

AIミニ四駆 ハッカソン 第一回

2015 08/16

株式会社アールティ

高橋良太

注意

- 23ページの内容が間違ってたので修正しました 8/20
- 修正箇所 上位byteと下位byteが逆

AICHIPのシステム構成

UserInterfaceへのアクセス
モーター等へのアクセス

ロボットの各機能
へのアクセス

マイコン lpc1343
のファームウェア

デジタル入出力
タイマー

AD変換
PWM出力
etc....

周辺ペリフェラル

モーター
単三電池2本

モータードライバ(Hブリッジ)

LED x 2

タクトスイッチ x 2

電源電圧監視回路

9軸センサ(ジャイロ, 加速度, 地磁気)

ミニ四駆
ハード(回路, メカ)

組み込み機器としてはこれで完結

AICHIPのシステム構成

Bluetooth越しにアクセス可能

UserInterfaceへのアクセス
モーター等へのアクセス

ロボットの各機能
へのアクセス

マイコン lpc1343
のファームウェア

デジタル入出力
タイマー

AD変換
PWM出力
etc....

周辺ペリフェラル

モーター
単三電池2本

モータードライバ(Hブリッジ)

LED x 2

タクトスイッチ x 2

電源電圧監視回路

9軸センサ(ジャイロ, 加速度, 地磁気)

ミニ四駆
ハード(回路, メカ)

組み込み機器としてはこれで完結

AICHIPのシステム構成

アプリケーション

今回作る部分

Bluetooth越しにアクセス可能

UserInterfaceへのアクセス
モーター等へのアクセス

ロボットの各機能
へのアクセス

デジタル入出力
タイマー

AD変換
PWM出力
etc....

周辺ペリフェラル

マイコン lpc1343
のファームウェア

モーター
単三電池2本

モータードライバ(Hブリッジ)

LED x 2

タクトスイッチ x 2

電源電圧監視回路

9軸センサ(ジャイロ, 加速度, 地磁気)

ミニ四駆
ハード(回路, メカ)

組み込み機器としてはこれで完結

Bluetooth越しに何
ができるのか？

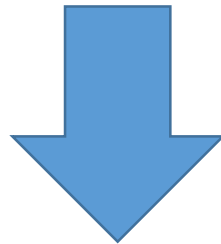
Bluetooth通信でできること!!

- モーターのコントロール
- LEDのコントロール

10msec周期

- 9軸センサの出力データの取得
- 電池電圧の取得

50msec周期



ミニ4駆の物理層にアクセスする最低限の仕組みを用意

PCとBluetooth接続した際のアプリの例

AICHIPへの送信コマンド

モーターにかけるdutyの変更
LED x 2 の点灯, 点滅
車体角度の書き換え

Processingで作ってみました

<https://www.youtube.com/watch?t=31&v=rk1TQI3hVoc>

PCとBluetooth接続してコマンドを送信している例



PCとBluetooth接続してデータをグラフに表示している例

AICHIPから送られてくるデータ

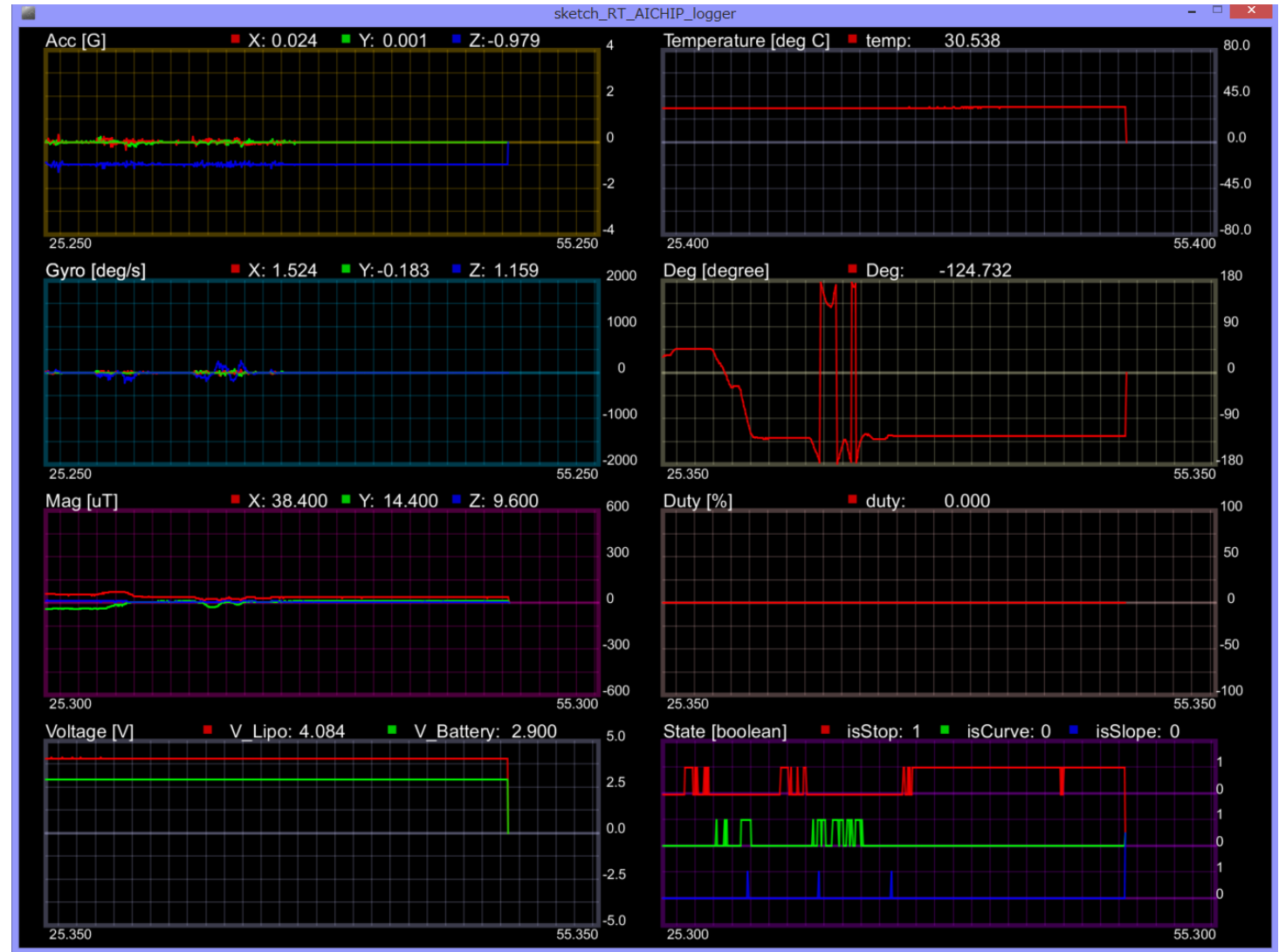
9軸センサの出力情報

車体角度

電源電圧の出力

リアルタイムに車体情報を表示する
データロガーを作ってみた

Processingで作ってみました



<https://www.youtube.com/watch?v=ixhc6ZnztPQ>

動画リンク

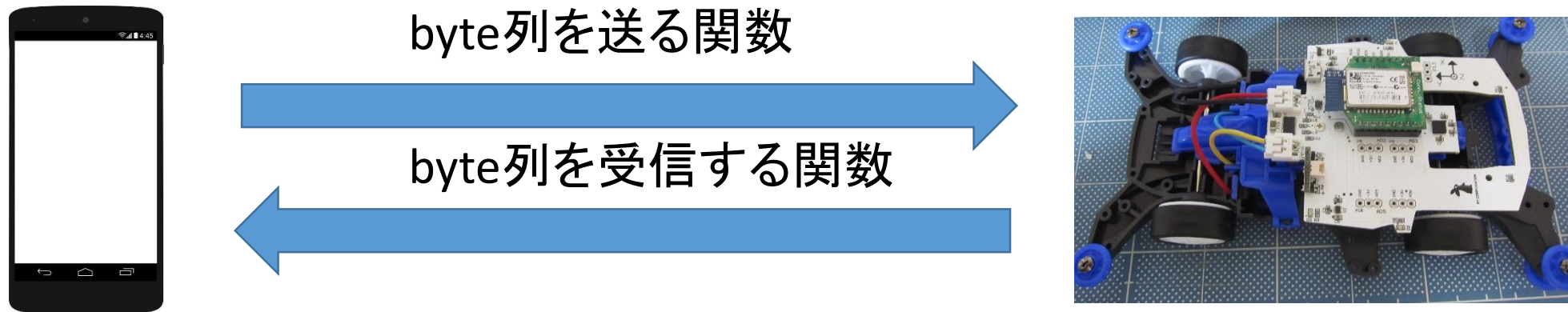
とりあえず, PCとのBluetooth通信で動くアプリは作ったが...

スマホとの通信で動くアプリはまだ....

みなさん, 協力してくれませんか

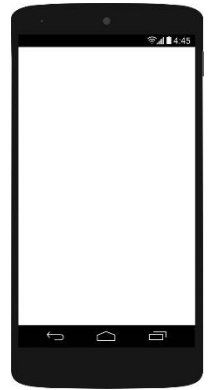
Bluetooth対Androidの通信

- SPP(Serial Port Profile)で通信
- byte型の配列を送受信 RS232Cと同様な感じ
- BluetoothChatというサンプルソースが下敷き
- BluetoothChat内のUUIDをSPP用に変更



byte列の送受信関数についてはBluetoothChatで提供されている

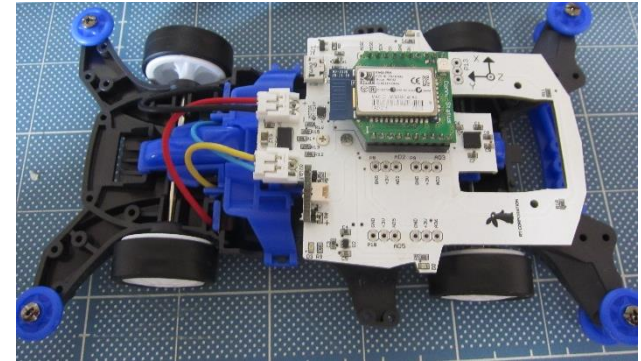
AICHIIPへの送信コマンド



byte列を送る関数



10msec毎に受信した文字列を確認



コマンドのフォーマットは決まっているので
フォーマット通りのbyte列を送信すればよい

定義されているコマンド一覧

- id0 モーターのdutyを設定
- id1 右(緑)LEDの制御コマンド
- id2 左(赤)LEDの制御コマンド
- id3 右(緑)LEDの点滅制御コマンド
- id4 左(赤)LEDの点滅制御コマンド
- id5 車体角度の指定コマンド

送信コマンドのプロトコル

- 長さ10byteで以下のフォーマット


0byte	1byte	2byte	3byte	4byte	5byte	6byte	7byte	8byte	9byte
header			id	datafield					
99	109	100	XX	XX	XX	XX	XX	XX	XX

固定 コマンド毎に異なる

- 意味のある10byteコマンドをAICHIP側が受信すると対応した動作を実行
- 送信コマンドはid 0,1, ... ,5の5種類
- BluetoothChatのbyte列送信関数を持ちいて10byteのコマンドを送る

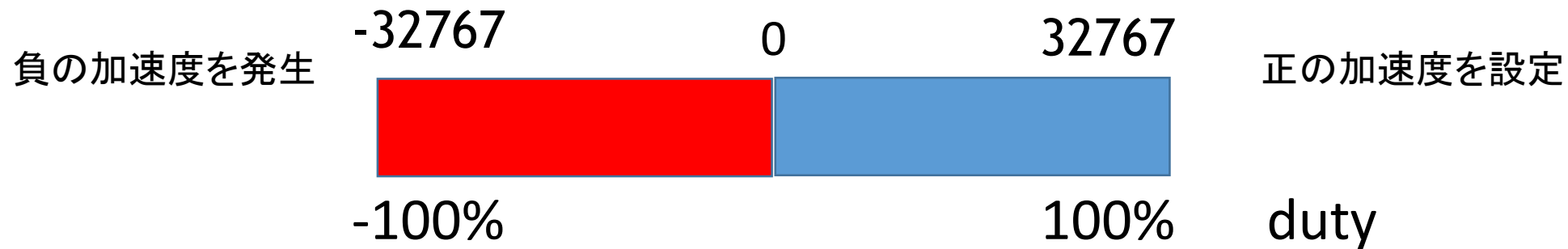
モーターduty設定コマンド

0byte	1byte	2byte	3byte	4byte	5byte	6byte	7byte	8byte	9byte
header			id	datafield					
99	109	100	0	XX	XX	0	0	0	0



16bit符号付整数 ←ここを与えたいdutyに応じて変更

アクセルの吹き出し具合



速度ではなく加速度を操作していることに注意

モーターduty設定コマンド(コード例)

```
/**
 * id 0: dutyの変更コマンド
 *
 * @param duty dutyを-1.0から1.0で指定 <br>
 *          負の値はモーターを逆の方向に回す
 * @return 10byteのコマンド配列
 */
byte[] command0(float duty) {
    byte[] command = new byte[10];
    int int_duty;
    int duty_L;
    int duty_H;

    int_duty = (int)(duty * 32767.0);
    if(int_duty < 0)
    {
        int_duty += 65535;
    }
    duty_L = int_duty & 0x000000ff ;
    duty_H = (int_duty & 0x0000ff00) >> 8;
```

```
//ヘッダー
command[0] = 99;
command[1] = 109;
command[2] = 100;
//id
command[3] = 0;
//値
command[4] = byte(duty_L);
command[5] = byte(duty_H);
//ダミー
command[6] = 0;
command[7] = 0;
command[8] = 0;
command[9] = 0;

return command;
```

```
}
```


LEDの点灯制御コマンド

- 右(緑)LEDの制御コマンド

0byte	1byte	2byte	3byte	4byte	5byte	6byte	7byte	8byte	9byte
header			id	datafield					
99	109	100	1	XX	0	0	0	0	0

- 左(赤)LEDの制御コマンド
- 1:LED点灯 0:LED消灯

0byte	1byte	2byte	3byte	4byte	5byte	6byte	7byte	8byte	9byte
header			id	datafield					
99	109	100	2	XX	0	0	0	0	0

1:LED点灯 0:LED消灯

LEDの点滅制御コマンド

- 右(緑)LEDの制御コマンド

0byte	1byte	2byte	3byte	4byte	5byte	6byte	7byte	8byte	9byte
header			id	datafield					
99	109	100	3	XX	XX	XX	XX	0	0

LED点灯時間 LED消灯時間

- 左(赤)LEDの制御コマンド

0byte	1byte	2byte	3byte	4byte	5byte	6byte	7byte	8byte	9byte
header			id	datafield					
99	109	100	4	XX	XX	XX	XX	0	0

LED点灯時間 LED消灯時間

LEDの点滅制御コマンド(コード例)

```
/**
 * id 4 左(赤)LEDの点滅制御コマンド
 *
 * @param 点滅時のon時間の指定 on_time[msec]
 * @param 点滅時のoff時間の指定 off_time[msec]
 * @return 10byteのコマンド配列
 */
byte[] command4(int on_time, int off_time) {
    byte[] command = new byte[10];
    int int_duty;
    int on_time_L;
    int on_time_H;
    int off_time_L;
    int off_time_H;

    on_time_L = on_time & 0x000000ff ;
    on_time_H = (on_time & 0x0000ff00)>>8;
```

```
//ヘッダー
command[0] = 99;
command[1] = 109;
command[2] = 100;
//id
command[3] = 4;
//値
command[4] = byte(on_time_L);
command[5] = byte(on_time_H);
command[6] = byte(off_time_L);
command[7] = byte(off_time_H);
//ダミー
command[8] = 0;
command[9] = 0;

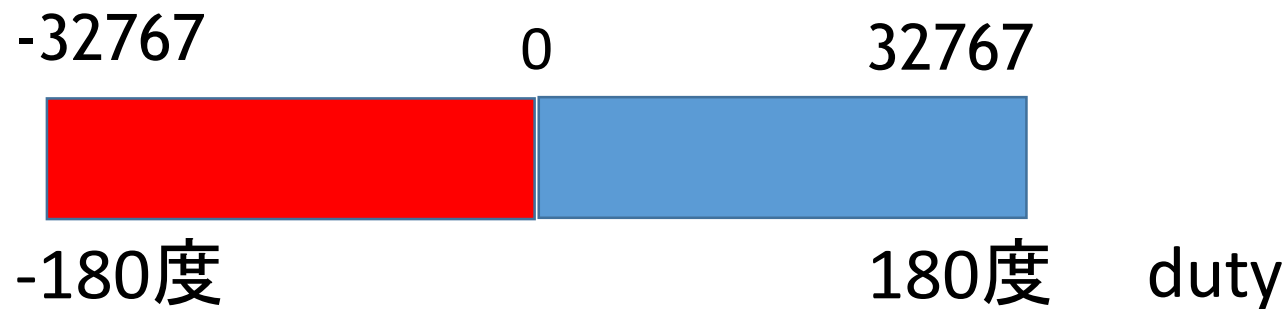
return command;
}
```

角度セットコマンド

- 右(緑)LEDの制御コマンド

0byte	1byte	2byte	3byte	4byte	5byte	6byte	7byte	8byte	9byte
header			id	datafield					
99	109	100	5	XX	XX	0	0	0	0

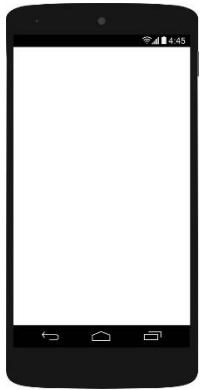
16bit符号付整数 ←ここをセットしたい角度に応じて変更



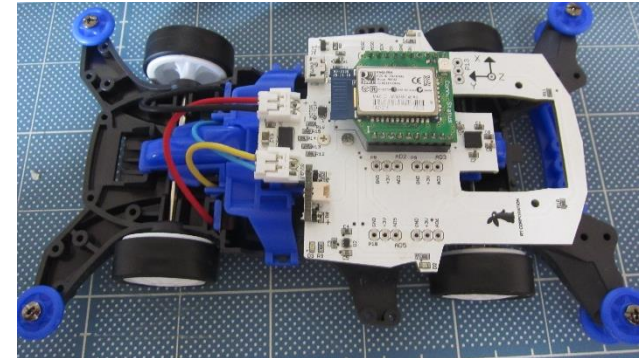
ジャイロセンサにより積算される角度のリセットに使用

AICHIPからのデータ受信

50msec毎に43byteのデータを送信



byte列を受信する関数



送られてくるbyte列を意味のあるデータに変換すればよい

受信データ一覧

- 加速度 x,y,z $\pm 16g$ 分解能 16bit
- 角速度 x,y,z $\pm 2000\text{deg/sec}$ 分解能 16bit
- 地磁気 x,y,z $\pm 1200\mu T$ 分解能:13ビット
- 温度
- 車体角度
- モーター用電池電圧
- マイコン用電池電圧
- モータのduty比
- 起動からの経過時間
- 車体の状況 (isStop, isCurve, isSlope)

受信データのプロトコル1

Byte	内容	Byte	内容
0	0xff	11	ACC Y 上位8bit (符号付)
1	0xff	12	ACC Z 下位8bit (符号付)
2	0x52	13	ACC Z 上位8bit (符号付)
3	0x54	14	TEMP 下位8bit (符号付)
4	0x34	15	TEMP 上位8bit (符号付)
5	0x57	16	GYRO X 下位8bit (符号付)
6	0x00	17	GYRO X 上位8bit (符号付)
7	タイムスタンプ	18	GYRO Y 下位8bit (符号付)
8	ACC X 下位8bit (符号付)	19	GYRO Y 上位8bit (符号付)
9	ACC X 上位8bit (符号付)	20	GYRO Z 下位8bit (符号付)
10	ACC Y 下位8bit (符号付)	21	GYRO Z 上位8bit (符号付)

受信データのプロトコル2


Byte	内容	Byte	内容
22	MAG X 下位8bit (符号付)	33	isCurve (符号なし)
23	MAG X 上位8bit (符号付)	34	isSlope (符号なし)
24	MAG Y 下位8bit (符号付)	35	経過時間 0byte (符号なし)
25	MAG Y 上位8bit (符号付)	36	経過時間 1byte (符号なし)
26	MAG Z 下位8bit (符号付)	37	経過時間 2byte (符号なし)
27	MAG Z 上位8bit (符号付)	38	経過時間 3byte (符号なし)
28	角度 下位8bit (符号付)	39	Lipo電圧 下位8bit (符号なし)
29	角度 上位8bit (符号付)	40	Lipo電圧 上位8bit (符号なし)
30	duty 下位8bit (符号付)	41	モーター電圧 下位8bit (符号なし)
31	duty 上位8bit (符号付)	42	モーター電圧 上位8bit (符号なし)
32	isStop (符号なし)		

受信データのプロトコル3

Byte	内容
0	0xff
1	0xff
2	0x52
3	0x54
4	0x34
5	0x57
6	0x00

受信データの
上位7byteは常に固定

Byte	内容
22	MAG X 下位8bit
23	MAG X 上位8bit



2byteにデータが分かれている
データを結合する必要がある

受信したbyte列を常に監視し続け上記の固定
パターンが出てきたらそこから36byte分が
有効データ

Byte列の結合

- 符号付と符号なしの2パターンで結合する時の例を示す.

```
int concatenate2Byte_int(int H_byte, int L_byte) {  
    int con;  
    con = L_byte + (H_byte<<8);  
    if (con > 32767) {  
        con -= 65536;  
    }  
    return con;  
}
```

符号あり

```
int concatenate2Byte_uint(int H_byte, int L_byte) {  
    int con;  
    con = L_byte + (H_byte<<8);  
    return con;  
}
```

符号なし

各データの物理量への変換式 1

加速度

加速度センサ値 : acc [16bit 符号付整数]

計算式 : $acc / 2048$ [g]

ジャイロセンサ

ジャイロセンサ値 : omega [16bit符号付整数]

ジャイロリファレンス値 : omega_ref [16bit符号付整数]

計算式: $(omega - omega_ref) / 16.4$ [deg/sec]

※ジャイロリファレンス値とはセンサが静止状態のときに出力される値

各データの物理量への変換式 2

地磁気センサ

地磁気センサ値 : mag [16bit符号付整数]

計算式 : $\text{mag} * 0.3$ [μT]

温度センサ

温度センサ値 : temp [16bit符号付整数]

計算式 : $\text{temp} / 340 + 35$ [$^{\circ}\text{C}$]

バッテリー電圧値

電圧値 : bat_v [16bit符号なし整数]

計算式 : $\text{bat_v} / / 13107$ [V]

各データの物理量への変換式 3

角度

角度 : deg [16bit符号付整数]

計算式 : $\text{deg} * 2 * \text{PI} / 32767.0$ [rad]

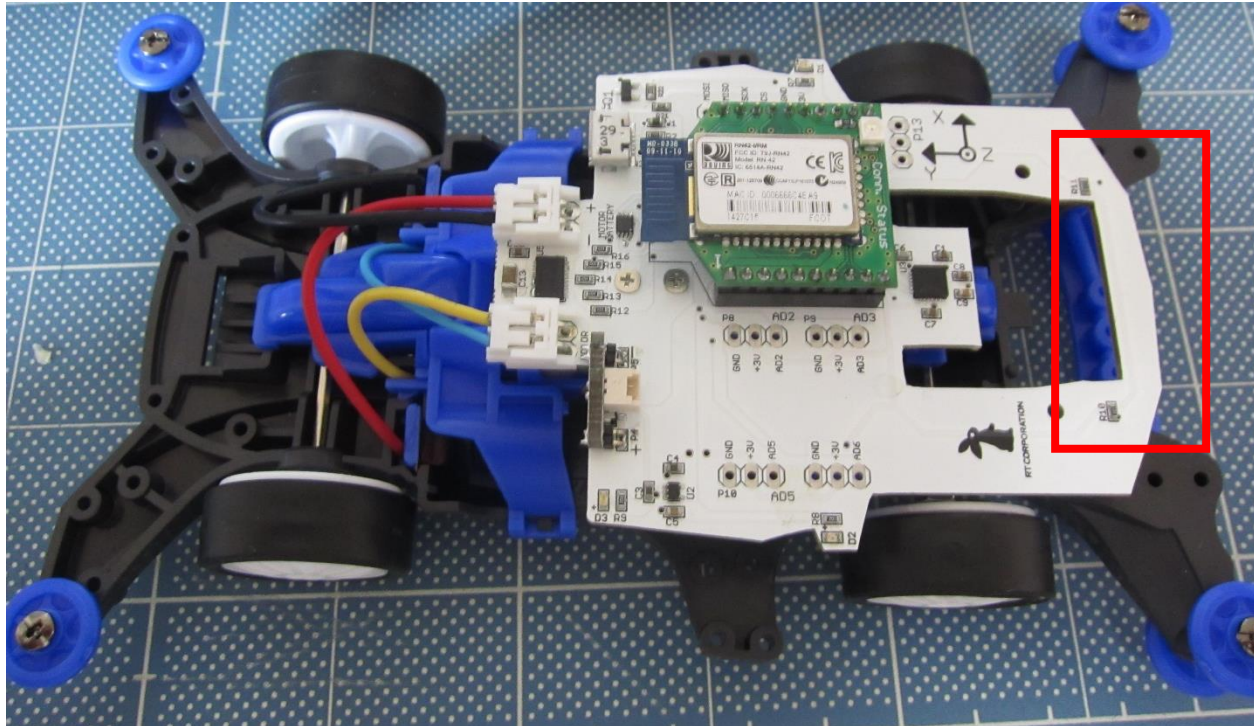
duty

duty値 : duty [16bit符号付整数]

計算式 : $\text{duty} / 32767 * 100$ [% (百分率)]

起動モードについて

Bluetooth経由でのデータ送受信のみを用いて
操作するモードにするには



左スイッチを押しながら起動

