【赤外線受信センサ編】

スマートリモコン製作

- 受信センサの仕組みを理解
- リモコン信号を取得し保存するプログラム

目　次　《赤外線受信センサ編》

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	○	○	今回はこの動画
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	○	○	別動画で配信
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）	-	○	
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。	-	○	

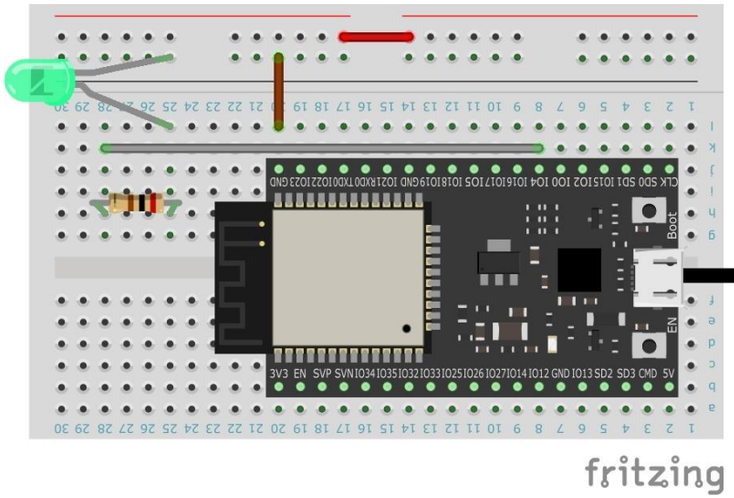
１－２．スマートリモコン製作の利用物品一覧

Hobby-ITサイトの
《概要編》ページから
ダウンロード可能

NO	項目	数量	イメージ	商品名	URL	購入先	価格	備考
1	ESP32開発ボード	1		ＥＳＰ３２－ＤｅｖＫｉｔＣ－３２Ｅ　ＥＳＰ３２－ＷＲＯＯＭ－３２Ｅ開発ボード　４ＭＢ	https://akizukidenshi.com/catalog/g/gM-15673/	秋月電子	1600	19Pin×2列仕様 （他社は15Pin×2列）
2	ブレッドボード６穴版　ＥＩＣ－３９０１	1		ミニブレッドボード　ＢＢ－６０１（白）	https://akizukidenshi.com/catalog/g/gP-12366/		460	
3	抵抗10オーム	3		カーボン抵抗（炭素皮膜抵抗）１／２Ｗ１０Ω　（１００本入）	https://akizukidenshi.com/catalog/g/gR-07795/		100	赤外線送信LED用
4	抵抗200オーム	2		カーボン抵抗（炭素皮膜抵抗）１／２Ｗ２００Ω　（１００本入）	https://akizukidenshi.com/catalog/g/gR-07807/		100	緑LED及びトランジスタ用
5	緑LED	1		３ｍｍ黄緑色ＬＥＤ　５７０ｎｍ７０度　ＯＳＧ８ＨＡ３Ｚ７４Ａ	https://akizukidenshi.com/catalog/g/gI-11637/		10	状態表示用
6	赤外線受信	1		赤外線リモコン受信モジュールＯＳＲＢ３８Ｃ９ＡＡ（２個入）	https://akizukidenshi.com/catalog/g/gI-04659/		100	
7	赤外線送信LED	3		５ｍｍ赤外線ＬＥＤ　９４０ｎｍＯＳＩ５ＬＡ５１１３Ａ　グレー（１０個入）	https://akizukidenshi.com/catalog/g/gI-12612/		100	
8	トランジスタ	1		トランジスタ　２ＳＣ２６５５Ｌ－Ｙ－Ｔ９Ｎ－Ｂ　５０Ｖ２Ａ（１０個入）	https://akizukidenshi.com/catalog/g/gI-08746/		130	赤外線送信LED用
9	ブレッドボードジャンパー	1		ブレッドボード・ジャンパーワイヤ　１４種類×５本	https://akizukidenshi.com/catalog/g/gP-02315/		300	
総合計							**2,900**	秋月電子は送料+500円必要

1－3．開発環境Arduinoについて

開発環境はArduinoを利用していきます。



ESP32 Development Board

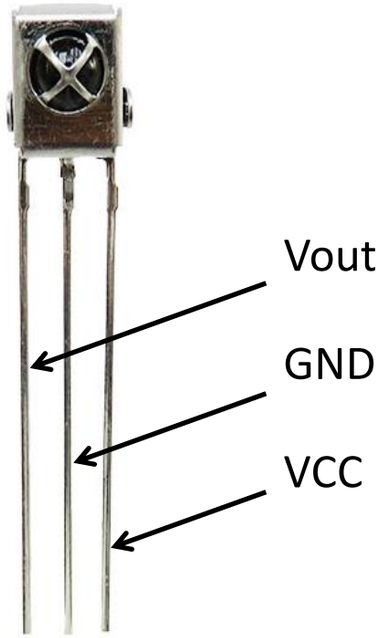
MicroUSB
Cable

P C (Windows/Mac/Linux)

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

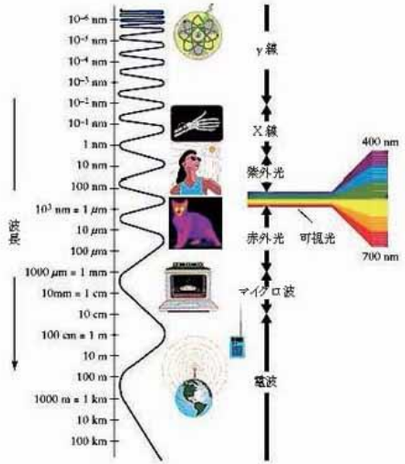
２－１．赤外線受信センサの仕組み

赤外線を受信するとVout端子がLOW（電圧）になる。（信号無しがHIGH）
このHIGH,LOWの信号を、ESP32のポートで認識して信号を取得する。



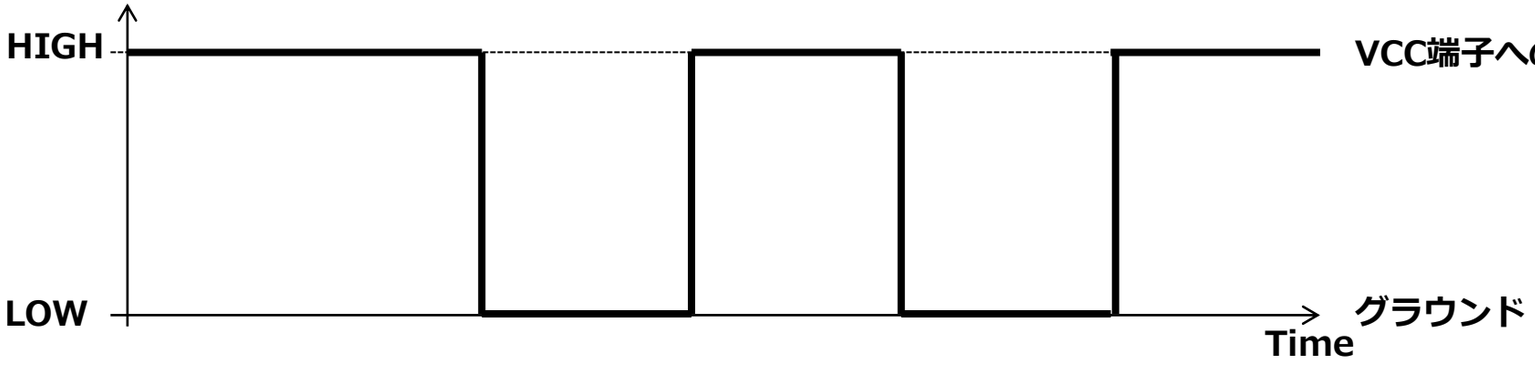
赤外線
（光と同じ電磁波の一種）

赤外線を受信すると
Voutポート電圧がLOWになる。
（信号なしがHIGH）



*1：電磁波と周波数の関係

Vout
出力電圧



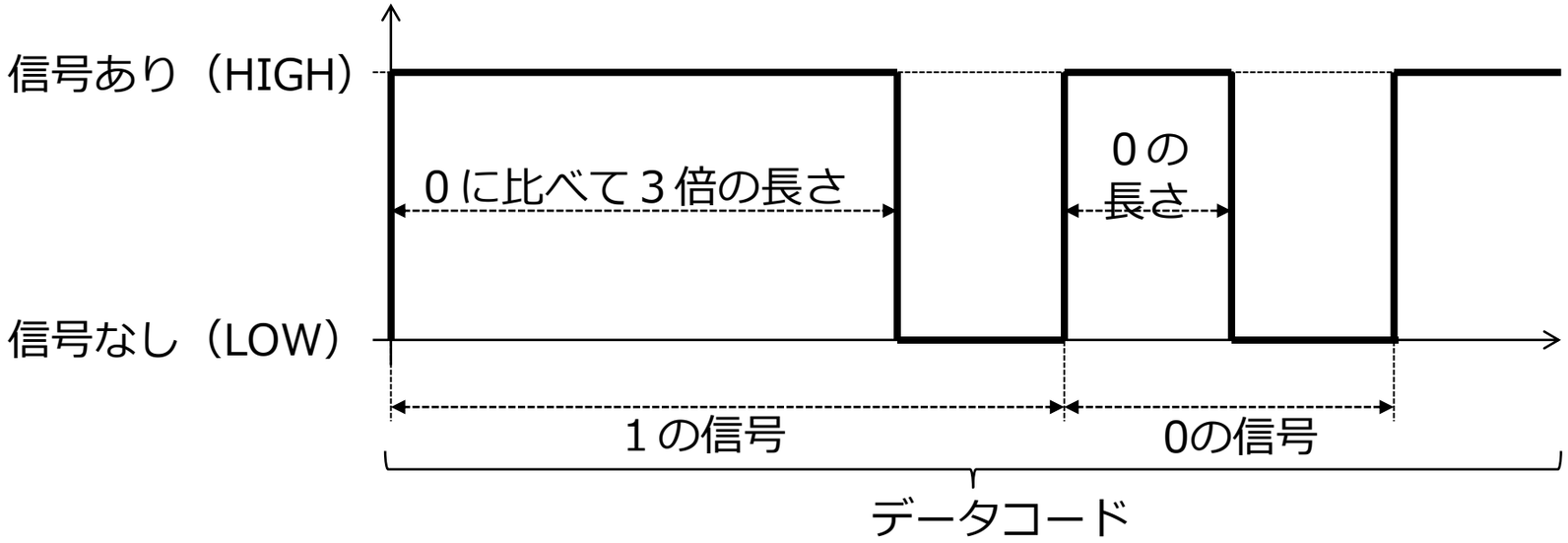
【配線】
・Vout：ESP32受信端子
・GND：ESP32のグランド端子
・VCC ：ESP32の3.3V端子

*1：https://www.jaima.or.jp/jp/analytical/basic/spectroscopy/world/

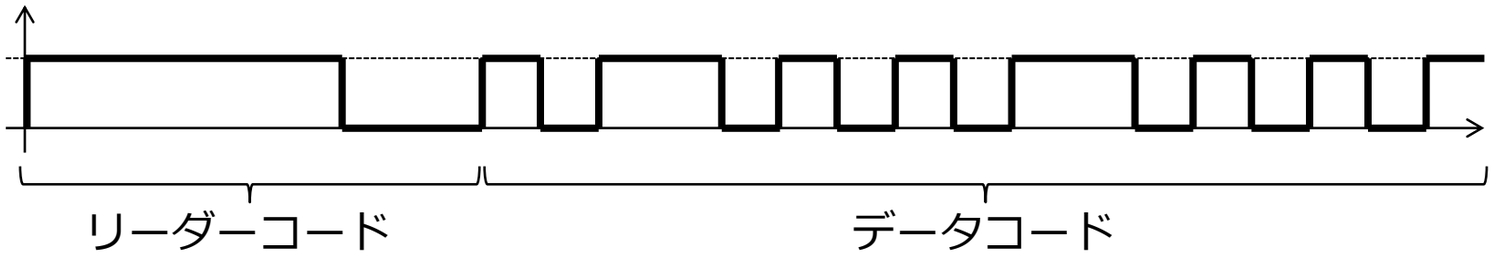
2－2．リモコン信号

0と1の識別はHIGH（又はLOW）の長さの違いで表している。

【0，1の信号例】

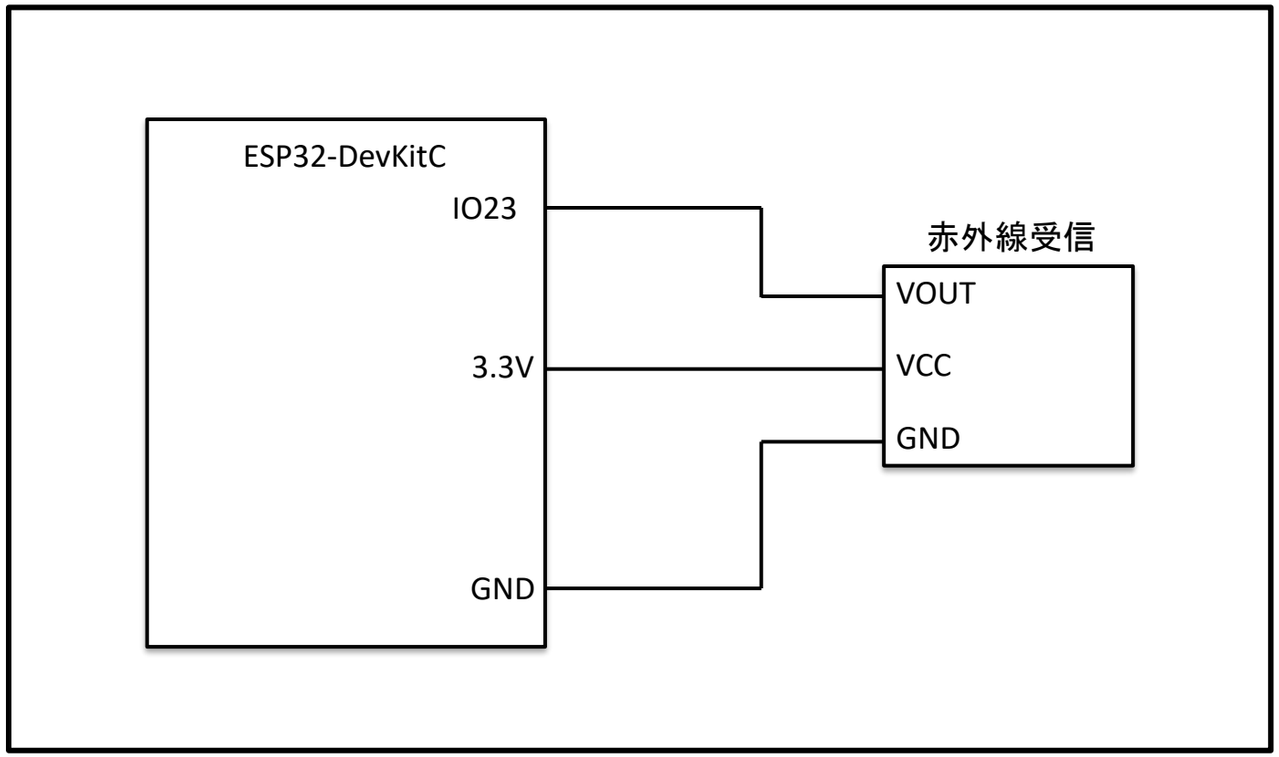


【参考：一般的なリモコン信号】



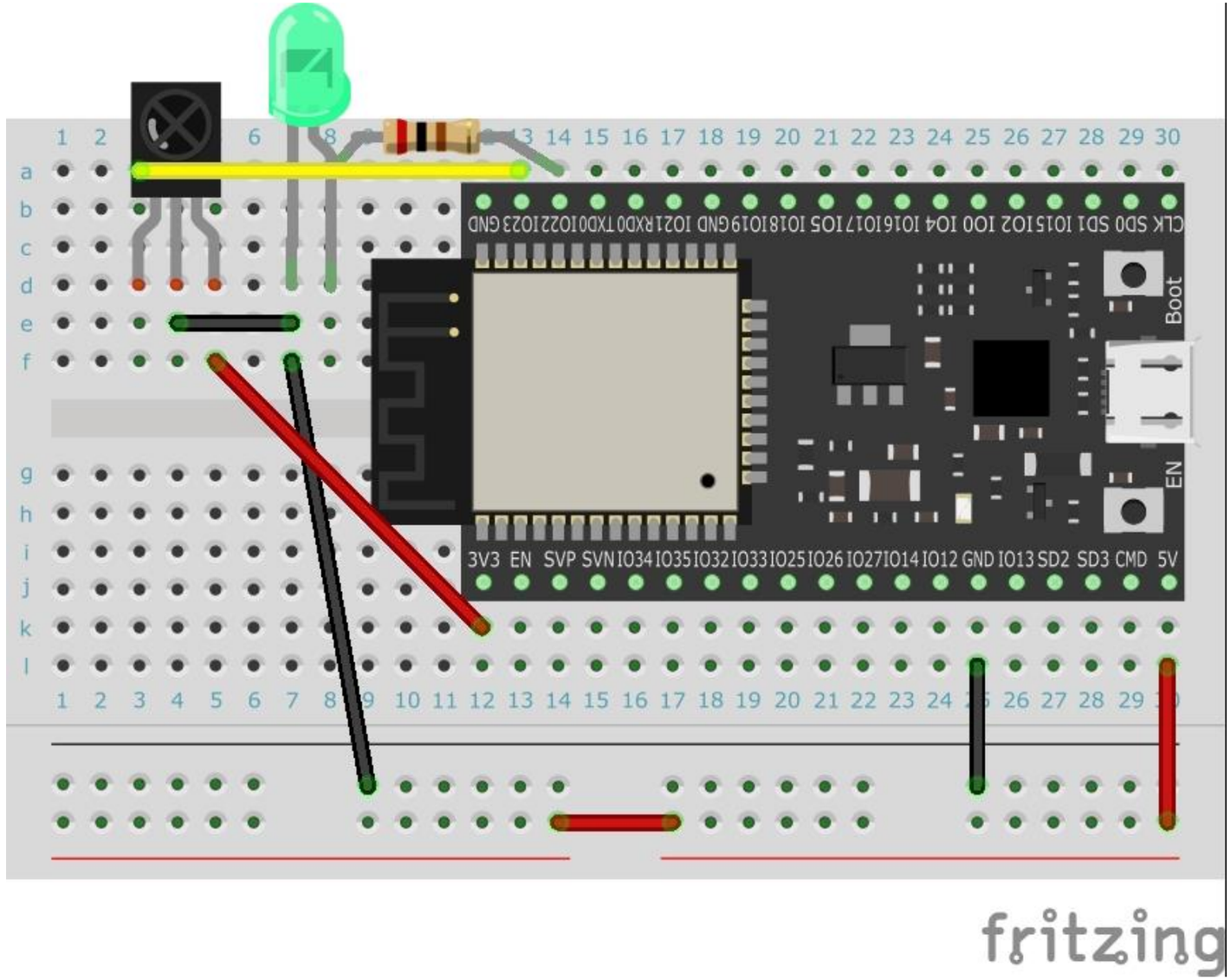
3．回路図

ESP32のIO23を受信設定し、赤外線受信センサのVoutからの信号を取得



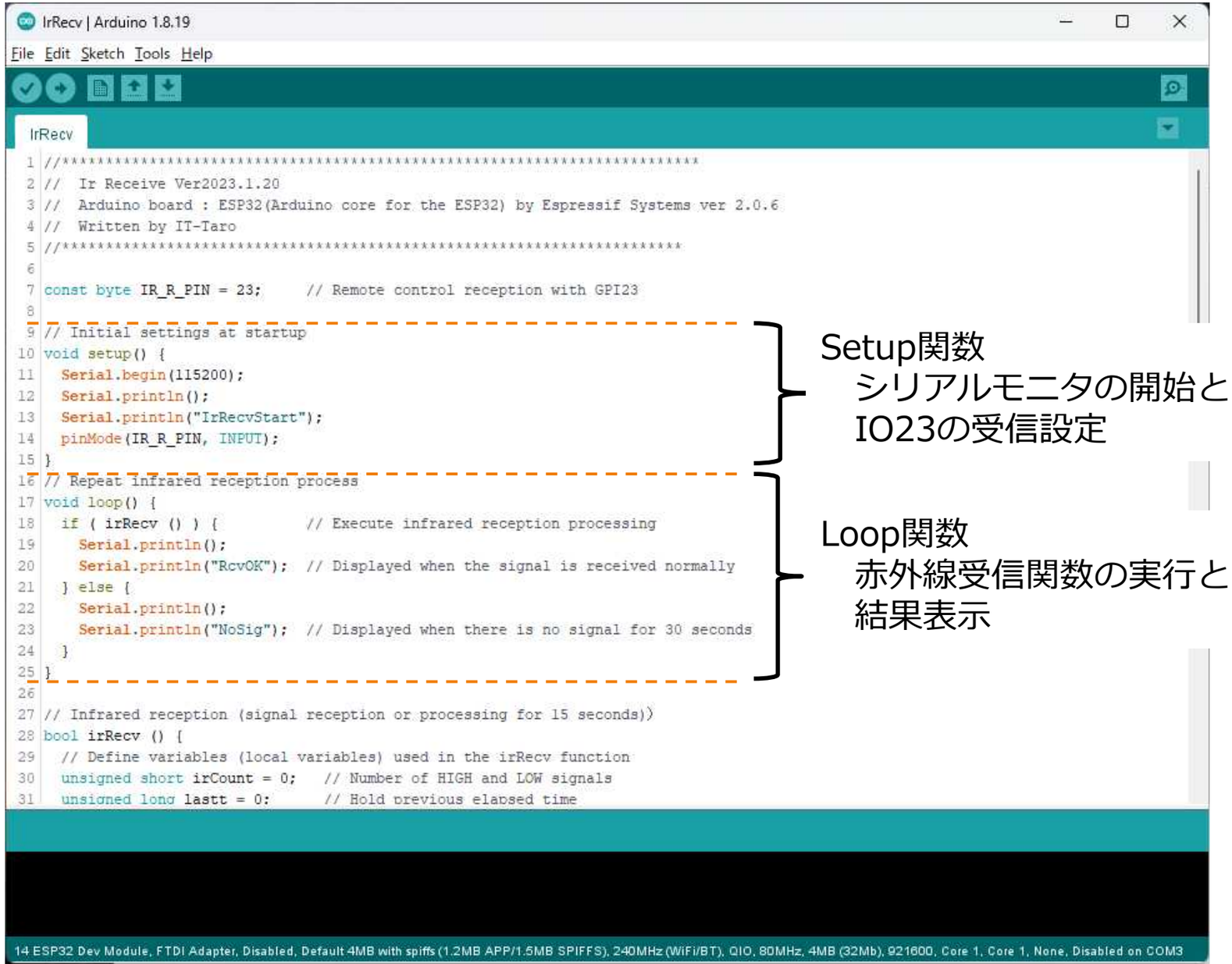
４．配線図

注）LEDの配線は《LED編》で実施しています。



5－1．ソフトウェア

注）LED編で基本を学習していますので赤外線受信センサに関するソフトウェア部分に関して理解していきます。



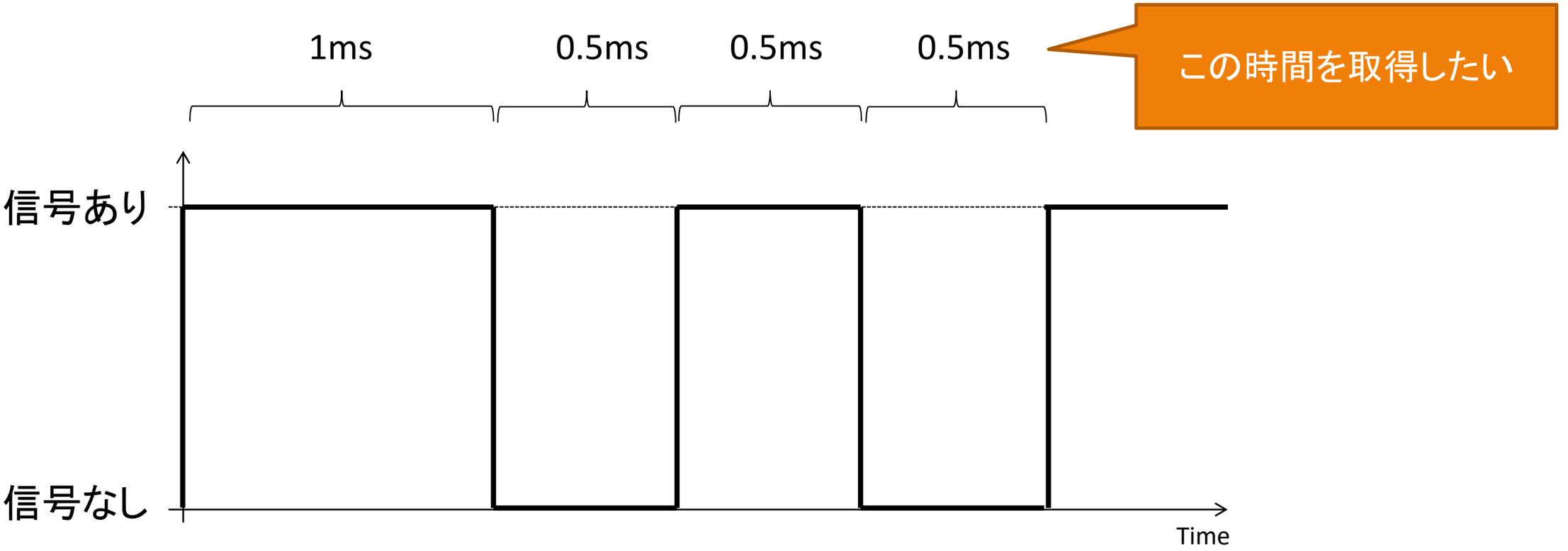

```
//*****************************************************************************
//  Ir Receive Ver2023.1.20
//  Arduino board : ESP32(Arduino core for the ESP32) by Espressif Systems ver 2.0.6
//  Written by IT-Taro
//*****************************************************************************

const byte IR_R_PIN = 23;     // Remote control reception with GPI23

// Initial settings at startup
void setup() {
  Serial.begin(115200);
  Serial.println();
  Serial.println("IrRecvStart");
  pinMode(IR_R_PIN, INPUT);
}
// Repeat infrared reception process
void loop() {
  if ( irRecv () ) {          // Execute infrared reception processing
    Serial.println();
    Serial.println("RcvOK");  // Displayed when the signal is received normally
  } else {
    Serial.println();
    Serial.println("NoSig");  // Displayed when there is no signal for 30 seconds
  }
}

// Infrared reception (signal reception or processing for 15 seconds))
bool irRecv () {
  // Define variables (local variables) used in the irRecv function
  unsigned short irCount = 0;   // Number of HIGH and LOW signals
  unsigned long lastt = 0;      // Hold previous elapsed time
```

Setup関数
シリアルモニタの開始と
IO23の受信設定

Loop関数
赤外線受信関数の実行と
結果表示

5－2．ソフトウェアで実現したいこと

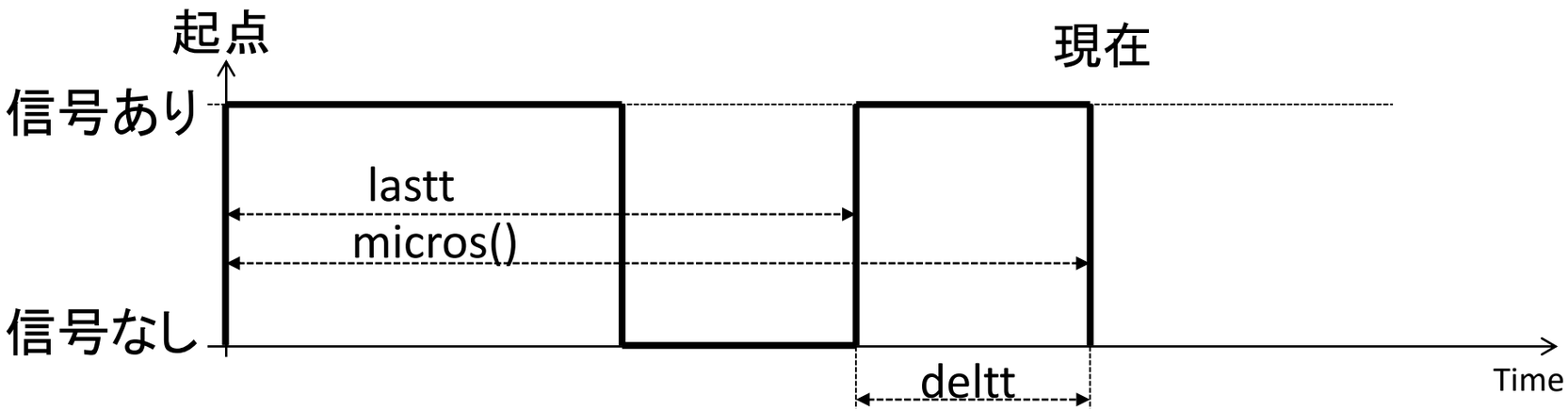
リモコン信号の「あり」と「なし」の時間の長さを取得する



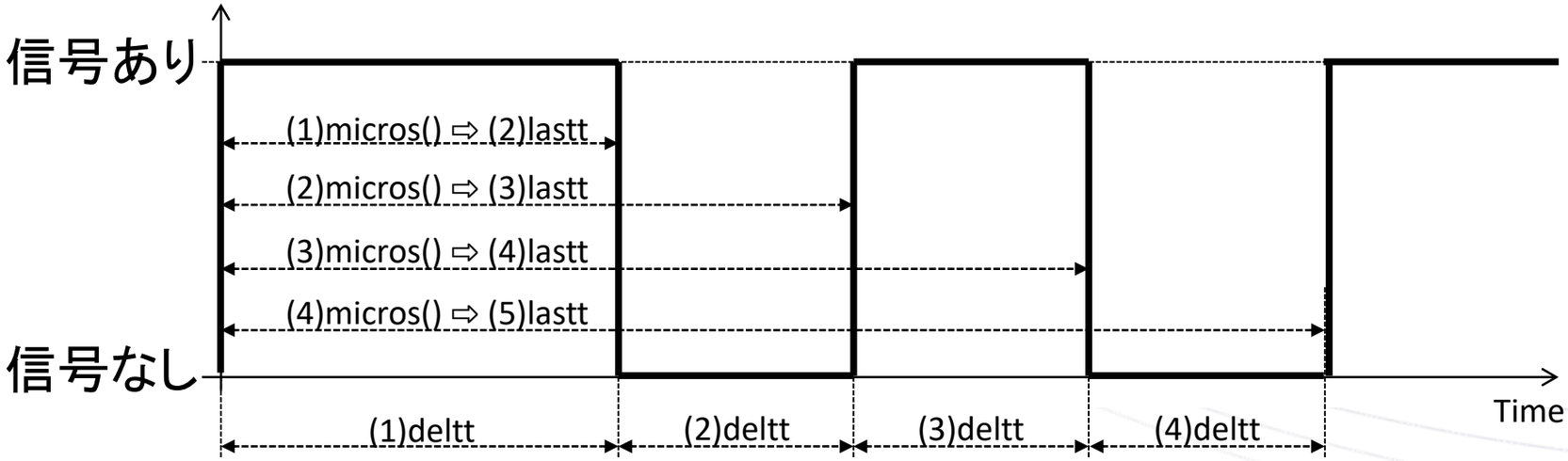
●現在の変更から前回の変更までの差分

現在の時間　**前回の時間**

この時間を取得　＝　micros()　－　lastt



●現在の時間を前回の時間に置き換えて計算していく



最後は0.5sec変更がないと終了と判断

5 – 2．ソフトウェアの理解

```
27 // Infrared reception (signal reception or processing for 15 seconds))
28 bool irRecv () {
29   // Define variables (local variables) used in the irRecv function
30   unsigned short irCount = 0;   // Number of HIGH and LOW signals
31   unsigned long lastt = 0;      // Hold previous elapsed time
32   unsigned long deltt = 0;      // Difference time from previous one
33   unsigned long sMilli;         // Start time of this process
34   unsigned long sMicro;         // Processing start time
35   unsigned long wMicro;         // wait start time
36   bool rState = 1;              // Infrared receiver module status 0: LOW, 1: HIGH
37   sMilli = millis();            // Get the current system time (get in milliseconds)
38   // Infinite loop until specific condition (signal received or 15 seconds elapsed)
39   while(1) {
40     // Get start time to wait for Ir reception
41     wMicro = micros();          // Get current system time (get in microseconds)
42     // Waiting for reception of inverted signal
43     while (digitalRead(IR_R_PIN) == rState) {
44       // When 0.5 seconds or more have passed after starting to wait
45       if (micros() - wMicro > 500000) {
46         // After waiting for 0.5 seconds or more
47         if ( irCount > 10 ) {
48           return true;    // Successfully completed
49         }
50         // If there are not more than 10 0,1 signals, receive again from zero due t
51         irCount = 0;
52         delay(1); // For watchdog timer (must be reset within 3 seconds)
53       }
54       // After 15 seconds or more of processing, T.O.
55       if ( millis() - sMilli > 15000 ) {
56         return false;    // Ends after 15 seconds (no reception)
57       }
58     }
59     // Get the current time and elapsed time at the start of signal reception
60     if ( irCount == 0 ) {
61       sMicro = micros();
62       lastt = 0;
63       irCount++;
64       Serial.println("ir:");
65     // Processing after starting signal reception processing (irCount is 1 or more)
66     } else {
67       // Calculate the elapsed time from the time when the status change of the infrared receiver last changed
68       deltt = ( (micros() - sMicro)/ 10 ) - lastt;
69       // Save the last changed elapsed time for the next elapsed time calculation
70       lastt = lastt + deltt;
71       irCount++;
72       Serial.print(deltt);
73       Serial.print(",");
74     }
75     // Change the value to detect state change in While next time
76     rState = !rState;
77   }
78 }
```

利用する
変数の宣言

①
While処理の
繰り返し

【主な変数とシステム関数】

起点時間　　：　sMicro　(Start Time)
前回の変更　　：　last　　(Last Time)
H,L変更の回数　：　irCount

millis()　　：　システム経過時間 ミリ秒　　←　0.5秒や15秒
maicros()　：　システム経過時間 マイクロ秒　←　リモコン信号

0.5秒以上変化なし

10回以上の信号は成功として
正常終了

10回以下の信号は雑音の可能
性があるためクリア

15秒経過で失敗終了

②
HIGH、LOWに変更が
あるまで繰り返し処理

初回変更はStartTime保持など

2回目以降は前回との差分時間を
取得し、シリアルモニタへ表示

HIGH、LOWに変更が
あった場合に処理
②のWhileを抜けて処理さ
れる。