ESP32-LEDPRO
(LED パネルコントローラ)
取扱説明書

マイクロファン
http://www.microfan.jp/
http://store.shopping.yahoo.co.jp/microfan/
https://www.amazon.co.jp/dp/B0BN681SQV

2023 年 11 月

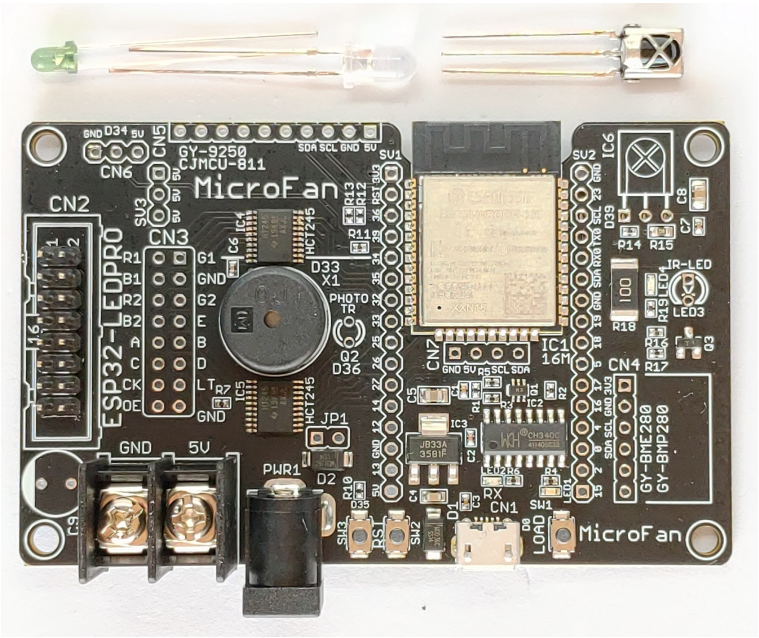
目次

表目次

図目次

第 1 章

ESP32-LEDPRO の紹介



1.1 製品概要

多数のカラー LED をマトリックス上に配置した LED パネルが、デジタルサイネージ（Digital Signage：電子看板）や、目を引く鮮やかな表示装置として使われ注目を集めています。ESP32-LEDPRO はこのようなデジタルサイネージなどを手軽に作成するための LED パネルの制御基板として開発されました。

HUB75 と呼ばれるインターフェースを使用する LED パネルは多数のカラー LED を並べた簡単な構造となっており、低価格であるという特徴がありますが、点灯制御の観点では、多数のカラー LED の色や明暗を制御するために、LED パネルに継続的に多量のデータを高速に送り込まなければならないため、その制御装置には高い性能が要求されます。

このような要求に応えるためには MCU の能力を最大限に活用したソフトウェアの開発が必要

で、様々な MCU 用に LED パネル制御用のソフトウェアが開発されています。

ESP32 にも以下に示す様に、LED パネルの制御用のソフトウェアがオープンソースで開発され公開されています。

- ESP32-HUB75-MatrixPanel-DMA
  `https://github.com/mrfaptastic/ESP32-HUB75-MatrixPanel-DMA`

制御用のスケッチは、ちらつきの少ない高いリフレッシュレートでの表示を行う一方で、表示に必要な CPU の負荷を低く抑えられるように、ESP32 に内蔵された DMA 機能を効果的に使用しています。

ESP32-LEDPRO は ESP-WROOM-32 を利用した LED パネル制御を手軽に行うための開発ボードとして開発されました。ESP32-LEDPRO は以下のような特徴を持っています。

- HUB75(E) インターフェースを使用する RGB LED マトリックスパネルを制御できます。
- ESP32E の 16M 版を搭載しています。
- ネットワークと接続するための WiFi や Bluetooth のネットワーク機能を利用できます。
- 電子工作で広く利用されている Arduino などの無償、便利、高機能な開発環境を利用してソフトウェアを開発できます。
- Arduino IDE で作成したスケッチを書き込むための USB インターフェースを装備しています。
- 圧電スピーカーを搭載しています。
- 明るさセンサーを搭載しています。
- 温度・湿度・気圧センサーなどを搭載することができます。
- 加速度・ジャイロセンサーを搭載することができます。
- 人感センサーを接続することができます。
- 赤外線リモコンの受光器を装備しています。
- 様々な情報を表示できる OLED ディスプレイ (別売) を搭載することができます。

1.2　購入・利用上の注意

ESP32-LEDPRO をご購入の際には、下記項目をご確認ください。

- 赤外線 LED の利用
  赤外線 LED を使用できるように設計されていますが、現在の基板は赤外線 LED の電流制限抵抗 R18 の指定が誤っているため、赤外線 LED の使用は推奨していません。赤外線 LED を使用する場合には、6.2 節をご参照ください。
- ESP-WROOM-32 の未接続端子
  内部のフラッシュメモリに接続されている ESP-WROOM-32 の 17-22 ピンは、使用上注意が必要なため未接続となっています。
- ESP-WROOM-32 の未引き出し端子

GPIO2 もしくは D2 は、LED1 に接続されていますが、基板の外に引き出す端子は割り当てられていません。

- GY-BME280/GY-BMP280 モジュールは別売りです。
- GY-MPU9250/GY-521(MPU6050)/CJMCU-811 モジュールは別売りです。
- HC-SR501 人感センサーは別売りです。
- OLED ディスプレイは別売りです。
  - `https://store.shopping.yahoo.co.jp/microfan/oled096-128x64-i2c-blue.html`
  - `https://www.amazon.co.jp/dp/B06Y4TKL1F`

1.3　マニュアルの記載内容に関して

ESP-WROOM-32 やそれに関連するハードウェアやソフトウェアは、機能の追加や改良が頻繁に行われているため、本文書で提供している情報は、ESP32-LEDPRO の購入者の利用時にはすでに古い情報になっている可能性があります。そのため、本文書で示している内容と異なる部分があったり、本文書で示している手順ではうまく動作しないことがあることと、その場合には、各自で対処方法を調査・確認していただく必要があることをご承知おきください。

本マニュアルの記載内容と、ご提供するソフトウェア、ハードウェアに差異がある場合には、ご指摘によりマニュアルの迅速な訂正を心がけますが、ご提供するソフトウェア、ハードウェアの現品の仕様が優先されます。

お伝えする内容と本質的な問題がない場合には、本マニュアルには、旧バージョンの製品の写真や他製品の写真などがそのまま使用されている場合がありますのでご承知おきください。

本書に記載されている内容に基づく作業、運用などにおいて、いかなる損害が生じても、弊社および著者をはじめとする本文書作成関連者は、一切の責任を負いません。

本文書に記載されている製品名などは、一般的にそれぞれの権利者の登録商標または商標です。

第 2 章

ESP32-LEDPRO の特徴

2.1 HUB75 コネクタ

ESP32-LEDPRO には、LED パネルを制御するために、HUB75 コネクタが装備されています。HUB75 コネクタで接続して制御できる LED パネルには、その点灯制御方式として代表的なものに、1/8 スキャン、1/16 スキャン、1/32 スキャンのパネルがあります。ESP32-LEDPRO は、広く利用されている 1/16 スキャンと、1/32 スキャンの LED パネルに対応しています。

LED パネルと ESP32-LEDPRO をフラットケーブルで接続するための一般的なコネクタとは別に、基板の裏面にピンソケットタイプのコネクタを取り付ける端子 CN3 が用意されています。CN3 にコネクタを接続すると、LED パネルの裏面の HUB75 コネクタ部分に ESP32-LEDPRO を直接取り付けることができます。

2.2 USB インターフェース

Arduino IDE で作成したスケッチを ESP-WROOM-32 に書き込むために、USB インターフェース (microB) を備えています。

USB インターフェースは以下のような目的で使用されます。

- ESP32-LEDPRO へのスケッチ（プログラム) の書き込み。
- ESP32-LEDPRO（LED パネルを除く）への電力供給。
- ESP32-LEDPRO と PC 間のシリアル通信。

2.3 電源回路

ESP-WROOM-32 に必要な 3.3V の電源は、(1) USB、(2) DC ジャックに接続された AC アダプタ、(3) ターミナルブロックに接続されたスイッチング電源のいずれかから取得された 5V の電源を電圧レギュレータで 3.3V に降圧して作成しています。

2.3.1　電圧レギュレータ

ESP-WROOM-32 は無線機能の利用時に 300mA 程度の電流を消費します。さらに、瞬間的ではありますが、突入電流として 1A 以上を消費することもあるようです。ESP32-LEDPRO で利用している電圧レギュレータ BL8071 は、少なくとも 1.5A 以上の電流を供給[*1]できますので ESP-WROOM-32 を余裕をもって稼働させることができます。

また、BL8071 の入力電圧から出力電圧のドロップダウンは 300mV 程度で、USB から電力を取得する場合、ショットキーダイオードの順方向電圧降下と合わせると電圧低下は 0.8V 程度となります。ESP-WROOM-32 が瞬間的に大きな電流を必要としている際に、USB からの供給電圧が定格の 5V をある程度下回っても、安定した電源電圧 3.3V を維持することができます。

2.3.2　LED パネルへの電力供給

LED パネルは接続されている枚数や表示パターンによりますが、大きな電力を消費します。

このため、USB 経由で PC から供給されている電力を LED パネルに供給すると、USB の規格を超えた大電力が必要とされることがあり、PC の USB 電流保護回路が十分でない場合には PC が壊れることがあります。

この様な問題を避けるため ESP32-LEDPRO では、USB からの電力を LED パネルへの電力供給に使用しないように設計されており、LED パネルへの電力供給は ESP32-LEDPRO に搭載された DC ジャックかターミナルブロックから得られた電力のみを使用するように構成されています。

このため、ESP32-LEDPRO を USB で PC に接続しただけの状態では、LED パネルに電力が供給されていないので LED パネルは点灯しません。LED パネルを正常に点灯させるためには、DC ジャックかターミナルブロックに電源を接続する必要があります。（実際には USB のみの接続でも、HUB75 の信号線から回り込んだ信号電圧が LED パネルの電源に回り込み、Vf が小さな LED パネルの赤色 LED が発光することがあります。）

- LED パネルの枚数が少なく表示パターンも電力を消費しないことが明らかな場合には、手軽な 5V の AC アダプタを使用して LED パネルに電源を供給できます。具体的には、ESP32-LEDPRO のターミナルブロックに LED パネルの電源ケーブルを接続し、DC ジャックに AC アダプタを接続します。
- LED パネルの電力消費が大きいことが想定される場合には、十分な能力を持ったスイッチングレギュレータを使用して LED パネルに電源を供給します。具体的には、ESP32-LEDPRO のターミナルブロックに LED パネルの電源ケーブルとスイッチングレギュレータからの電源ケーブルを接続します。

USB の電力は LED パネルに供給されませんが、LED パネル用に DC ジャックかターミナ

[*1] ただし USB2.0 からの供給電流は最大で 500mA、USB3.0 からは 900mA です。また、放熱の制限で継続して 1.5A の電流を使用することはできません。

ルブロックに電源が供給され、なおかつ USB が接続されていない場合には、その電力は ESP-WROOM-32 とその周辺回路に供給されます。このため、ESP32-LEDPRO のプログラミング終了後は、USB 接続を切った状態で ESP32-LEDPRO を運用することができます。

2.4　OLED ディスプレイ

ESP32-LEDPRO には OLED ディスプレイの接続端子が装備されているため、手軽かつ安定して接続し、利用することができます。

OLED ディスプレイの端子と表示モジュールに関しては、6.7.4 節をご参照ください。

ESP32-LEDPRO では、上方の表示は基本的に LED パネルに行うことができますが、システムの運用上の様々な管理情報を表示したい場合には、この OLED ディスプレイが役立ちます。

OLED ディスプレイは、128x64 ドットのグラフィックディスプレイになっており、ボードの稼働状態や利用者に伝えたい情報を、画像や文字で分かり易く表示できるようになります。

ネット上などで公開されている ESP-WROOM-32 のサンプルスケッチでは、IP アドレスや様々な情報を PC 上でシリアルモニタに表示する例が多いですが、実際の運用では ESP-WROOM-32 を PC に接続して使用することは少ないため、運用時に必要な情報を確認することができないという問題があります。

ESP32-LEDPRO では、面倒な配線等を行うことなく開発ボード上に OLED ディスプレイを搭載できるため、PC と切り離して単独で運用している場合でも、様々な情報を OLED に表示し確認することができます。

第 3 章

利用の準備

ESP32-LEDPRO の利用に先立って、必要に応じて、フォトトランジスタや赤外線受光器のはんだ付けを行います。

ピンヘッダやコネクタ（ピンソケット）等は、必要に応じて別途ご入手ください。

3.1 部品表

ESP32-LEDPRO キットの部品表を表 3.1 に示します。基板が破損している場合には、ご利用になる前にマイクロファンにお問い合わせください。

表 3.1 部品表

部品	シンボル	規格等	個数
プリント基板	ESP32-LEDPRO	Rev.1	1
フォトトランジスタ	Q2	NJL7502L	1
赤外線 LED	LED3	OSI5FU5111C-40	1
赤外線受光器	IC6	TL1838	1

3.2 最初の動作確認

ESP32-LEDPRO は、製造時の基本的な動作確認として、LED1（青色）を点滅させるスケッチを書き込んで動作確認を行っています。

ESP32-LEDPRO をご購入なさったら、利用に先立ち基板を USB ケーブルで PC に接続してください。LED1 が点滅し、ESP32-LEDPRO が稼働することが確認できます。

ここで問題があれば、マイクロファンにお問い合わせください。

3.3 部品のはんだ付け

必要に応じて、赤外線受光器、フォトトランジスタなどのはんだ付けを行ってください。

3.4　ESP32-LEDPRO と LED パネルの接続

3.4.1　電源

LED パネルの電源ケーブルは、ESP32-LEDPRO のターミナルブロックに接続します。5V と GND の極性を間違わないように注意してください。

3.4.2　信号線

LED パネルに付属のフラットケーブルで、ESP32-LEDPRO と LED パネルを接続します。

LED パネルには、入力用と出力用の 2 つの HUB75 コネクタが配置されています。一般的に LED パネルの HUB75 の入力用のコネクタは、LED パネル背面の左側にあります。LED パネルの HUB75 の入力コネクタの位置を確認し、そのコネクタと ESP32-LEDPRO の HUB75 コネクタをフラットケーブルで接続します。ESP32-LEDPRO の HUB75 コネクタ側では基板の左側にフラットケーブルのコネクタの凸部分が来る様に注意してください。

3.4.3　複数の LED パネルの接続

HUB75 インターフェースの LED パネルは、芋づる式に接続して大きなパネルを構成できます。一般的に LED パネルの背面の右側にある HUB75 の出力コネクタと、次のパネルの背面の左側にある HUB75 の入力コネクタをフラットケーブルで接続します。必要に応じてこの作業を繰り返します。

3.4.4　HUB75 の E 信号端子の設定

使用する LED パネルが 1/16 スキャンか 1/32 スキャンかによって、HUB75 の E 信号の接続が異なるため、ESP32-LEDPRO 裏面の SJ1 の適切な端子をはんだでショートさせる必要があります。

詳細は 6.5 節の SJ1 の項を参照してください。

3.5　ESP32-LEDPRO と電源の接続

3.5.1　スイッチングレギュレータ

LED パネルは大量の電力を消費するので、可能であれば、十分な容量を持った 5V のスイッチング電源を用意しそれを使用してください。スイッチング電源の電源ケーブルは、LED パネルの電源ケーブルを接続したのと同じターミナルブロックに接続します。

3.5.2　AC アダプタ

LED パネルが 1-2 枚で、表示パターンも電力を消費しないものの場合には、手軽な AC アダプタを使用して電力を供給することができます。AC アダプタの出力は ESP32-LEDPRO の DC ジャックに接続して使用します。

5V の AC アダプタは最大の物でも 4A 程度の容量のようなので、LED パネルの使用電流が AC アダプタの容量を超えないように十分に管理して使用してください。

第 4 章

Arduino スケッチ環境の整備

4.1 ESP32 用 Arduino 開発環境のインストール

Arduino の開発環境のインストールは以下の 2 段階の手順で行います。

- 基本となる Arduino IDE のインストール
- ESP32 用の開発機能の追加

この後は、必要に応じて、各種のライブラリの追加インストールを行います。

下記のインストール法がわかりにくい様であれば、WEB で検索をするとインストール法を示したページが複数見つかるので、ご自身がわかりやすいと思うページを参照してインストールを行ってください。

4.1.1 基本となる Arduino IDE のインストール

以下のページからダウンロードオプションで、ご自身が使用している OS 用のインストールパッケージを選択しダウンロードしインストールしてください。

`https://www.arduino.cc/en/software`

Arduino IDE がインストールできたら起動してください。

メニュー等を日本語化するために、Arduino IDE の [File] ⇒ [Preferences...] ⇒ [Settings] タブの [Language:] を日本語に設定してください。

4.1.2 ESP32 用の開発機能の追加

ESP32 用の Arduino は以下の WEB ページで公開されています。

`https://github.com/espressif/arduino-esp32`

インストール方法も示されているので、示されている手順に従って ESP32 用の Arduino のインストールを行ってください。

4.2　ESP32 用 Arduino 開発環境の設定

Arduino IDE のメニューの「ツール」を選択してメニューを表示してください。このメニューの中から、まず、開発用のボードと PC と ESP32-LEDPRO の接続を行うポートを設定します。

開発ボードの選択は、[ボード:] メニューの esp32 の中から、初めの方に表示されている [ESP32 Dev Module] を選択してください。

また、[ポート:] は、 ESP32-LEDPRO が接続されているポートが COMX(X は数値) の形式で表示されているので、それを選択して設定してください。

MAC の場合には、ポートの選択方法は WEB などで確認してください。

最後に、フラッシュサイズに関連する以下の設定をメニューから行ってください。他の設定は既定の状態で問題ありません。

基本的には、16M のフラッシュサイズに適合させます。

- Flash Size
  16MB を選択
- Partition Scheme
  ２種類ある　[16M Flash (—-)]　のどちらかを選択

4.3　サンプルスケッチの実行

ESP32 用の Arduino をインストールすると、Arduino IDE の [ファイル] ⇒ [スケッチの例] に、ESP32 用の多くのサンプルスケッチが追加されます。これらのサンプルスケッチを試すことで、ESP32 のプログラミングを学ぶことができます。

ESP32-LEDPRO に初めから書き込まれている LED の点滅スケッチとかち合いますが、ここでは、ESP32-LEDPRO の動作確認のために、LED 点滅スケッチの実行（更新）を試してみましょう。

4.3.1　BLINK:LED の単純な点滅

電子工作界の hello world、LED の点滅スケッチを実行しましょう。

Arduino IDE の [ファイル] ⇒ [スケッチの例] ⇒ [01.Basics] から Blink を選択してください。ESP32-LEDPRO の LED は 2 番ピンに接続されているので、スケッチの pinMode(),digitalWrite() の第 1 引数の LED_BUILTIN を 2 に変更します。

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 2 as an output.
  pinMode(2, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(2, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(2, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

図 4.1　BLINK:LED の単純な点滅

4.3.2　スケッチのコンパイルと ESP32-LEDPRO への書き込み

スケッチの入力・修正が終わったら、まず問題なくコンパイルを行えるかどうか、Arduino IDE の左上部のチェックマーク [検証] のアイコンをクリックして、スケッチをコンパイルします。

問題なくコンパイルできたならば、先ほどのアイコンの右隣の右矢印マーク [書き込み] のアイコンをクリックします。スケッチの再コンパイルの後に、Arduino IDE の下部のメッセージエリアに白色の文字で多数の行のメッセージが出て、スケッチの書き込みが開始されます。

この時、ESP32-LEDPRO の LED2:RX が赤色で点滅し、スケッチの書き込みのための受信が行われていることを示します。

スケッチが ESP32-LEDPRO に正しく書き込まれたら、ボード上の LED が点滅します。

LED2:RX が点滅し、スケッチが更新されたことは確認できますが、先に書き込まれていたスケッチと同じなので、代り映えがせず本当にスケッチが更新されたかわかりにくいと思われる場合には、delay() の引数を変更して、LED の点滅の間隔などを変えてみるとよいでしょう。

4.4　ライブラリのインストール

ESP32-LEDPRO を最大限に活用するためには、LED パネルやセンサーを操作するためのいくつかのライブラリをインストールする必要があります。ここでは、LED パネルの制御用ライブラリと環境センサー BME280 のライブラリのインストール法を簡単に説明します。

4.4.1　ESP32-HUB75-MatrixPanel-DMA ライブラリ

ESP32-LEDPRO の機能の中核となる LED パネルの操作ライブラリとして、ESP32-HUB75-MatrixPanel-DMA ライブラリをインストールします。

ESP32-HUB75-MatrixPanel-DMA ライブラリは、Arduino IDE のライブラリマネージャを

利用してインストールすることができます。ライブラリマネージャの検索フィルタに [HUB75 DMA] を入力して絞り込むと、以下のような項目が表示されます。

- ESP32 HUB75 LED MATRIX PANEL DMA Display by Faptastic

インストールボタンが表示されるので、最新バージョンを選択して、ライブラリをインストールします。

このライブラリは、以下の URL で取得することもできます。

- `https://github.com/mrfaptastic/ESP32-HUB75-MatrixPanel-DMA`

ライブラリのインストール後、Arduino IDE メニューから [ファイル] ⇒ [スケッチの例] を選択すると、リストに [ESP32 HUB75 LED MATRIX PANEL DMA Display] フォルダが追加されているのが確認できます。このフォルダの中には、１０個程度のサンプルスケッチが格納されています。

サンプルスケッチの内容は、以下の URL でも確認できます。

- `https://github.com/mrfaptastic/ESP32-HUB75-MatrixPanel-DMA/blob/master/examples/README.md`

4.4.2　BME280 ライブラリ

BME280 用のライブラリは、Arduino IDE のライブラリマネージャを利用してインストールすることができます。BME280 のライブラリは多数提供されていますが、ここでは、[BME280 by Tyler Glenn] とタイトルが付けられているライブラリを選択してインストールします。

ライブラリマネージャの検索フィルタに [BME280 tyler] を入力して絞り込んでください。

最新バージョンを選択して、インストールボタンをクリックしライブラリをインストールします。

なお、このライブラリは、以下の URL で取得することもできます。

- `https://github.com/finitespace/BME280`

ライブラリのインストール後、Arduino IDE メニューから [ファイル] ⇒ [スケッチの例] を選択すると、リストに [BME280] フォルダが追加されているのが確認できます。BME280 フォルダの中を確認するといくつかのサンプルスケッチがあり、選択して実行することができます。

第 5 章

LED パネルの表示

下記のページもご参照ください。

- https://www.microfan.jp/2023/01/esp32-hub75-matrixpanel-dma/

5.1 パネルの機能検査

LED パネルの最初のプログラムとして、短く簡潔なプログラムで、実用的なプログラムを作ってみましょう。LED パネルを入手したら、まずすべての LED が点灯することと、RGB の LED が正しく発色することを確認する必要があります。その確認のプログラムを以下に示します。

以下のプログラムは、64 × 32 ピクセルの LED パネルを対象としています。

```
#include <ESP32-HUB75-MatrixPanel-I2S-DMA.h>

#define RES_X 64 // 使用する LED パネルの横方向のピクセル数
#define RES_Y 32 // 楯方向のピクセル数
#define CHAIN 1 // 使用する LED パネルの横方向の連結数

#define PIN_E 32 // HUB75 の E 端子の定義：規定値がないので必要に応じて定義する必要がある

MatrixPanel_I2S_DMA *dma_display = nullptr;

void setup() {
  HUB75_I2S_CFG mxconfig(RES_X, RES_Y, CHAIN);

  if (RES_Y == 64)
    mxconfig.gpio.e = PIN_E;  // 縦が 64 ピクセルで 1/32 スキャンのパネルを使用する場合

  // Display Setup
  dma_display = new MatrixPanel_I2S_DMA(mxconfig);
  dma_display->begin();
}

void loop() {
  dma_display->fillScreenRGB888(127, 127, 127); // 白は全 LED の点灯になり電流を消費するので、値を抑えています
  delay(2000);
  dma_display->fillScreenRGB888(255, 0, 0); // LED の発色強度は、赤、緑、青の順番で指定: 0-255
  delay(2000);
  dma_display->fillScreenRGB888(0, 255, 0);
  delay(2000);
  dma_display->fillScreenRGB888(0, 0, 255);
  delay(2000);
  dma_display->fillScreenRGB888(0, 0, 0);
  delay(1000);
}
```

図 5.1　パネルの機能検査

このプログラムを実行すると、LED パネルに問題がなければ、白、赤、緑、青の順番で LED パネル全面が光り、一旦すべて消えて、また、白、赤、緑、青の表示を繰り返します。

このプログラムを実行して、点灯していないピクセルや白色表示の際に白ではない色に見えるピクセルがある場合には、そのピクセルの LED が不良です。また、赤、緑、青の点灯順序が異なる場合には、色を指定する信号線の配置が間違っていることがわかります。

中国から LED パネルを仕入れていると、ピクセルの発光不良のほか、青と緑の信号線が間違っている製品に出くわすことがあります。LED パネルの入手時の受入検査は重要です。

使用するパネルが 64 × 32 ピクセル以外の場合、3,4 行目の RES_X, RES_Y の値を使用するパネルのピクセル数に変更します。

5.2　色の信号線の交換

我々が何度か経験した例ですが、緑色と青色の信号線が入れ変わっている LED パネルがありました。

この様な場合には、パネルの機能検査のスケッチを例とすると、以下のように色の信号線の割り当てを変更することにより、通常の LED パネルと同様に使えるようになります。

```
#include <ESP32-HUB75-MatrixPanel-I2S-DMA.h>

#define RES_X 64 // 使用する LED パネルの横方向のピクセル数
#define RES_Y 32 // 楯方向のピクセル数
#define CHAIN 1 // 使用する LED パネルの横方向の連結数

#define PIN_E 32 // HUB75 の E 端子の定義：規定値がないので必要に応じて定義する必要がある

#define R1 25 // 色の信号線の端子番号を定義
#define G1 26
#define BL1 27
#define R2 14
#define G2 12
#define BL2 13

MatrixPanel_I2S_DMA *dma_display = nullptr;

void setup() {
  HUB75_I2S_CFG mxconfig(RES_X, RES_Y, CHAIN);

  mxconfig.gpio.g1 = BL1; // デフォルトの色の信号番号を実際の信号番号の割り当てに変更
  mxconfig.gpio.b1 = G1;
  mxconfig.gpio.g2 = BL2;
  mxconfig.gpio.b2 = G2;

  if (RES_Y == 64)
    mxconfig.gpio.e = PIN_E;  // 縦が 64 ピクセルで 1/32 スキャンのパネルを使用する場合

  // Display Setup
  dma_display = new MatrixPanel_I2S_DMA(mxconfig);
  dma_display->begin();
}

void loop() {
  dma_display->fillScreenRGB888(127, 127, 127); // 白は全 LED の点灯になり電流を消費するので、値を抑えています
  delay(2000);
  dma_display->fillScreenRGB888(255, 0, 0); // LED の発色強度は、赤、緑、青の順番で指定: 0-255
  delay(2000);
  dma_display->fillScreenRGB888(0, 255, 0);
  delay(2000);
  dma_display->fillScreenRGB888(0, 0, 255);
  delay(2000);
  dma_display->fillScreenRGB888(0, 0, 0);
  delay(1000);
}
```

図 5.2　色の信号線の入れ替え

5.3　サンプルスケッチ

サンプルスケッチをいくつか実行してみましょう。

5.3.1　1_SimpleTestShapes

サンプルスケッチの [1_SimpleTestShapes] を選択してください。9,10 行目で使用するパネルサイズが 64x32 に設定され、11 行目でパネル 1 枚の使用が指定されています。

このスケッチをコンパイルして実行すると、初めに簡単な図形が表示され、次に文字列が表示されます。上 2 列の文字列と記号は、色を変えながら表示されます。

5.3.2　クロック信号の位相の変更

この [1_SimpleTestShapes] の文字表示の 3 行目には、LED MATRIX!と表示されます。しかしながら、使用するパネル（で使用されている LED 表示制御 IC）によっては、LED の L の文字の左側の縦棒が表示されないことがあります。

この様な場合には、スケッチの 104 行当たりでコメントアウトされている以下の行のコメントを外してください。

```
mxconfig.clkphase = false;
```

図 5.3　クロックの位相変更

この変更の後に再度コンパイル実行すると、LED の文字が正しく表示されるようになります。

この変更が有効だったパネルは、他のプログラムでもクロックの位相変更のコードを入れるようにしてください。

5.3.3　2_PatternPlasma

サンプルスケッチ [2_PatternPlasma] は、幻想的と言ったらいいのか、そんな感じの表示のデモです。

このスケッチは、62 行目のあたりに書かれているように、縦横 64 ピクセルの LED パネルを 2 枚接続した LED パネルが対象になっています。例えば、手持ちの LED パネルが 64x32 が 1 枚だけの場合には、PANEL_HEIGHT を 32 に、PANELS_NUMBER を 1 に変更してコンパイルします。64x32 の LED パネルを複数枚持っている場合には、それらを接続して PANELS_NUMBER を適切に設定してプログラムを実行すると、複数枚の LED パネルの簡単な使用例になります。

また、必要に応じて、mxconfig が定義されている 116 行付近より後に、クロックの位相変更用のコードを加えてください。これ以降のサンプルプログラム等でも同様です。

5.3.4　3_DoubleBuffer

[3_DoubleBuffer] は、ダブルバッファを使用し、ちらつきなく画面更新をする方法を示したデモです。

29 行目付近で定義された numSquares 個の正方形の画像が LED パネル内を動き回ります。この様に動き回る画像を書き換える処理を行うためには、画像の更新を行う前に、一旦今の画像をバッファから消去する処理を行います。すると、画像が書かれた状態と、画像が消去された状態が入り乱れて LED パネルに表示されるため、画面がちらついて正常に表示されない状況になります。

ダブルバッファ機能は、画像を記録するバッファを、表示用と書き換え用の 2 つに分け、現時点で表示に使用されていない書き換え用のバッファに対して画面の消去や図形の書き込みを行います。LED パネルには表示用のバッファの内容が表示されているため、書き換え用のバッファに対する操作は LED パネルに反映されず、LED パネルの表示がちらつくことはありません。書き換え用のバッファに対する画像の更新処理が終了したら、そのバッファの内容を今度は LED パネルに表示されるバッファとして切り替えます。この様にすると、バッファの書き換えのために画像を消したり書いたりしている過程が LED パネルに表示されずに済むために、スムーズな表示が得られます。

ダブルバッファは動きのある画像をちらつきなく表示するために重要な機能ですが、通常よりも 2 倍の RAM を使用するので、使用の際には利用できる LED パネルの数やピクセル数が半減する問題がある点に注意が必要です。

ダブルバッファの機能は 3_DoubleBuffer の 41 行目付近で有効化され、65 目付近で表示用と書き換え用のバッファ切り替えが行われています。この 2 つの行をコメントアウトすると、ダブルバッファの機能を使わない基本的なプログラムになります。この様な変更によりダブルバッファを使わない状態での画像更新処理に変更すると、LED パネルが激しくちらつくことに驚かれるでしょう。

このサンプルプログラムでは、LED パネルのピクセル数の指定が行われていませんが、プログラムを動かしてみると 64x32 の LED パネルで正しく動きます。これは、ライブラリのデフォルトの LED パネルサイズが 64x32 になっていることを示しています。ライブラリのソースコードを確認してもよいですが、2_PatternPlasma のコメントの 95 行から 110 行あたりに、mxconfig のデフォルト値が記載されているので確認してみてください。

5.3.5　ChainedPanels

LED パネルを組み合わせて大きな表示装置を構成するためには、ChainedPanels を参照してください。

画像による説明も含めた、以下の説明も提供されています。

- `https://github.com/mrfaptastic/ESP32-HUB75-MatrixPanel-DMA/blob/master/examples/ChainedPanels/README.md`
- `https://github.com/mrfaptastic/ESP32-HUB75-MatrixPanel-DMA/blob/master/doc/VirtualMatrixPanel.pdf`

ぜひたくさんの LED パネルを組み合わせて、インパクトのある表示装置を楽しんでください。

5.3.6　温度等の表示

BME280 を使用して気温、湿度、気圧を取得し、それを LED パネルに表示する例を示します。

```
#include <ESP32-HUB75-MatrixPanel-I2S-DMA.h>
#include <BME280I2C.h>
#include <Wire.h>

#define RES_X 64
#define RES_Y 32
#define CHAIN 1

MatrixPanel_I2S_DMA *dma_display = nullptr;
BME280I2C bme;

void setup() {
  HUB75_I2S_CFG mxconfig(RES_X, RES_Y, CHAIN);

  // Display Setup
  dma_display = new MatrixPanel_I2S_DMA(mxconfig);
  dma_display->setBrightness8(128);  // range is 0-255, 0 - 0%, 255 - 100%
  dma_display->begin();

  Wire.begin();

  while (!bme.begin()) {
    delay(1000);
  }
}

void printValue(double v, char *unit, uint16_t color) {
  dma_display->setTextColor(color);

  if (v < 100)
    dma_display->print("  ");
  else if (v < 1000)
    dma_display->print(" ");

  dma_display->print(v);
  dma_display->println(unit);
}

void loop() {
  float temp, hum, pres;
  BME280::TempUnit tu(BME280::TempUnit_Celsius);
  BME280::PresUnit pu(BME280::PresUnit_hPa);

  bme.read(pres, temp, hum, tu, pu);

  dma_display->fillScreen(0);
  dma_display->setCursor(0, 0);

  printValue(temp, "'C", dma_display->color565(0, 255, 0)) ;
  printValue(hum, "%", dma_display->color565(255, 0, 0)) ;
  printValue(pres, "hPa", dma_display->color565(0, 0, 255)) ;

  delay(2000);
}
```

図 5.4　温度・湿度・気圧の表示

残念ながら、ESP-WROOM-32 や LED パネルはそれなりの発熱体なので、放熱などを考慮しない場合、計測された温度や湿度はあまり適切な値とは言えないようです。

第 6 章

資料

6.1　ESP32-LEDPRO の回路図

ESP32-LEDPRO の回路図を図 6.1、部品表を表 6.1 に示します。

内部のフラッシュメモリに接続されている ESP-WROOM-32 の 17-22 ピンは、取扱に注意を要するため未接続となっています。

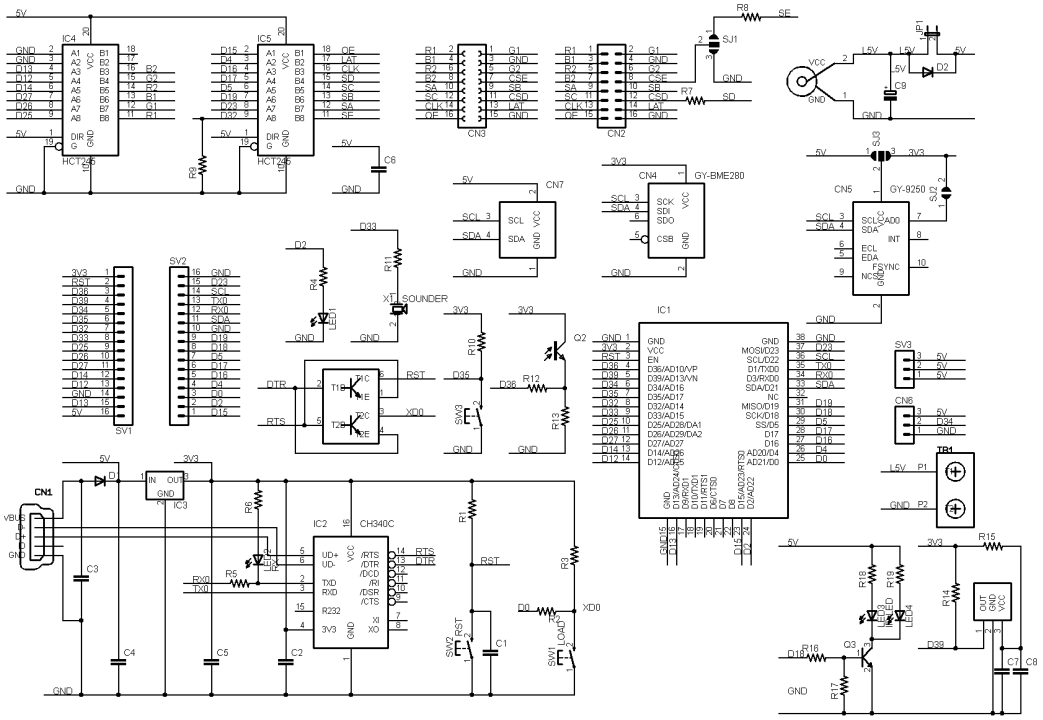


図 6.1　ESP32-LEDPRO の回路図

表 6.1　部品表

部品	シンボル	規格等	1
プリント基板	ESP32-LEDPRO	Rev.1	1
IC	IC1	ESP-WROOM-32(E, 16M)	1
	IC2	CH340C	1
	IC3	BL8071CLATR33	1
	IC4, IC5	74HCT245	2
	IC6	TL1838	1
トランジスタ	Q1	UMH3N	1
	Q3	SS8050	1
フォトトランジスタ	Q2	NJL7502L	1
ショットキーダイオード	D1, D2	SS14	2
圧電スピーカー	X1		1
発光ダイオード	LED1	青	1
	LED2, LED4	赤	2
赤外線発光ダイオード	LED3	OSI5FU5111C-40	1
抵抗	R1, R3, R9, R10, R13, R17	10K Ω	6
	R2, R6, R7, R8, R11	1K Ω	5
	R4, R5, R12	470 Ω	3
	R14	22K Ω	1
	R15	100 Ω	1
	R16	330 Ω	1
	R18	10 Ω	1
	R19	2.2K Ω	1
セラミックコンデンサ	C1, C2, C3, C6, C7	0.1 μ F	5
	C4	10 μ F	1
	C5, C8	47 μ F	2
タクトスイッチ	SW1, SW2, SW3	2 端子	3
USB	CN1	microB	1
ピンヘッダー	CN2	2x8	1
ターミナルブロック	TB1		1
DC ジャック	PWR1	内径 2.1mm	1

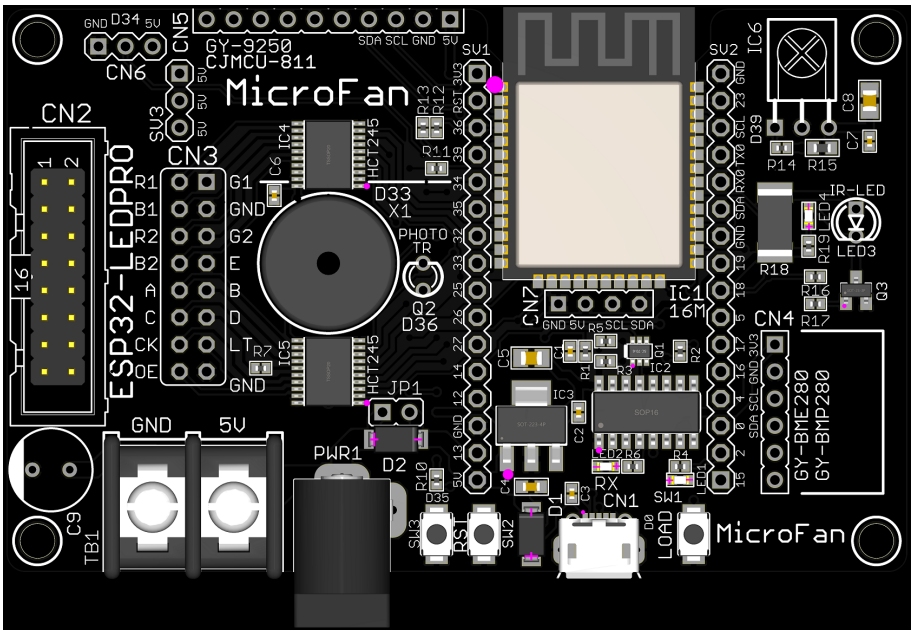


図 6.2　ESP32-LEDPRO の部品配置

6.2　基板上の入出力

ESP32-LEDPRO の基板上の入出力を表 6.2 に示します。

表 6.2　基板上の入出力

シンボル	信号線	備考
SW1	D0	ブートモード移行用
SW2	EN	リセット用
SW3	D35	プルアップあり
LED1	D2	
LED2	受信データモニタ	
LED3	D18	赤外線 LED
LED4	D18	赤外線 LED モニタ
X1	D33	圧電スピーカー
Q2	D36	明るさセンサー（フォトトランジスタ）
IC6	D39	赤外線受信機

SW1 は、リセット時のブートモード（Arduino IDE などからのスケッチ書き込み）の切り替え用ですが、スケッチが走り始めたあとは、一般的な入力用のスイッチとして利用することができます。

LED3 は赤外線 LED の接続用です。ただし現バージョンの基板では、電流制限抵抗 R18 の部品指定に誤りがあり小さな値となっているため、使用を推奨していません。

赤外線 LED を接続する場合には、赤外線 LED に直列に 33 Ω程度の抵抗を追加して基板に接続する必要があります。

LED4 は、発光状態が目に見えない赤外線 LED3 が点灯しているかどうかを確認するためのインジケータとして設置されています。LED3 に赤外線 LED を装着しない場合には、LED4 は D18 で制御する単純な LED 出力となります。

6.3　HUB75 コネクタ用端子

HUB75 のコネクタ接続用に使用される CN2 および CN3 の信号は位置は、1/32 スキャン用のコネクタの一般的な配置となっています。

なお、コネクタが接続されていない CN3 は、基板の背面にピンソケットを接続して、LED パネルに直接続する使い方を想定して用意されています。

HUB75 端子 CN2 のそばに配置されている C9 のパターンに、電解コンデンサを接続することができます。一般的には必要ありませんが、LED パネルへの電源供給に関して電解コンデンサを付与したい場合には、ここに接続することができます。

6.3.1　SV3: 5V 電源端子

CN3 には、5V に変換された信号端子出力が多数接続されています。ESP32-LEDPRO を LED パネルの制御に使用せず、5V の信号線で何かを制御しようとした場合、CN3 には、GND 端子はいくつか配置されていますが、5V の電源端子が全く配置されていません。SV3 は、5V 電源が必要な時にそれが取り出せるように CN3 のそばに配置された電源端子となっています。

6.4　電源端子

6.4.1　DC ジャック

LED パネルに供給する 5V 電源を接続することができます。内径 2.1mm の一般的な AC アダプタを接続できます。

5V の AC アダプタは 4A 程度が最大のようなので、ESP32-LEDPRO で制御するパネルが複数枚になったり、表示パターンが電力を消費する全点灯に近いような場合には、AC アダプタを使用せず下記のターミナルブロックに大容量のスイッチングレギュレータの出力を接続します。

6.4.2　ターミナルブロック

ESP32-LEDPRO の DC ソケットに 5V 電源を接続している場合に、その電力を LED パネルに供給するために、LED パネルの電源ケーブルをこのターミナルブロックに接続してください。

また、LED パネルの消費電力が大きく、AC アダプタでは不十分と考えられる場合には、大容

量のスイッチングレギュレータの出力もターミナルブロックに接続してください。

電源としてスイッチングレギュレータを使用する場合には、必ずしもその出力を ESP32-LEDPRO のターミナルブロックに接続する必要はありません。しかしながら、LED パネルが必要とする電力を、ESP32-LEDPRO を経由せずに直接スイッチングレギュレータから LED パネルに電源ケーブルを接続して利用する場合には、ESP-WROOM-32 とその周辺回路に電力を供給するために、ESP32-LEDPRO に USB か AC アダプタを接続してください。

6.5　ジャンパー

6.5.1　JP1: LED パネルへの電力供給

LED パネルは数アンペアの大電流を使用する可能性があるため、USB からのみの給電で LED パネルを点灯させると、USB の規格を超えて PC 等に電流が要求される危険があります。この様な危険を回避するため ESP32-LEDPRO では、LED パネルへの 5V の供給は USB からは行われず、PWR1 か TB1 に接続された 5V の外部電源からのみ行われるようになっています。

逆に、PWR1 か TB1 に 5V の外部電源が接続されているが USB が接続されていない場合には、D2 を通して外部電源の電力が ESP-WROOM-32 とその周辺に供給されるようになっています。これにより、プログラムの書き込みが終わった後は、USB コネクタを外して ESP32-LEDPRO を運用することができます。

このように、通常は LED パネルの電力使用量の大きさに対する安全策がとられていますが、LED パネルの電力消費量が低い表示パターンでの表示実験などを行っている場合には、LED パネル用に外部電源をいちいち接続することが面倒に感じられることがあります。

この様な場合に JP1 をショートさせると、USB 側から LED パネルに 5V の電源が供給されるようになり、外部電源を接続する必要がなくなります。ただし、この様な設定をした場合には、使用電力量が増大した場合の USB 電源保護の安全機構がありません（PC 内部の安全機構のみに頼ることになります）ので、十分にご理解の上 JP1 のショートを行ってください。

6.5.2　SJ1: HUB75 の端子設定

基板裏面の HUB75 端子の CN3 のそばに SJ1 があります。

HUB75 コネクタは、1/16 スキャン用と 1/32 用で端子の一部の用途が異なっています。具体的には、1/16 では 7 番ピンが GND になっているのに対して、1/32 では E 端子として使用されています。

SJ1 は、7 番ピンを GND か E の制御用の信号かに切り替えるために使用します。

- 1/16 スキャンの LED パネルしか使用しない場合
  SJ1 の GND 側と中央の間にはんだをもってショートさせてください。
- 1/32 スキャンの LED パネルを使用するか、使用の可能性がある場合
  SJ1 の E 側と中央の間にはんだをもってショートさせてください。

SJ1 を 1/32 スキャンの LED パネル用に設定した場合には、E 用の信号線が強制的に GND にショートさせられる状態になりますが、R8 を組み込むことでショート状態にならないように配慮しています。

結局のところ、1/32 スキャンの LED パネルを使用する可能性がある場合には、SJ1 を 1/32 スキャン用に設定した状態で 1/16 スキャンの LED パネルを使用しても（理想的な状態ではありませんが）問題なく使用できるようになっています。

6.5.3　SJ2: 加速度・ジャイロセンサーの I2C アドレス

6.7.2 節でも示していますが、CN5 に接続する加速度・ジャイロセンサーの I2C アドレスの選択用のジャンパーパッドです。

6.6　SV1,SV2: 信号引き出し用端子

ESP32-LEDPRO には、ESP-WROOM-32 の信号線を引き出して利用するための端子 SV1, SV2 が用意されています。SV1,SV2 のピン配置を表 6.3 に示します。

表 6.3　SV1,SV2 ピン配置

備考	SV1 信号線	ピン番号	SV2 信号線	備考
	GND	15	GND	
	3.3V	14	5V	
EN	RST	13	D23	MOSI
AD10/VP	D36	12	D22	SCL
AD13/VN	D39	11	D1	TX0
AD16	D34	10	D3	RX0
AD17	D35	9	D21	SDA
AD14	D32	8	D19	MISO
AD15	D33	7	D18	SCK
AD28/DA1	D25	6	D5	SS
AD29/DA2	D26	5	D17	
AD27	D27	4	D16	
AD26	D14	3	D4	AD20
AD25	D12	2	D0	AD21
AD24	D13	1	D15	AD23

LED1 に接続されている D2 は開発ボード外に引き出されていません。

6.7　拡張端子

6.7.1　CN4: 環境センサー

CN4 には、環境センサー GY-BME280, GY-BMP280 を接続することができます。CN4 のピン配置を表 6.4 に示します。

電源は 3.3V を供給しており、センサーと I2C 信号線は 3.3V で駆動されます。

表 6.4　CN4 ピン配置

ピン番号	信号線	備考
1	3.3V	
2	GND	
3	SCL	A5
4	SDA	A4
5	-	
6	-	

温度センサを使用する場合には、その計測値の信頼性に関する注意が必要です。

ESP32-LEDPRO では、LED パネルに常にデータを送り続けるため、スリープなどを利用して電力消費を抑えることができず、ESP-WROOM-32 から継続的な発熱があります。また、LED パネルも表示内容によりますが、それなりに発熱します。

このため ESP32-LEDPRO の周囲は、周辺環境よりも温度が上昇するため、そこに接続されている温度センサの計測値は一般的に本来の値よりも高くなります。また、湿度も温度との関係で算出されるため、BME280 で計測される湿度も一般的には温度の違いによる影響を受けます。

温度や湿度の計測値を積極的に使用したい場合には、センサー周囲の空気の循環等に配慮して、ESP32-LEDPRO そのものや LED パネルの発熱の影響を軽減させることが必要になります。

6.7.2　CN5: 加速度・ジャイロセンサー, CO2 センサー

CN5 には、加速度・ジャイロセンサー GY-521, GY-9250 か、CO2 センサー CJMCU-811 を取り付けることができます。CN5 に GY-521 もしくは CJMCU-811 を取り付ける場合には、基板の右側の電源端子の位置を合わせて接続してください。

CN5 のピン配置を表 6.5 に示します。

CN5 に GY-521, GY-9250 を接続した場合には、SJ2 をはんだでショートさせることにより、I2C のアドレスを変更することができます。

GY-521 と GY9250 はモジュール上に電源レギュレータが載っておりモジュール内で 5V を 3.3V に変換して使用します。このため、これらのモジュールを使用する場合には、SJ3 の 9250 側 5V と中央のランドをはんだでショートさせてください。一方 CJMCU-811 は 3.3V の電源を供給しなければならないので、SJ3 の 811 側 3.3V と中央のランドをはんだでショートさせてくだ

表 6.5　CN5 ピン配置

ピン番号	信号線	備考
1	5V or 3.3V	SJ3 で設定
2	GND	
3	SCL	A5
4	SDA	A4
5	-	
6	-	
7	AD0	SJ2 で設定
8	-	
9	-	GY-9250 のみ
10	-	GY-9250 のみ

さい。

6.7.3　CN6: 人感センサー

CN6 には、人感センサー HC-SR501 を接続することができます。CN6 のピン配置を表 6.6 に示します。

表 6.6　CN6 ピン配置

ピン番号	信号線	備考
1	GND	
2	D34	
3	5V	

HC-SR501 は電源電圧は 5V となっていますが、内部では 3.3V のレギュレータで降圧され、回路も出力も 3.3V で駆動されています。このため、HC-SR501 の出力は 3.3V となっているため直接 ESP-WROOM-32 に入力することができます。

6.7.4　CN7: OLED ディスプレイ搭載用コネクタ

基板上に OLED ディスプレイを搭載するための CN7 端子を備えています。CN7 のピン配置を表 6.7 に、推奨する OLED ディスプレイを図 6.3 に示します。また、推奨する OLED ディスプレイのネットショップ URL を以下に示します。

- `https://store.shopping.yahoo.co.jp/microfan/oled096-128x64-i2c-blue.html`
- `https://www.amazon.co.jp/dp/B06Y4TKL1F`

表 6.7　CN7(OLED ディスプレイ) ピン配置

ピン番号	信号線	備考
1	GND	
2	5V	
3	D22	SCL
4	D21	SDA

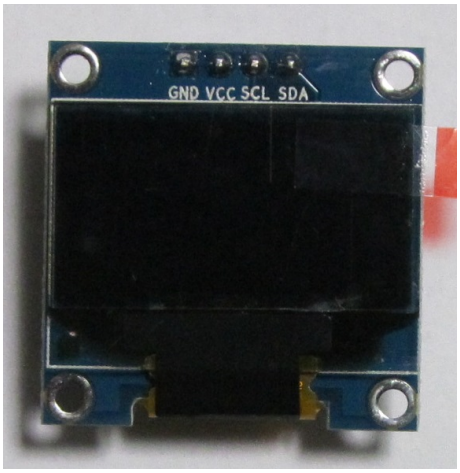


図 6.3　OLED ディスプレイ

OLED ディスプレイに要求される機能を以下に示します。

- モジュールを直接端子に刺すためには、信号線の並びが表**??**の順になっていること。
- SCL, SDA の信号線が 3.3V 対応であること。
- ESP32-LEDPRO には I2C 用のプルアップ抵抗が組み込まれていないため、SCL, SDA の信号線にプルアップ抵抗が付与されていること。
- ESP32-LEDPRO からの電源として 5V を供給しているため、3.3V の電圧レギュレータが内蔵されていること。
- 使用するライブラリにもよりますが、コントローラに SSD1306 か SH1106 を使用していること。

OLED ディスプレイの SDA, SCL 信号線には、4.7K-10K Ωのプルアップ抵抗が組み込まれています。このため、CN7 に OLED ディスプレイを接続している場合には、ブレッドボード上で I2C デバイスを使用する場合に、SDA, SCL に別途プルアップ抵抗を接続する必要はないのでご注意ください。(付けた場合には、OLED ディスプレイのプルアップ抵抗との合成抵抗値となります。)

第 7 章

購入および問い合わせ先

7.1　ご協力のお願い

製品をより良くし、多くの方々にお楽しみいただけるよう、製品の向上に努めて参ります。問題点やお気づきの点、あるいは製品の企画に対するご希望などございましたら、microfan_shop@yahoo.co.jp までご連絡いただけますようよろしくお願いいたします。末永くご愛顧いただけますよう、お願いいたします。

7.2　販売：ネットショップ

製品の販売はネットショップで行っています。対面販売は行っておりません。

- マイクロファン Yahoo!ショップ
  WEB アドレス：https://store.shopping.yahoo.co.jp/microfan/
- アマゾン
  WEB アドレス：https://www.amazon.co.jp/s?merchant=A28NHPRKJDC95B

7.3　製品情報

マイクロファン ラボ
WEB アドレス：http://www.microfan.jp/
マイクロファンの製品情報や活用情報を紹介しています。

7.4　問い合わせ先

株式会社ピープルメディア　マイクロファン事業部
E-Mail: microfan_shop@yahoo.co.jp
TEL: 092-938-0450
お問い合わせは基本的にメイルでお願いいたします。

7.5　所在地

株式会社ピープルメディア　マイクロファン事業部
〒811-2316 福岡県糟屋郡粕屋町長者原西 2-2-22-503