Home Appliance Remote Control with Smartphone

- Implementation of EEPROM in ESP32
- Home appliance operation by storing and reading remote control signals

Table of Contents

1. Overview
1-1. Overall flow of smart remote controller production
1-2. About the development environment Arduino

2. EEPROM implementation

3. File structure of the program

4. Arduinoprogramming

5. HTML programming

6. Javascriptprogramming

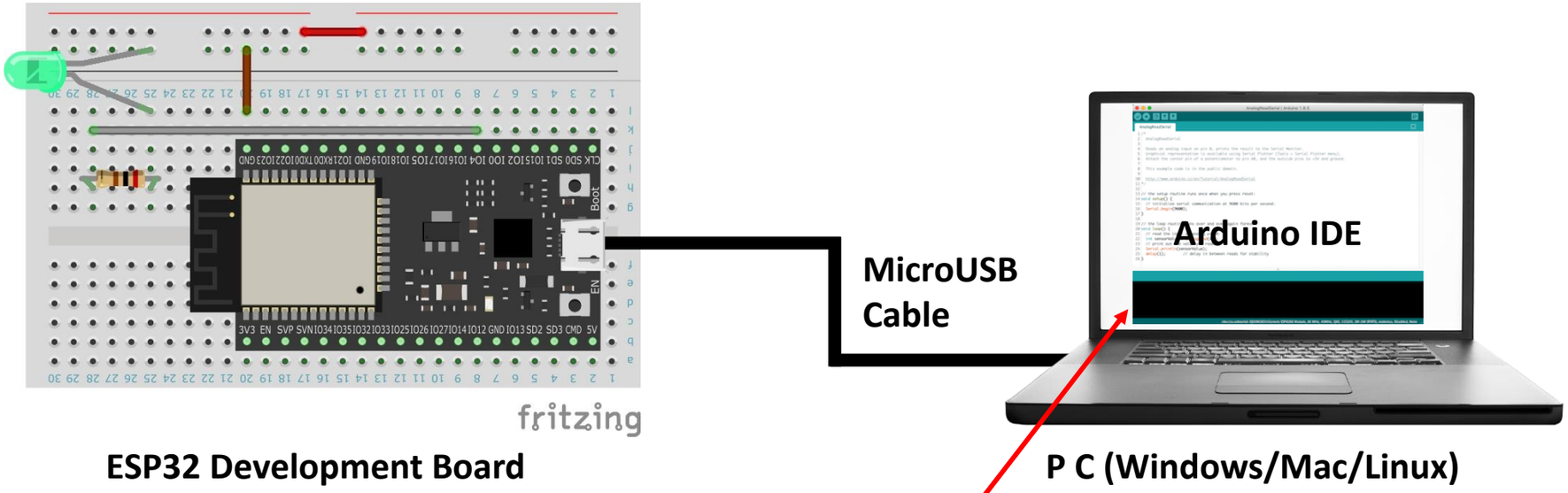
7. Operation overview of each program

1-1. Overall flow of Smart Remote Controller production

No	Item	Content	Hard	Soft	Note
1	Overview	Overall flow, system configuration, items used, reasons for selection, development environment, etc.	-	-	Delivered in another video
2	LED	Learn the basics for beginners. We will make "L blinking" that lights up and blinks the LED.	○	○	
3	Infrared receiving sensor	Description of infrared receiving sensor Schematic to Wiring, Software	○	○	
4	Infrared transmission LED	Infrared transmission LED description Schematic to Wiring, Software	○	○	
5	LED operation with smartphone(at home)	We will create software to operate the LED with smartphone. (Web server function, SPIFFS operation)	-	○	
6	Remote control with smartphone(at home)	We will create software that to operate the remote control with smartphone indoors. (Button name, signal save/read)	-	○	this time this video
7	Operate from outside And AI speaker cooperation	We will create software to operate the remote control with smartphone from the outdoors, and AI speaker cooperation.	-	○	Delivered in another video

1-2. the development environment “Arduino”

We will use Arduino as the development environment.



ESP32 Development Board

P C (Windows/Mac/Linux)

【Arduino Official site】
https://www.arduino.cc/
Downloadable

2. EEPROM

What is EEPROM

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

EEPROM (Electrically Erasable Programmable Read-Only Memory) is a type of non-volatile memory*1
ESP32 is a pseudo EEPROM that uses a part of Flash memory as EEPROM.
Since the SPIFFS implemented last time is handled as a file, it will be a relatively large amount of data. SPIFFS is used for the remote control signal.
For handling small data, EEPROM is easier to use because it can be handled by specifying the data type and memory location.
This time it will be used to save the button name.

Programming

```
#include <EEPROM.h>
```

Load the library and make it available.

```
EEPROM.begin(650);
```

Declare a start and make it available.
(It defines the use of 650 bytes. Maximum 4 Kbytes)

Memory location　set value

```
EEPROM.put<st_remocon>(memPos, remRom);
EEPROM.get<st_remocon>(memPos, remRom);
```

Write with Put
Read with get

```
EEPROM.commit();
```

Perform the write

3. File structure of the program

Program type

●File structure

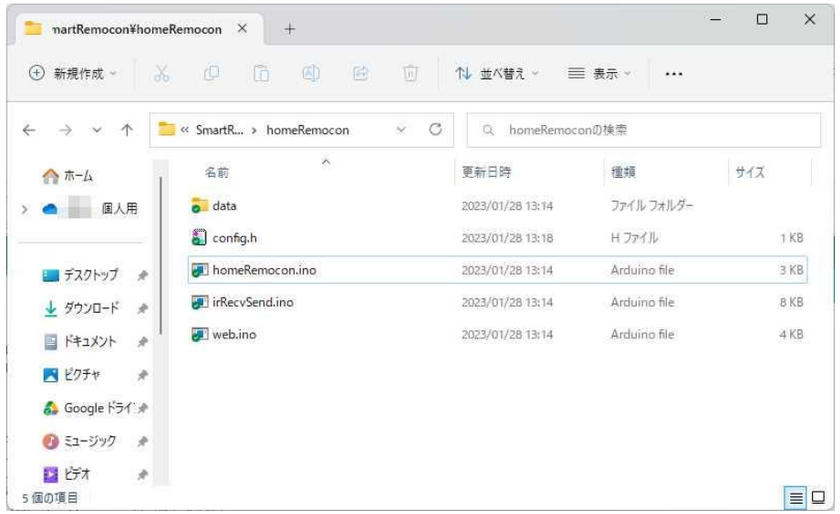
- homeRemocon
  - homeRemocon.ino : Programs such as setup functions and loop functions
  - config.h : Preferences such as Wi-Fi SSID, password, IP address
  - irRecvSend.ino : Program related to remote control operation
  - web.ino : A program related to the web server
  - data
    - top.html : Top screen (button control screen)
    - set.html : Settings screen
    - rem.js : Perform button settings etc.
    - favicon.ino : It doesn't have to be. browser icon mark

Arduino

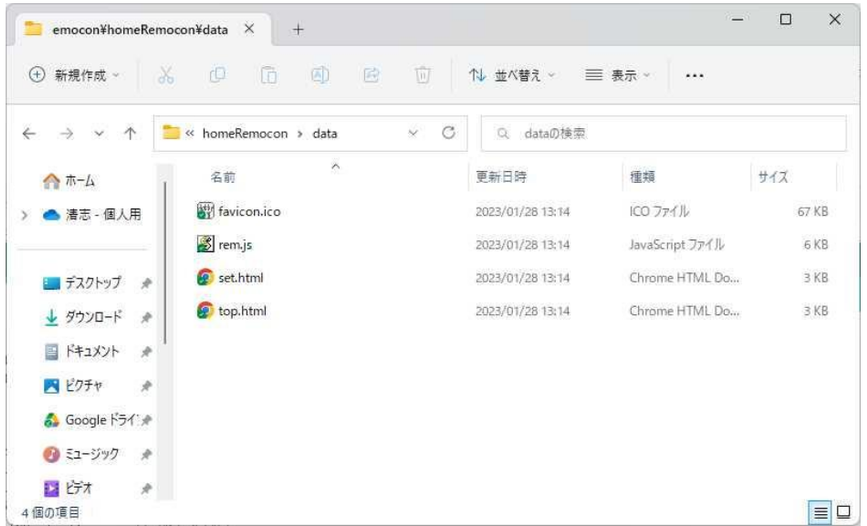
HTML

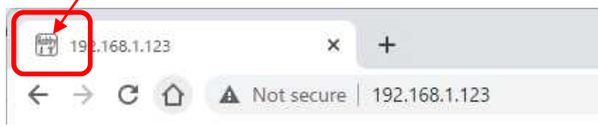
Javascript

This sketch folder



data folder used by SPIFFS





4. Arduino program

homeRemocon | Arduino 1.8.19

File Edit Sketch Tools Help

homeRemocon | config.h | irRecvSend | web

```
//*****************************************************************************
//  Home Remocon Ver2023.1.27
//  Arduino board : ESP32(Arduino core for the ESP32) by Espressif Systems ver 2.0.6
//  Written by IT-Taro
//*****************************************************************************

// Load library and Config file
#include <WiFiClientSecure.h>
#include <EEPROM.h>
#include <SPIFFS.h>
#include <ESPAsyncWebServer.h>
#include "config.h"

// for WebServer
AsyncWebServer webServer ( 80 );
// Declare the type (structure) used in EEPROM (the type that saves the button name)
struct st_remocon {
  char remo_name[65];
};
// For saving remote control data (1500 unsigned short type arrays)
unsigned short irData[1500];
// Valuable
bool ledFlag = true;       // LED control flag

// First thing to do when booting
void setup(void) {
  // Serial setting
  Serial.begin(115200);
  Serial.println ( );
  // SPIFFS start
  SPIFFS.begin();
```

Done Saving.

P33 Dev Module, FTDI Adapter, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, Core 1, Core 1, None, Disabled on COM3

Create a new tab from this triangle mark

Create it in a new tab like this, and program it by dividing it into four tabs.
(It is the same even if you program all in one file without dividing the files. It is divided for easy understanding.)

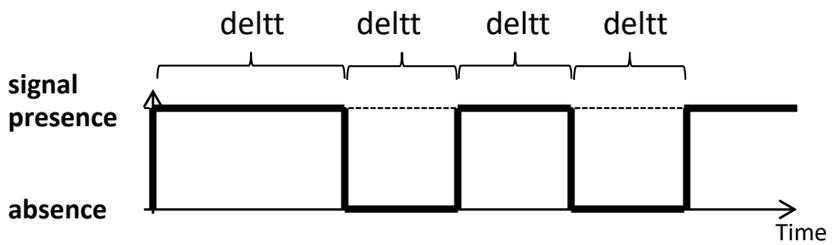
4. Arduino program (save remote control signal to SPIFFS file)

● **Acquisition of signal presence/absence time (understood by infrared receiving sensor)**

irRecvSend.ino

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

Hold all acquired times in an array



● **Save the acquired time interval in a SPIFFS file (excerpt)**

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();
```

Create file name (remote control number is file name)

open file for writing

Write data to file (number of times with/without signal)

Repeat the number of times with and without a signal

Write data to file (time width of signal)

close file

4. Arduino program (save button name to EEPROM)

homeRemocon.ino

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

Define the data to be handled by EEPROM in a structure
Define Char type because only button name is handled
(Since it is a 65Byte definition, it is about 60 characters in English and 30 characters in Japanese)

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)
```

Define a variable that handles the button name
Define a variable that handles the button number

Get HTTP GET parameter (button number)

Get HTTP GET parameters (button name)

Convert button number from string to numeric

Save data starting with “o:” to distinguish from garbage

Define a struct as a variable

Save the button name in a defined structure

Calculate the storage memory location of EEPROM

Write to EEPROM

Write execution

4. Arduino program (read button name from EEPROM)

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 }
```

Define variables to create the data to send.
(Transmission data is in JSON format)

Define variables to store data read from EEPROM

Process 10 buttons with a for statement.

Compute a memory location.

Just in case, set "n" to clarify the difference from "O".

Reads information from EEPROM.

If there is information, it starts with "O:“
so it is determined whether it exists

From the second time, add a comma to separate them.

Change the acquired data from Char type to String type

Add button number and button name to send data

Add to send data

Reply with sent data in HTML

Send data (example)
{“1”:”Light ON”,”2”:”Light OFF”}

5. HTML program

```
<!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>
```

Style Sheet
Set design-related items such as screen size, background color, and button size.

Javascript
Since the Javascript definition and file are specified, request the file from the web server

Javascript can dynamically change HTML elements without refreshing the web page.

Display 10 remote control buttons
The table is used so that it is arranged neatly, but the table line itself is not displayed.

Status display
Displays status, such as operation completion.

6. Javascript program

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

Executed
when a Javascript file is loaded

```
// ● 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 request
[http://192.168.1.123/getrem]

Rewrite and display the returned button information in HTML

6. Javascript program

```
// ● 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;
  });
}
```

Judging whether processing is in progress

HTTP Get request
[http://192.168.1.123/ cntrem?n=x]

Display completed or failed in the status column depending on the response

Remote control transmission processing

6. Javascript program

```
// ●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();
    }
  });
}
```

Judging whether processing is in progress

HTTP Get request
[http://192.168.1.123/ setrem?n=x&a=xxxxx]

Display completed or failed in the status column depending on the response

Remote control reception processing

6. Javascript program

```
// ● 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();
  }
}
```

Blink processing during remote control reception setting
(The red character flashes every second.)

```
// ● 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>";
}
```

Display processing at the time of failure

7. Operation overview of each program

- **When displaying the TOP page on a smartphone**