電子工作 RELAY LED

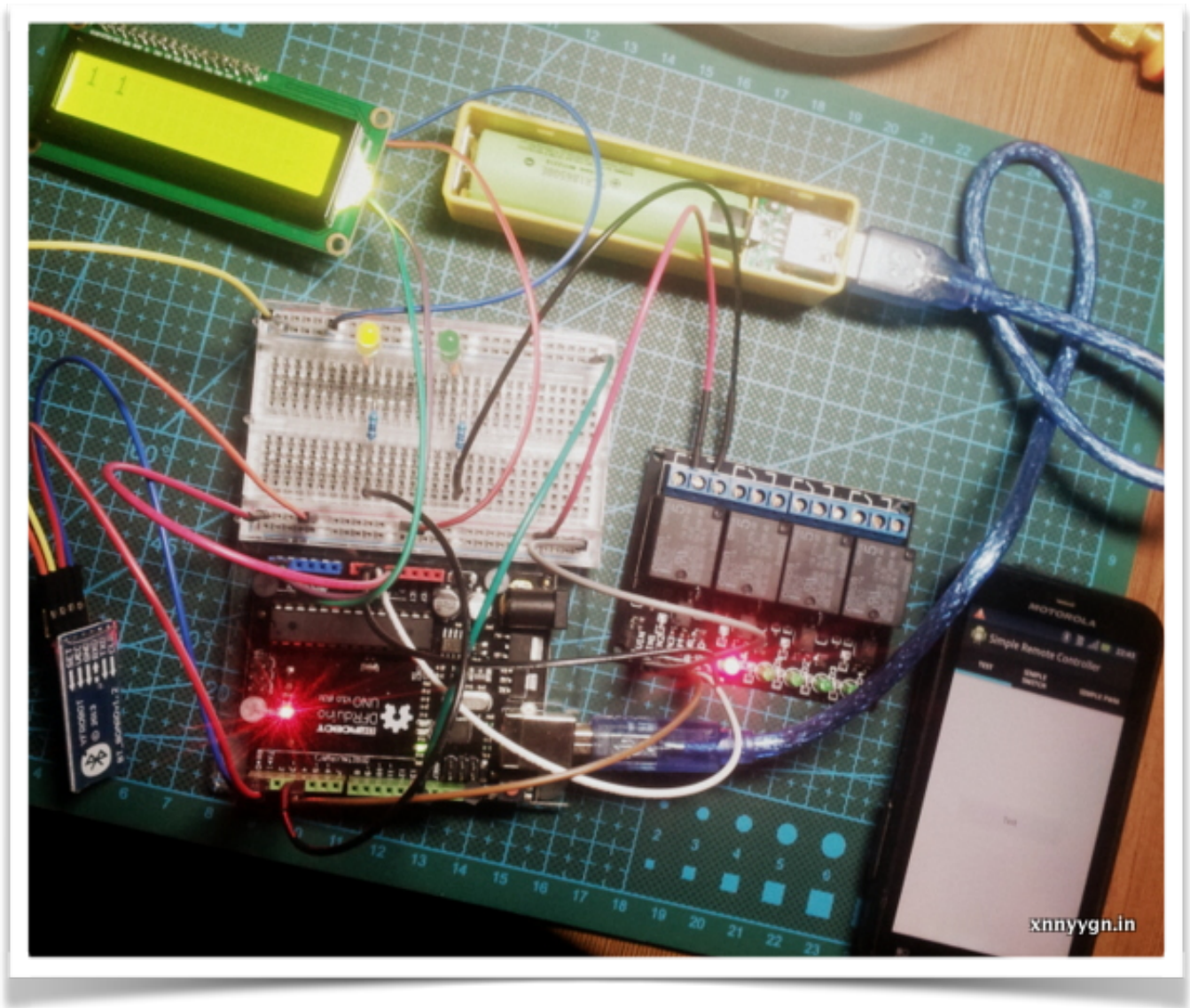
XnnYygn 2014-11-17

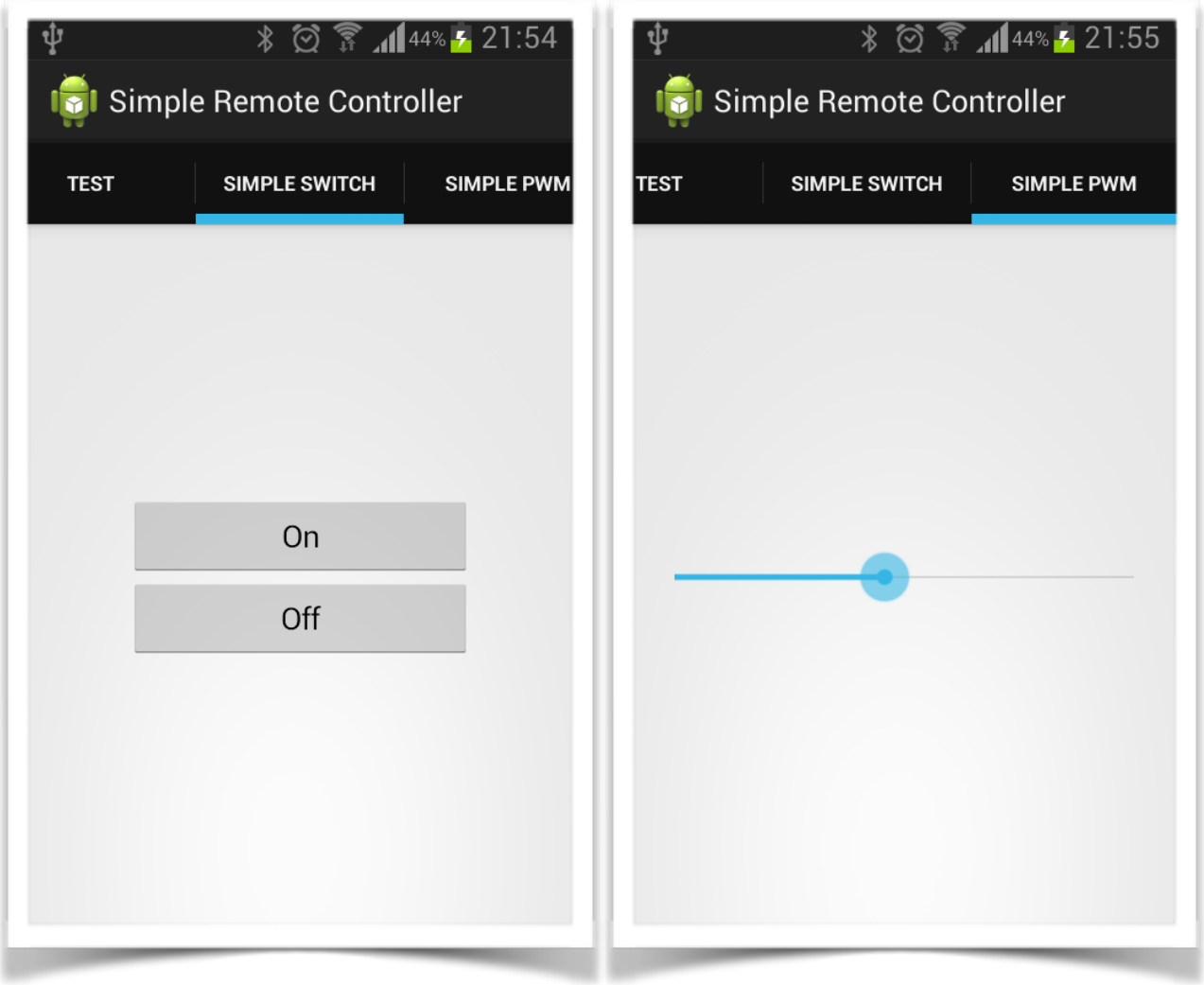
1. overview

1.1 keyword

relay, bluetooth, bc04, android, pwm, 18650, protocol

1.2 presentation



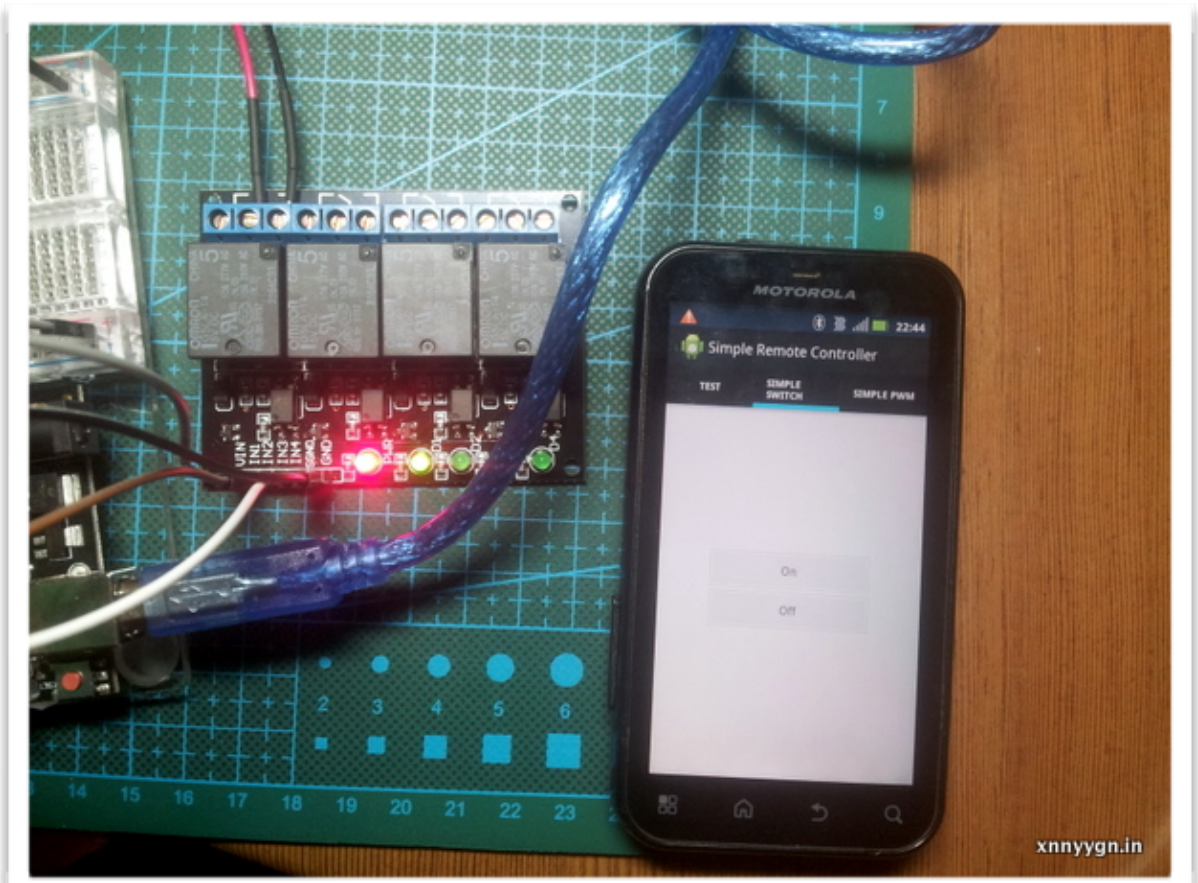


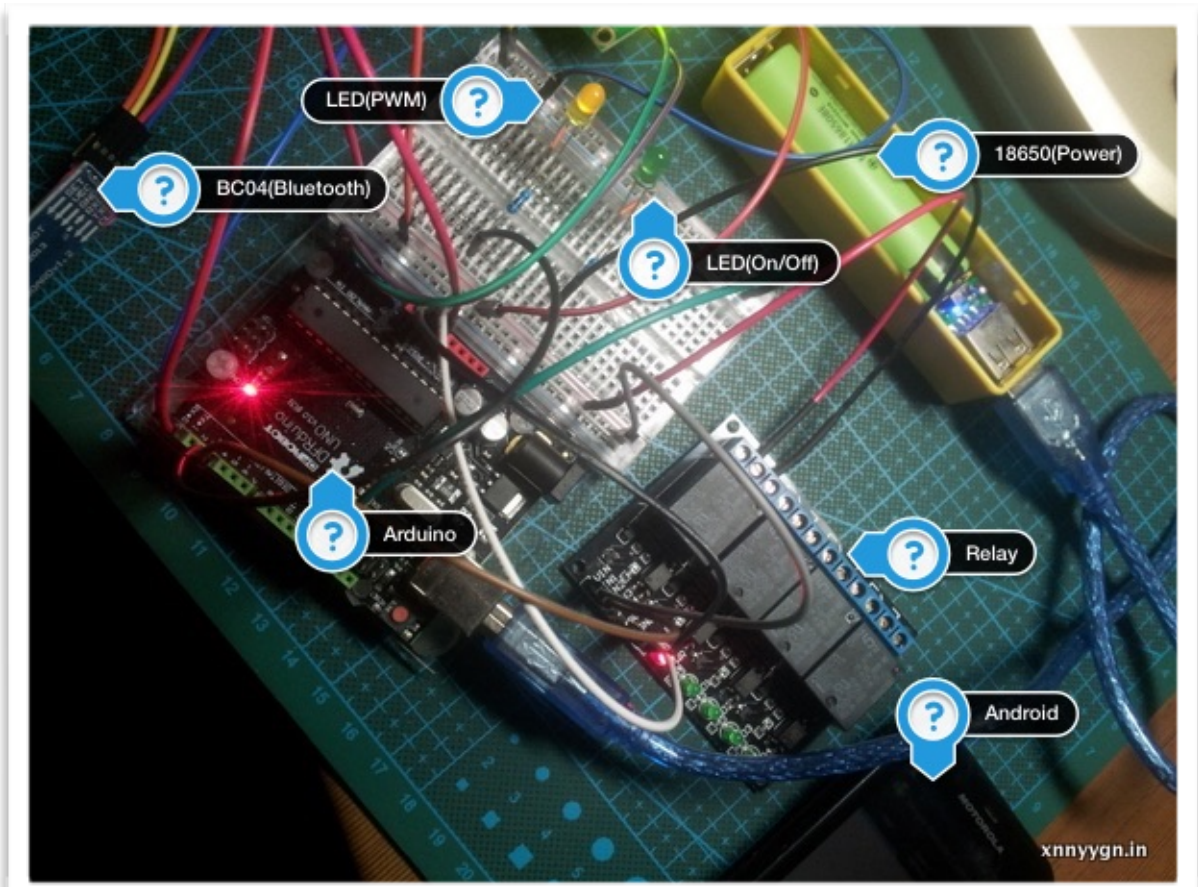
1.3 how it works

まずBLUETOOTHで作品を接続して、コントロールのインターフエイスでLEDをつけるかとうかを操作できる。それに、LEDの明さも調整できる。

実は作品は100V以上の高圧の電源も操作できるが、安全のため今回はただ普通のLEDを試した。

1.4 pictures


MOTOROLA
22:44
Simple Remote Controller
TEST
SIMPLE SWITCH
SIMPLE PWM
On
Off
xnnyygn.in

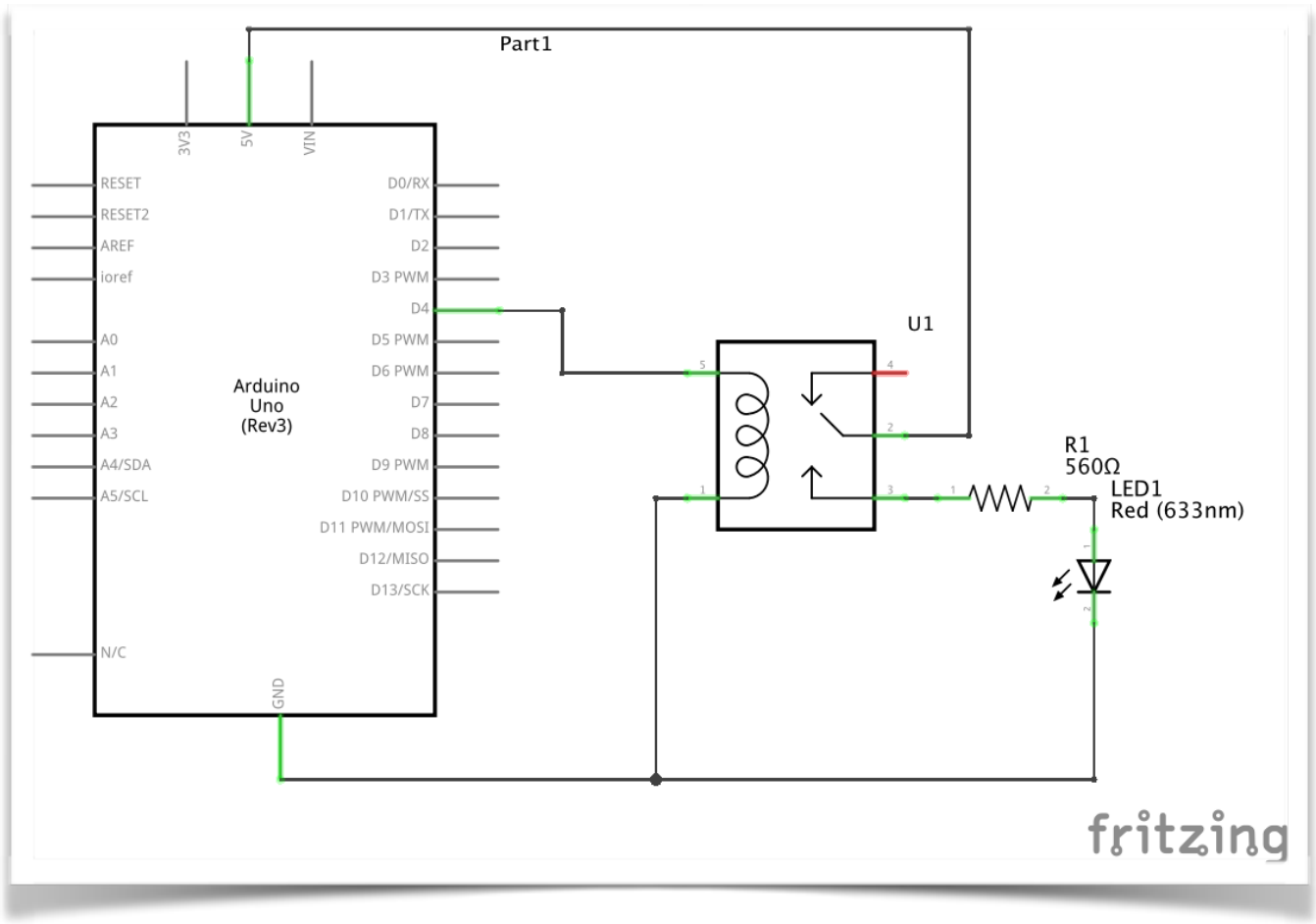

LED(PWM)
BC04(Bluetooth)
18650(Power)
LED(On/Off)
Arduino
Relay
Android
xnnyygn.in

2. design

2.1 parts

Name	Description
Arduino UNO x 1	controller
Relay(OMORO) x 1	MCU: 5v
BC04 x 1	bluetooth
LED x 2	
Resistance 560R x 2	
18650 power	printer cable is OK
LCD with I2C x 1	for debug
Android Phone	for remote controller

2.2 schematic



2.3 pin data sheet and connection

Part	Pin From	Pin To(Arduino)
Relay	IN1	D4
	VCC	5V
	GND	GND
	SWITCH MIDDLE	5V
	SWITCH RIGHT	(560R)
LED(PWM)	(+)	D3(PWM)
	(-)	GND
BC04	VCC	5V
	GND	GND
	TXN	RXN(yes, it's correct)
	RXN	TXN(yes, it's correct)
LCD with I2C	SDA	A4
	SCL	A5
	VCC	5V
	GND	GND

もう一度いう、BC04のTXNとArduinoのRXN(D0)を接続するのはまちがいないです。

2.4 simple serial protocol

2.4.1 overview

Protocol	Request	Response
TEST	[1:LENGTH][1:PROTOCOL_CODE]	[1:LENGTH][1:RETURN_CODE]
SIMPLE_SWITCH	[1:LENGTH][1:PROTOCOL_CODE] [1:SWITCH_VALUE]	[1:LENGTH][1:RETURN_CODE]
SIMPLE_PWM	[1:LENGTH][1:PROTOCOL_CODE] [1:PWM_VALUE]	[1:LENGTH][1:RETURN_CODE]

[1:LENGTH]の意味はフィールドの名前はLENGTH、長さは一（byte）。

2.4.1 protocol code

Protocol	Code
TEST	1
SIMPLE_SWITCH	2
SIMPLE_PWM	3

2.5 program

2.5.1 android bluetooth api

Description	Code
test if device support bluetooth	BluetoothAdapter.getDefaultAdapter() != null

scan device	IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
	registerReceiver(mDeviceReceiver, filter);
	mBluetoothAdapter.startDiscovery();

stop discovery	mBluetoothAdapter.cancelDiscovery();
UUID	private static final UUID sUUID = UUID.fromString(“00001101-0000-1000-8000-00805F9B34FB");

connect to device	mBluetoothDevice = bluetoothAdapter.getRemoteDevice(address);
	mBluetoothSocket = mBluetoothDevice.createRfcommSocketToServiceRecord(sUUID);
	mBluetoothSocket.connect();

2.5.2 IO protocol

```
public interface IoProtocol<T1, T2 extends Serializable> {

    /**...*/
    int getProtocolType();

    /**...*/
    void write(OutputStream out, T1 payload) throws IOException;

    /**...*/
    T2 read(InputStream in) throws IOException;

}
```

with bluetooth socket

```
Log.d(TAG, "write data, protocol " + protocol.getProtocolType() + ", payload " + payload);
protocol.write(mBluetoothSocket.getOutputStream(), payload);
Log.d(TAG, "read data");
T2 result = protocol.read(mBluetoothSocket.getInputStream());
```

SIMPLE SWITCH protocol implementation, SingleIntegerProtocol

```java
public class SingleIntegerProtocol implements IoProtocol<Integer, Integer>{

    private final int protocolType;

    public SingleIntegerProtocol(int protocolType) { this.protocolType = protocolType; }

    @Override
    public int getProtocolType() { return protocolType; }

    /**...*/
    @Override
    public void write(OutputStream out, Integer payload) throws IOException {
        /* ProtocolUtils.writeSyncHead(out);
        out.write(protocolType);
        out.write(1);
        out.write(payload);
        out.flush(); */
        out.write(2);
        out.write(getProtocolType());
        out.write(payload);
        out.flush();
    }

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

}
```

コメントにあるのはつぎの車モデルのプロトコルです。
InputStreamは直接のreadWithTimeoutはないので、自分で簡単の方を実現しました。

2.5.3 arduino serial server

```
if(Serial.available()) {
  if(workingStatus == STATUS_WAIT_LEN) {
    dataLength = Serial.read();
    lcd.clear();
    lcd.print(dataLength);
    lcd.print(' ');
    workingStatus = STATUS_WAIT_PT;
  } else if(workingStatus == STATUS_WAIT_PT) {
    protocolType = Serial.read();
    lcd.print(protocolType);
    lcd.print(' ');
    if(dataLength > 1) {
      workingStatus = STATUS_WAIT_PL;
    } else {
      workingStatus = STATUS_DONE;
    }
  } else if(workingStatus == STATUS_WAIT_PL) {
    payload = Serial.read();
    lcd.print(payload);
    workingStatus = STATUS_DONE;
  }
}
```

arduinoは状態変更でサービスを実現した。４つの状態です。

#define STATUS_WAIT_LEN 0
#define STATUS_WAIT_PT 1 // PROTOCOL
#define STATUS_WAIT_PL 2 // PAYLOAD
#define STATUS_DONE 3

つぎはプロトコルコードを判断し、LEDを操作する。

```
if(workingStatus == STATUS_DONE) {
  switch(protocolType){
    case PT_TEST:
      replyCode(48);
      break;
    case PT_SIMPLE_SWITCH:
      digitalWrite(PIN_RELAY, payload == 1 ? HIGH : LOW);
      replyCode(48);
      break;
    case PT_SIMPLE_PWM:
      if(payload < 0) payload = 0;
      if(payload > 255) payload = 255;
      analogWrite(PIN_PWM_LED, payload);
      replyCode(48);
      break;
  }
  workingStatus = STATUS_WAIT_LEN;
}
```

2.6 others

2.6.1 通信の雑音

デバイスを接続したりするのは、意外にデータが出て来る。それは有効なコマンドではないので、作品はうまく処理できない。でも、それ以外は思う通り動かせる。
この通信の雑音について、実はもう一つのプロトコルを開発しました。次回の車モデルに使われる。その原理はデータの前に同期するためデータをつけることだ。

2.6.2 RELAYの音

実はビデオはあるが、でもうまく取れないので、今回はなしにしよう。ビデオで、携帯のボタンを押すとすぐRELAYの音は出て来るので、まるで、携帯のボタンの音だ。

2.6.3 18650と電源

18650は普通のバッテリーよりずっと大きい、そして、電圧もだ。18650のは3.7V。実は18650は車モデルのため買ったものだ。作品でただパソコンでLEDを操作するのを誤解されないように使われる。

2.6.4 同じな製品

実は携帯でライトをコントロールする製品はあります。原理はたぶんこの作品と同じです。

2.6.5 BC04のbitrate

コードにあるserailのbitrateは115200です。BC04の本来のbitrateは9600です。ちょっとパソコンでATコマンドで変えた。なぜならば、USBのbitrate(9600)と区別したいのです。