



This text is CC 0 ( Public Domain )

# ESP CAR CONTROLLER ではじめる IoT 実験

Ver 1.3 ( 20160311 )  
編著 日本 Android の会秋葉原支部ロボット部

誰でも  
優しくない  
キット付

## セット内容

ESP CAR CONTROLLER (ESP8266 ベースボード)

LED 緑 赤 青

電解コンデンサ 100  $\mu$ F

コンデンサ 0.1  $\mu$ F(104と表記)

ピンヘッダ 6 ピン分+余分+ジャンパー

レギュレーター LD33CV

表面実装の抵抗 330  $\Omega$  x 3 10k  $\Omega$  x 5 +

USBシリアルモジュール

USBシリアルモジュール接続用ケーブル

## モーター実験セット(オプション)

モータードライバ(TC117HS) x 2

コンデンサ(0.1  $\mu$ F…104と記述) x 2

単3 x 2 乾電池ボックス

ミニギアモーターセット x 2 (ちっちゃいものくらぶ「[ちびギアモータープーリー・タイヤセット](#)」)

ミニギアモーター乾基板 x 2

ビニール線

ちびギアモーター x 2

LED チカチカ実験ができる!!



- 肝心の、ESP8266 Wi-Fi モジュールは、別途お買い求めください!!!!

発展応用例  
のリモコン  
ロボットカー!!



# Table of Contents

セット内容.....	1
モーター実験セット(オプション).....	1
ESP8266 とは?.....	3
ESP CAR CONTROLLER.....	4
回路図.....	4
ESP CAR CONTROLLER 組み立て方法(1) LED チカチカ実験のための組立.....	5
ESP CAR CONTROLLER 組み立て方法(2) モータコントロール実験のための組立.....	7
USBシリアルアダプタを使う.....	10
PCにドライバをインストール.....	10
ESP CAR CONTROLLERとの接続.....	10
コラム 付属品でないUSBシリアルアダプタを使う場合.....	11
動作確認をしましょう.....	11
ATコマンドを試してみましょう.....	11
プログラムしよう!!.....	13
開発のしかた.....	13
ESP8266のブートローダーモードとLEDの接続.....	13
LEDをチカチカしよう.....	13
WebからLEDをチカチカしよう.....	14
センサーをつなげる.....	16
モーターを動かす.....	18
電源を工夫する.....	19
Esp Car Gadgets の作例.....	20
その1.....	20
その2.....	20
ソフトウェア作例.....	21
コントロールアプリ.....	21
Esp Car Controller For Android.....	21
アプリ設定方法.....	21
タップしましょう.....	21
操作説明.....	22
Esp Car Controller For iOS.....	22
ESP8266側のモーターコントロールファームウェア.....	22
トラブルシューティング!!.....	23
動かなくなった!.....	23
Wi-Fiが接続できない.....	23
モーターコントロールが不安定.....	23
書き込みできない!.....	23
シリアルポートのデバイスネーム.....	23
ArduinoIDEでUSBシリアルアダプタの確認と設定をする.....	23
Macの場合の注意点.....	23
Linuxの場合の注意点.....	24
シリアル通信ソフト.....	24
Arduino IDEをシリアル通信ソフトとして使用する.....	24
こわしちゃったよ!!.....	24
もっと知りたい!!.....	24

# ESP8266 とは？



小さな Wi-Fi モジュールです。このモジュールを使うことで、既存のマイコンや電子工作などを Wi-Fi に接続することができます。このモジュール自体にもマイコン機能があり、これにスイッチやセンサーを繋げてデータを収集したり、ライトや動くものをつなげて Wi-Fi 経由で操作したりできます。このようなモジュールはこれがはじめて、というわけではないのですが、既存のものに比べて以下の特徴があります。

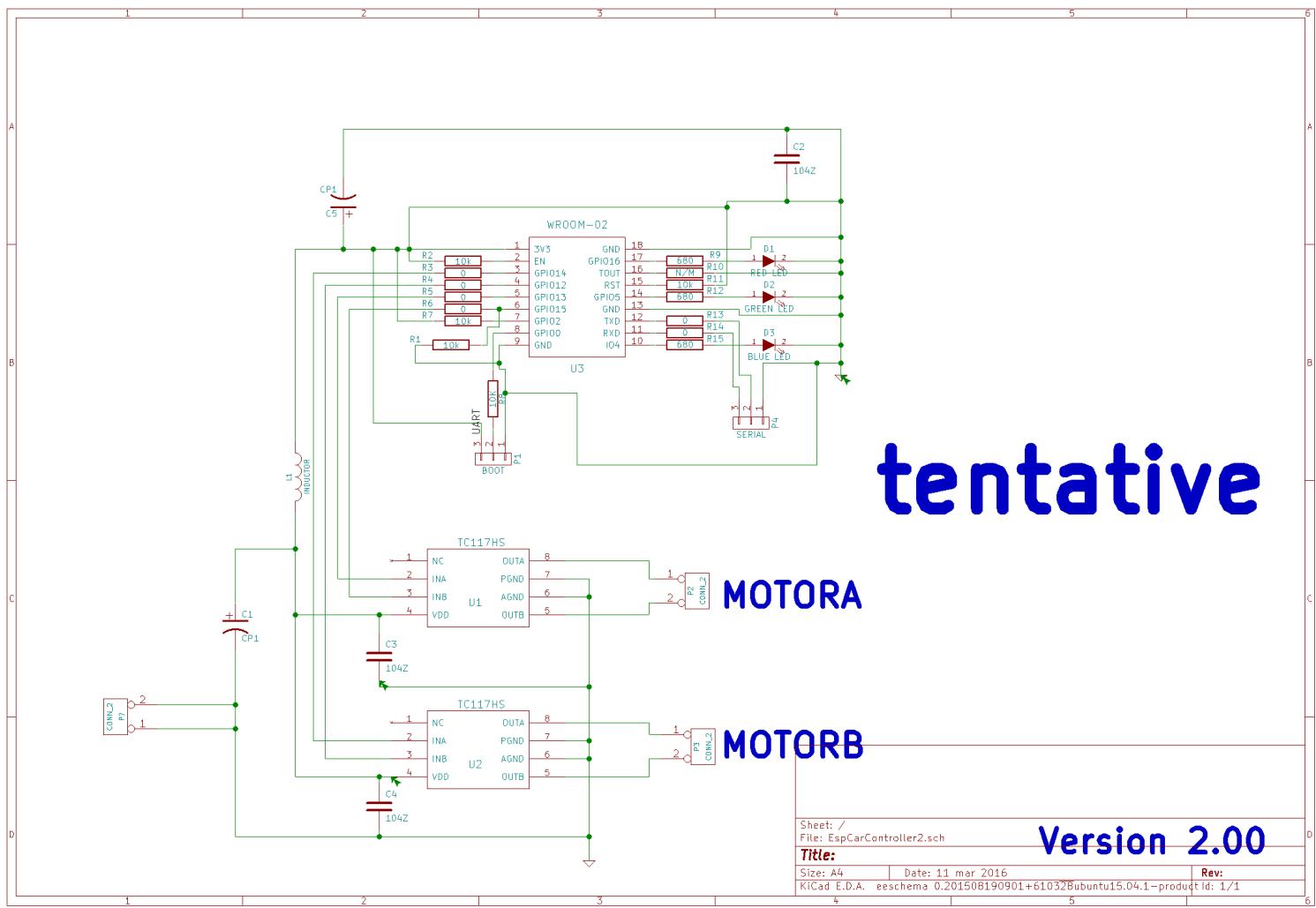
- 技適が通っている！
  - 今まで、海外製の同じようなモジュールで面白そうなものがあっても日本では使えませんでした。どうして？ 電波をある程度以上出力するものは、総務省の「技術基準適合証明」（技適）というものを持ってないと日本では使えません。
  - 技適の対象は、電波を出すものまるごとないといけないという建前があり、今まで Wi-Fi を使うものはノート PC まるごとだとかで証明を受けないといけませんでした。でもそれだと USB ドングル形式の Wi-Fi アダプタなどが使えなくなってしまいますよね。ということでモジュールで取得する道が少しずつ拓かれてきました。
  - ESP8266 は電源と通信の配線、簡単な通信方法だけで Wi-Fi が使えます。このような使いやすいモジュールが技適を通しておおっぴらに使えるということ、更に自分でモジュール自体にプログラムもでき、内部のプログラムを書き換えるても技適の適用内ということで、大きなエポックメーティングとなりました。
- 安い！
  - いくら使いやすいモジュールで、日本国内で使えても 1 万円だと、入手するのに 1000 個買わないといけないとかでは馬鹿らしいですよね。中国などでは性能はどうなんだか怪しいけれど、日本よりもはるかに安い値段で流通しているのに……。今回 ESP8266 は数百円で、それもパーツ販売店などで 1 つから購入できます!!!!
- さて、この期待の ESP8266 ですが、残念ながらコストの関係で今回のセットには ESP8266 は入ってません。別途お買い求めください！
  - 売っているところ
  - 秋月電子通商
  - スイッチサイエンス
  - Amazon
  - ほか



# ESP CAR CONTROLLER

- いくらESP8266だけでマイコン機能もあるといっても、いろいろつなげるにはLEDや電源、PCとの通信モジュールなどを配線していかなければなりません。自分でいちいち回路をつくるのはめんどくさいな…
- 今回そのためのベースボードをESP CAR CONTROLLERをロボット部メンバーのやましうさんさんが作成しました。
- ESP CAR CONTROLLERとは、ESP8266にモーター制御機能、LEDの表示機能、シリアルアダプタとの通信機能、プログラムなどをしやすくするためのコネクタなどを追加したものです。
- この小冊子には、ESP CAR CONTROLLER基板にLED実験ができるようにパーツをセットしています。
- 更にモーターコントロール用の追加パーツを使うことで、ロボットカーなどが作れるよ!

## 回路図

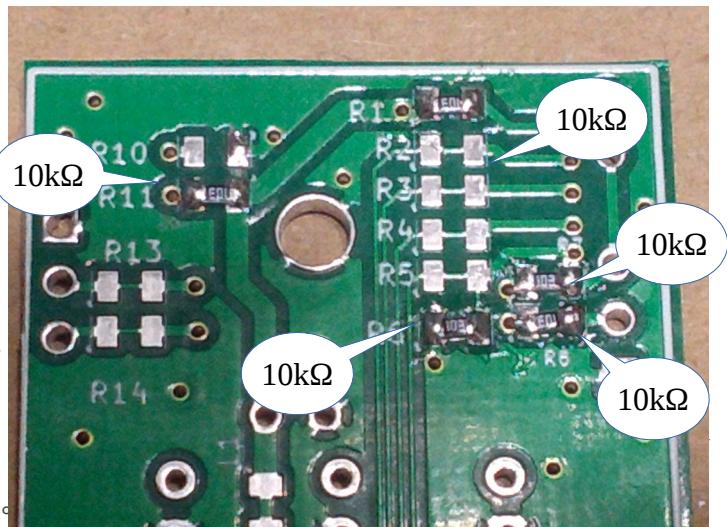


# ESP CAR CONTROLLER 組み立て方法(1) LED チカチカ実験のための組立

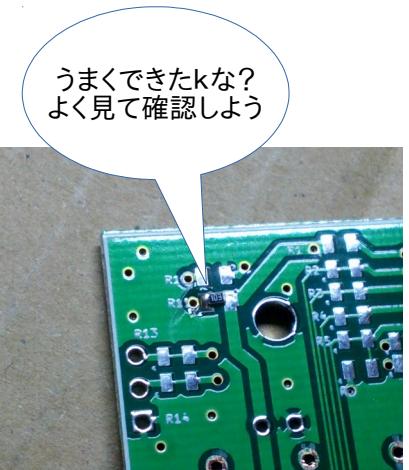
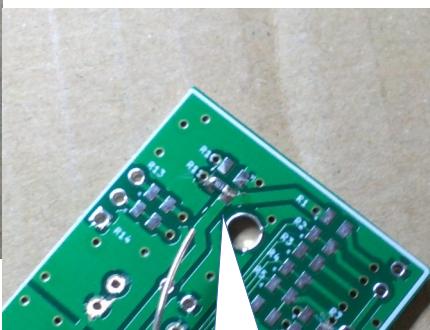
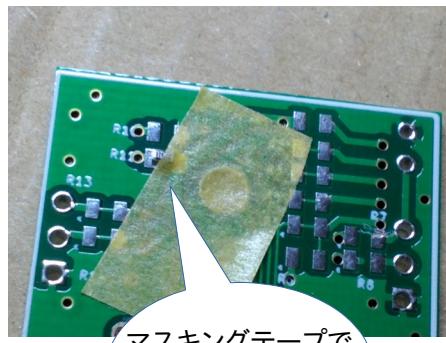
それでは、基板の組み立てをしていきます。

用意するもの：はんだごて、はんだ、ニッパー、マスキングテープなど

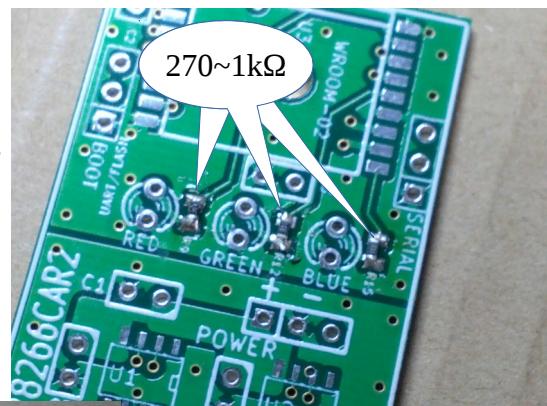
1.ウラ面のパッドに表面実装抵抗を5つ取り付けます。



抵抗の値は $10\text{k}\Omega$ です。表面実装のパーツの取り付け方は、マスキングテープで正しい位置に押さえ、端っこにはんだごてをあててはんだづけしていきます。

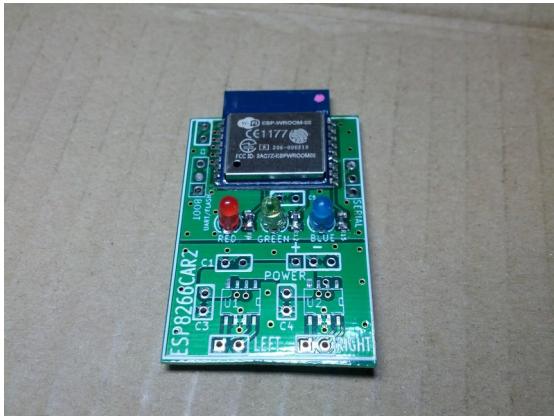


2.表面のLED用抵抗を3つ取り付けます。

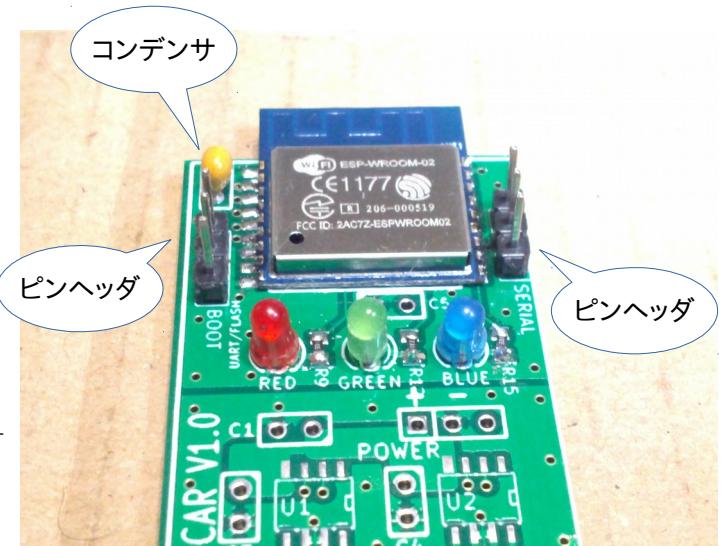


抵抗の値は $270\Omega$ ぐらいから $1\text{k}\Omega$ ぐらいの間で適当で。





4.LED を3つ取り付けます。足の長いほうが上側です。

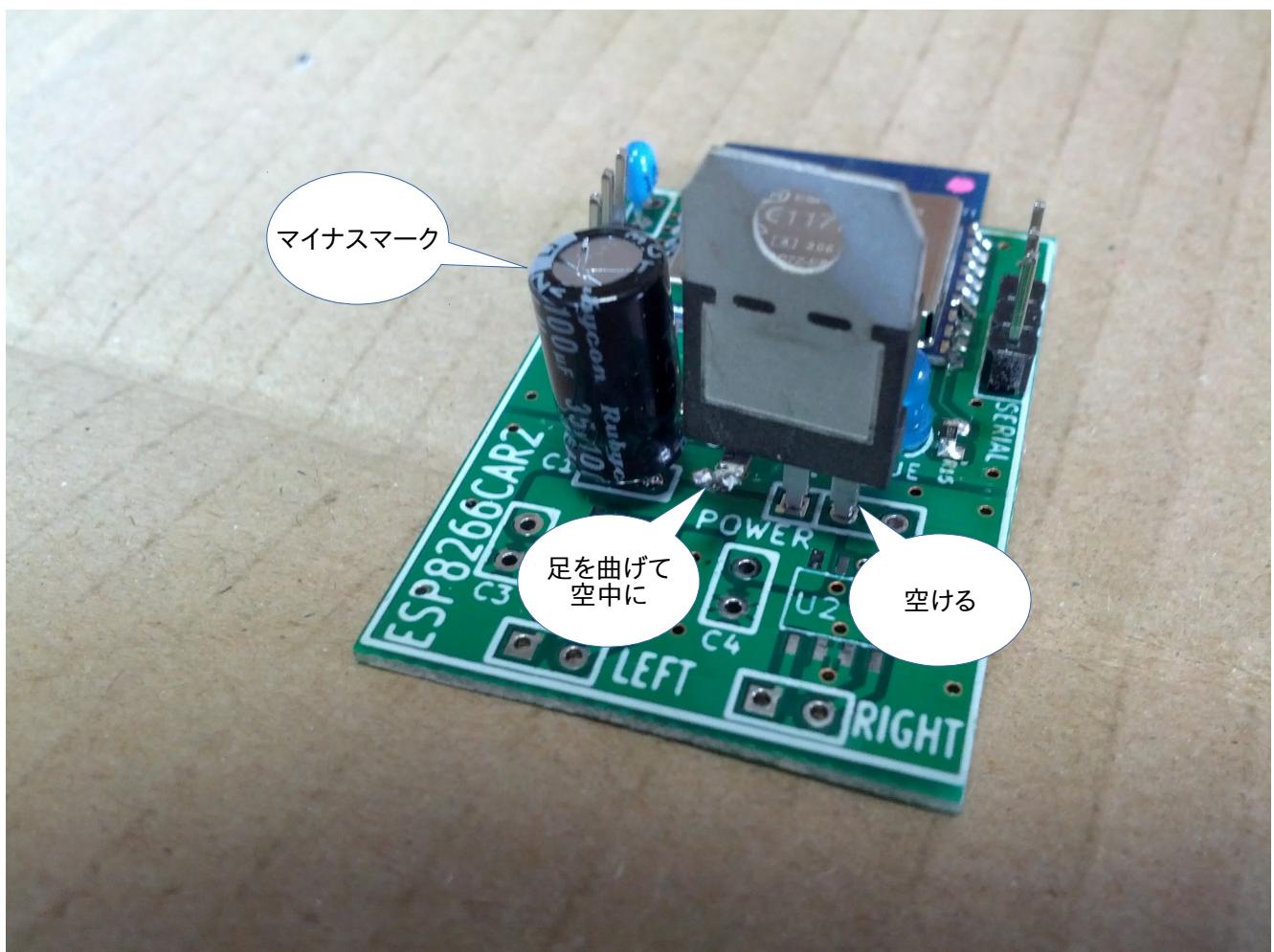


5.104のコンデンサ、およびピンヘッダを取り付けます。

7.コンデンサ、レギュレータを取り付けます。

なおモータドライバを購入された方は、コンデンサ、レギュレータを取り付ける前に「**ESP CAR CONTROLLER 組み立て方法(2)モータコントロール実験のための組立**」(後述)を行なってください。

コンデンサは白いマイナスマークが写真の左側です。レギュレーターは写真では見づらいですが、写真で一番右の足を手前に曲げて空中に上げています。POWERの一番右の穴は空けておいてください。



## ESP CAR CONTROLLER 組み立て方法(2) モータコントロール実験のための組立

モータードライバーなどを取付て、ロボットリモコンカーなどにチャレンジする場合の組み立て方法です。

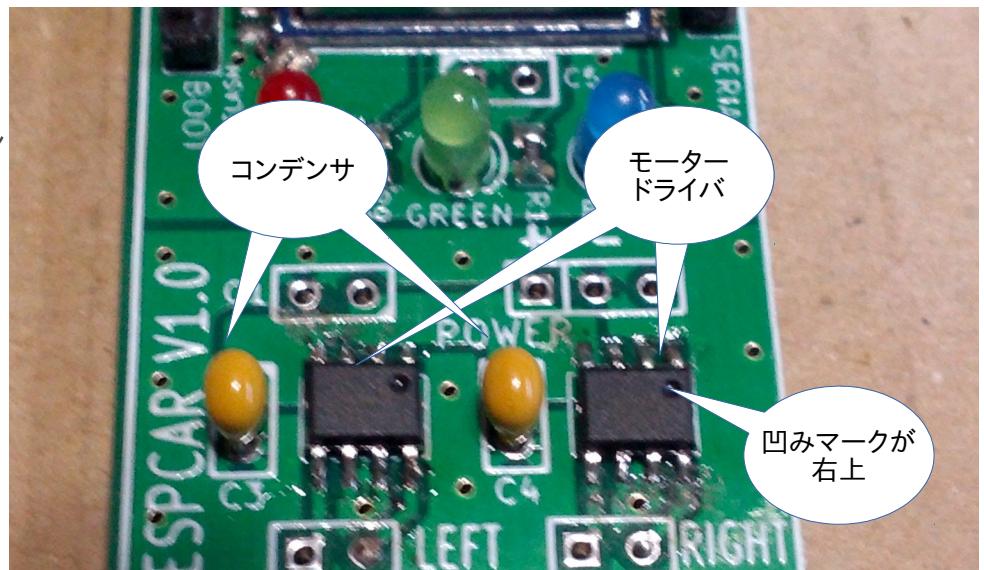
追加に必要なパーツです。

- モータードライバ(TC117HS) x 2
- コンデンサ(0.1μF…104と記述) x 2
- 単3x2 乾電池ボックス
- ちびギアモーターセット x 2 (ちっちゃいものくらぶ「[ちびギアモータ+プーリー・タイヤセット](#)」)
- ちびギアモーター専用基板 x 2
- ビニール線

このほか、電池および両面テープは別途ご用意ください。

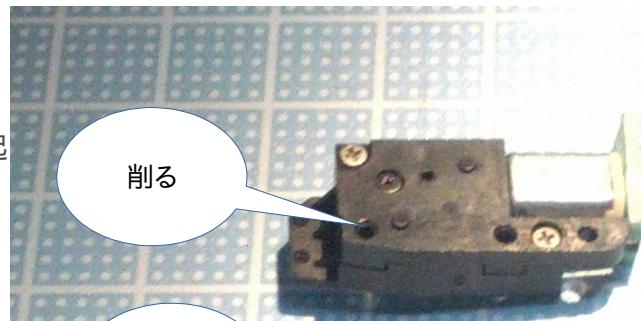


1.モータードライバを2つ取り付けます。



3.ちびギアモーターに基板を取り付け、はんだづけします。

4.ちびギアモーターの邪魔な突起をカッターなどで削ります。



5.ちびギアモーターにタイヤを取り付けます。



プーリー・タイヤ  
取り付けネジ1  
通常はこちらを使う

プーリー・タイヤ  
取り付けネジ2  
1がゆるくなった時に使う  
あまり強く締めないで!

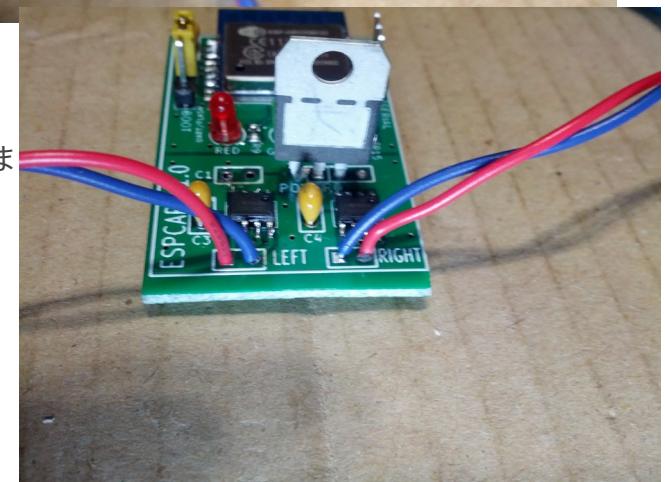
6.ちびギアモーターにリード線をはんだづけします。



7.ESP CAR CONTROLLER の他の部分を組み立てます。(参照:「ESP CAR CONTROLLER 組み立て方法(1) LED チカチカ実験のための組立」)

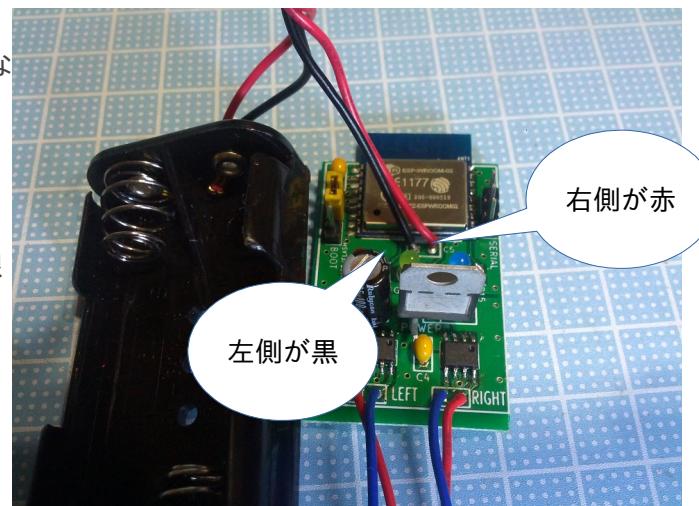
8.ここで適宜、「動作確認をしましょう」や「LED をチカチカしよう」を参照して通信や LED の点滅などがうまくいくことを確認してください。

9.ちびギアモーターと ESP CAR CONTROLLER を接続します。

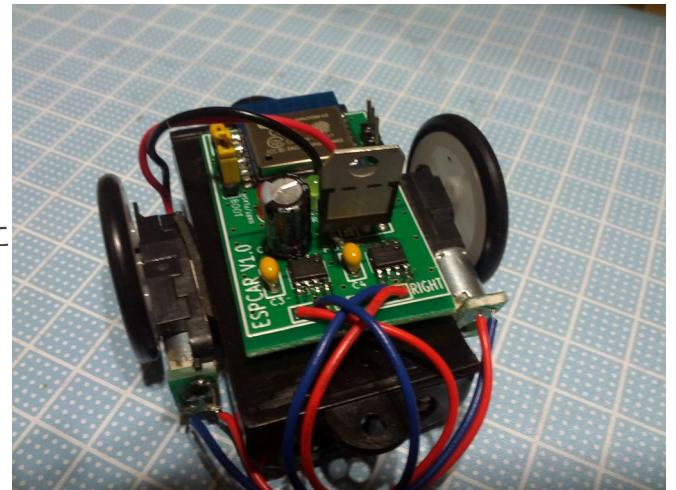


10. 適宜「モーターを動かす」のテストプログラムでモータが動くか確認します。(LEFT モーターは書き込みモード時にはちゃんと動かなくなります。書き込みが終わったらジャンパーを通常モードにして電源を入れなおしてテストしてください)

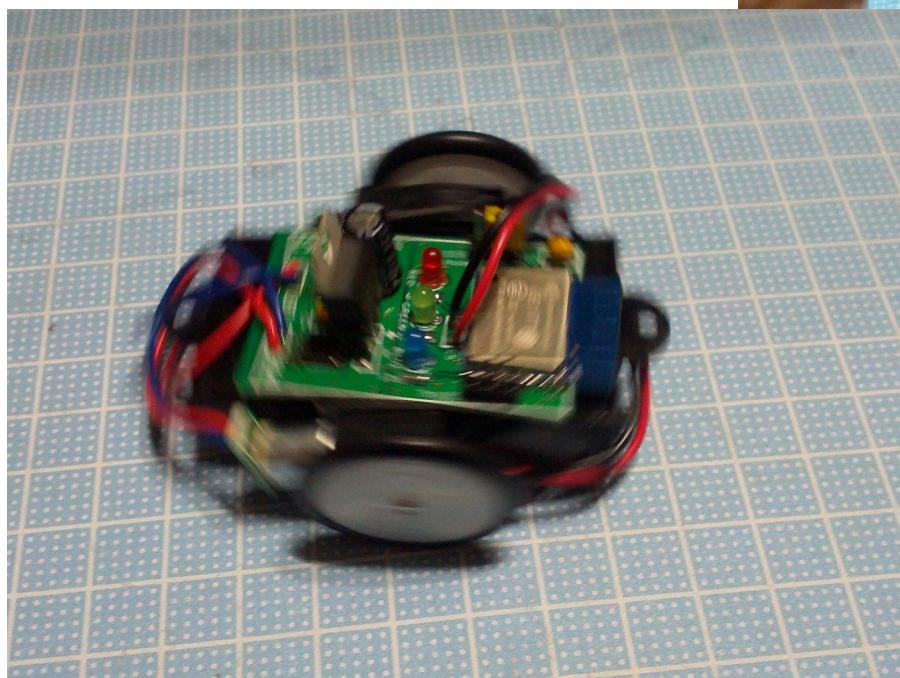
11. 電池ボックスをはんだづけします。基板中央の C5 の位置に取り付けます。後述の完成図を見て、リード線の長さは適宜調節してください。



12. 基板、ちびギアモーターを電池ボックスに両面テープで固定します。作例ではねじは使っていません。遊んでいるうちにふんずけたりなどしても、テープが外れることでギアモーター部が壊れないようにと考えています。必要に応じて、ネジ止めなど工夫してください。



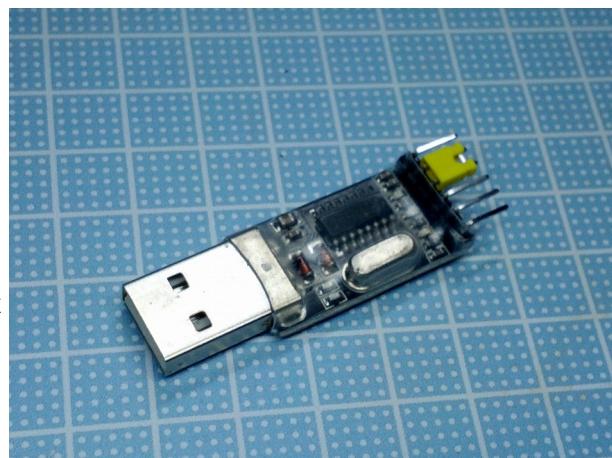
13. 単3電池を2本取り付けます。スイッチは電池を取り付／取り外すことで ON／OFF します。



14. 完成！遊びましょう！

# USBシリアルアダプタを使う

PCに付属のUSBシリアルアダプタを接続します。



## PCにドライバをインストール

MS-Windows、およびLinux(Ubuntu Linux 15.04 64bit版でチェックしました)の場合は接続するだけで自動でインストールされます。なお、MS-Windowsで認識しない場合はWindows Updateを試してみてください。

Mac OSX の場合は公式ページ [http://www.wch.cn/download/CH341SER\\_MAC\\_ZIP.html](http://www.wch.cn/download/CH341SER_MAC_ZIP.html) からドライバをダウンロードしてインストールします。10.8以降ではセキュリティとプライバシーの設定で、「すべてのアプリケーションを許可」にします。更に 10.10 以降ではターミナルから "sudo nvram boot-args=\"kext-dev-mode=1\" " と実行して再起動します。(野良署名付きドライバ: <http://blog.sengotta.net/signed-mac-os-driver-for-winchiphead-ch340-serial-bridge/> )

取り付けたシリアルポートのデバイスネームを確認してください。もしわからぬ場合は巻末トラブルシューティングの「シリアルポートのデバイスネーム」を参照してください。

## ESP CAR CONTROLLERとの接続

USBシリアルアダプタから、ESP CAR CONTROLLERへの接続をします。

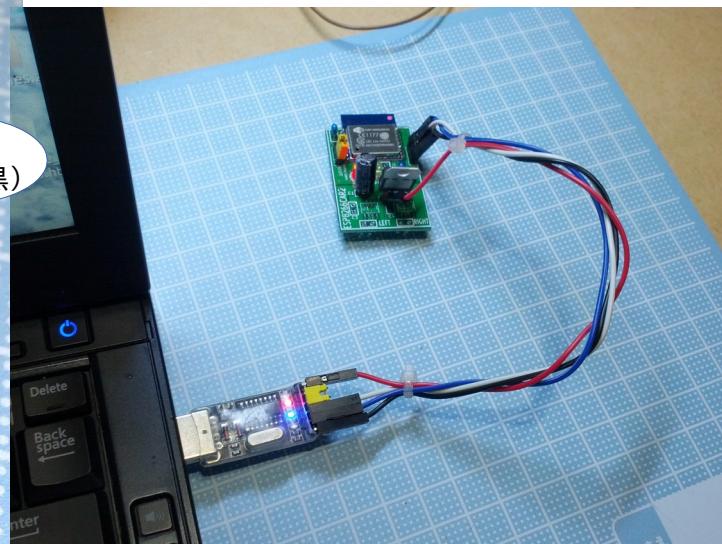
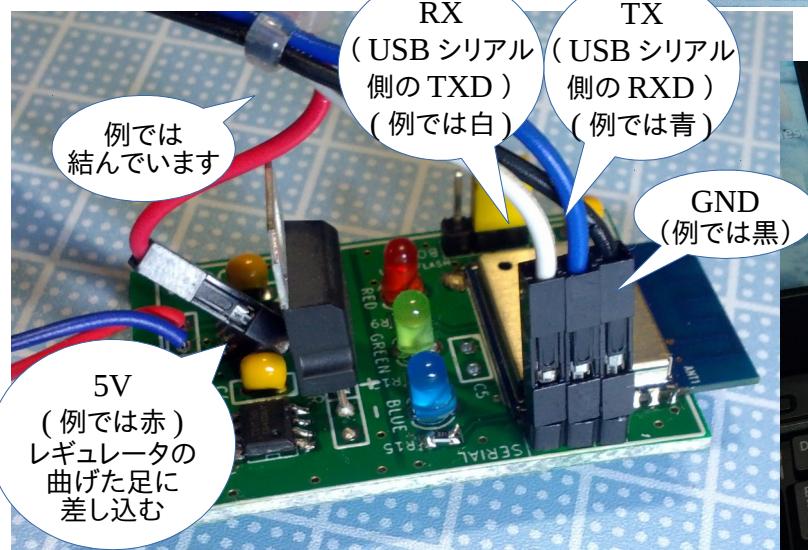
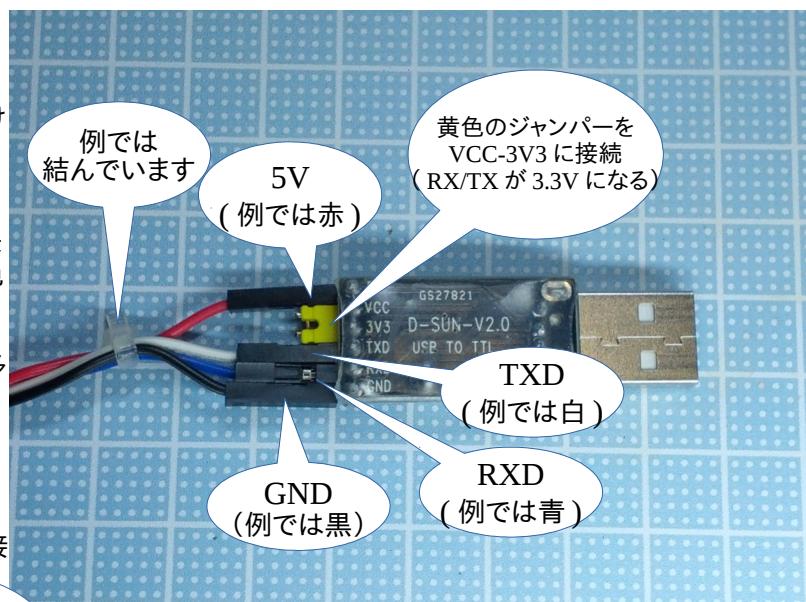
まず、USBシリアルアダプタにケーブルを取り付けます。

写真では赤黒青白のケーブルを使っていますがキットに付属しているケーブルは都合により色は決まっていません。適宜お持ちのケーブルに併せて色は変更してください。

(なお、慣例でGNDは黒、5Vは赤にすることが多いのもし付属のケーブルにこの色が含まれていたらなるべく合わせてください。)

す。

ESP CAR CONTROLLERには写真のように接続します。



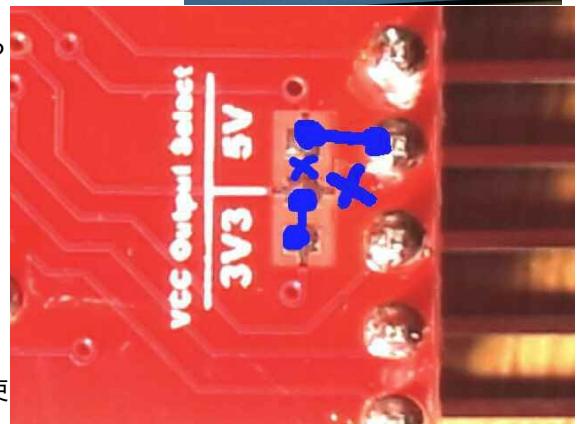
## コラム 付属品でない USBシリアルアダプタを使う場合

付属のUSBシリアルアダプタ以外のものも使うことができます。ESP CAR CONTROL基板で実験するためには以下の配線が必要になります。

- TX (3.3V TTLレベル)…TXD,TXOなどと記述されている場合もあります。
- RX (3.3V TTLレベル)…RXD,RXIなどと記述されている場合もあります。
- GND
- VCC (5V300mA以上の供給が必要もし、モーターなどを動かす場合は更に必要)

これに合ったものであればいいのですが、安価なシリアルアダプタではこのようなものを満たすものは見つけるのは難しいかもしれません。合ったものでなくとも手を加えることで使えるものに改造することができます。その場合のノウハウを、ちっちゃいものくらぶのCH340モジュールを例に説明します。

ちっちゃいものくらぶのCH340アダプタの場合、以下の2点の注意が必要です。



- ① そのままでは5Vで通信しますので、3.3Vレベルにする必要があります。
- ② ESP CAR CONTROLLERには5Vを供給するように、VCCから5Vが出るようにします。

そのためには、写真のように工作を行ないます。

- ウラ面の5Vのパッドがつながっている部分をカット
- ウラ面の3.3Vのパッドと隣のパッドを接続
- ウラ面の隣のパッドからVCCへの供給をカット
- ウラ面のVCCと5Vのパッドを接続

なおこのアダプタの場合は付属のモジュールと同じドライバが使えますが、チップの違うアダプタを使う場合はそれにあわせたドライバをPCにインストールしてお使いください。

## 動作確認をしましょう

シリアル通信ソフトで、115200bps、フロー制御なし、改行コードCR+LF、データ長8、parity無し、ストップピット1として設定して通信します。「AT」と入力して「OK」と出れば通信は無事できています。シリアル通信ソフトって何? わからない場合は巻末のトラブルシューティングを参照してください。

## ATコマンドを試してみましょう

ここでは、購入時に設定されているATコマンドでの簡単な操作を試してみます。

```
AT+CWMODE=3  
OK
```

でクライアントモードになります。

```
AT+RST  
OK
```

で一度リセットします。

```
AT+CWJAP="cnccafe", "freebeer"
```

で既存の Wi-Fi に接続します。cnccafe,freebeer は筆者の実験室の ESSID とパスワードです。ご使用の Wi-Fi 環境にあわせて変更してください。

```
WIFI CONNECTED  
WIFI GOT IP  
OK
```

上記のような返事が帰ってきたら OK です。

```
AT+CIFSR
```

で IP アドレスが表示されます。(例)

```
+CIFSR:APIP, "192.168.4.1"  
+CIFSR:APMAC, "1a:fe:34:ef:54:22"  
+CIFSR:STAIP, "192.168.11.8"  
+CIFSR:STAMAC, "18:fe:34:ef:54:22"  
OK
```

TCP/IP によるマルチセッション接続を有効にします。(サーバを有効にするのに必要です)

```
AT+CIPMUX=1  
OK
```

TCP/IP 8266 ポートでサーバを起動します。

```
AT+CIPSERVER=1, 8266  
OK
```

ここで、適当な PC から TELNET クライアントで 8266 に接続します。

```
0, CONNECT  
と出ます。
```

TELNET クライアントから hello と入力すると、

```
+IPD, 0, 7:hello
```

と出て、TELNET クライアントと通信ができていることがわかります。このようなやりとりを Arduino などのマイコンで受けることで、Arduino で Wi-Fi 接続を利用することができます。

AT コマンドはプログラムしなくてもいいのですが、ちょっと使い方が悪いですね。では、自分でプログラムにチャレンジしてみましょう!!

# プログラムしよう!!

## 開発のしかた

Arduino IDE という開発環境を使ってコンパイル・書き込みをします。(なお、この方法を使うと前ページの AT コマンドの動作は上書きされて使えなくなります。)

<http://arduino.cc> より、開発環境 Version 1.6.5 をダウンロード、インストールしてください。

開発環境より、「ファイル」-「環境設定」で、Additional Boards Manger URLs:に

[http://arduino.esp8266.com/staging/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/staging/package_esp8266com_index.json)

を入力し、OK を押します。

「ツール」-「ボード」-「Boards Manager...」で「esp8266 by ESP8266 Community」

と出てくるので「Install」を押します。

インストールが終わると、「ツール」-「ボード」に「Generic ESP8266 Module」が出てくるのでそれを選択します。「ツール」-「ポート」を、使用する USB シリアルアダプタのものに変更しておきます。わからない場合は巻末のトラブルシューティングを参照してください。

## ESP8266 のブートローダーモードと LED の接続

起動時に GPIO がどのレベルになるかで、ESP8266 の動作モードが以下のように変わります。

UART Download Mode	GPIO0 Lo GPIO2 Hi GPIO15 Lo
Normal	GPIO0 Hi GPIO2 Hi GPIO15 Lo
SD-Card Boot	GPIO0 Lo GPIO2 Lo GPIO15 Hi

写真のようにジャンパーを BOOT 側にしてから電源を入れることで、UART Download Mode となり、シリアルアダプタ(UART)からプログラミングができます。

電源の ON/OFF はレギュレータの足へケーブル抜き差ししてできるのが簡単ですね!

また、LED の接続は以下の割り当てになっています。

- GPIO16 RED LED
- GPIO5 GREEN LED
- GPIO4 BLUE LED

## LED をチカチカしよう

「ファイル」「スケッチの例」で「ESP8266」「Blink」を選択します。出てくる BUILTIN\_LED というのは



```
void setup() {
  pinMode(16, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(4, OUTPUT);
}

void loop() {
  int i;
  for ( i=0; i<8; i++){
    if ( 0 < ( i & 1 ) ){
      digitalWrite(16, HIGH);
    } else {
      digitalWrite(16, LOW);
    }
    if ( 0 < ( i & 2 ) ){
      digitalWrite(5, HIGH);
    } else {
      digitalWrite(5, LOW);
    }
    if ( 0 < ( i & 4 ) ){
      digitalWrite(4, HIGH);
    } else {
      digitalWrite(4, LOW);
    }
    delay(1000);
  }
}
```

マイコンボードへの書き込みが完了しました。

```
starting app without reboot
esp8266_send_command: sending command header
esp8266_send_command: sending command payload
esp8266_send_command: receiving 2 bytes of data
closing bootstrap
idle, Serial: 80 MHz, 40MHz, DIO, 115200, 512K (94K SPIFFS), ok, Disabled, None on /dev/ttyUSB0
```



GPIO13 番のことなので、これを以下のように書き換えます。

```
void setup() {  
    pinMode(4, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(4, LOW);  
    delay(1000);  
    digitalWrite(4, HIGH);  
    delay(2000);  
}
```

実行して青い LED が光れば OK です。



このボタンを  
押してコンパイル  
と書き込み

```
ESP8266 Blink Example  
void setup() {  
    pinMode(16, OUTPUT);  
    pinMode(5, OUTPUT);  
    pinMode(4, OUTPUT);  
}  
  
void loop() {  
    int i;  
    for ( i=0; i<8; i++ ) {  
        if ( 0 < ( i & 1 ) ) {  
            digitalWrite(16, HIGH);  
        } else {  
            digitalWrite(16, LOW);  
        }  
        if ( 0 < ( i & 2 ) ) {  
            digitalWrite(5, HIGH);  
        } else {  
            digitalWrite(5, LOW);  
        }  
        if ( 0 < ( i & 4 ) ) {  
            digitalWrite(4, HIGH);  
        } else {  
            digitalWrite(4, LOW);  
        }  
        delay(1000);  
    }  
}
```

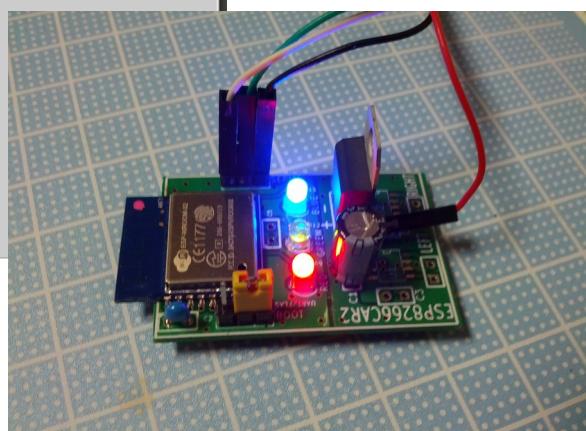
こっちの  
プログラムは  
LED 3つを点滅  
させるように  
なっている

正常に  
書き込みが  
できたときの  
表示

```
マイコンボードへの書き込みが完了しました。  
.....  
starting app without reboot  
espcomm_send_command: sending command header  
espcomm_send_command: sending command payload  
espcomm_send_command: receiving 2 bytes of data  
closing bootloader  
Module, Serial, 80 MHz, 40MHz, DIO, 115200, 512K (64K SPIFFS), ck, Disabled, None on /dev/ttyUSB0
```

## Web から LED をチカチカしよう

サンプルの Advanced Web Server を基にして、Web ブラウザから3色の LED を ON/OFF できるようにしてみました。



```

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

const char* ssid = "cnccafe";
const char* password = "freebeer";

ESP8266WebServer server(80);

const int Rled = 16;
const int Gled = 5;
const int Bled = 4;
void returnHTMLDisp(){
    char temp[300];
    int sec = millis() / 1000;
    int min = sec / 60;
    int hr = min / 60;

    sprintf ( temp, 300,
    "

```

```

void ledROff() {
    Serial.println("RED LED OFF");
    digitalWrite(Rled, LOW);
    returnHTMLDisp(); }
void ledGOn() {
    Serial.println("GREEN LED ON");
    digitalWrite(Gled, HIGH);
    returnHTMLDisp(); }
void ledGOFF() {
    Serial.println("GREEN LED OFF");
    digitalWrite(Gled, LOW);
    returnHTMLDisp(); }
void ledBOn() {
    Serial.println("BLUE LED ON");
    digitalWrite(Bled, HIGH);
    returnHTMLDisp(); }
void ledBOff() {
    Serial.println("BLUE LED OFF");
    digitalWrite(Bled, LOW);
    returnHTMLDisp(); }
}

void handleNotFound(){
    String message = "File Not Found\n\n";
    message += "URI: ";
    message += server.uri();
    message += "\nMethod: ";
    message += (server.method() == HTTP_GET)?"GET":"POST";
    message += "\nArguments: ";
    message += server.args();
    message += "\n";
    for (uint8_t i=0; i<server.args(); i++){
        message += " " + server.argName(i) + ":" + server.arg(i) + "\n";
    }
    server.send(404, "text/plain", message);
}

void setup(void){
    pinMode(Rled, OUTPUT);
    pinMode(Gled, OUTPUT);
    pinMode(Bled, OUTPUT);
    digitalWrite(Rled, LOW);
    digitalWrite(Gled, LOW);
    digitalWrite(Bled, LOW);
    Serial.begin(115200);
    WiFi.begin(ssid, password);
    Serial.println("");

    // Wait for connection
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.print("Connected to ");
    Serial.println(ssid);
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
    if (MDNS.begin("esp8266")) {
        Serial.println("MDNS responder started");
    }
    server.on("/", handleRoot);
    server.on("/Ron", ledRON);
    server.on("/Roff", ledROff);
    server.on("/Gon", ledGOn);
    server.on("/Goff", ledGOFF);
    server.on("/Bon", ledBOn);
    server.on("/Boff", ledBOff);
    server.onNotFound(handleNotFound);
    server.begin();
    Serial.println("HTTP server started");
}

void loop(void){
    // Handle incoming connections
    server.handleClient();
}

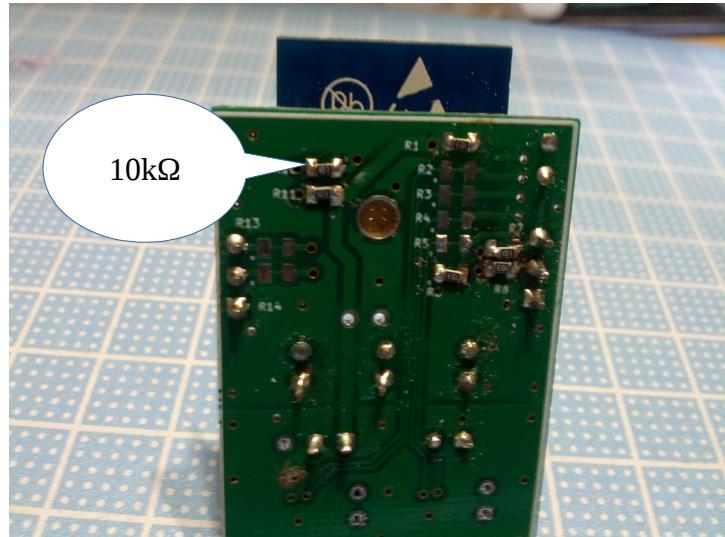
```

## センサーをつなげる

ESP8266 には、アナログの信号を読み取ることのできるポートが1つあります。これにセンサーを繋げてみましょう。例では CdS という光センサーをつなげてみました。例と同じように実験をするには、別途以下の部品を買い揃えてください。

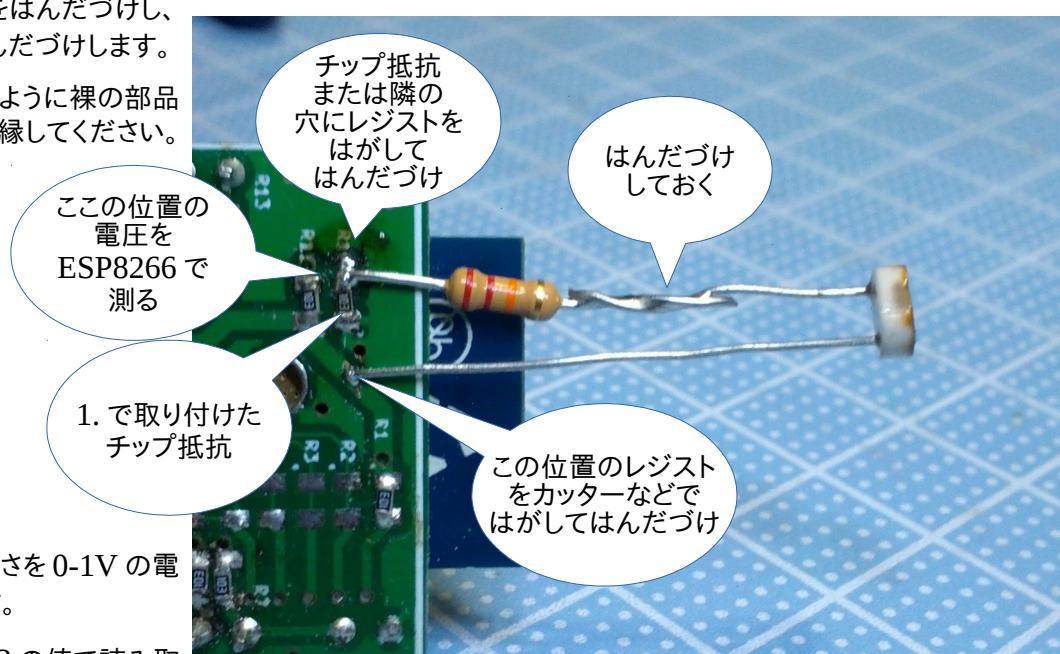
- CdS 光センサー 明抵抗  $10k\Omega$  のものを使いましたが、明抵抗  $50k\Omega$  ぐらいの方が使いやすいでしょう。
- 1/4W 抵抗  $22k\Omega$  抵抗値調整用 CdS の明抵抗が十分高ければ必要ありません。
- チップ抵抗  $10k\Omega$  分圧専 キット付属の余りをを使いました。

1.チップ抵抗を基板の裏にはんだづけします。



2.CdS と  $22k\Omega$  抵抗をはんだづけし、更に写真の箇所にはんだづけします。

作例ではわかりやすいように裸の部品を使ってますが適宜絶縁してください。



3.この回路は光の明るさを 0-1V の電圧に変換する回路です。

ESP8266 では 0-1023 の値で読み取ります。

しかしながらこの回路にかかる電圧は 3.3V なのでうまくして信号変化が 0-1V に収まるようにしないといけません。ここでは CdS を使いましたが、他のセンサーを使う場合も同様にして工夫してみてください。

4.このスケッチを実行すると、明るさを表示するWebサーバとして動作します。このスケッチはサンプルプログラムの「HelloServer」を基に作成しました。

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

extern "C" {
#include "user_interface.h"
}

const char* ssid = "cnccafe";
const char* password = "freebeer";

ESP8266WebServer server(80);

const int led = 13;

void handleRoot() {
    digitalWrite(led, 1);
    int ret;
    ret=system_adc_read();
    String message = "<!DOCTYPE html><title>Sensor Server</title>";
    message +="<p><h1>Ambient: ";
    message += ret;
    message +="</h1></p>";
    server.send(200, "text/html", message);
    Serial.println(message);
    digitalWrite(led, 0);
}

void handleNotFound(){
    digitalWrite(led, 1);
    String message = "404 Not Found\n";
    server.send(404, "text/plain", message);
    digitalWrite(led, 0);
}

void setup(void){
    pinMode(led, OUTPUT);
    digitalWrite(led, 0);
    Serial.begin(115200);
    WiFi.begin(ssid, password);
    Serial.println("");
}

// Wait for connection
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.print("Connected to ");
Serial.println(ssid);
Serial.print("IP address: ");
Serial.println(WiFi.localIP());
```



```
if (MDNS.begin("esp8266")) {
    Serial.println("MDNS responder started");
}

server.on("/", handleRoot);

server.on("/inline", [](){
    server.send(200, "text/plain", "this works as well");
});

server.onNotFound(handleNotFound);

server.begin();
Serial.println("HTTP server started");

void loop(void){
    server.handleClient();
}
```

# モーターを動かす

ここで使用するモータードライバ TC117HS は、電源 2.2V～5.5V で内部抵抗 1Ω、1 A 程度流せることができる使いやすい H ブリッジドライバです。

プログラムからは、以下のようにコントロールします。

- モータ A (LEFT) コントロール:

GPIO13 → INA

GPIO 15 → INB

- モータ B (RIGHT) コントロール:

GPIO14 → INA

GPIO12 → IN B

	待機状態	正転	逆転	ブレーキ
INA	L	H	L	H
INB	L	L	H	H

右に、モーターのテストプログラムを記します。前進→後退→右回転→左回転→停止 を繰り返します。

うまくいったら、他のプログラムを参考に Web 経由でのコントロールなどにチャレンジしてみましょう！

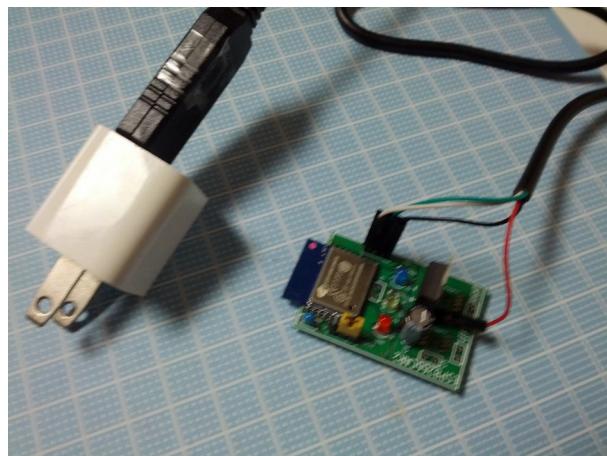
```
void setup() {
  Serial.begin(115200);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(14, OUTPUT);
  pinMode(15, OUTPUT);
}

void loop() {
  // Forward
  Serial.println("Forward");
  digitalWrite(13, HIGH);
  digitalWrite(15, LOW);
  digitalWrite(14, HIGH);
  digitalWrite(12, LOW);
  delay(3000);
  // Back
  Serial.println("Back");
  digitalWrite(15, HIGH);
  digitalWrite(13, LOW);
  digitalWrite(12, HIGH);
  digitalWrite(14, LOW);
  delay(2000);
  // Right Turn
  Serial.println("Right");
  digitalWrite(15, LOW);
  digitalWrite(13, HIGH);
  digitalWrite(14, LOW);
  digitalWrite(12, HIGH);
  delay(2000);
  // Left Turn
  Serial.println("Left");
  digitalWrite(15, HIGH);
  digitalWrite(13, LOW);
  digitalWrite(14, HIGH);
  digitalWrite(12, LOW);
  delay(2000);
  // STOP
  Serial.println("Stop");
  digitalWrite(15, HIGH);
  digitalWrite(13, HIGH);
  digitalWrite(14, HIGH);
  digitalWrite(12, HIGH);
  delay(2000);
  // IDLE
  Serial.println("Idle");
  digitalWrite(15, LOW);
  digitalWrite(13, LOW);
  digitalWrite(14, LOW);
  digitalWrite(12, LOW);
  delay(2000);
}
```

## 電源を工夫する

プログラム中は PC の USB コネクタから 5V を取っていますが、プログラム開発が終わって PC から切り離して使いたいときはどうしたらいいでしょうか？

USB シリアルアダプタをそのまま USB 電源アダプタや、モバイルバッテリーに接続すれば OK です（写真はちょっとかっこいい USB シリアルシリアルアダプタを使っています）。



USB シリアルアダプタが勿体ない時は、USB の線をバラして 5V と GND をモジュールに接続すれば OK です。

また、今回使用しているレギュレーターは LD33CV(ST 社 LD1117 の 3.3V 仕様)で、15V までの入力できます。9V のバッテリや 12V の AC アダプタなども使えます。

また、乾電池 2 本で使用することもできます。その場合の接続方法は、「ESP CAR CONTROLLER 組み立て方法(2) モータコントロール実験のための組立」での電池ボックスの接続を参考にしてください。

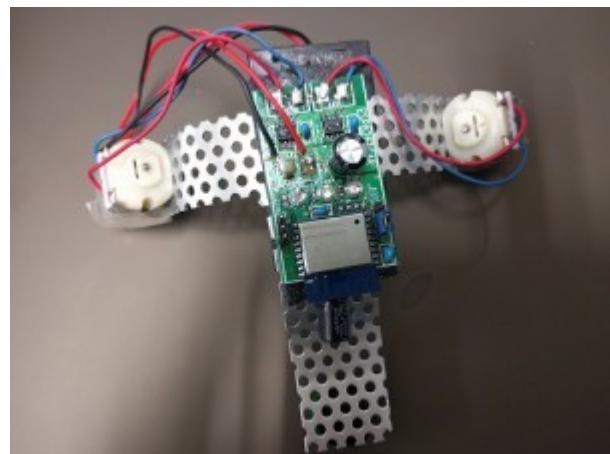
# Esp Car Gadgets の作例

## その1

ESP 8266 でリモコンロボットを動かしてみました。(ロボットと言うのも何なので、ESP CARにしてみました。)

<http://www.qa65000.com/2016/01/30/esp-car-gadgets/>

モーター軸を直接床に付けて駆動する方法をとり、減速ギアなどがないりません。きびきびと機動します。  
モータに FA-130 を使いましたが、これは電流をあまりにも引っ張るため結構厳しいです。FA130 モーターを使わなければコイルや、電解コンデンサはすくなくて済むと思います。  
EPS8266 は電圧・ノイズにかなりシビヤです。モーターのノイズでフリーズとかしますのでご注意ください。  
動画は youtube で「Esp Car Controller Demo」として公開しています。



## その2

FA-130 でなく低電流モーターを使い、LiPo バッテリと共にレゴブロック互換の筐体に収めたものです。  
Scratch でコントロールするようにしています。

<http://tiisai.dip.jp/?p=3665>



# ソフトウェア作例

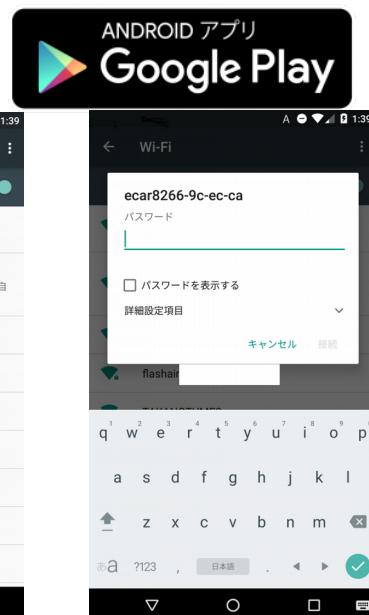
## コントロールアプリ

esp8266 用の udp でコントロールするアプリをリリースしました。後述のファームウェアを、ESP8266 に書き込むことで、このアプリが動作するようになります。

データの保持には@pik さんの [FMX.IniFile](#) を使用させて頂いています。ありがとうございます。

## Esp Car Controller For Android.

アプリは Google Play で¥200 で販売中です。



### アプリ設定方法

esp8266 が access point となります。Android の Wi-Fi を設定します。

'ecar8266-xx-xx-xx'を選択しましょう。 xx xx xx の部分は esp8266 内部の mac アドレスです。

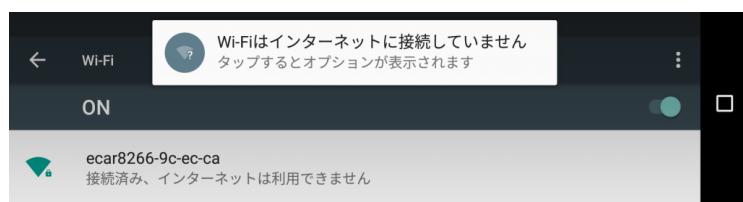
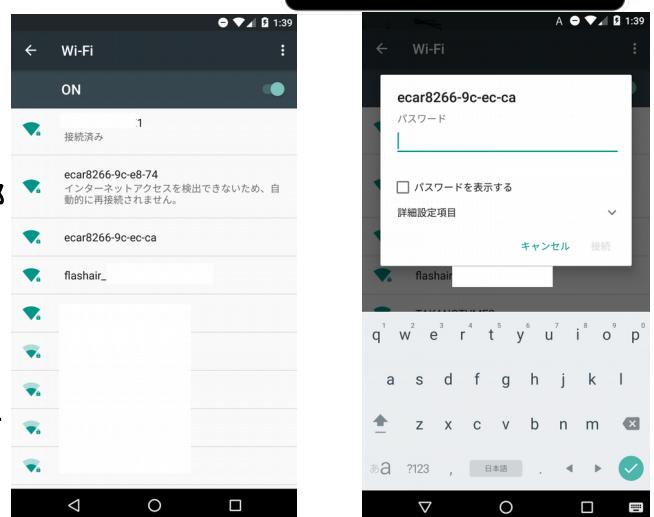
例では ecar8266-9c-ec-ca を選択します。

password は 12345678 です。

通常はここでつながります。ip アドレスが割り当てられているか確認しましょう。

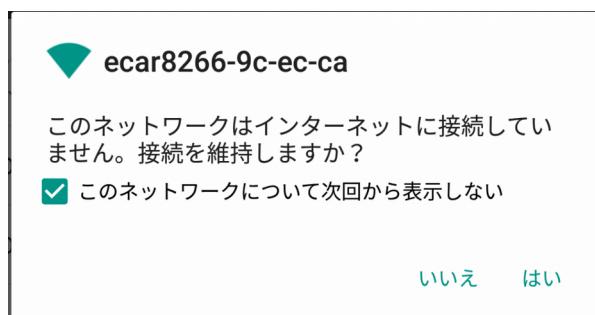
確認は詳細設定を押します。

Nexus5x の場合ここからタップ要求が必要でした。約10秒程度待つと下記の画面がでますのでタップしましょう。



### タップしましょう。

すると下記の画面がでますので、



はい。押します。

ここで ip アドレスが割り当てられているか確認しましょう。確認は詳細設定を押します。

## 操作説明

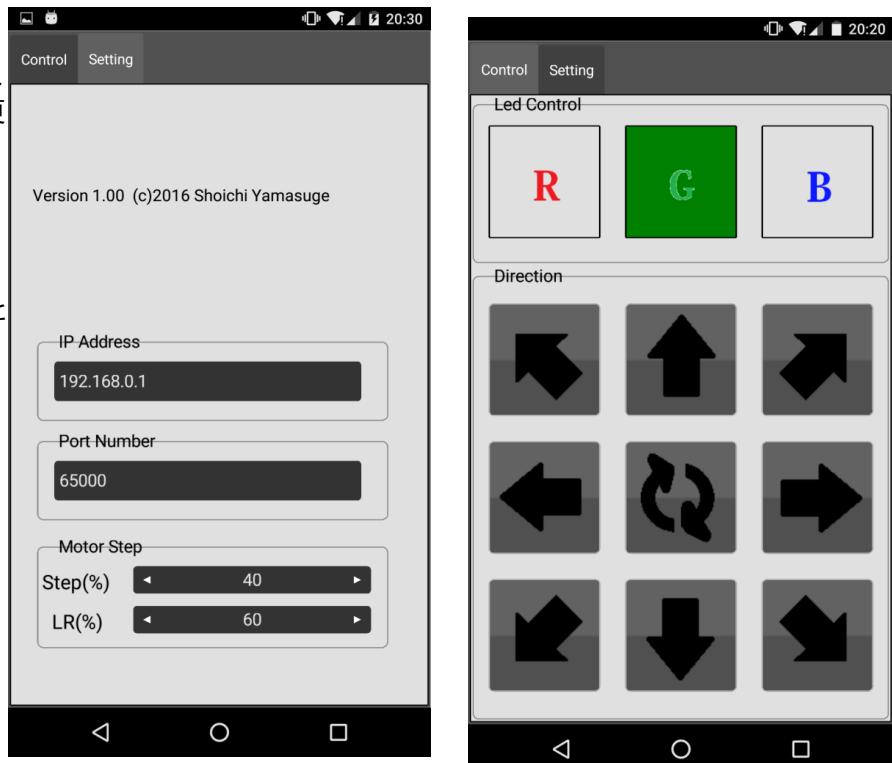
1.ip アドレスのところに 192.168.0.1 を入れます。これはガジェット側のソースを変更していなければこれで ok です。

2.ポート番号もこのままで 65000 で良いです。

3.step ですが、これは最大出力を 100%としたときに何%出力を出すか?という設定です。スピードコントロールとして設定します。

4.これは右前／左前、右後／左後方向に弱くするモーター側を正常時の何%出すかという設定です。適当に設定します。

次に上の Control のところを押します。



では、楽しく遊びましょう。

## Esp Car Controller For iOS.

iOS についても、Android と同様のアプリを App Store で \$1.99 で販売しています。

詳細はこちら: <http://www.qa65000.com/2016/02/01/esp-car-controller-for-ios/>



## ESP8266 側のモーターコントローラームウェア

ESP8266 側のソースコードは GitHub にて公開中です。

<https://github.com/i65000/EspCarController/>

# トラブルシューティング!!

## 動かなくなった!

ジャンパーを BOOT 側にしてプログラムを書き込みし、動作しますが、2回めの動作はジャンパーを戻してから電源を ON にしないといけません（ジャンパーを BOOT 側にしたままだとプログラム書き込み待機状態となる）。

## Wi-Fi が接続できない

ESP8266 が接続できる Wi-Fi は 2.4GHz のみとなっています。5GHz 帯の Wi-Fi には接続できませんのでご注意ください。また、2.4GHz 帯のものも、電波が混雑していると接続が不安定になります。

## モーター制御が不安定

例えば FA130 モータなどは、ストールした時に 1000mA ぐらい流れたりします。そのために、ESP8266 への電源電圧が不安定となっていないかを確認しましょう。

また、モーターはノイズが多く出ます。対策としてコンデンサーをモータに直接つけるのも効果的です。

## 書き込みできない！

付属の USB シリアルアダプタは、USB ポートの 5V 電圧が足りないと問題が起こりやすくなります。その場合、PC の USB ポートに直接接続してみてください。

また、下記「シリアルポートのデバイスネーム」も参照してください。

## シリアルポートのデバイスネーム

取り付けたシリアルポートのデバイスネームは、MS-Windows の場合は COM0 や COM12 のように、Mac OSX の場合は tty.なんとかんとか のような名前、Linux の場合は ttyUSB0 や ttyUSB1 のような名前になります。もしわからぬ場合は、下記「ArduinoIDE で USB シリアルアダプタの確認と設定をする」を参照ください。

## ArduinoIDE で USB シリアルアダプタの確認と設定をする

- 「ツール」-「シリアルポート」で設定します。また、今システムで認識している USB ポートの確認もできます。認識しているものが一覧で表示されます。
- それが USB シリアルアダプタのポートなのかわからない場合は、シリアルアダプタを取り付けたときと、取り外したときを比較して、取り付けた時に現れるシリアルポートを選択します。Mac の場合は 2 つ同時に現れます、tty という名前のついているほうが正解です。
- なお、シリアルポートは何度も付け外ししていると番号が変わることがあります。その際は同様に「ツール」-「シリアルポート」で選択しなおしてください。

## Mac の場合の注意点

/dev/cu.usbserial 等になっている場合は、/dev/tty.usbserial を選択

## Linux の場合の注意点

もし、シリアルデバイス( /dev/ttyUSB0 等 )を使用する権限が無い場合は、一般ユーザで動かすために、利用ユーザを dialout グループに登録する。※デバイスファイルは dmesg コマンドなどで確認可能。

## シリアル通信ソフト

MS-Windows の場合は Tera Term、Linux の場合は minicom が良く使われます。Mac OSX の場合は jerm というソフトがおすすめです(ダウンロードしてください)。jerm -b 115200 -p none -d 8 -s 1 -f none /tty/usb なんとか(調べたデバイス名)として起動します。

ちょっとよくわからないな…という方は、Arduino IDE をシリアル通信ソフトとして使うことができます。

## Arduino IDE をシリアル通信ソフトとして使用する

「ツール」-「シリアルモニタ」で起動します。「115200bps」「CR および LF」として設定してください。

## こわしちゃったよ!!

小さいパーツではんだづけをするために、なかなかうまくいかないことが多いと思います。パーツなどを壊してしまっても、遠慮せずにご相談ください。相談先は下記の「日本 Android の会ロボット部」まで!

## もっと知りたい!!

開発やノウハウ、新版のソフトウェアや基板の公開については <http://www.qa65000.com/> を参照してください。また、日本 Android の会ロボット部にて、月例で勉強会を行なっています。興味のある方は ML(メーリングリスト)にご登録ください。

<http://groups.google.com/group/robot-android-group-japan-akb>

ML では勉強会のアナウンスや部員が興味をもった様々な話題がなされています。

