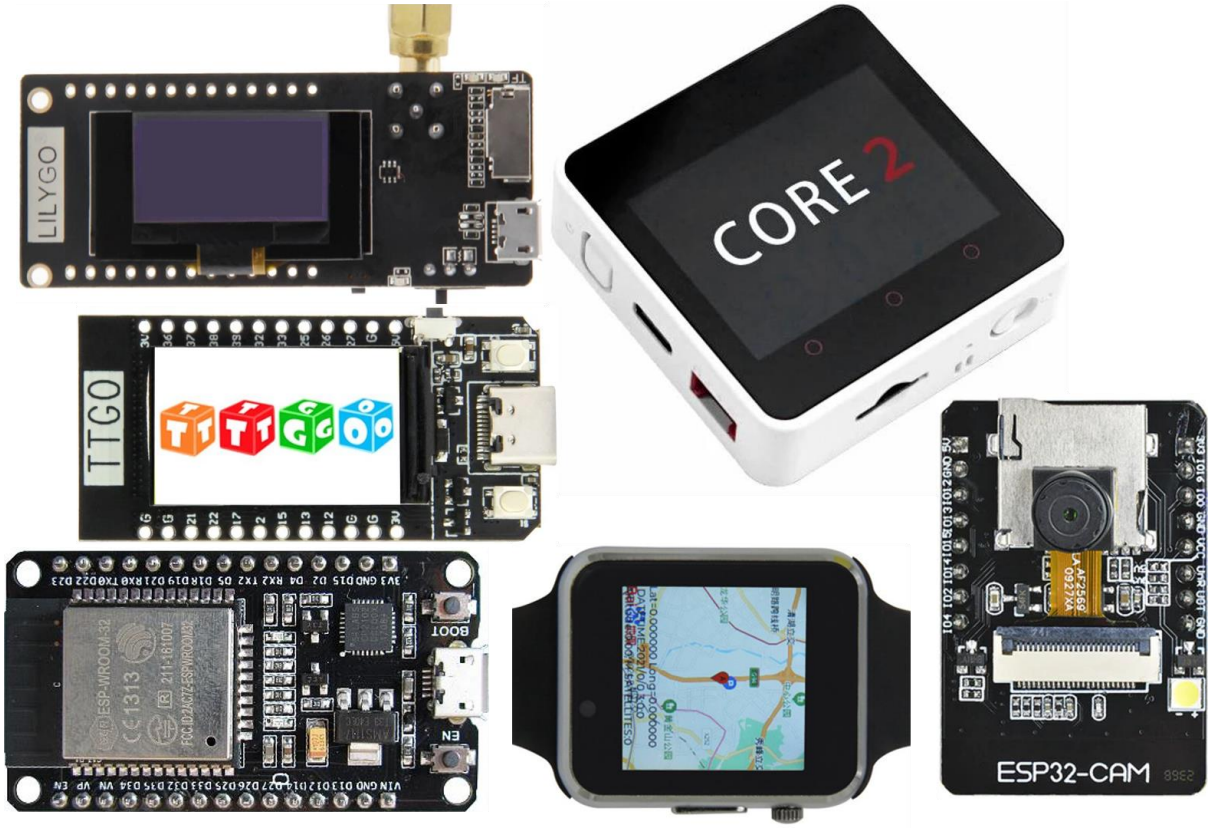
PREFACE

The ESP32 microcontroller is incorporated in several formats ranging from a development board to a camera based module to an integrated watch with touchscreen and GPS. The variety of different ESP32 formats illustrate the diversity of projects centred on the ESP32 microcontroller. *ESP32 Formats and Communication* develops projects with the ESP32 DEVKIT DOIT, the TTGO T-Display V1.1, the TTGO LoRa32 V2.1 1.6, the ESP32-CAM, the TTGO T-Watch V2 with GPS and the M5Stack Core2 modules.



Each ESP32 module format has different features making some formats better suited for a particular project. The TTGO T-Display V1.1, the TTGO LoRa32 V2.1 1.6 and M5Stack Core2 have built-in display screens. The TTGO LoRa32 V2.1 1.6 incorporates a LoRa (Long Range) module for transmitting and receiving messages. In Chapter 6 *LoRa and microsatellites*, satellites circling 550km above the Earth are tracked with the TTGO LoRa32 V2.1 1.6 module. The ESP32-CAM module is built around a 2M-pixel OV2640 camera. Chapter 11 *ESP-CAM camera* describes streaming images with the WebSocket protocol to a remote ESP32 microcontroller connected to an LCD screen or with Wi-Fi communication to an app. Several projects are developed with the TTGO T-Watch V2 in Chapter 4 *TTGO T-Watch V2* with Bluetooth communication, GPS position detection and route tracking, infrared

signalling and accessing information over the internet. The M5Stack Core2 module incorporates a touch LCD screen, Bluetooth and Wi-Fi communication, a microphone and speaker, as well as an accelerometer and gyroscope, making the M5Stack Core2 module extremely versatile. The M5Stack Core2 module features in several Chapters.

The focus of the book is ESP32 microcontroller projects with different communication protocols. Wi-Fi communication to access information over the internet or to control remote devices through an app is utilised in Chapters 12 *Control apps*, Chapter 13 *Remote control motors* and Chapter 14 *Remote control ESP32-CAM robot car*. The WebSocket protocol allows two-way real-time communication between an ESP32 microcontroller and a webpage as illustrated in Chapter 8 *Websocket, remote access and OTA*. Bluetooth Low Energy (BLE) communication is outlined in Chapter 5 *BLE beacons* with the ESP32 microcontroller acting as a BLE beacon to transmit information to mobile phones, Android tablets and other devices close to the beacon. Email communication with an ESP32 microcontroller and generation of QR (Quick Response) codes to instruct an ESP32 microcontroller to control a connected device are described in Chapter 7 *email and QR codes*. Transmission and storage of image data and the fast display of images on an app is described in Chapter 10 *Managing images*. The MESH communication protocol, outlined in Chapter 3, enables communication between ESP32 microcontrollers without a Wi-Fi connection. The ESP-NOW protocol also enables communication between ESP32 devices without a Wi-Fi connection. In Chapter 9 *MQTT*, a remote ESP32 microcontroller communicates with ESP-NOW to a second ESP32 microcontroller, which has a Wi-Fi connection to transmit smart meter data to an MQTT broker with the information displayed over the internet and accessible from anywhere in the world. The ESP32 microcontroller-based internet radio, MP3 player and Bluetooth speaker projects in Chapter 2 *I2S audio*, all use the I2S (Inter-Integrated circuit Sound) communication protocol for transmission of encoded audio data. The ESP32 microcontroller transmits audio data to a Bluetooth speaker or receives audio data to output audio signals using an audio decoder.

Each Chapter of the book focuses on a communication protocol and is stand-alone, so you can read any Chapter without having to start from the beginning of the book. The order of Chapters listed in the previous paragraph demonstrates the arbitrary nature of ordering communication protocols. The last three Chapters develop the required apps and IoT techniques for remote control of an ESP32-CAM robot vehicle with an app displaying camera images and the Chapters benefit from being read sequentially.

Schematic diagrams were produced with *Fritzing* software (www.fritzing.org), with the emphasis on maximising the clarity of the layout and minimising overlapping connections between devices. All apps were built with *MIT App Inventor* (appinventor.mit.edu). Each app build is documented and a level of experience with *MIT App Inventor* is assumed. The previous book *Electronics Projects with the ESP8266 and ESP32* includes several chapters on building apps, which incorporate remote-device control, GPS data and *Google Maps*.

All sketches were built in the Arduino IDE (Integrated Development Environment). Some programming expertise with the Arduino IDE is assumed, but all sketches are completely described and comprehensively commented. Authors of libraries used in the book are acknowledged in each Chapter with library details included in Chapter 15 *Libraries*, which also demonstrates building a library.

All the Arduino IDE sketches and *MIT App Inventor* source code is available to download from *GitHub* (github.com/Apress/ESP32Communication). The Arduino programming environment and libraries are constantly being updated, so information on the consequences of the updates is also available on the *GitHub* website.