# **Chapter 1** Getting Started with ESP32 Development Board

The billions of devices connected to the Internet collect and exchange data from the Internet of Things (IoT) [1]. Due to the availability of low-cost microprocessors and wireless networks, all devices, from smartphones to consumer products, are now accessible to everyone through IoT technologies. Modern devices are equipped with sensors that make it possible to perceive the environment and automatically transmit real-time data. IoT devices bring the digital and physical world together, allowing things to process information and respond quickly and accurately.

As a hardware engineer, striving to develop cost-effective IoT solutions in a specific problem area is a constant challenge [2]. In the past, the world of IoT development was characterized by the clear division of processors, wireless communications, and software applications, which increased the expense and complexity of bringing ideas to life. However, the emergence of the ESP32 introduced by Espressif Systems in 2016 is a game-changer. With the ESP32, wireless communication components are integrated into a single processor chip, reducing development costs and making complex IoT applications more accessible to understand.

By the end of this chapter, you should have a firm grasp of IoT principles, comprehend why the ESP32 is the preferred processor for IoT development, and have the necessary practical skills to set up and program the ESP32 development board using the Arduino IDE.

In this chapter, you will be able to

- Explore real-world applications and the transformative impact of IoT on industries and daily life.
- Explore the key features of the ESP32 that make it an optimal choice for hardware developers.
- Get started with setting up the development environment for ESP32.
- Identify the ESP32 Development Board pinouts and its functions.

## 1.1 The “Things” on the Internet of Things

According to the Global Standards Initiative, the Internet of Things (IoT) is an interconnected physical device, also known as “things,” embedded with electronics, software, sensors, actuators, and wireless communication to collect and transmit data over the network [1]. In 1999, the term “Internet of Things” was coined by *Kevin Ashton*, *the Father of IoT*. He stated that “computers that knew everything there was to know about things, using data they gathered without any help from us, we would be able to track and count everything and greatly reduce waste, loss, and cost.” The main goal of IoT devices is to enable a world where everyday objects can communicate, share data, and contribute to a smarter, more efficient, and connected environment. This vision has impacted the development and application of IoT technologies in several industries, altering how people engage with and perceive the world. In the present, as per Oner V. [2] and Khan M.A. [3] perspectives provide a summary of the definition of IoT.

- **Connectivity**: Connecting devices to the Internet, local networks, and one another is the core of IoT. Connectivity is essential for communication and data sharing for the "things" on the IoT to work.
- **Identification**: Every IoT device in a network has a unique identity. This identification ensures that each device is identifiable, making it easier to manage and communicate with many connected devices.

- **Automation**: IoT devices are made to function independently, which means they can complete tasks and make decisions with minimal to no human intervention. In a variety of applications, this autonomy improves responsiveness and efficiency.
- **Interoperability**: Regardless of the manufacturers or technologies involved, it describes how various systems and devices work together and communicate without problems. It guarantees the harmonious operation of various IoT components.
- **Scalability**: The ability of an IoT system to grow and adapt as the number of connected devices increases. Expanding the network without compromising its efficacy is possible with a scalable IoT infrastructure to manage the growing number of devices and data.
- **Security**: This involves implementing measures to protect the confidentiality, integrity, and availability of the data exchanged between devices. Robust security measures are critical to protect against unauthorized access, data breaches, and other cyber threats in the interconnected IoT ecosystem.
- **Real-Time**: IoT devices operate and communicate in real-time, enabling instant data processing, analysis, and response. This real-time capability is critical for applications that require immediate feedback.
- **Intelligence**: IoT devices have embedded processors, algorithms, and sensors that give them intelligence. Smart devices can gather, process, and react to data on their own, which helps with adaptive behavior and better decision-making. Advances in embedded processors and software development have enabled deploying optimized Artificial Intelligence (AI) predictive models to constrained devices. Embedded AI empowers IoT devices by allowing them to make intelligent decisions and perform adaptive responses based on data analysis and learning.

## 1.2 Applications of the Internet of Things

### 1.2.1 Consumer IoT

The digital age has emerged IoT as a transformative force, connecting people's daily lives with technologies. Although there are many IoT applications, they can be divided into two main categories: consumer and industrial IoT. *Consumer IoT* integrates smart and connected devices into everyday consumer products and services. *Home automation*, also known as *smart home* [3], integrates advanced technologies and systems to automate and remotely control various aspects of a home. The primary goal is to increase comfort, efficiency, safety, and energy savings. Smart homes use interconnected devices and sensors that can be controlled through a central hub or smartphone, allowing homeowners to monitor and manage their homes remotely.

Smart lighting, thermostats, plugs and switches, door locks, security cameras, entertainment systems, and appliances are common home automation components. In addition to being more comfortable and convenient, home automation increases home security and energy efficiency.

There are smart home IoT devices available commercially built on ESP32 processors. One such product is Shelly [4] as shown in **Figure 1**, a range of smart home automation devices that let users control their house with a tablet or smartphone.


**Figure 1**: *The bare circuit board of the Shelly Plus 1 WiFi-operated Smart Switch reveals an ESP32 chip inside the device. Adopted from Shelly Web Page* [4] *and ESPHome Devices* [5].

Another example is the SONOFF POW Elite [6], a smart energy meter for a smart home, depicted in **Figure 2**. This power meter switch, which is based on an ESP32 chip, is capable of tracking how much power is used by each appliance in the room. It allows users to control and monitor their home appliances using a smartphone remotely.

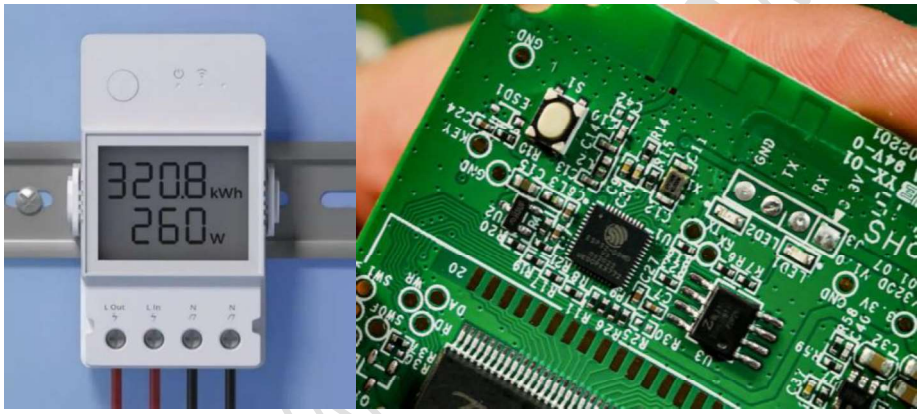



**Figure 2**: *An ESP32 chip is visible on the circuit board of a Sonoff POW Elite energy meter. Taken from Sonnof Official and Smarty Ninja* [6], [7].

## 1.2.2 Industrial IoT

*Industrial IoT* (IIoT) implements IoT technologies in business and industrial settings to enhance productivity, streamline workflows, and facilitate data-driven decision-making. IIoT uses sensors and networked devices to gather and process data in real-time, improving productivity and providing operational insights. *Smart building management* is one instance of this, which uses technology to design intelligent, effective, and sustainable structures. An example of a real-world application is the HVAC controller [8], which, as **Figure 3** illustrates, uses the ESP32 development board as its primary processor. It was used to monitor and manage the laboratory HVAC system at a pharmaceutical company.

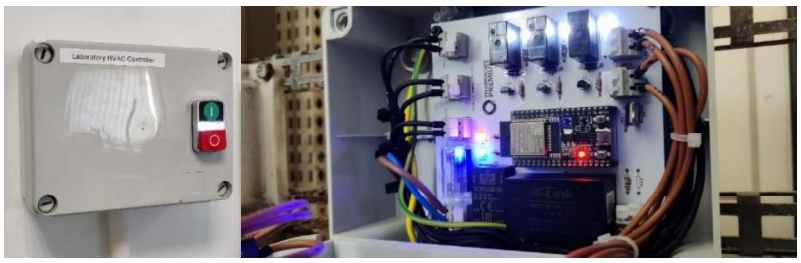


**Figure 3**: *Phamacare Premium laboratory HVAC controller using ESP32 development board. Taken from Keith Micallef* [8].

Building safety has undergone a paradigm shift with the integration of IoT into smoke and fire detection systems, which offers a proactive and intelligent method of safeguarding people and property. It enables more effective and coordinated emergency responses and increases detection speed and accuracy. One example is X-sense CO and smoke detectors [9]. This type of alarm offers 2-in-1 protection by simultaneously detecting the presence of carbon monoxide (CO) and smoke in the air, as shown in **Figure 4**.

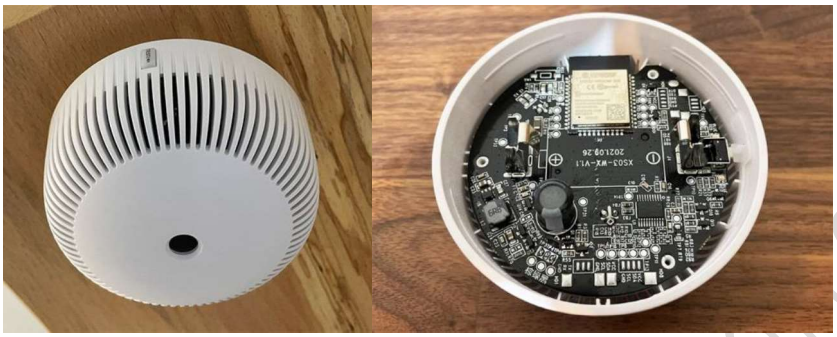


**Figure 4**: *X-Sense XS03-WX WiFi Smoke Detector and its internal hardware shows an ESP32 module. Adopted from M Reviews* [10].

*Smart farming* [3], or *precision agriculture*, uses advanced technologies such as IoT to optimize various aspects of agricultural practices. This integration of digital solutions allows farmers to make data-driven decisions, enhance efficiency, and improve overall productivity. As technology advances, integrating IoT into agriculture is expected to play a critical role in addressing the challenges of food security, environmental protection, and the evolving needs of the agricultural industry. Some smart agriculture IoT devices based on ESP32 are depicted in **Figure 5**.


*(a)* *(b)*

**Figure 5**: *Smart agriculture IoT devices based on the ESP32 chip (a) STEMinds Eduponics, and (b) Wondernica Agromon agriculture wireless sensor. Adopted from Crowd Supply* [11] and Wondernica Sdn Bhd [12].

The STEMinds Eduponics is a smart agriculture and IoT board with an ESP32 module. It has integrated sensors, including light (e.g., BH1750), temperature, humidity, and air pressure sensors (e.g., BME280), as well as interfaces for connecting the external pump, soil moisture sensor, water quality sensor, and other environmental sensors [11]. Numerous agricultural applications are possible with Eduponics, such as smart irrigation, hydroponics, aquaponics, and IoT weather stations. The Eduponics mobile application for iOS and Android can control and monitor it, and the Arduino IDE and MicroPython can be used to program it.

Agromon is built around the ESP32-WROOM-32 [13], a plug-and-play device for smart farming. Based on Sigfox, LoRa, and Wi-Fi, wireless communication is designed for various applications, including low-power sensor networks. Agromon is an IoT smart agricultural sensor

that monitors rainfall, evaporation, soil moisture, and nutrients, allowing farmers to improve crop production while lowering water consumption and crop losses.

## 1.3 Basic Architecture of IoT Solution

An IoT solution comprises hardware and software components that form an intelligent cyber-physical system. IoT solutions differ in their applications. Despite many variations, all IoT solutions follow the same basic architecture and flow [2]. The IoT architecture consists of four main components [14] as shown in **Figure 6**. The "things" of connected devices that collect and process sensor data and then transfer it to the cloud or a local data server for data storage, analysis, and extracting information from sensor data, the data is displayed and visualized to the applications so that the user can understand the information from the device for data-driven decision making.

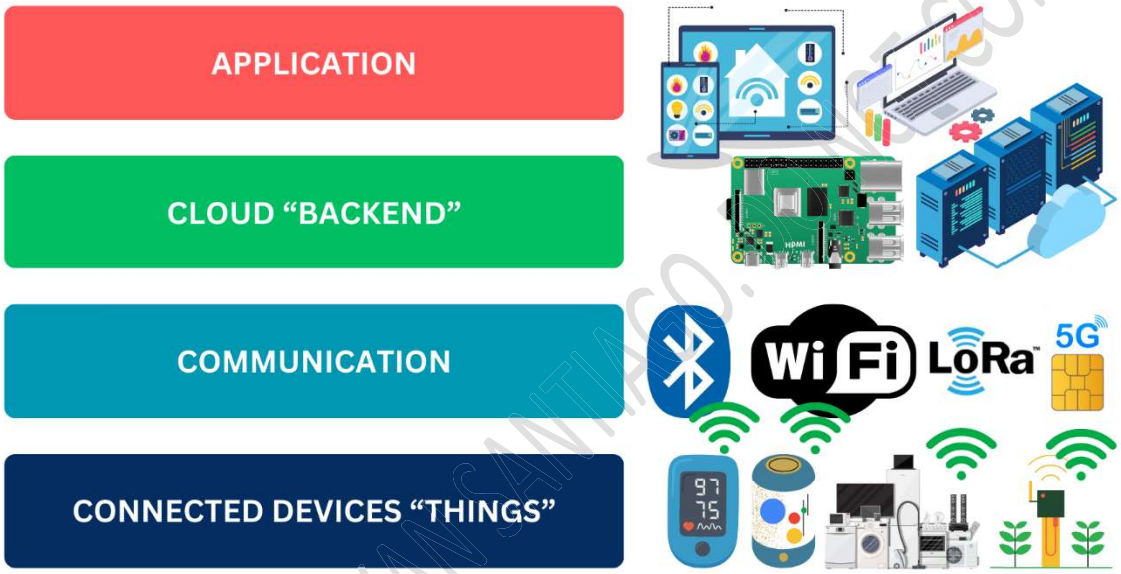



**Figure 6**: *The four layers of IoT architecture*

- **Perception Layer**: Connected devices, "Things" is the heart of any IoT solution is the hardware device with embedded sensors, actuators, and processing units (e.g., microcontroller unit (MCU), system-on-chip (SoC)) that collect and transmit data and interact with the physical world. The intelligence of IoT devices is encapsulated with software flashed in program memory, also called firmware. It bridges the hardware and higher-level functions and determines how the device processes data, communicates with other devices, and performs specific tasks. Firmware is often designed to be lightweight, efficient, and capable of over-the-air updates for continuous improvement.
- **Network Layer**: IoT devices rely on communication protocols to share information with backend systems. This includes wired protocols such as Ethernet or wireless protocols such as Bluetooth, Wi-Fi, LoRa (Long Range), or cellular networks. The choice of communication protocol depends on factors such as range, power consumption, and data transfer requirements.
- **Cloud "Backend" Layer**: The backend system serves as a command center and processes the data flow from connected devices. It includes servers, databases, and application logic responsible for data storage, analysis, and decision-making. At this level, security measures are paramount to ensure the data's confidentiality and integrity. Cloud platforms are often used for scalable and flexible backend infrastructure. It enables central management of devices and data, providing efficient resource utilization and easy updates. On the other hand, local servers (i.e., Raspberry

Pi board or any single-board computer) offer more control over data storage and processing, making them suitable for applications with strict data privacy and security requirements. However, it lacks the scalability and accessibility benefits of cloud platforms. Choosing the backend for a particular IoT solution considers factors such as data sensitivity, the geographical distribution of devices, and the desired level of control and customization.

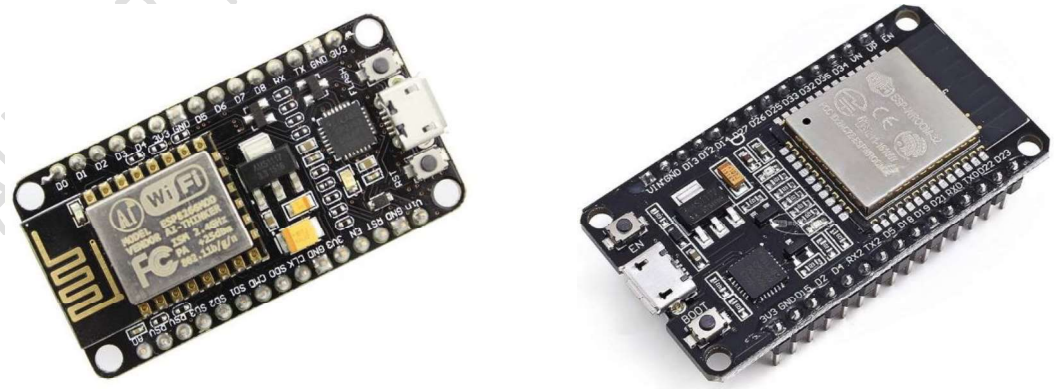
- **Applications Layer**: The end-user application is how individuals interact with and control IoT devices. This can be a mobile application, web dashboard, human-machine interface (HMI), or voice assistance (e.g., Google Home, Alexa Smart Home). These applications allow users to monitor device status, set preferences, receive alerts, and engage in two-way communication within the IoT ecosystem.

## 1.4 ESP32 for IoT Development

Founded in 2008, *Espressif Systems* has grown from a small startup in China to a multinational semiconductor company pioneer in the IoT industry [15]. Innovative wireless communications and low-power Artificial Intelligence of Things (AIoT) solutions were the primary focus of Espressif Systems, which also produced the well-known ESP8266, ESP32, ESP32-S, ESP32-C, and ESP32-H chip, module, and development board series. AIoT is the integration of AI technologies with IoT systems to provide intelligent services and capabilities to IoT devices, enabling real-time decision-making in complex environments [16]. Espressif is dedicated to *democratizing access to AIoT by making its solutions and technology open source*, allowing developers worldwide to utilize this technology to create solutions for global issues.

In 2014, the release of the revolutionary ESP8266 microcontroller, shown in **Figure 7 (a)**, was a significant factor in the growth of IoT devices [17], [18]. With its single-core Tensilica processor with built-in Wi-Fi connectivity, this device is well-known as a cost-effective solution for connecting devices to the Internet.

An improved version of ESP8266 called ESP32, shown in **Figure 7 (b)**, was introduced in 2016. Figure 8 depicts the functional block diagram of the ESP32 chip [17], [19]. This device has a 32-bit Tensilica Xtensa LX6 dual-core SoC processor that delivers increased speed, memory, General-Purpose Input/Output (GPIO) pins, and peripherals. Additionally, it has integrated 2.4 GHz Wi-Fi, Bluetooth Low Energy (BLE) v4.2, and Bluetooth Classic into a single chip. The most recent ESP32 series is built around a 32-bit RISC-V processor. It offers an accelerator for neural network and signal processing code, Wi-Fi connectivity at the 5GHz band, BLE v5.0, and Zigbee/Thread for wireless mesh networks.



*(a)* *(b)*

**Figure 7**: *ESP IoT development board (a) Espressif NodeMCU module V1.0 and (b) DOIT ESP32-DevKit V1. Adopted from Nabto* [18], [19].

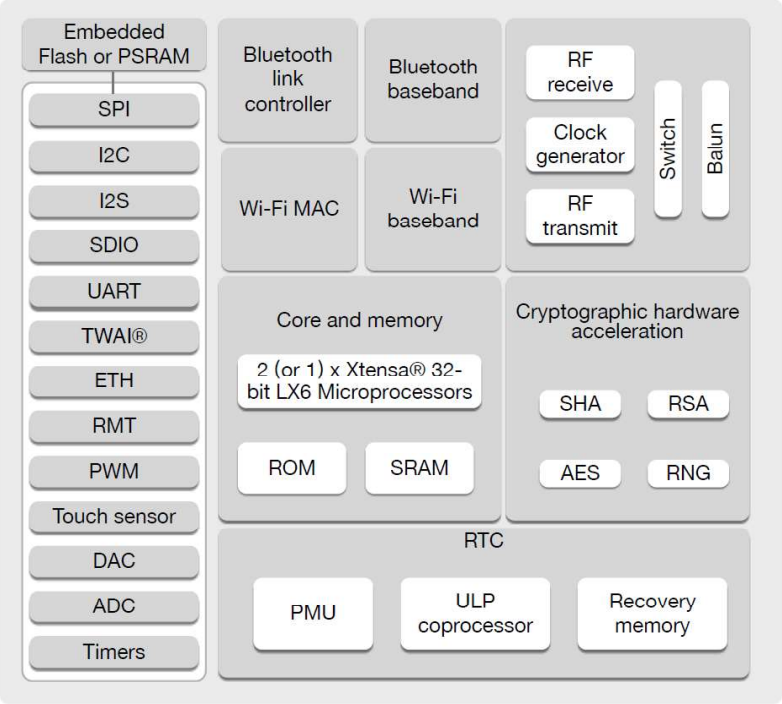



**Figure 8**: *Espressif ESP32 chip functional block diagram. Obtain from the ESP32 series datasheet* [20].

The widely used microcontrollers ESP32 and ESP8266, created by Espressif Systems, are well-known for their versatility and affordability, particularly in IoT development. A summary of specifications for ESP8266-12 and ESP32 is shown in the table below. Despite certain similarities, the ESP32 is a more sophisticated replacement for the ESP8266 due to some key differences.

**TABLE 1**

*ESP8266-12 and ESP32 specifications and comparisons*

| SPECIFICATIONS | ESP8266 | ESP32 |
|---|---|---|
| *Core* | 1 | 2 |
| *Architecture* | 32-bits | 32-bits |
| *Clock* | Xtensa LX106 80-160MHz | Xtensa LX106 160-240MHz |
| *Wi-Fi* | IEEE802.11 b/g/n | IEEE802.11 b/g/n |
| *Bluetooth* | None | Classic and Low Energy v4.2 |
| *RAM* | 160KB | 520KB |
| *Flash memory* | Extern QSPI 512KB - 4MB | Extern QSPI 4MB - 16MB |
| *GPIO pin* | 17 | 34 |
| *DAC pin* | None | 2 x 8-bit resolution |
| *ADC pin* | 1 x 10-bit resolution | 18 x 12-bit resolution |
| *PWM pin* | 8-channels | 16-channels |
| *Peripheral Interface* | SPI-I2C-UART-I2S-IR | SPI-I2C-UART-I2S-CAN |
| *Operating Voltage* | 3.3V | 3.3V |
| *On-chip sensor* | None | Touch, Temp., and Hall Effect |

Although the ESP8266 is an affordable and accessible IoT solution, the ESP32 expands on the success of its predecessor with better performance, more features, and more capabilities. The decision between the two usually depends on the project's specific requirements. In general, applications that require higher processing power, embedded AI computing, more sophisticated wireless connectivity, and greater I/O capabilities should use the ESP32.

## 1.5 Getting Started with ESP32 Using Arduino IDE

Both ESP8266 and ESP32 support the *Real-Time Operating System* (RTOS) based *Software Development Kit* (SDK) using the *Espressif IoT Development Framework* (ESP-IDF). ESP-IDF officially supports *Eclipse* and *Visual Studio Code*. However, ESP-IDF requires in-depth C, C++, and RTOS programming skills. To simplify firmware development, both can be programmed using the Arduino IDE. This book uses the Arduino IDE as the development framework for ESP32. The following components and steps to install Arduino ESP32 support are as follows:

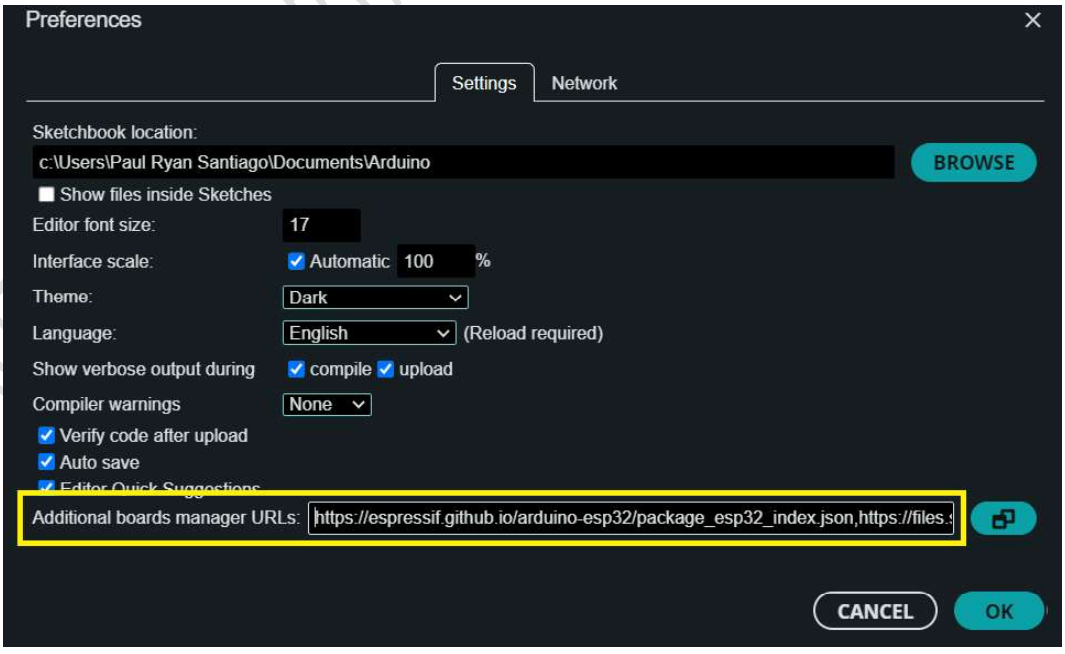
### 1.5.1 Hardware Requirements

- An ESP32 Development Board (i.e., LILYGO TTGO LORA32 OLED V2.6.1)
- Micro USB cable
- Computer running on Windows OS

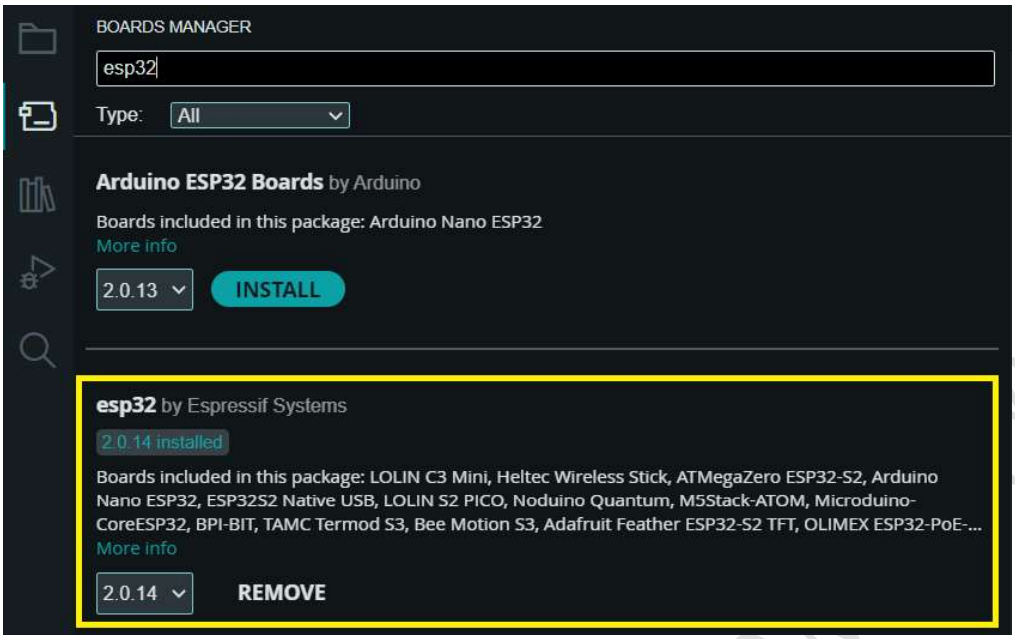
### 1.5.2 Software Requirements

Getting started with the ESP32 using the Arduino IDE is a straightforward process. Here are step-by-step instructions for setting up the environment and starting programming with the ESP32 [21], [22].

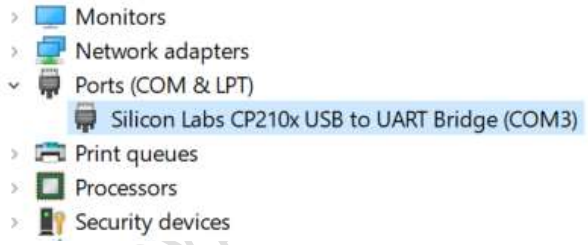
1. **Install Arduino IDE**
   - Download and install the recent Arduino IDE from the official Arduino website https://www.arduino.cc/en/software
2. **Install ESP32 Board Arduino Support**
   - Open the Arduino IDE.
   - Go to "**File**" > "**Preferences**."
   - In the "**Additional Boards Manager URLs**" field, add the stable release link: https://espressif.github.io/arduino-esp32/package_esp32_index.json
     Enter the release links above into the Additional Board Manager URLs field. Once done, you can now install ESP32 Arduino support.




   - Click "**OK**" to close the Preferences window.
   - Go to "**Tools**" > "**Board**" > "**Boards Manager**."
   - Search for "**esp32**" and install the "**esp32 by Espressif Systems**."




3. **Installation of USB Serial Driver for ESP32**
   - Connect the ESP32 board to the PC using the micro-USB cable. Then, type "**Device Manager**" in the taskbar search box and select from the menu. Check the list of identified COM ports in Windows Device Manager. The following figure shows a detected ESP32 board.




   - If the device driver does not install automatically, download and extract the installation file CP210xVCPInstaller_x64, then install the Silicon Labs CP210x USB to UART Bridge VCP Driver for Windows from the link below.
     https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads
     Choose one of the two download options, as shown in the figure below.


     After installation, the ESP32 serial COM port (e.g., Silicon Labs CP210x USB to UART Bridge (COM3)) should be detected and appear in the Device Manager COM ports.

## 1.5.3 Basic ESP32 Board Pinout Guide

Before you start programming the ESP32 board, it is essential to know the configuration of its GPIO pins. ESP32 has 34 GPIO pins, which can be assigned various functions by programming the corresponding registers [20], [23]. GPIOs come in multiple varieties, such as digital-only, analog-enabled, capacitive-touch-enabled, and so on. Digital GPIOs can be configured from both analog and capacitive touch-enabled GPIOs.

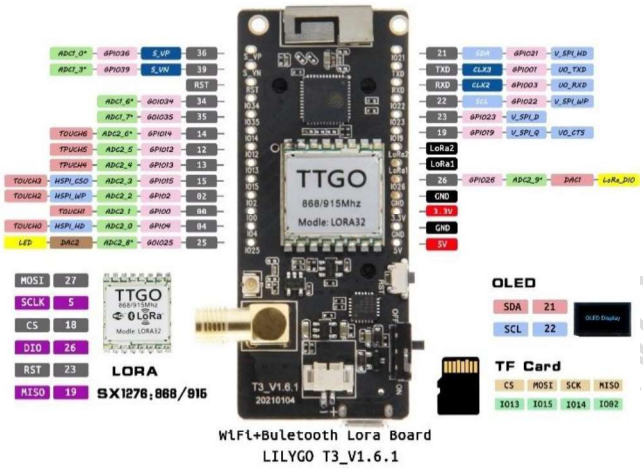



**Figure 9**: *TTGO LILYGO LoRa32 OLED IoT development board pinout. Adopt from the TTGO LoRa32 T3 V2.6.1 Github repository [24].*

When set as a digital input, most digital GPIOs can be configured as internal pull-up or pull-down impedance. The pinout of the LILYGO TTGO LoRa32 OLED ESP32 development board is shown in **Figure 9** for reference, and most digital I/O pins are bidirectional. It is possible to multiplex these pins with other interfaces such as I2C, UART, SPI, and so on. The following summary shows which pins are best used as inputs and outputs, and which ones you must be careful with [25], [26].

- **Digital Input/Output Pins**
  *GPIO pins 34 through 39 are inputs only*. There is no internal pull-up or pull-down circuitry on these pins. Thus, it cannot be used as an output pin.
    - GPIO 34
    - GPIO 35
    - GPIO 36
    - GPIO 39

  Each digital output pin is assigned to configurable current ratings. There are four possible configurations for the current ratings of the digital output pins.
    - 0: ~5 mA
    - 1: ~10 mA
    - 2: ~20 mA (default)
    - 3: ~40 mA

  *It is advised to keep the current consumption per GPIO pin below 20mA*.
- **Analog to Digital Converter (ADC)**
  ESP32 supports 18 ADC channels, each with a 12-bit ADC resolution. These analog pins can be used as ADC and corresponding channels [20], [27]. The ability to read a voltage level found on an analog pin between 0 and a certain maximum value is an analog-to-digital conversion. The value read can change by varying the voltage applied to the analog pin. With a resolution of up to 12 bits, 4096 different values can be read. That means that 0V will create a digital value of 0, while 3.3V will produce a

digital value of 4095, and the voltage ranges between these will generate a correspondingly scaled digital value.

- ADC1_CH0 (GPIO 36)
- ADC1_CH1 (GPIO 37)
- ADC1_CH2 (GPIO 38)
- ADC1_CH3 (GPIO 39)
- ADC1_CH4 (GPIO 32)
- ADC1_CH5 (GPIO 33)
- ADC1_CH6 (GPIO 34)
- ADC1_CH7 (GPIO 35)
- ADC2_CH0 (GPIO 4)
- ADC2_CH1 (GPIO 0)
- ADC2_CH2 (GPIO 2)
- ADC2_CH3 (GPIO 15)
- ADC2_CH4 (GPIO 13)
- ADC2_CH5 (GPIO 12)
- ADC2_CH6 (GPIO 14)
- ADC2_CH7 (GPIO 27)
- ADC2_CH8 (GPIO 25)
- ADC2_CH9 (GPIO 26)

*Channel ADC2 pins cannot be used when Wi-Fi is in use. So, if you're using Wi-Fi and having trouble getting the value from an ADC2 GPIO, use channel ADC1, and the problem should be solved*.

- **Digital to Analog Converter (DAC)**
  Two 8-bit DAC channels can convert the digital value 0~255 to the analog voltage 0~Vref (3.3V). These two DAC channels support a power supply as an input voltage reference [28]. The DAC channels are designated to the listed GPIO pins.
  - GPIO25 (DAC channel 1)
  - GPIO26 (DAC channel 2)
- **Integrated SPI Flash**
  Some ESP32 development boards have exposed GPIOs 6 through 11. Using these pins for other purposes is not advised, as they are linked to the ESP-WROOM-32 chip's integrated SPI flash. Hence, *avoid using these pins*.
  - GPIO 6 (SCK/CLK)
  - GPIO 7 (SDO/SD0)
  - GPIO 8 (SDI/SD1)
  - GPIO 9 (SHD/SD2)
  - GPIO 10 (SWP/SD3)
  - GPIO 11 (CSC/CMD)
- **Strapping Pins**
  Some GPIOs are called *strapping pins*, which change the pin state at boot or reset to HIGH or output PWM signals [25]. This means unexpected results may occur when resetting or booting the ESP32 if output devices are connected to these GPIOs.
  - GPIO 1
  - GPIO 3
  - GPIO 5
  - GPIO 6 to GPIO 11 (connected to the ESP32 integrated SPI flash)
  - GPIO 14
  - GPIO 15

  *You may have difficulty uploading new code or flashing the ESP32 with new firmware if you have peripherals connected to the strapping pins. This could be because these peripherals prevent the ESP32 from accessing the correct mode*.

- **Inter-IC Communication pins (I2C)**
  ESP32 has two I2C bus interfaces, which can serve as I2C master or slave, depending on the user's configuration. When using the ESP32 with the Arduino IDE, it should use the ESP32 I2C default pins.
    - GPIO21 (SDA)
    - GPIO22 (SCL)

  Typically, an I2C slave device has a 7-bit or 10-bit addressing mode. ESP32 supports I2C Standard and Fast modes, which can go up to 100KHz and 400KHz, respectively. I2C uses two bidirectional lines pulled up by resistors. The frequency of SCL is influenced by both the pull-up resistor and the wire capacitance. Therefore, users are strongly recommended to choose appropriate pull-up resistors to make the frequency accurate. The recommended value for pull-up resistors usually ranges from 1K Ohms to 10K Ohms. Keep in mind that the higher the frequency, the smaller the pull-up resistor is required, but not less than 1K Ohms. Indeed, large resistors will decrease the current, which will increase the clock switching time and reduce the frequency [29]. It recommends a range of 2K Ohms to 5K Ohms.
  *Two I2C buses can be used simultaneously. However, I2C pin interfaces should be defined during initialization and provide more user flexibility.*
  *In many breakout boards, the SDA line may also be labeled as SDI and the SCL line as SCK.*
- **Serial Peripheral Interface GPIO pins (SPI)**
  The SPI Master Driver is a program that controls the ESP32's SPI peripherals while they act as masters. ESP32 integrates four SPI peripherals. SPI0 and SPI1 are used internally to access the ESP32's connected flash memory. SPI2 and SPI3 are general-purpose SPI controllers, sometimes called HSPI and VSPI, respectively [30]. They are open to users. SPI2 and SPI3 have independent signal buses, each with the same name. By default, the pin mapping for SPI is:

| SPI | MOSI | MISO | CLK | CS |
|---|---|---|---|---|
| *VSPI* | GPIO 23 | GPIO 19 | GPIO 18 | GPIO 5 |
| *HSPI* | GPIO 13 | GPIO 12 | GPIO 14 | GPIO 15 |

- **Universal Asynchronous Receiver Transmitter (UART)**
  ESP32 has three UART interfaces, i.e., UART0, UART1, and UART2, that work at 3.3V TTL level, which provide asynchronous communication (e.g., RS485) at a speed of up to 5 Mbps. Each UART has a set of registers to simplify programming and provide greater flexibility. UART provides a widely used and cost-effective method to realize full-duplex or half-duplex data exchange between devices [25], [31].
  The UART interface is designated as RX and TX pins. The following table indicates the RX and TX pins for each of the three UART ports available in ESP32.

| UART PORT | RX | TX | USABILITY |
|---|---|---|---|
| *UART 0* | GPIO3 | GPIO1 | No, it is reserved for programming the ESP32 |
| *UART 1* | GPIO9 | GPIO10 | Yes, but it requires the reassignment of pins |
| *UART 2* | GPIO16 | GPIO17 | Yes |

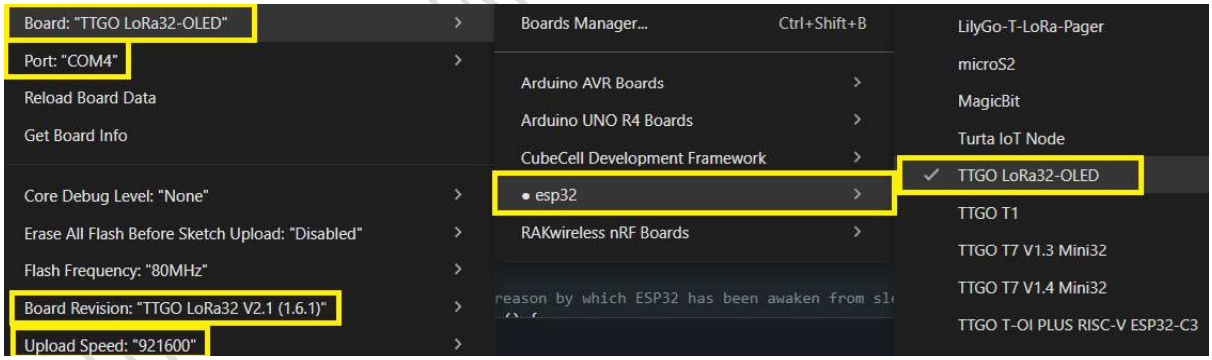
  The UART pins consist of UART0 with GPIO pins D3 and D1 as RX0 and TX0, and UART2 with pins D16 and D17 as RX2 and TX2, respectively. While UART1 does not have an accessible GPIO assigned, it can be accessed by assigning an accessible GPIO to the TX and RX pins.
  *It is recommended to keep the UART0 for programming to access the USB port on the ESP32 development board. It is suitable for printing debugging information on the IDE's serial monitor.*

- **Capacitive Touch GPIO pins**
  ESP32 has ten capacitive-sensing GPIOs, which detect variations induced by touching or approaching the GPIOs with a finger or other objects [25]. The ten capacitive-sensing GPIOs are listed below.
  - T0 (GPIO 4)
  - T1 (GPIO 0)
  - T2 (GPIO 2)
  - T3 (GPIO 15)
  - T4 (GPIO 13)
  - T5 (GPIO 12)
  - T6 (GPIO 14)
  - T7 (GPIO 27)
  - T8 (GPIO 33)
  - T9 (GPIO 32)

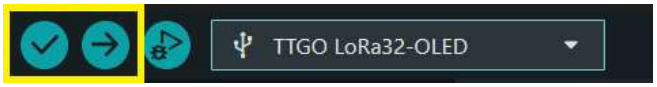
## 1.5.4 Set Up and Programmed ESP32 Using the Arduino IDE

After installing the necessary software for the ESP32, you may use the Arduino IDE to upload your first sketch. Programming the ESP32 with the Arduino IDE is a common choice due to its simplicity and compatibility with the Arduino ecosystem. This step-by-step guide sets up the ESP32 on the Arduino IDE.

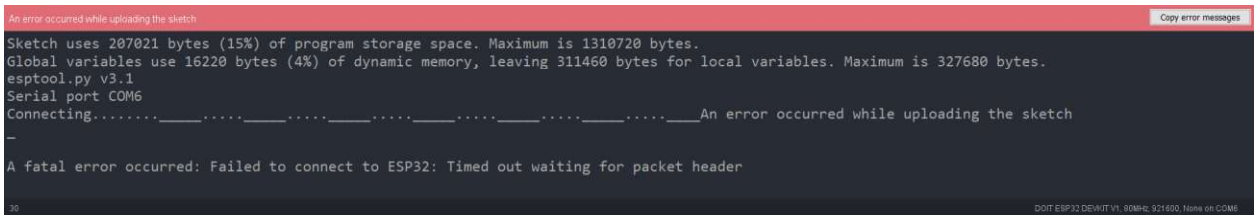
1. **Configuring the ESP32 Board on Arduino IDE**
   - Go to "**Tools**" > "**Board**" > "**esp32**" and select "**TTGO LoRa32-OLED**" as your target board. Then go to "**Board Revision**", there are three choices, select "**TTGO LoRa32 V2.1 (1.6.1)**" as its board version.
   - In the same "**Tools**" menu, set the "**Upload Speed**" to "**921600**" baud.
   - Go to "**Tools**" > "**Port**" and select the COM port that corresponds to your connected ESP32.




2. **Verify and Upload Your First Sketch**
   - Go to "**File**" > "**Examples**" > "**01.Basics**" and select "**Blink**."
   - Click the top left arrow button or go to "**Sketch**" > "**Upload**" (Ctrl+U) to compile (✓) and upload (→) the sketch to your ESP32.




After the upload is complete, the Arduino sketch alternately turns the ESP32 built-in LED on and off every second. Under certain circumstances, an error occurs while uploading the sketch, such as being unable to connect to the ESP32 board to upload a new sketch, as shown.




In this case, you need to *hold down the BOOT button* of the ESP32 board while uploading the sketch and then release it once the connection is established. You should see "Upload Completed," confirming the sketch was successfully flashed. You may repeat this process whenever you want to upload a new sketch. However, if you make your ESP32 board automatically flash a sketch, you need to connect a 10uF electrolytic capacitor between the EN pin and the ground [32].

## 1.6 Summary

This chapter began by acquainting you with the various "Things" that comprise IoT, ranging from home devices and wearables to industrial automation systems, and explored their real-world use cases. Then it examined how industrial IoT differs from consumer IoT in terms of architecture, scalability, and reliability, followed by a breakdown of a typical IoT solution stack (i.e., devices, connectivity, cloud, and applications). A core part of the chapter introduced the ESP32 development board, its role in IoT, essential features, and rich wireless capabilities. This chapter guided you through setting up the ESP32 development environment, such as installing the Arduino IDE setup, and providing a detailed pin-out guide covering GPIO, ADC, PWM, I2C, SPI, and UART. Finally, you explored a basic LED blink program Arduino example, uploaded it to confirm correct setup, and diagnosed common issues such as port configuration and boot-mode errors. This foundation empowers you to confidently progress into sensor integration, wireless networks, and cloud-based applications in later chapters.

## References

[1] D. R. Kiran, "Production Planning and Control A Comprehensive Approach," in *Production Planning and Control*, Butterworth-Heinemann, 2019, pp. 495–513. doi: 10.1016/B978-0-12-818364-9.00035-4.

[2] V. O. Oner, *Developing IoT Projects with ESP32 Automate Your Home or Business with Inexpensive Wi-Fi Devices*. 2021.

[3] M. Ayoub Khan, "Internet of Things; A Hardware Development Perspective," 2022. [Online]. Available: https://www.routledge.

[4] Shelly, "Easy Smart Home Automation." Accessed: Nov. 10, 2023. [Online]. Available: https://www.shelly.com/en

[5] ESPHome Devices, "Shelly Plus 1PM." Accessed: Nov. 11, 2023. [Online]. Available: https://devices.esphome.io/devices/Shelly-Plus-1PM

[6] Sonoff Official, "Sonoff POW Elite." Accessed: Nov. 11, 2023. [Online]. Available: https://sonoff.tech/product/diy-smart-switches/pow-elite/

[7] Smarty Ninja, "Sonoff POW Elite – power meter with relay, WiFi. Power up to 20A." Accessed: Nov. 11, 2023. [Online]. Available: https://www.smarty.ninja/en/wifi/sonoff-pow-elite/

[8] Keith Micallef, "Facebook | Arduino Academy." Accessed: Nov. 11, 2023. [Online]. Available: https://www.facebook.com/groups/ArduinoAcademy/permalink/6604371359626242/?mibextid=zDhOQc

[9] "Combination Smoke and Carbon Monoxide Alarm | X-Sense." Accessed: Nov. 11, 2023. [Online]. Available: https://www.x-sense.com/collections/combination-smoke-and-carbon-monoxide-alarms

[10] Mark B., "X-Sense XS03-WX WiFi Smoke Detector Review – MBReviews," M Reviews. Accessed: Nov. 11, 2023. [Online]. Available: https://www.mbreviews.com/x-sense-xs03-wx-wifi-smoke-detector-review/
[11] "Eduponics Mini v2.0." Accessed: Nov. 11, 2023. [Online]. Available: https://www.crowdsupply.com/steminds/eduponics-mini-v2
[12] Wondernica Sdn Bhd, "Wondernica Agromon Agriculture | Facebook." Accessed: Nov. 11, 2023. [Online]. Available: https://www.facebook.com/WondernicaResearch/posts/wondernica-agromon-agriculture-0g-sigfox-transmitter-installation-at-roselle-far/799060873940335/
[13] "Agromon: Smart Agriculture Using ESP32-WROOM-32U | Espressif Systems." Accessed: Nov. 11, 2023. [Online]. Available: https://www.espressif.com/en/news/Agromon
[14] R. Heredia, "4 Layers of IoT Architecture Explained," Zipit Wireless, Inc. Accessed: Nov. 12, 2023. [Online]. Available: https://www.zipitwireless.com/blog/4-layers-of-iot-architecture-explained
[15] Espressif Systems, "About Espressif," Espressif Systems. Accessed: Nov. 12, 2023. [Online]. Available: https://www.espressif.com/en/company/about-espressif
[16] Science Direct, "Artificial Intelligence of Things - An Overview | ScienceDirect Topics," Elsevier. Accessed: Jul. 30, 2025. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/artificial-intelligence-of-things
[17] S. Cording, "What Is the ESP32? Its Brief History and How to Get Started," Elektor Magazine. Accessed: Nov. 12, 2023. [Online]. Available: https://www.elektormagazine.com/articles/what-is-the-esp32
[18] I. Hubschmann, "ESP8266 for IoT: A Complete Guide," Nabto. Accessed: Nov. 12, 2023. [Online]. Available: https://www.nabto.com/esp8266-for-iot-complete-guide/
[19] I. Hubschmann, "ESP32 for IoT: A Complete Guide," Nabto. Accessed: Nov. 12, 2023. [Online]. Available: https://www.nabto.com/guide-to-iot-esp-32/
[20] Espressif Systems, "ESP32 Series Datasheet," 2022. [Online]. Available: https://www.espressif.com/en/support/download/documents.
[21] Espressif Systems, "Installing — Arduino-ESP32 2.0.14 documentation," Arduino-ESP32 Espressif. Accessed: Nov. 13, 2023. [Online]. Available: https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/installing.html
[22] Espressif Systems, "Establish Serial Connection with ESP32," ESP-IDF Programming Guide. Accessed: Nov. 13, 2023. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/establish-serial-connection.html
[23] Espressif Systems, "GPIO & RTC GPIO - ESP32," ESP-IDF Programming Guide. Accessed: Nov. 26, 2023. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html#gpio-rtc-gpio
[24] "GitHub - LilyGO/TTGO-LoRa32-V2.1: T3." Accessed: Jul. 30, 2025. [Online]. Available: https://github.com/LilyGO/TTGO-LoRa32-V2.1
[25] Espressif, "Schematic Checklist - ESP32 - — ESP Hardware Design Guidelines latest documentation," Espressif Systems (Shanghai) Co., Ltd. Accessed: Jul. 31, 2025. [Online]. Available: https://docs.espressif.com/projects/esp-hardware-design-guidelines/en/latest/esp32/schematic-checklist.html?highlight=strapping%20pin#strapping-pins
[26] Espressif, "GPIO: Arduino ESP32 Documentation," Espressif Systems (Shanghai) Co., Ltd. Accessed: Jul. 26, 2025. [Online]. Available: https://docs.espressif.com/projects/arduino-esp32/en/latest/api/gpio.html
[27] Espressif Systems, "Analog to Digital Converter - ESP32," ESP-IDF Programming Guide. Accessed: Nov. 26, 2023. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/v4.2/esp32/api-reference/peripherals/adc.html#
[28] Espressif, "Digital To Analog Converter (DAC) - ESP32 - — ESP-IDF Programming Guide v5.5 documentation," Espressif Systems (Shanghai) Co., Ltd. Accessed: Jul. 31, 2025. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/dac.html

[29] Espressif, "Inter-Integrated Circuit (I2C) - ESP32 - — ESP-IDF Programming Guide v5.5 documentation," Espressif Systems (Shanghai) Co., Ltd. Accessed: Jul. 31, 2025. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/i2c.html
[30] Espressif, "SPI Master Driver - ESP32 - — ESP-IDF Programming Guide v5.5 documentation," Espressif Systems (Shanghai) Co., Ltd. Accessed: Jul. 31, 2025. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/spi_master.html
[31] Espressif, "Universal Asynchronous Receiver/Transmitter (UART) - ESP32 - — ESP-IDF Programming Guide v5.5 documentation," Espressif Systems (Shanghai) Co., Ltd. Accessed: Jul. 31, 2025. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/uart.html
[32] Last Minute Engineers, "How to Solve 'A Fatal Error Occurred: Failed to Connect to ESP32: Timed Out Waiting for Packet Header,'" LastMinuteEngineers.com. Accessed: Jul. 31, 2025. [Online]. Available: https://lastminuteengineers.com/esp32-fatal-error-fix-tutorial/