~~2015 年 3 月 5 日~~

2015 年 3 月 16 日

京都産業大学　蚊野浩

1. システムの外観

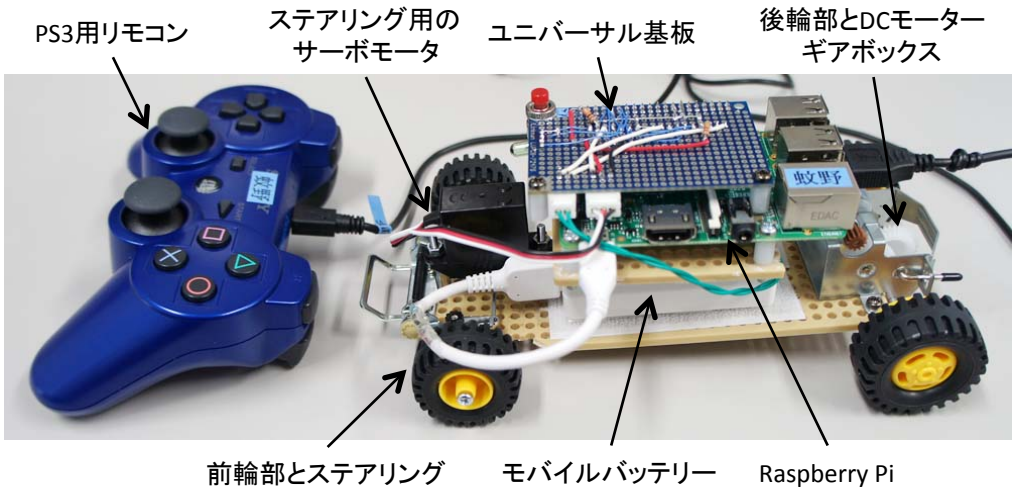

図 1　Raspberry Pi を用いた模型自動車

2. 部品リストと説明

表 1 部品リスト

名称・型番	数量	単価	コメント
Raspberry Pi Model B+	1	5,000	
マイクロ SD カード 8GB	1	1,000	
タミヤ、楽しい工作シリーズ No.112	1	1,500	前後輪周りを利用
タミヤ、楽しい工作シリーズ No.98	2	400	ユニバーサルプレート 1.5 枚使用
ユニバーサル基板 Linkman LUPCB-7247S	1	110	
モータードライバ IC TA7291P	1	150	後輪の DC モーター駆動用
74HC125	1		RaspPi の GPIO のバッファ用
その他、コンデンサ、抵抗、LED、スイッチ、コネクタ、40 ピン基板用ピンフレーム			
サーボモーター、GWS/MICRO/STD/F	1	1,100	1.8kgcm(4.8V), 28x14x30
PS3 用ワイヤレスリモコン ソニー CECHZC2J	1	5,000	
モバイルバッテリー　パナソニック QE-QL105	1	1,500	2,900mAh/3.7V, 1A/5V
充電専用 Micro-USB ケーブル,10cm	1	500	
その他、スペーサ、マジックテープ、ネジなど	合計 17,000 円程度		

幾つかの部品について説明する。

① モータードライバ IC TA7291P: 電子工作で DC モーターの駆動によく使われる IC である。後に示す図 3 の回路図からわかるように、この IC の 2 本の出力端子を DC モーターの 2 本の端子に接続する。端子間に流れる電流の方向によって回転方向が決まり、電圧・電流によって回転速度・トルクが決まる。

② サーボモーター:モーターの軸角度を外部からの制御信号によってコントロールできるモーター。制御信号としてパルス状の方形波を用いる。パルス幅によって角度を制御する。

3. システム構成と回路図

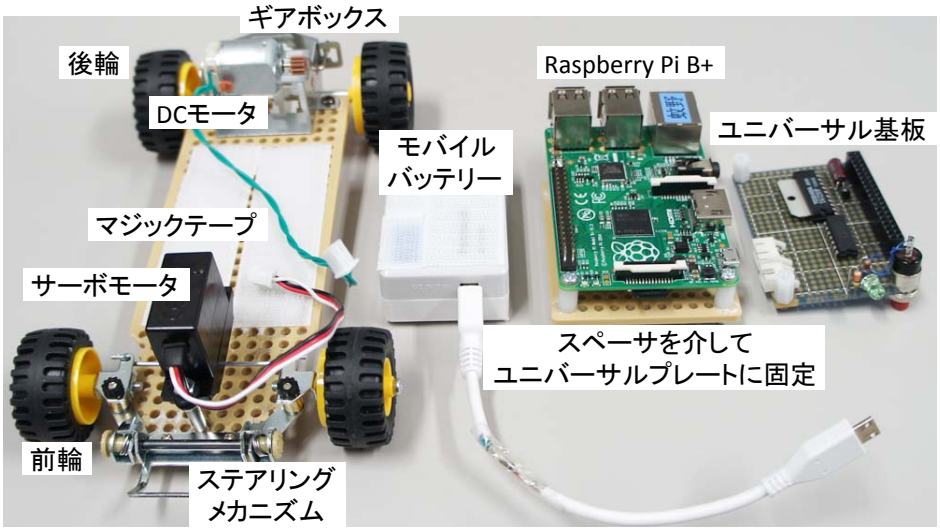

図 2 主な要素を展開した写真

図 2 に主な構成要素を示す。図左の台車は、ユニバーサルプレートを加工し、タミヤ楽しい工作シリーズ No.112「バギー工作基本セット」を装着したものである。台車中央に、マジックテープでモバイルバッテリーを固定する。Raspberry Pi 基板を、スペーサを用いて、ユニバーサルプレートに固定する。ユニバーサルプレートの裏面を、マジックテープでモバイルバッテリーの上に固定する。図右のユニバーサル基板は DC モーターの駆動回路などを実装し、Raspberry Pi 基板のピンヘッダと接続する。

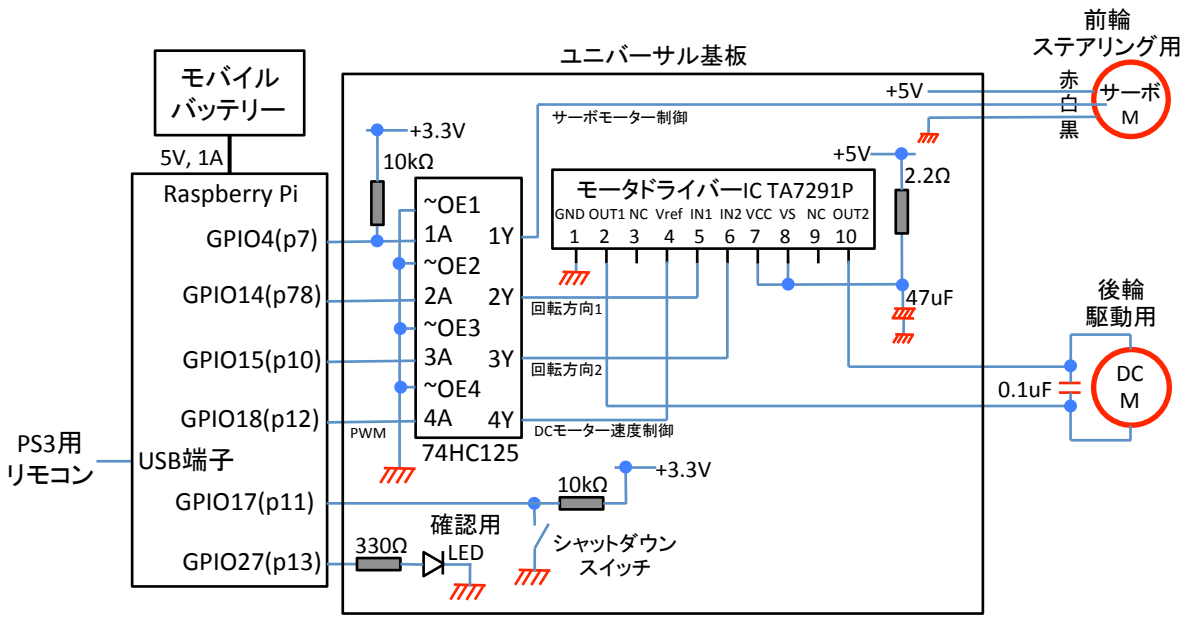

図 3 システムの接続とユニバーサル基板の回路図

図 3 にシステムの接続とユニバーサル基板の回路図を示す。回路設計に対するコメントを以下に示す。

① Raspberry Pi の GPIO 端子を保護するため、バッファ IC として 74HC125 を用いた。シャトダウンスイッチと動作確認用 LED は後付けしたため、GPIO17 と GPIO27 はバッファされていない。手作り基板の場合、GPIO 端子を外部回路に直接接続するのではなく、バッファ IC を介して接続するのが安全であると判断した。

② 当初、DC モーターを駆動すると Raspberry Pi が暴走した。暴走する原因は、DC モーターのノイズとモバイルバッテリーの電圧降下が考えられる。電源として AC アダプターを用いると、AC アダプターの仕様がモバイルバッテリーと同じ 5V・1A であっても、暴走しなかった。このことから暴走の原因は、DC モーターに電流が流れたことによる、モバイルバッテリーの電圧降下であると判断した。DC モーターに供給する電流の変化を緩和することにより症状が改善すると考え、TA7291P の VCC 端子と+5V の間に低抵抗（2.2Ω）を挿入した。この対策により、暴走しなくなった。

③ 当初、電源投入時などにサーボモーターが勝手に動くという問題が発生した。サーボモーターは、Raspberry Pi から制御パルスが入力された時だけ動作し、それ以外は静止して欲しいのだが、電源投入時とシャトダウン時に位置を変えることがあった。その原因は、電源投入時とシャトダウン時の信号値が不安的になっていることである。一般的に、入出力に利用する端子は、未定義状態においてハイインピーダンスに設定され、信号値が不安的になる。そこで、http://homepage1.nifty.com/Yamatatsu/robot/rc-servo.html の記述も参考にし、GPIO4 の出力を、10kΩを介して 3.3V にプルアップした。これによって電

源投入時の不安的な動作はなくなったようである。しかし、シャトダウン時に生じる動きは、少し、残っている。なお、Raspberry Pi の GPIO は+3.3V/0V で動作するので、間違えて、5V を接続しないこと。

4. プログラム

模型自動車用のプログラムを python で開発した。そのプログラムを付録 1 に示す。このプログラムの動作については、コード中のコメントを参照のこと。

前輪のステアリングに用いるサーボモーターにパルス状の方形波を送る必要がある。サーボモーターの制御に適したプログラムとして ServoBlaster を利用した。ServoBlaster の使い方については、書籍「Raspberry Pi で遊ぼう 改訂第２版」（ラトルズ発行、2014 年 1 月）の pp.181-187 を参考にした。

後輪を駆動する DC モーターの速度・トルクを制御するために、PWM（Pulse Width Modulation、パルス幅変調）を用いた。図 3 において、TA7291P の Vref 端子の電圧を 0V～VCC の間で調整すると、DC モーターの速度・トルクを変えることができる。このようなアナログ電圧による調整と同等の機能を、デジタル的に実現する方法として PWM がある。PWM では、Vref に方形波を加え、方形波のデューティ比によって速度・トルクを調整する。Raspberry Pi は、GPIO18 を PWM に利用できるように設計されているので、この端子を利用した。

Raspberry Pi を搭載した模型自動車は、プログラムを開発する段階ではディスプレイ、キーボード、マウスを接続し、Linux パソコン状態で使用する。一方、これらの周辺機器を接続しないで模型自動車を動作させるには、電源投入後、自動的に付録 1 のプログラムが起動される必要がある。付録 1 のプログラムを/home/pi/python/autorun.py とする。~~/etc/init.d/にシェルスクリプトを置いて、実行可能状態にすると、起動時に自動的に実行される。今回、autorun_script というファイル名で、付録 2 のものを準備した。これを作成した後、ターミナルから以下のコマンドを入力する（ターミナルは/etc/init.d/にあるとする）。~~

~~# sudo chmod 755 autorun_script~~

~~# sudo update-rc.d autorun_script defaults~~

まず，raspi-config を操作し，起動時にキャラクター端末で login 待ちするモードにする．次いで，/etc/inittab を編集し，

1:2345:respawn:/sbin/getty 38400 tty1 の行を#でコメントアウトし，

1:2345:respawn:/bin/login –f pi tty1 /dev/tty1 2>&1

を入力する．これで，login 待ちしないで pi で自動的にログインする．次いで，.profile の最後に行に

sudo python python/autorun.py &

を入力する．以上で，起動時に autorun.py がバックグランドで自動的に実行される．Raspberry Pi を再起動すると、自動的に autorun.py が実行される。autorun.py が動作していることは、ユニバーサル基板上の LED が点灯することことによって確認できる。また、ユニバーサル基板上にシャトダウンボタンがあり、autorun.py は、これが押された場合に、sudo shutdown –h now を実行する。

5. ノウハウ

5.1 PS3 リモコンと接続

PS3 用のワイヤレスリモコン DUALSHOCK3 と Raspberry Pi を USB で接続すると、デバイスファイル /dev/input/js0 が自動的に生成される。ここから、リモコンのデータをバイト列として読み取ることができる。リモコンの状態とバイト列の関係は、書籍「格安 PC ボードで始める電子工作超入門」（日経 Linux 編、日経 BP 社発行、2014 年 3 月）の pp.73-75 を参考にした。なお、この書籍にはブルーツースによるワイヤレス通信の方法が記載されているが、安定して動作させることができなかった。USB による有線接続は安定している。

このリモコンはワイヤレスであり、加速度センサなどのセンサ類も実装されている。これらの機能を使いこなすためには、さらなる調査が必要である。

5.2 台車に用いるユニバーサルプレートの加工

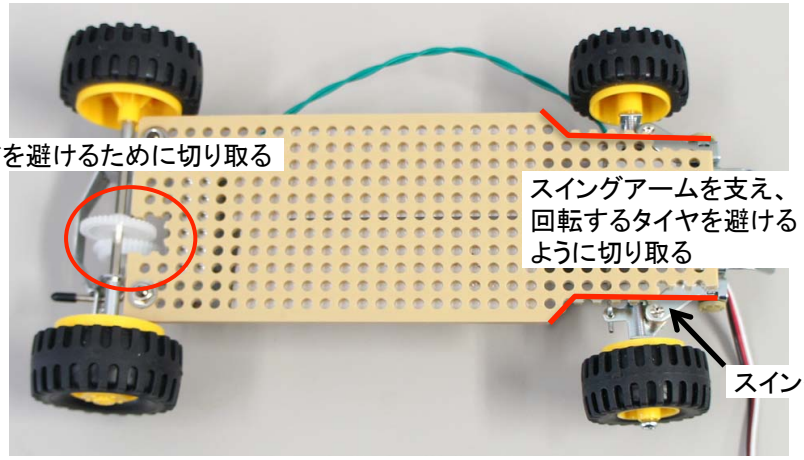

図 4 台車に用いるユニバーサルプレートの加工

「バギー工作基本セット」の前輪、後輪を取り付けるために、図 4 のように、ユニバーサルプレートを若干、加工した。前輪のスイングアームがプレートの上面で支えられるように、ユニバーサルプレートを削り過ぎないこと。

5.3 ステアリング

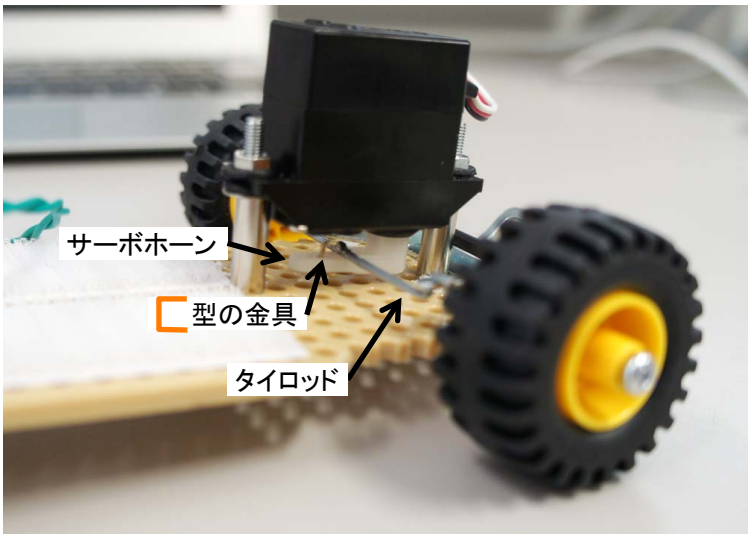

図 5 サーボモーターによる前輪のステアリング

前輪はアッカーマンタイプステアリング機構でタイヤが左右に動くようになっている。これは、タイヤの方向を回転させるアップライトがタイロッドで結合された機構であり、タイロッドを左右に移動させることで、タイヤの方向を制御する。図 5 のように、「コ」の字型の金具をタイロッドに半田付けし、この金具をサーボホーンの穴に挿入することで、サーボモターとステアリング機構を接続する。サーボモーターは±45 度程度の範囲で動作させる。この範囲を超えると、メカニズムを損傷させる可能性が高いので、注意が必要である。

5.4 モバイルバッテリーでの動作可能時間

今回利用したモバイルバッテリーは 2900mAh/3.7V の電池容量で、出力は 5V・1A である。2900mAh/3.7V は、3.7V の電圧で 1A（=1000mA）の電流を出力した場合、2.9 時間もつ、ということである。5V・1A で利用する場合、3.7V から 5V への変換によるロスを考慮して２時間弱利用できることになる。今回のシステムの電流量は、DC モーター、サーボモーターを同時に動かした場合でも、1A 未満のようであるから（全てを動作させてもシステムがダウンしない）、２時間以上の利用が可能であると期待できる。実際、１時間程度、連続して動作させても、バッテリーの残量表示は橙色（約 60～30%）にならなかった。

6. 課題、バグ

現在のシステムの課題、不具合、バグをリストアップする。

~~① /etc/init.d/autorun_script を実行可能にするとシャットダウンが完了しない。シャットダウンの完了は、Raspberry Pi 基板上の緑 LED が完全に消灯することと、ディスプレイへの信号が無くなることで確認しているが、これらが、完全には消えない。したがって、ロ~~

~~グアウトの時に再起動を選択しても、シャットダウンしないので、再起動もしない。ファイルシステムが壊れている可能性がある。今のところ、十分に時間を開けて、電源を落としている。~~

② ~~シャットダウンボタンの動作がおかしい。①の問題があるので、シャットダウンボタンが効いて sudo shutdown -h now が実行されても、完全にはシャットダウンしない。それだけではなく、autorun.py が完全には終了していない状態になる（リモコンのボタンが一部効く）。また、autorun.py はポーリングによってシャットダウンボタンの状態を読み取っているが、リモコンを机の上などに置くことで、リモコンから何も信号が送られないと、ポーリングがストップしてしまい、シャットダウンボタンが効かない。~~

①と②の問題は，/etc/init.d/を使うのではなく，ユーザ pi の.profile に実行命令を置くことで解決した.

③ ステアリングの動作が不安定。リモコンのボタンによってサーボモーターの回転位置を制御しているが、レスポンスが悪い。その原因の一つは、Linux がリアルタイム OS ではないため、python プログラムの実行に不規則な遅延が発生していることであろう。また、python はインタプリタで実行されるため、実行そのものが遅い、ということもあると考えている。