

# RumiCarハンズオン！

自動運転アルゴリズムを  
楽しく手軽に体感しよう@横浜

講師：千葉 瑠実佳  
RumiCar開発部  
広報・海外展開担当



# RumiCar 開発部 紹介



千葉 瑞実佳



千葉 吉輝



片岡 秀明



小野寺 修

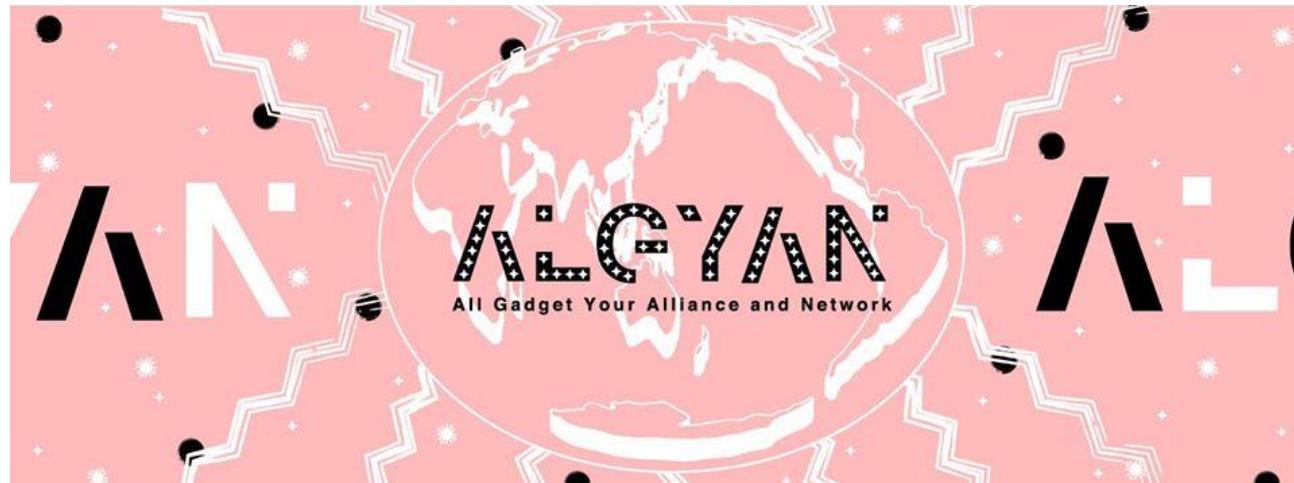


石垣 翔子



稻玉 繁樹

# 全力で協力したALGYANの運営 委員の皆さん



# 目次

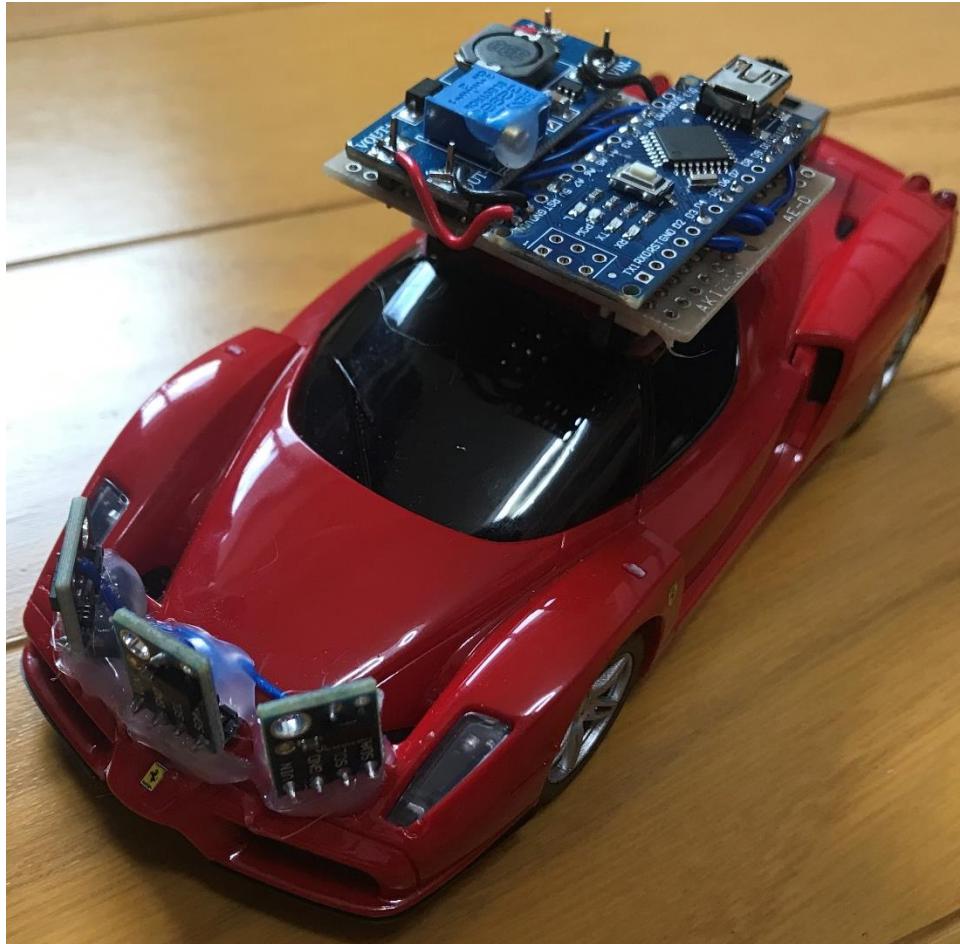
- 学べる内容
- RumiCarとは
- 準備
- 距離の測定
  - Exercise-1.1 中央のセンサで測距しよう！
  - Exercise-1.2 3つのセンサで測距だ
  - Exercise-1.3 シリアルプロッタを使ってみよう
- モーター制御
  - Exercise-2.1 ハンドルを切る
  - Exercise-2.2 速度制御
  - Exercise-2.3 前進と後進
  - Exercise-2.4 ジグザグ走行
- 自律走行基礎
  - Exercise-3.1 安全に停止する車
  - Exercise-3.2 市街地を走る車

青字のところで  
実際にプログラ  
ミングするよ

# 学べる内容

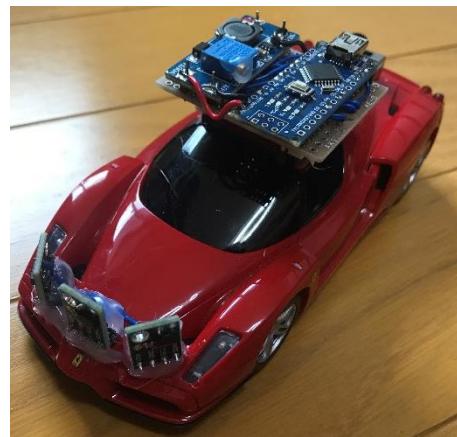
- RumiCarの仕組み
  - センサー
  - モータドライバ
- レーザー測距モジュールを使った距離測定
- モータドライバ制御
- 自律走行基礎
- その他として、良いプログラム開発のための関数化の意義、なども学べます

# RumiCarとは

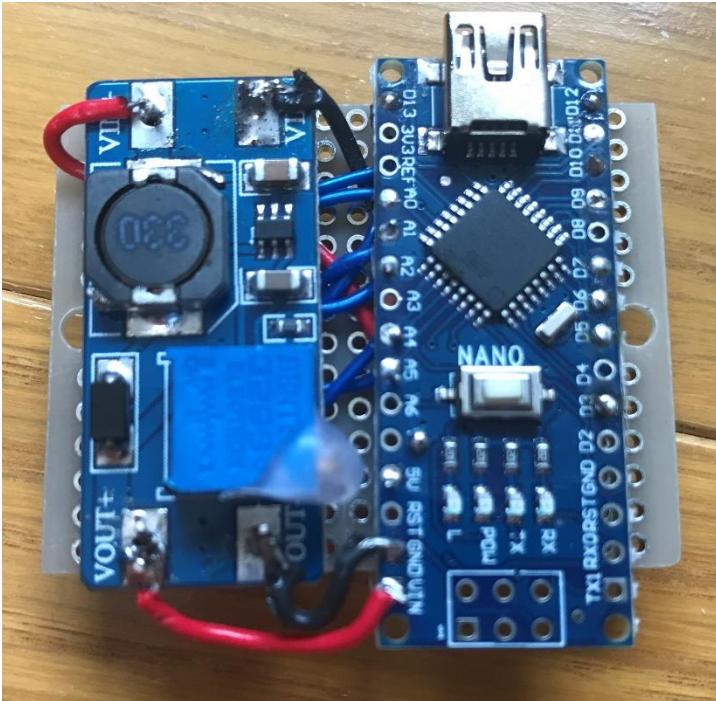


# RumiCarとは

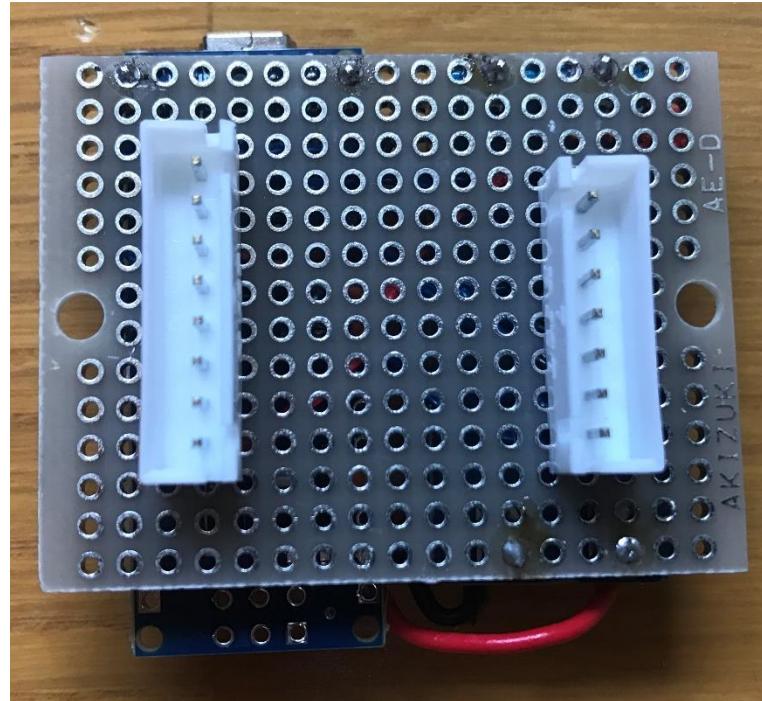
- 複数の車体と多様な種類のコンピュータモジュールを組み合わせて自律走行プログラムを開発可能なプラットフォームです
- レーザー測距モジュールを搭載していて障害物迄の距離を計測することができます
- モータの回転を制御することにより車速制御できます



# コンピューターモジュール (CM)

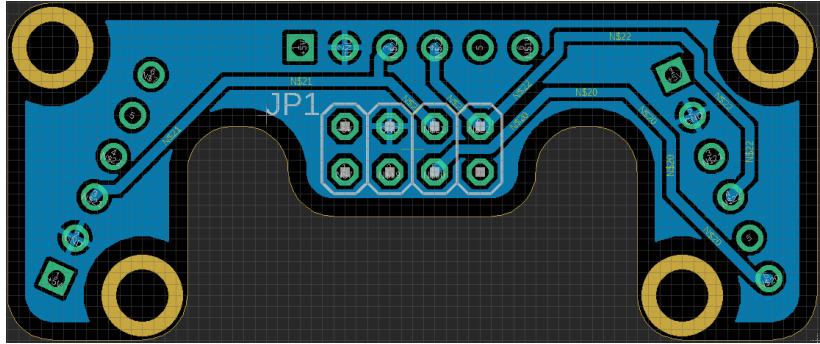


表面



裏面

# コンピューターモジュール (CM)基板化

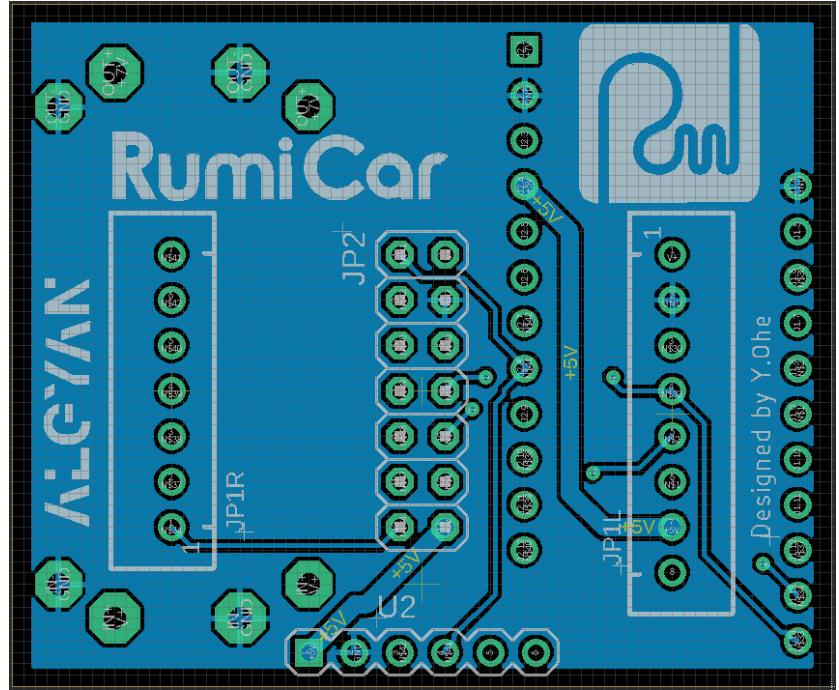


センサー基板

開発・設計してくださったのは  
大栄 豊 氏



SeeedのFusion PCBに発注。  
納品された。



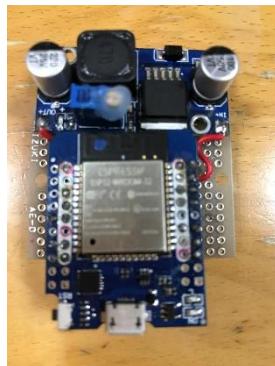
コンピューターモジュール基板

# 組み合わせ自由自在

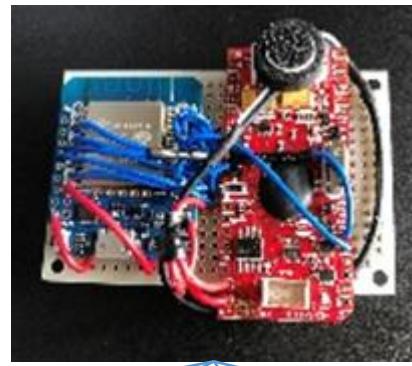
Arduino Nano



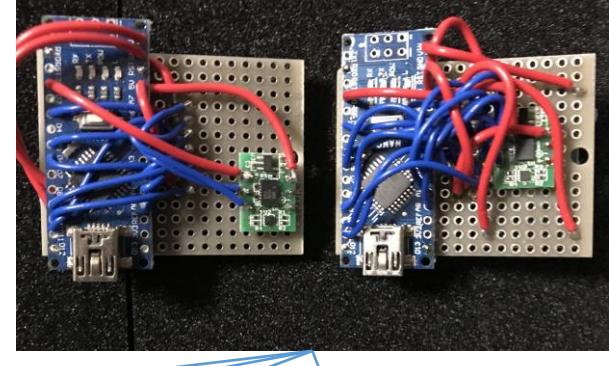
ESP32



ESP32+EasyVR3



Arduino Nano+BMX055

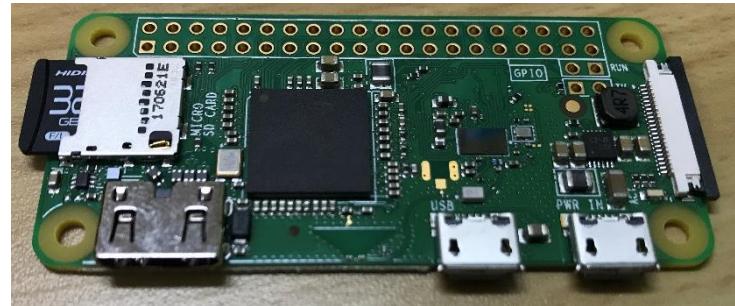


# 組み合わせ自由自在

次なる取り組み(挑戦)

Raspberry Pi

- ・コンピューターモジュール化
- ・モーター制御
- ・画像認識



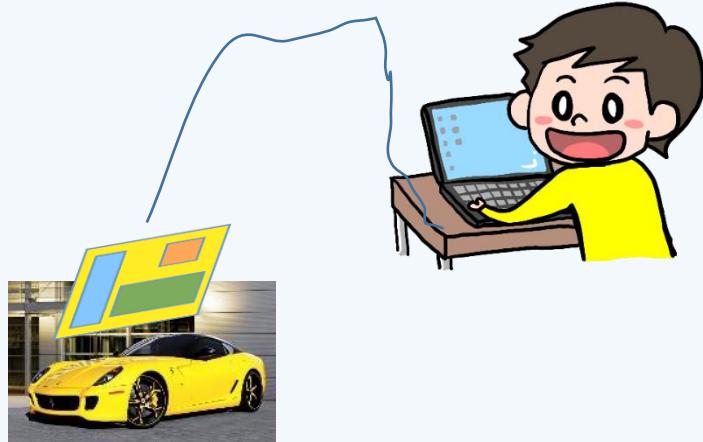
Pythonでコードを書き、  
モーター制御などしてください  
る  
石垣 翔子 氏

# コンピュータモジュール比較

ボード	Arduino Nano	ESP32	Raspberry Pi Zero W	Obniz	Spresense
電源	7-12V(VIN) (5V ラインへの供給 は非推奨)	2.2-3.6V(3V3へ入力の場合) 3.7-6.0V(VINへ入力の場合)	5V	5V	3.6-4.4V
内部動作電圧	5V	3.3V	3.3V	3.3V	
I/Oピン電圧	5V			5V(電圧選択 可能)	5V、 3.3V切替
I2C電圧	5V	3.3V	3.3V		1.8V
外部電源供給能力	ピン当たり40mA 全体で100mA	最大12mA	ピン当たり16mA 全体で50mA	5Vモードで 1A	6mA
消費電力(電流)	65mA	WiFi時160-260mA	125mA WiFi時270mA	平均170mA (WiFi不使用 時)	
PWMピン数	6本	4本	ハード2本	6本	
電源関係構成	電源:MT3608 VL/DRV: 5 V ラインよ り	PQ3RD23 (またはLM2596)	電源:MT3608 (またはTPS63020) VL/DRV:PQ3RD23 (またはSOT-23AP7333)	MT3608 (または TPS63020)	

# プログラミングで楽しめるよ

自分でプログラミング  
できるよ！



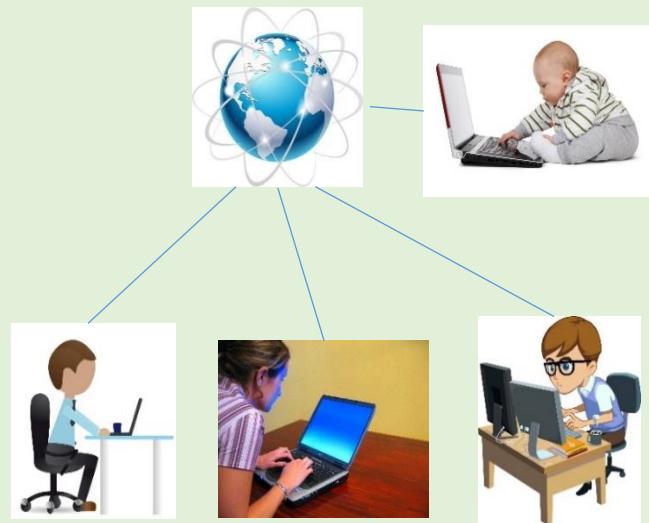
- RumiCarがあれば自分のパソコンでRumiCarへプログラムできるよ、そして自由に走らせよう
- AIに挑戦しよう
- みんながアッと驚くプログラムを作ってみよう！

# 世界の友達と交流しよう！

予定・計画中です

プログラムのダウンロードや交換ができるよ

- 自分のプログラムをサーバーにアップしよう
- おもしろそうなプログラムを見つけたら自分のRumiCarにダウンロードして走らせてみよう
- 世界の友達と交流しよう



# みんなで学ぼう！

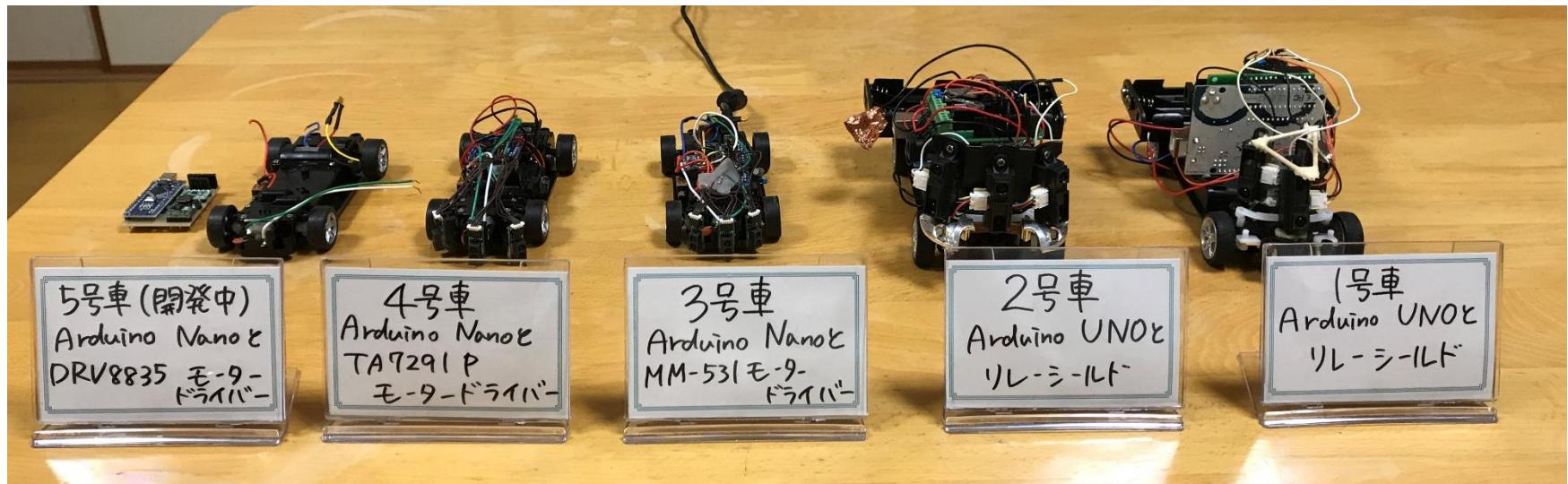
予定・計画中です

教室を開催するよ



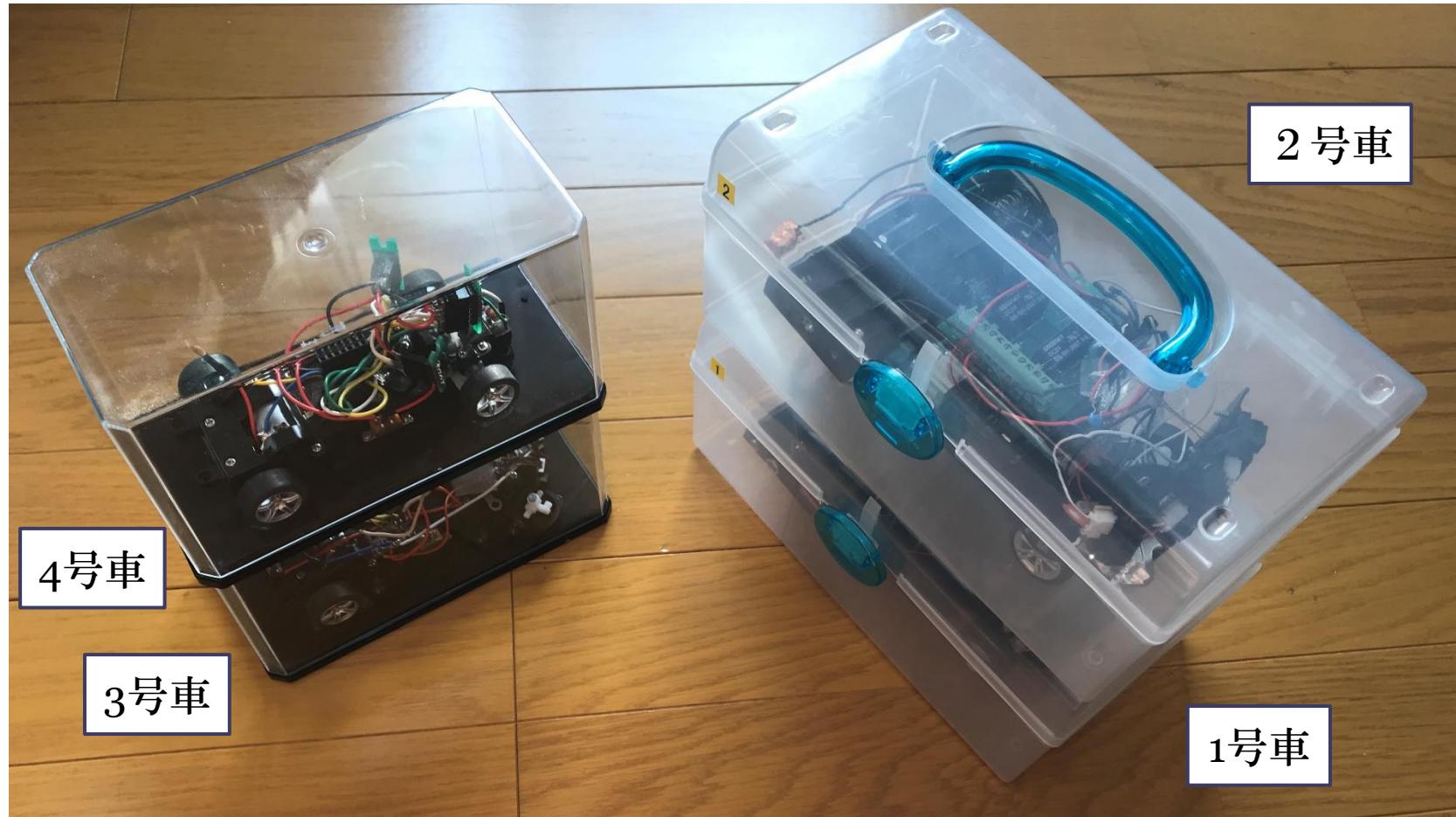
- RumiCarを作ろう！
- コンピュータモジュールを作ろう！
- プログラムを作ろう！

# 歴代RumiCar(1号車～5号車)

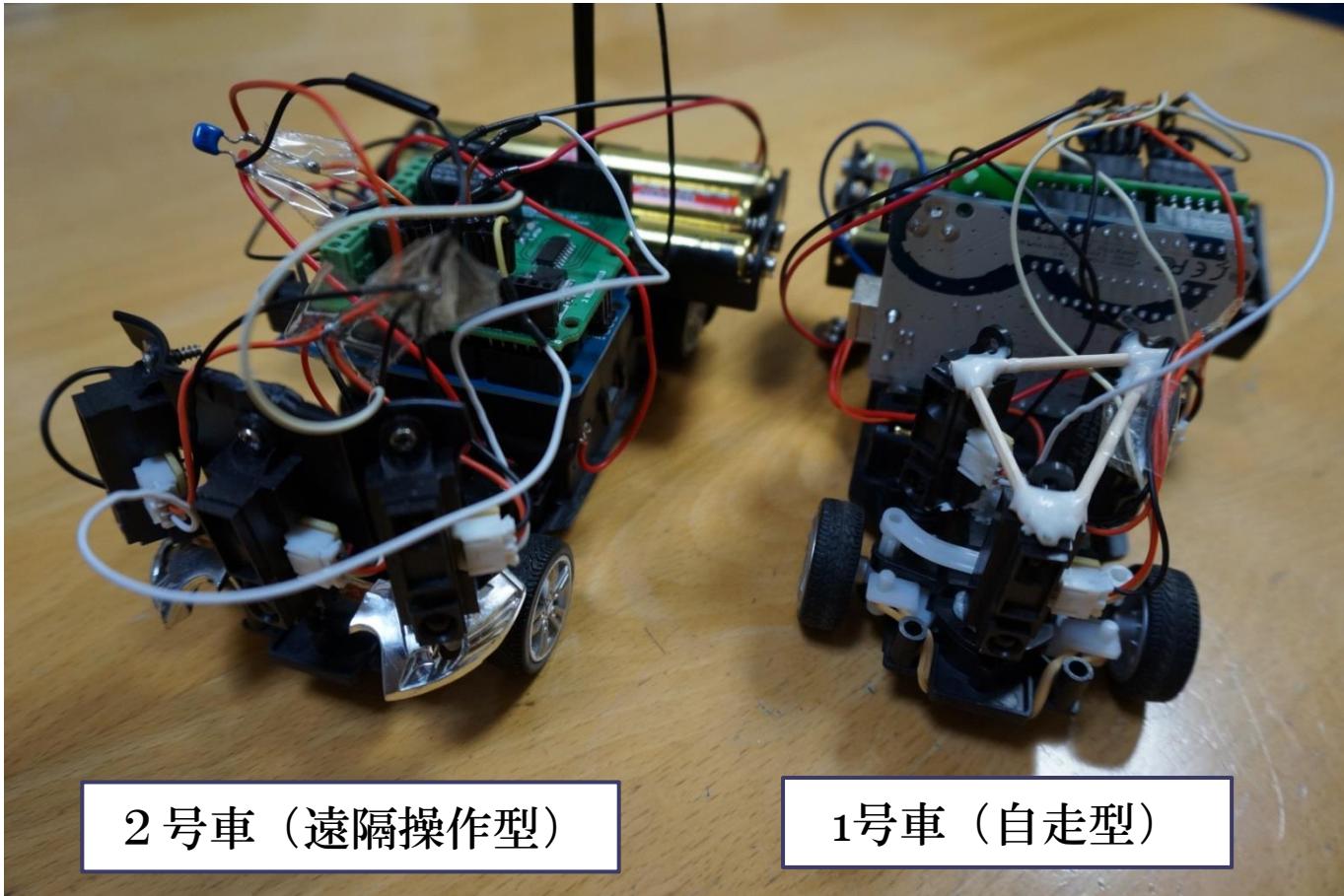


2017年8月3日撮影

# RumiCarの歴史(博物館)



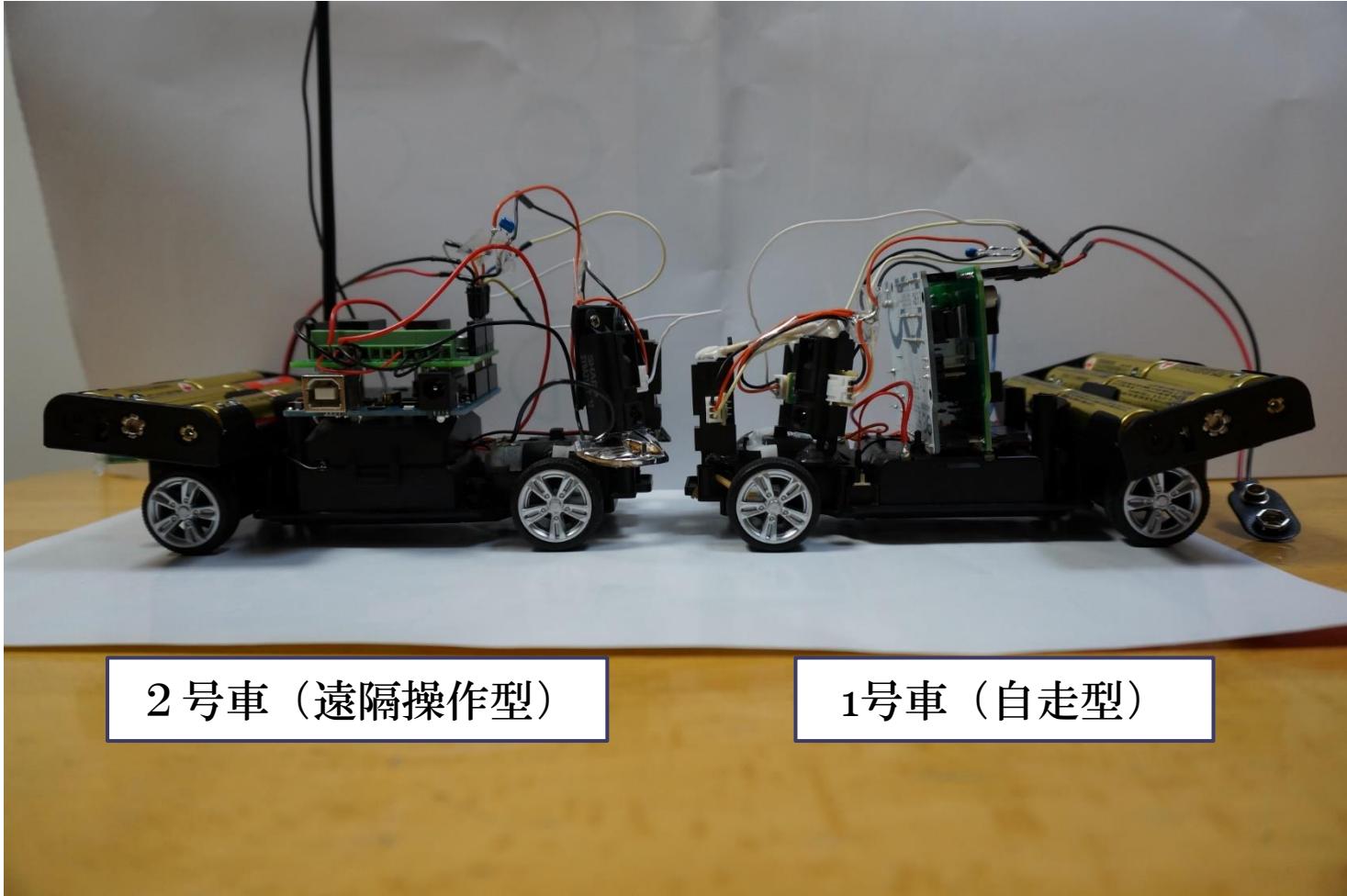
# RumiCarの歴史(1号車と2号車)



2号車（遠隔操作型）

1号車（自走型）

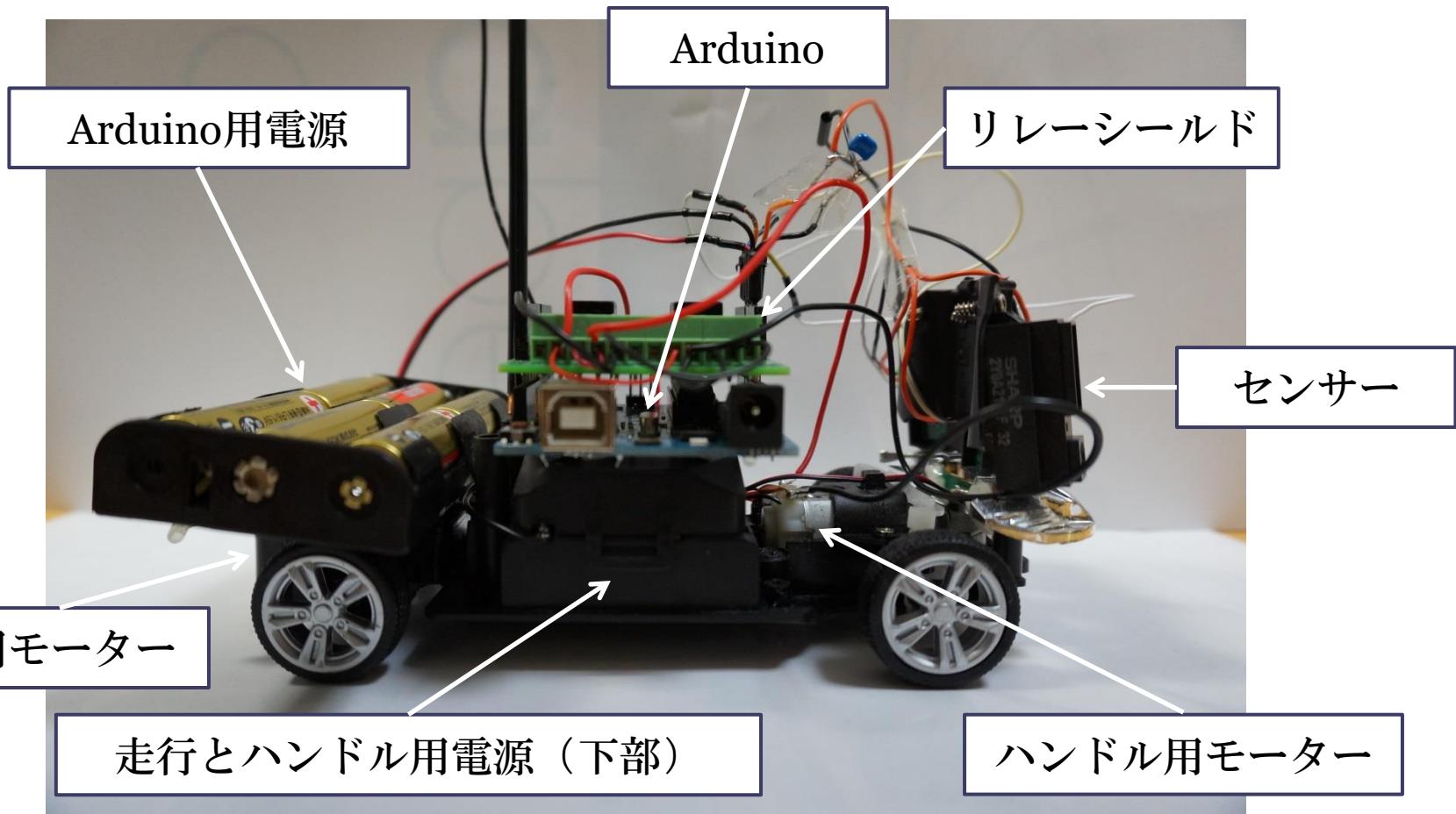
# RumiCarの歴史(1号車と2号車)



2号車（遠隔操作型）

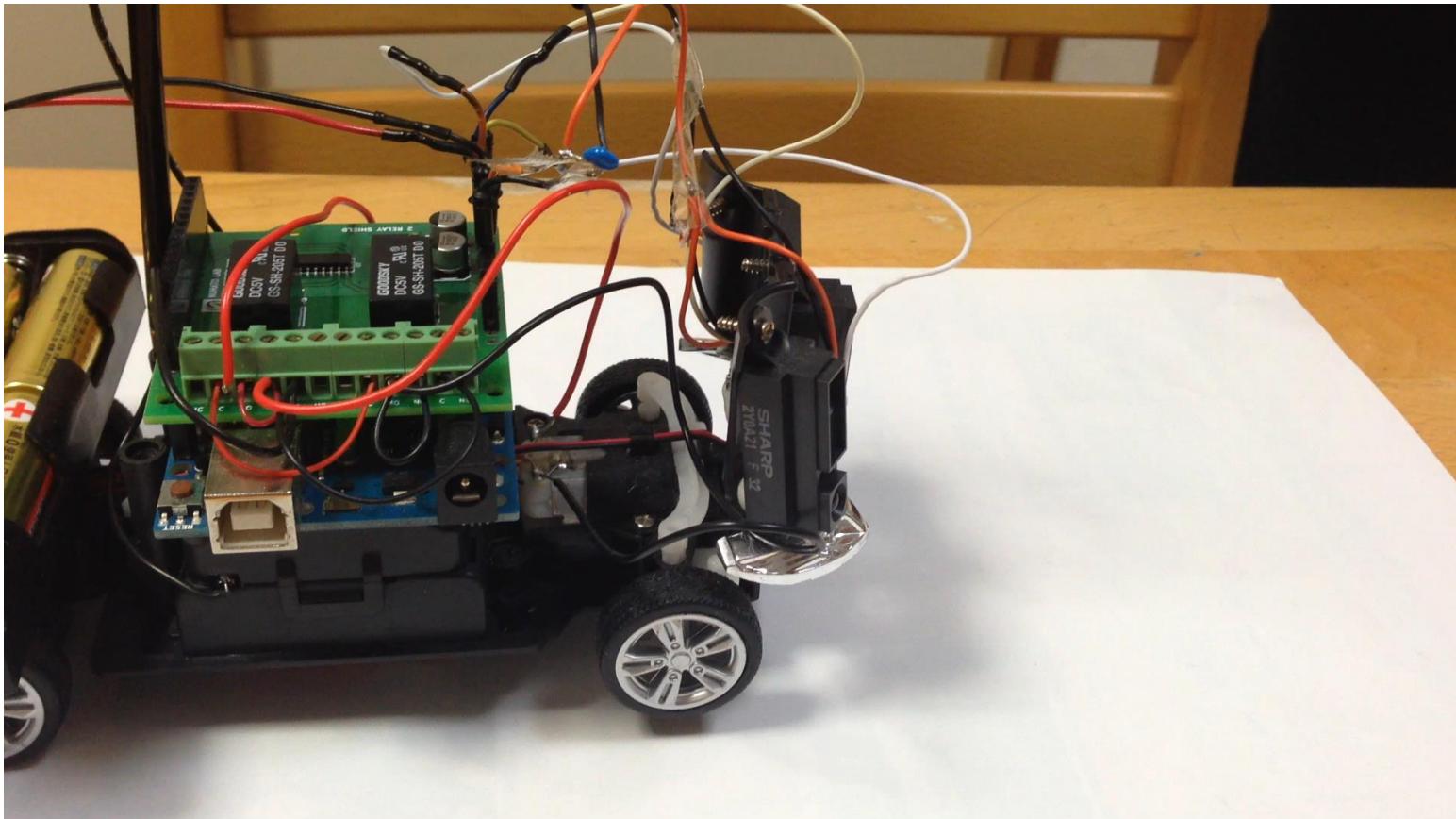
1号車（自走型）

# 2号車の説明

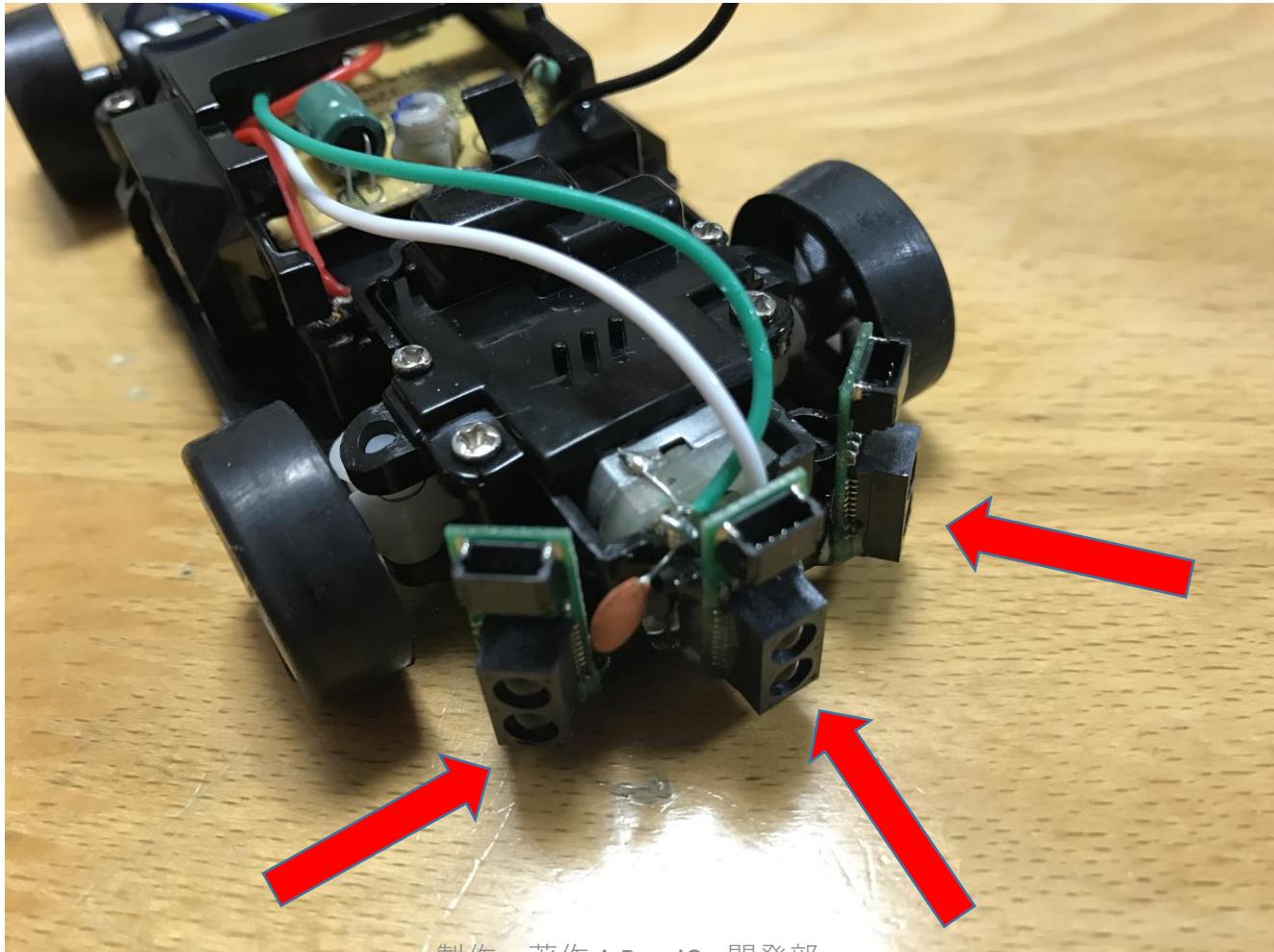


# センサー動作試験（動画）

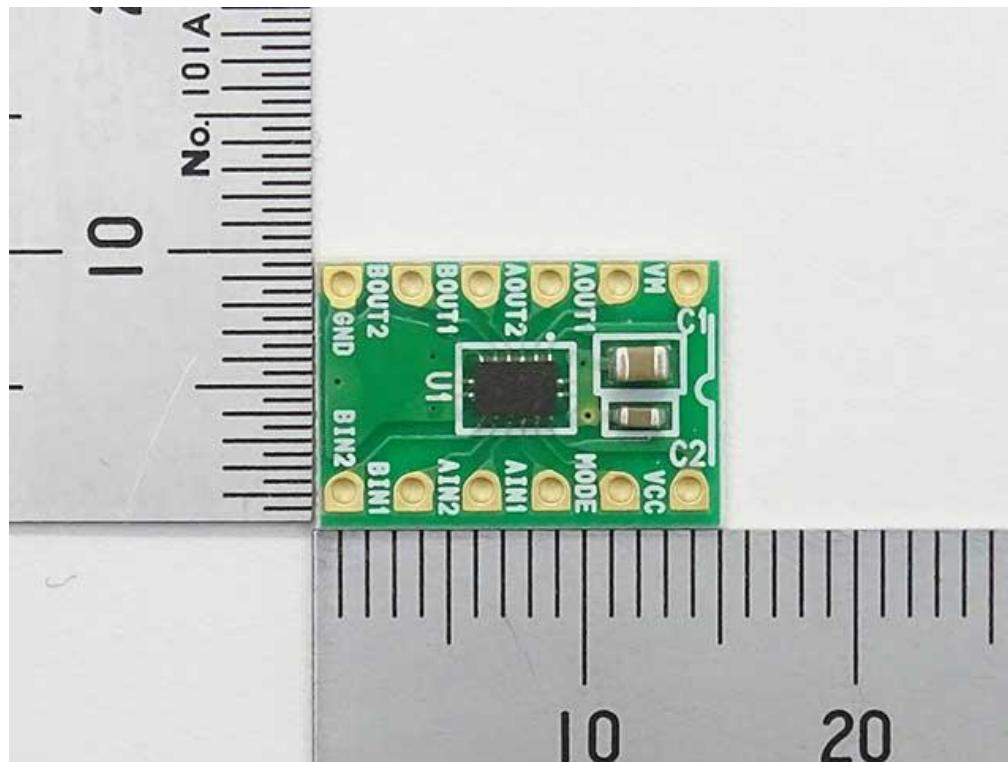
[動画のYouTubeリンク](#)



# 3、4号車センサー部分



# モータドライバ



DRV8835

DRV8835 秋月電子通商より

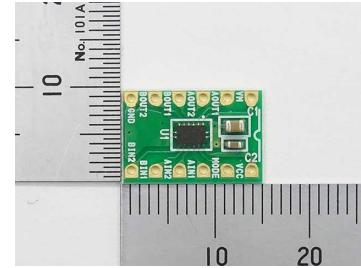
# モータドライバ

- RumiCarに搭載されているモータドライバはDRV8835です

DRV8835

- DRV8835の特徴

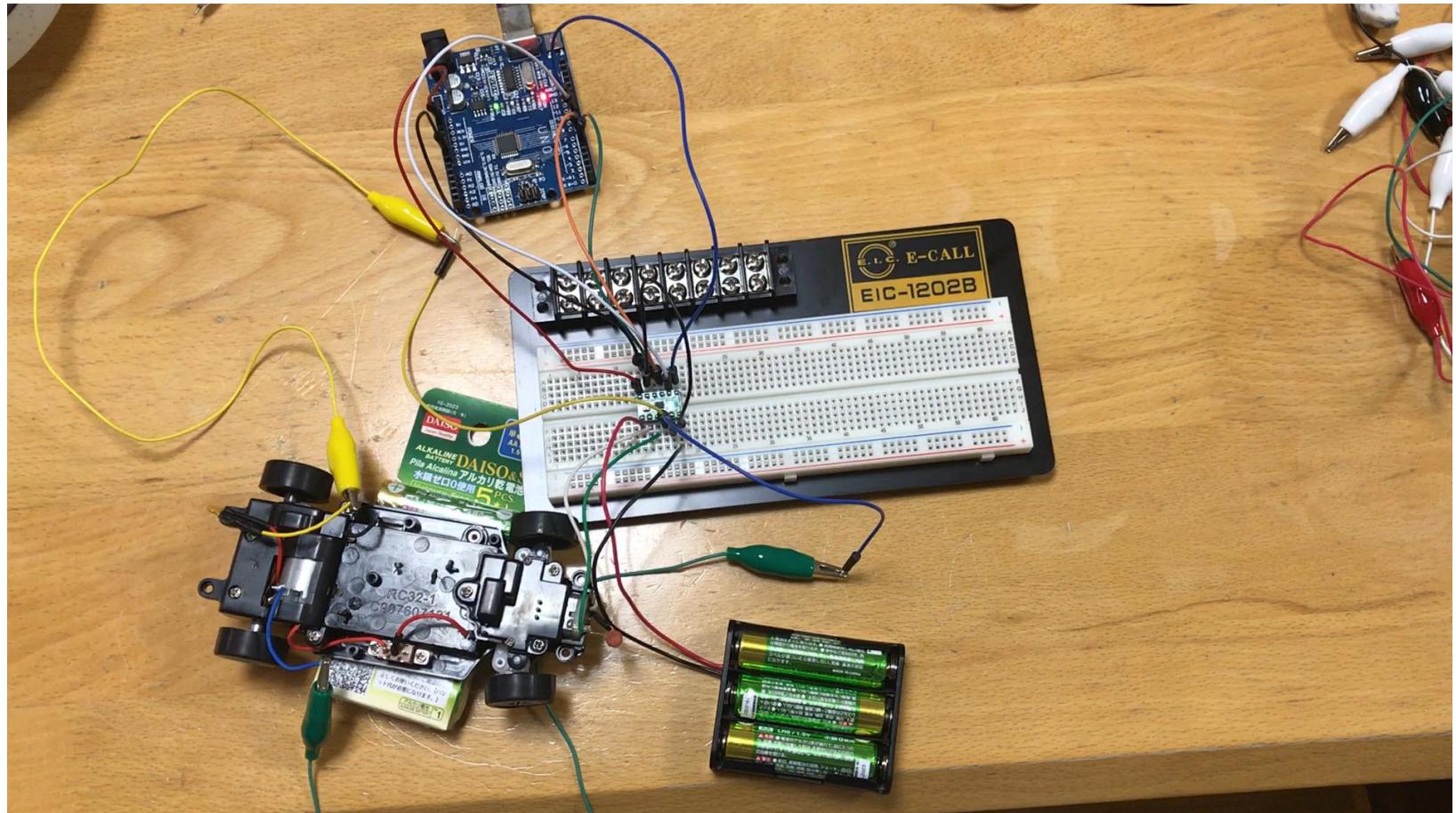
- 2 チャンネル
- 各チャネル1.5Aまで流せる
- PWM(250kHz)
- 低消費電力
- 低電圧降下（モータ側）
- ロジック電源（VCC）とモータ電源が独立している
  - ロジック電源電圧は2から7V
  - モータ電源電圧は11Vまで



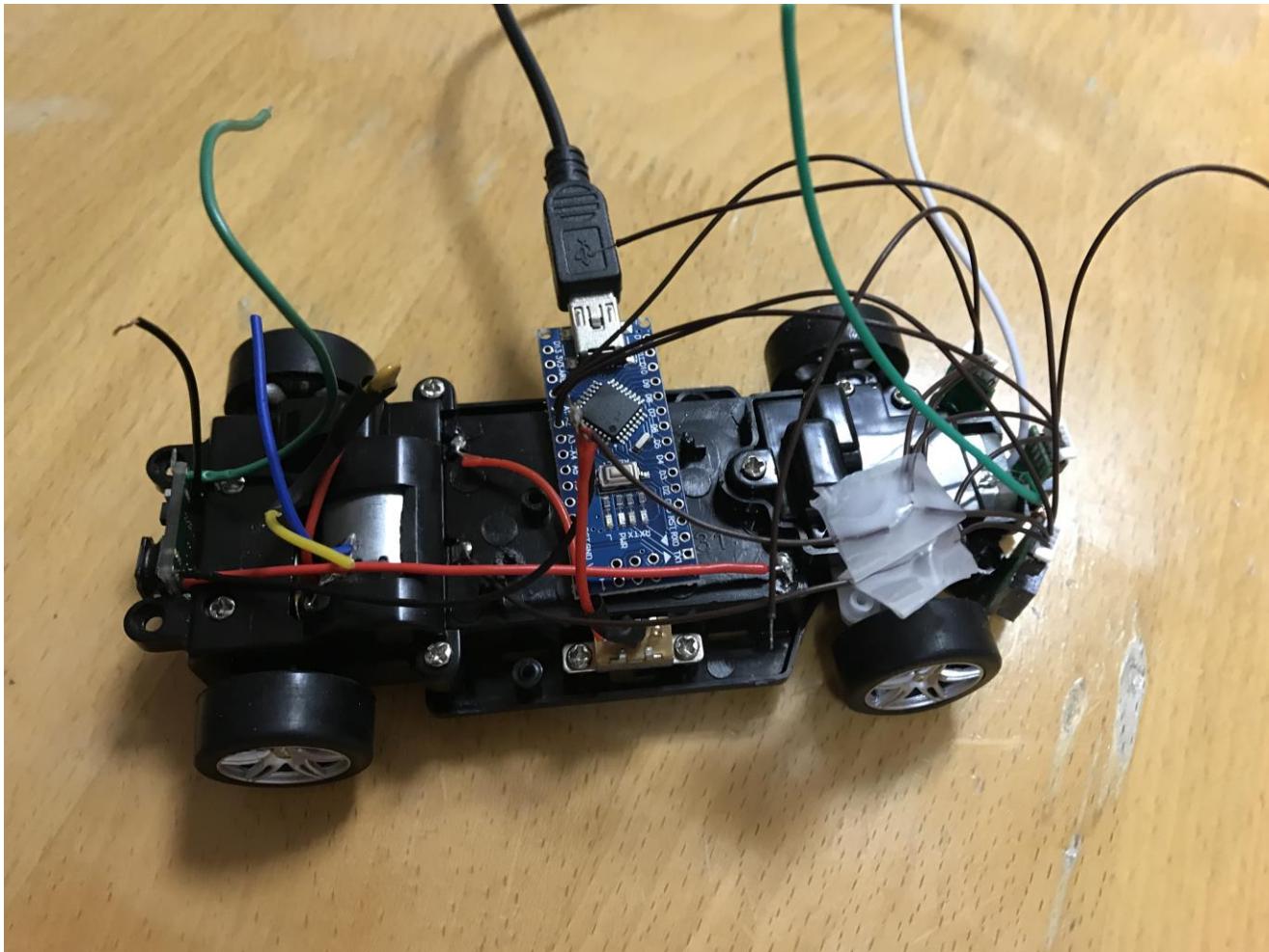
DRV8835 秋月電子通商より

# 開発中動画（動画）

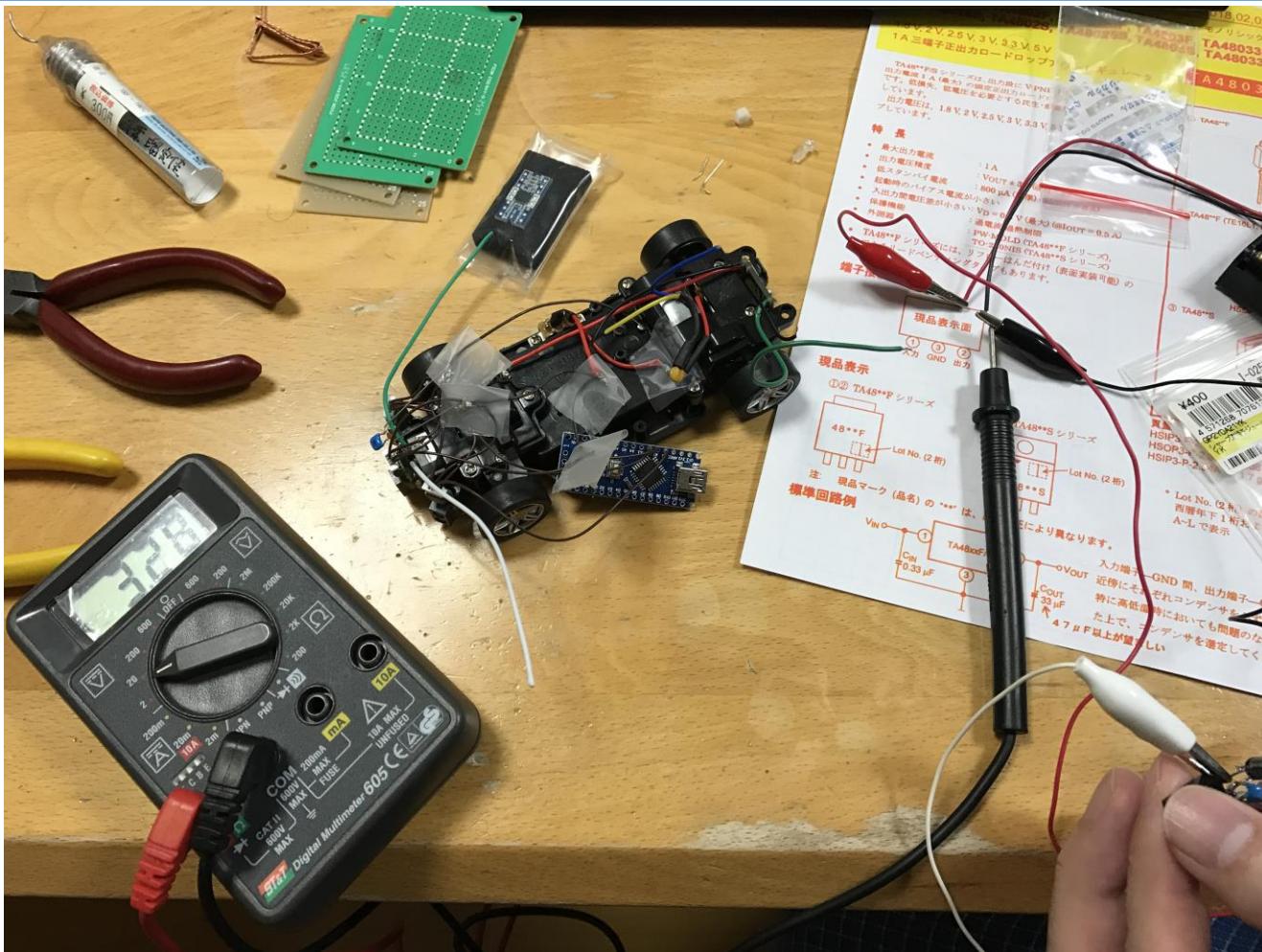
[動画のYouTubeリンク](#)



# 開発中写真

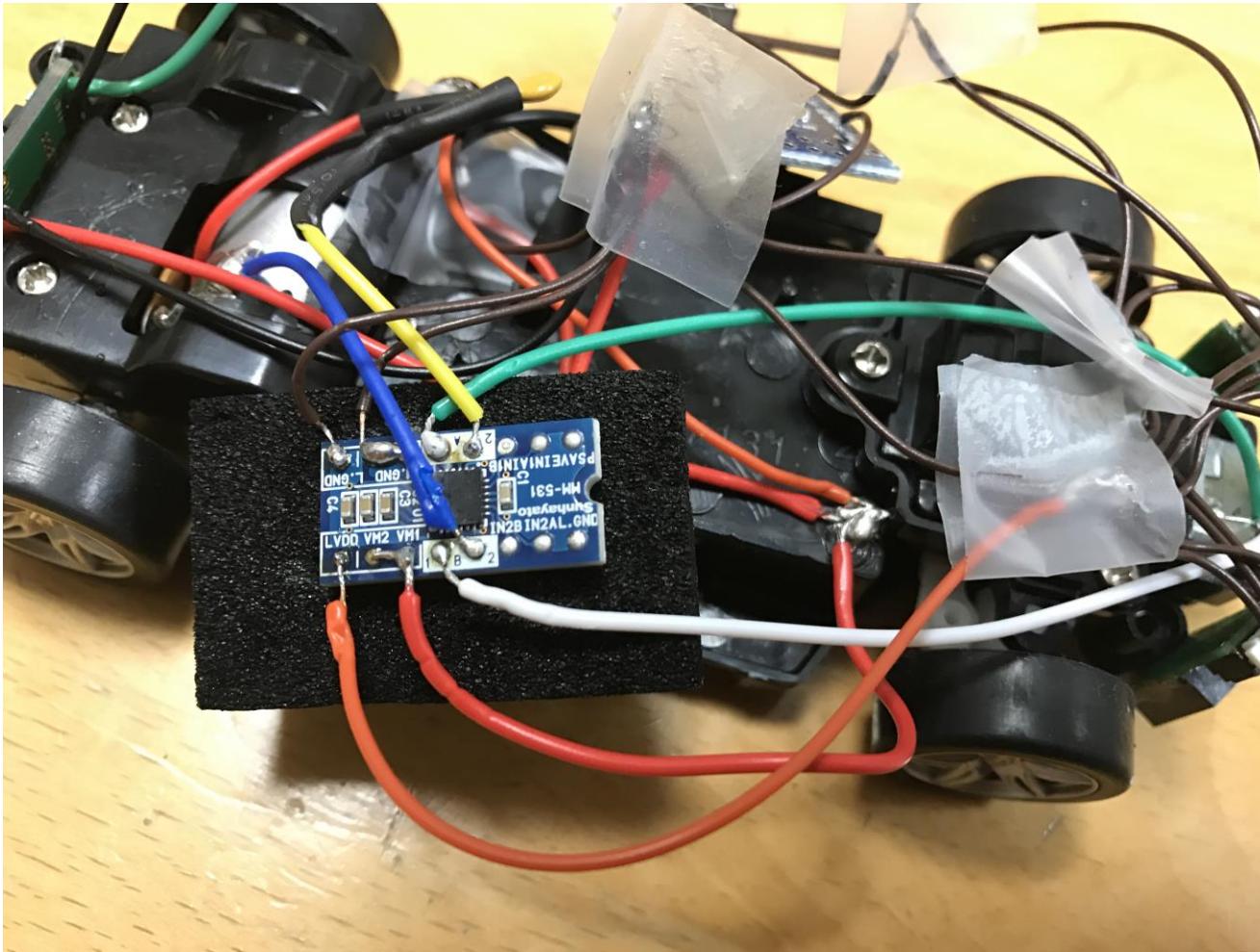


# 開発中写真



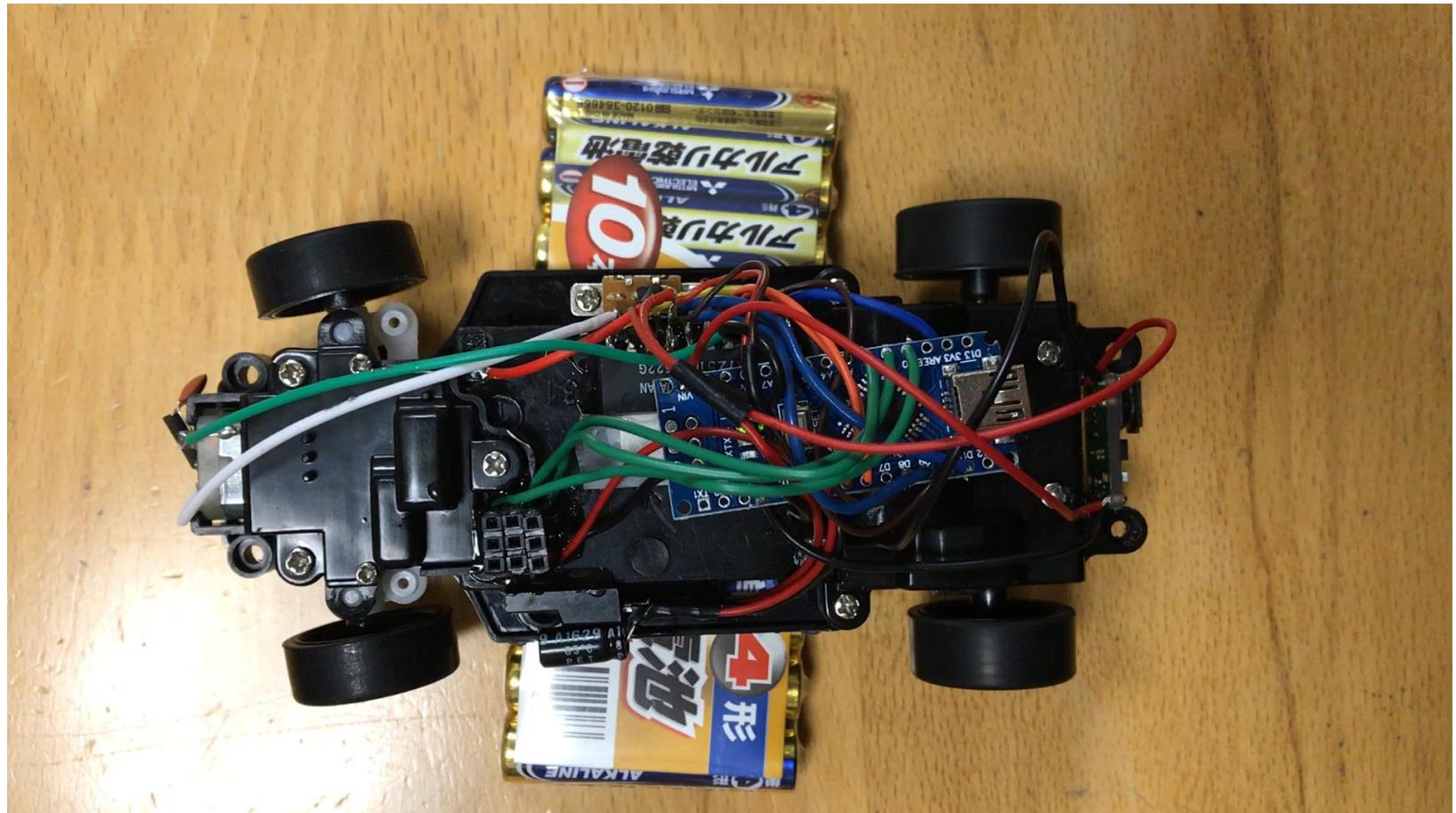
製作・著作：RumiCar開発部

# 開発中写真



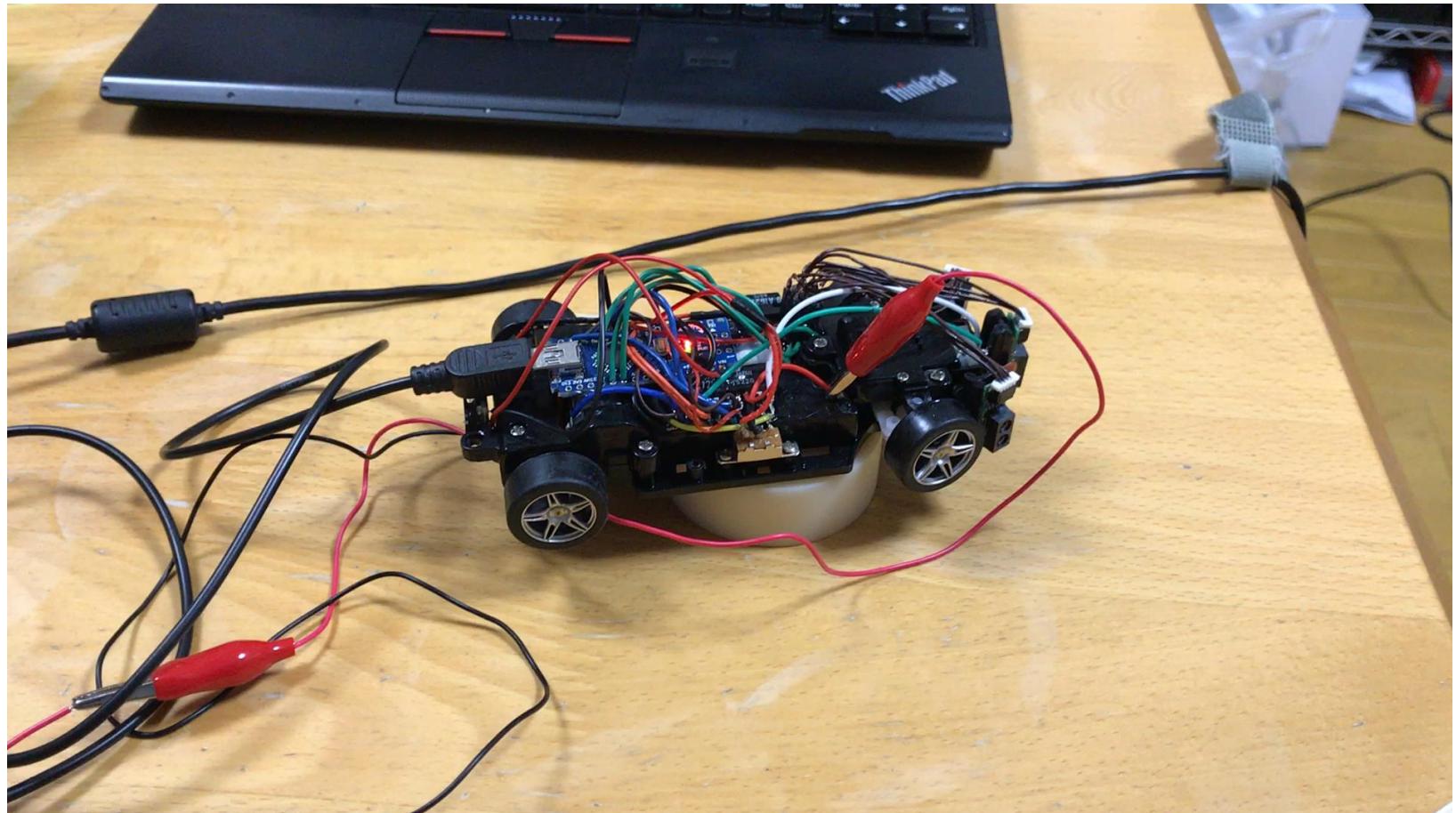
# 動作試験（動画）

[動画のYouTubeリンク](#)



# 動作試験（動画）

[動画のYouTubeリンク](#)

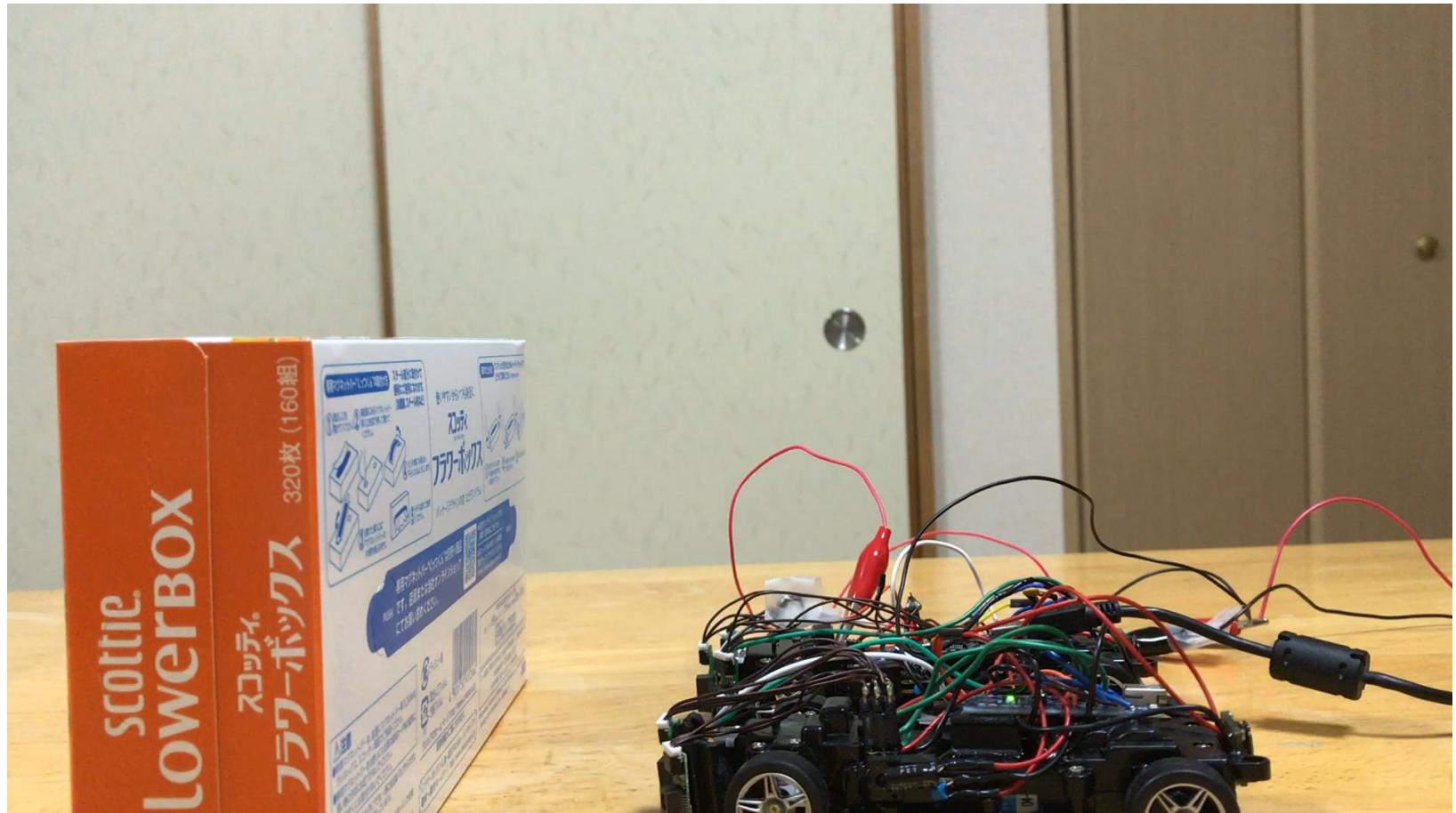


# 4号車システム部分

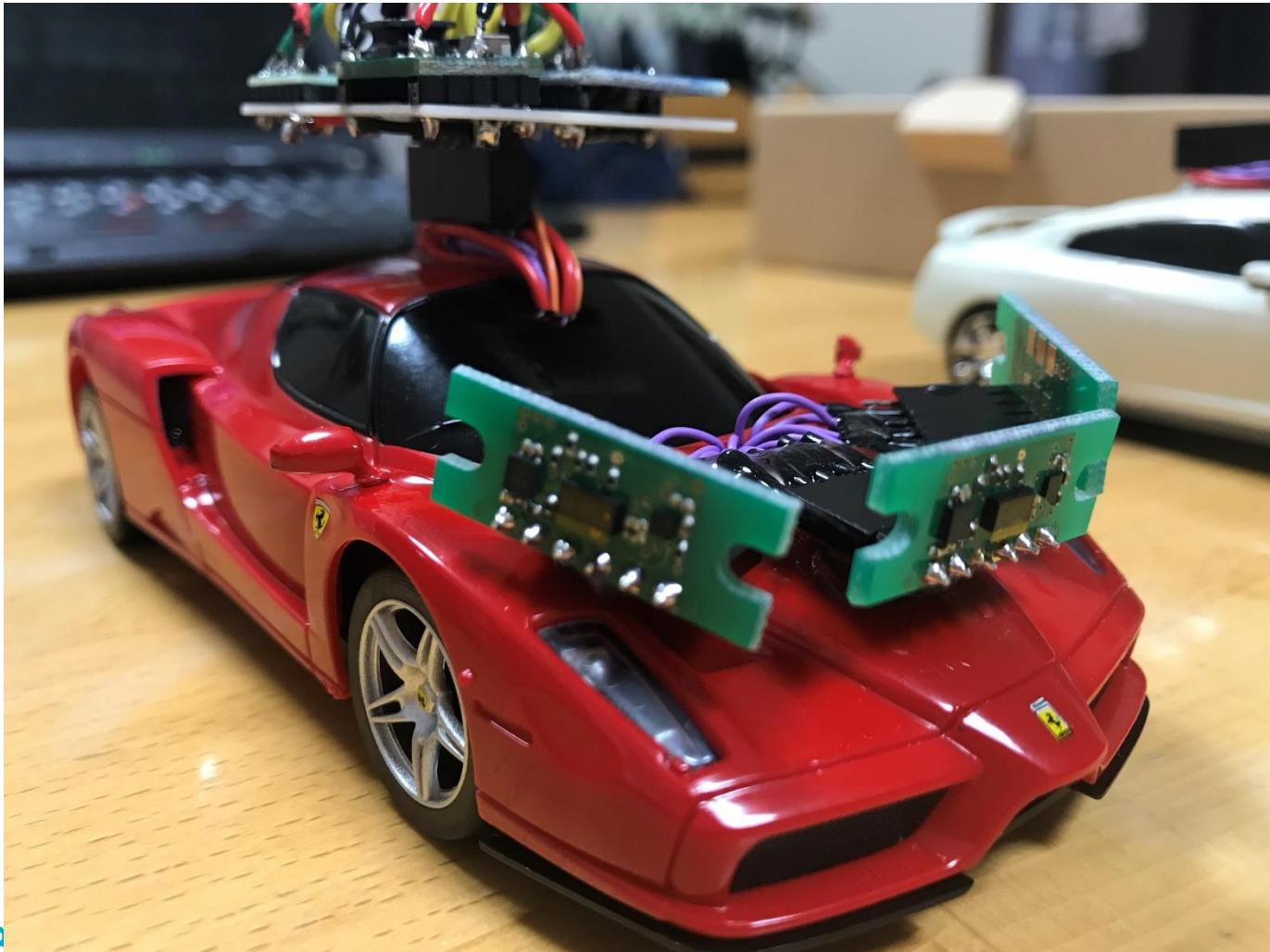


# 動作試験（動画）

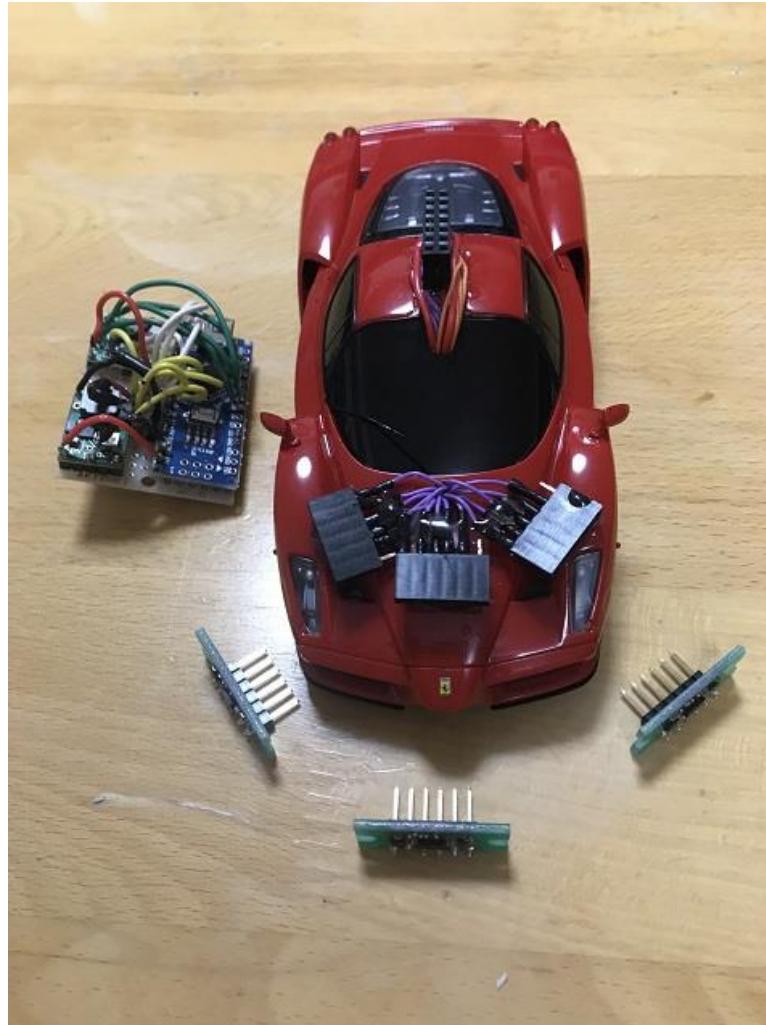
[動画のYouTubeリンク](#)



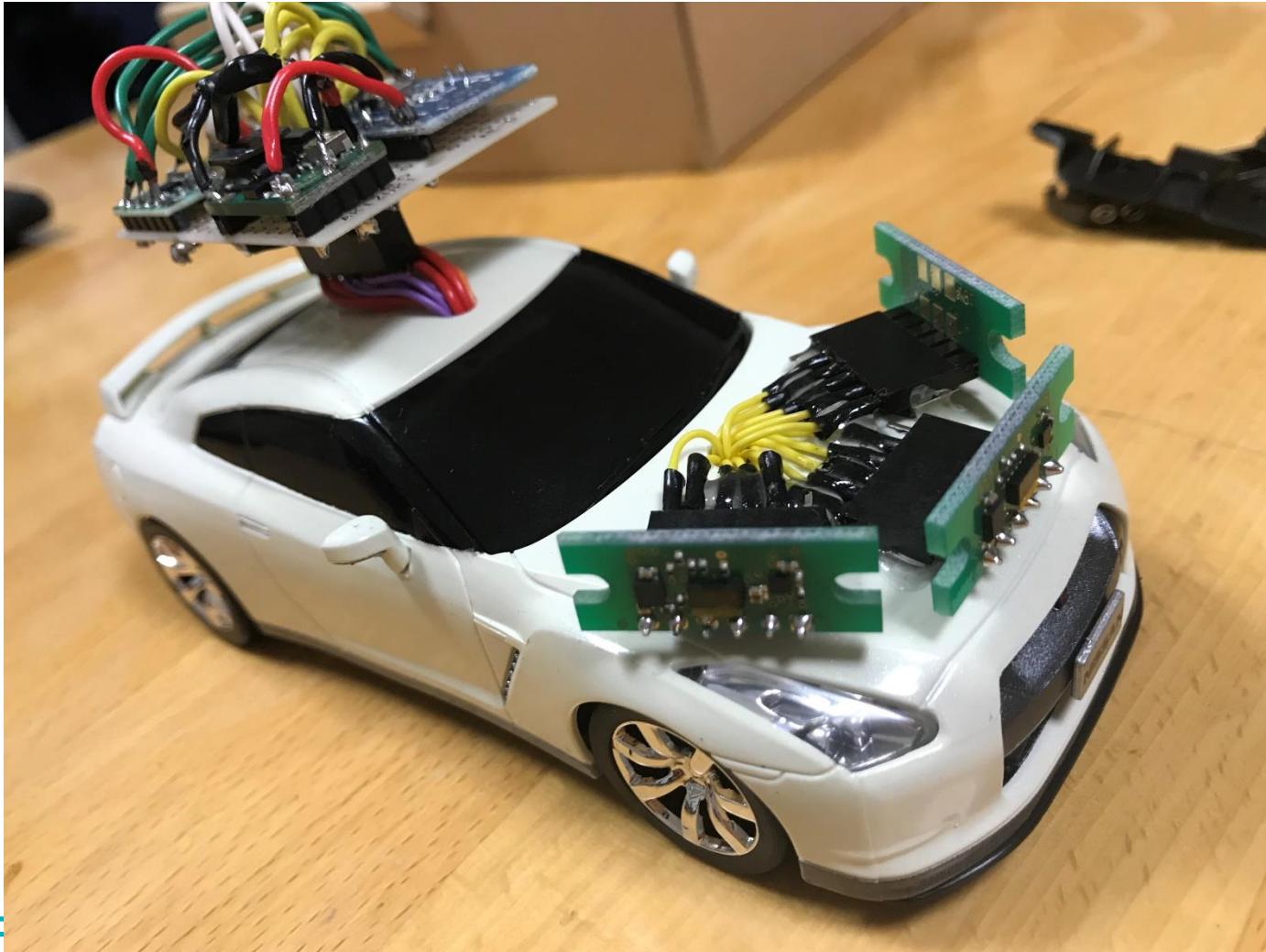
# 6号車



# 6号車



7号車



# インターフェイスの標準化

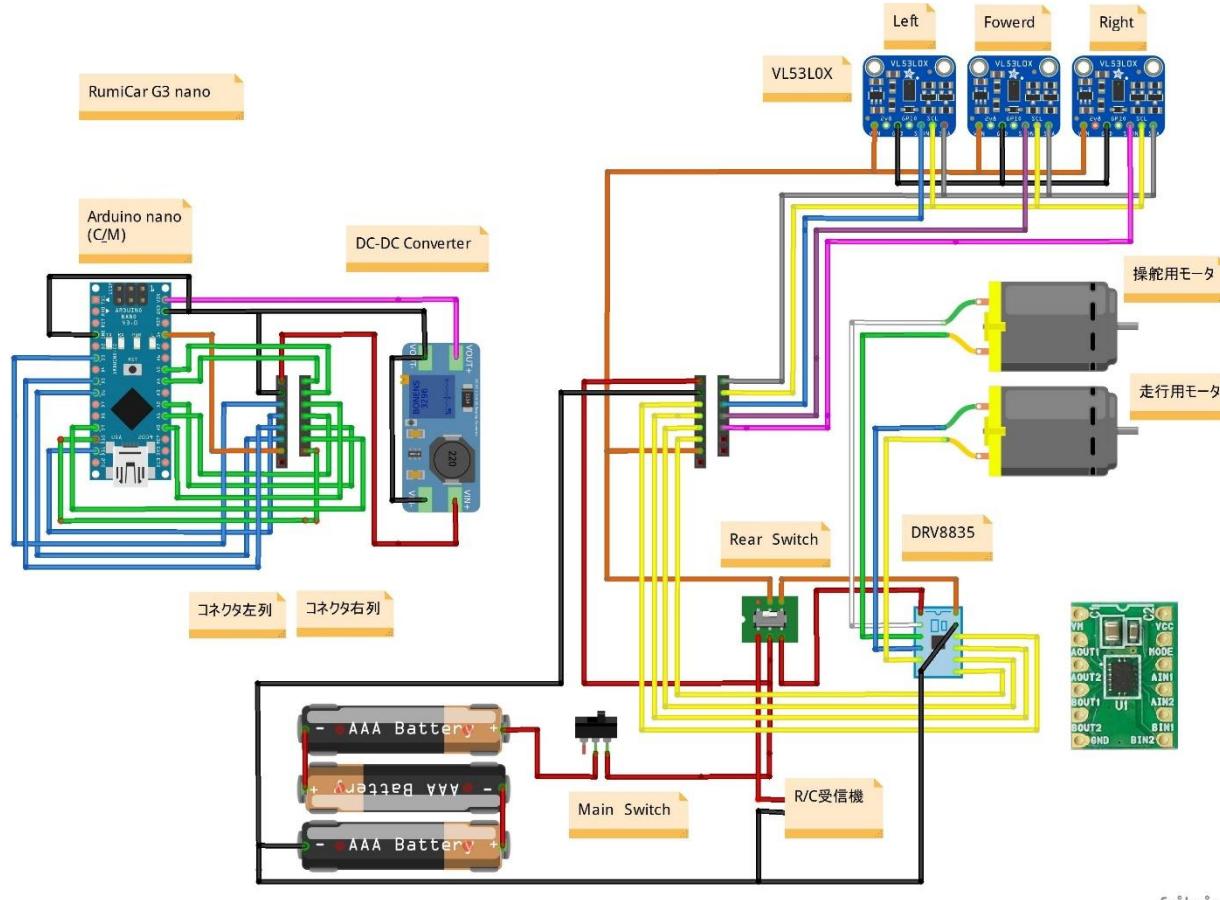


# RumiCarの歴史と構成

号車	1号車	2号車	3号車	4号車	5号車
ベース車スケール	1/24			1/32	
コントローラー	Arduino UNO		Arduino Nano		モジュール式 (選択交換可能)
モータ制御	リレーシールド	MM-531	TA7291P		DRV8835
測距モジュール	GP2Y0A21YK	GP2Y0E02A			VL53L0X
測距方式		赤外線			レーザー
Arduino電源	単三乾電池 6本 (9V)	単四乾電池(4.5V)よりLMR62421を使い 7Vに昇圧			
測距モジュール 電源	Arduinoより供給		TA48033Sで3.3Vを生成		

モータドライバBD6212HFPは小さすぎて使用断念

# 第3世代配線図



fritzing

# インターフェイス

従来

1	8
2	9
3	10
4	11
5	12
6	13
7	14

NEW

1	8
2	9
3	10
4	11
5	12
6	13
7	14

7.5

誤挿入防止用ダミーピン

# 8×7ピン インターフェイス

表 3 第三世代 車体-CM 接続インターフェイス(EX-IF)

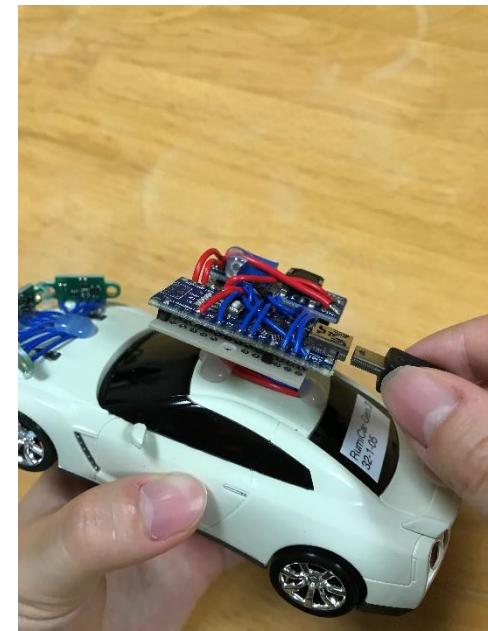
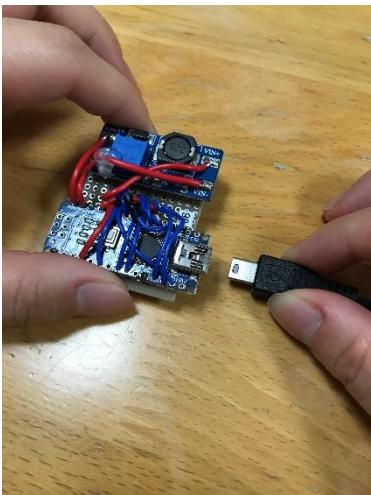
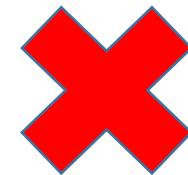
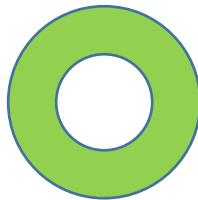
I/O	用途	ピン番号		用途	I/O
O	電源(4.5V)	1	8	SDA	I
O	GND	2	9	SCL	I
I	AIN1:右折側: <b>A3: E-GPIO04</b>	3	10	SHDN0(左)	I
I	AIN2:左折側: <b>A11: E-GPIO26</b>	4	11	SHDN1(中)	I
I	BIN1:前進側: <b>A5: E-GPIO27</b>	5	12	SHDN2(右)	I
I	BIN2:後進側: <b>A6: E-GPIO25</b>	6	13	<b>SERVO 用 A9 番ピン</b>	I
I	信号最大電圧及び電力供給(通常コンピュータの VCC)(5V または 3.3V)VL53L0X と DRV8835 の VCC へ	7	14	<b>SERVO 用 A10 番ピン</b>	I

誤挿入防止用7.5をダミーピンとして追加

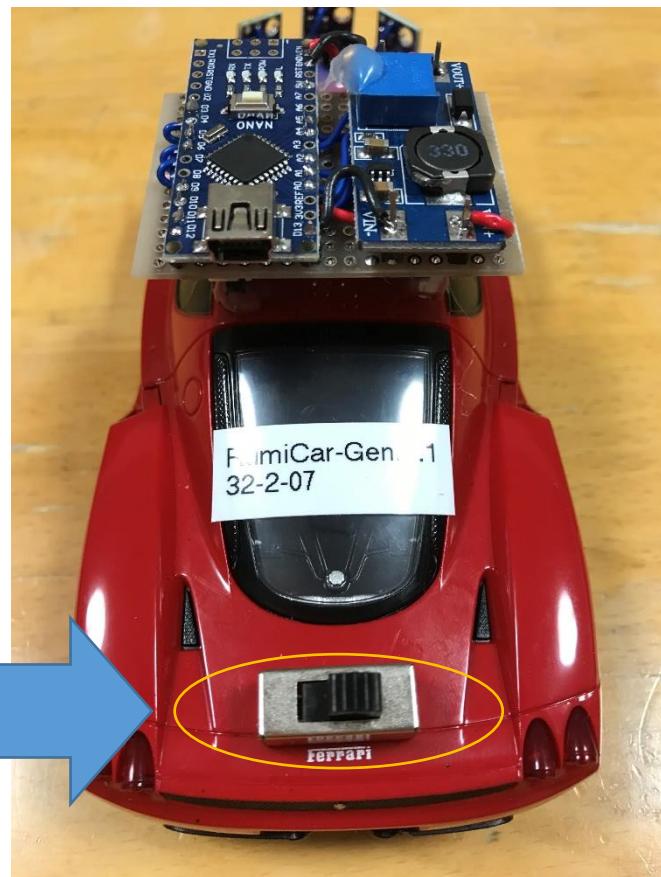
# 準備

注意点、必要なもの、準備

# ケーブルの抜き差し注意点



# コンピュータ制御へ設定



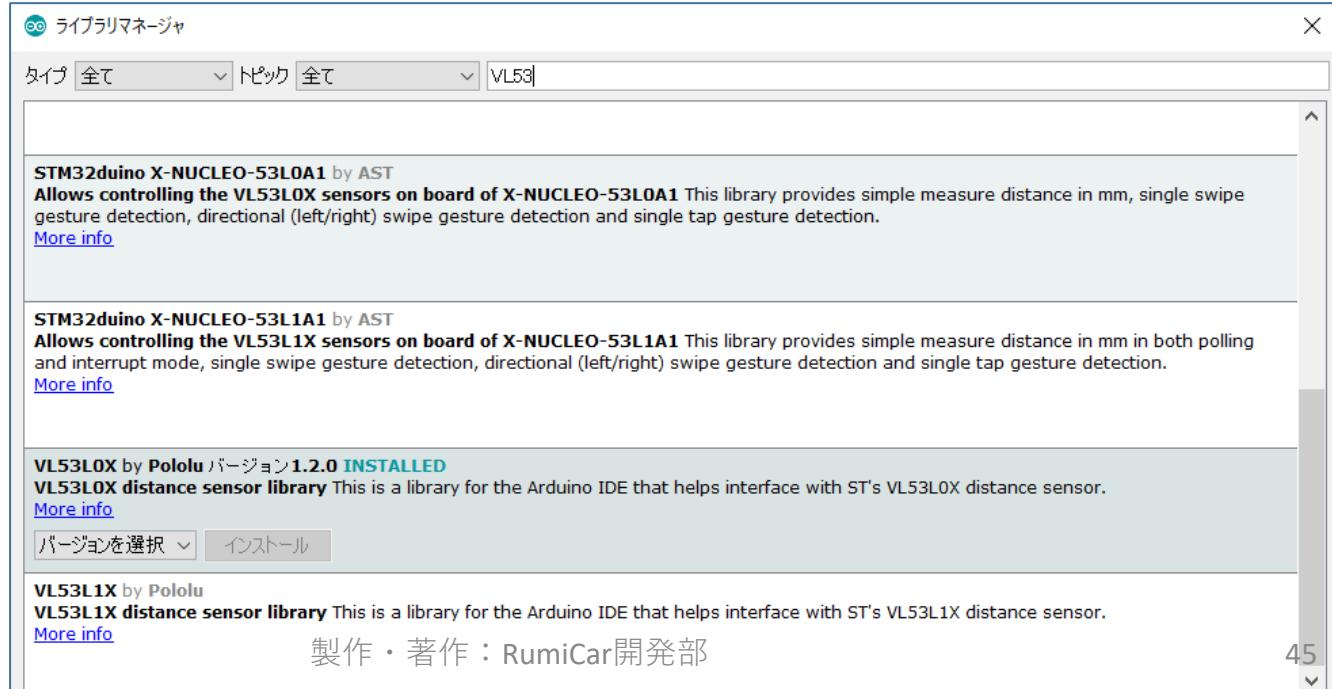
- (一部のモデルのみ)
- マニュアル制御とコンピュータ制御の切り替えスイッチ好きモデルは切り替えスイッチを**右側**へ切り替えます
- 右側がコンピュータ制御側です
- 右側にするとCM（ArduinoやESP32）からの命令で走ります
- （左側にするとラジコンで操作できるようになります）

# ケーブルの抜き差し注意点

- まず、コンピュータモジュールにUSBケーブルを接続する
- それから、コンピュータモジュール(以下CMと呼びます)を車体に接続する
- **注意！**車体にCMを装着した状態でコンピュータモジュールにUSBケーブルを接続しないこと
- 車体にCMを装着したままUSBケーブルを抜かないこと
- CMコネクタ部分に過度な力が加わり破損の原因になります

# 必要なもの

- Arduino IDE
- VL53L0Xライブラリ (by Pololu) がインストールされていること



STM32duino X-NUCLEO-53L0A1 by AST  
Allows controlling the VL53L0X sensors on board of X-NUCLEO-53L0A1 This library provides simple measure distance in mm, single swipe gesture detection, directional (left/right) swipe gesture detection and single tap gesture detection.  
[More info](#)

STM32duino X-NUCLEO-53L1A1 by AST  
Allows controlling the VL53L1X sensors on board of X-NUCLEO-53L1A1 This library provides simple measure distance in mm in both polling and interrupt mode, single swipe gesture detection, directional (left/right) swipe gesture detection and single tap gesture detection.  
[More info](#)

**VL53L0X by Pololu バージョン1.2.0 INSTALLED**  
VL53L0X distance sensor library This is a library for the Arduino IDE that helps interface with ST's VL53L0X distance sensor.  
[More info](#)

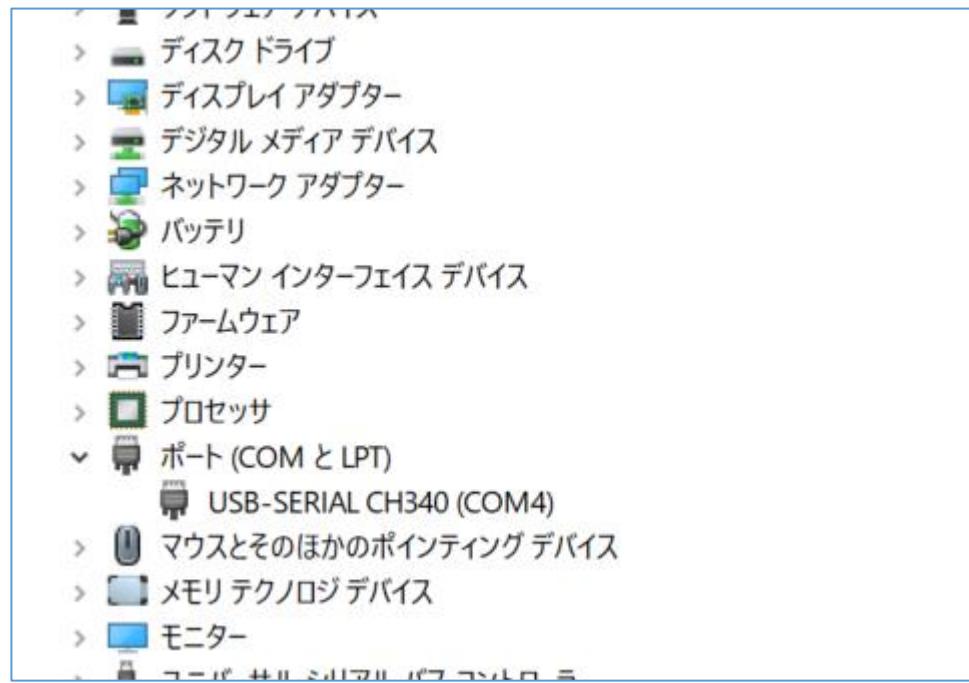
バージョンを選択 インストール

VL53L1X by Pololu  
VL53L1X distance sensor library This is a library for the Arduino IDE that helps interface with ST's VL53L1X distance sensor.  
[More info](#)

製作・著作：RumiCar開発部

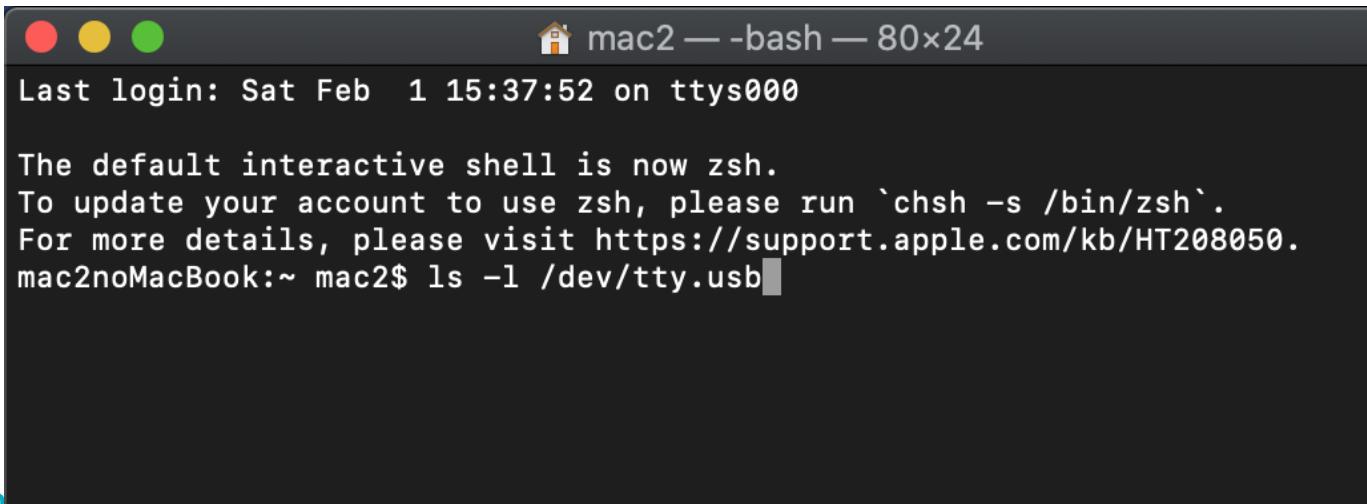
# COMポートの確認 (Windowsの場合)

- RumiCarのコンピュータモジュールをUSBケーブルでパソコンに接続する
- デバイスマネージャで接続されたCOMポート番号を確認する。**CH340を探します**（番号を覚えておくこと）



# COMポートの確認 (Macの場合)

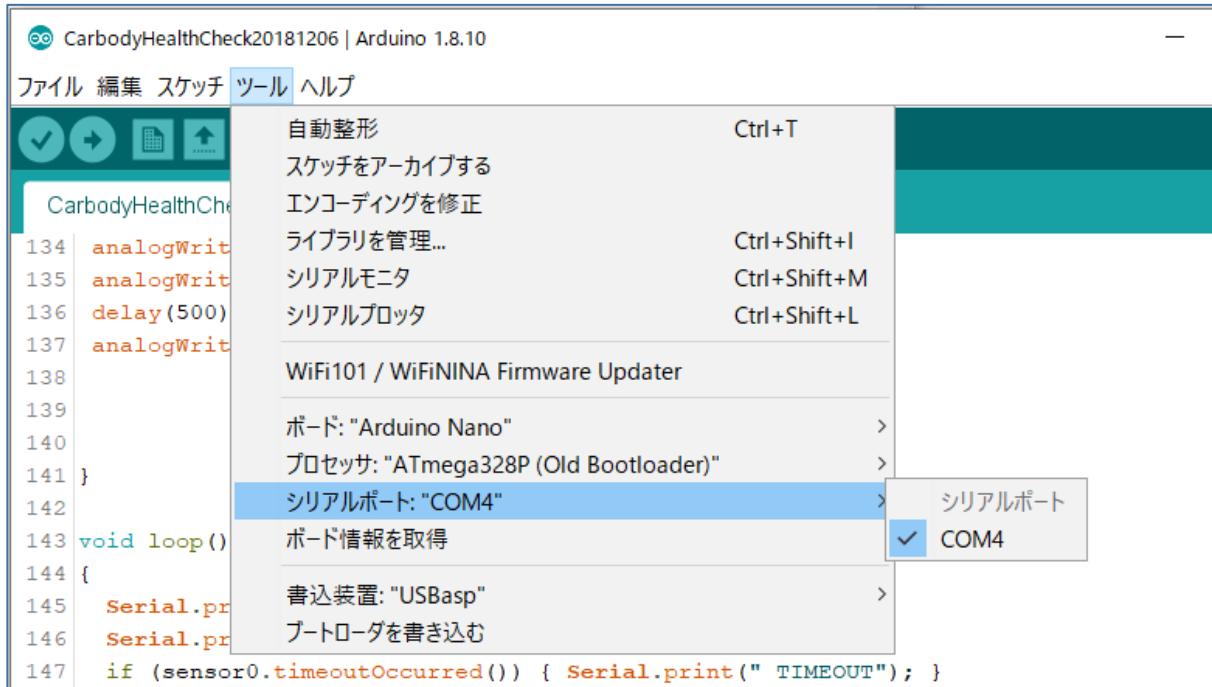
- Terminalソフトを起動します
- コマンド”ls -l /dev/tty.usb”と入力することで接続されているCOMポートの一覧を表示させることができます
- Arduinoが接続されたCOMポート番号を確認する  
CH340を探します（番号を覚えておくこと）



```
mac2 — bash — 80x24
Last login: Sat Feb  1 15:37:52 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
mac2noMacBook:~ mac2$ ls -l /dev/tty.usb
```

# Arduino IDEの設定

- ボード : Arduino Nano
- プロセッサ : ATmega328P(Old Bootloader)
- シリアルポート : デバイスマネージャで確認した番号

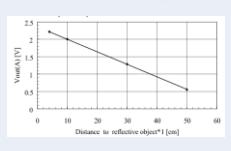


# 距離の測定

レーザー測距モジュール

# 測距モジュール

## ・測距モジュールの比較

型番	GP2Y0A21YK0F	GP2Y0E02A	VL53L0X	HC-SR04
測定方法	赤外線	赤外線	レーザー	超音波
出力	アナログ 	アナログ 	デジタル値	アナログ
推奨電源電圧	5V	3.3V	2.6-5.5V (ボード依存)	5V
消費電力	最大40mA	最大36mA	20mA	15mA
測定可能距離	80cm	50cm	2m	2m
RumiCarでの採用	1、2号車	3、4号車	5号車以降	非採用
備考	出力と距離が比例しない			高速移動時のエコーピンセル処理問題

グラフはシャープ社データシートより

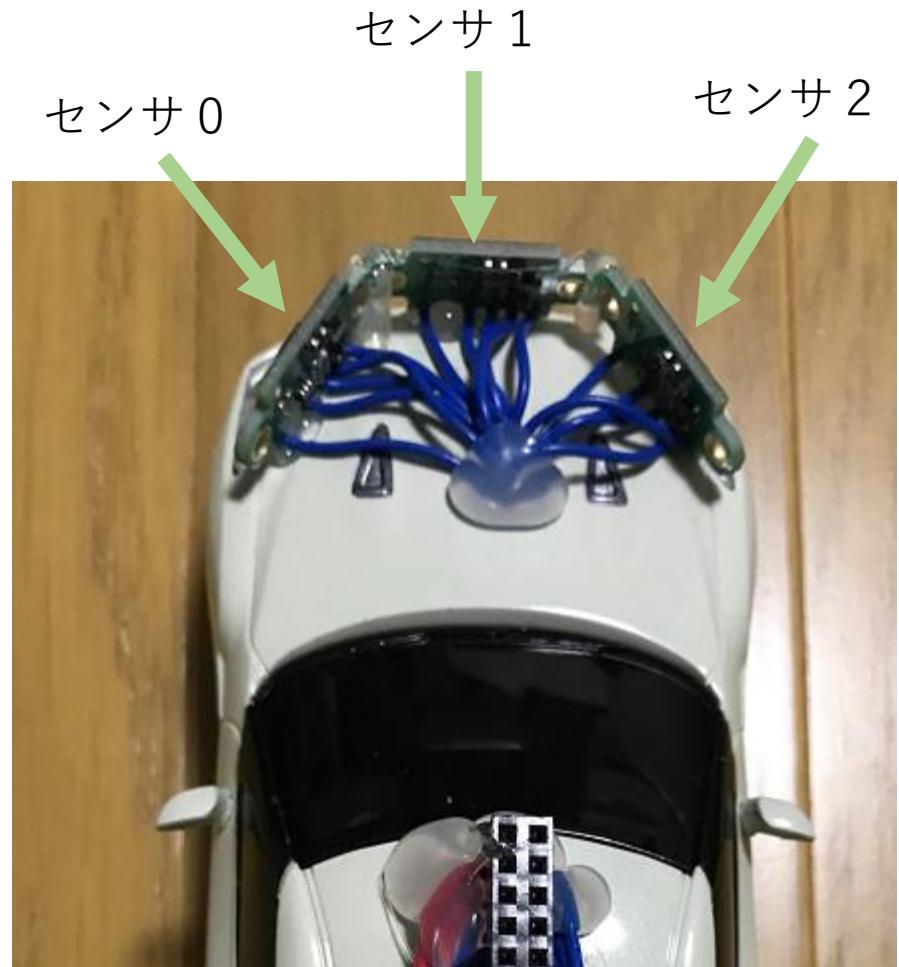
# レーザー測距モジュール

- 車体前方に 3 基搭載

左 : 0

中央 : 1

右 : 2



# VL53L0X

- RumiCarに搭載されているレーザー測距モジュールはVL53L0Xです
- VL53L0Xの特徴
  - 最大 2 m まで計測可能
  - I2Cで測定値を取得可能
  - I2Cアドレスはソフトウェアにより再設定可能
  - パラメータ設定により大まかにモード設定可能
    - 高速測定モード
    - 長距離測定モード
    - 高精度測定モード
    - サンプルプログラムは高速測定モードに設定されています

# 測距方法

- 各測距モジュールの測距値の取得は、例えば下記の命令を使います。結果は整数値(int)で mm(ミリメートル) の単位で得ることができます

`readRangeSingleMillimeters()`

- センサ1の測定値を取得するには例えば下記のようにします  
`i = sensor1.readRangeSingleMillimeters();`

# GitHubからExerciseをDL

<https://github.com/algyan/RumiCar>

algyan / RumiCar

Watch 1 Star 0 Fork 4

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security 0 Insights Settings

「RumiCarハンズオン中継！自動運転アルゴリズムを楽しく手軽に体感しよう！#2」で使用するExerciseです。Exercise-A2.3とExercise-A3.1は上級編なので、イベント中には使わないです。 <https://algyan.connpass.com/event/169...>

Manage topics

15 commits 3 branches 0 packages 0 releases 3 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

Rumika685	Create README.md	Latest commit d4baa603 4 minutes ago
Excise-1.1	Split to RumiCar Lib.	2 days ago
Excise-1.2	Split to RumiCar Lib.	2 days ago
Excise-2.1	Split to RumiCar Lib.	2 days ago
Excise-2.2	Split to RumiCar Lib.	2 days ago
Excise-2.3	Split to RumiCar Lib.	2 days ago
Excise-2.4	Split to RumiCar Lib.	2 days ago
Excise-3.1	Split to RumiCar Lib.	2 days ago
Excise-3.2	Split to RumiCar Lib.	2 days ago
Excise-A2.3	Split RumiCar Lib.	2 days ago
Excise-A3.1	Split RumiCar Lib.	2 days ago
Excise	Split to RumiCar Lib.	2 days ago
README.md	Create README.md	4 minutes ago

**【注意】Exercise-A2.3と3.1は、このイベントでは使用しません。**



Exerciseのヘッダー作製、リファクタリング、上級編プログラムの追加をしてくださったのは  
**稻玉 繁樹 氏**

# Exercise-1

距離を測ってみよう

# Exercise-1.1

中央のセンサで測距しよう！

# Exercise-1.1

## 中央のセンサで距離を測る(準備)

- RumiCarの車体の電源をオフにします
- CMをRumiCarの車体より取り外してください
- CMにUSBケーブルを接続します
- USBケーブルをパソコンに接続します
- CMをRumiCarの車体に取り付けます

# Exercise-1.1

## 中央のセンサで距離を測る

- Arduino IDEでファイル”Exercise”を開きます
- プログラムの下の方のところに計測値を表示するコードを追加します

```
225  
226 void loop()  
227 {  
228  
229 }
```



- Serial.print命令やSerial.println命令でシリアルモニタにデータを表示できるよ
- Serial.printは改行無し
- Serial.printlnは表示後改行ありだよ

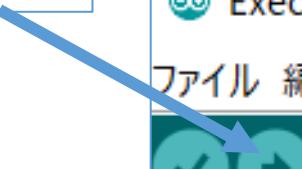
```
225  
226 void loop()  
227 {  
228 Serial.println(sensor1.readRangeSingleMillimeters());  
229 }
```

# Exercise-1.1

## 中央のセンサで距離を測る

- コンパイルとボードへの書き込み

押す

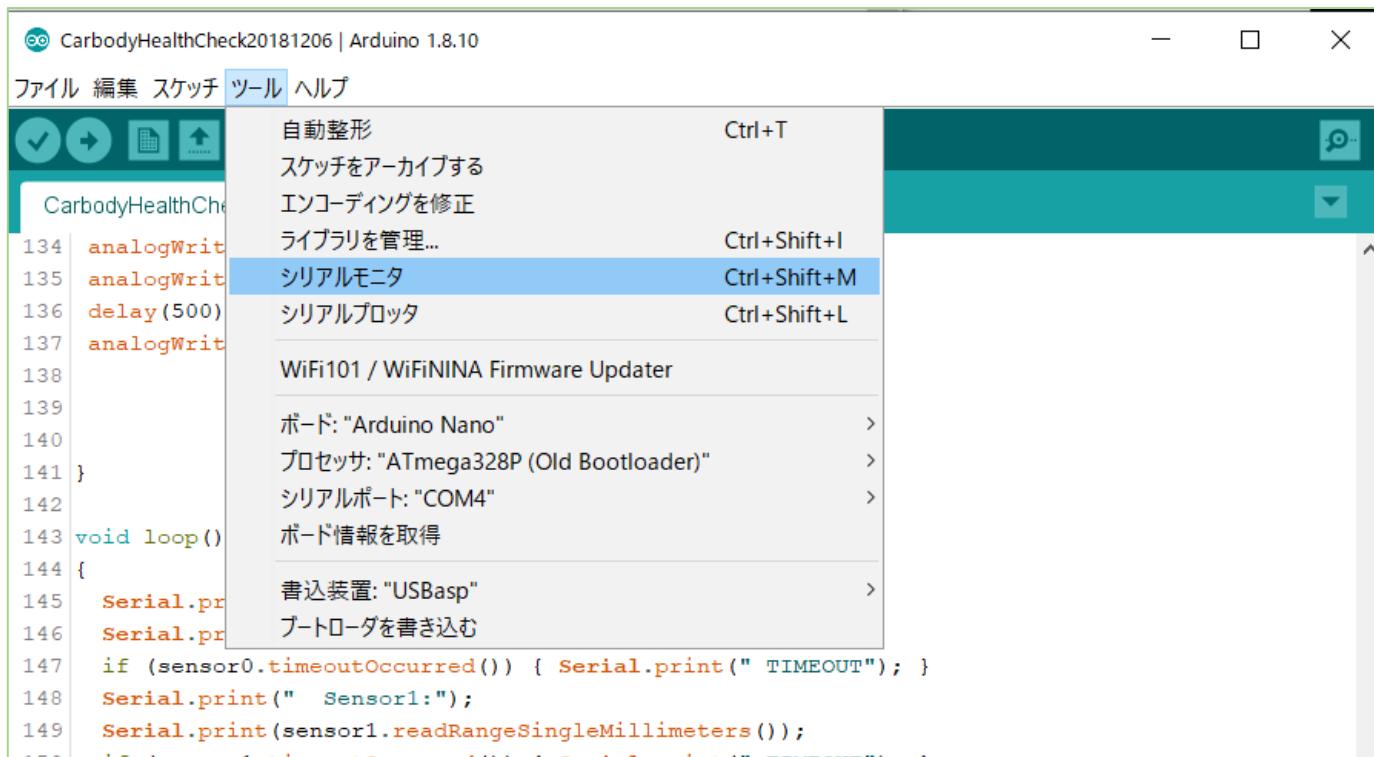


```
ExerciseAnswerkey-1.1 | Arduino 1.8.10
ファイル 編集 スケッチ ツール ヘルプ
Execute Answerkey-1.1
211
212 void loop()
213 {
214
215 Serial.println(sensor1.readRangeSingleMillimeters());
216 }
```

# Exercise-1.1

## 中央のセンサで距離を測る（続き）

- シリアルモニタで測定値を表示させます
  - ツール -> シリアルモニタ



# Exercise-1.2

3つのセンサで測距だ！

# Exercise-1.2

## 3つのセンサーで距離を測る

- ファイル"Exercise-1.2"を開きます

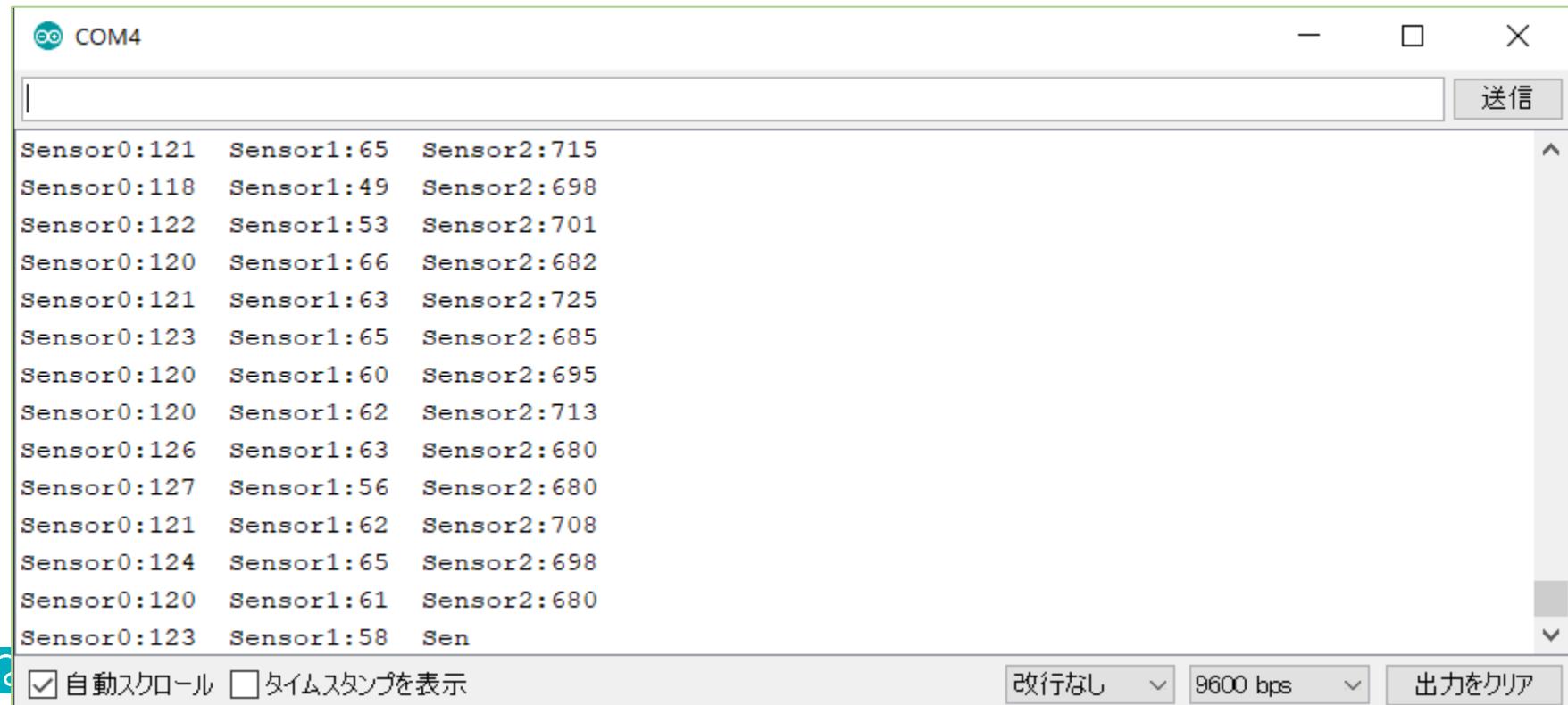
センサー	センサーの場所
Sensor0	左
Sensor1	中央
Sensor2	右

```
123
124 void loop()
125 {
126     Serial.print("Sensor0:");
127     Serial.print(sensor0.readRangeSingleMillimeters());
128     Serial.print(" Sensor1:");
129     Serial.print(sensor1.readRangeSingleMillimeters());
130     Serial.print(" Sensor2:");
131     Serial.println(sensor2.readRangeSingleMillimeters());
132 }
```

## Exercise-1.2

### 3つのセンサで距離を測る（続き）

- シリアルモニタで測定値を表示させる
  - ツール -> シリアルモニタ



The screenshot shows a Windows-style application window titled "COM4". The window has standard minimize, maximize, and close buttons at the top right. The main area is a text-based terminal window with a light gray background and white text. It displays a series of sensor readings in a tab-separated format. The data consists of three columns: Sensor0, Sensor1, and Sensor2. The values fluctuate between 120 and 715. At the bottom of the window, there are several control buttons: "自动スクロール" (Auto Scroll) with a checked checkbox, "タイムスタンプを表示" (Show timestamp), "改行なし" (No line break), "9600 bps" (Baud rate), and "出力をクリア" (Clear output). A vertical scroll bar is visible on the right side of the text area.

```
Sensor0:121  Sensor1:65  Sensor2:715
Sensor0:118  Sensor1:49  Sensor2:698
Sensor0:122  Sensor1:53  Sensor2:701
Sensor0:120  Sensor1:66  Sensor2:682
Sensor0:121  Sensor1:63  Sensor2:725
Sensor0:123  Sensor1:65  Sensor2:685
Sensor0:120  Sensor1:60  Sensor2:695
Sensor0:120  Sensor1:62  Sensor2:713
Sensor0:126  Sensor1:63  Sensor2:680
Sensor0:127  Sensor1:56  Sensor2:680
Sensor0:121  Sensor1:62  Sensor2:708
Sensor0:124  Sensor1:65  Sensor2:698
Sensor0:120  Sensor1:61  Sensor2:680
Sensor0:123  Sensor1:58  Sen
```

自動スクロール  タイムスタンプを表示  改行なし  9600 bps  出力をクリア

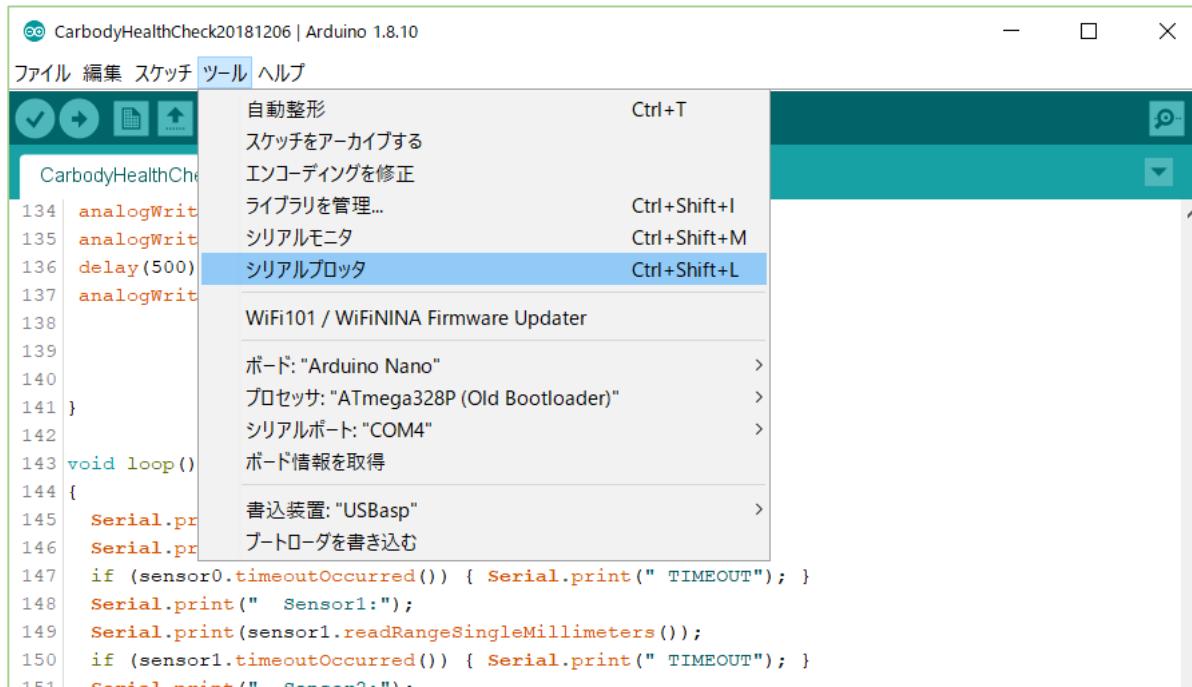
# Exercise-1.3

シリアルプロッタを使ってみよう

# Exercise-1.3

## シリアルプロッタを使ってみよう

- ・（プログラムはそのまま）
- ・シリルモニタを閉じる
- ・ツール -> シリアルプロッタ
  - ・※シリアルモニタのウインドウが開いている場合は閉じる必要があります



# 休憩室



	紫外線硬化樹脂	ホットメルト接着剤 (通称グルー)
長所	<ul style="list-style-type: none"><li>短時間で固まる</li><li>強度も高い</li><li>透明で見栄えが良い</li><li>重ね塗りが可能</li></ul>	<ul style="list-style-type: none"><li>安い</li><li>失敗しても剥がしてやり直しが効く</li></ul>
短所	<ul style="list-style-type: none"><li>長期間経過すると脆くなり崩れる</li></ul>	<ul style="list-style-type: none"><li>透明でないので見栄えが良くない (イ〇スタ映えしない)</li><li>紫外線硬化樹脂程強度がない</li></ul>

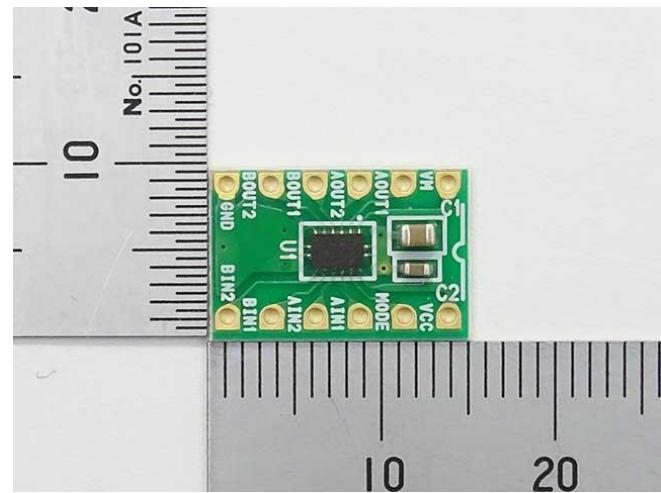


# モータ制御

ハンドルを動かす、走る

# モータドライバ

- RumiCarに搭載されているモータドライバはDRV8835です
- DRV8835の特徴 [DRV8835](#)
  - 2 チャンネル
  - 各チャネル1.5Aまで流せる
  - PWM(250kHz)
  - 低消費電力
  - 低電圧降下（モータ側）
  - ロジック電源（VCC）とモータ電源が独立している
    - ロジック電源電圧は2から7V
    - モータ電源電圧は11Vまで



# DRV8835諸元(1/2)

$T_A = 25^\circ\text{C}$  (unless otherwise noted)

		MIN	NOM	MAX	UNIT
$V_{CC}$	Device power supply voltage	2		7	V
$V_M$	Motor power supply voltage	0		11	V
$V_{IN}$	Logic level input voltage	0		$V_{CC}$	V
$I_{OUT}$	H-bridge output current <sup>(1)</sup>	0		1.5	A
$f_{PWM}$	Externally applied PWM frequency	0		250	kHz

(1) Power dissipation and thermal limits must be observed.

DRV8835 Dual Low-Voltage H-Bridge ICデータシート 2016年8月9日 より

# DRV8835諸元(2/2)

$T_A = 25^\circ\text{C}$ ,  $V_M = 5 \text{ V}$ ,  $V_{CC} = 3 \text{ V}$  (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
<b>POWER SUPPLY</b>						
$I_{VM}$	VM operating supply current	No PWM, no load	85	200		$\mu\text{A}$
		50 kHz PWM, no load	650	2000		
$I_{VMO}$	VM sleep mode supply current	$V_M = 2 \text{ V}$ , $V_{CC} = 0 \text{ V}$ , all inputs 0 V	5			$\text{nA}$
		$V_M = 5 \text{ V}$ , $V_{CC} = 0 \text{ V}$ , all inputs 0 V	10	95		
$I_{VCC}$	VCC operating supply current		450	2000		$\mu\text{A}$
$V_{UVLO}$	VCC undervoltage lockout voltage	$V_{CC}$ rising		2		$\text{V}$
		$V_{CC}$ falling		1.9		
<b>LOGIC-LEVEL INPUTS</b>						
$V_{IL}$	Input low voltage			0.3 $\times V_{CC}$		$\text{V}$
$V_{IH}$	Input high voltage		0.5 $\times V_{CC}$			$\text{V}$
$I_{IL}$	Input low current	$V_{IN} = 0$	-5	5		$\mu\text{A}$
$I_{IH}$	Input high current	$V_{IN} = 3.3 \text{ V}$		50		$\mu\text{A}$
$R_{PD}$	Pulldown resistance		100			$\text{k}\Omega$
<b>H-BRIDGE FETS</b>						
$R_{DS(ON)}$	HS + LS FET on resistance	$V_{CC} = 3 \text{ V}$ , $V_M = 3 \text{ V}$ , $I_O = 800 \text{ mA}$ , $T_J = 25^\circ\text{C}$	370	420		$\text{m}\Omega$
		$V_{CC} = 5 \text{ V}$ , $V_M = 5 \text{ V}$ , $I_O = 800 \text{ mA}$ , $T_J = 25^\circ\text{C}$	305	355		
$I_{OFF}$	OFF-state leakage current			$\pm 200$		$\text{nA}$
<b>PROTECTION CIRCUITS</b>						
$I_{OCP}$	Overcurrent protection trip level		1.6	3.5		$\text{A}$
$t_{DEG}$	Overcurrent de-glitch time			1		$\mu\text{s}$
$t_{OCR}$	Overcurrent protection retry time			1		$\text{ms}$
$t_{DEAD}$	Output dead time			100		$\text{ns}$
$t_{TSD}$	Thermal shutdown temperature	Die temperature	150	160	180	$^\circ\text{C}$

# IN/INモード

- IN/INモードとPHASE/ENABLEモードを持っています
- RumiCarではDRV8835をIN/INモードで利用します

Table 3. IN/IN Mode

MODE	xIN1	xIN2	xOUT1	xOUT2	FUNCTION (DC MOTOR)
0	0	0	Z	Z	Coast
0	0	1	L	H	Reverse
0	1	0	H	L	Forward
0	1	1	L	L	Brake

DRV8835 Dual Low-Voltage H-Bridge ICデータシート 2016年8月9日 より

# Phase/Enableモードの場合

Table 4. Phase/Enable Mode

MODE	xENABLE	xPHASE	xOUT1	xOUT2	FUNCTION (DC MOTOR)
1	0	X	L	L	Brake
1	1	1	L	H	Reverse
1	1	0	H	L	Forward

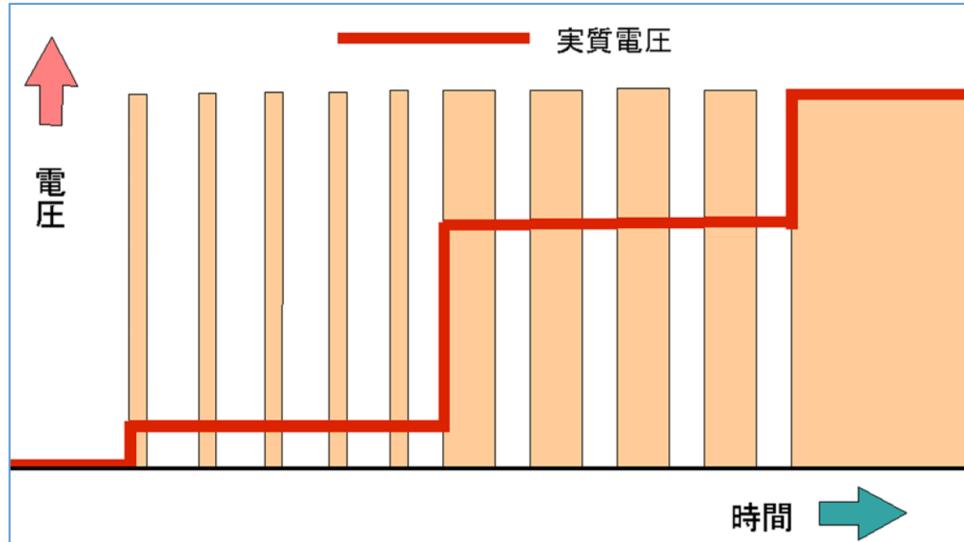
- Enableが1の時、モーターが回転
- Enableが0の時、モーターが停止
- Phaseが1の時は逆回転の向き
- Phaseが0の時は正転の向き

Enableが回転数、Phaseが回転方向

DRV8835 Dual Low-Voltage H-Bridge ICデータシート 2016年8月9日 より

# PWMの説明

- RumiCarでは走行速度制御にPWM技術を利用しています
- PWMとは”Pulse Width Modulation”の略語で電圧を短い間隔でオンとオフを繰り返すことによって見かけの電圧を変化させる技術です



# ArduinoのPWM

- ArduinoのPWMはタイマと連動しています
- Arduinoのタイマは3個あります

タイマ	ピン	周波数
タイマ0	5、 6	980Hz
タイマ1	9、 10	490Hz
タイマ2	3、 11	490Hz

- ArduinoのServoライブラリを使うとタイマ1は使えなくなってしまいます

Arudino版でのピンアサイン		
機能	タイマ	ピン
操舵	タイマ2	3、 11
走行	タイマ0	5、 6

# RumiCarでのPWM制御

- RumiCarのプログラムサンプルではピンにマクロ設定をしています
- 例えば右にハンドルを曲げたいときはAIN1をオン、AIN2を"0"に設定します
- ※この例はArduino版のピンアサインです。ESP版などはピンアサインが異なりますがマクロを使ってピンアサインの違いを意識することなくプログラムを開発できるようにしてあります

タイマ	機能	ピン	マクロ設定
タイマ2	右折	3	AIN1
	左折	11	AIN2
タイマ0	前進	5	BIN1
	後進	6	BIN2

# RumiCarでのRC関数

- RumiCarではプログラムを開発しやすいように下記関数を用意しました

動作	関数の型	関数名	第一引数 (int)	第二引数 (int)	動作	エラー時の戻り値
操舵	int	RC_steer	RIGHT	(無し)	右操舵	0
			LEFT		左操舵	
			CENTER		中央	
走行	int	RC_drive	FREE	0から255 (PWM値)	ニュートラル	0
			REVERSE		後進	
			FORWARD		前進	
			BRAKE		ブレーキ	

- RC\_driveのFREE時の第二引数の値は指定は必要ですが値自体は無視されます
- RC\_driveの第二引数の値はPWM値であり値が大きいほどモータに電力を掛けることができます
- 第一引数はint型です。この表の値を指定すると最終的にマクロで展開されます

# 関数を用意することの意義

- RumiCarではマクロ設定や独自関数としてRC関数を追加するなどにより下記のメリットを提供します
  - 複数の命令をまとめているのでプログラムがすっきりし可読性が向上する
  - 固定の引数の記載を省くことによりプログラムがすっきりし可読性が向上する
  - 不要な記載を省くことによるプログラムミス低減
  - ハードウェアの差異によるピンアサインや制御方法等の考慮の不要化
  - コンピュータモジュール間の差異を吸収しプログラムの共通化

# 共通化ストラクチャ

Arduino	
機能	ピン
右折	3
左折	11
前進	5
後進	6

ハードウェアの違いによるピンアサインと制御方法の差異吸収

ESP32	
機能	ピン
右折	4
左折	26
前進	27
後進	25

機能	マクロ
右折	AIN1
左折	AIN2
前進	BIN1
後進	BIN2

ESP32  
チャンネル番号に書換

VL53L0Xアドレス再設定

マクロ	ピン	
	Arduino	ESP32
SHUTO	14	19
SHUT1	15	18
SHUT2	16	5

共通関数化

RC
RC_steer
RC_drive

Arduino

analogWrite

ESP32

ledcSetup

ledcAttachPin

ledcWrite

命令の違い隠蔽

RC

RC\_analogWrite

# RC\_steer関数の利用例

- RC\_steer関数を利用した操舵プログラミングの例です
- 左側にオリジナル命令での記述を示します

オリジナル命令での記述

```
123  
124 void loop()  
125 {  
126     analogWrite(AIN1, 255);  
127     analogWrite(AIN2, 0);  
128     delay(500);  
129     analogWrite(AIN1, 0);  
130     analogWrite(AIN2, 255);  
131     delay(500);  
132 }
```



RC関数利用の記述

```
210  
211 void loop()  
212 {  
213     RC_steer(RIGHT);  
214     delay(500);  
215     RC_steer(LEFT);  
216     delay(500);  
217 }
```

# Exercise-2

モード制御

# Exercise-2.1

ハンドルを切る

# Exercise-2.1

## ハンドルを切る(準備)

- RumiCarの車体の電源をオフにします
- CMをRumiCarの車体より取り外してください
- CMにUSBケーブルを接続します
- USBケーブルをパソコンに接続します
- ファイル"Exercise-2.1"を開きます

# Exercise-2.1

## ハンドルを切る（プログラム）

```
210
211 void loop()
212 {
213     RC_steer(RIGHT);
214     delay(500);
215     RC_steer(LEFT);
216     delay(500);
217 }
```

- delay命令はそのまま何もしないで指定時間だけ待つよ
- delayの引数の単位はミリ秒だよ
- このプログラムはハンドルを切ったまま0.5秒(500ミリ秒)そのままハンドルを保持するよ

- ハンドルを右に切る
- 0.5秒そのまま
- ハンドルを左に切る
- 0.5秒そのまま

繰り返す

## Exercise-2.1 ハンドルを切る(試験)

- ・コンパイルしてプログラムをCMに書き込みます
- ・CMからUSBケーブルを取り外します
- ・CMをRumiCarの車体に取り付けます
- ・RumiCar車体の電源をオンにしてください
- ・ハンドルが左右に曲がっていますか？

# Exercise-2.2

速度制御

## Exercise-2.2 速度制御(準備)

- RumiCarの車体の電源を**オフ**にします
- CMをRumiCarの車体より取り外してください
- CMにUSBケーブルを接続します
- USBケーブルをパソコンに接続します
- ファイル"Exercise-2.2"を開きます

# Exercise-2.2

## 速度制御(プログラム)

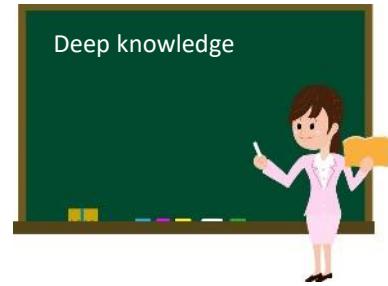
```
210
211 void loop()
212 {
213     RC_drive(FORWARD, 255);
214     delay(500);
215     RC_drive(FORWARD, 200);
216     delay(500);
217     RC_drive(FORWARD, 150);
218     delay(500);
219 }
```

- `RC_drive`の第二引数、255とか200とか150はPWMの値です
- 255が最大で128が最大値の約半分の値になります
- 値を下げてゆくとタイヤの回転もだんだんと遅くなります

## Exercise-2.2

### 速度制御(実走試験)

- CMからUSBケーブルを取り外します
- CMをRumiCarの車体に取り付けます
- まず、いきなり走らせないでRumiCarの車体を手で持ち上げたままにします
- RumiCar車体の電源をオンにしてください
- タイヤの回転が変化していますか？
- RumiCarをそっと机に置いてみましょう

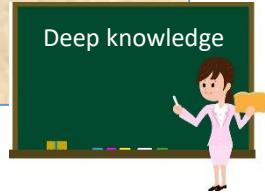


# Deep knowledge

直流モータの低回転制御

稻玉 繁樹 氏

# RumiCarのモータ制御



- RumiCarは2つのモータを制御しています。
- ステアリング

//操舵の関数

```
int RC_steer (int direc ) {
```

```
    RC_analogWrite(AIN1, 255);  
    RC_analogWrite(AIN2, 0);
```

- 走行

//走行の関数

```
int RC_drive(int direc, int ipwm) {
```

```
    RC_analogWrite(BIN1, 0);  
    RC_analogWrite(BIN2, ipwm);
```

オンオフ制御

全力でモータをオンしている

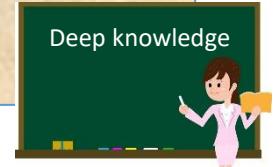
- ・角度が決められない
- ・電流が流れすぎている

PWM制御

電圧を変えている

- 車両条件などで速度安定しない
- 速度と電圧が比例しない

# RumiCarのモータ制御



- 低速で動かしたいが、走行指令 120/255 以下は動かない

□ 原因 1 : PWM周期が高すぎる

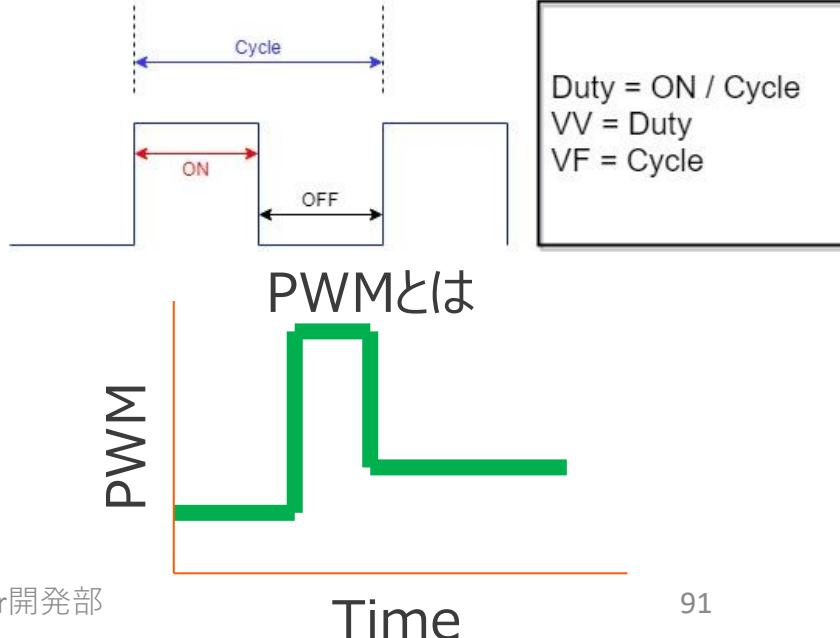
□ 原因 2 : DCモータは始動時に大きい電流が必要

$$Z=j\omega L \quad \cdots (1) \quad \omega=2\pi f \quad \cdots (2)$$

≒ 見かけの抵抗が高くなりPWM指令が低いと電流が小さく始動できない  
よって、使えるPWM範囲が狭くなっている。

□ 対策 1 : PWM周期を変更

□ 976kHz → 244Hz



□ 対策 2 : 始動ブーストを導入

□ 始動時の数10ms はPWM大きく

# Advanced ソフトの解説



## □ A2-3 低速で走らせるため

```
// TCCR0B |= B00000011;           // 64分周= 16M/ 64/256=976Hz #default
TCCR0B |= B00000100;           // 256分周= 16M/ 256/256=244Hz
```

### // 始動後に低速へ

```
RC_drive(FORWARD, 200);      RC_delay(20);
RC_drive(FORWARD, 64);        RC_delay(500);
```

## □ B2-3 低速動作をライブラリへ

```
// Private Variable
#define UP_SPEED      (200)          // StartUP speed
#define UP_TIME       (3)            // 30 [ms]
```

```
RC_drive(FORWARD, 40);      RC_delay(1000);
RC_drive(BRAKE, 255);       RC_delay(500);
RC_drive(REVERSE, 40);      RC_delay(1000);
RC_drive(BRAKE, 255);       RC_delay(500);
```

PWM周期を下げる

→ 低い値でも電流が流れる  
始動時強く、動き出したら弱く  
→ 40位まで動くようになりました

## □ B2-5 縦列駐車にチャレンジ

動画で確認！

PWMドライバを定時割込化  
始動時ブーストを自動化  
→ 意識せずに低速動作が可能

注：PWM周期変更によりDelay()の周期  
がずれるため、RC\_delay()を使うこと

# Exercise-2.3

前進と後進

## Exercise-2.3 前進と後進(準備)

- RumiCarの車体の電源をオフにします
- CMをRumiCarの車体より取り外してください
- CMにUSBケーブルを接続します
- USBケーブルをパソコンに接続します
- ファイル"Exercise-2.3"を開きます

# Exercise-2.3

## 前進と後進(プログラム)

```
210  
211 void loop()  
212 {  
213     RC_drive(FORWARD, 255);  
214     delay(500);  
215     RC_drive(REVERSE, 255);  
216     delay(500);  
217 }
```

- RC\_driveの第一引数がFORWARDを指定した時に前進
- 第一引数がREVERSEの時に後進

- 前進する
- 0.5秒そのまま（前進し続ける）
- 後進する
- 0.5秒そのまま（後進し続ける）

繰り返す

## Exercise-2.3

### 前進と後進(実走試験)

- ・コンパイルしプログラムをCMへ書きこみます
- ・CMからUSBケーブルを取り外します
- ・CMをRumiCarの車体に取り付けます
- ・まず、いきなり走らせないでRumiCarの車体を手で持ち上げたままにします
- ・RumiCar車体の電源をオンにしてください
- ・タイヤが前進と後進を繰り返してますか？
- ・RumiCarをそっと机に置いてみましょう

## Exercise-2.3

### 前進と後進(発展問題)

- （発展問題）なるべくスリップしないようにして前進と後進を繰り返すようにするにはどうしたら良いでしょうか？

# Exercise-2.4

ジグザグ走行

## Exercise-2.4 ジグザグ走行（準備）

- RumiCarの車体の電源をオフにします
- CMをRumiCarの車体より取り外してください
- CMにUSBケーブルを接続します
- USBケーブルをパソコンに接続します
- ファイル”Exercise-2.4”を開きます

# Exercise-2.4

## ジグザグ走行(プログラム)

```
210
211 void loop()
212 {
213     RC_drive(FORWARD, 255);
214     RC_steer(RIGHT);
215     delay(500);
216     RC_steer(LEFT);
217     delay(500);
218 }
```

- 前進する
- ハンドルを右に切る
- 0.5秒そのまま（右に進み続ける）
- ハンドルを左にきる
- 0.5秒そのまま（左に進み続ける）

繰り返す

## Exercise-2.4

### ジグザグ走行(実走試験)

- ・コンパイルしプログラムをCMへ書きこみます
- ・CMからUSBケーブルを取り外します
- ・CMをRumiCarの車体に取り付けます
- ・まず、いきなり走らせないでRumiCarの車体を手で持ち上げたままにします
- ・RumiCar車体の電源をオンにしてください
- ・RumiCarは予想通りの動きをしていますか？
- ・RumiCarをそっと机に置いてみましょう

# 自律走行基礎

センサによる車体制御

# 自律走行基礎

- 自律走行はセンサ等で得られた情報や、既に自分が持っている情報など、またそれらを組み合わせて自動車を自動で最適に運転させることです
- この章では自律走行の基礎として簡単な例を考えてみます

# 自律走行基礎(状況の仮定)

- 次のような単純な自律走行の状況を仮定します
- 前進走行中に前方に障害物を検知したら障害物に衝突しないように安全に停車する
- 障害物を検知して停車中に障害物がなくなったら再度前進を再開する

# Exercise-3

自律走行プログラムを開発しよう！

# Exercise-3.1

安全に停止する車

# Exercise-3.1

## 安全に停止する車

- 単純な条件として具体的に下記を仮定します
  - RumiCarの中央センサ(Sensor1)で障害物を検知しながら、障害物が検知されない場合は前進します
  - 障害物の例えば10cm手前で停車、障害物がなくなったら再度前進再開します

# Exercise-3.1

## RumiCarでの開発にあたっての考慮事項

- 距離の測定誤差
  - サンプルプログラムのVL53L0X設定である高速測定モードでは高速に測定するため若干精度が犠牲になります。そのため測距時に1~2cmの誤差が発生します。余裕を見込んだ距離判定をしましょう
- 測定不能時の処理
  - 測定結果で8190前後の値はレーザーの反射が得られない場合の値です。つまりは測定範囲内に障害物を検知できなかった場合の値になります

## Exercise-3.1

### RumiCarでの開発にあたっての考慮事項(続き)

- 地面等の検知

- センサのレーザーは円錐形に照射されます。

RumiCarは小さいので車上のセンサ搭載位置が低く、円錐形に照射されたレーザーはいずれ距離測定限界より先に地面や床等を捉えてしまいます。この距離はRumiCarモデル32型でだいたい30cm前後です。これらより障害物としての検知距離は30cm以下としてください

- 急停止処理

- 走行状態から停止や後進命令を出しても慣性（勢い）が付いているため直ぐには停止できません。できれば上手な減速や停止の方法を考慮しましょう



# Exercise-3.1

## プログラミング上の条件まとめ

項目	条件
使用センサ	中央センサ(Sensor1)のみ使用する
挙動	設定距離内に障害物を検知したら停止する
設定距離	10cmとする。ただし誤差を考え実範囲は $10 \pm 2\text{cm}$ とする
信頼距離	30cm以内とする
測定不能時の値	8190、但し誤差を含むことがあるので注意

# Exercise-3.1

## 開発時のちょっと一工夫

- 判断条件を単純に障害物がない、とのロジックとしてしまうと前方に障害物がない場合はRumiCarがどこまでも走っていってしまいます。これだとうっかり机の上などにRumiCarをおいてしまうとRumiCarが机から飛び降りてしまします。机の上で動作テストなどをする可能性がある場合においては安全も考えて例えば**30cm以内に障害物がある場合のみ走る**などと条件を追加するなど工夫すると良いでしょう。開発時にRumiCarをうっかり破損させてしまう危険性を低減できます

# Exercise-3.1

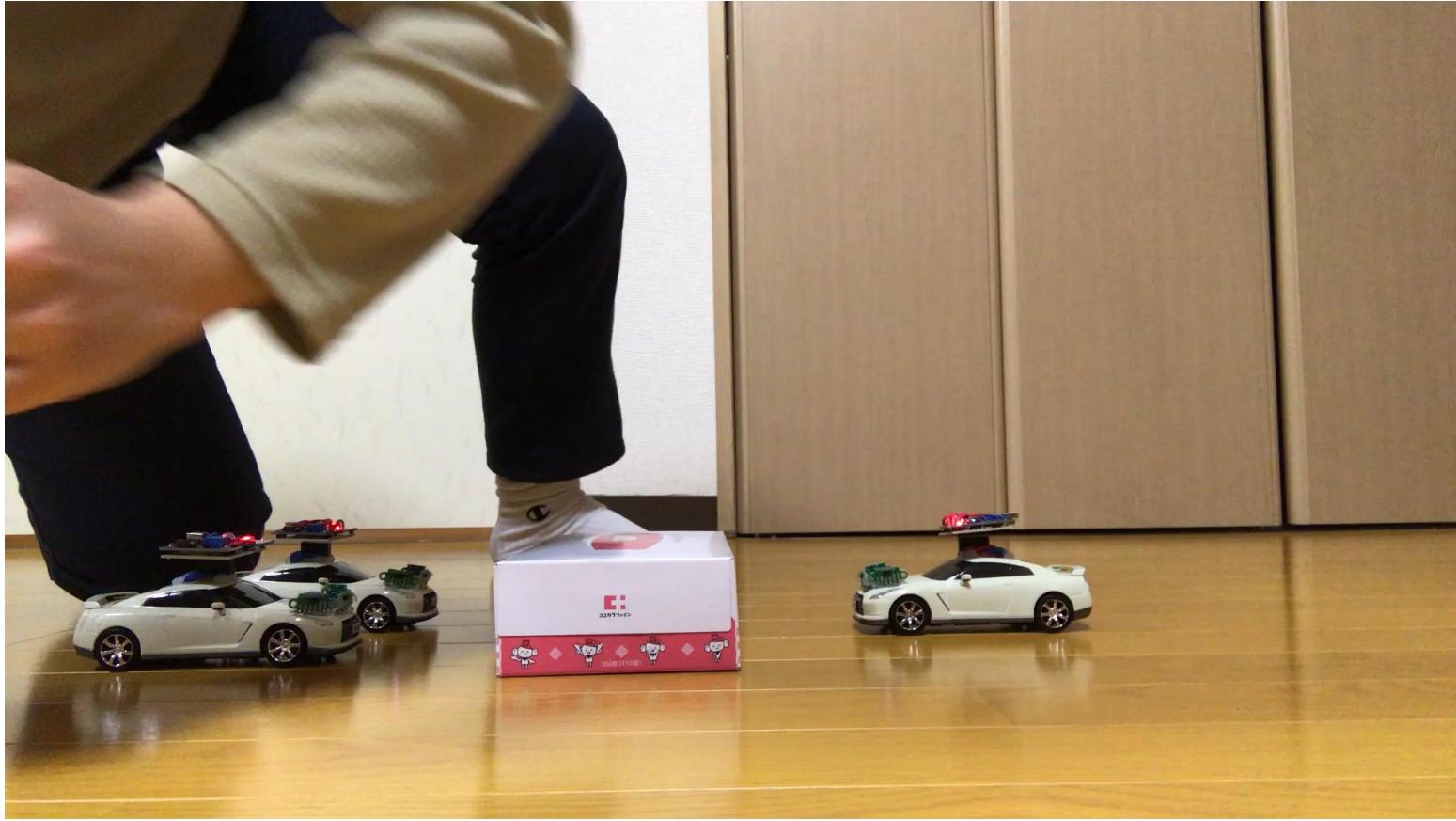
## 安全に停止する車（動画）

[動画のYouTubeリンク](#)



# Exercise-3.1 動画のYouTubeリンク

## 安全に停止する車（動画）



# Exercise-3.1

## 安全に停止する車(準備)

- RumiCarの車体の電源をオフにします
- CMをRumiCarの車体より取り外してください
- CMにUSBケーブルを接続します
- USBケーブルをパソコンに接続します
- ファイル"Exercise-3.1"を開きます

# Exercise-3.1

## 安全に停止する車(プログラム)

```
210
211 void loop()
212 {
213 int ispeed = 255;
214 int idist1;
215 idist1=sensor1.readRangeSingleMillimeters();
216 if ( idist1 < 300 ){
217   if ( idist1 > 120 ){
218     RC_drive(FORWARD,ispeed);
219   }else if (idist1 < 80){
220     RC_drive(REVERSE,ispeed);
221   }else{
222     RC_drive(BRAKE,ispeed);
223   }
224 }else{
225   RC_drive(FREE,ispeed);
226 }
227 }
```

- 障害物が30cm以内にあるか？
- (無い場合は何もしない)
- (ある場合は次を実行)
  - 障害物が12cmを超える->前進
  - 障害物が8cm未満->後進
  - 8cm以上12cm以下->ブレーキ

繰り返す

# Exercise-3.1

## 安全に停止する車(実走試験)

- ・コンパイルしプログラムをCMへ書きこみます
  - ・CMからUSBケーブルを取り外します
  - ・CMをRumiCarの車体に取り付けます
  - ・RumiCar車体の電源をオンにしてください
  - ・手を中心センサーに近づけたり話したりしてみましょう。RumiCarは予想通りの動きをしていきますか？
- 
- ・RumiCarをそっと机に置いてみましょう

# Exercise-3.2

市街地を走る車

## Exercise-3.2

### 市街地を走る車

- 単純な条件として具体的に下記を仮定します
  - 市街地に見立てた、道路の両側に壁がある道路を走ります
  - 両側の壁の距離を測って道路の中心を走ります
  - 先行車の例えは10cm手前で停車、先行車がなくなったら再度前進再開します

## Exercise-3.2 [動画のYouTubeリンク](#)

### 市街地を走行する車（動画）



## Exercise-3.2 市街地を走行する車(準備)

- RumiCarの車体の電源をオフにします
- CMをRumiCarの車体より取り外してください
- CMにUSBケーブルを接続します
- USBケーブルをパソコンに接続します
- ファイル"Exrcise-3.2"を開きます

# Exercise-3.2

## 市街地を走行する車(プログラム)

```
210  
211 void loop()  
212 {  
213  
214     int ibound =250;  
215     int s0, s1, s2;  
216     s0=sensor0.readRangeSingleMillimeters();  
217     s1=sensor1.readRangeSingleMillimeters();  
218     s2=sensor2.readRangeSingleMillimeters();  
219  
220     if(s1<100){  
221         RC_drive(REVERSE,150);  
222     }else if (s1<150){  
223         RC_drive(FORWARD,150);  
224     }else if (s1<250){  
225         RC_drive(FORWARD,200);  
226     }else{  
227         RC_drive(FORWARD,255);  
228     }  
229     if(s0>s2){  
230         RC_steer(LEFT);  
231     }else{  
232         RC_steer(RIGHT);  
233     }  
234 }
```

- 速度のパラメータ（150や200、255）を変更してみよう！
- 距離のパラメータ（100や150、250）を変更してみよう！
- どうなるかな？？

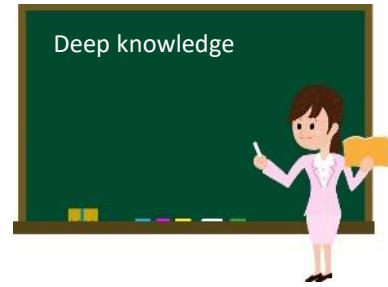
- 障害物が10cm未満->ゆっくり後進
- 障害物が15cm未満->ゆっくり前進
- 障害物が25cm未満->ちょっと速めに前進
- 上記以外は全速前進

- 左右の壁の距離を比較して
- 壁から近い方から離れる（壁から遠い方へハンドルを切る）

## Exercise-3.2

# 市街地を走行する車(実走試験)

- ・コンパイルしプログラムをCMへ書きこみます
- ・CMからUSBケーブルを取り外します
- ・CMをRumiCarの車体に取り付けます
- ・RumiCar車体の電源をオンにしてください
- ・RumiCarをそっとコースに置いてみましょう



# Deep knowledge

ラズパイサンプルプログラムについて

石垣 翔子 氏

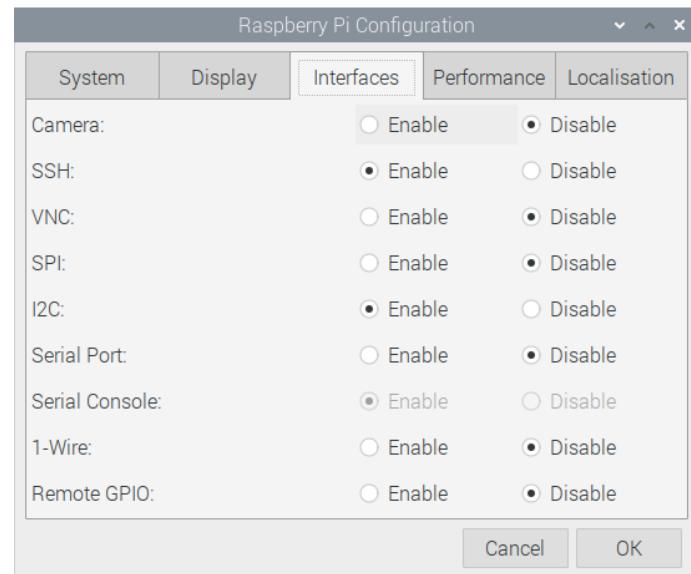
# ラズパイサンプルプログラムについて



- PWM制御を含め全てのGPIO制御に  
[pigpioライブラリ](#)を使用しています。  
プログラム実行前にpigpiodを起動してください。

起動コマンド : sudo pigpiod

- 測距モジュールはI2Cを使用するため、  
事前にI2Cを有効にしてください。



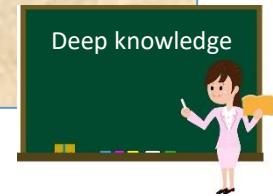
# ラズパイサンプルプログラムについて



- ・ ソフトウェアPWMとハードウェアPWM
  - ・ ラズパイで独立して制御できるハードウェアPWMは2つ  
12/32番ピンと33/35番ピンがHardware PWMとして使用可能
  - ・ ソフトウェアPWMは全てのGPIOピンで使用できます
  - ・ 急停止など正確な制御を要する走行用モータにハードウェアPWMを使用します

# ラズパイサンプルプログラムについて

Deep knowledge



## VL53L0Xのプログラム実行に必要なライブラリのインストール方法

- GitHubに公開されているライブラリを使用します
- git clone [https://github.com/johnbryanmoore/VL53L0X\\_rasp\\_python](https://github.com/johnbryanmoore/VL53L0X_rasp_python)
  - cloneしたVL53L0X\_rasp\_pythonフォルダ内でmakeを実行

### 注意点

- サンプルプログラムは上記レポジトリが/home/pi/Documentsにcloneされることを前提としています。必要に応じてVL53L0X.py内の共通ライブラリのパスを変更してください。
- VL53L0XのVccにラズパイの3.3V出力以外を使用する場合はラズパイのGNDピンをVL53L0Xと共にGNDに接続してください

# 付録

# I2Cアドレス

- RumicarではI2C通信を利用します。現在使用中のアドレスです

センサ	アドレス		設定方法
VL53L0X (レーザ測距モジュール)	Sensor0	左	0x20
	Sensor1	中央	0x21
	Sensor2	右	0x22
	Sensor3	後部	0x52
BMX055 (9軸センサ)	加速度センサ		CMの後部に設置 ソフトウェアにより変更不可
	ジャイロセンサ		基板上にてジャンパ設定
	磁気センサ		0x19
	0x69		0x13

# イベント履歴

日付	イベント	開催場所	備考
2017年8月5-6日	MakerFaireTokyo2017	東京ビックサイト	
2018年5月18-20日	MakerFaireBayArea2018	San Mateo, USA	
2018年8月4-5日	MakerfaireTokyo2018	東京ビックサイト	
2019年3月15日	Hello Tomorrow Global Challenge	Paris, France	投資家イベント
2019年8月3-4日	MakerFiareTokyo2019	東京ビックサイト	
2019年11月30日	IoT ALGYANつくるよ 2 !	情報科学専門学校	
2020年4月18日	IoT ALGYAN RumiCarハンズオン中継	オンライン	
2020年4月25日	IoT ALGYAN RumiCarハンズオン中継	オンライン	

# 最後に

RumiCar関連情報

# RumiCar関連情報

- RumiCar開発部Webページ
  - <https://www.rumicar.com/>
- Facebook RumiCarグループ
  - <https://www.facebook.com/groups/rumicar>
- RumiCar開発部 連絡先メールアドレス
  - [info@RumiCar.com](mailto:info@RumiCar.com)
- ALGYAN（あるじゃん）主催公式イベント
  - <https://algyan.connpass.com>



RumiCarハンズオン！に  
ご参加頂きましてどうも  
ありがとうございました

これで終了です

リアルイベント  
お楽しみに！！