```
//SR20SN0047-BtWifi_ver1.0  //20230331
//SR20SN0046-BtWifi_ver1.0  //20230331
//SR20SN0045-BtWifi_ver1.0  //20230331
//SR20SN0044-BtWifi_ver1.0  //20230331
//SR20SN0042-BtWifi_ver1.0  //20230119
//SR20SN0043-BtWifi_ver1.0  //20221025

//コントローラごとに割り振る
#define BtName "SR20BT-047"
const char wifiAp_ssid[] = "SR20WiFi-047";  // SSID
const char wifiAp_pass[] = "sr20wifi-047";  // password

/*
 ESP-EROOM-32D
 フラッシュメモリ：4MB(32Mb)
 コンパイル時メモリ不足が発生するのでツールで"Huge APP (3MB No OTA/1MB SPIFFS)"を選択
 環境設定でより省ｇサイナ詳細な情報を表示する　コンパイルにチェック
 AP mode
 */
/*
 *               39: GND4
 *  1: GND1                                  38: GND3
 *  2: 3V3                                   37: IO23
 *  3: EN                                    36: IO22
 *  4: SENSOR_NP(in)  -----|                 35: TXD0
 *  5: SENSOR_VN(in)  -----|                 34: RXD0
 *  6: IO34(in)       -----|                 33: IO21
 *  7: IO35(in)       -----|                 32: NC
 *  8: IO32(out)      -----|                 31: IO19
 *  9: IO33                                  30: IO18
 *  10: IO25   LED_G(out)                    29: IO5
 *  11: IO26   LED_R(out)                    28: IO17  TXD2
 *  12: IO27                                 27: IO16  RXD2
 *  13: IO14                                 26: IO4
 *  14: IO12                                 25: IO0
 *  15:GND2 16:IO13 17:SD2 18:SD3 19:CMD 20:CLK 21:SD0 22:SD1 23:IO15 24:IO2
 *
 *
 */

#include <uTimerLib.h> // interrupt timer module

#include"BluetoothSerial.h"
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

//Wifiap mode
#include <WiFi.h>

//#include <EEPROM.h>
//#include "SPIFFS.h"

//Wifista mode
#include <WebServer.h>
//#include <DNSServer.h>// x20220826
#include "SPIFFS.h"

void intervalTimer(void);
void setup_bt(void);
void loop_bt(void);
void setup_wifiAp(void);
void loop_wifiAp(void);
void setup_wifiSta(void);
void loop_wifiSta(void);
void readConfigFile(void);
void webconfig(void);

BluetoothSerial SerialBT;//
WiFiServerserver(80);//WebサーバのHTTPならポート番号は80
```

```
//WiFiServerserver(50000);//WebサーバのHTTPならポート番号は80

char c1 = 0x00, c2 = 0x00, comm_mod = 0x00;

const int ledR_pin = 25;
const int ledG_pin = 26;
int ledR = HIGH, ledG = HIGH;
int ledR_OffOnBlk = 0, ledG_OffOnBlk = 0;
int ledRIntervalCnt = 0, ledGIntervalCnt = 0;
int serial2TxComCnt = 0;

void intervalTimer() {//01s interval
  //ledR_OffOnBlk = 3;//test
  switch(ledR_OffOnBlk){
    case 0:
      break;
    case 1://off
      digitalWrite(ledR_pin, HIGH);
      break;
    case 2://on
      digitalWrite(ledR_pin, LOW);
      break;
    case 3://blink 2s
   ledRIntervalCnt++;
   if(20<=ledRIntervalCnt){
     ledRIntervalCnt = 0;
     if(ledR == HIGH){ ledR = LOW; }
     else{             ledR = HIGH;}
     digitalWrite(ledR_pin, ledR);  // LEDの点灯／消灯を行う
    }
   break;
  case 4://blink 1s
   ledRIntervalCnt++;
   if(10<=ledRIntervalCnt){
     ledRIntervalCnt = 0;
     if(ledR == HIGH){ ledR = LOW; }
     else{             ledR = HIGH;}
     digitalWrite(ledR_pin, ledR);  // LEDの点灯／消灯を行う
    }
   break;
  case 5://blink 0.5s
   ledRIntervalCnt++;
   if(5<=ledRIntervalCnt){
     ledRIntervalCnt = 0;
     if(ledR == HIGH){ ledR = LOW; }
     else{             ledR = HIGH;}
     digitalWrite(ledR_pin, ledR);  // LEDの点灯／消灯を行う
    }
    break;
  case 6://blink 0.3s
   ledRIntervalCnt++;
   if(3<=ledRIntervalCnt){
      ledRIntervalCnt = 0;
      if(ledR == HIGH){ ledR = LOW; }
      else{             ledR = HIGH;}
      digitalWrite(ledR_pin, ledR);  // LEDの点灯／消灯を行う
    }
    break;
  case 7://blink 0.1s
    ledRIntervalCnt++;
    if(1<=ledRIntervalCnt){
      ledRIntervalCnt = 0;
      if(ledR == HIGH){ ledR = LOW; }
      else{             ledR = HIGH;}
      digitalWrite(ledR_pin, ledR);  // LEDの点灯／消灯を行う
    }
    break;
  default:
    ledR_OffOnBlk = 0;
}
```

```
//LED_G
//ledG_OffOnBlk = 4;//test // 1:off 2:on 3:blk2s 4:blk1s 5:blk0.5s 6:blk0.3 7:blk0.1s
switch(ledG_OffOnBlk){
  case 0:
    break;
  case 1://off
    digitalWrite(ledG_pin, HIGH);
    break;
  case 2://on
    digitalWrite(ledG_pin, LOW);
    break;
  case 3://blink 2s
    ledGIntervalCnt++;
    if(20<=ledGIntervalCnt){
      ledGIntervalCnt = 0;
      if(ledG == HIGH){ ledG = LOW; }
      else{             ledG = HIGH;}
      digitalWrite(ledG_pin, ledG);  // LEDの点灯／消灯を行う
    }
    break;
  case 4://blink 1s
    ledGIntervalCnt++;
    if(10<=ledGIntervalCnt){
      ledGIntervalCnt = 0;
      if(ledG == HIGH){ ledG = LOW; }
      else{             ledG = HIGH;}
      digitalWrite(ledG_pin, ledG);  // LEDの点灯／消灯を行う
    }
    break;
case 5://blink 0.5s
  ledGIntervalCnt++;
  if(5<=ledGIntervalCnt){
    ledGIntervalCnt = 0;
    if(ledG == HIGH){ ledG = LOW; }
    else{             ledG = HIGH;}
    digitalWrite(ledG_pin, ledG);  // LEDの点灯／消灯を行う
  }
  break;
case 6://blink 0.3s
 ledGIntervalCnt++;
 if(3<=ledGIntervalCnt){
    ledGIntervalCnt = 0;
    if(ledG == HIGH){ ledG = LOW; }
    else{             ledG = HIGH;}
    digitalWrite(ledG_pin, ledG);  // LEDの点灯／消灯を行う
  }
    break;
  case 7://blink 0.1s
    ledGIntervalCnt++;
    if(1<=ledGIntervalCnt){
      ledGIntervalCnt = 0;
      if(ledG == HIGH){ ledG = LOW; }
      else{             ledG = HIGH;}
      digitalWrite(ledG_pin, ledG);  // LEDの点灯／消灯を行う
    }
    break;
  default:
    ledG_OffOnBlk = 0;
 }
 if( 0 < serial2TxComCnt ){
  serial2TxComCnt--;
 }
}

int serial2NonSignalCnt=0;
void setup()
{
  delay(100);//100ms
  //Serial.begin(115200);
```

```
 Serial.begin(9600);
Serial.println("SETUP0123456789123456");

 Serial2.begin(9600);
  pinMode(ledR_pin, OUTPUT);     // set the LED_R pin mode
  pinMode(ledG_pin, OUTPUT);     // set the LED_G pin mode
  delay(10);
Serial.println("SETUP0123456789");

 digitalWrite(ledR_pin, HIGH);//LED_R(25) OFF
 digitalWrite(ledG_pin, HIGH);//LED_G(26) OFF

 TimerLib.setInterval_us(intervalTimer, 100000); // 0.1秒ごとにintervalTimer関数を実行する

  ledG_OffOnBlk = 7;// 1:off 2:on 3:blk2s 4:blk1s 5:blk0.5s 6:blk0.3 7:blk0.1s
  do{
 delay(10);//10ms
 if (Serial2.available()) { // if there's bytes to read from the client,
   c2 = Serial2.read(); // read a byte, then
   Serial.write(c2); // print it out the serial monitor
   switch(c2){ //com_mode 設定
   default:// SR20 P51=0000 の時  最初に'SunRay'が入ってくる。
       comm_mod = 0x30;
      Serial.println("default");
 case 0x30://"0" //Bluetooth(RN42-I/RM)
      comm_mod = 0x30;
     Serial.println("Bluetooth0");
     ledG_OffOnBlk = 2;// 1:off 2:on 3:blk2s 4:blk1s 5:blk0.5s 6:blk0.3 7:blk0.1s
      setup_bt();
      break;
 case 0x31://"1" //Bluetooth(ESP32)
      comm_mod = 0x31;
     Serial.println("Bluetooth1");
     Serial2.print("1");
     ledG_OffOnBlk = 2;// 1:off 2:on 3:blk2s 4:blk1s 5:blk0.5s 6:blk0.3 7:blk0.1s
      setup_bt();
      break;
 case 0x32://"2" //WiFiAp
      comm_mod = 0x32;
     Serial.println("WiFiAp2");
     Serial2.print("2");
     ledG_OffOnBlk = 4;// 1:off 2:on 3:blk2s 4:blk1s 5:blk0.5s 6:blk0.3 7:blk0.1s
      setup_wifiAp();
    break;
 case 0x33://"3" //WiFiSta
      comm_mod = 0x33;
     Serial.println("WiFiSta3");
         Serial2.print("3");
         ledG_OffOnBlk = 5;// 1:off 2:on 3:blk2s 4:blk1s 5:blk0.5s 6:blk0.3 7:blk0.1s
          setup_wifiSta();
        break;
      };
   }//if (Serial2.available()) {
   else{
    //Serial.println("serial2NonSignalCnt++");
     serial2NonSignalCnt++;
     if( 200<serial2NonSignalCnt ){//信号が2秒間来ないときbluetoothとして
       Serial.println("TimeOut:Bluetooth");
        comm_mod = 0x30;
       Serial.println("Bluetooth0");
       ledG_OffOnBlk = 2;// 1:off 2:on 3:blk2s 4:blk1s 5:blk0.5s 6:blk0.3 7:blk0.1s
        setup_bt();
      }
    }
  }while(!comm_mod);// comm_mod が設定されるまでLoop
}

char i;
void loop(){
     switch(comm_mod){//com mode
```

```
case 0x30://"0" //Bluetooth(RN42-I/RM)
    Serial.print(comm_mod);
    Serial.println("Bluetooth");
     loop_bt();
     break;
case 0x31://"1" //Bluetooth(ESP32)
    Serial.print(comm_mod);
    Serial.println("Bluetooth");
     loop_bt();
     break;
case 0x32://"2" //WiFiAp
    Serial.print(comm_mod);
    Serial.println("WiFiAp");
     loop_wifiAp();
     break;
case 0x33://"3" //WiFiSta
    Serial.print(comm_mod);
    Serial.println("WiFiSta");
     loop_wifiSta();
     break;
default:
     break;
};
}

//#define BtName "SR20BT-01" //先頭で設定  機種ごとに変わる
void setup_bt(){ //BlueTooth
   SerialBT.begin(BtName); //Bluetooth device name
   Serial.println(BtName);
   Serial.println("The device started, now you can pair it with bluetooth!");
}

void loop_bt(){
  while(1){
   //if (SerialBT.available()) {
   while (SerialBT.available()) {
     c1 = SerialBT.read();
     Serial.write(c1);// print it out the serial monitor
     Serial2.write(c1);// to SR20
    }
   //if (Serial2.available()) {
   while (Serial2.available()) {
     c2 = Serial2.read();// from SR20
     Serial.write(c2); // print it out the serial monitor
     SerialBT.write(c2);
     serial2TxComCnt = 10;//1s in ( void intervalTimer() {//01s interval )
    }
   if(c2=='#'){ Serial.println(); c2 = 0; }
   if(serial2TxComCnt){// in void intervalTimer()  (serial2TxComCnt--)
     ledR_OffOnBlk = 7;// 1:off 2:on 3:blk2s 4:blk1s 5:blk0.5s 6:blk0.3 7:blk0.1s
    }
    else{
     ledR_OffOnBlk = 1;// 1:off 2:on 3:blk2s 4:blk1s 5:blk0.5s 6:blk0.3 7:blk0.1s
    }
  }
}

//#define WifiApSsid "SR20WiFi-01" //先頭で設定  機種ごとに変わる //#define で設定できない？
//#define WifiApPass "sr20wifi-16"  //先頭で設定  機種ごとに変わる
void setup_wifiAp(){
  /*
    const char ssid[] = "SR20WiFi-16";  // SSID
    //const char ssid[] = WifiApSsid;  // SSID //#define で設定できない？
    const char pass[] = "sr20wifi-16";  // password
    //const char pass[] = WifiApPass;  // password //#define で設定できない？
    const IPAddress ip(10, 0, 0, 1);      // IPアドレス
    const IPAddress subnet(255, 255, 255, 0); // サブネットマスク

    WiFi.softAP(ssid, pass);          // SSIDとパスの設定
  */
```

```
 //const char wifiAp_ssid[] = "SR20WiFi-16";  // SSID  //先頭で設定
 //const char wifiAp_pass[] = "sr20wifi-16";  // password  //先頭で設定
 const IPAddress ip(10, 0, 0, 1);      // IPアドレス
 const IPAddress subnet(255, 255, 255, 0); // サブネットマスク

WiFi.disconnect();
 WiFi.mode(WIFI_AP);             //ESP32をAPモードにします

 //アクセスポイントを起動する
 WiFi.softAP(wifiAp_ssid, wifiAp_pass);          // SSIDとパスの設定

 delay(100);                       // 追記：このdelayを入れないと失敗する場合がある
WiFi.softAPConfig(ip, ip, subnet); // IPアドレス、ゲートウェイ、サブネットマスクの設定

 IPAddress sr20IP = WiFi.softAPIP();  // WiFi.softAPIP()でWiFi起動
 delay(10);
 /* Start Web Server server */
 server.begin();                   // サーバーを起動(htmlを表示させるため)

 // 各種情報を表示
Serial.print("WifiAp_SSID: ");
Serial.println(wifiAp_ssid);
 Serial.print("WifiAP IP address: ");
   Serial.println(sr20IP);
   Serial.println("Server start!");
    //while(1){ }
}

void loop_wifiAp(){
  while(1){
    WiFiClient client = server.available();   // listen for incoming clients
    //if (client) {                             // if you get a client,
    while (client) {                            // if you get a client,
      //Serial.println("New Client.");          // print a message out the serial port
      while (client.connected()) {           // loop while the client's connected
        //clientStopCnt = 1;
//        if (client.available()) {            // if there's bytes to read from the client,
//          String line = client.readStringUntil('\n');          // Get Line data until '\n'
//          Serial.print(line);
//          Serial2.print(line);
//         }

        //if (client.available()) {            // if there's bytes to read from the client,
        while (client.available()) {            // これが早い！！ これはだめ  最初にSC、、、をskysafariからおくってこない
          c1 = client.read();             // read a byte, then
          Serial.write(c1);                    // print it out the serial monitor
          Serial2.write(c1);                   // to SR20
     }

     //if (Serial2.available()) {            // if there's bytes to read from the Serial2,
     while (Serial2.available()) {
       c2 = Serial2.read();             // read a byte, then
       Serial.write(c2); // print it out the serial monitor
        client.write(c2);                   // from SR20
       serial2TxComCnt = 10;//1s in ( void intervalTimer() {//01s interval )
     }
     if(c2=='#'){ Serial.println(); c2 = 0; }
   }
   /*
   // close the connection:
   client.stop();
  Serial.println("Client Disconnected.");
   */
 }

if(serial2TxComCnt){// in void intervalTimer()  (serial2TxComCnt--)
  ledR_OffOnBlk = 7;// 1:off 2:on 3:blk2s 4:blk1s 5:blk0.5s 6:blk0.3 7:blk0.1s
 }
 else{
  ledR_OffOnBlk = 1;// 1:off 2:on 3:blk2s 4:blk1s 5:blk0.5s 6:blk0.3 7:blk0.1s
```

```
    }
  }
}


//Wifista mode
const IPAddress apIP(192,168,1,100);
//const IPAddress subnet(255, 255, 255, 0); // サブネットマスク
WebServer webServer(80);//ポート番号は80
//WebServerwebServer(50000);//ポート番号は80
//const char* WIFIMGR_ssid = "WIFIMGR_ESP32";

//const char* WIFIMGR_ssid = "WiFiManager_forSR20";//ver2
const char WIFIMGR_ssid[] = "WiFiManager_forSR20";//ver2
//const char* WIFIMGR_pass = "xxxxxxxx";
const char WIFIMGR_pass[] = "xxxxxxxx";

//DNSServerdnsServer;//x20220826

//const char* ssid     = "106F3FD9B204";
//const char* password = "sxg9vv3dxawf8";
// parameters setting
const String defaultSSID = "myssid";
const String defaultPASSWD = "12345678";
String ssid = defaultSSID;
String passwd = defaultPASSWD;

// scan SSID
#define SSIDLIMIT 30
Stringssid_rssi_str[SSIDLIMIT];
Stringssid_str[SSIDLIMIT];

// SPIFFS config filename
const char* configfile = "/contig.txt";

void setup_wifiSta(){

  //   init filesystem
 if(!SPIFFS.begin(true)){
     Serial.println("SPIFFS Mount Failed");
  }
  // read config from SPIFFS
 Serial.println("read config from SPIFFS");
 readConfigFile();

// Serial.println(ssid.c_str() );//, passwd.c_str());//20220826

  // wifi connect
  uint8_t retry = 0;
 WiFi.begin(ssid.c_str(), passwd.c_str());
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
//    delay(200); // 200ms
    delay(100); // 200ms //20220826
    retry ++;
//    if (retry > 50) { // 200ms x 50 = 10 sec
    if (retry > 200) { // 200ms x 50 = 10 sec //20220826
     Serial.println("wifi connection timeout");
     Serial.print('N');
     Serial2.print('N');//0x4e
      //              01234567890123450123456
     //Serial2.println("wifi connection timeout");//x20220816
     webconfig(); // enter webconfig
     // next to reboot
     }
  }
 //Serial.println();//CR+LF
 Serial.println("wifi connection completed");
 Serial.printf("Connected, IP address: ");
 Serial.print(ssid);
```

```
 Serial.println(WiFi.localIP());

 delay(100);//100ms
 Serial.println('Y');
 Serial2.println('Y');//0x59 + LF //usart1 case 30: case 31://SR20側でやり取り
 delay(100);//100ms
Serial2.print("WiFi:");//SR20側で表示 //WiFi:106F3FD9B204
 Serial2.println(ssid);//SR20側で表示  //
 delay(100);//100ms
Serial2.println(WiFi.localIP());//SR20側で表示//192.168.0.11
 delay(100);//100ms

 while(1){
   if (Serial2.available()) {            // if there's bytes to read from the client,
     c2 = Serial2.read();     // read a byte, then
     Serial.write(c2);
     if( c2 == 0x62 ){// shift onpls //"b"(0x62)//SR20側でやり取り
       break;
      }
     else if( c2 == 0x63 ){// shift keep //"c"(0x63)  //SR20側でやり取り
       // saisettei
       webconfig(); // enter webconfig
       // next to reboot
       break;
      }
    }
  }

 // We start by connecting to a WiFi network

 Serial.print("Connecting to ");
 Serial.println(ssid);

 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
  }

 Serial.println("");
 Serial.println("WiFi connected.");
 Serial.println("IP address: ");
Serial.println(WiFi.localIP());

 server.begin();
}

//*****************************************
void webconfig() {

  Serial.println("WebConfig mode: ");
  //Serial.print("'");//0x27
  //Serial2.print("'");//0x27

  configserver();

  uint8_t configloop = 1;
//  uint8_t configloop = 0;
  while (configloop == 1) {
//   dnsServer.processNextRequest();//x20220826
    webServer.handleClient();
    delay(10);//////??????????
   }

// digitalWrite(LED_BUILTIN, LOW);

 WiFi.disconnect(true);
 WiFi.mode(WIFI_OFF);
}
```

```
void configserver() {
 WiFi.disconnect(true);
 WiFi.mode(WIFI_OFF);
  delay(100);
 WiFi.mode(WIFI_AP);
  WiFi.softAP(WIFIMGR_ssid); // no password
//  WiFi.softAP(WIFIMGR_ssid,WIFIMGR_pass); // with password

  delay(200); // Important! This delay is necessary
//  delay(100); // Important! This delay is necessary //20220822 tuika

 WiFi.softAPConfig(apIP,apIP,IPAddress(255,255,255,0));

//dnsServer.setErrorReplyCode(DNSReplyCode::NoError);//x20220826
//  dnsServer.start(53, "*", apIP);//x20220826

  webServer.on("/", wifimgr_top);
  webServer.on("/wifiinput", HTTP_GET, wifiinput);
  webServer.on("/wifiset", HTTP_GET, wifiset);
  webServer.on("/reboot", reboot);
  webServer.on("/doreboot", doreboot);
 webServer.onNotFound(wifimgr_top);
  webServer.begin();
}

String maskpasswd(String passwd){
  String maskpasswd = "";

  for (int i=0; i<passwd.length(); i++) maskpasswd = maskpasswd + "*";
  if (passwd.length() == 0) maskpasswd = "(null)";

  return maskpasswd;
}

void wifimgr_top() {

  String html = Headder_str();
  html += "<a href='/wifiinput'>WIFI setup</a>";
//  html += "<a href='/wifiinput'><button>WIFI setup</button></a>"; //test <button>
  html += "<hr><h3>Current Settings</h3>";
  html += "SSID: " + ssid + "<br>";
  html += "passwd: " + maskpasswd(passwd) + "<br>";
 html += "<hr><p><center><a href='/reboot'>Reboot</a></center>";
  html += "</body></html>";
  webServer.send(200, "text/html", html);
}

//スタイルシート
String Headder_str() {
  String html = "";
  html += "<!DOCTYPE html><html><head>";
  html += "<meta name='viewport' content='width=device-width, initial-scale=1.3'>";
  html += "<meta http-equiv='Pragma' content='no-cache'>";
  html += "<meta http-equiv='Cache-Control' content='no-cache'></head>";
  html += "<meta http-equiv='Expires' content='0'>";
  html += "<style>";
//  html += "a:link, a:visited { background-color: #009900; color: white; padding: 5px 15px;";
  html += "a:link, a:visited { background-color: green; color: white; padding: 6px 15px;";// 押す前
  html += "text-align: center; text-decoration: none;  display: inline-block;}";//上から続き
//  html += "a:hover, a:active { background-color: green;}";//押したとき
  html += "a:hover, a:active { background-color: #7CFC00; color: black;}";//押したとき Grass Green

  html += "bo32{ background-color: #EEEEEE;}";
  html += "input[type=button], input[type=submit], input[type=reset] {";
  html += "background-color: #000099;  border: none;  color: white;  padding: 5px 20px;";
  html += "text-decoration: none;  margin: 4px 2px;";
  html += "</style>";
  html += "<body>";
//  html += "<h2>WIFIMGR</h2>";
  html += "<h2>WiFi Manager(Station mode) for SR20</h2>";//ver2
```

```
  return html;
}

void InitialConfigFile(){
  Serial.printf("SPIFFS initial file: %s\n", configfile);

  ssid = defaultSSID;
  passwd = defaultPASSWD;

  WriteConfigFile();
}

//*****************************************
void WriteConfigFile(){

  ssid.trim();
  passwd.trim();

  Serial.printf("SPIFFS writing file: %s\n", configfile);
  File fw = SPIFFS.open(configfile, "w");
  fw.println(ssid);
  fw.println(passwd);
  fw.close();

  delay(100);
}

//*****************************************
void readConfigFile(){
  String numstr;
  Serial.printf("SPIFFS reading file: %s\n", configfile);
  File fr = SPIFFS.open(configfile, "r");
  if (fr) {
  //if (0) {
    ssid = fr.readStringUntil('\n');
    ssid.trim();
    if (ssid =="") ssid = defaultSSID;
    passwd = fr.readStringUntil('\n');
    passwd.trim();
    if (passwd == "" && ssid == defaultSSID) passwd = defaultPASSWD;

    fr.close();

  } else {
    //InitialConfigFile
    Serial.println("read open error");
    Serial.print("SPIFFS data seems clash. Default load...");
    InitialConfigFile();
  }

}

//*****************************************
void DeleteConfigFile(){
 SPIFFS.remove(configfile);
  SPIFFS.end();
}


void wifiinput() {
  String html = Headder_str();
  html += "<a href='/'>TOP</a> ";
  html += "<hr><p>";
  html += "<h3>WiFi Selector</h3>";
  html += WIFI_Form_str();
  html += "<br><hr><p><center><a href='/'>Cancel</a></center>";
  html += "</body></html>";
  webServer.send(200, "text/html", html);
}
```

```
//*****************************************
String WIFI_Form_str(){

 Serial.println("wifi scan start");

 // WiFi.scanNetworks will return the number of networks found
 uint8_t ssid_num = WiFi.scanNetworks();
 Serial.println("scan done\r\n");

 if (ssid_num == 0) {
  Serial.println("no networks found");
 } else {
  Serial.printf("%d networks found\r\n\r\n", ssid_num);
   if (ssid_num > SSIDLIMIT) ssid_num = SSIDLIMIT;
   for (int i = 0; i < ssid_num; ++i) {
     ssid_str[i] = WiFi.SSID(i);
    String wifi_auth_open = ((WiFi.encryptionType(i) == WIFI_AUTH_OPEN)?" ":"*");
     ssid_rssi_str[i] = ssid_str[i] + " (" + WiFi.RSSI(i) + "dBm)" + wifi_auth_open;
     ssid_rssi_str[i] = ssid_str[i] + wifi_auth_open;
    Serial.printf("%d: %s\r\n", i, ssid_rssi_str[i].c_str());
     delay(10);
   }
 }

 String str = "";
 str += "SSID:";
 str += "<form action='/wifiset' method='get'>";
 str += "<select name='ssid' id ='ssid'>";
 for(int i=0; i<ssid_num; i++){
   str += "<option value=" + ssid_str[i] + ">" + ssid_rssi_str[i] + "</option>";
 }
 str += "<option value=" + ssid + ">" + ssid + "(current)</option>";
 if (ssid != defaultSSID){
   str += "<option value=" + defaultSSID + ">" + defaultSSID + "(default)</option>";
 }
 str += "</select><br>\r\n";
 str += "Password:<br><input type='password' name='passwd' value='" + passwd + "'>";
 str += "<br><input type='submit' value='set'>";
 str += "</form><br>";
 str += "<script>document.getElementById('ssid').value = '"+ ssid +"';</script>";
 return str;
}


void wifiset(){

 ssid = webServer.arg("ssid");
 passwd = webServer.arg("passwd");
 ssid.trim();
 passwd.trim();

 WriteConfigFile();

  // 「/」に転送
 webServer.sendHeader("Location", String("/"), true);
 webServer.send(302, "text/plain", "");
}

void reboot() {
  String html = Headder_str();
  html += "<hr><p>";
  html += "<h3>reboot confirmation</h3><p>";
  html += "Are you sure to reboot?<p>";
  html += "<center><a href='/doreboot'>YES</a> <a href='/'>no</a></center>";
  html += "<p><hr>";
  html += "</body></html>";
  webServer.send(200, "text/html", html);
}

void doreboot() {
```

```
  String html = Headder_str();
  html += "<hr><p>";
  html += "<h3>rebooting</h3><p>";
  html += "The setting WiFi connection will be disconnected...<p>";
  html += "<hr>";
  html += "</body></html>";
  webServer.send(200, "text/html", html);

  // reboot esp32
  Serial.println("reboot esp32 now.");
 Serial2.print("c");//reboot

//  digitalWrite(LED_BUILTIN, LOW);

//  delay(2000); // hold 2 sec
  delay(1000); // hold 1 sec
  ESP.restart(); // restart ESP32
}

void loop_wifiSta(){
  while(1){
  WiFiClient client = server.available();   // listen for incoming clients
    //if (client) {                             // if you get a client,
   while (client) {                             // if you get a client,
    //Serial.println("New Client.");           // print a message out the serial port
    while (client.connected()) {           // loop while the client's connected
//      if (client.available()) {             // if there's bytes to read from the client,
//        String line = client.readStringUntil('\n');          // Get Line data until '\n'
//        Serial.print(line);
//        Serial2.print(line);
//      }

     //if (client.available()) {             // if there's bytes to read from the client,
     while (client.available()) {             // これが早い！！
       c1 = client.read();             // read a byte, then
       Serial.write(c1);                    // print it out the serial monitor
       Serial2.write(c1);                   // to SR20
     }

     //if (Serial2.available()) {             // if there's bytes to read from the client,
     while (Serial2.available()) {             // if there's bytes to read from the client,
       c2 = Serial2.read();              // read a byte, then
      Serial.write(c2); // print it out the serial monitor
       client.write(c2);                    // from SR20
      serial2TxComCnt = 10;//1s in ( void intervalTimer() {//01s interval )
     }
     if(c2=='#'){ Serial.println(); c2 = 0; }
     }
     /*
     // close the connection:
     client.stop();
    Serial.println("Client Disconnected.");
     */
   }
  if(serial2TxComCnt){// in void intervalTimer()  (serial2TxComCnt--)
    ledR_OffOnBlk = 7;// 1:off 2:on 3:blk2s 4:blk1s 5:blk0.5s 6:blk0.3 7:blk0.1s
   }
   else{
    ledR_OffOnBlk = 1;// 1:off 2:on 3:blk2s 4:blk1s 5:blk0.5s 6:blk0.3 7:blk0.1s
   }
 }
}
```