AI on the Edge with IoT; Using ESP32 & IDF

Curriculum Outline

Analog Data | **April 18–19, 2026** | Instructor: **Rajath Kumar K S** | edgeai.analogdata.io

---

Day 1 — ESP32-S3 & IDF Foundations

Saturday, April 18, 2026

Module 0 — Toolchain Setup & Docker Dev Environment

What you'll do: Pull the pre-built Docker image, flash your first IDF project, and verify your entire toolchain works — all within the first hour.

Experiments:

- Flash the Hello World project to your XIAO ESP32-S3
- Explore the IDF project structure and build system

Hardware: XIAO ESP32-S3, USB-C cable

Module 1 — GPIO: Digital Input & Output

What you'll do: Control LEDs and read button presses using the ESP-IDF GPIO driver.

Experiments:

- LED Blink — Blink an LED at a configurable rate
- Button-Controlled LED — Toggle LED on button press with software debouncing
- RGB LED Color Sequencer — Cycle through 7 colors using 3 GPIO pins

Hardware: LED, RGB LED, 2× Push Buttons, resistors

Module 2 — PWM: LED Dimming with Potentiometer

What you'll do: Use the LEDC peripheral to generate PWM signals and map potentiometer position to LED brightness in real time.

Experiments:

- Fixed PWM — Set LED to 50% brightness using 13-bit resolution
- Pot → Brightness — Turn the knob, watch the LED dim and brighten live

- Buzzer Tone — Play a 440 Hz tone using PWM on the passive buzzer

Hardware: LED, Potentiometer, Passive Buzzer

Module 3 — ADC & Capacitive Touch

What you'll do: Read analog voltages accurately and use the ESP32-S3's built-in capacitive touch sensor without any external IC.

Experiments:

- ADC with Voltage Display — Read pot position as voltage (mV) with 64-sample oversampling
- Capacitive Touch Sensor — Touch a wire, trigger an event — no external hardware needed
- Volume Knob Simulator — Pot sets level (0–100%), touch pad toggles mute

Hardware: Potentiometer, bare jumper wire (touch pad)

Module 4 — I2C: OLED Display

What you'll do: Communicate with the SSD1306 OLED over I2C and render text, graphics, and live sensor data on the 128×64 display.

Experiments:

- I2C Bus Scan — Discover all I2C devices on the bus by address
- OLED Hello World — Display text and workshop branding on screen
- Live Bar Graph — Pot reading displayed as a real-time animated bar on the OLED

Hardware: SSD1306 I2C OLED Display, Potentiometer

Module 5 — UART: Serial Communication

What you'll do: Use UART to control hardware via serial commands and parse real GPS location data from the NEO-6M module.

Experiments:

- LED Control via Terminal — Type ON / OFF in the serial monitor to control the LED
- GPS NMEA Parser — Read live NMEA sentences from NEO-6M and extract latitude/longitude in decimal degrees

Hardware: LED, NEO-6M GPS Module

Module 6 — SPI: Flash Memory

What you'll do: Talk to the W25Q32 SPI flash chip — read its manufacturer ID, write a string to memory, and read it back to verify.

Experiments:

- JEDEC ID Read — Identify the flash chip manufacturer and capacity over SPI
- Write & Read Back — Write "AnalogData2026" to address 0x000000 and verify on readback

Hardware: W25Q32 SPI Flash (4MB)

Module 7 — FreeRTOS: Real-Time Multitasking

What you'll do: Build a multi-task firmware architecture using the five most important FreeRTOS primitives. Understand why tasks deadlock and how to prevent it.

Experiments:

- Tasks — Run sensor and display tasks in parallel, pinned to separate cores
- Queues — Pass structured sensor data safely between tasks without shared memory
- Mutexes — Prevent OLED corruption when two tasks try to write simultaneously
- Event Groups — Wait until WiFi AND sensor are both ready before starting the data pipeline
- Software Timers — Blink a heartbeat LED every 500ms without blocking any task

Hardware: LED, OLED Display

Day 2 — Connectivity, Cloud & AI on the Edge

Sunday, April 19, 2026

Module 8 — WiFi: Modes & Connectivity

What you'll do: Connect to a router, create your own hotspot, and scan nearby networks — all three WiFi modes in one session.

Experiments:

- Station Mode (STA) — Connect to workshop WiFi router and get an IP address
- Access Point Mode (AP) — Turn the ESP32-S3 into a hotspot named AnalogData-ESP32
- WiFi Scan — List all nearby networks with SSID, RSSI signal strength, and channel

Hardware: XIAO ESP32-S3 (WiFi built-in)

Module 9 — HTTP: Web Client & Web Server

What you'll do: Make HTTP requests to external APIs and serve a web page from the ESP32 itself.

Experiments:

- HTTP GET — Fetch live IST time from a public API and print it to serial
- HTTP POST — Send DHT11 readings as JSON to a server endpoint
- Embedded Web Server — Control the LED from any browser on the same network — no app needed

Hardware: LED

Module 10 — MQTT: Publish / Subscribe Messaging

What you'll do: Implement the full MQTT model — broker, publisher, and subscriber — and build a live sensor dashboard.

Experiments:

- MQTT Publish — Send DHT11 temperature and humidity to a broker every 5 seconds
- MQTT Subscribe + LED Control — Control LED from your phone using MQTT Explorer or IoT MQTT Panel
- Live Dashboard — DHT11 data streams to Node-RED or Home Assistant dashboard in real time

Hardware: DHT11 Sensor, LED

Module 11 — mDNS, TCP & UDP

What you'll do: Discover the ESP32 by name on the network, build a raw TCP echo server, and broadcast sensor data over UDP without any connection overhead.

Experiments:

- mDNS — Access your board at http://esp32sensor.local — no IP address needed
- TCP Echo Server — Connect via `nc` from your laptop, send text, get it echoed back
- UDP Broadcast — Broadcast sensor JSON to all devices on the network every 2 seconds

Hardware: XIAO ESP32-S3

Module 12 — Sleep Modes & Power Management

What you'll do: Squeeze months of battery life from a 1000 mAh cell by using the ESP32-S3's sleep modes correctly.

Experiments:

- Light Sleep (Timer Wakeup) — CPU sleeps 5 seconds, wakes seamlessly, execution continues
- Deep Sleep (Timer Wakeup) — Board sleeps at ~14µA, boots every 30 seconds, tracks boot count in RTC memory
- Deep Sleep (GPIO Wakeup) — Board sleeps until button is pressed — ideal for door/window sensor

products
- Production Pattern — Wake → Read DHT11 → Publish MQTT → Sleep 60s — estimated 6 months on 1000 mAh

Hardware: Push Button, DHT11 Sensor

Module 13 — TinyML Basics: Data Collection & Training

What you'll do: Collect a labeled dataset from the DHT11 sensor, train a classifier in a Jupyter Notebook, and export a quantized TFLite model — ready to run on the chip.

Experiments:

- Data Logger — Firmware streams timestamp, temperature, humidity, label as CSV over serial
- Jupyter Training — Load dataset → normalize → train Dense NN → evaluate accuracy
- INT8 Quantization — Convert float32 model to INT8 TFLite (target: <5KB, suitable for OTA)

Hardware: DHT11 Sensor

Module 14 — TFLite Micro: On-Device Inference

What you'll do: Embed the trained model in firmware and run live inference on the ESP32-S3 — no cloud, no internet, no latency.

Experiments:

- Model Embedding — Convert `.tflite` → C array using `xxd`, include in firmware build
- Live Inference — Read DHT11 every 5 seconds → classify as normal / hot / humid → print result to serial

Hardware: DHT11 Sensor

Module 15 — Real-Time Audio Keyword Spotting *(Demo)*

What you'll watch & understand: Full pipeline from microphone to GPIO — say "Hey Analog", the LED triggers in under 50ms.

Pipeline: I2S Mic → PCM Audio → MFCC Features → TFLite CNN → Keyword Decision → GPIO

Topics covered:

- I2S microphone setup and DMA ring buffer
- MFCC feature extraction with ESP-DSP accelerated FFT
- CNN model architecture (Conv1D → MaxPool → Dense → Softmax)
- Sliding window inference for real-time detection
- Repo walkthrough: training dataset, Jupyter notebook, pre-trained model, full firmware

Module 16 — Camera + Person Detection *(Demo)*

What you'll watch & understand: End-to-end ML — from data collection with the OV2640 camera to on-device MobileNetV2 inference — with a face detection bonus.

Topics covered:

- Capture 96×96 JPEG images via HTTP from the ESP32 camera
- Train MobileNetV2 with transfer learning in Google Colab
- Fine-tune, quantize to INT8 (~900KB model)
- Run inference on-device — person detected → GPIO trigger + MQTT alert
- OV2640 camera configuration and frame capture
- Transfer learning and fine-tuning strategy
- PSRAM usage for frame buffer (8MB on XIAO ESP32-S3)
- Face detection with YuNet (~150ms inference)
- Full Colab notebook from raw images to deployed model

Module 17 — MQTT with AWS IoT Core

What you'll do: Connect the ESP32-S3 to AWS IoT Core over mutual TLS and build a full edge-to-cloud telemetry loop — sensor data flows all the way to DynamoDB.

Experiments:

- AWS Setup — Create a Thing, download device certificate and private key, attach IoT policy
- TLS Connection — Embed X.509 certificates in firmware, connect to AWS endpoint on port 8883
- Publish Telemetry — DHT11 readings publish as JSON to `analog-data/sensors` topic
- IoT Rule — Route incoming messages to DynamoDB table and CloudWatch metric — live in the console

Hardware: DHT11 Sensor

Your Hardware Kit

Every attendee receives the following kit (hardware is returned after the workshop):

Component	What You'll Use It For
Seeed Studio XIAO ESP32-S3	Main microcontroller — runs all experiments
LEDs (assorted)	GPIO, PWM dimming
RGB LED	Color sequencing, multi-channel PWM
Push Buttons (2×)	Digital input, debouncing
Breadboard + Jumper Wires	All circuit assembly

Component	What You'll Use It For
Potentiometer (10kΩ)	ADC readings, PWM control
I2C OLED Display (SSD1306)	Live data display
DHT11 Sensor	Temperature & humidity — TinyML dataset
SPI Flash (W25Q32, 4MB)	SPI communication, data storage
NEO-6M GPS Module	UART parsing, NMEA sentences
Passive Buzzer	PWM tones
INA219 Current Sensor	I2C, power measurement

What You Take Home

- All firmware source code for every experiment
- Jupyter notebooks for TinyML training (runs on Google Colab — free)
- Access to the workshop GitHub repository with full project structure
- Pre-built Docker image `analogdata/esp32s3-devenv:latest` — your dev environment forever

Prerequisites

Requirement	Level
C programming basics	Comfortable with structs, pointers, loops
Terminal / command line	Able to run commands, navigate directories
Python basics	For Jupyter notebook sessions
Prior ESP32 / embedded experience	Not required — we start from zero on Day 1

Questions?
Reach us at support@analogdata.io | build.analogdata.io | +91 96633 53992