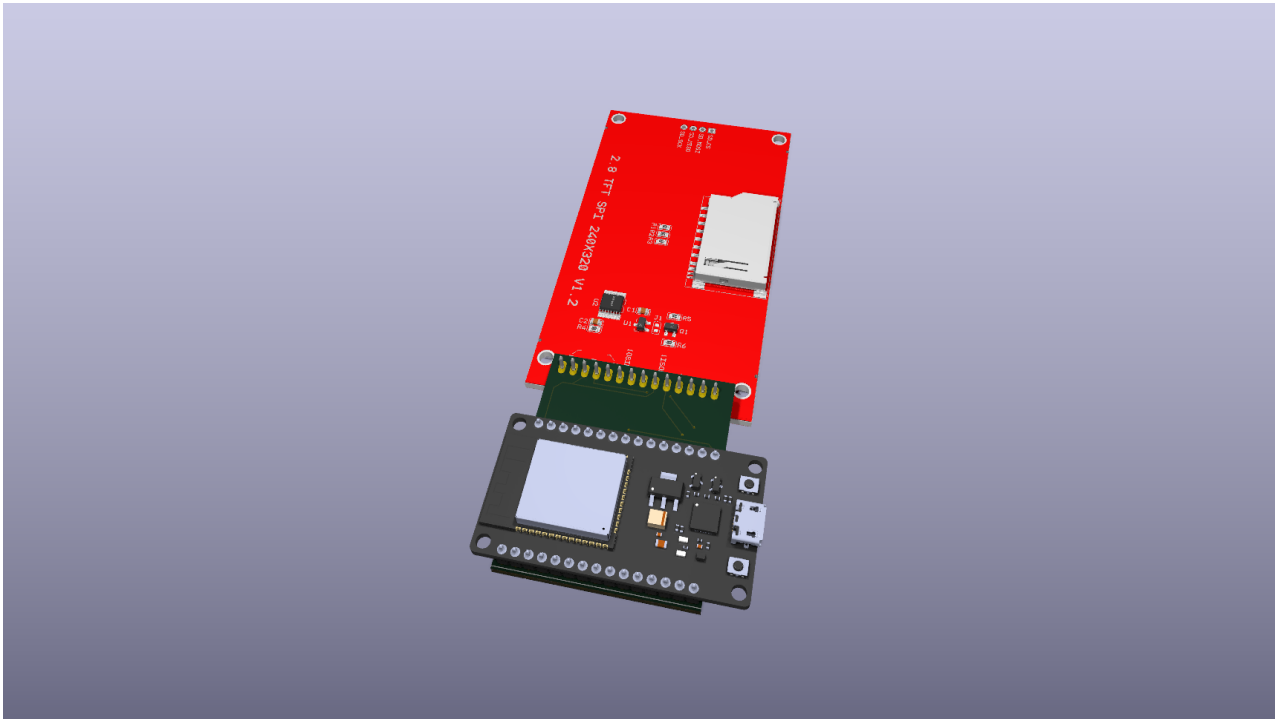
OPEN-SOURCE PROJECT DOCUMENTATION

ESP Morse Station

Firmware v4.0

A self-contained ESP32 Morse terminal with a touch interface, dedicated Morse key, local website, ESP-NOW messaging, learning tools, persistent statistics, RGB feedback, and six games.



Project	ESP Morse Station
Creator	Chitraksh
Platform	ESP32 Dev Module + 2.8-inch ILI9341 touch display
Source format	One Arduino sketch: touch_firmware.ino
Document purpose	Public build, operation, and software reference

1. Project overview

The ESP Morse Station turns a low-cost ESP32 and SPI touch display into a compact communication and learning console. The device can translate keyed Morse, play text as Morse, exchange short messages between stations, host an offline browser interface, preserve statistics and game scores, and provide visual or audio feedback.

Feature summary

Category	Included in firmware v4.0
Morse tools	Learn, Practice, Translate, Text to Morse, Web Morse, Morse to Web
Messaging	ESP-NOW Send and Inbox with selectable station identity
Website	Offline Wi-Fi access point with two-way text and Morse exchange
Interface	Touch-first app menu, optional Back key, portrait and landscape modes
Personalization	Six themes, brightness, WPM, sound, LED, orientation, game levels
Games	Flappy, 2048, Simon, Space Fighter, Donkey Kong, and Pac-Man
Saved data	Settings, touch calibration, learning statistics, and game high scores

How the complete system works

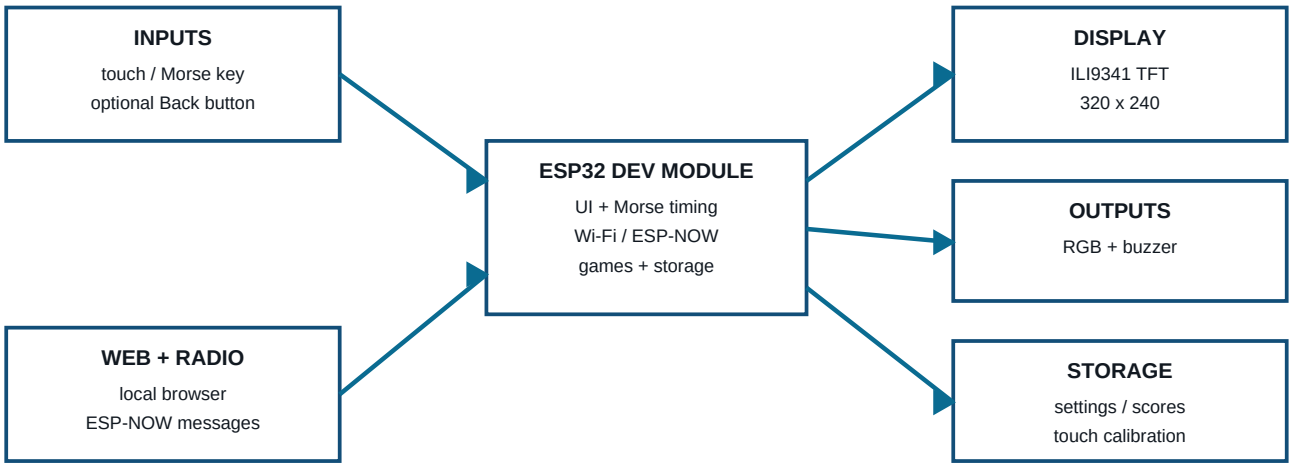


Figure 1. The complete v4.0 system. The website and ESP-NOW radio are built into the ESP32; no internet connection is needed.

Contents

Hardware and wiring / firmware installation / interface and settings / Morse and communications / games and statistics / software architecture / PCB and troubleshooting / open-source publication notes.

2. Hardware and actual wiring

ESP32 pin	Connection	Purpose
5V / VIN	TFT VCC and RGB module +	5 V supply rail used by the working prototype
GND	TFT GND, key return, buzzer return	Common ground
GPIO2	TFT DC / RS	Display command/data selection
GPIO4	TFT RST	Display reset
GPIO5	TFT CS	Display chip select
GPIO13	Normally-open Morse key to GND	Dedicated Morse input
GPIO14	RGB module R	Red PWM channel
GPIO15	Touch T_CS	XPT2046 chip select
GPIO18	TFT SCK + touch T_CLK	Shared SPI clock
GPIO19	TFT SDO + touch T_DO	Shared SPI MISO
GPIO23	TFT SDI + touch T_DIN	Shared SPI MOSI
GPIO25	Optional Back button to GND	Physical Back action
GPIO26	RGB module B	Blue PWM channel
GPIO32	RGB module G	Green PWM channel
GPIO33	Buzzer signal / positive lead	Morse and interface tones
Not connected	Touch IRQ	Firmware uses polling

Shared SPI bus

The display and touch controller use the same GPIO18, GPIO19, and GPIO23 bus. The firmware calls **SPI.begin(18,19,23)** once, gives the initialized bus to the touch library, and keeps separate chip-select lines for TFT and touch.

RGB module used by this project

The current prototype uses the tested **5 V common-anode RGB module with its own onboard resistors**. Its + connection goes to 5 V and its R, G, and B connections go to GPIO14, GPIO32, and GPIO26. This wiring description is for that resistor-equipped module, not for a bare four-lead RGB LED.

Control change in v4.0

GPIO14, GPIO32, and GPIO26 previously served as Up, Down, and Select. They now operate the RGB module, so physical navigation is disabled in the current build. Every app is designed for touch operation; GPIO13 remains the Morse control and GPIO25 can provide Back.

3. Firmware installation

What is required

Item	Project configuration
Arduino board	ESP32 Dev Module
ESP32 board package	Arduino-ESP32; the working build uses 3.3.10
Display library	TFT_eSPI by Bodmer
Touch library	XPT2046_Touchscreen by Paul Stoffregen
Included with ESP32 core	SPI, WiFi, FS, WebServer, Preferences, esp_now
Firmware file	touch_firmware.ino - the complete project is kept in one sketch

TFT_eSPI User_Setup

Setting	Value
Driver	ILI9341_DRIVER
TFT_MISO / TFT_MOSI / TFT_SCLK	19 / 23 / 18
TFT_CS / TFT_DC / TFT_RST	5 / 2 / 4
SPI_FREQUENCY	27000000 in the current working setup

Upload procedure

- Install the ESP32 board package and the two external libraries listed above.
- Apply the TFT_eSPI settings from this page.
- Open touch_firmware.ino in Arduino IDE as the only project source file.
- Select ESP32 Dev Module and the board's serial port, then upload.
- After restart, complete the four-point touch calibration when it appears.

Important include order

The sketch includes **FS.h**, declares **using fs::FS;**, and then includes **WebServer.h**. Keep this order because it is the compatibility fix used by the working ESP32 3.3.10 build.

First successful boot

A successful boot initializes the display, loads saved settings, starts the MorseStation Wi-Fi access point, starts ESP-NOW and the web server, then opens calibration or the main menu.

4. Interface, settings, and touch

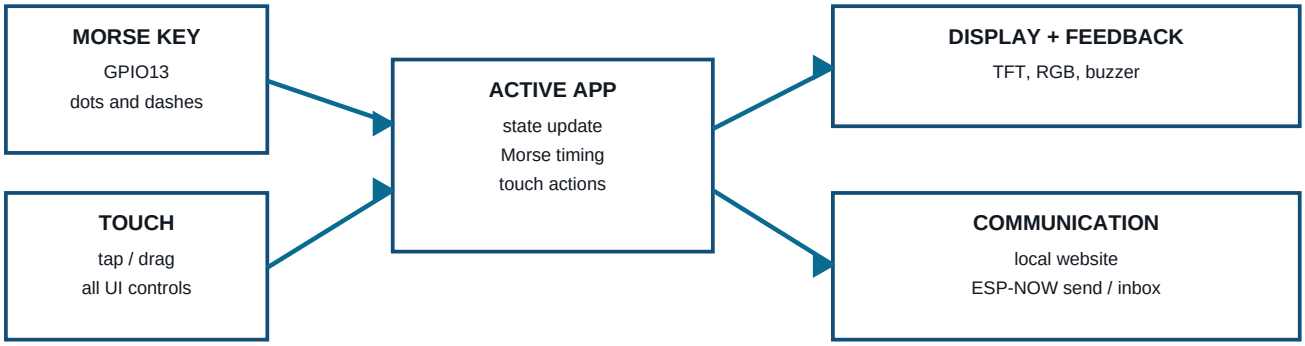


Figure 2. Touch controls the operating system while the dedicated key remains available to Morse-aware apps and games.

Home screen behavior

- Tap an app once to select and highlight it.
- Tap the selected app again to open it.
- Use on-screen Back controls to leave an app; the optional GPIO25 button provides the same action.
- The menu scrolls so every application remains reachable without navigation buttons.

Settings

Setting	Available behavior
Theme	TERM, AMBER, CYBER, AQUA, CANDY, XP
Brightness	Display brightness level used by the interface
Orientation	Portrait or landscape with an adaptive logical canvas
WPM	Morse timing used for dot, dash, letter, and word detection
Sound / LED	Enable or disable buzzer and RGB feedback
Station ID	Alpha, Bravo, Charlie, Delta, Echo, Station 1, Station 2
Game levels	Separate difficulty selection for each game
Reset actions	Reset statistics or restore factory settings

Touch calibration

Calibration is stored independently for portrait and landscape. The firmware samples and filters the XPT2046 in polling mode, rejects low-pressure readings, and maps the raw coordinates to the active logical screen.

5. Morse tools and communication

Morse applications

App	Purpose
Learn	Browse characters and their dot-dash patterns
Practice	Answer generated exercises at easy, medium, or hard level
Translate	Decode live key presses into text
Text to Morse	Enter text with the on-screen QWERTY keyboard and play its Morse
Web Morse	Display plain Morse received from the website and its translation
Morse to Web	Key a message on the device, review the translation, and publish both to the site

Local website

Figure 3. The website is hosted entirely by the ESP32 and can be displayed on a phone, laptop, or classroom smart board.

Network	Value
Wi-Fi name	MorseStation
Password	morse1234
Address	http://192.168.4.1
Internet required	No

ESP-NOW

Send broadcasts a short message together with the selected station ID. Other compatible stations place the packet in Inbox and display a notification. This operates independently of the browser page.

6. Games, statistics, and RGB feedback

Included games

Game	Controls and score
Flappy	Morse key makes the bird flap; score increases through gaps
2048	Touch swipes move and combine tiles; highest tile/score is retained
Simon	Repeat the displayed touch sequence; completed rounds form the score
Space Fighter	Touch left/right zones move; Morse key shoots
Donkey Kong	Touch movement and action controls navigate platforms and barrels
Pac-Man	Touch direction controls navigate the maze and collect points

Persistent statistics

The Statistics app stores characters typed, accumulated accuracy, practice time, fastest WPM, longest session, and the high score for every included game. ESP32 Preferences/NVS preserves these values after restart.

Game difficulty

Settings contains a Game Levels screen with an independent difficulty value for Flappy, 2048, Simon, Space Fighter, Donkey Kong, and Pac-Man.

RGB meanings

Context	Behavior
System status	Firmware selects colors for ready, sending, receiving, and error states
RGB Play	Touch color wheel and R/G/B controls allow temporary color mixing
Leaving RGB Play	Normal system-status behavior resumes
LED disabled	Settings suppresses RGB output

Buzzer

GPIO33 provides Morse side tone and interface/game feedback when Sound is enabled. The implementation uses non-blocking key timing so the display and networking continue to run.

7. Software architecture

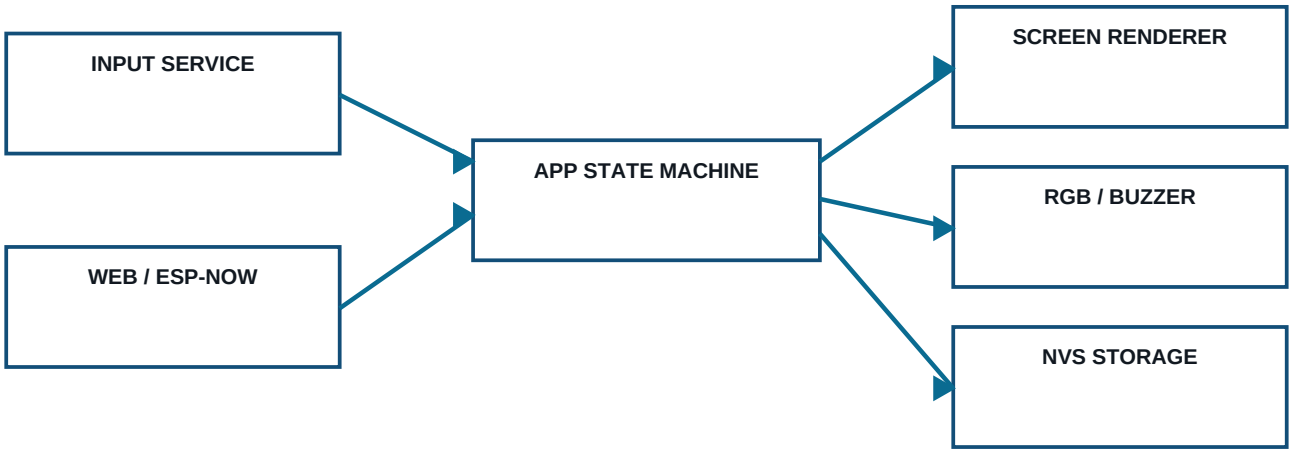


Figure 4. The one-file sketch is organized as cooperating subsystems around an application state machine.

Runtime loop

Order	Work performed
1	Serve pending local website requests
2	Poll the optional Back input and clear one-frame input flags
3	Process Morse key edges, dot/dash timing, and character gaps
4	Read filtered touch samples and produce app actions
5	Update communication status, notifications, RGB, and buzzer
6	Update the active app or game
7	Redraw changed UI or the next animation frame

Main data structures

AppState identifies the active screen. DeviceStats stores learning and game records. TouchCalibration stores orientation-specific mapping. MessagePacket and InboxMessage support ESP-NOW. ButtonState is reused for the optional Back input and touch-generated actions.

Storage

Preferences namespaces preserve user configuration, statistics, and touch calibration. Factory Reset clears saved values and returns the firmware to its first-boot state.

Source organization

The public firmware remains a single Arduino sketch for straightforward classroom use. Sections are separated by clear comments for pin definitions, data types, hardware startup, state handling, drawing, games, touch processing, RGB, web services, Morse conversion, and communication.

8. Physical build and troubleshooting

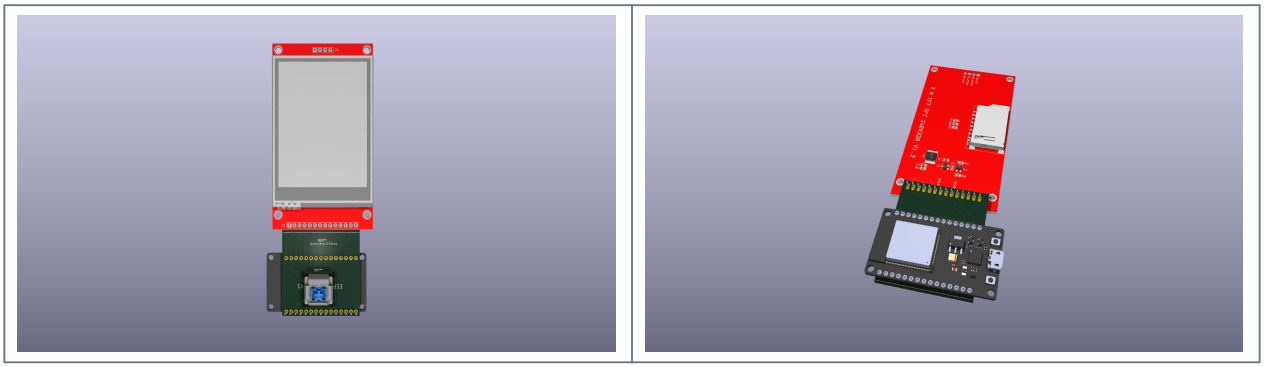


Figure 5. Conceptual front and perspective renders of the carrier-board arrangement. The renders communicate the intended form; the verified pin order controls the final wiring.

Current physical concept

The ESP32 and display connect through aligned headers on a compact carrier PCB. Only the traced portion requires board material; unused portions of the modules can extend beyond the carrier. RGB and buzzer placement remains flexible because those parts are connected separately. The final assembly is intended to mount on a stand or support structure.

Troubleshooting

Symptom	Correction
Display remains blank	Use the TFT_eSPI pin setup on page 4 and confirm the display header orientation
Touch does not respond	Use touch CS GPIO15, keep IRQ disconnected, and run calibration
Touch is offset after rotation	Open Settings and calibrate the selected orientation
Menu stops after leaving Settings	Upload the current v4.0 sketch; the return-state fix is included
Every tap opens immediately	Use the current v4.0 sketch, where first tap selects and second tap opens
Morse decoding feels wrong	Adjust WPM in Settings and leave a clear gap between letters
Website does not open	Join MorseStation with password morse1234, then open 192.168.4.1
Morse is absent from the site	In Morse to Web, tap SEND TO SITE after keying the message
RGB colors are exchanged	Use R=GPIO14, G=GPIO32, and B=GPIO26
Old navigation buttons do nothing	This is expected: those three GPIOs now drive RGB; use touch

9. Open-source publication reference

Recommended release contents

File	Purpose
touch_firmware.ino	Complete buildable firmware
README	Short project introduction, photographs, wiring, and upload steps
LICENSE	The chosen open-source license and copyright notice
docs/	This project guide, diagrams, PCB exports, and presentation material
hardware/	PCB source, Gerbers, and connector/pin reference when released

Version statement

This document describes firmware **v4.0** and the tested breadboard/carrier-board configuration dated 30 July 2026. It does not describe removed experiments such as the Sketch handwriting recognizer, SD-card booting, battery charging, or LoRa hardware.

Contribution areas

- Additional Morse learning content and accessibility improvements.
- New touch-first games that preserve Morse key timing and interface responsiveness.
- ESP-NOW pairing, message history, and multi-station classroom features.
- Optional SD storage, RTC, battery monitoring, sensors, and external input devices.
- PCB, enclosure, and mounting files based on the published pin map.

Project conventions

Keep SPI initialized once, keep touch in polling mode, preserve GPIO13 for the Morse key, avoid blocking delays during normal operation, save new settings through Preferences, and ensure every added screen is fully usable with touch.

Official software references

Arduino-ESP32 documentation
Espressif ESP-NOW documentation
TFT_eSPI repository
XPT2046_Touchscreen repository

Publication checklist

[ ] Firmware v4.0 included [ ] TFT_eSPI setup included [ ] Wiring matches page 3
[ ] Both renders included [ ] License included [ ] Version/date stated
[ ] Website instructions included [ ] Removed features not advertised [ ] Credits retained

Documentation baseline	Current touch_firmware.ino, working v4.0 feature set, and the two supplied project renders.