電子工作 model car

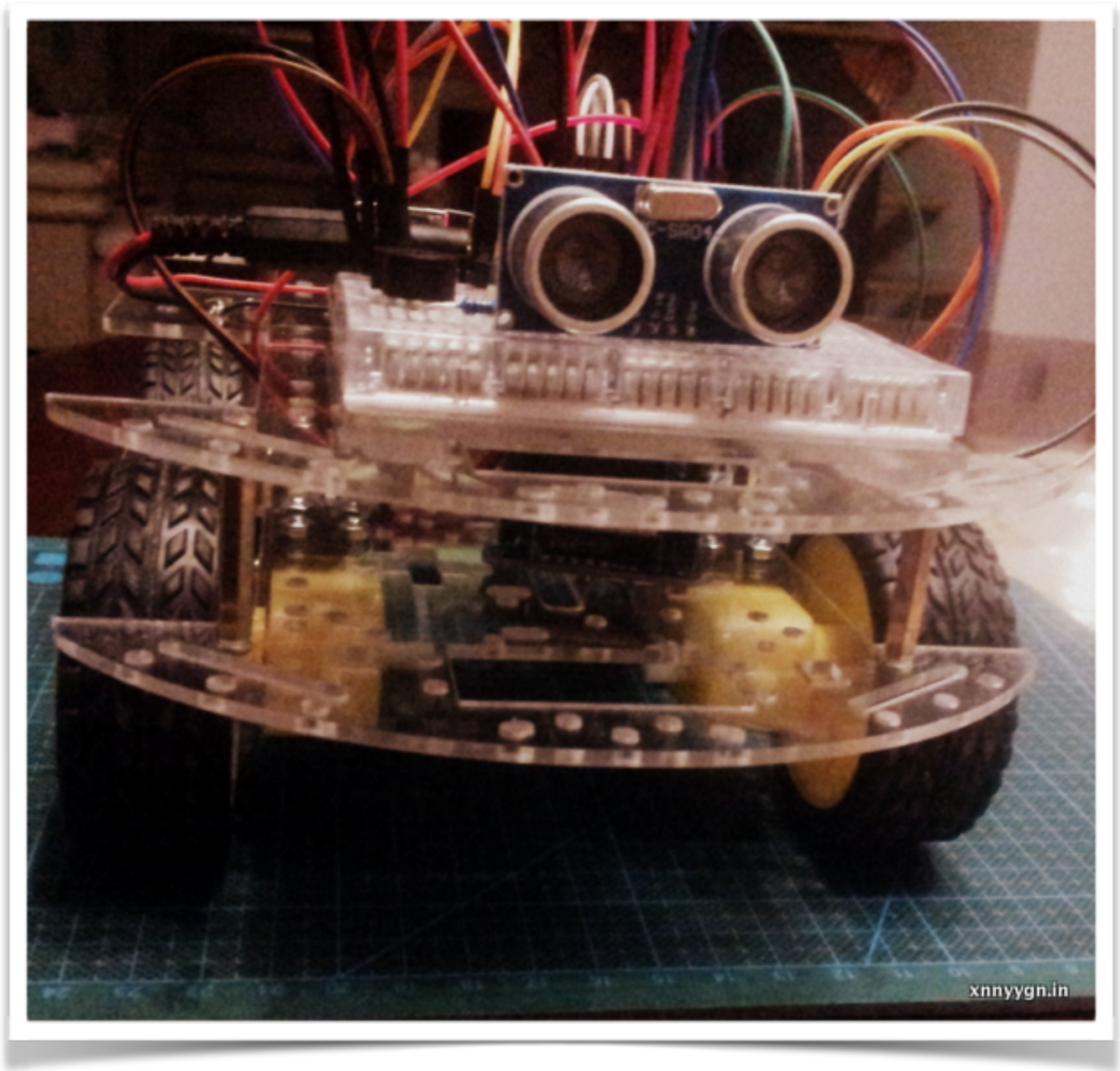
XnnYygn 2014-11-18

1 overview

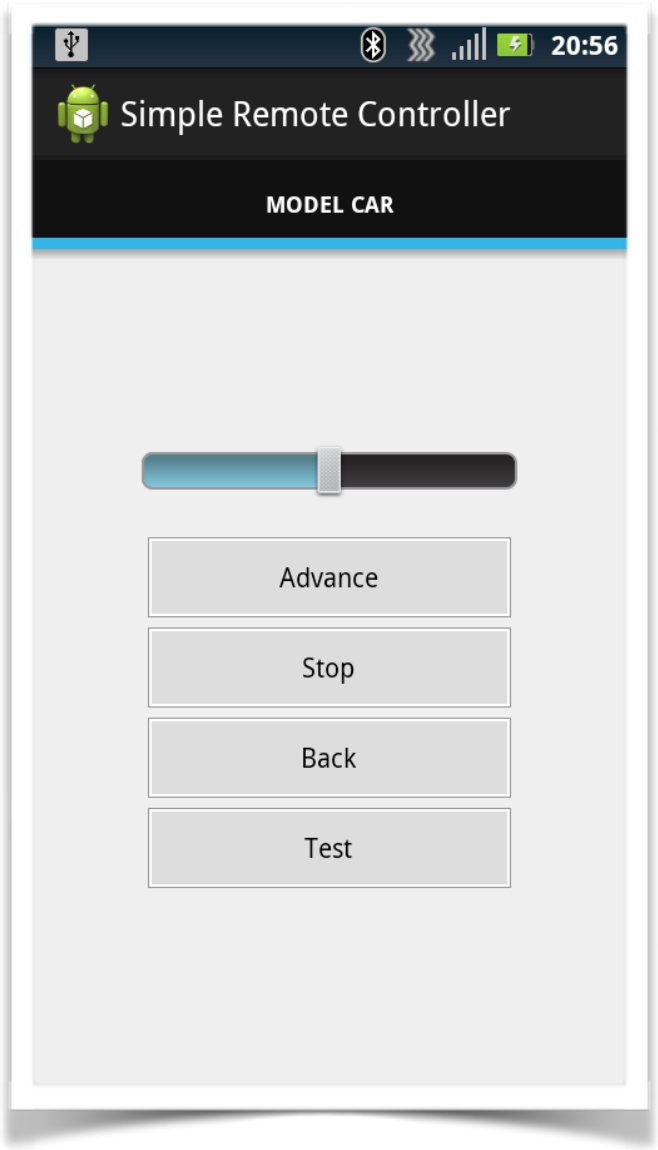
1.1 keyword

model car, bluetooth(bc04), l298n, ultrasonic sensor(hcsr04), motor, speed measurement

1.2 presentation



android controller

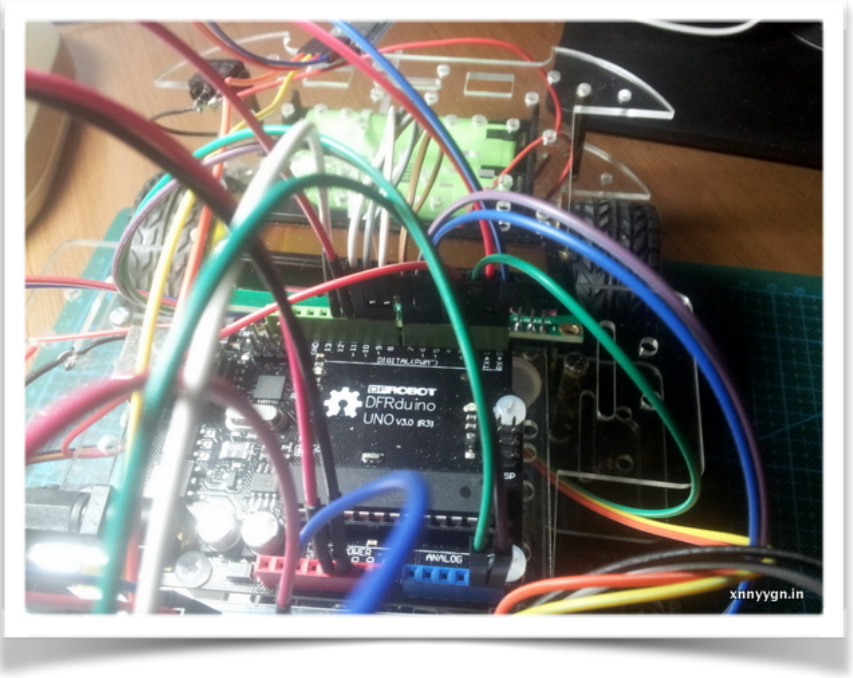


1.3 how to play

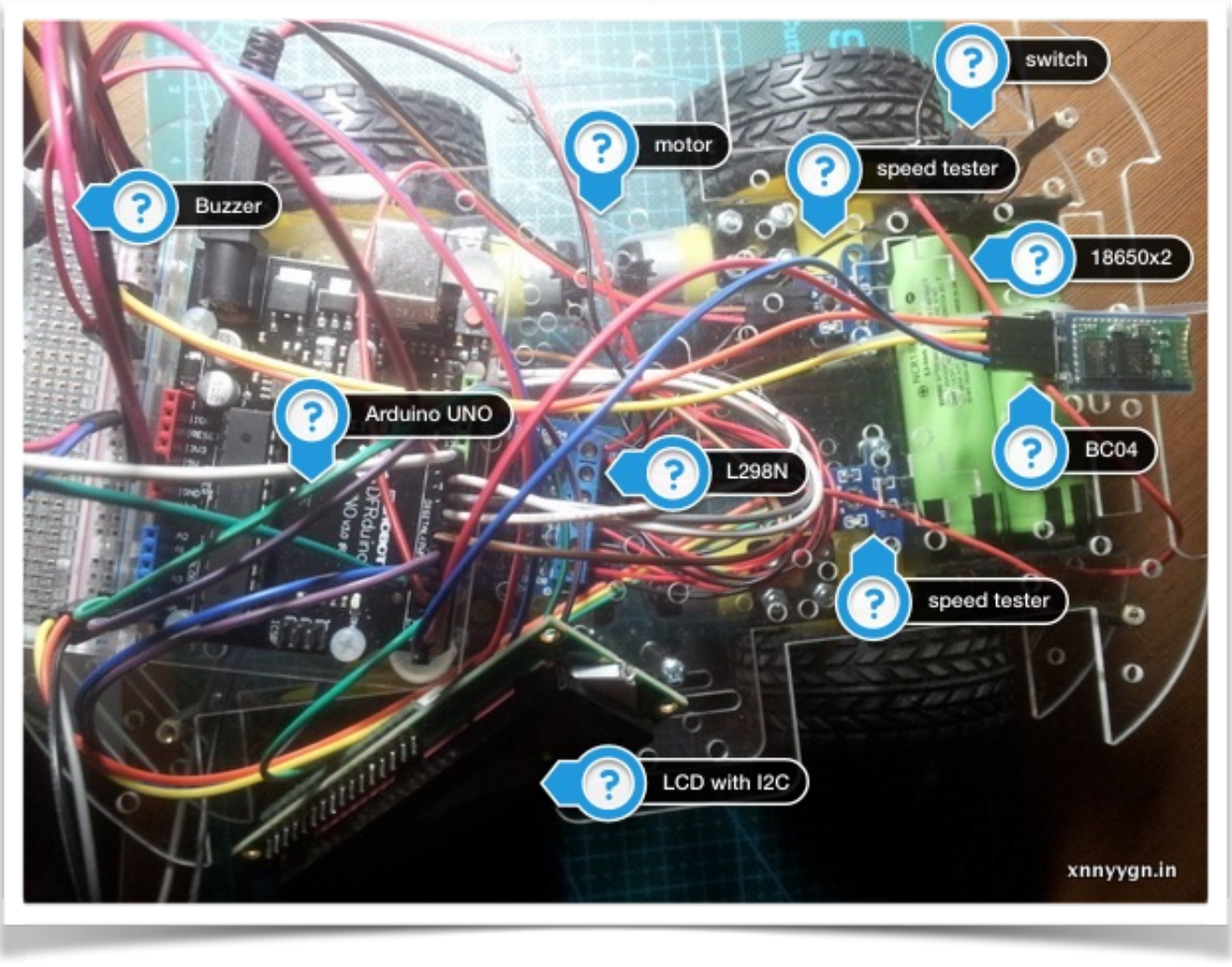
widget	action
speed seekbar	middle is full power advance or back left most is turn left(left min, right max) right most is turn right(left max, right min)
advance	left middle speed, right middle speed
stop	stop
back	left middle speed back, right middle speed back
test	test communication

実はXYZ加速度センサーで左とか右とか操作もできるが、調整の必要がある。簡単のため、ここで直接数字でコントロールする。

1.4 pictures

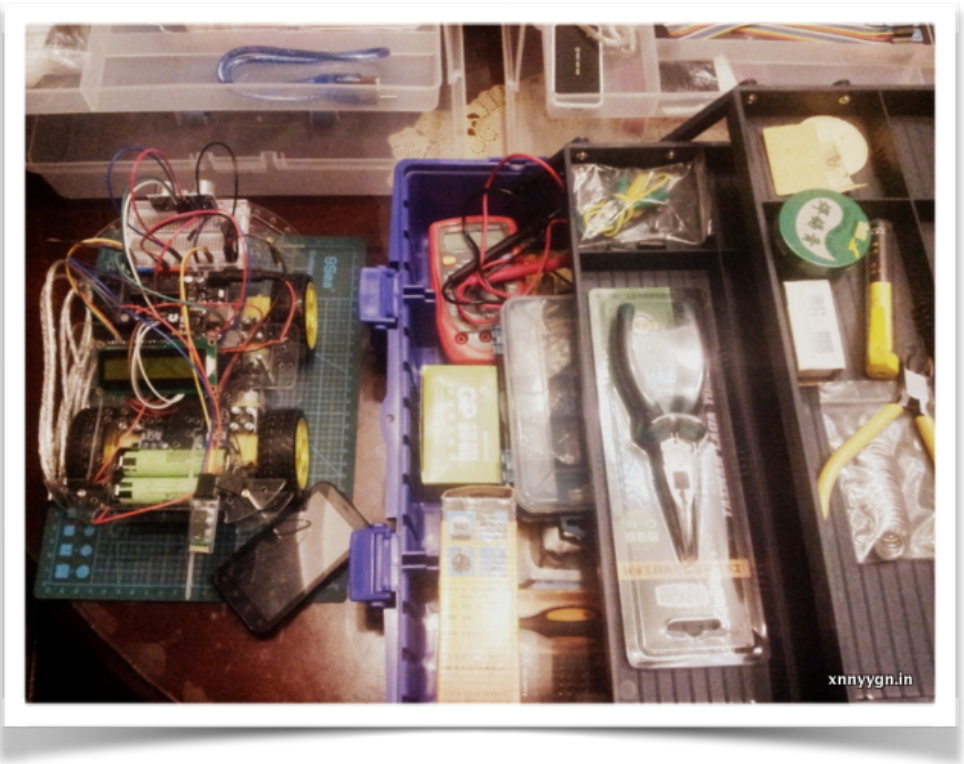


D0(RXN) ~ D12全部接続しました。これ以上はシールドやArduino Megaなどが必要です。

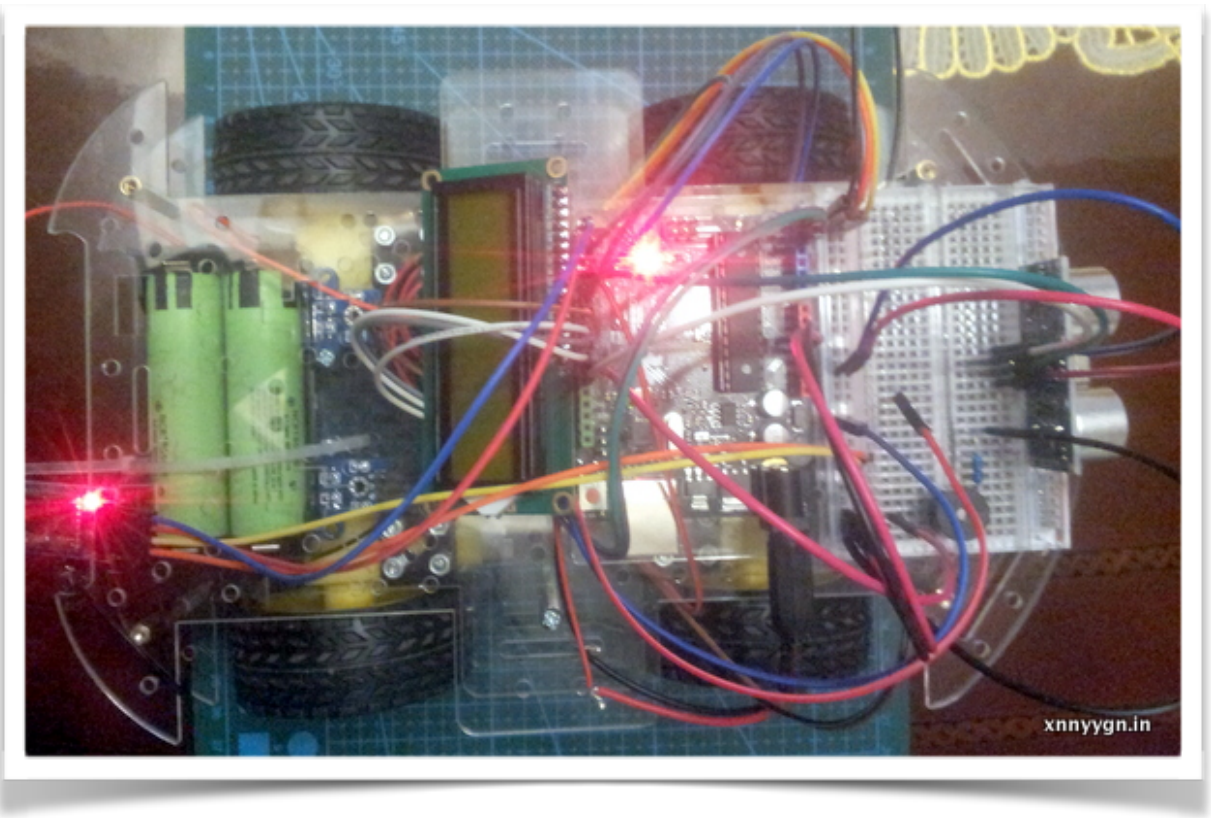


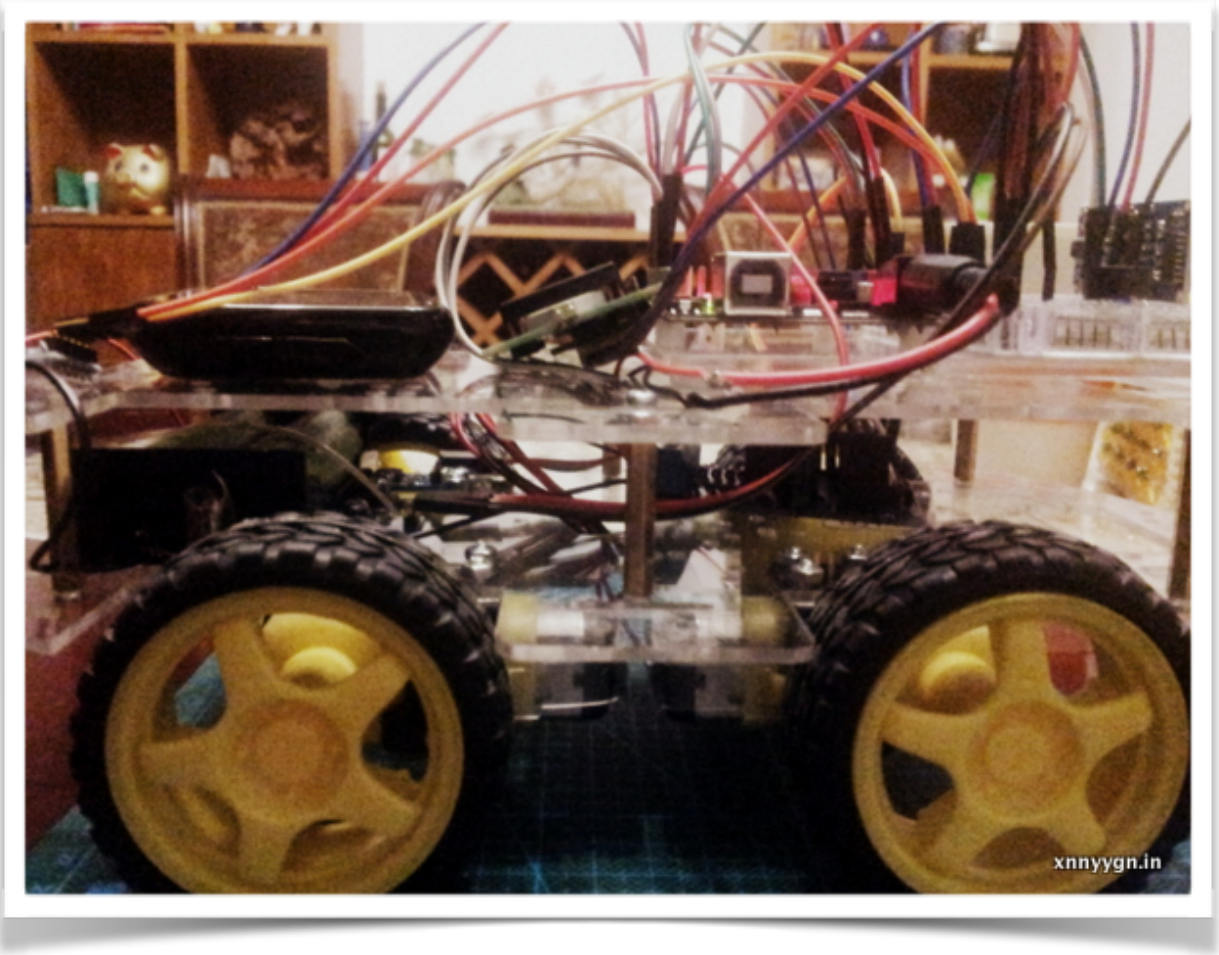
部品は多いので、乱れに見えるかもしれません。

左の速度と右の速度。普通のモーターなので、速くない。



今まで全部のツール。この作品のためといえるのも過言ではない。





車の上と右側。

2 design

2.1 features

- 前に動く
- 後ろに動く
- すぐ止める
- 左と右の違う速度で左や右を向かって動く
- 速度を測る

2.2 parts

Name	Description
Arduino UNO x 1	controller
BC04 x 1	bluetooth
L298N x 1	motor controller
basic model car set x 1	motor x 4(1:48), speed tester module x 2
18650 x 2	power, 7.4v
18650 battery box(2) x 1	
HCSR04 x 1	ultrasonic sensor
buzzer x 1	
1KR x 1	for buzzer
LCD with I2C x 1	
breadboard x 1	
switch x 1	

2.3 pin data sheet and connections

2.3.1 power solution

電源は２つの18650です。電圧は7.4Vになります。直接その電源をモーターの電源としてL298Nに輸入します。
もう一方、この電源はArduinoの電源として使われます。
L298Nのコントロールするための電源はArduinoから提供します。

2.3.4 VCC and GND

- HCSR04
- speed test module x 2
- LCD with I2C
- BC04
- L298N

2.3.3 arduino UNO D0 ~ D12

Arduino pin	part	part pin
D0(RXN)	BC04(bluetooth)	TXN
D1(TXN)	BC04(bluetooth)	RXN
D2	speed test module(left)	S
D3	speed test module(right)	S
D4	HCSR04(ultrasonic)	TRIG
D5(PWM)	L298N	M2 ENB
D6	L298N	M1 ENA
D7	L298N	M1 IN1
D8	L298N	M1 IN2
D9	L298N	M2 IN3
D10	L298N	M2 IN4
D11	buzzer	
D12	HCSR04(ultrasonic)	ECHO

2.3.3 others

- LCD with I2C SDA -> Arduino A4
- LCD with I2C SCL -> Arduino A5

2.4 arduino

コードは多いので、９つのファイルに分けます。

Tab Name(File Name)	Description
modelcar.ino	main
motor_ctrl.(hlcpp)	motor controller
serial_ctrl_server.(hlcpp)	server for serial control
speed_tester.(hlcpp)	speed measurement
hcsr04(library)	ultrasonic sneosr

```
27 // pinM11, pinM12, pinM1Pwm, pinM21, pinM22, pinM2Pwm
28 MotorCtrl motorCtrl(7, 8, 6, 9, 10, 5);
29 SerialCtrlServer serialCtrlServer;
30 HCSR04 hcsr04(4, 12); // pin trig, pin echo
31 SpeedTester speedTester;
32
33 LiquidCrystal_I2C lcd(0x27, 16, 2);
```

```
37 void setup() {
38   motorCtrl.init();
39   serialCtrlServer.init();
40   hcsr04.init();
41 
42   speedTester.init();
43   attachInterrupt(0, speedTestIncreaseRightCount, CHANGE);
44 
45   lcd.init();
46   lcd.backlight();
47 
48   Serial.begin(115200);
49   workingStatus = STATUS_STOP;
50 
51   pinMode(2, INPUT);
52   pinMode(3, INPUT);
53 }
54 
55 void loop() {
56   checkDistance();
57   serialCtrlServer.read();
58   processCmd();
59   speedTest();
60 }
```

見た通り、中心のLOOPは簡単な 4 つのコマンドだけです。

```
62 void checkDistance() {
63   float distance = hcsr04.getDistance(2320);
64   if(distance >= 0 && distance < 40) {
65     if(workingStatus == STATUS_ADVANCE) {
66       stopMe();
67       buzzWarning();
68     }
69   }
70 }
```

まずHCSR04で距離を測ります。ここにある2320は「もし2.320秒で信号は戻らないなら、すぐキャンセルします」ということです。もし指定しなかったら、前に障害物がない場合、ここは何秒以上かかります。
それに、状態を確認も必要です。後ろに戻る時、前に障害物があっても大丈夫でしょう。

```
128 void processModelCarFreeCmd() {
129   int pwm1 = serialCtrlServer.getPayloadByte(0);
130   int pwm2 = serialCtrlServer.getPayloadByte(1);
131   switch(workingStatus) {
132     case STATUS_ADVANCE:
133       motorCtrl.run(pwm1, pwm2);
134       break;
135     case STATUS_BACK:
136       motorCtrl.run(-pwm1, -pwm2);
137       break;
138   }
139 }
140 
```

実は９つのファイルに分けてると、コードはとても簡単に見えます。ここは一つの典型です。
携帯からのデータを左と右のPWM値としてコントロールに輸入することです。
モーターのコントロールは自動的に一つのPWM値を３つのコマンドに分けます。

PWM1 180 =>

digitalWrite(M1_1, HIGH);
digitalWrite(M1_2, LOW);
digitalWrite(M1_ENA, 180);

もし数値はゼロなら、止まります。ゼロ以下なら、後ろに戻るということになります。

コードはmotor_ctrl.cppにあります。

```
41 void MotorCtrl::setPinValueByPwm(int pinMx1, int pinMx2, int pinMxPwm, int pwm) {
42   if(pwm > 255) {
43     setPinValue(pinMx1, HIGH, pinMx2, LOW, pinMxPwm, 255);
44   } else if(pwm > 0) {
45     setPinValue(pinMx1, HIGH, pinMx2, LOW, pinMxPwm, pwm);
46   } else if(pwm == 0) {
47     setPinValue(pinMx1, LOW, pinMx2, LOW, pinMxPwm, 0);
48   } else if(pwm > -255) {
49     setPinValue(pinMx1, LOW, pinMx2, HIGH, pinMxPwm, -pwm);
50   } else {
51     setPinValue(pinMx1, LOW, pinMx2, HIGH, pinMxPwm, 255);
52   }
53 }
```

コマンドを処理する

```
103 void processCmd() {
104   if(serialCtrlServer.cmdAvailable()) {
105     switch(serialCtrlServer.getProtocolType()) {
106       case PROTOCOL_MODELCAR_TEST:
107         serialCtrlServer.done(SerialCtrlServer::RETURN_CODE_OK);
108         break;
109       case PROTOCOL_MODELCAR_BASIC_OP:
110         processModelCarBasicOpCmd();
111         serialCtrlServer.done(SerialCtrlServer::RETURN_CODE_OK);
112         break;
113       case PROTOCOL_MODELCAR_SPEED:
114         processModelCarSpeedCmd();
115         serialCtrlServer.done(SerialCtrlServer::RETURN_CODE_OK);
116         break;
117       case PROTOCOL_MODELCAR_FREE:
118         processModelCarFreeCmd();
119         serialCtrlServer.done(SerialCtrlServer::RETURN_CODE_OK);
120         break;
121       default:
122         serialCtrlServer.done(SerialCtrlServer::RETURN_CODE_UNKNOWN_PROTOCOL);
123         break;
124     }
125   }
126 }
127
```

プロトコルのコードを判断し、別々のコマンドを処理する仕組みです。
普通コマンドを処理する流れはこれです（serial_ctrl_server.hにあります）

```
24    public:
25
26      static const int RETURN_CODE_OK = 48;
27      static const int RETURN_CODE_UNKNOWN_PROTOCOL = 49;
28
29      void init();
30      int read();
31      bool cmdAvailable();
32      int getProtocolType();
33      int getPayloadLength();
34      byte getPayloadByte(int index);
35      void done(int returnCode);
```

setupでinit()。loopの中に
read -> cmdAvailable? -> getXxx -> done -> read …

なお、もう一つ注意するべきことがあります。serial_ctrl_server.cppでは

```
22  int SerialCtrlServer::read() {
23    int nBytes = Serial.available();
24    if(readStatus == STATUS_DONE || nBytes == 0) {
25      return 0;
26    }
27
28    for(int i = 0; i < nBytes; i++) {
```

ここavailableから戻る数値を使ってループするのは必要です。なぜならば、一番上のループ（modelcar.loop)に距離を測るコードがあります。毎回距離は測ると、時間が一秒ぐらいかかります。もし携帯から８bytesのデータが来ると、８秒以上反応できないということになります。だから、毎回データが来ったら、コマンドの最後まで読みます。

```
72  void speedTest() {
73    switch(speedTester.check(SPEED_TEST_INTERVAL)) {
74      case SpeedTester::TEST_SIDE_LEFT:
75        detachInterrupt(0);
76        attachInterrupt(1, speedTestIncreaseLeftCount, CHANGE);
77        lcd.clear();
78        lcd.print("LSV: ");
79        lcd.setCursor(0, 1);
80        lcd.print(speedTester.getLeftSpeed() * speedK);
81        lcd.print("m/s");
82        break;
83      case SpeedTester::TEST_SIDE_RIGHT:
84        attachInterrupt(0, speedTestIncreaseRightCount, CHANGE);
85        detachInterrupt(1);
86        lcd.clear();
87        lcd.print("RSV: ");
88        lcd.setCursor(0, 1);
89        lcd.print(speedTester.getRightSpeed() * speedK);
90        lcd.print("m/s");
91        break;
92    }
93  }
94
```

このコードには、スピードを両方で測ることではありません。最初はそういうつもりですが、でもArduinoは２つの中断はできませんようです。そしたら、「左一秒右一秒」ということになります。
それに、speed testerの一部のコードはspeed_tester.(hlcpp)にあるません。attachInterruptなどのコードはmodelcar.inoに置きます。一緒にspeed_tester.cppに置くのはちょっと難しいことになります。

最後はHCSR04のコード

```
18  float HCSR04::getDistance(int timeout) {
19    digitalWrite(_pinTrig, LOW);
20    delayMicroseconds(2);
21    digitalWrite(_pinTrig, HIGH);
22    delayMicroseconds(10);
23    digitalWrite(_pinTrig, LOW);
24
25    unsigned long duration = pulseIn(_pinEcho, HIGH, timeout);
26    return duration == 0 ? -1 : duration / 58.0;
27  }
```

あんまり難しくないが、ここにあるtimeoutは必要です。
実に距離を測る時、一秒以上信号が戻らないなら、一秒以上の距離があるんでしょう。
以上はArduino側のコードです。

2.5 bluetooth client

前回のRELAY-LEDのアプリの基に作られました。違うのはプロトコルとコントロールのインターフェイスです。

プロトコルには、前回言った通り通信の雑音データがあるので、今回同期するヘッドを付けました。そして、Arduino側もっと簡単に読むためにプロトコルを改正しました。

```
    @Override
    public void write(OutputStream out, DualSideSpeed payload) throws IOException {
        ProtocolUtils.writeSyncHead(out);
        out.write(getProtocolType());
        out.write(2);
        out.write(payload.getPwm1());
        out.write(payload.getPwm2());
    }

    @Override
    public Integer read(InputStream in) throws IOException {
        return ProtocolUtils.readWithTimeout(in);
    }
```

```
public static void writeSyncHead(OutputStream out) throws IOException {
    for(int i = ProtocolConstants.SYNC_HEAD_LEN - 1; i > 0; i--) {
        out.write(ProtocolConstants.SYNC_HEAD_BODY);
    }
    out.write(ProtocolConstants.SYNC_HEAD_END);
}
```

SYNC_HEAD_LEN今回5を設定しました。実はArduinoはいくら同期するデータがいいというコードを設計したので、具体的に幾つを設定するのが大丈夫です。調整するとき、雑音データは3byteぐらいですので、同期データは3以上設定しました。

全部加えて、典型は

SEND: [4: SYNC_HEAD_BOY][1:SYNC_HEAD_END][1:PROTOCOL_CODE]
[1:PAYLOAD_LENGTH][n:PAYLOAD]

RECEIVE: [1:RETURN_CODE]

Arduino側のコード

```
int incomingByte = Serial.read();
if(readStatus == STATUS_START && incomingByte == SYNC_BODY) {
  readStatus = STATUS_SYNC_BODY;
}else if(readStatus == STATUS_SYNC_BODY && incomingByte == SYNC_END) {
  readStatus = STATUS_SYNC_END;
}else if(readStatus == STATUS_SYNC_END) {
  protocolType = incomingByte;
  readStatus = STATUS_PROTOCOL;
}else if(readStatus == STATUS_PROTOCOL) {
  payloadLength = incomingByte;
  readStatus = STATUS_PAYLOAD_LENGTH;
}else if(readStatus == STATUS_PAYLOAD_LENGTH) {
  if(currentReadCount > PAYLOAD_BUFFER_SIZE) {
    readStatus = STATUS_DONE;
    break;
  }else {
    if(currentReadCount < payloadLength) {
      payloadBuffer[currentReadCount++] = incomingByte;
    }
    if(currentReadCount == payloadLength) {
      readStatus = STATUS_DONE;
      break;
    }
  }
}
```

状態の変更

STATUS_START -> STATUS_SYNC_BODY -> STATUS_SYNC_END -> STATUS_PROTOCOL
-> STATUS_PAYLOAD_LENGTH -> STATUS_DONE

インターフェイス

```
@Override
public void onStopTrackingTouch(SeekBar seekBar) {
    int value = seekBar.getProgress();
    if (value < 255) {
        mActivity.send(mDirectionProtocol, new DualSideSpeed(Math.max(value, 30), 255));
    } else {
        mActivity.send(mDirectionProtocol, new DualSideSpeed(255, Math.max(seekBar.getMax() - value, 30)));
    }
}
```

車モデルを左や右に向かって動かすのは左と右の速度を調整するのは必要です。もし左に向かうら、左０、右255（最大）はどうですかね。実際できないです。車は止まります。でも、10とか、20とか、それなら大丈夫です。なぜ左の電流（0）は右に影響するのはちょっとわからないですね。でももしあったら、ゼロ以上に設定すれば大丈夫です。

2.6 problems

製品から見ると、ちょっと簡単かもしれません。それはそんですね、前に障害物があったら、止めるだけです。もし自動的に後ろに戻て、別の方向へ向かえばかなりいいでしょう。でも、子供から遠隔操作で車モデルを動かすのが憧れる私にとって、今のは充分でしょう。
でも、これを作るのは全部順調ではないです。部品から組み立てたり、プログラムを調整したりするのは３日間経ちました。もし携帯のアプリの開発も含めて、一週間になるんでしょう。幸いアプリは前回の作品（RELAY-LED、５日間かかった）でだいぶ開発しました、今回はただ一部を改正するだけで済みます。
下のは作品を作る間あった問題です。参考になれば嬉しいですが。

2.6.1 Arduinoのコードはアップロードできない

ArduinoのD0(RXN), D1(TXN)はアップロードする時BC04のTXN, RXNを連携しないでください。BC04だけではなく、他の部品もです。

なお、BC04はBLUETOOTHの部品ですが、ただBC04で遠隔でコードはアップロードできません。

2.6.2 モーターは動かない

USBはただArduinoに電力を提供します。モーターの電源は別です。
USBの電力でモーターを動けますか? たぶんできます。でもとても遅いかもしれません。
今回は18650 x 2（7.4v）の電源でもそんなに速くありません。

2.6.2 モーターは動いてすぐ停まる

電圧は低い可能性があります。一日で私の18650の電圧は4.2vから3.3vになりました。これは3200mAhなのよ。
解決の方法もう一つがあります。両方は255で出力ではない、200以下に調整してみてください。

2.6.3 車は反応ない

コードのせいかもしれません。
Arduinoは直接デバックできないので、エンジニアの能力を発揮して問題を探そう。
Arduino UNOは一つのserial portしかありませんので、megaなら２つのserial portでデバックできるかもね。私がBLUETOOTHという通信手段を選んで、今回の作品でいろんな不便さがありますでしょう。もしRaspberry Piの無線WIFIなら、コードのアップロードも余裕でしょう。でも、Raspberry Piは直接のPWMはない。そして、どうやったら中断で速度を測るのが調べる必要があります。
最後、私はHCSR04で距離を測るとそのあと通信のコードの組み合わせから問題を見つかりました。

2.6.4 LCDの字が暗くなる

電圧を測るべきです。そして、LCDのbacklightを消してください。

2.6.5 L298N音がある

入力の電流でモーターを動かないです。ゼロではない、30ぐらい試してください。

2.6.6 前に障害物がないのに、突然距離が出る

一つは電圧は低い。
もう一つは前に向かう時電流の雑音。これはちょっと調べことはない。器材などは必要です。
解決方法はたぶんHCSR04を独立の電路に設置するかも。
これは常に出てるわけではないので、もっと調べる必要があります。