スマホで家電リモコン編

- ESP32におけるEEPROMの実装
- リモコン信号の保存、読出による家電操作

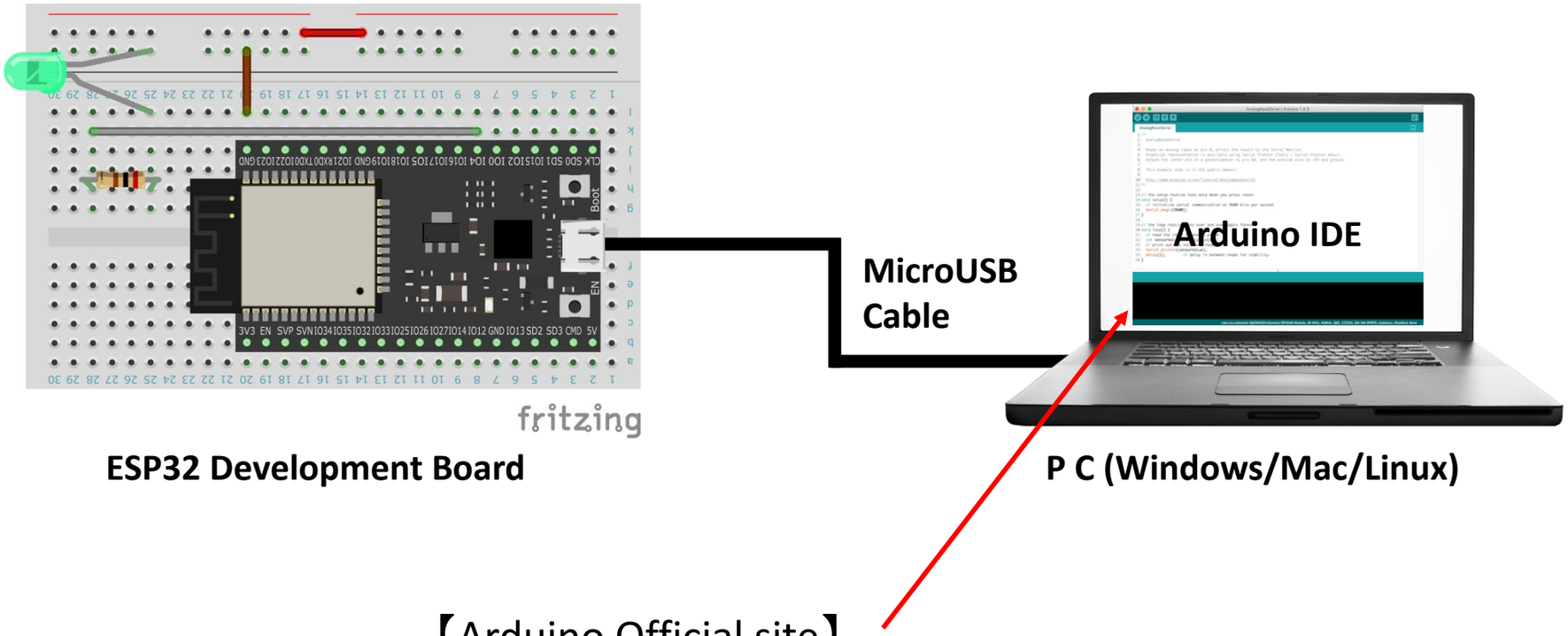
目　次　《スマホで家電リモコン編》

1－1．スマートリモコン製作全体の流れ

No	項目	内容	ハード	ソフト	記事
1	概要	全体の流れ、システム構成、利用物品、 選定理由、開発環境など	-	-	別動画で配信
2	LED	初めて電子工作される方向けの基本を行います。 LEDの点灯、点滅を行う「Lチカ」を製作します。	○	○	
3	赤外線受信センサ	赤外線受信センサーの説明 回路図から配線、ソフトウェア	○	○	
4	赤外線送信LED	赤外線送信LEDの説明 回路図から配線、ソフトウェア	○	○	
5	スマホでLED操作 （宅内）	工作したリモコンのLEDを屋内のスマホから操作する ソフトウェアを製作します。（Webサーバ機能、SPIFFS操作）	-	○	
6	スマホでリモコン操作 （宅内）	工作したリモコンを屋内のスマホから操作する ソフトウェアを製作します。（ボタン名、信号保存・読出）	-	○	今回はこの動画
7	屋外からスマホで操作 及び、AIスピーカ連携	工作したリモコンを屋外からスマホで操作したり AIスピーカ連携を実現するソフトウェアを製作します。	-	○	別動画で配信

１－２．開発環境Arduinoについて

開発環境はArduinoを利用していきます。



【Arduino Official site】
https://www.arduino.cc/
ダウンロード可能

2. EEPROM

EEPROMとは

*1：https://ja.wikipedia.org/wiki/EEPROM

EEPROM（Electrically Erasable Programmable Read-Only Memory）は不揮発性メモリの一種*1
ESP32では、Flashメモリの一部をEEPROMとして利用する疑似EEPROMとなります。
前回、実装したSPIFFSはファイルとして扱うので、比較的大きなデータとなります。リモコン信号はSPIFFSで利用します。
小さなデータを扱うのにはEEPROMの方がデータ型やメモリ位置を指定して扱えるので利用しやすいです。
今回はボタン名を保存するのに利用します。

プログラミング

```
#include <EEPROM.h>
```
ライブラリを読み込み、利用できるようにします。

```
EEPROM.begin(650);
```
開始を宣言し利用できるようにします。
（650Byte分の利用を定義しています。最大4Kbyte）

メモリ位置　設定値

```
EEPROM.put<st_remocon>(memPos, remRom);
```
Putで書き込み

```
EEPROM.get<st_remocon>(memPos, remRom);
```
getで読み込み

```
EEPROM.commit();
```
書き込みを実行させる

3. プログラムのファイル構成

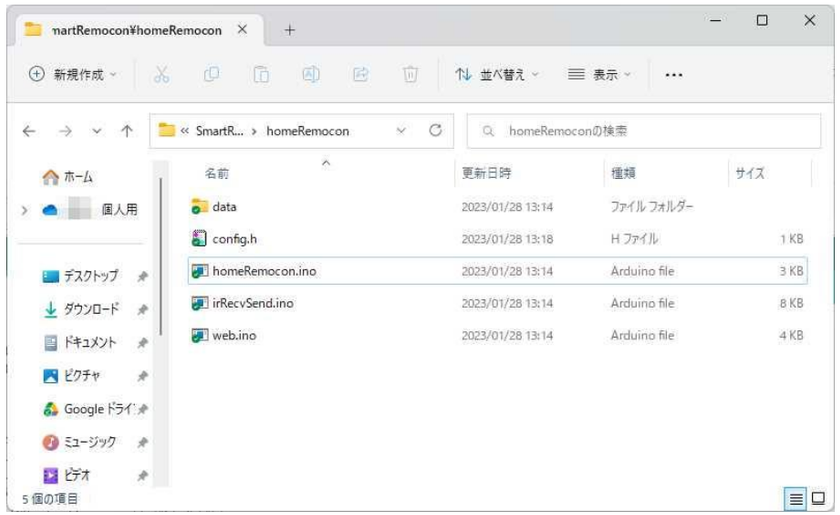
●ファイル構成

プログラム種別

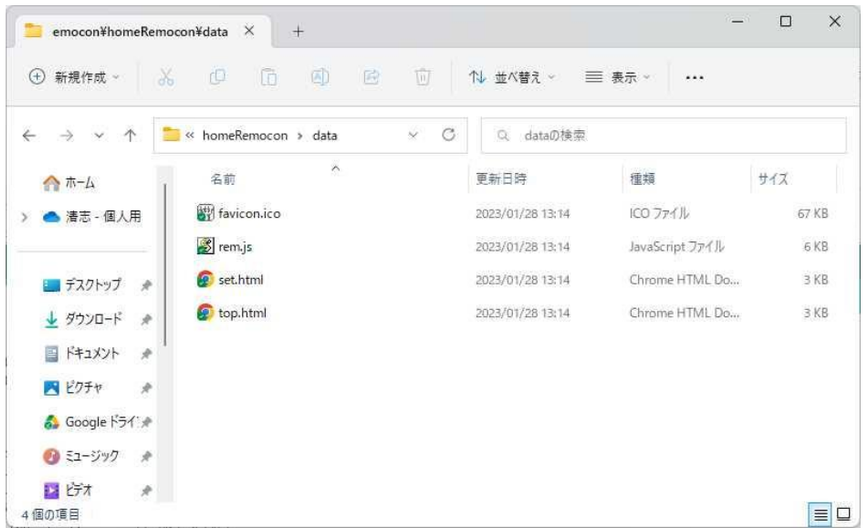
- homeRemocon
  - homeRemocon.ino : setup関数やloop関数などのプログラム
  - config.h : Wi-FiのSSID、パスワード、IPアドレスなどの環境設定
  - irRecvSend.ino : リモコン操作に関するプログラム
  - web.ino : Webサーバに関するプログラム
  - data
    - top.html : トップ画面（ボタン制御画面）
    - set.html : 設定画面
    - rem.js : ボタン設定や操作を実施
    - favicon.ino : 無くても良い。ブラウザのアイコンマーク

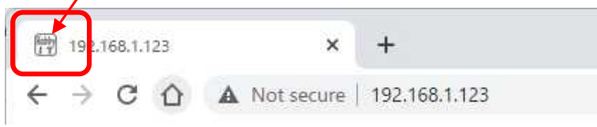
Arduino / HTML / Javascript

今回のスケッチフォルダ

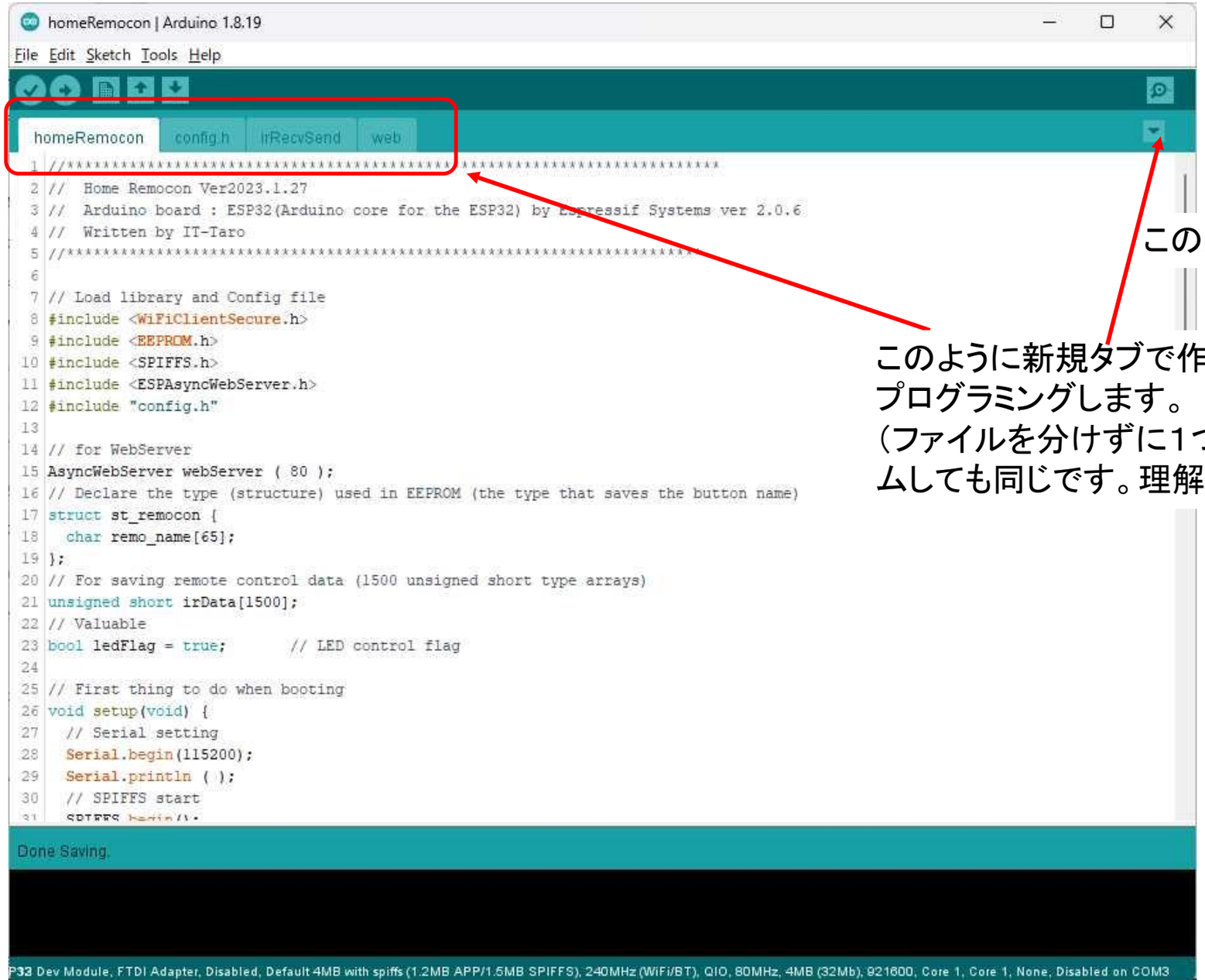


SPIFFSで利用するdataフォルダ





4. Arduinoプログラム



このように新規タブで作成し、4つのタブに分けて
プログラミングします。
（ファイルを分けずに1つのファイルに全てプログラムしても同じです。理解やすいように分けてます）

この三角マークから新規タブ作成

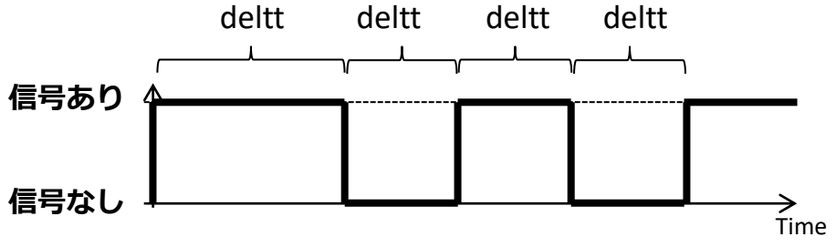
4．Arduinoプログラム（リモコン信号をSPIFFSのファイルへ保存）

●信号のあり･なし時間を取得（赤外線受信センサで理解）

irRecvSend.ino

```
61      deltt = ( (cMicro - sMicro)/ 10 ) - lastt;
62      irData[(irCount - 1 )] = deltt;
63      // Save last changed elapsed time for next elapsed time ca
```

取得した時間を配列で全て保持



●取得した時間間隔をSPIFFSのファイルに保存（該当部分を抜粋）

irRecvSend.ino

```
103  // Create a file name to save the remote control signal (the file name is the button number)
104  String t_file = "/" + setNumStr;
105  Serial.println( "recvFile:" + t_file);
106  // open file in write mode
107  File fw = SPIFFS.open(t_file.c_str(), "w");
108  // Write remote control signal length first (first line)
109  fw.println( String( irLength, HEX ) );
110  // Write the time length of 0 and 1 of the remote control signal (from the second line)
111  for (int i = 0; i < irLength; i++) {
112    fw.println( String( irData[i], HEX ) );
113  }
114  // Close the file when writing is complete
115  fw.close();
```

ファイル名を作成（リモコン番号がファイル名）

ファイルを書き込みでオープン

ファイルにデータを書き込み（信号あり・なしの回数）

信号あり・なしの回数を繰り返す

ファイルにデータを書き込み（信号の時間幅）

ファイルをクローズ

4. Arduinoプログラム（ボタン名をEEPROMへ保存）

homeRemocon.ino

```
// Declare the type (structure) used in EEPROM (the type that saves the button name)
struct st_remocon {
  char remo_name[65];
};
```

最初にEEPROMで扱うデータを構造体で定義
ボタン名のみを扱うのでChar型を定義
（65Byte定義なので、英語で60、日本語で30文字程度）

irRecvSend.ino

homeRemocon | config.h | irRecvSend | web

```
// Save button name to EEPROM and remote control data to file
bool saveIr(unsigned short irLength, AsyncWebServerRequest *request){
  String setirname = "";
  String setNumStr = "";
  // Get and check button number (HTTP GET request parameter)
  if (request->hasParam("n")) {
    setNumStr = request->getParam("n")->value();
  } else {
    return false;
  }
  // Get and check button name (parameter of HTTP GET request)
  if (request->hasParam("a")) {
    setirname = request->getParam("a")->value();
  } else {
    return false;
  }
  // Convert the button number from String type to int type
  int setNum = setNumStr.toInt();
  // Append the identification character "O:" to the beginning of the button name
  setirname = "O:" + setirname;
  // Define a variable with matching type for storage in EEPROM
  st_remocon remRom;
  // Convert from String to char type (Length +1 to add end character)
  setirname.toCharArray(remRom.remo_name, setirname.length()+1);
  // Calculate memory location and write to EEPROM
  int memPos = (65 * setNum);
  EEPROM.put<st_remocon>(memPos, remRom);
  EEPROM.commit();
  Serial.println("setIr:" + String(setNum) + ":" + setirname);
  // Create a file name to save the remote control signal (the file name is the button number)
```

ボタン名を扱う変数を定義
ボタン番号を扱う変数を定義
HTTPのGETパラメータを取得（ボタン番号）
HTTPのGETパラメータを取得（ボタン名）
ボタン番号を文字列から数値に変換（メモリ位置に利用）
ゴミデータとの区別に“o:”で始まるデータを保存
構造体を変数に定義
ボタン名を定義した構造体に保存
EEPROMの保存メモリ位置を算出(1ボタンに65Byte利用)
EEPROMへ書き込み
書き込みの実行

4. Arduinoプログラム（ボタン名をEEPROMから読み出し）

web.ino

```
64 void getRemocon(AsyncWebServerRequest *request) {
65   // Create transmission data (JSON format)
66   String senddata = "{";
67   // Declare a variable to store EEPROM data
68   st_remocon remRom;
69   // Read 10 pieces of button information and reply
70   for (byte i = 0; i < 10; i++) {
71     // Calculate EEPROM memory location
72     int memPos = (65 * i);
73     // Erase so that the previous value 'O:' does not remain
74     remRom.remo_name[0] = 'n';
75     // Get data from EEPROM
76     EEPROM.get<st_remocon>(memPos, remRom);
77     // Check if data is saved
78     if (remRom.remo_name[0] == 'O' && remRom.remo_name[1] == ':' ) {
79       // If the response string length exceeds 1, add "," (delimiter from the second and subsequent characters)
80       if (senddata.length() > 1) {
81         senddata += ",";
82       }
83       // Replace the returned value with String type once (to remove "O:")
84       String getirname = String(remRom.remo_name);
85       // Create reply string (from 2 to the end to remove "O:")
86       senddata += "\"" + (String)i + "\":\"" + getirname.substring(2,getirname.length()) + "\"";
87     }
88   }
89   // Add "}" at the end to close the JSON data
90   senddata += "}";
91   // Send the created response (JSON) data from the web server
92   request->send(200, "text", senddata);
93   Serial.println( "getRemocon:" + senddata);
94 }
```

送信するデータを作成するため、変数を定義します。（送信データはJSON形式です）

EEPROMから読み出したデータを格納する変数を定義

for文でボタン10個分を処理します。

メモリ位置を計算します。

念のため、“n”を設定し、“O”との違いを明確にします。

EEPROMから指定メモリ位置の情報を読み出します。

情報があれば”O:”で始まるので存在するか判定

2回目以降はカンマを追加し、区切りにする。

取得したデータをChar型からString型に変更

送信データにボタン番号とボタン名を追加

送信データに追加

HTMLで送信データを返答

送信データ（例）
{“1”:”ライトON”,”2”:”ライトOFF”}

5. HTMLプログラム

```
<!doctype html>
<!-- ◆◆◆HTML Tag◆◆◆ -->
<html>
 <!-- ◆◆◆head Tag◆◆◆ -->
 <head>
  <meta charset='UTF-8'/>
  <meta name='viewport' content='width=device-width'/>
  <!-- ##### StyleSheet ##### -->
  <style type='text/css'><!--
   #contents { width: 100%; max-width: 320px; }
   #menu{ color: #fff; background: #222; }
   .underTheEarthKai {
     background-image: radial-gradient(50% 150%, #CCCCCC 5%, #777777 100%);
   }
   button { width:155px; height:35px }
   #dispStatus{ color: #f00; }
   footer { text-align: right; }
  --></style>
  <!-- ##### Javascript ##### -->
  <script type='text/javascript' src='rem.js'></script>
 </head>
 <!-- ◆◆◆Body Tag◆◆◆ -->
 <body class='underTheEarthKai'><center><div id='contents'>
  <header><h3>Smart Remote controller</h3></header>
  <div id='menu'>Controller Screen</div>
  <div align=right><a href='/set'>[Setting]</a></div>
  <!-- ##### Button Tag ##### -->
  <table>
   <tr>
    <td><button id='btn0' class='cntbtn' onClick="snd(0)">
     <font size=+1><span id='spn0'>-</span></font></button></td>
     ～ （省略） ～
    <td><button id='btn9' class='cntbtn' onClick="snd(9)">
     <font size=+1><span id='spn9'>-</span></font></button></td>
   </tr>
  </table>
  <!-- ##### DivTag（Display Status） ##### -->
  <div id='dispStatus'><br></div>
  <!-- ##### Footer Tag ##### -->
  <footer><font size=-1>©Hobby-IT</font></footer>
 </div></center></body>
</html>
```

StyleSheet
画面の大きさや背景色、ボタンの大きさなどの
デザインに関する内容を設定

Javascript
Javascriptの定義、ファイルを指定しているので、
Webサーバにファイルを要求

JavascriptはHTMLの要素を、Webページを更新せずにダイナミックに変更できる。

リモコンボタン10個を表示
Tableは綺麗に並べるように利用しているが、テーブル線自体は
表示させていない。

ステータス表示
操作の完了などのステータスを表示します。

6. Javascriptプログラム

```
// ● onload is executed when the screen is loaded
window.onload = function () {
  // ● Execute remote control button information update processing
  updateIr();
}
```

Javascriptファイルが
読み込まれた時に実行される

```
// ● Acquisition and display of remote control button information
function updateIr() {
  var xhr = new XMLHttpRequest();
  var url = window.location.href;
  var urlarr = url.split("/");
  // ● Create an access URL (example: http://192.168.1.123:12193/getrem)
  url = "http://" + urlarr[2] + "/getrem";
  xhr.timeout = 5000;
  xhr.ontimeout = function(e){
    document.getElementById('dispStatus').innerHTML = "<b>Failed to access the device</b>";
  };
  xhr.open("GET", url);
  xhr.send("");
  xhr.addEventListener("load",function(ev){
    var resGtStr = xhr.responseText;
    var gtRecv =  JSON.parse(resGtStr);
    // ● Check the data for 10 buttons
    for ( i=0 ; i<10 ; i++) {
      // ● Check if received data (button name) exists
      if ( gtRecv[i] != "" && typeof gtRecv[i] !== "undefined" ) {
        var idname = "spn" + i;
        if ( document.getElementById(idname) != null ) {
          // ● Enter the button name when "spn0-9" exists (control screen)
          document.getElementById(idname).innerHTML = gtRecv[i];
        } else {
          // ● If "spn0-9" does not exist (setting screen), enter the button name in "ir0-9"
          idname = "ir" + i;
          document.getElementById(idname).value = gtRecv[i];
        }
      } else {
        // ● If "btn0-9" exists (control screen) and there is no button name, disable the button
        var idname = "btn" + i;
        if ( document.getElementById(idname) != null ) {
          document.getElementById(idname).disabled = true;
        }
      }
    }
  });
}
```

HTTP Get　リクエスト
[http://192.168.1.123/getrem]

返答されたボタン情報を
HTMLに書き換えて表示

6. Javascriptプログラム

```
// ● Define global variables (used in send/receive functions)
irFlg=false;    // Reception processing flag (true: processing, false: processing possible)
flgRed=true;    // Status display display/non-display flag
count=0;        // Count every 1 second for timeout judgment
rcvTimer=15;    // timeout seconds

// ● Remote control signal processing
function snd(setNum) {
 // ● Judgment during processing
 if (irFlg) {
  // ●If processing is in progress, display processing and exit.
  document.getElementById('dispStatus').innerHTML = "<b>Processing</b>";
  return;
 }
 // ● Set the action flag as being processed, and perform display processing during reception
 irFlg=true;
 document.getElementById('dispStatus').innerHTML = "<b>Sending remote control</b>";
 var xhr = new XMLHttpRequest();
 var url = window.location.href;
 var urlarr = url.split("/");
 // ● Create an access URL (example: http://192.168.1.123:12193/cntrem?n=1)
 url = "http://" + urlarr[2] + "/cntrem?n=" + setNum;
 xhr.timeout = 5000;
 xhr.ontimeout = function(e){
  dispfail();
 };
 xhr.open("GET", url);
 xhr.send("");
 xhr.addEventListener("load",function(ev){
  var resStr = xhr.responseText;
   // ●When OK is received, the status is displayed in the if statement. Otherwise, display the state inside else
  if ( resStr.indexOf("OK") != -1 ) {
   document.getElementById('dispStatus').innerHTML = "<b>Transmission Completed!</b>";
  } else {
   document.getElementById('dispStatus').innerHTML = "<b>Transmission Failure!</b>";
  }
  // ● Return the processing flag
  irFlg=false;
 });
}
```

処理中か判断

HTTP Get　リクエスト
[http://192.168.1.123/ cntrem?n=x]

応答によりステータス欄に完了か失敗を表示

リモコン送信処理

6. Javascriptプログラム

```
// ●Remote control reception processing
function rcv(setNum){
 // ● Processing counter reset
 count = 0;
 // ● Judgment during processing
 if (irFlg) {
  // ●If processing is in progress, display processing and exit.
  document.getElementById('dispStatus').innerHTML = "<b>Processing</b>";
  return;
 }
 // ● Set the action flag as being processed, and perform display processing during reception
 irFlg=true;
 setMsgTenmetu();
 // ● Acquire the entered button name
 var idname = "ir" + setNum;
 var setName = document.getElementById(idname).value;
 // ● Access to main unit for reception setting processing
 var xhr = new XMLHttpRequest();
 var url = window.location.href;
 var urlarr = url.split("/");
 url = "http://" + urlarr[2] + "/setrem?n=" + setNum + "&a=" + setName;
 xhr.timeout = rcvTimer * 1000;
 xhr.ontimeout = function(e){
  dispfail();
 };
 xhr.open("GET", url);
 xhr.send("");
 xhr.addEventListener("load",function(ev){
  var resStr = xhr.responseText;
  // ●When OK is received, the status is displayed in the if statement. Otherwise, display the state inside else
  if ( resStr.indexOf("OK") != -1 ) {
   // ● Flag to receive completion. Complete display in status display
   irFlg=false;
   document.getElementById('dispStatus').innerHTML = "<b>Setting Completed!</b>";
  } else {
   // ●Failure display
   dispfail();
  }
 });
}
```

処理中か判断

HTTP Get　リクエスト
[http://192.168.1.123/ setrem?n=x&a=xxxxx]

応答によりステータス欄に完了か失敗を表示

リモコン受信処理

6. Javascriptプログラム

```
// ● Blink processing of status display (during remote control reception)
function setMsgTenmetu(){
  // ● Reception is not complete. and before timeout
  if (irFlg == true && count < rcvTimer )   {    // If reception is not completed
    // ● "flgRed" alternately displays the IF statement and the else statement every 1 second (blinking during reception)
    if(flgRed){
      document.getElementById('dispStatus').innerHTML = "<b>Receiving signals (" + rcvTimer + " seconds)</b>";
    }else{
      document.getElementById('dispStatus').innerHTML = "<br>";
    }
    // ● Invert status display status
    flgRed=!flgRed;
    // ● After 1 second, execute "setMsgTenmetu()" again
    setTimeout("setMsgTenmetu()",1000);
    count++;
  // ●If it has timed out, go to failure processing.
  } else if (count >= rcvTimer) {
    dispfail();
  }
}

// ● Display when failure occurs
function dispfail(){
  // ● Match the count to the timer out so as not to blink
  count=rcvTimer;
  irFlg=false;
  // ● Show failure in status
  document.getElementById('dispStatus').innerHTML = "<b>Setting Failure!</b>";
}
```

リモコン受信設定中の点滅処理
（1秒毎に赤字が点滅します。）

失敗時の表示処理

7. 各プログラムの動作概要

●スマホでTOP画面を表示時