【赤外線送信LED編】

スマートリモコン製作

- 赤外線送信LED、トランジスタを理解
- リモコン信号（３８kHz変調）をプログラム

目　次　《赤外線送信LED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	○	○	今回はこの動画
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）	-	○	別動画で配信
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。	-	○	

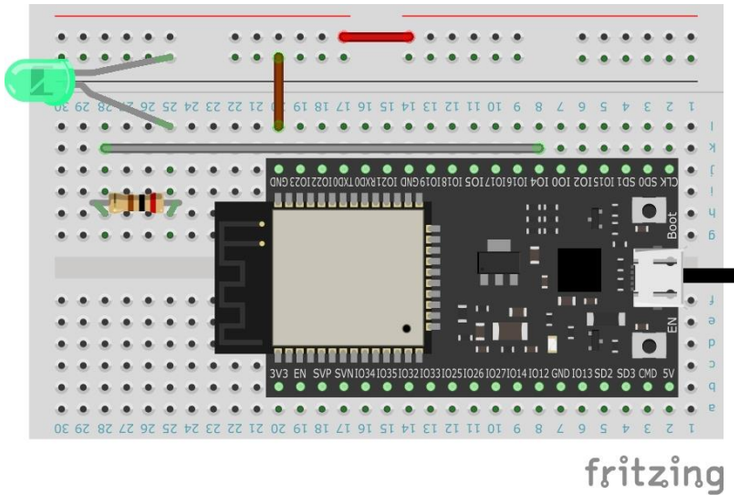
１－２．スマートリモコン製作の利用物品一覧

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/		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/	秋月電子	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　９４０ｎｍ ＯＳＩ５ＬＡ５１１３Ａ　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．赤外線送信LED

赤外線送信LED「OSI5LA5113A」は、順電圧が1.35Vで順電流が100mAで利用可能

- “OSI5LA5113A” Data Sheet

▪Absolute Maximum Rating (Ta=25℃)

Item	Symbol	Value	Unit
DC Forward Current	I_F	100	mA
Pulse Forward Current#	I_{FP}	1000	mA
Reverse Voltage	V_R	5	V
Power Dissipation	P_D	180	mW
Operating Temperature	Topr	-30 ~ +85	℃
Storage Temperature	Tstg	-40~ +100	℃
Lead Soldering Temperature	Tsol	260℃/5sec	-

Pulse Width ≦100us, Duty ≦1/100

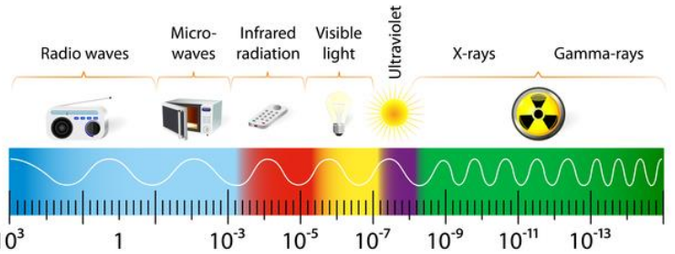
▪Electrical -Optical Characteristics (Ta=25℃)

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
DC Forward Voltage*1	V_F	I_F=100mA	-	1.35	1.6	V
DC Reverse Current	I_R	V_R=5V	-	-	10	μA
Peak Wavelength*2	λ_P	I_F=50mA	-	940	-	nm
Radiant Power*3	Po	I_F=50mA	-	12	-	mW
Radiant Intensity*4	Ie	I_F=50mA	35	55	-	mW/Sr
50% Power Angle	$2\theta_{1/2}$	I_F=50mA	-	15	-	deg

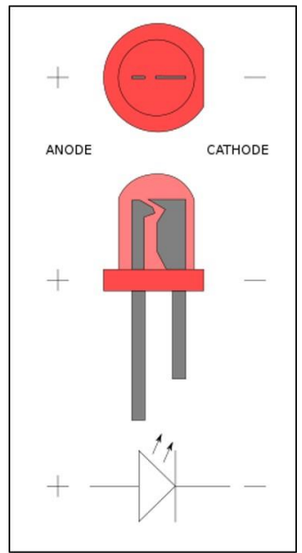
*1 Tolerance of measurements of forward voltage is ±0.1V
*2 Tolerance of measurements of peak wavelength is ±1nm
*3 Tolerance of measurements of Radiant Power is ±15%
*4 Tolerance of measurements of Radiant Intensity is ±15%

赤外線
（光と同じ電磁波の一種）

順電圧　：　1.35 V
順電流　：　100mA



*1 : https://k-comfort.co.jp/post-knowledge-infrared-1/

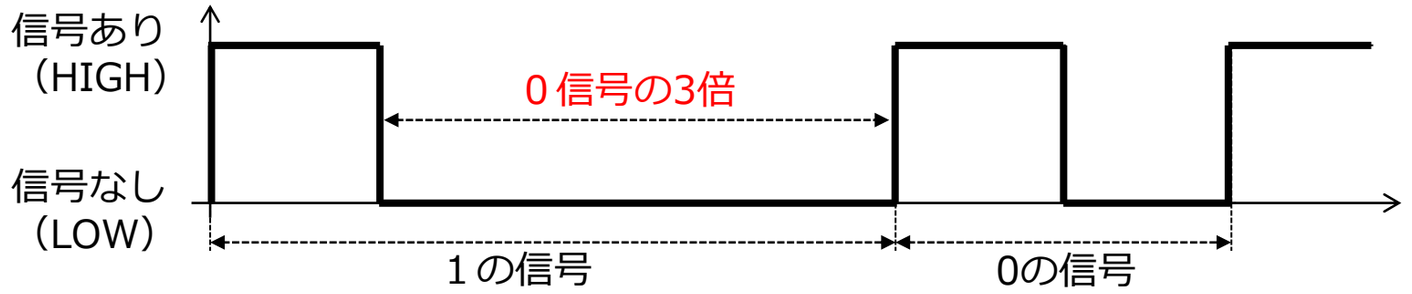


【注意】
足が長い方がアノードで
プラスを配線

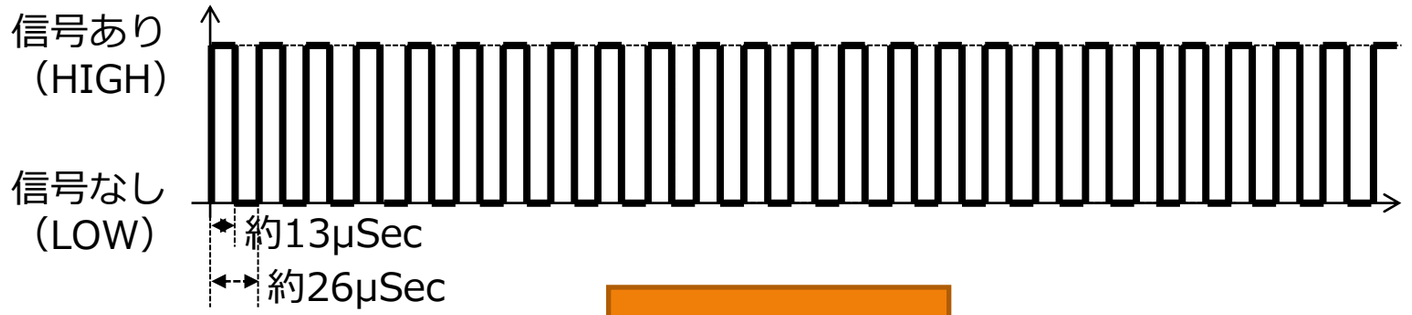
2－2．リモコン信号

リモコン信号の０，１は、HIGH又はLOWの時間を長くすることで表している。
実際の送信信号は、38KHzの信号で変調されている。

送信信号
（変調前）

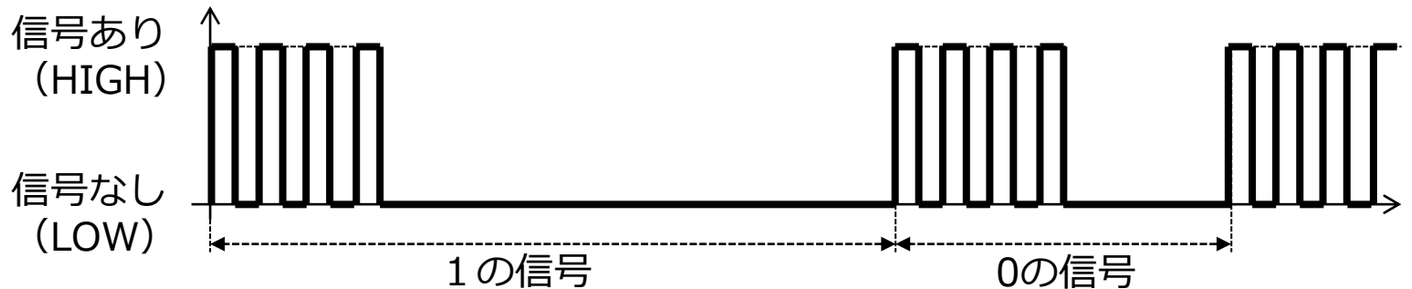


38KHzの搬送波



【赤外線周波数】
周波数：310～320 THz
波長：940～950 nm
（この赤外線を搬送波で変調）

実際の送信信号
（変調後）



【変調理由】（想定）
・最大出力を大きくし距離を出す
・雑音の影響を軽減する
・消費電力を下げる
・LEDの寿命を延ばす

２－３．抵抗値計算（赤外線送信LED）

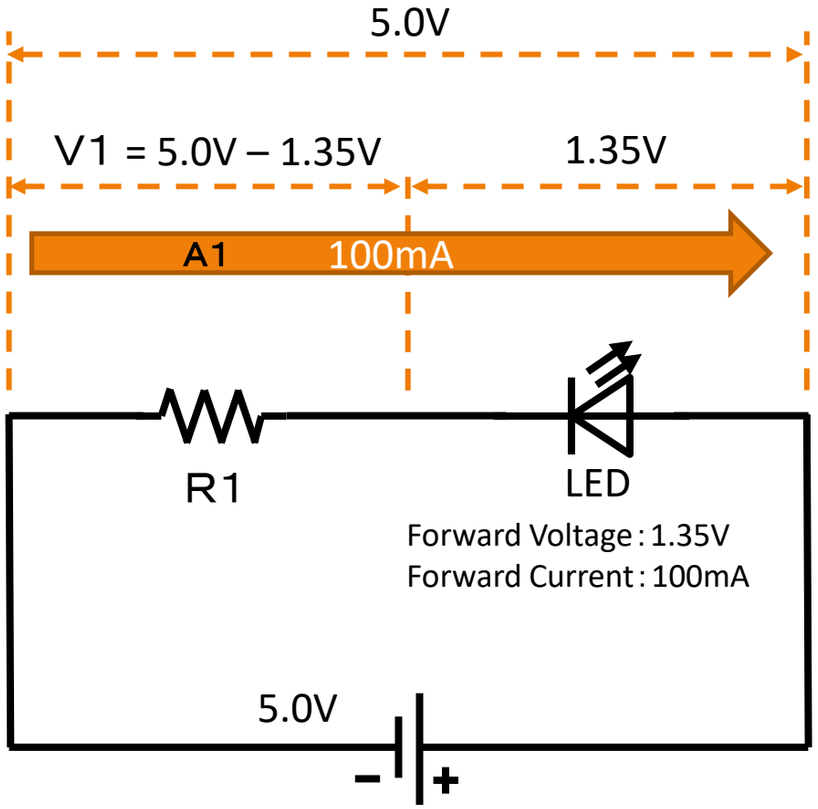
検討中

抵抗値は小さいとLEDを壊れてしまい、大きいとリモコンとして距離が出ないものになる。
信号のON/OFFを行っているため、定常的にかかる電流値を順電流として抵抗値は10Ωを採用。

● 抵抗値計算

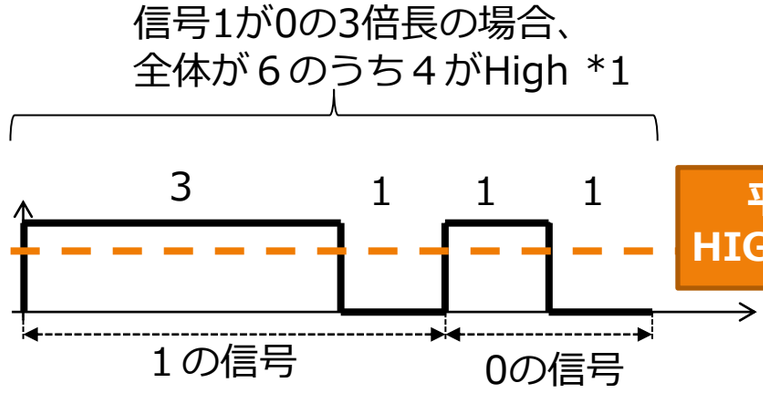
順電圧　：　1.35 V
順電流　：　100mA

$$R1 = \frac{V1}{A1} = \frac{5.0\,V - 1.35\,V}{100mA} = 36.5\,\Omega$$



● 信号のON/OFFを考慮

*1: 0,1のビット数が同等と仮定



平均すると
HIGH時間は2/3

Duty [HIGH(ON)の全体割合]

$$\Rightarrow \frac{4}{6} = \frac{2}{3} \Rightarrow \frac{1}{3}$$

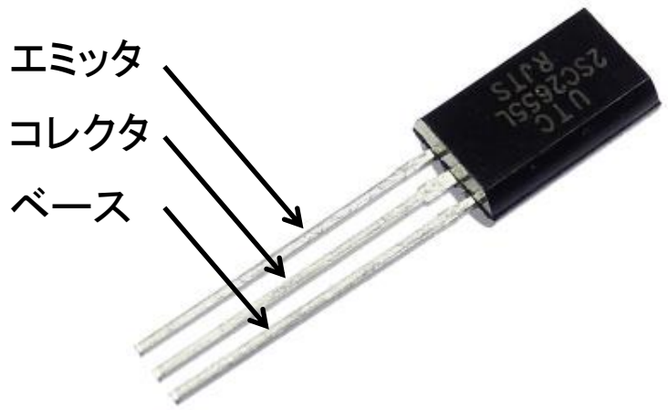
38KHz変調
なので、1/2

信号のON/OFFを考慮し平均100mAを流すとして計算

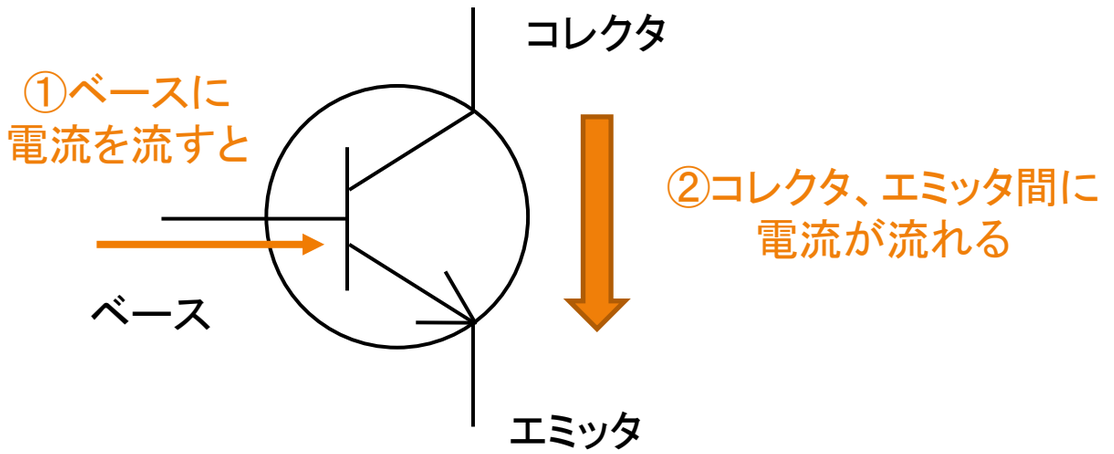
$$\Rightarrow 36.5\,\Omega \times \frac{1}{3} \fallingdotseq 10\,\Omega$$

２－４. トランジスタ

● トランジスタ

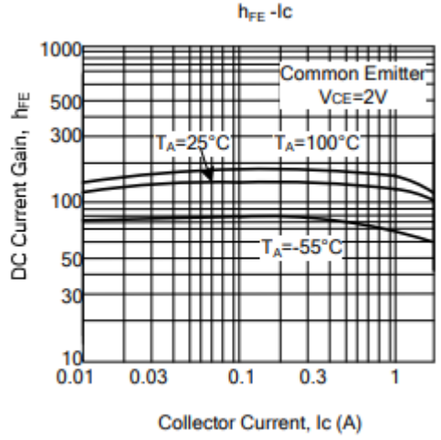


● トランジスタの動作



● トランジスタのベース抵抗計算

“2SC2655L” Data Sheet



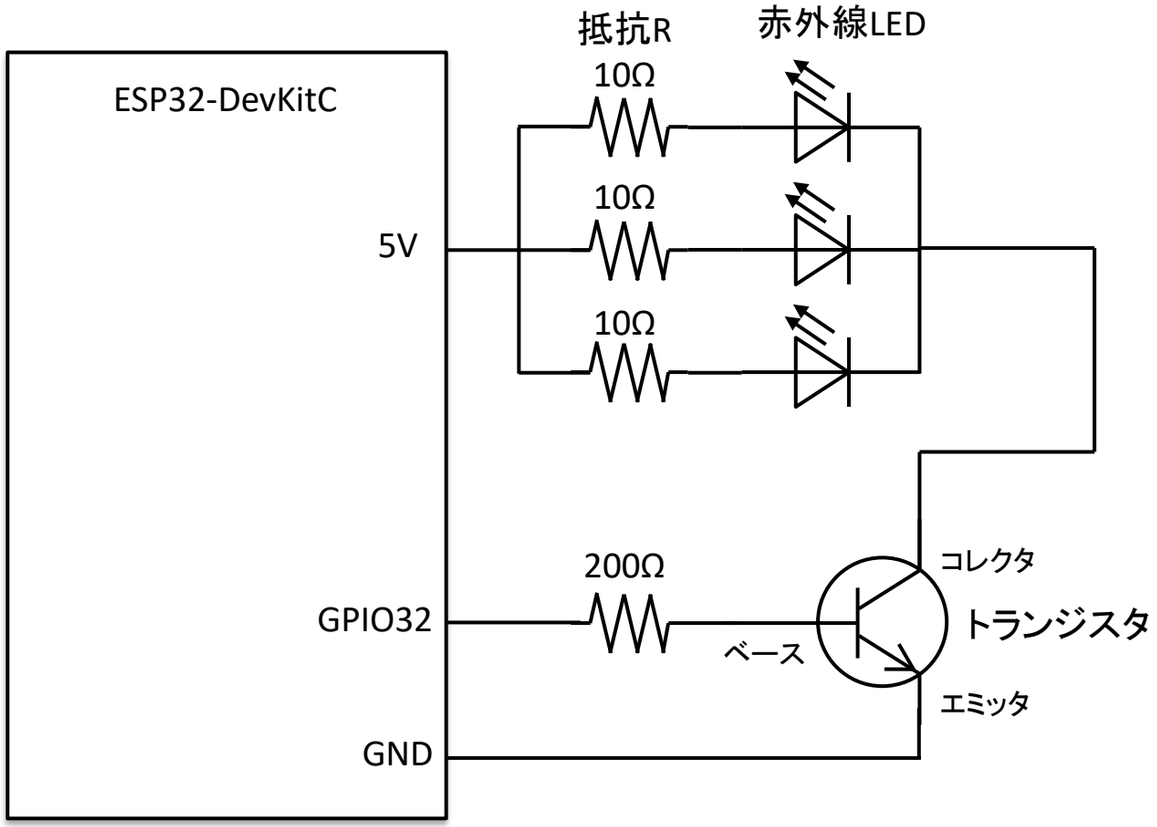
増幅率hFEはおおよそ100倍程度

《トランジスタ仕様》
Base- Emitter Saturation Voltage (VBE)：　1.2 V
DC Current Gain (hFE)：　100

【ベースに接続する抵抗】
①赤外線送信LEDに流したい電流【コレクタ～エミッタ間電流】
⇨　100mA(順電流) × 3LED × 3 *1 = 900 mA
**　　*1:PWMを考慮し瞬間的に3倍流す**
②ベースに流したい電流　900mA / 100(hFE)
⇨　9 mA
③ベースに設置する抵抗
(3.3 V – 1.2 V) / 9 mA = 0.23KΩ ⇨ 約 200Ω

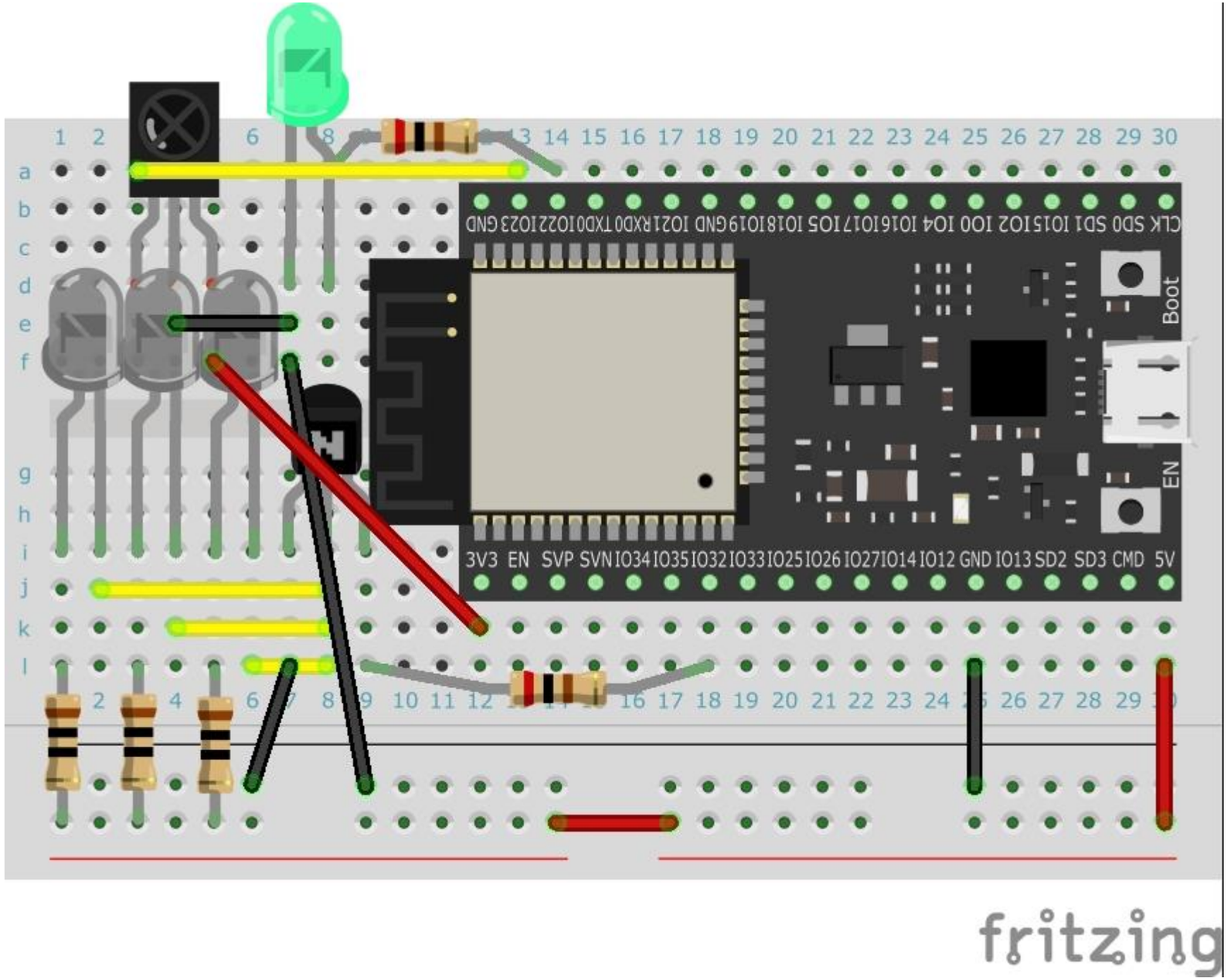
3．回路図

ESP32のIO23をOUTPUT設定し、トランジスタのベースに電流を流すことで、
コレクタ、エミッタ間の流れる大きな電流を制御し、赤外線送信LEDをON/OFFする。



４．配線図

注）LED,赤外線受信センサは以前の動画で、既に
配線していますので追加で配線していきます。



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送信LEDに関するソフトウェア部分に関して理解していきます。

IrSend | Arduino 1.8.19

File Edit Sketch Tools Help

IrSend §

```
1  //*****************************************************************************
2  //  Ir Send Ver2023.1.22
3  //  Arduino board : ESP32(Arduino core for the ESP32) by Espressif Systems ver 2.0.6
4  //  Written by IT-Taro
5  //*****************************************************************************
6
7  const byte IR_S_PIN = 32;       // Transmission control with GPIO32
8  // Set transmission data
9  unsigned short irData[] = { 303,171,...(omission)...,43,44 };  // ### [Data rewrite required] ###
10
11 // Initial settings at startup
12 void setup() {
13   Serial.begin(115200);
14   Serial.println();
15   pinMode(IR_S_PIN, OUTPUT);
16
17   // Execute function to transmit infrared
18   irSend ();
19   Serial.println("SndOK");
20 }
21
22 void loop() {
23
24 }
25
26 // Infrared transmission processing
27 void irSend () {
28   // Local variable definition
29   unsigned short irCount = 0; // Number of HIGH and LOW signals
30   unsigned long l_now = 0;    // Hold transmission start time
31   unsigned long sndt = 0;     // Elapsed time from transmission start
32   // Calculate the number of HIGH and LOW signals
33   irCount = sizeof(irData) / sizeof(irData[0]);
34   // Acquire transmission start time
```

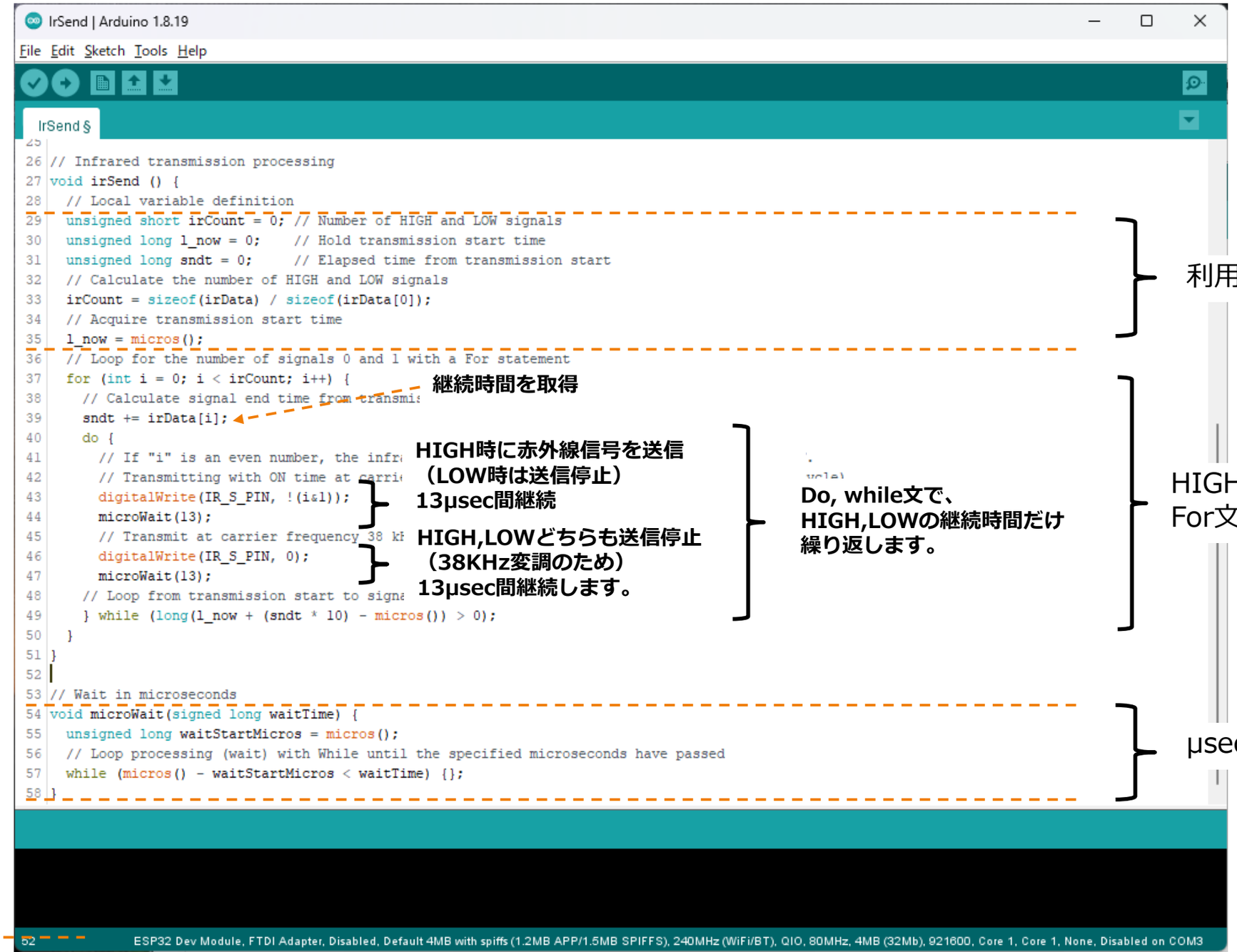
35　ESP32 Dev Module, FTDI Adapter, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, Core 1, Core 1, None, Disabled on COM3

赤外線受信センサを電子工作時に取得したリモコン信号を、ここに設定

Setup関数
　シリアルモニタの開始と
　IO32の出力設定
　赤外線送信関数irSendの実行

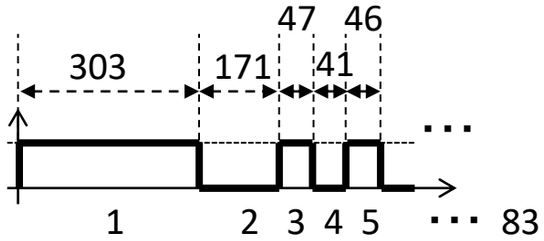
Loop関数
　処理なし

5 – 2．ソフトウェア


IrSend | Arduino 1.8.19
File Edit Sketch Tools Help
IrSend §
26 // Infrared transmission processing
27 void irSend () {
28 // Local variable definition
29 unsigned short irCount = 0; // Number of HIGH and LOW signals
30 unsigned long l_now = 0; // Hold transmission start time
31 unsigned long sndt = 0; // Elapsed time from transmission start
32 // Calculate the number of HIGH and LOW signals
33 irCount = sizeof(irData) / sizeof(irData[0]);
34 // Acquire transmission start time
35 l_now = micros();
36 // Loop for the number of signals 0 and 1 with a For statement
37 for (int i = 0; i < irCount; i++) {
38 // Calculate signal end time from transmis
39 sndt += irData[i];
40 do {
41 // If "i" is an even number, the infr
42 // Transmitting with ON time at carri
43 digitalWrite(IR_S_PIN, !(i&1));
44 microWait(13);
45 // Transmit at carrier frequency 38 kH
46 digitalWrite(IR_S_PIN, 0);
47 microWait(13);
48 // Loop from transmission start to signa
49 } while (long(l_now + (sndt * 10) - micros()) > 0);
50 }
51 }
52
53 // Wait in microseconds
54 void microWait(signed long waitTime) {
55 unsigned long waitStartMicros = micros();
56 // Loop processing (wait) with While until the specified microseconds have passed
57 while (micros() - waitStartMicros < waitTime) {};
58 }
52
ESP32 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
継続時間を取得
HIGH時に赤外線信号を送信
（LOW時は送信停止）
13μsec間継続
HIGH,LOWどちらも送信停止
（38KHz変調のため）
13μsec間継続します。
Do, while文で、
HIGH,LOWの継続時間だけ
繰り返します。
利用する変数の宣言
HIGH,LOWの数だけ
For文で繰り返します。
μsec単位のwait処理

5－3．ソフトウェア

●赤外線受信センサで取得したデータ（HIGH,LOWの時間間隔）



unsigned short irData[] = { 303,171,47,41,46,41,46,128,46,128,・・・（省略）・・・,44,43,44 };

1　2　3　4　5　6　7　8　9　10 ・・・（省略）・・・ 81, 82,83 ⇒ 83個のHIGH,LOW

●HIGH,LOWの数

irCount = sizeof(irData) / sizeof(irData[0]);

カンマで区切られた数を取得。つまり、HIGH,LOWの数 ➡ 83

●赤外線送信LED制御

digitalWrite(IR_S_PIN, !(i&1));

0：赤外線が送信されない
1：赤外線が送信される

“i”は0～(irCount-1)まで順番に処理されます。
“i&1”はビット演算で、アンド処理なので、偶数はゼロで奇数が1になります。

偶数

iの2進数	x	x	x	x	x	x	0	アンド	
1の2進数	0	0	0	0	0	0	1	➡	0

奇数

iの2進数	x	x	x	x	x	x	1	アンド	
1の2進数	0	0	0	0	0	0	1	➡	1

“!(i&1)”は上記を反転させるので、偶数が1,奇数がゼロになります。
そのため、偶数の時に送信、奇数の時に送信しないということになります。