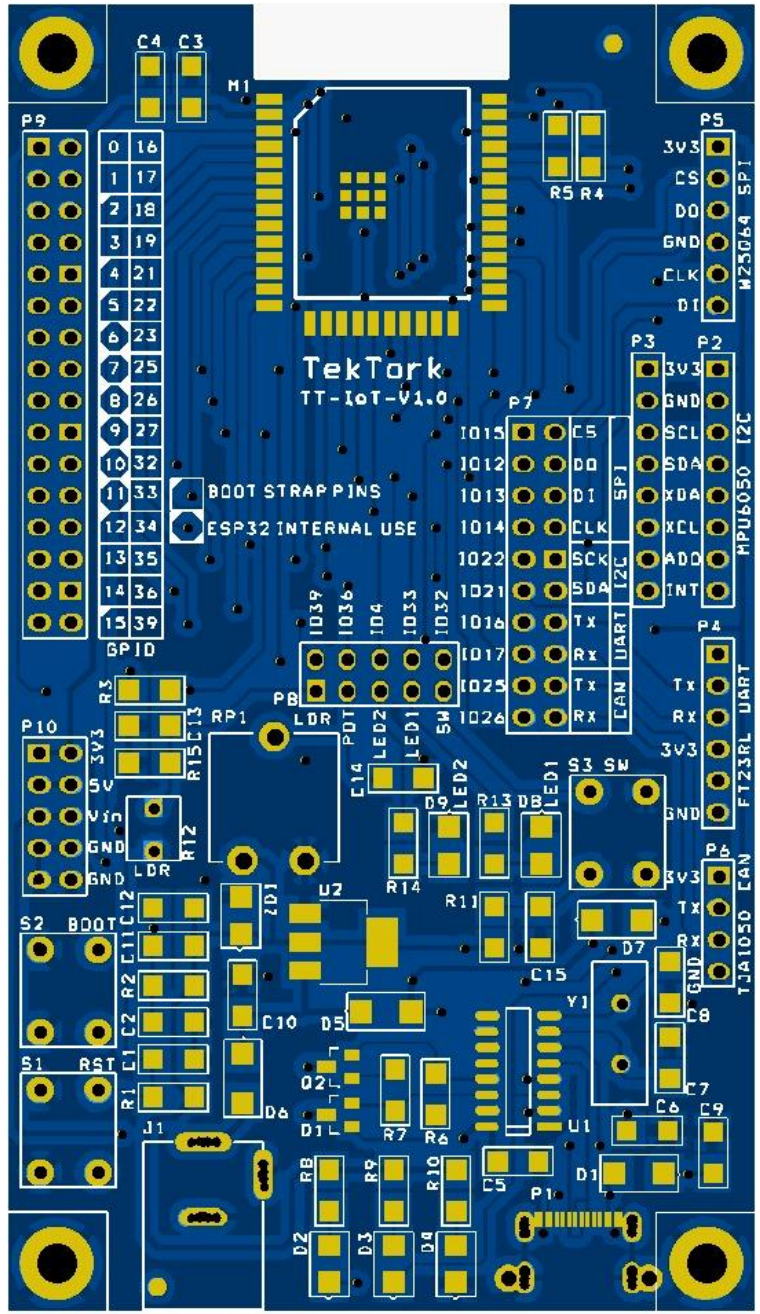
Introduction
to
TekTork IoT Kit V1.0

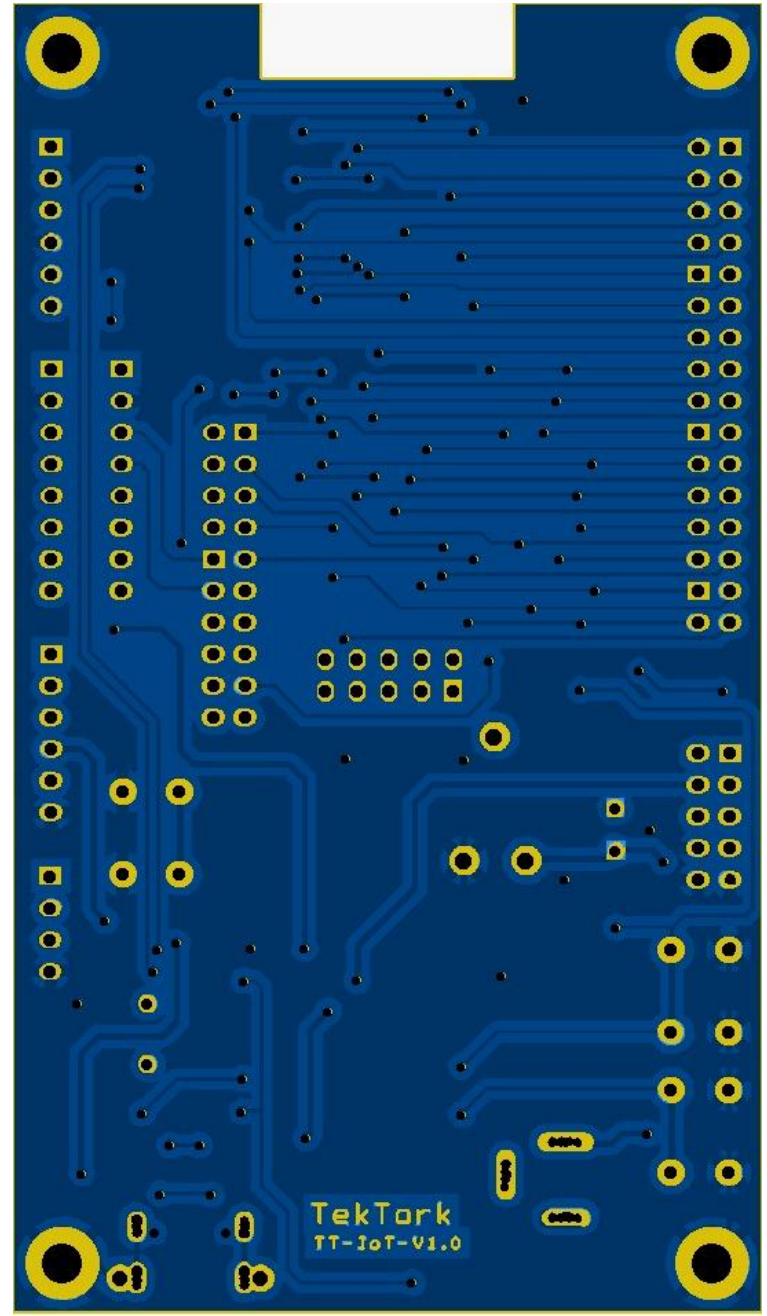
TekTork Private Limited, Coimbatore
info@tektork.com
https://tektork.com

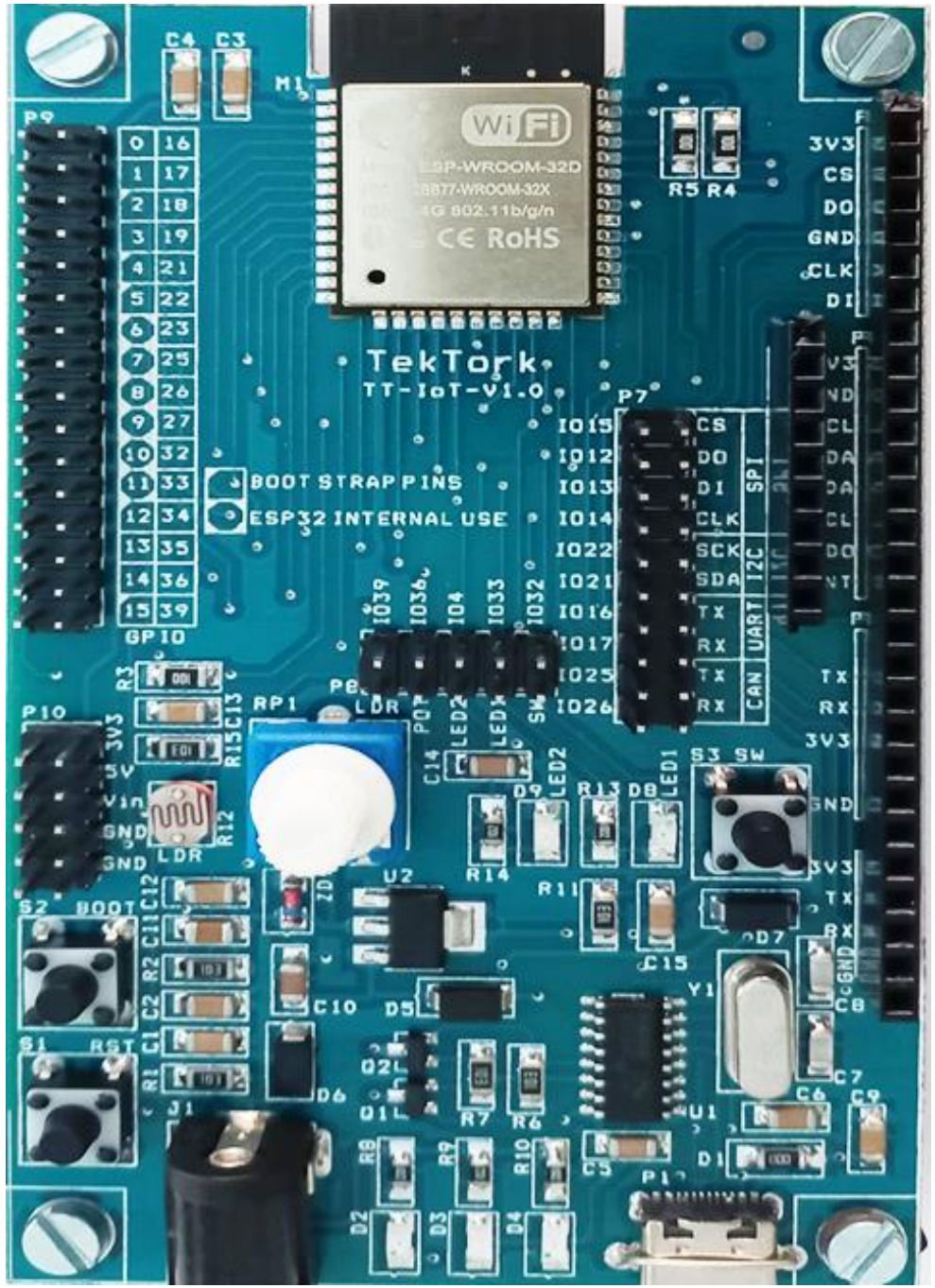
https://github.com/tektork/iotkit

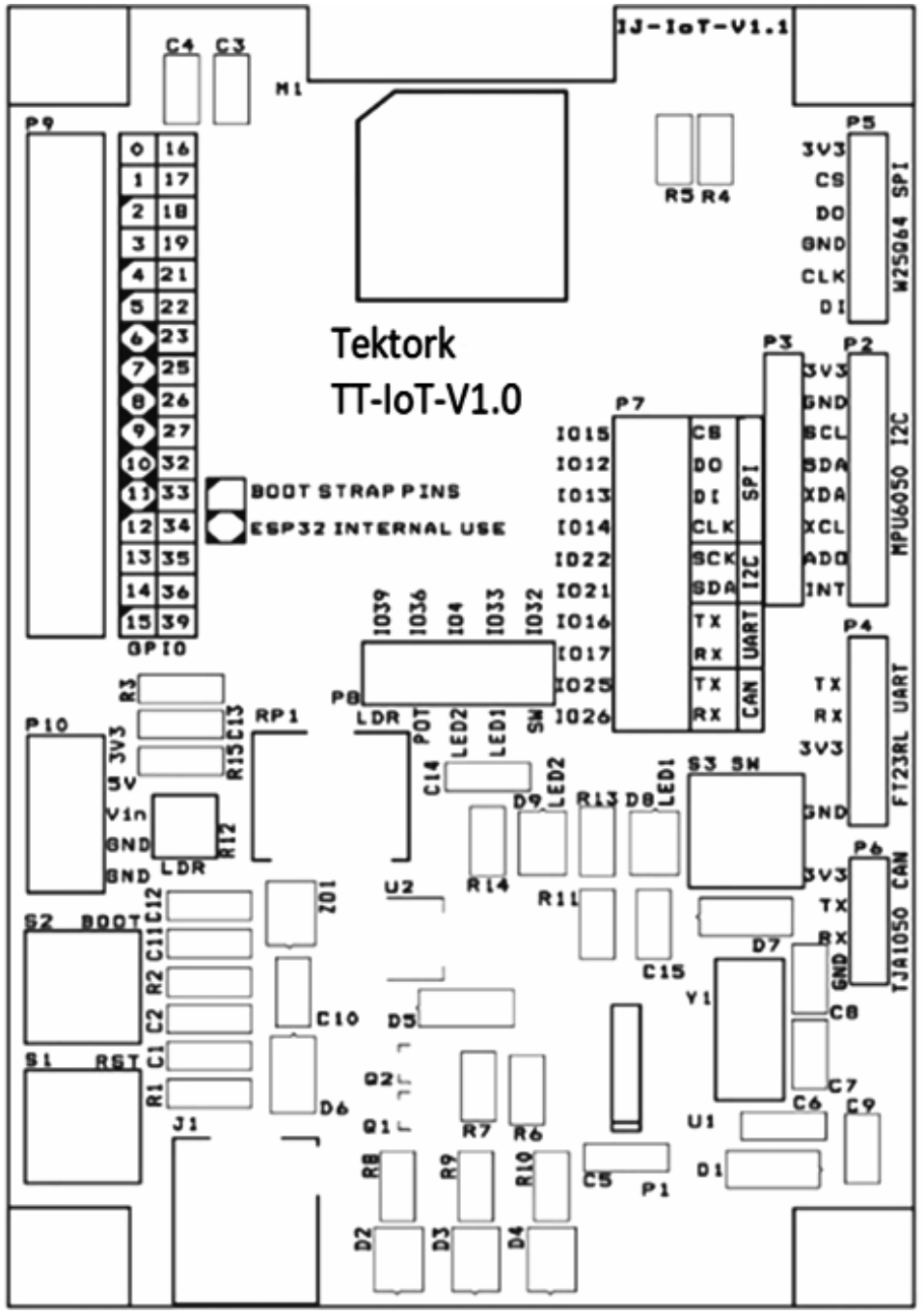
PCB Top Side



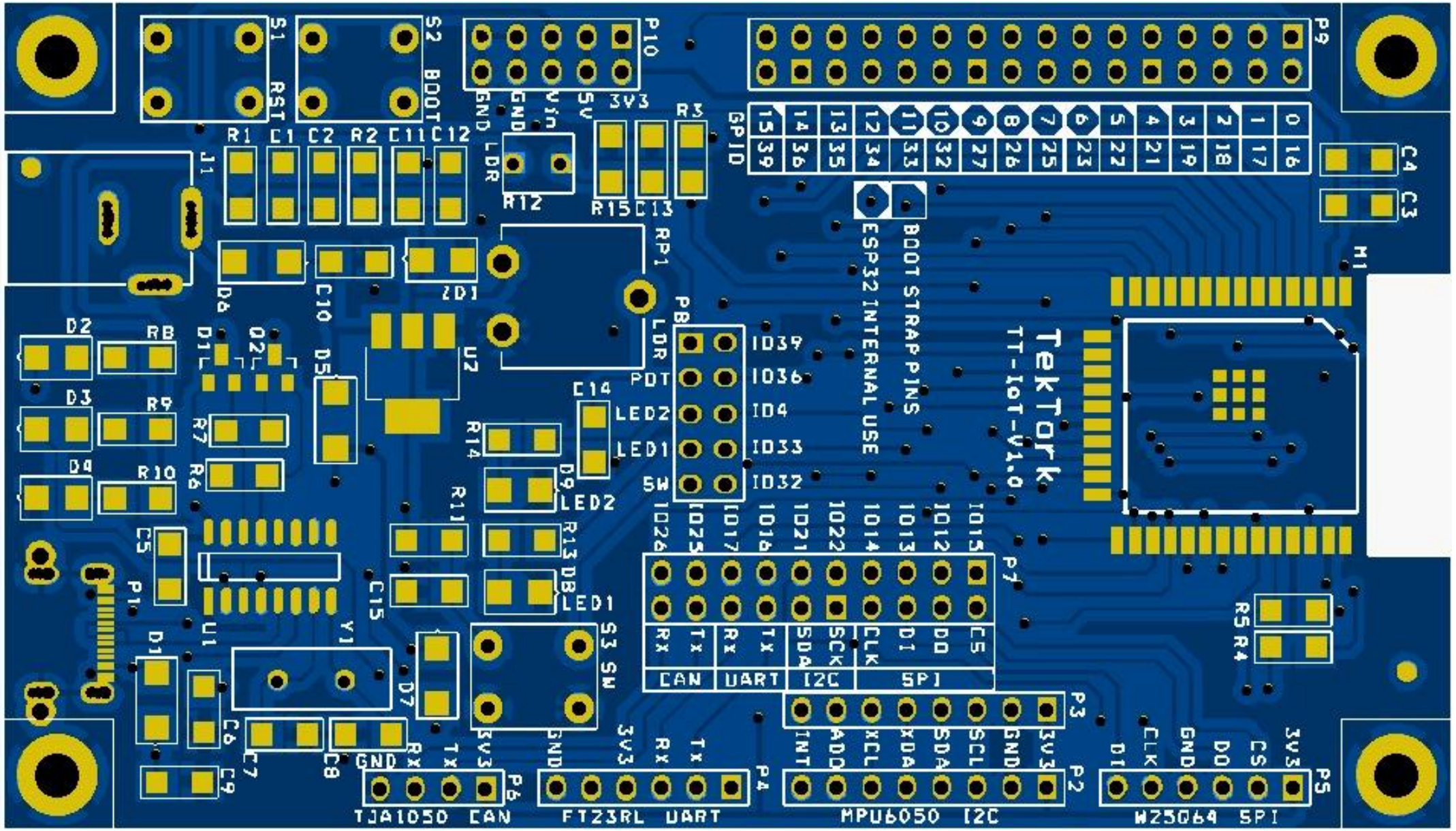
PCB Bottom Side

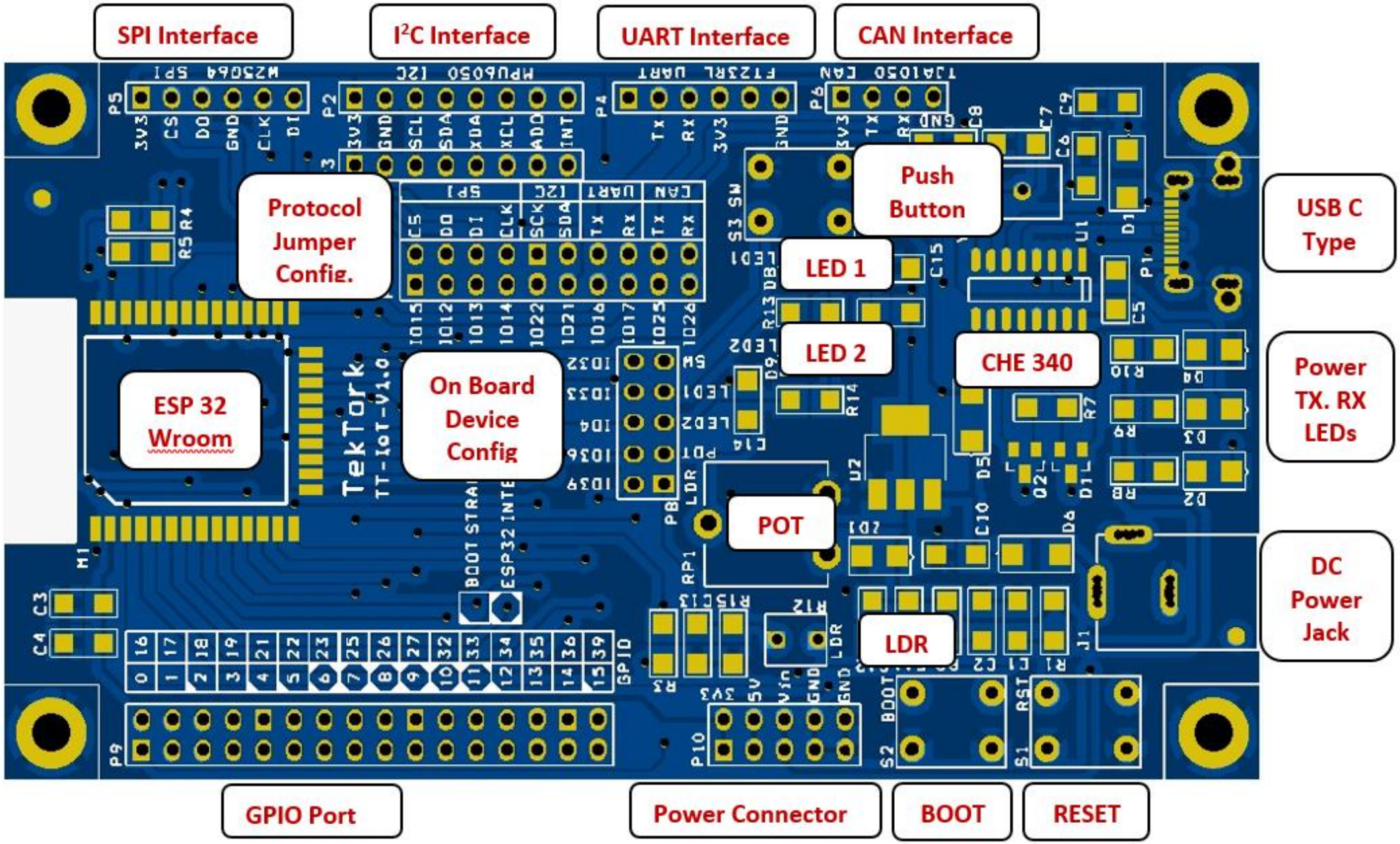



TekTork
TT-IoT-V1.0
BOOT STRAP PINS
ESP32 INTERNAL USE
GPIO
P9
P10
P7
P8
S1 RST
S2 BOOT
S3 SW
J1
P1
U1
U2
Y1
LDR
LED1
LED2
SPI
I2C
UART
CAN

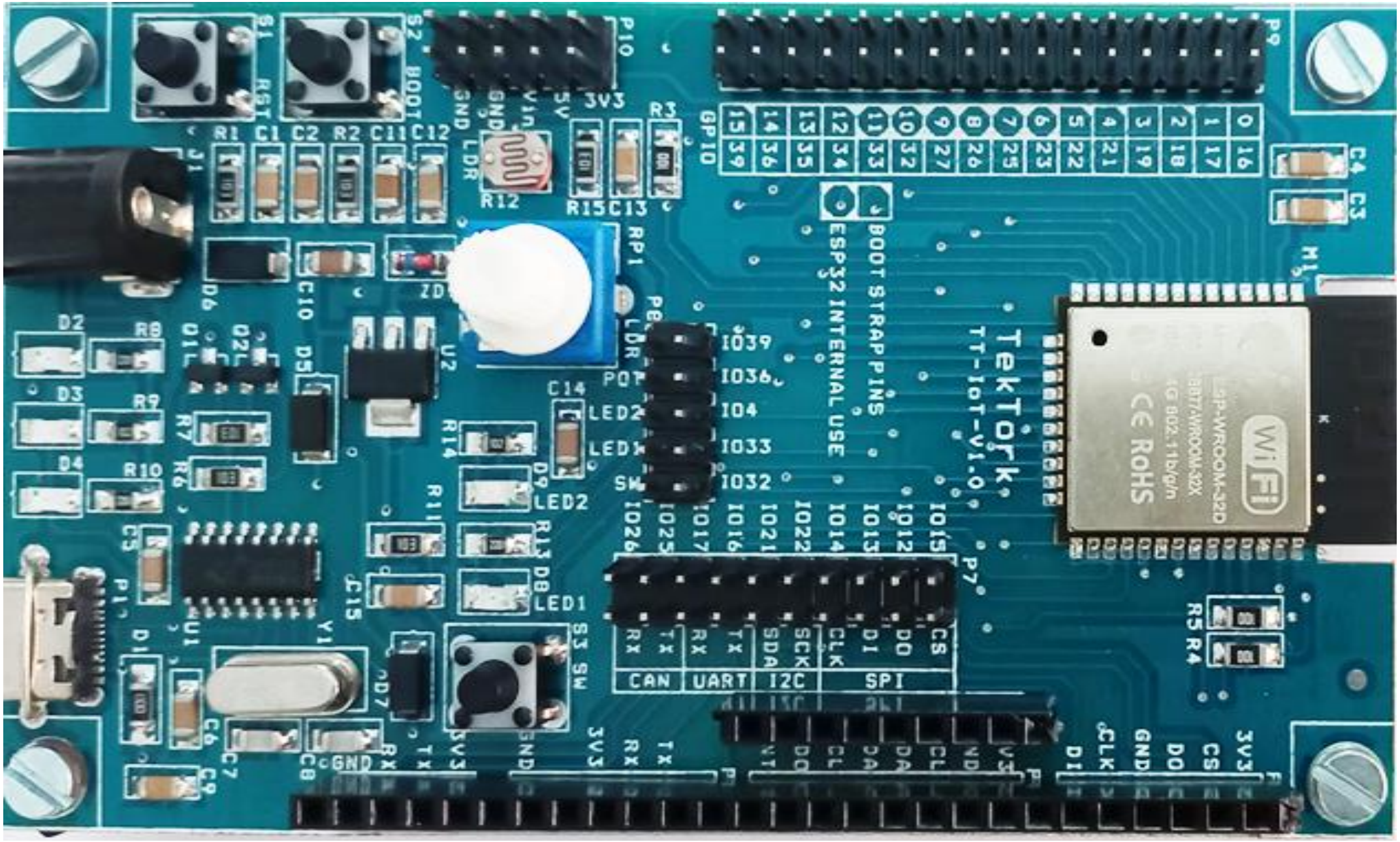

TT-IoT-V1.1
Tektork
TT-IoT-V1.0
BOOT STRAP PINS
ESP32 INTERNAL USE
GPIO
P9
P10
P7
P8
S1 RST
S2 BOOT
S3 SW
J1
P1
U1
U2
Y1
LDR
LED1
LED2
SPI
I2C
UART
CAN
W25Q64 SPI
MPU6050 I2C
FT23RL UART
TJA1050 CAN

Plain PCB Top Side- Horizontal




SPI Interface
I²C Interface
UART Interface
CAN Interface
Protocol Jumper Config.
Push Button
USB C Type
LED 1
LED 2
CHE 340
Power TX. RX LEDs
ESP 32 Wroom
On Board Device Config
POT
DC Power Jack
LDR
GPIO Port
Power Connector
BOOT
RESET

Components Fixed PCB



Core Features

- The board is built around the **ESP WROOM32 Wi-Fi (802.11 b/g/n) + BLE Module (BLE 4.2)**.
- Dual-core 32-bit Xtensa LX6 processor, up to 240 MHz, with 520 KB SRAM and 4MB Flash.
- This makes it highly versatile for a range of wireless applications, as the core module can be swapped out depending on the specific project needs.
- It includes on-board **Reset and Boot Push Buttons** for easy programming and system control.
- It's (**ESP WROOM32**)the brain of the board, handling all the processing, connectivity, and I/O operations.

Software & Connectivity

- The board supports **LwIP** (Lightweight IP) and **FreeRTOS**, which are both popular choices for embedded systems.
- LwIP provides a lightweight networking stack, while FreeRTOS is a real-time operating system that's essential for managing multiple tasks efficiently.
- Connectivity is robust, featuring a **CH340 USB to UART Converter**, which allows for a direct connection to a computer for programming and serial communication.
- It also has an **on-board Type C Connector** for power, programming, and use as a serial port. For external power, there is an **on-board DC power jack**.

Comparison Table

Feature	TekTork IoT Kit	Typical ESP32 Dev Kit
Module	ESP32-WROOM-32	ESP32-WROOM-32/32D/32U
Power	USB-C, DC Jack, 5V/3.3V Pins	micro-USB, VIN, 5V/3.3V Pins
Boot/Reset	On-board Buttons	Yes
User IO (LEDs/Buttons)	2 LEDs, 1 Button, Pot, LDR	Often 1 LED
Expansion Connectors	UART, I2C, SPI, CAN via jumpers	GPIO Headers
Add-on Modules	MPU6050, TJA1050, W25Q64, FT232	Via breadboard
Board Size	60 × 105 mm	Smaller, less robust
Target Use	Training/Prototyping	General Embedded
USB Port	C Type	Micro USB B Type

GPIO Pins

GPIO Pins

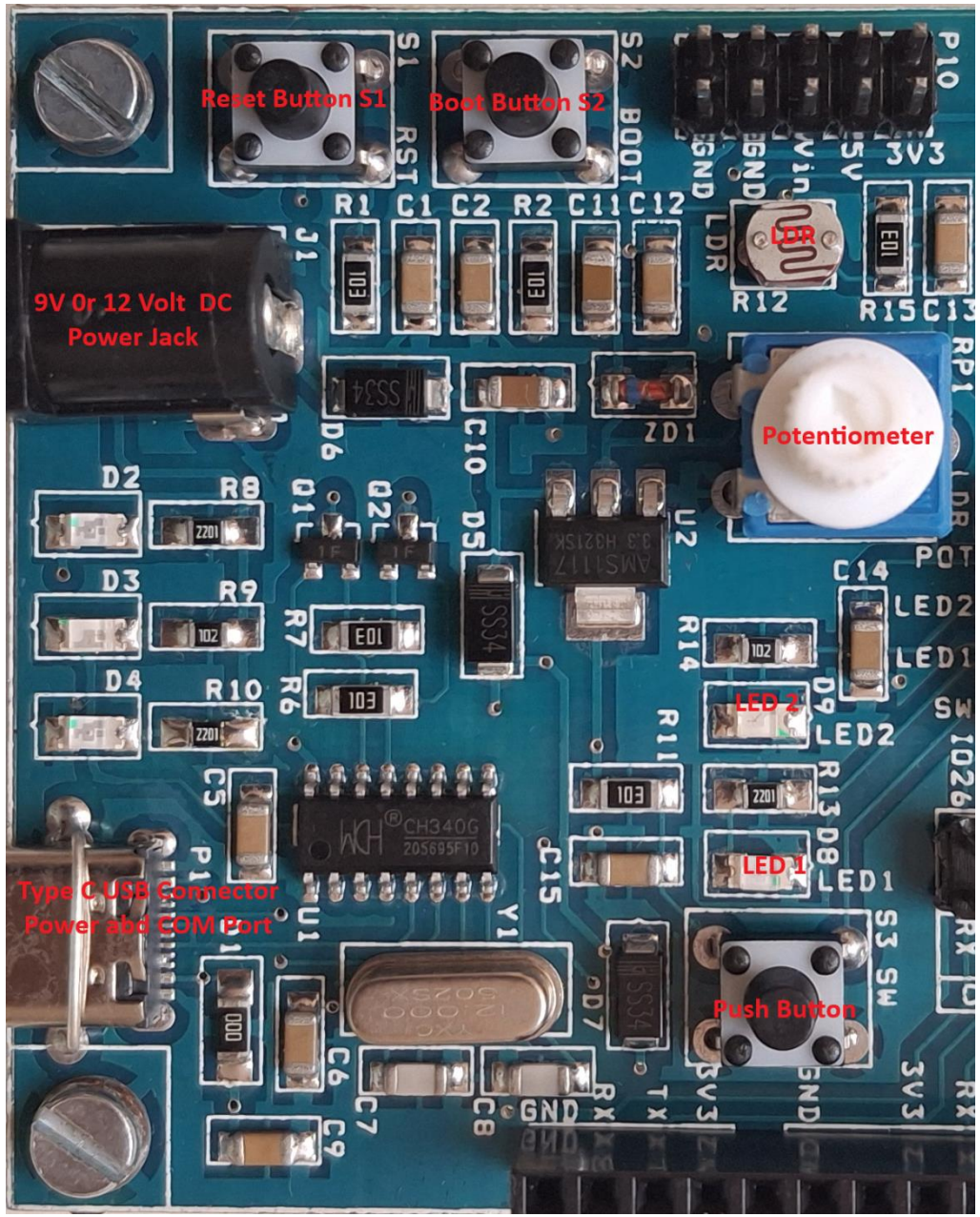
- The large rows of headers on the left and right sides are the **General Purpose Input/Output (GPIO) pins**.
- These pins are crucial for connecting the board to other sensors, actuators, and electronic components.
- The pins are labeled P1 through P10, with some marked for specific functions like SPI, UART, and I2C.

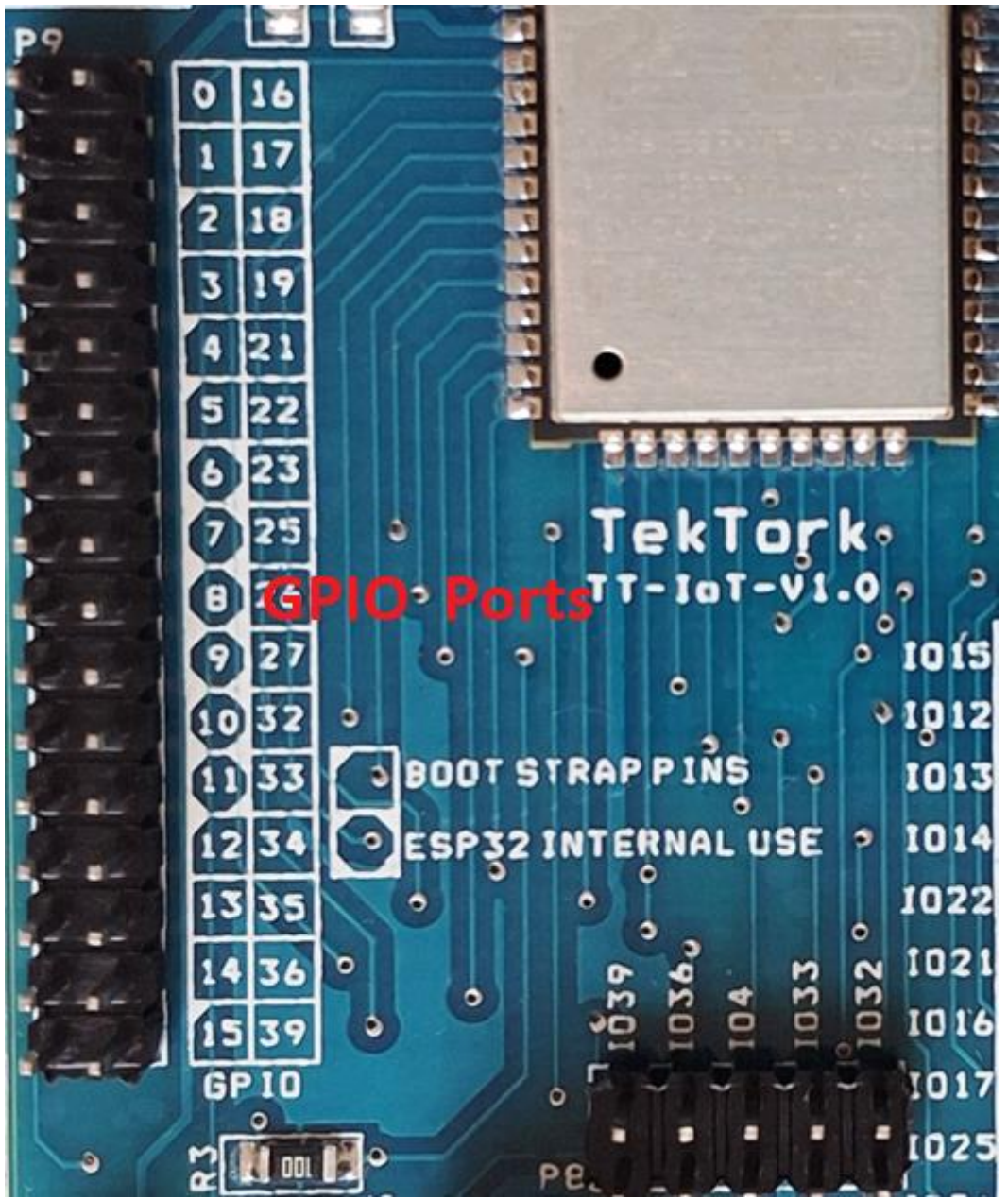
GPIO Pin Details

GPIO Pin	Spl Notes	FXN 1	FXN 2
0	General	ADC 11	TOUCH 1
1	General	TX 0	
2	Bootstrap	ADC 12	TOUCH 2
3	General	RX 0	
4	Bootstrap	ADC 10	TOUCH 0
5	Bootstrap	VSPI SS	
6	Internal use	FLASH CK	
7	Internal use	FLASH D0	
8	Internal use	FLASH D1	
9	Internal use	FLASH D2	RX 1
10	Internal use	FLASH D3	TX 1
11	Internal use	FLASH CMD	
12	Bootstrap	ADC 15	TOUCH 5
13	General	ADC 14	TOUH 14
14	General	ADC 16	TOUCH 6
15	Bootstrap	ADC 13	TOUCH 13

GPIO Pin	Spl Notes	GPIO Pin	FXN2
16	General	RX 2	
17	General	TX 2	
18	General	VSPI SCK	
19	General	VSPI MISO	
21	General	I2C SDA	
22	General	I2C SCL	
23	General	VSPI MOSI	
25	General	ADC 18	DAC 1
26	General	ADC 19	DAC 2
27	General	ADC 17	TOUCH 7
32	General	ADC 04	TOUCH 9
33	General	ADC 05	TOUCH 8
34	General	ADC 06	
35	General	ADC 07	
36	General	ADC 00	
39	General	ADC 03	

On-board Components





On-board Components

- The board always provides two controllable LEDs and two separate analog inputs (POT and LDR), robustly supporting hands-on learning and complex experiments.

- Overall, the jumper system empowers users to quickly switch pin assignments or experiment with different GPIOs, making the platform both flexible and accessible for both structured labs and open-ended projects.

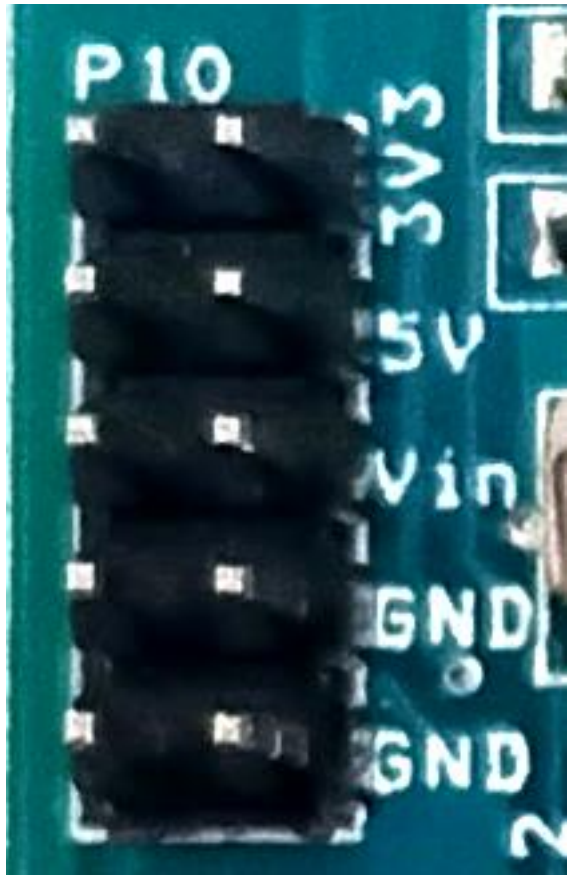
- **User LEDs (2)** - Connected to default GPIOs (GPIO33 and GPIO 4)
- **One Push Button**- Connected to default GPIO (GPIO32))
- **One 10K Analog POT** (potentiometer)-- Connected to default GPIO (GPIO36)
- **LDR** (Light Dependent Resistor) -- Connected to default GPIO (GPIO39)

These components are connected to default I/O pins via jumpers, meaning you can easily reconfigure them to any other I/O pin for different applications.

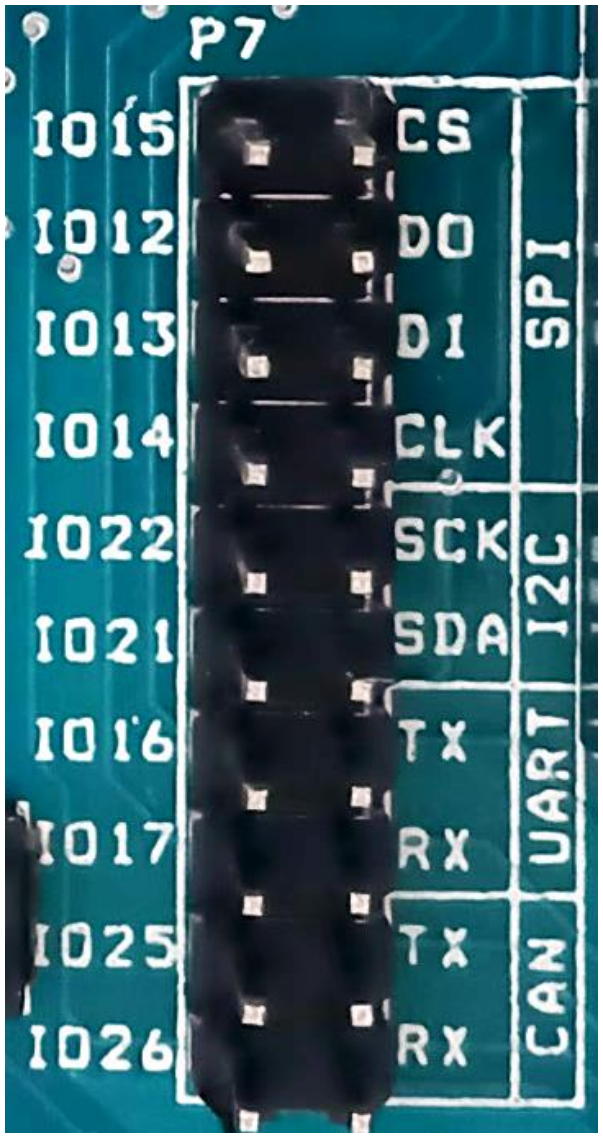
On-board Expansion Connectors

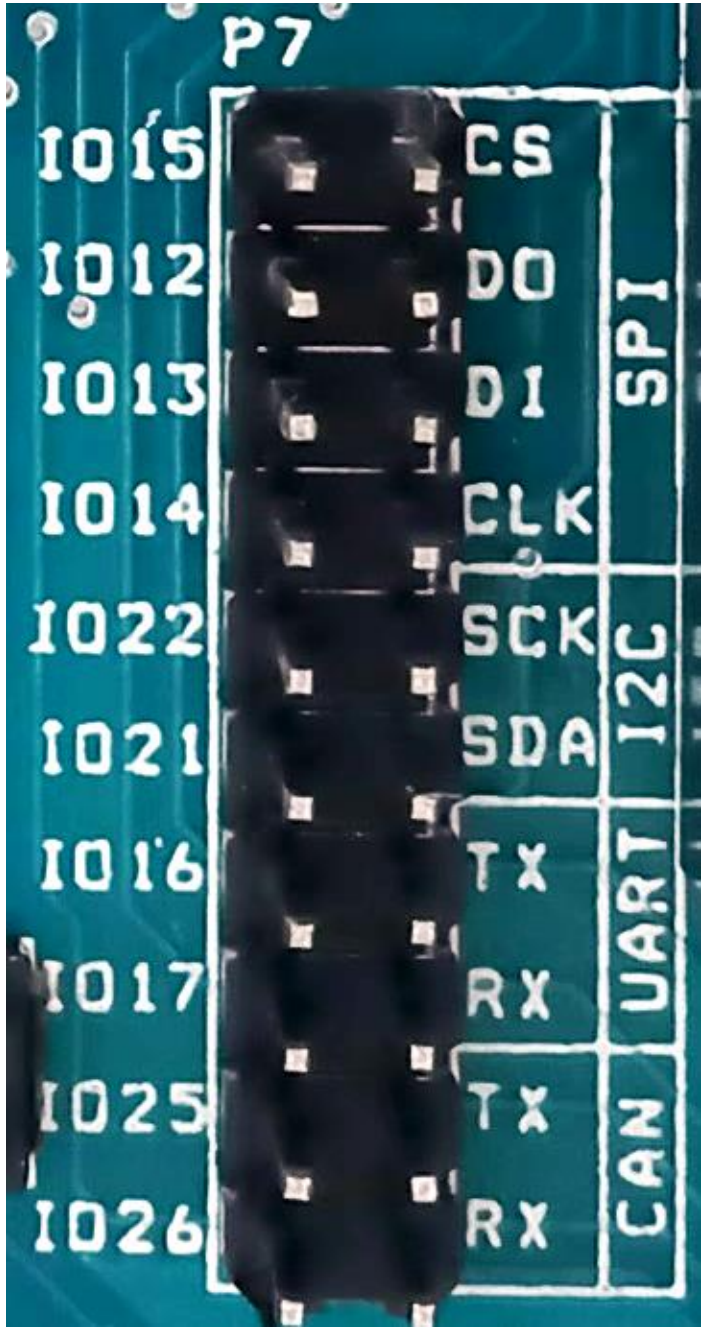
- **Power Connectors/Headers**
  - 5V, 3.3V, and Gnd to power additional components.Modules.
- **Communication Connectors/Headers**
  - On-board connectors for **UART, I2C, SPI, and CAN Modules**.
  - UART (P4), CAN (P4), I2C (P3), SPI (P5).
  - Similar to the user components, these can be connected to any alternate I/O pins using jumpers.

Power Connectors/Headers



Communication Connectors/Headers





GPIO PIN	Purpose
GPIO 32	S3 SW User Button
GPIO 33	LED1 – User LED
GPIO 04	LED2- User LED
GPIO 36	POT – Analog Input
GPIO 39	LDR – Analog Input

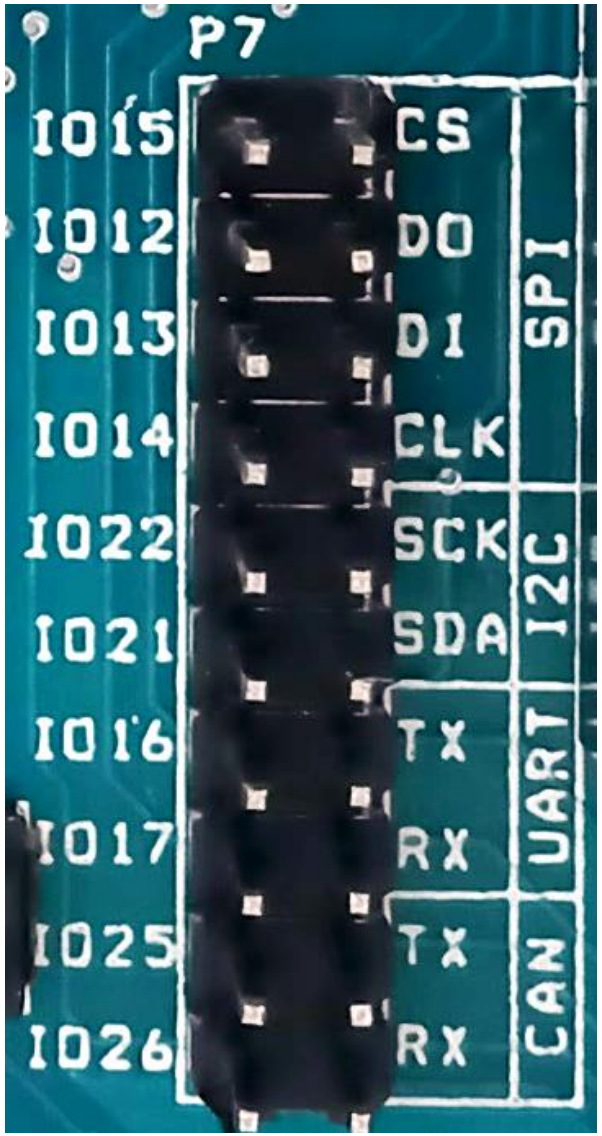
SPI	Purpose
GPIO 15	Chip Select CS
GPIO 12	Data Out
GPIO 13	Data In
GPIO 14	Serial Clock SCL

I2C	Purpose
GPIO 22	Serial Clock SCK
GPIO 23	Serial Data Association - SDA

UART	Purpose
GPIO 16	TX
GPIO 17	Rx

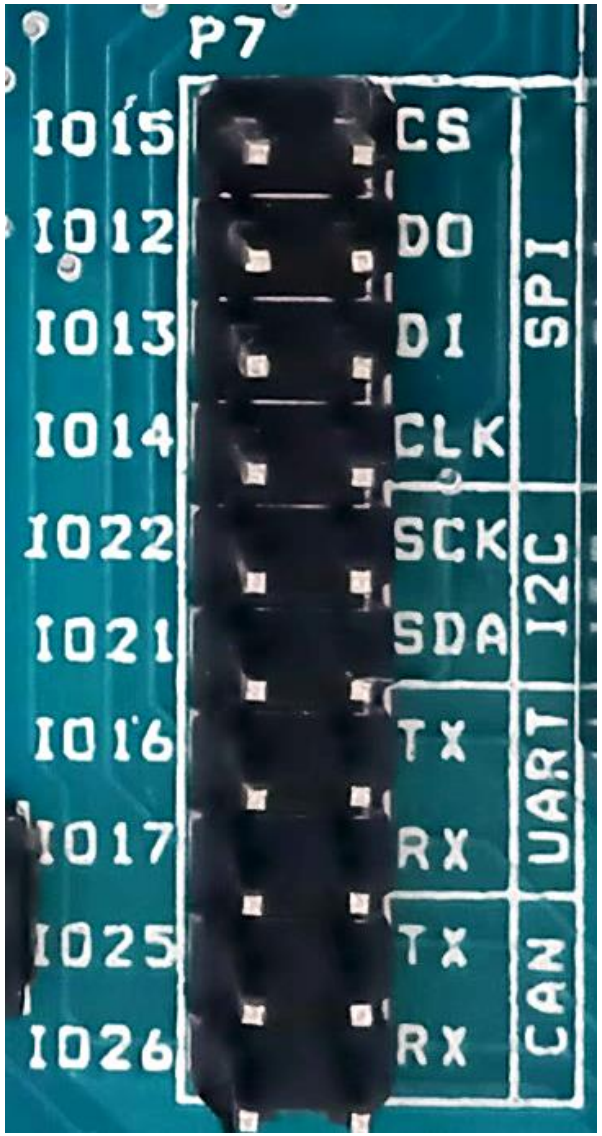
CAN	Purpose
GPIO 26	TX
GPIO 27	Rx

Pin Labels & Protocols Jumper Configuration



- These headers allow for easy connection of SPI, UART, and I2C devices directly to the development board, making prototyping simple and organized.
- The pin labels correspond to standard electronic communication protocols used to connect the main microcontroller to other sensors and components.
- Here's a breakdown of the labels and the protocols they represent:

Pin Labels & Protocols Jumper Configuration Cond..

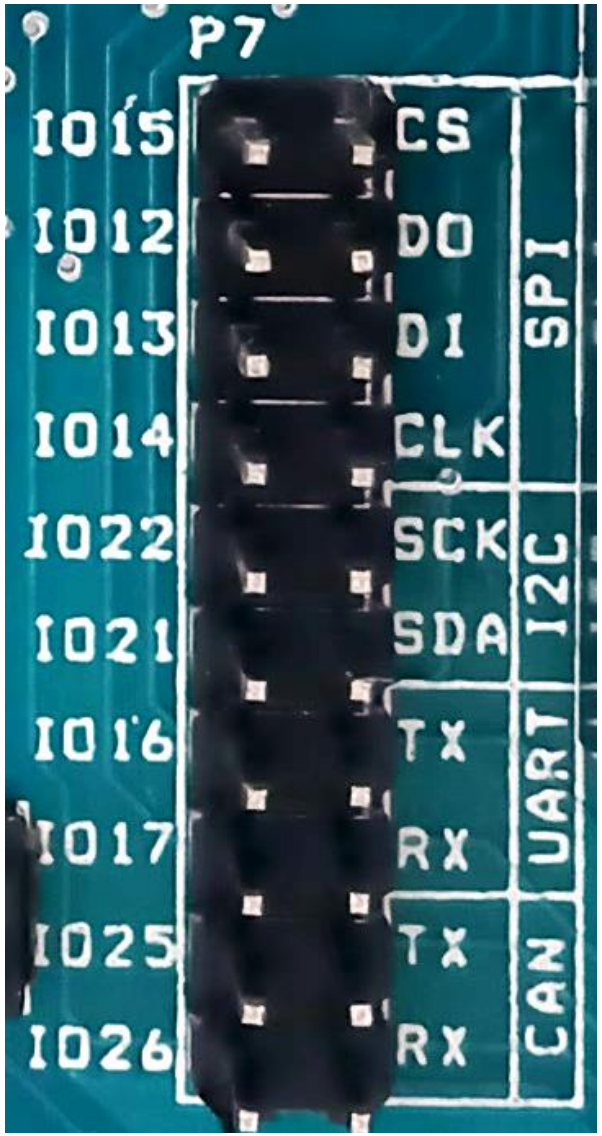


SPI (Serial Peripheral Interface)

This is a synchronous serial communication protocol. The pins for this protocol are not fully visible but typically include:

- **MOSI (Master Out Slave In):** For sending data.
- **MISO (Master In Slave Out):** For receiving data.
- **SCK (Serial Clock):** A shared clock signal.
- **CS (Chip Select):** To select a specific device.

Pin Labels & Protocols Jumper Configuration Cond..



- **I2C (Inter-Integrated Circuit)**

This is a two-wire, multi-master serial bus. The pins for this protocol are clearly labeled:

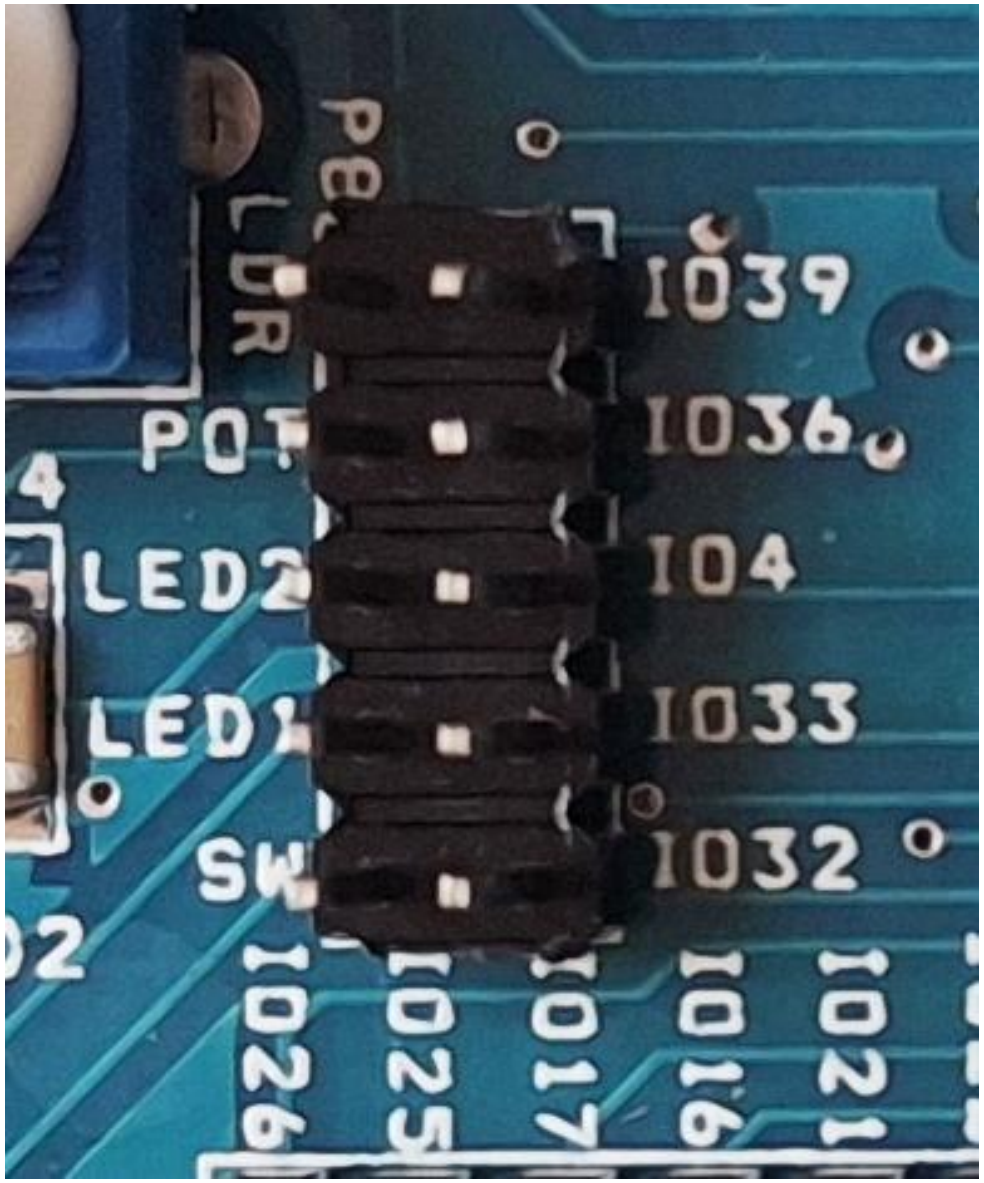
- **SCL (Serial Clock Line):** This pin carries the clock signal.
- **SDA (Serial Data Line):** This pin carries the data.

- **UART (Universal Asynchronous Receiver-Transmitter)**

This is a serial communication protocol that uses two dedicated lines for data transmission and reception:

- **TX (Transmit):** Sends data.
- **RX (Receive):** Receives data.

On Board Sensors Configuring Jumpers

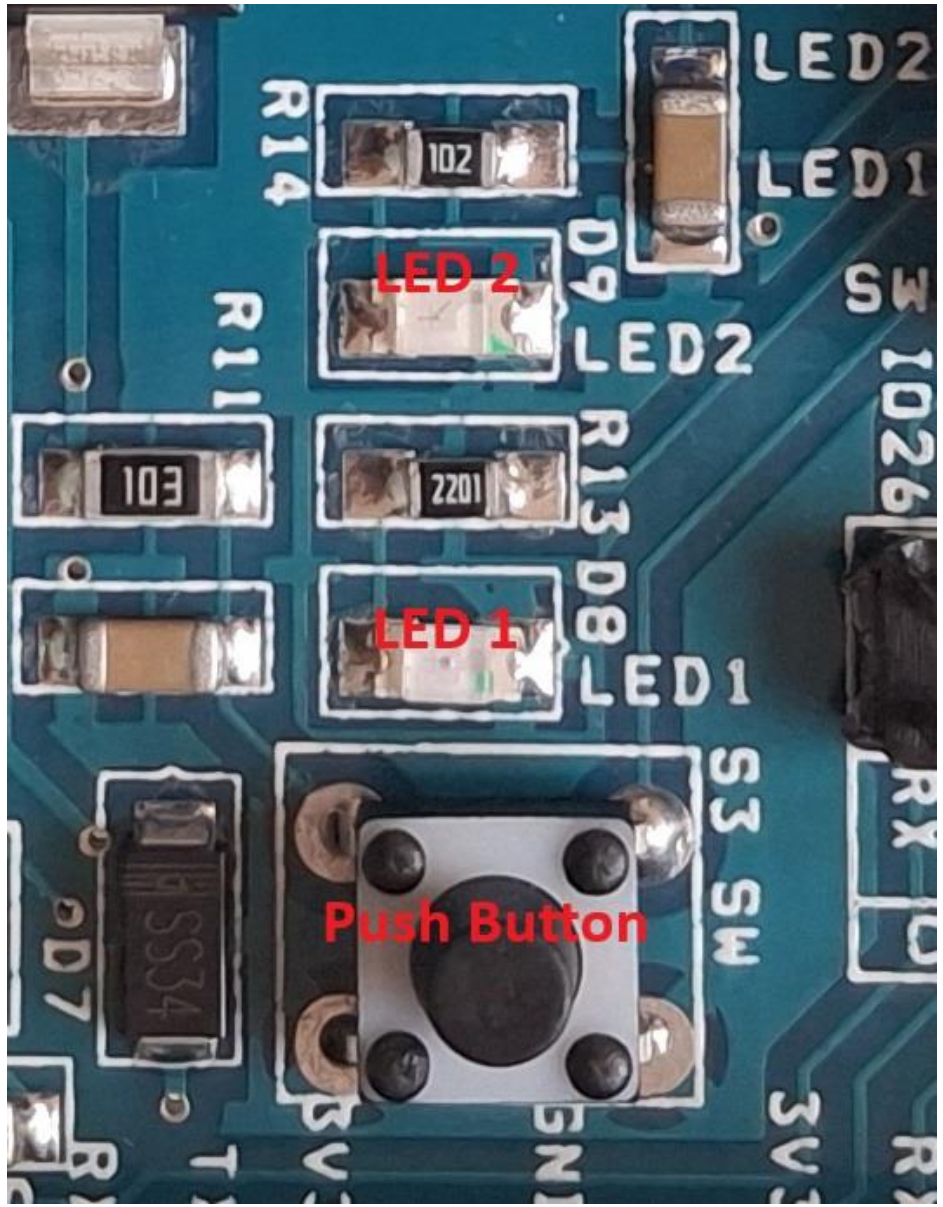


- **User button (SW1)**: A physical button that can be pressed to trigger a specific action in your code.
- **USER_LED1 and 2**: Light-emitting diodes (LEDs) that you can program to turn on and off.
- **POT**: Likely a **potentiometer**, which is a variable resistor. You can use it to create an analog input, like controlling the brightness of an LED or the speed of a motor.
- **LDR**: A **light-dependent resistor**. Its resistance changes based on the amount of light it receives. This allows you to measure light intensity.

Buttons (Boot and Reset)

- **BOOT Button**
  - This is used to put the ESP32 into bootloader mode, which is necessary for flashing new firmware or code.
- **RESET Button**
  - This button restarts the board and re-runs the uploaded program from the beginning.

On Board LEDs and Button (Digital I/O)



LED2 at GPIO 04
User LED 2

LED 1 at GPIO 33
User LED 1

Switch S3 – at GPIO 32
User Button

On board Pot and LDR (Analog Input)

Analog Input
LDR at GPIO 39

Analog Input
POT at GPIO 36

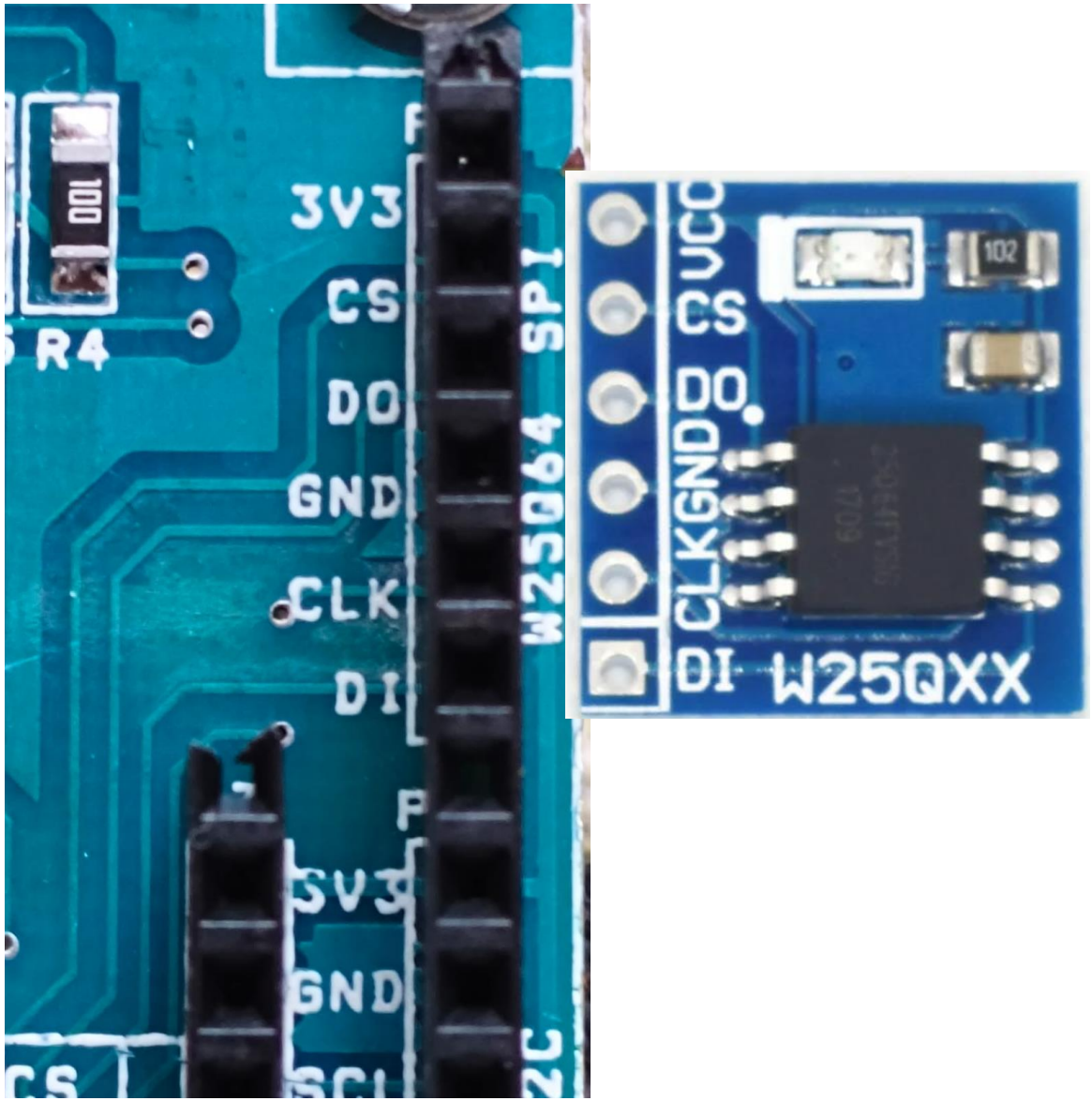
On-board Expansion Connectors Cond..

- **Specialized Connectors/Headers**

  - The board has connectors for direct plugging of a **3-Axis Gyroscope and Accelerometer (MPU6050)** and a **CAN Transceiver (TJA1050)**.

  - It also includes an **USB2UART FT232 Module** and an **8 MB SPI based Data Flash module (W25Q64)**.

W25QXX Flash Memory Module Interfacing



- On the left, the photo shows a section of an **ESP32 development board**, with pins labeled for the **SPI protocol**.
- This protocol is used to communicate with various devices, including sensors, displays, and memory chips.

The relevant pins shown here are:

- **3V3:** The power supply pin, providing 3.3 volts.
- **CS (Chip Select):** This pin is used to select a specific device on the SPI bus when multiple devices are connected.
- **DO (Data Out):** Also known as **MISO (Master In Slave Out)**, this is the data line for the slave device to send data back to the master (the ESP32).
- **GND:** The ground pin, the common reference point for all electrical components.
- **CLK (Clock):** The clock signal that synchronizes the data transfer between the master and slave devices.

W25QXX Flash Memory Module

- On the right, a small blue module is a **W25QXX flash memory module**.
- This module is a type of **non-volatile memory** used to store data that needs to be retained even when the power is off.
- It's often used for things like storing sensor data logs, configuration files, or other information.

W25QXX

- This is the product family name for the flash memory chip, such as the W25Q64 shown in the image, which stands for **64-megabit** flash memory.

W25QXX Flash Memory Module

Pin Labels

- The module has pins that correspond to the SPI protocol, allowing it to be easily connected to the development board.
- These pins are:

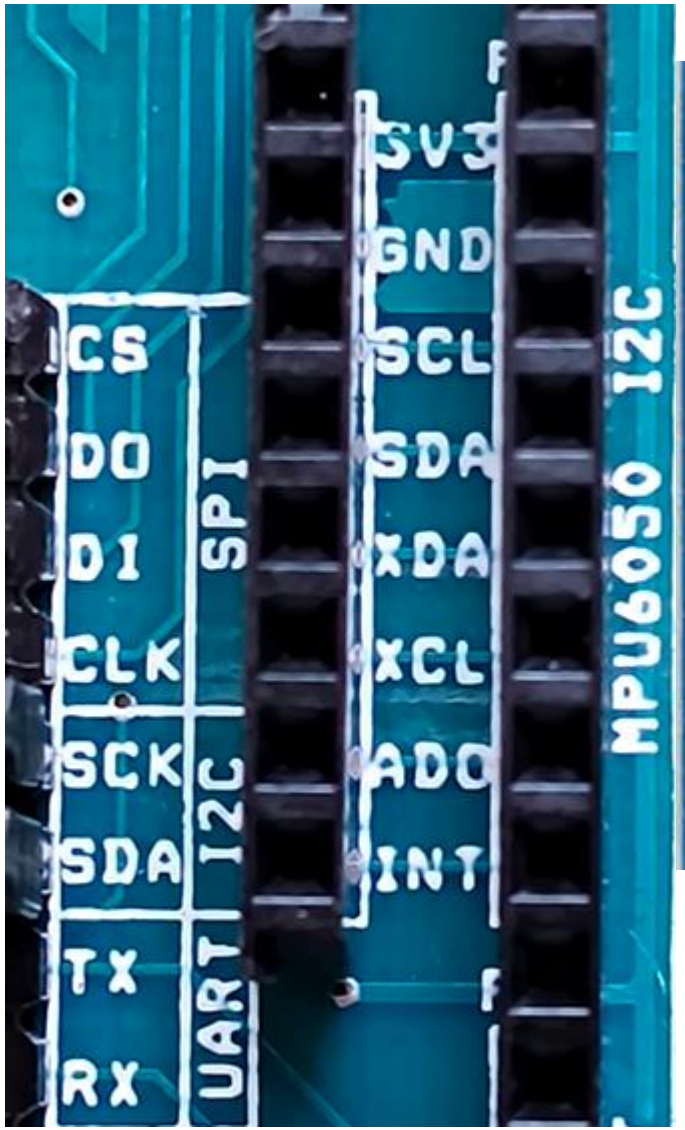
  - **VCC:** Power supply.
  - **CS (Chip Select):** To activate the memory chip.
  - **DO:** Data output.
  - **GND:** Ground.
  - **CLK (Clock):** Clock signal.
  - **DI (Data In):** Also known as **MOSI (Master Out Slave In)**, this is the data line for the master (the ESP32) to send data to the slave (the memory chip).

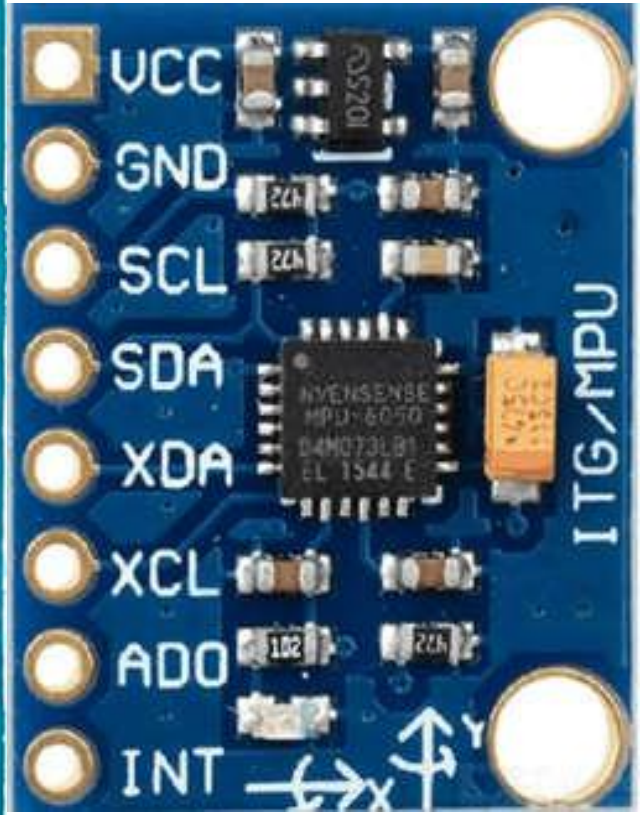
W25QXX Flash Memory Module

Pin Mapping Between Board and W25QXX Module

ESP32 Board Pin Label	W25QXX Module Pin
3V3	VCC
CS	CS
DO	DO
GND	GND
CLK	CLK
DI	DI

MPU6050 Module Interfacing





Pin Descriptions of I^2C (Inter-Integrated Circuit)

The pins on the board are labeled to match the MPU-6050 sensor module. They provide power and communication interfaces for a microcontroller. Here's what each pin is for:

- **VCC**: This pin is for the power supply, typically 3.3V or 5V.
- **GND**: The ground pin, which completes the electrical circuit.
- **SCL (Serial Clock)**: This is the clock line for **I^2C (Inter-Integrated Circuit)** communication. It's a synchronous serial protocol used to send and receive data.
- **SDA (Serial Data)**: This is the data line for the I^2C protocol. It carries the actual data.
- **XDA** and **XCL**: These are pins for an **auxiliary I^2C bus**. They allow the MPU-6050 to connect to other sensors, like a magnetometer, to create a 9-axis system.
- **ADO (I^2C Address)**: This pin sets the I^2C address of the device. Changing its state (high or low) allows you to use two MPU-6050 sensors on the same I^2C bus.
- **INT (Interrupt)**: This pin can be configured to send an interrupt signal to the microcontroller when specific events occur, such as a change in motion.

Pinout and Functionality of MPU6050

The MPU-6050 module has several pins for power, communication, and other functions. Here's a breakdown of the pins visible in your image:

- **VCC**: This pin supplies power to the module. It typically requires a voltage between 3.3V and 5V.
- **GND**: This is the ground pin, providing the common reference point for the circuit.
- **SCL (Serial Clock)**: A pin for the I^2C communication protocol. It's used for the clock signal that synchronizes data transfer between the microcontroller and the sensor.
- **SDA (Serial Data)**: This is the data pin for I^2C communication. It carries the actual data being sent to and from the sensor.

Pinout and Functionality

- **XDA** and **XCL**: These pins are used for the auxiliary I²C bus. They allow the MPU-6050 to connect to other I²C devices, such as a magnetometer, to create a 9-axis motion tracking system.

- **ADO (I²C Address)**: This pin determines the I²C address of the module. Tying this pin to GND or VCC allows you to select between two different addresses, which is useful for using multiple MPU-6050 modules on the same I²C bus.

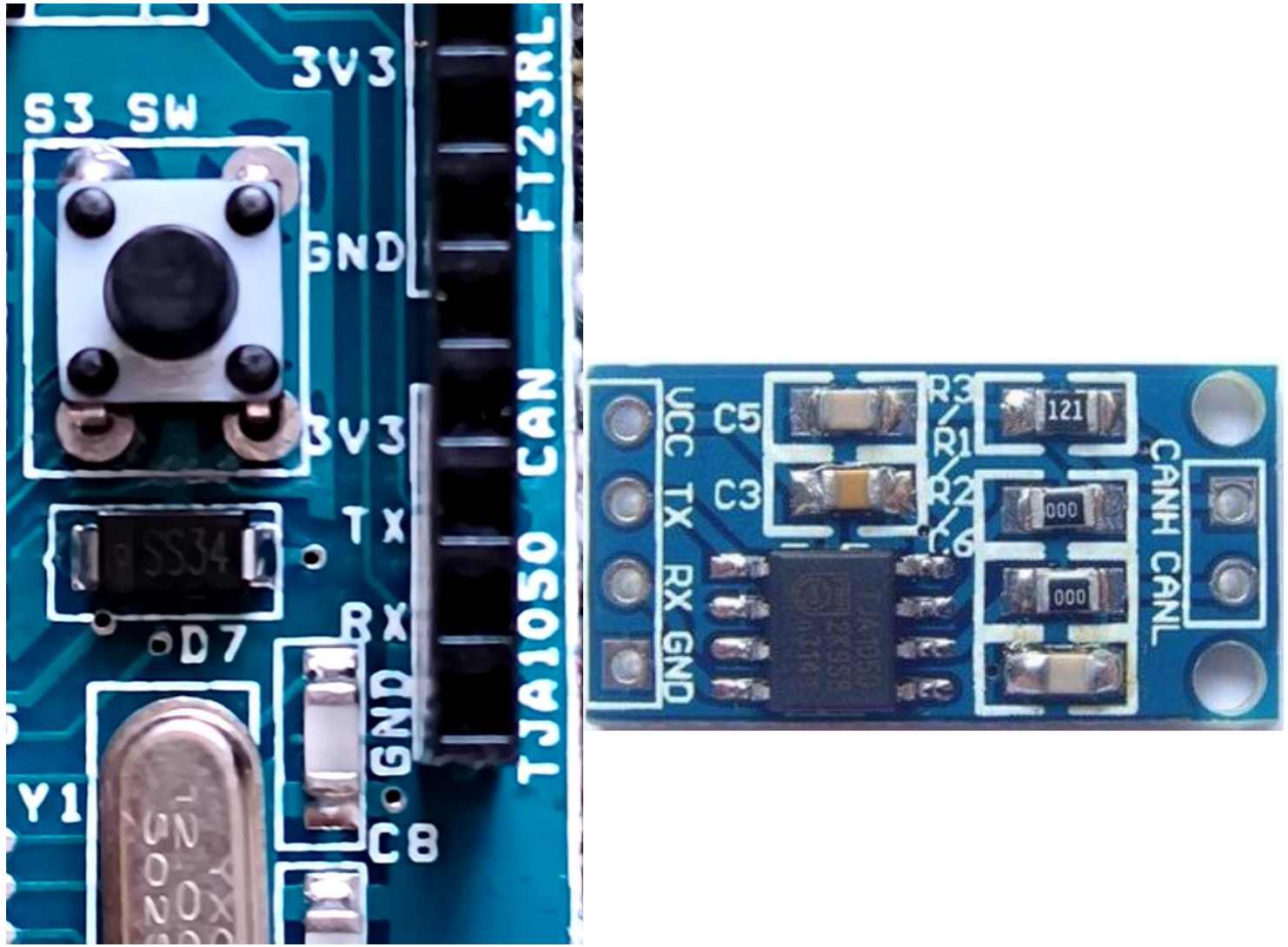
- **INT (Interrupt)**: This pin can be configured to send an interrupt signal to the microcontroller when a specific event occurs, such as a change in motion or a data ready state.

MPU6050 Module

Pin Mapping Between Board and MPU6050 Module

ESP32 Board Pin Label	MPU6050 Module Pin
3V3	VCC
GND	GND
SCL	SCL
SDA	SDA
XDA	XDA
XCL	XCL
ADO	ADO
INT	INT

TJA1050 CAN bus Module Interfacing



TJA1050 CAN bus transceiver module

- The module acts as an **interface** between a microcontroller's CAN controller and the physical CAN bus.
- It converts the digital signals from the microcontroller into the differential signals required to communicate over the CAN bus.
- It also does the reverse, converting the bus signals back into digital signals for the microcontroller.

Pinout and Functions

The pins on the module are clearly labeled, indicating their purpose:

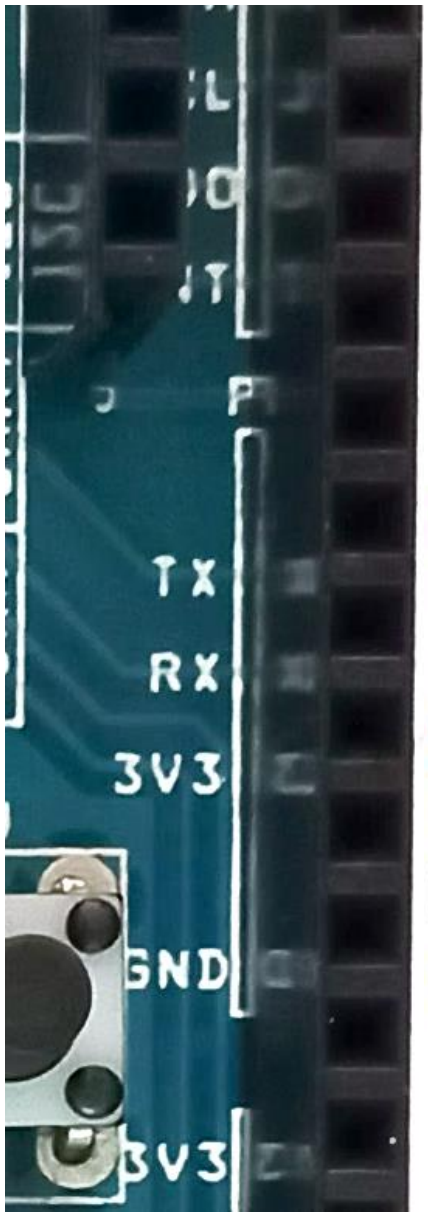
- **VCC**: Power supply input, typically 5V.
- **GND**: Ground connection.
- **TXD**: **T**ransmit **D**ata. This is the input for data being sent from the microcontroller to the CAN bus.
- **RXD**: **R**eceive **D**ata. This is the output for data received from the CAN bus, which is sent to the microcontroller.
- **CANH**: **CAN H**igh. This is one of the two differential signal lines of the CAN bus. It's the high-level wire.
- **CANL**: **CAN L**ow. This is the other differential signal line of the CAN bus. It's the low-level wire.

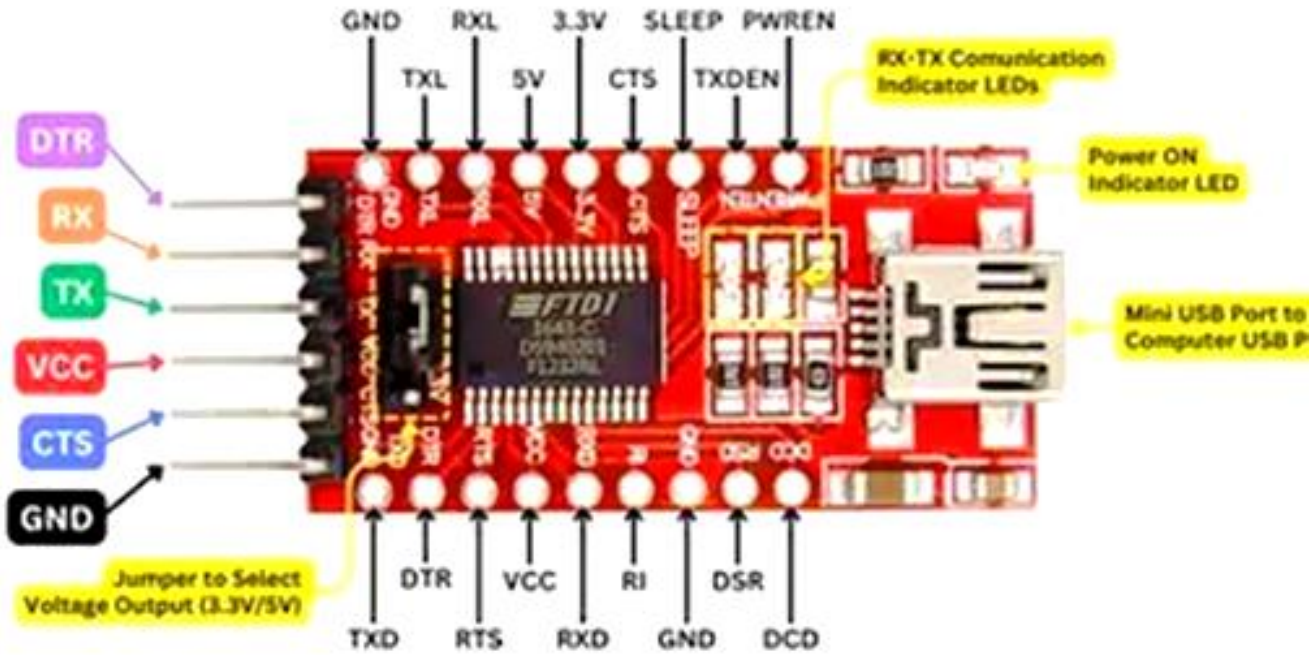
TJA1050 CAN Module

Pin Mapping Between Board and MPU6050 Module

ESP32 Board Pin Label	TJA1050 CAN Module Pin
3V3	VCC
TX	RX
RX	TX
GND	GND

FTDI FT232RL Module Interfacing





FTDI FT232RL

- The main function of this board is to act as a bridge between your computer's USB port and a microcontroller's serial communication pins.

- To program them or to receive serial data (for debugging, for example), you need a separate USB-to-serial converter. The FTDI breakout board serves this purpose.

Pinout and Connections

As shown in the image, the board has a set of pins that you connect to your microcontroller.
These typically include:

- **GND:** Ground connection.
- **RXD:** Receive data pin. This connects to the microcontroller's **TX** (transmit) pin. Data sent from the microcontroller is received here.
- **TXD:** Transmit data pin. This connects to the microcontroller's **RX** (receive) pin. Data sent from the computer is transmitted here.

Pinout and Connections

- **VCC:** Power supply pin, which provides power to the connected device. The voltage level is often selectable between **3.3V** and **5V** using a jumper on the board.

- **DTR:** Data Terminal Ready pin. This pin is crucial for **auto-resetting** the microcontroller. When you upload a new program (or "sketch" in Arduino terminology), the DTR signal automatically pulls the microcontroller's reset pin low, allowing the new code to be loaded without you having to manually press the reset button.

- **CTS:** Clear To Send. Used for hardware flow control, but often left unconnected in simple applications.

FTDI FT232RL Module

Pin Mapping Between Board and FT23RL Module

ESP32 Board Pin Label	FTDI FT232RL Module Pin
DTR	DTR
GND	GND
TX	RX
RX	TX
3.3V	VCC
CTS	CTS
GND	GND

Understanding the Jumper System

The board uses **jumpers** to connect on-board components like the user button, LEDs, potentiometer (POT), and Light Dependent Resistor (LDR) to specific GPIO (General Purpose Input/Output) pins on the ESP32 module.

- **Default Configuration:**

  - Initially, the jumpers connect these components to a set of pre-defined, "default" GPIO pins.

  - This allows you to use the board out of the box without needing to change anything.

Understanding the Jumper System Cond..

Custom Configuration:

- The primary benefit of this system is that you can **remove the jumpers** from their default positions.
- This disconnects the component from the default pin, freeing it up for another use.
- You can then use a jumper wire to connect the component to **any other available GPIO pin** on the board, according to your specific design needs.

Understanding the Jumper System Cond..

Example :Reconfiguring USER LED 2

Default State

- Initially, **USER LED 2** is connected to its default GPIO pin (in this case, GPIO4) via a jumper.

Removal

- To change this connection, you would **remove the jumper** that links USER LED 2 to GPIO4.

Understanding the Jumper System Cond..

New Connection

- You then use a jumper wire to connect one end of the **USER LED 2** pin to your desired new GPIO pin, such as **GPIO25**.

- This method provides **flexibility** for prototyping, especially when you need to use a specific GPIO pin for a different peripheral that might have specific requirements (like PWM or an interrupt).

Understanding the Jumper System Cond..

Additional Features

The board also offers other useful features for a developer:

Multiple User Buttons:

- The **BOOT button**, typically used for entering programming mode, can also be configured to function as a regular **user button** during program runtime.
- This gives you a second button to work with.

Understanding the Jumper System Cond..

Dual Analog Inputs

- The board includes a potentiometer (POT) and an LDR, both of which are **analog inputs**. This allows you to test and develop applications that read variable resistance values.

Multiple LEDs

- With two user LEDs, you can test different visual feedback loops, such as status indicators or simple blinking patterns.

Sample Programs

https://github.com/asc-es-iot/asc-esp32-iot-board

LED Blink at https://github.com/asc-es-iot/asc-esp32-iot-board/tree/main/basics/led_blink

Button and LED at https://github.com/asc-es-iot/asc-esp32-iot-board/tree/main/basics/button

Analog Input (POT) and Serial Output at

https://github.com/asc-es-iot/asc-esp32-iot-board/tree/main/basics/potentiometer

Light Sensor LDR and Serial Output at

https://github.com/asc-es-iot/asc-esp32-iot-board/tree/main/basics/ldr

https://sites.google.com/view/tt-iiot

Arduino IDE Installation

To install the Arduino IDE, you need to download the appropriate version for your operating system from the official Arduino website.

Download the Installer:

- Navigate to the Arduino Software page and download the .exe installer for Windows, the .dmg file for macOS, or the .AppImage for Linux.

- For Windows, the installer is recommended as it includes the necessary drivers.

Arduino IDE Installation Cond..

Run the Installer

- **Windows**: Double-click the downloaded .exe file and follow the prompts. Be sure to allow the driver installation when prompted.
- **macOS**: Open the .dmg file and drag the Arduino application into your Applications folder.
- **Linux**: Make the .AppImage file executable by right-clicking it, going to **Properties** or **Permissions**, and checking "Allow executing file as program." Then, double-click to run it.

Launch the IDE

- Once installed, open the Arduino IDE.

ESP32 Board Installation

By default, the Arduino IDE does not support ESP32 boards. You must add the ESP32 board manager package.

Add Board Manager URL

- Open the Arduino IDE and go to **File > Preferences** (on macOS, this is **Arduino > Preferences**).
- In the "Additional Board Manager URLs" field, paste the following URL: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
- If there are already other URLs in the field, you can add a comma , and then paste the ESP32 URL. Click **OK**.

Install the ESP32 Package

- Go to **Tools > Board > Boards Manager...**
- In the search bar, type esp32.
- Select the **"esp32 by Espressif Systems"** entry and click the **Install** button. This may take a few minutes as it downloads the necessary files.

Select Your Board and Port

- After the installation is complete, close the Boards Manager.
- Connect your ESP32 board to your computer via USB.
- Go to **Tools > Board** and navigate to the **ESP32** section. Select your specific ESP32 board model (e.g., “DOIT ESP32 DEVKIT V1").
- Go to **Tools > Port** and select the COM port that corresponds to your ESP32 board.
- Your Arduino IDE is now configured to program ESP32 boards.

Open the Example Sketch

- Launch the **Arduino IDE**.
- Go to **File > Examples > 01.Basics > Blink**.
- This will open a new window with the pre-written Blink code.

Configure Your Board and Port

Before you can upload the code, you must tell the IDE which board you are using and which port it is connected to.

- Connect your Arduino board to your computer using a USB cable.
- In the Arduino IDE, go to **Tools > Board** and select your specific board (e.g., **DOIT ESP32 DEVKIT V1**).
- Go to **Tools > Port** and select the **COM port** that corresponds to your board. On Windows, it will be labeled COMx, while on macOS and Linux, it will likely be something like

Upload the Program

Once your board and port are correctly selected, you can upload the code.

- Click the **Upload** button (the right-arrow icon) in the toolbar.
- The IDE will compile the code and then upload it to your board. You will see a progress bar at the bottom of the IDE.
- The TX (transmit) and RX (receive) LEDs on your Arduino board will blink rapidly during the upload process.
- Once the upload is complete, a "Done uploading" message will appear in the IDE's status bar.

Verify the Blink

- After a successful upload, the onboard LED (connected to pin 33 on the TekTork IoT board) will begin to **blink**.

- It will turn on for one second, then off for one second, and this pattern will repeat continuously.

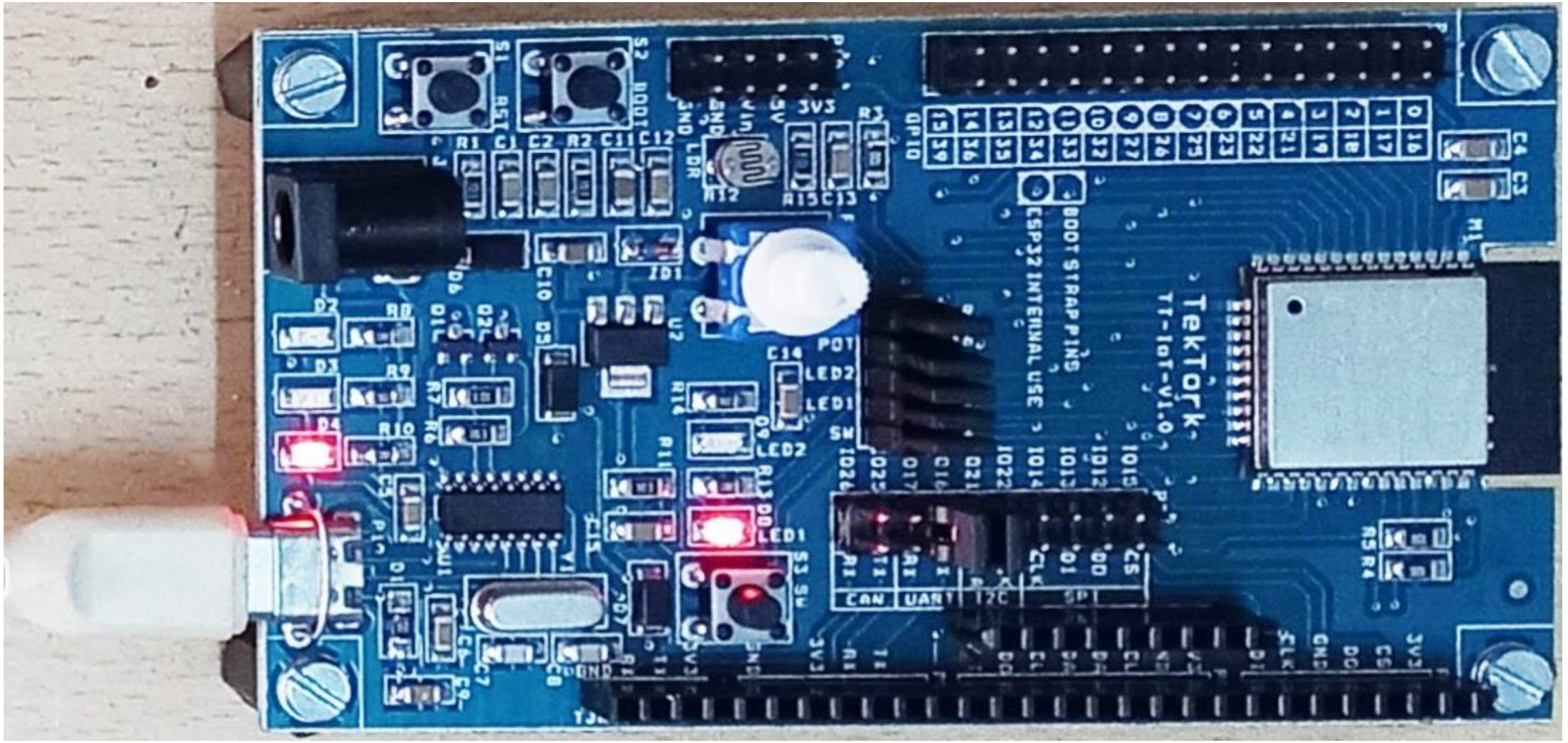
Sample Programs for Testing the Kit

```
// the setup function runs once when you press reset or power the board
void setup()
{
  // initialize digital pin GPIO33 as an output. Tektork Board LED Connected to GPIO33
  pinMode(33, OUTPUT);
}

// the loop function runs over and over again forever

void loop()
{
  digitalWrite(33, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(33, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Output


TekTork-
TT-IoT-V1.0
ESP32 INTERNAL USE
BOOT STRAPPING
CAN
UART
I2C
SPI
LED1
LED2
POT
LDR

Physical Specifications & Use Case

- The PCB is a **High quality double sided Glass Epoxy** board with a size of **60mm * 105mm**.

- Due to its comprehensive set of features and robust design, it's described as being "Best Suitable for **Training and Prototyping IoT Products**.

Why This Board Is a Perfect Fit for You

Modular & Flexible

- Jumper-based IO mapping lets you experiment with alternate configurations—perfect for assessments and debugging.

Simulation-Friendly

- You can prototype logic in Wokwi or Tinkercad, then deploy it directly to this board with minimal changes.

System-Oriented

- Supports layered architecture—hardware, RTOS, networking(WiFi.BLE Module and LwIP Stack), and cloud integration.

Why This Board Is a Perfect Fit for You

- The board you have is a **TekTork TT-IoT-V1.0**, which is an ESP32-based development board.

- This board is well-suited for a variety of projects, especially for beginners and those working with the Internet of Things (IoT).

Here is why this board is a great choice:

Integrated Wi-Fi and Bluetooth:

- The core of the board is an **ESP32 microcontroller**, which has built-in Wi-Fi and Bluetooth capabilities.
- This makes it ideal for IoT projects that require wireless communication, such as connecting to a network, controlling devices remotely, or collecting data from sensors.

Clear Pin Labels

- The silkscreen on the board clearly labels the GPIO pins and their functions. This makes it easy to connect external sensors and components without constantly referring to a datasheet.
- The labels also indicate specific buses like **UART**, **I2C**, and **SPI**, which are crucial for communicating with a wide range of devices. * **USB-to-Serial Converter**:
- The board has a dedicated chip (likely an FT232RL) for **USB-to-serial communication**.
- This allows you to program the board and communicate with it directly from your computer using a standard USB cable, simplifying the development process.

Breadboard Friendly Design

- The pin headers are designed to be easily plugged into a breadboard, allowing for quick prototyping and circuit building without the need for soldering.

Precautions and Connection

Connection Steps

- **Use a Data-Capable USB Cable:**
  - Not all USB cables are the same. Some are designed only for charging and lack the data lines necessary for communication between your computer and the board.
  - Use the cable that came with your board or one you know supports data transfer.
- **Connect to a USB Port:**
  - Plug one end of the USB cable into your computer's USB port and the other end into the USB-C port on your ESP32 board.

Precautions and Connection Cond..

- **Check for Drivers:**
  - Most modern operating systems will automatically install the correct drivers (like the CP210x or CH340) for the USB-to-serial converter chip on the board.
  - If the board isn't recognized, you may need to manually download and install the drivers from the chip manufacturer's website.
- **Select the Correct Port:**
  - In the Arduino IDE, go to **Tools > Port** and select the newly appeared COM port (Windows) or /dev/tty... port (macOS/Linux).
  - If you're unsure which one it is, unplug the board, check the menu, and then plug it back in to see which port appears.

Programming Precautions

- **Power Supply:**
  - Most development boards are designed to be powered directly from the USB port during programming.
  - Avoid using an external power supply at the same time, as this can cause voltage conflicts and potentially damage the board.
- **Use a Breadboard:**
  - If you're adding components like LEDs, sensors, or buttons, use a breadboard to build your circuit.
  - Avoid placing the bare board directly on a metallic surface, as this could create a short circuit and damage the board.

Programming Precautions Cond..

Hold the BOOT Button (If Necessary): Some ESP32 boards don't have an automatic reset circuit. If you get a "Failed to connect" error during upload, you may need to manually put the board into programming mode.

1. Press and hold the **BOOT** button on the board.

2. Click the **Upload** button in the Arduino IDE.

3. Wait for the "Connecting..." message to appear in the IDE.

4. Release the **BOOT** button.

5. The code should begin uploading.

Programming Precautions Cond..

Pin Usage:

Be mindful of which GPIO pins you use for your projects. Some pins have special functions or are used by the ESP32's internal systems.

- **Pins to Avoid:** It's generally a good practice to avoid GPIO 6 to 11, as they are often used for the onboard flash memory.

- **Input-Only Pins:** Pins 34-39 are input-only and cannot be used as outputs.

Troubleshooting:

If you encounter problems, first check the simple things. Is the USB cable working? Is the correct port selected? Have you installed the board's drivers? Checking these can save you a lot of time.

Thank You