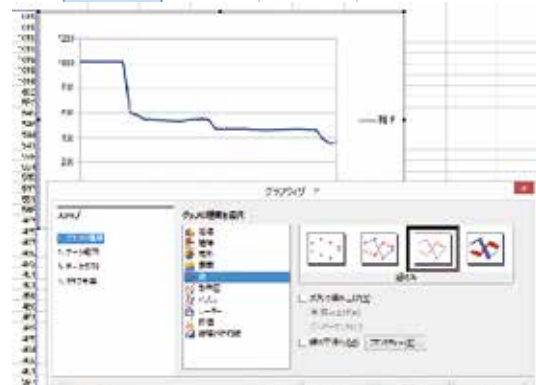
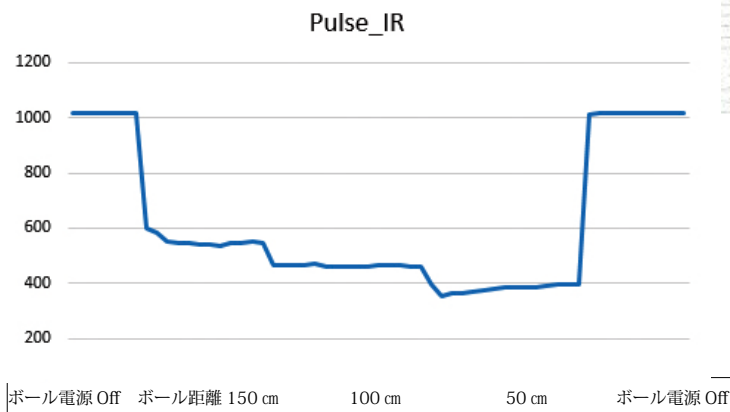
1. シリアルモニターの計測データを範囲指定 (データ全部を指定したい時は [Ctrl+A]) して、コピーします [Ctrl+C]。
2. 貼り付けたい表計算ソフトの位置 (先頭のセル) を指定して、ペーストします。 [Ctrl+V]
3. グラフ化したいデータを範囲指定して、グラフ化処理をします。
4. グラフの形などは自分が分かりやすい形を選びます。

無題 1 - OpenOffice

データベース時に、データ種別を決めておきます。区切りのオプション：固定幅 (F)

「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され、セルに配置されます。

※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。



☞ 図は、ロボカップジュニア公式競技ボールを変調赤外線センサで計測したデータです。

- ・グラフで見ると、距離ごとに計測数値が違うことも理解でき、プログラムの調整に役立ちそうです。
- ・ボールから離れているとき、中間距離時、近い距離など、ボールからの距離ごとにデータが違うことが分かり、プログラムで工夫をしてロボットの行動を変えることが可能です。

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 はマイコンボードが中心となって構成されています。つまり、マイコン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、マイコン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.

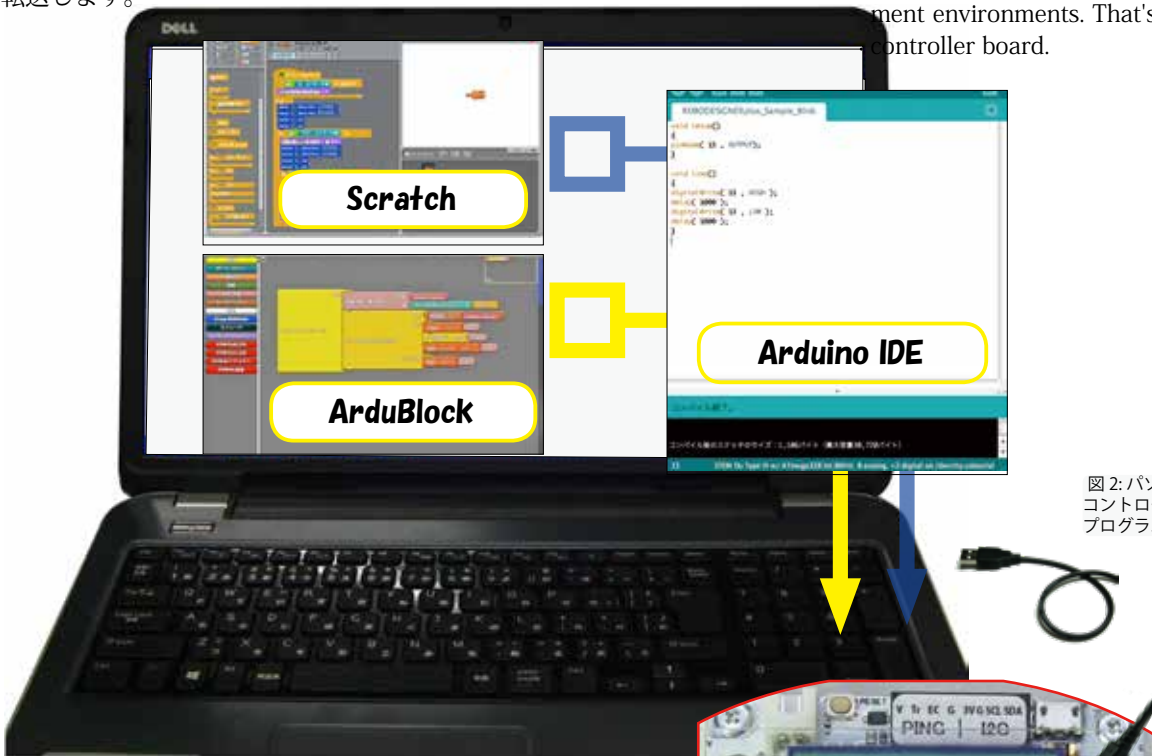


図 2: パソコンから、コントローラボードにプログラムを転送する

図 1: Arduino を使ってプログラムを作成

パソコンとマイコンボードの接続には、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ます、プログラムの動きはシミュレータで確認できます。

◆ ArduBlock は、パソコンとマイコン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、マイコンボードは独立で、プログラムにしたがって指令を出すことになります。実際は、転送の際に、パソコンとマイコンボードとの間に USB ケーブルが中継することになりますし、マイコンボードには、センサからの信号も入ることになりますので、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？

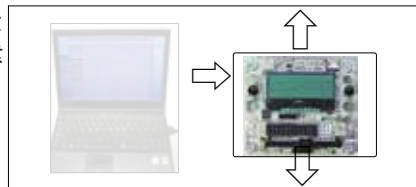


図 3: マイコンボードから各パーツに指令が出る

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

1. 前項の説明は、概略的なもので、良くわからないでしょう。では、実際に、ロボットの構成を考えてみた方がわかりやすいかもしれません。
2. 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しょう。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)。
3. 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すが、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な感じです。

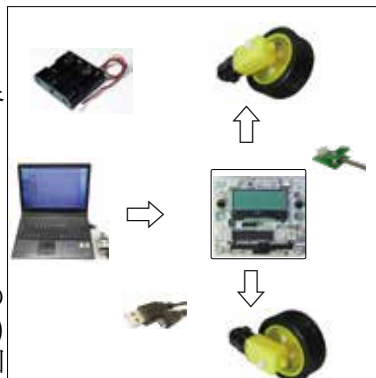


図4:構成図

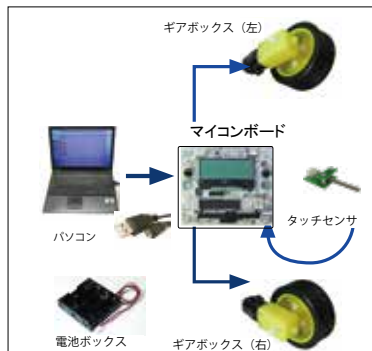


図5:マイコンボードにプログラムを転送

4. 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マイコンボードに転送します。
5. 転送が終了したら、USB ケーブルは、マイコンボードから外してしましましょう。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。

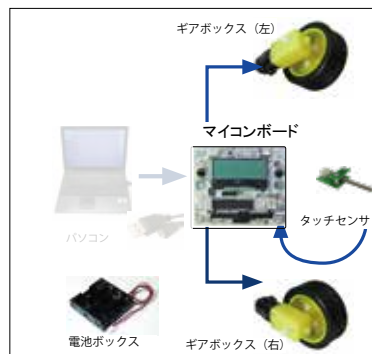
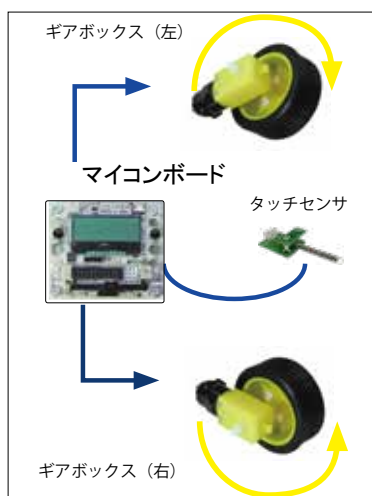


図6:USB ケーブルを切り離す

6. これは、パソコンとマイコン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、マイコンボードに転送されたプログラムを実行します。
7. プログラムが実行されると、まず、マイコンボード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。
8. 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、信号がマイコンボードに送られます。
9. 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、マイコンボード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。



10. 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。

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 actually, you take off).

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 forwarded 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 forwarded 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 ahead with this, doesn't it?

8. That a robot is keeping moving ahead 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 wheel 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. 実際にとってみよう (実験)

では、実際にとってみます。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っている、「ロボットをつくろう」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)

マイコンボード (RDC-104)	× 1
ギアードモータ (RDO-502)	× 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-104 に取り付けられているものです)	

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るのです。

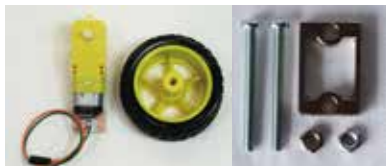
ただし、最後の手段なので、覚悟を決めてやってください。それから、人と季節によるのですが、

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

使用するモータの部品袋には下記の部品が入っています。実験には、モータ部を使って行いますので、その部品だけを準備します。残りは保管しておきます。



モータについている部品一式

- 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

マイコンボードに付いている部品



- マイコンボードに取付いている部品
- 樹脂スペーサM3x10mm 4個
- M3x 4 mmネジ 4本
- 同梱している部品
- M3x8mmネジ 4本

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. マイコンボード確認

マイコンボードRDC-104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、
 そのままロボットの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(赤い矢印の部分)。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。
 写真では8mmネジで裏から止めています。



図2：使用工具 ⊕ドライバー ○ナットドライバー

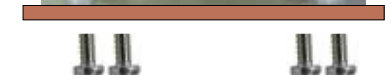
では、取り付けていきます。
 写真の位置にある4箇所の穴に、最初に樹脂スペーサを取り付けます。



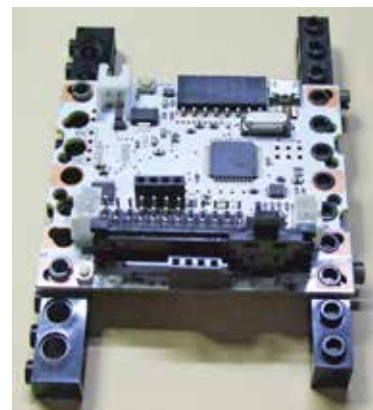
ネジの取り付けは、⊕ドライバーで行います。スペーサがすべる場合は、ナ
 ットドライバー、又は、ラジオペンチで挟むと便利です。
 4つの穴全部に取り付けたら終了です。



スペーサ取り付け位置



ロボットプレートなどへのスペーサ固定例(カットモデル)

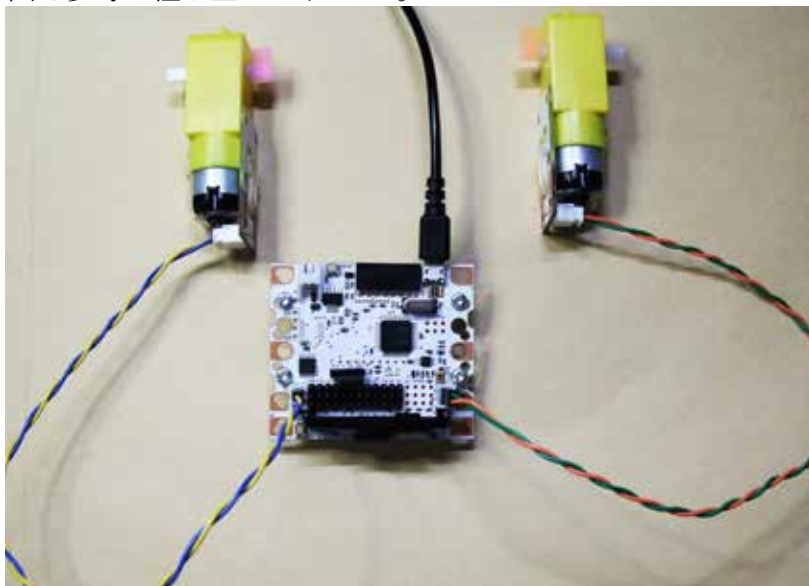


ポッチ付アームへの取り付け例

6.1.2.2. 実験機組み立て

ネジ/スペーサ 取り付け

モータ回転実験時に実験機が移動しないように、タイヤを付けずにギアード
 モータのみでの実験機を作ります。
 下の図を参考に組み立ててください。



モータ回転を確認出来るように、糊付き付箋製品ポストイットなどを小片
 に切ってモータ出力軸に旗として付けておき、回転を見やすくします。

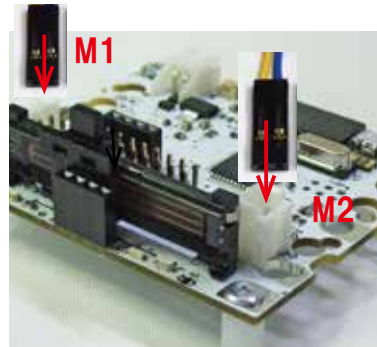
6.1.2.4. 実験装置の配線

配置が完了したら、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。

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	
赤黒線	電池ボックス ⇄ V1コネクタ	電池ボックスの赤黒コード
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

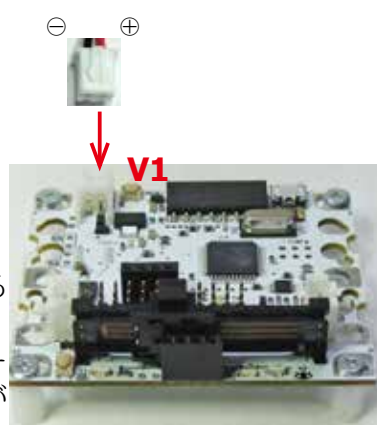
モータコードの取り付け

表の上から順番に配線を行きましょう。まずは、左モータとマイコンボードのM1、右モータとマイコンボードの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。今回は限れば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、マイコンボードの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。ソケット部根本の基板裏面に印刷しています。電池ボックスの配線は、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ようにはできていますが注意が必要なお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か?と、コネクタ部分を分解し、逆接続の実験などなさないようにしてください。



マイクロUSB ケーブル接続

パソコンとマイコンボードをマイクロ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、Win10以前のPCは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。
* マイコンボードの回路はUSBケーブルからの電力で動作します。モータを回転させる電力はUSBケーブルでは不足しますので、電池を接続して使います。



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 のマイコンボードは、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、単純計算を超高速度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いますが、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、○○○の部分は、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。どんなに高速で動いても、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 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

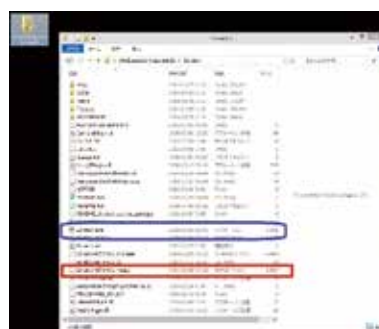
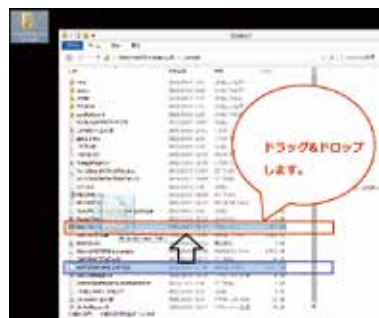
6.2.1. Scratch を起動する。


1. windows の場合：

[WinScratch1.4-stemdu01] --> [Scratch] 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] 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。)

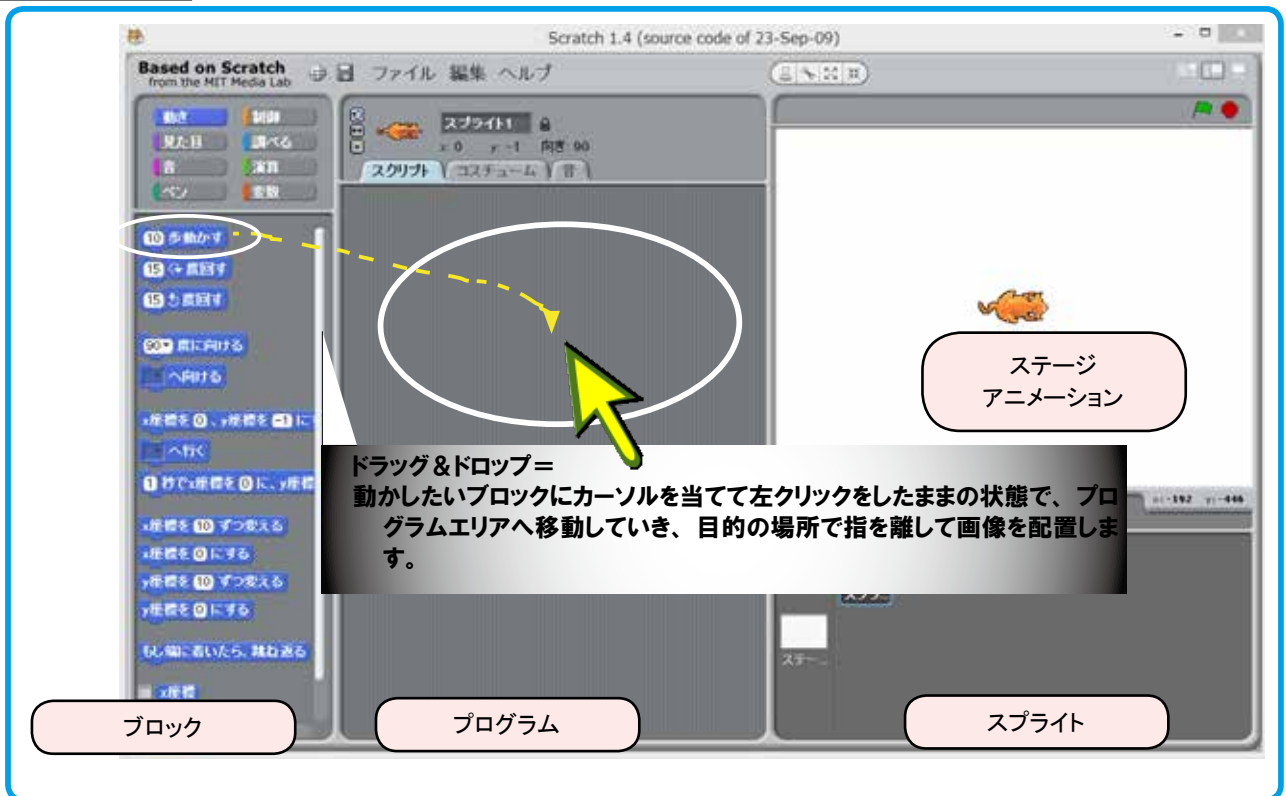
3. First, a use language is established. Please confirm the screen structure of Scratch of the next page and establish it by the menu of the linguistic setting of a menu bar in the upper part.
 When an icon is clicked, a sub-window of [(Set language) which establishes a language] shows. (In case of Japanese, 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。

The screenshot shows the Scratch 1.4 interface with various elements highlighted by callouts:

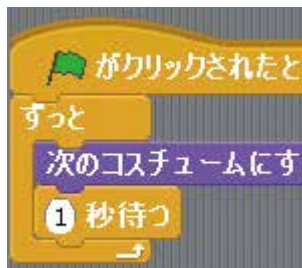
- 言語を設定 (Set Language):** A list of 50 languages including Bahasa Indonesia, Català, Creole, Czech, Dansk, Deutsch, Festi, English, Español, Euskera, Français, Français (Canada), Gallego, Hrvatski, Italiano, Kinyarwanda, Lietuvių, Magyar, Nederlands, Norsk, Polski, Português, Português (Brasil), Română, Slovak, Slovenščina, Tiếng Việt, Türkçe, suomi, svenska, Íslenska, Ελληνικά, Български, Македонски, Монгол хэл, Русский, Українська, العربية, فارسی, नेपाली, मराठी, తెలుగు, and にほんご (Japanese).
- 共有 (Share):** Buttons for sharing projects.
- 保存 (Save):** Buttons for saving projects.
- 共有 (Share):** Another set of sharing buttons.
- ファイル (File):** A menu with options like 新規開く... (New), 保存する (Save), 名前をつけて保存... (Save with name...), プロジェクトを読み込む... (Load project...), スプライトを書き出す... (Export sprite...), プロジェクトのメモ... (Project notes...), and 終了 (Quit).
- 編集 (Edit):** A menu with options like 削除の取り消し (Undo), ステップ実行を開始 (Start stepping), ステップ実行を設定... (Set stepping), 音の圧縮... (Compress audio), 画像の圧縮... (Compress image), and モーターのブロックを表示する (Show motor blocks).
- モータブロック表示 (Show Motor Blocks):** A sub-menu with options like モーターを 秒オンにする (Turn motor on for seconds), モーターをオンにする (Turn motor on), モーターをオフにする (Turn motor off), モーターのパワーを 100 に設定 (Set motor power to 100), and モーターの回転を 反転させる (Reverse motor rotation).
- アニメーション (Animation):** A menu with options like スプライトを縮小 (Shrink sprite) and スプライトを大きくする (Enlarge sprite).
- プログラム (Program):** A menu with options like 削除 (Delete), 複製 (Duplicate), and ヘルプ (Help).
- 音 (Sound):** A menu with options like ニューの音を鳴らす (Play new sound), 終わるまで ニューの音を鳴らす (Play new sound until done), すべての音を止める (Stop all sounds), 40 のドラムを 0.2 拍鳴らす (Play 40 drum sounds for 0.2 beats), 0.2 拍体心 (Play 0.2 beat drum), 60 の音符を 0.5 拍鳴らす (Play 60 notes for 0.5 beats), 楽器を 1 にする (Set instrument to 1), 音量を -10 ずつ変える (Change volume by -10), 音量を 100 にする (Set volume to 100), and テンポを 20 ずつ変える (Change tempo by 20).
- 制御 (Control):** A menu with options like がクリックされたとき (When clicked), スプライト1 がクリックされたとき (When sprite 1 clicked), and 10 回繰り返す (Repeat 10 times).
- 条件 (Conditions):** A menu with options like かつ (And), または (Or), and ではない (Not).
- センサー値 (Sensor Values):** A menu with options like タイマー (Timer), 音量 (Volume), うるさい (Loudness), スライダー センサーの値 (Slider sensor value), スプライト1 音量 (Sprite 1 volume), スライダー センサーの値 (Slider sensor value), 音センサーの値 (Sound sensor value), 明るさ センサーの値 (Light sensor value), and 動き センサーの値 (Motion sensor value).
- 動き (Motion):** A menu with options like 移動 (Move), 回転 (Turn), and スケッチ (Sketch).
- 見た目 (Appearance):** A menu with options like コスチュームを コスチューム1 にする (Set costume to costume 1), 次のコスチュームにする (Next costume), 横向き (Flip horizontally), 縦向き (Flip vertically), 拡大 (Zoom in), 縮小 (Zoom out), 色 (Color), and 大きさ (Size).
- 調べる (Sensing):** A menu with options like 下に触れた (Touched down), 色に触れた (Touched color), スペース キーが押された (Space key pressed), スプライト1 の x座標 (Sprite 1 x-coordinate), 色が 色に触れた (Color touched), and スライダー センサーの値 (Slider sensor value).
- 演算 (Operators):** A menu with options like 1 から 10 までの乱数 (Random number between 1 and 10).
- センサー (Sensors):** A menu with options like 明るさ (Light), スライダー (Slider), 音量 (Volume), うるさい (Loudness), 動き (Motion), and 色 (Color).



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。マウスでポインター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」で全画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。



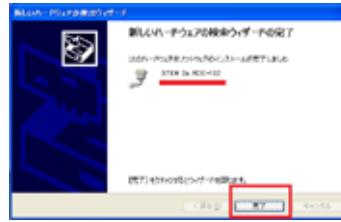
Controller board and connection experiment.

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

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。



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.

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

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

* 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.

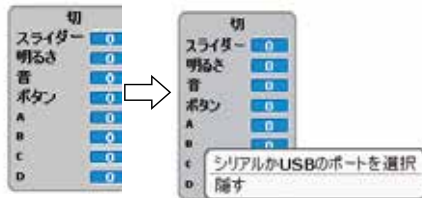
The sketch which includes motor control, Excuted, make, when, a connected motor begins to move, so please be careful so as not to drop it from the top of the desk.

(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。

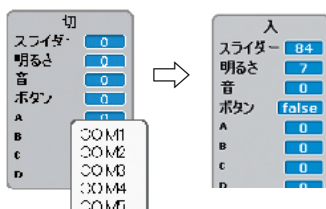
1. [Arduino-IDE]>[スクラッチの例]>[STEMDu]>[Type_1]の[ScratchBoard_104.ino]を開きます。
2. [Arduino-IDE]>[ツール]>[シリアルポート]にて調べた COM 番号に設定します。
3. Arduino-IDE のアップロードボタン(→)をクリックし、マイコンボード (RDC) に書き込みます。
4. 書き込みに成功するとメッセージが表示されます。



(3). 【Scratch での準備】 通信設定を行います。

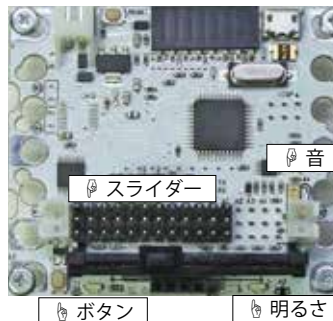


1. センサ - ブロックを選択し、右クリックすると、メニューが現れます。「ScratchBoard 監視板を表示」を選択し、クリック実行します。
2. ステージに「ScratchBoard 監視板」が出現します。



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。

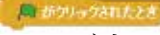
・通信設定完了すると、「切」→「入」へ変化し、ボード搭載センサのデータ値が表示されます。



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。
 - ・ 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「スライダー」に選択変更します。
2. このスクリプトではスタートが  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 のスライダー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 は、接続状態のまま、マイコンボードが動作。

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。

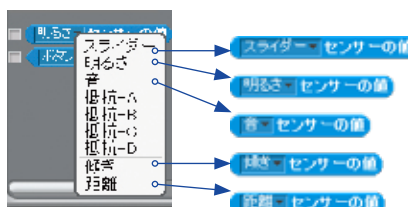


6.2.7. Scratch 使用時のロボットで多く使うスクリプト

(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。
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".

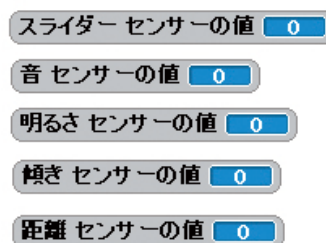
A condition of the connected position and microcomputer board RDC move in Scratch.

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

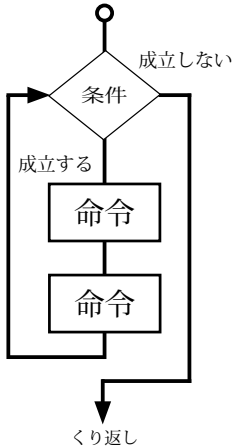
- (1). Motor
 1. [Edit] clicks> [A block of a motor is indicated.] in a block of a motor.
 2. It's added to the "moving" list of block palettes.
 3. A motor can use 2 kinds, A and B.
 4. Motor terminal for StemDu and RDC
- (2). Sensor
 1. A sensor can choose the following 5 kinds.
 2. When on the sensor - 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.



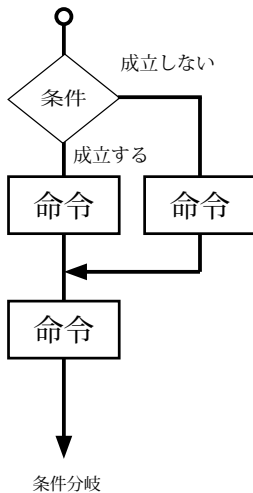
6.2.8. 例題

(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○○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。

(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。

(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。

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) 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..

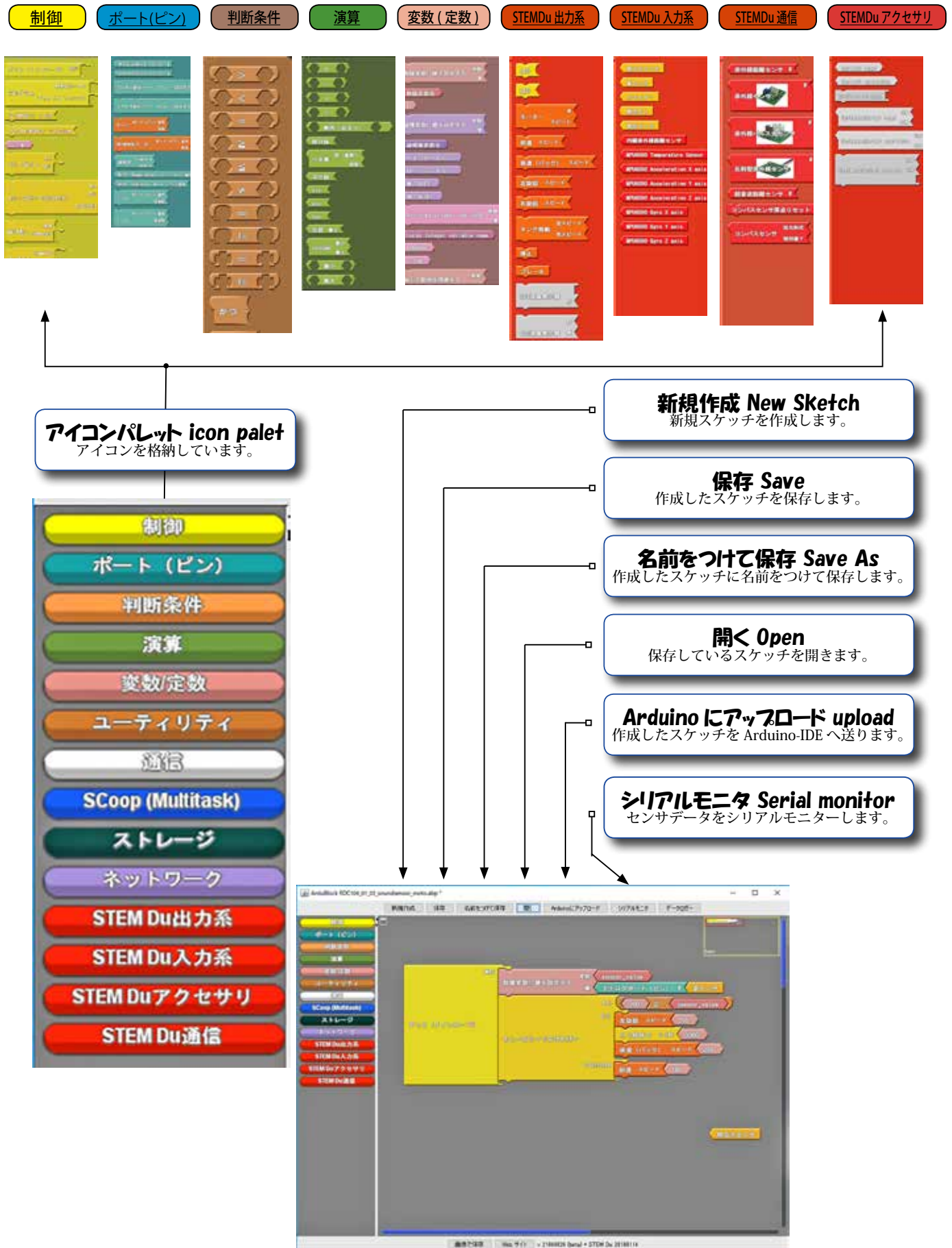
4. Please make the point of view you'd like to make bring about changes "threshold value of a program" based on a made chart.

5. Program "threshold value" as the condition value and make the program from which a movement changes.

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

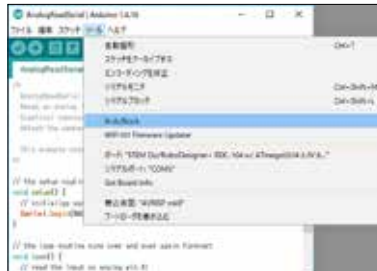
6-3. ArduBlock でプログラム実験

6.3.1. ArduBlock 画面の構成



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 と実験機のマイコンボード 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。

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

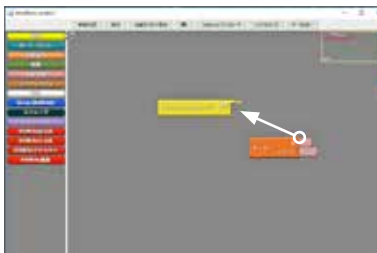
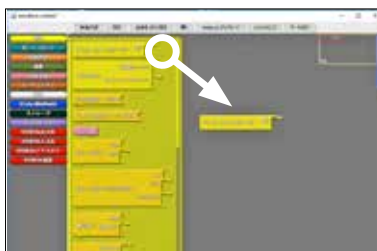
ブロックを配置してみましょう。今回、はじめに作成するのは、永遠にモータを正回転方向に低速で回転させるプログラムです。その配置は次のようになります。

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 出力系** } 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。

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。



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].

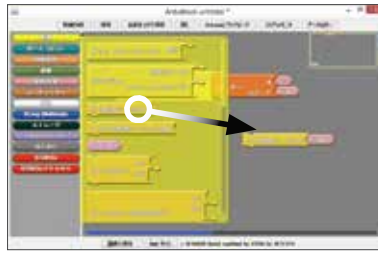
The left screen is ArduBlock which stood up.

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

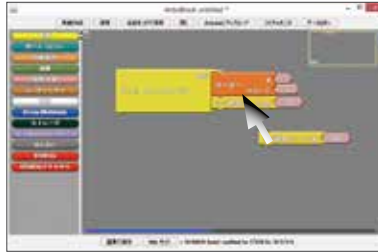
We'll arrange a block. To make it this time and first?
The program which makes a motor revolve at low speed in the positive direction of rotation direction eternally. The arrangement starts to be the next.

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.

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。

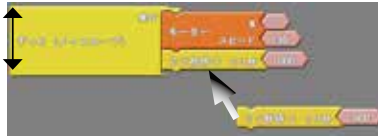


6. [ミリ秒待つ] を、3で [モータ] を結合させた [ずっと (メインループ)] ブロックのパーツ位置に追加で収まるようにドロップするとループ枠が広がり「カチッ」と音がして、追加ブロックが、ループにはまり込み結合します。



* 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、ボードマネージャーの「STEM Du/RoboDesigner+RDC104w/ATmega32U43.3V8MHz」に **●マーク**、「接続 COM 番号」に **✓マーク** が付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。



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.**

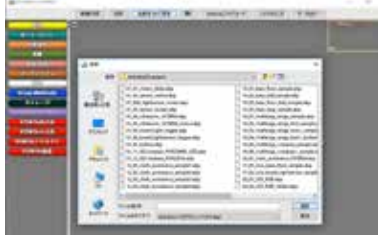
3. Making program preservation

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

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。



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. マイコン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マイコンボードの電源表示 (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、アップロード時はマイコンボードの LED を見て確認してください。

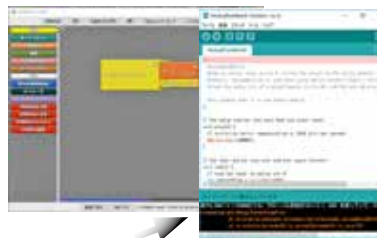
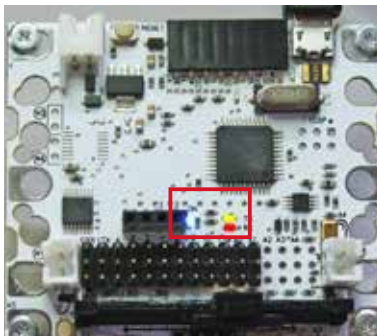
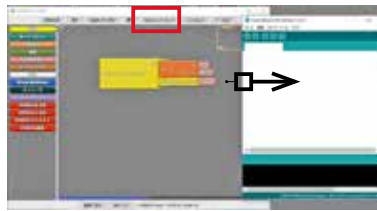
5. 成功すると「ボードへ書き込みが完了しました。」と、下のメッセージ欄に表示されます。

6. 不成功の場合はプログラム行の最初行がピンクで警告表示されます。

下のメッセージ欄にエラーメッセージを橙色表示

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] > [ボードマネージャ] > [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。



```
processing.app.debug.RunnerException at cc.arduino.packages.uploaders.SerialUploader.uploadUsingPreferences(SerialUploader.java:162) at cc.arduino.UploaderUtils.upload(UploaderUtils.java:78) at processing.app.Sketch.upload(Sketch.java:1187) at processing.app.Sketch.exportApplet(Sketch.java:1160) at processing.app.Sketch.exportApplet(Sketch.java:1132) at processing.app.Editor.$DefaultExportHandler.run(Editor.java:2409) at java.lang.Thread.run(Thread.java:745)
Caused by: processing.app.SerialException: シリアルポート「COM5」をタッチできませんでした。
at processing.app.Serial.touchForCDCReset(Serial.java:87)
at cc.arduino.packages.uploaders.SerialUploader.uploadUsingPreferences(SerialUploader.java:146)
... 6 more
Caused by: jssc.SerialPortException: Port name - COM5; Method name - openPort(); Exception type - Port not found.
at jssc.SerialPort.openPort(SerialPort.java:167)
```

- Arduino メニューバーの [ツール] > [マイコンマネージャ] メニューから接続したい Arduino ボードの名前を選びます。(RDC-104 は、STEM Du/RoboDesigner+ RDC-104 w/ ATmega32U4 - 3.3V 8MHz)
- 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 COM5 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。
- 「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。

* Windows10 以前の PC 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います。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。

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。

7. 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。

Program upload

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

- A microcomputer board isn't connected to a USB port.
 - communication port COM (No) The setting is wrong.
 - working environment of a board isn't right.
 - then a communication error occurs, and writing in to a microcomputer board is failed.
 - 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)
 - 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.
 - 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.

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.

8. 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。

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 出力系** } 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。

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



アップロードしたコード

```

// 「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
    
```

◆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。

「後進 (モータ逆回転) スピード 255」へ変更し、アップロード

```

// 「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);
}
    
```

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実行するのではなく、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。



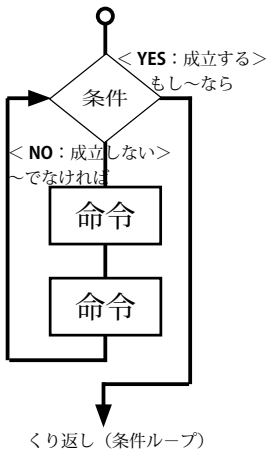
ロボットのプログラミングを行う際に、多く用いられる書式として、**繰り返し(ループ)**があります。繰り返しは制御文と呼ばれる命令のひとつですが、制御文とはなんでしょうか？ある決まった動作を1回だけ行う場合は必要ありません。あるプログラムを繰り返し行う場合に必要となる構文で、自律型ロボットのプログラミングだけでなく、ほとんどのプログラミングにおいて必要不可欠なものです。



ArduBlock で使う制御文

初期化を伴うメインループ

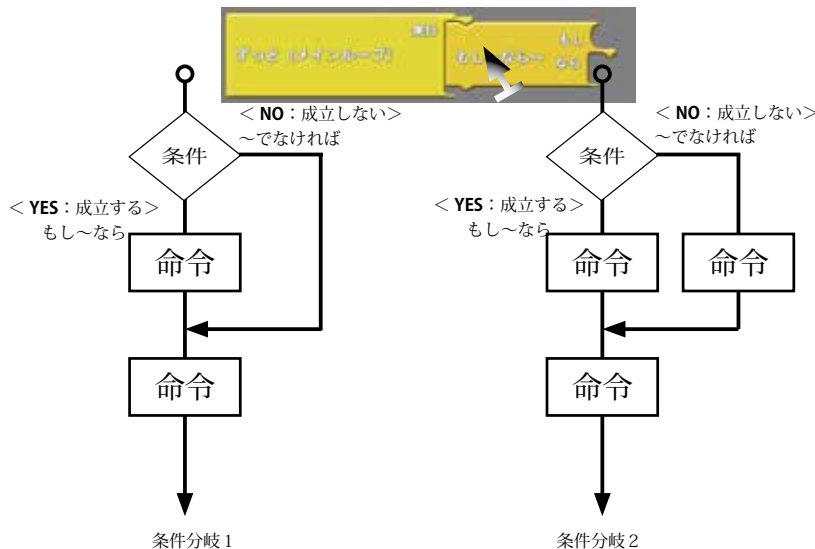
状況に応じて動きを変えて実行する(場合分け)ための命令を**条件分岐**といいます。**[もし~なら~]**で動きを変えていきます。条件が満たされない場合もありますので、**[もし~なら~、でなければ~]**も利用します。



くり返し(条件ループ)



[ずっと(メインループ)]に入れて使います。



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 programming 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。

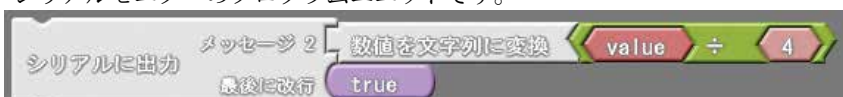


タイルを配置してみましょう。今回、作成するのは、スライダの位置によりモータ回転に変化を与えるプログラムです。

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	整数変数に値を設定する 変数 integer variable name 値 0	. 作成する変数[Value]を入れ替える 2行目. スライダと入れ替える 数値変数に値を設定する 変数 value 値 スライダ
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) value
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) 4
⑤	STEM Du 入力系	スライダ	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。 value ÷ 4
⑨	STEM Du 出力系	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。 モーター M 1 スピード value ÷ 4

・この列が、シリアルモニターのプログラムユニットです。



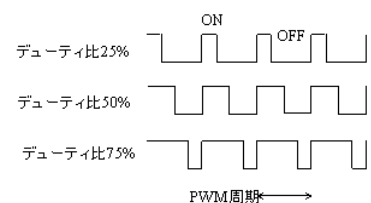
- ①. 左列アイコンパレット一番上にある { **制御** } をクリックし、出現するサブウィンドウから [ずっと (メインループ)] を選択し中央のプログラムフィールドにドラッグ & ドロップします。
- ②. 左列アイコンパレット { **変数 / 定数** } をクリックし、出現するサブウィンドウから [数値変数に値を設定する] を選択し中央のプログラムフィールドにドラッグ & ドロップします。
1 行目: Value に変更
2 行目: [スライダー] と入れ替えます。
- ③. 左列アイコンパレット { **変数 / 定数** } をクリックし、出現するサブウィンドウから [整数の変数名] 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 入力系** } 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 出力系** } 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。

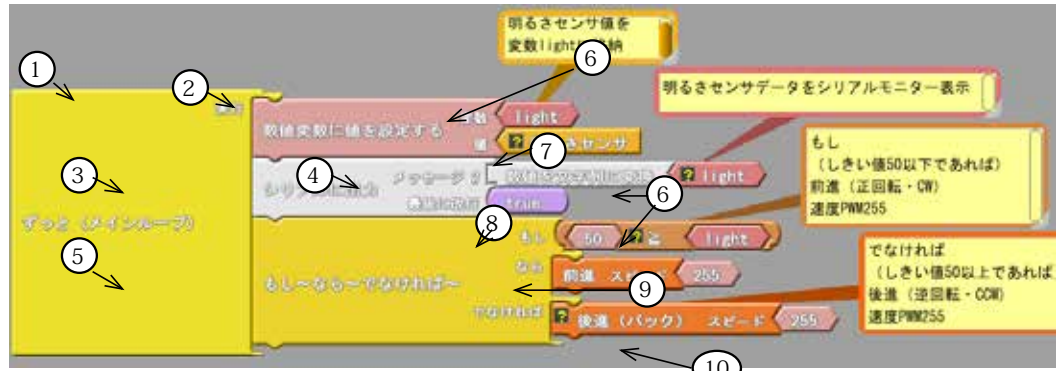
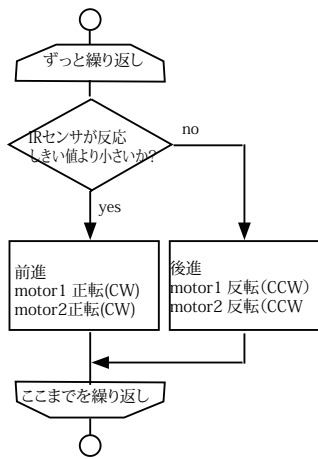
モータの駆動時において、高速にスイッチの ON-OFF を繰り返し、ON になっている時間と OFF になっている時間の比を変更することによって、見かけ上、モータにかかる電圧を変更する駆動も可能になります。

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。

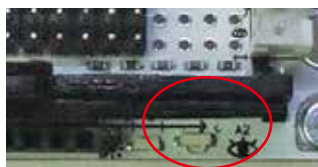


6-5. 制御文 (分岐) と入力

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	そのまま使用
②	変数 / 定数	数値変数に値を設定する	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 数値変数に値を設定する
③	通信	シリアルモニタに出力	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 入力系	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。
⑨	STEM Du 出力系	前進 スピード 255	そのまま使用
⑩	STEM Du 出力系	後進 (バック) スピード 255	そのまま使用



- ・マイコンボードの明るさ (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。
- ・アイコンボード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、マイコンボード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。

RDS-Series

ロボット動作モデルは、作成しだい追加掲載します。

RDC-104使用動作モデル

□RDS-TEC31

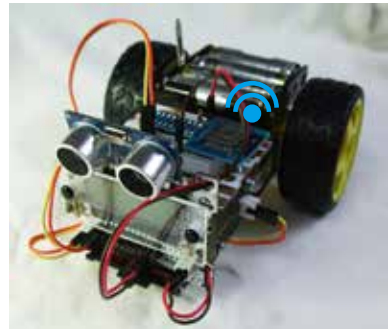
ライトレース/
超音波衝突回避ロボ



RDC-104使用動作モデル

□RDS-TEC31WiFi

WiFi-IoTロボ/ライトレース
超音波衝突回避ロボ



RDC-103使用動作モデル

□RDS-TEC34

サッカーチャレンジロボ
ドリブル・キック



RDC-103使用動作モデル

□RDS-X25

PID制御ロボ スタータセット
軌道制御オドメトリ
WiFi-IoT拡張

