Integration of Tektork IoT kit with Node-RED

1. Introduction

1.1. Overview of ESP32 in IoT Systems

The ESP32 is a versatile and powerful microcontroller featuring integrated Wi-Fi and Bluetooth capabilities, making it a popular choice for Internet of Things (IoT) applications. It supports a broad range of sensors and peripherals and provides robust wireless communication, which is essential for connecting embedded devices to cloud platforms and local servers. The ESP32's low power consumption, rich peripheral sets, and extensive software ecosystem enable developers to implement diverse IoT solutions, including data acquisition, device control, and remote monitoring. This microcontroller bridges the gap between physical sensors and IoT software frameworks such as MQTT, Node-RED, and cloud services like ThingSpeak and Firebase, facilitating seamless device-to-cloud integration and real-time data visualization.

1.2. Objective

This manual aims to guide users through the comprehensive setup, programming, and deployment of various IoT projects by interfacing Tektork IoT Kit with multiple popular platforms and technologies including ThingSpeak, MQTT brokers, Firebase cloud databases, Node-RED for flow-based programming, and the TIG Stack (Telegraf, InfluxDB, Grafana) for advanced data storage, monitoring and visualization. The objective is to provide a step-by-step reference combining hardware setup, firmware development, platform configuration, and dashboard creation, enabling readers to design scalable, networked IoT systems from scratch with detailed technical insights and best practices.

1.3. Required Hardware and Software

Hardware Requirements

For this manual and related projects, the hardware setup is kept minimal and focused on essential components:

- Tektork IoT Kit: The core microcontroller module with built-in Wi-Fi and Bluetooth connectivity, serving as the primary IoT device.
- USB Type-C Cable: Used to both power the ESP32 board and program it from a PC or laptop.
- PC or Laptop: Required for software development, programming, and interfacing with cloud platforms and local systems.

This simplified hardware setup optimizes project accessibility, making it feasible for users to start IoT development with the Tektork IoT Kit, connecting cable, and a host computer without additional peripheral sensors or modules.

Software Requirements:

The software environment consists of tools and platforms necessary for programming the ESP32, managing MQTT communications, cloud data aggregation, and IoT visualization:

- Arduino IDE or ESP-IDF: Primary development environments for writing and uploading firmware to the ESP32 microcontroller.
- MQTT Broker Service: Public brokers like HiveMQ or Mosquitto provide the messaging backbone for IoT device communication.
- ThingSpeak Account: A cloud-based IoT data platform for acquiring, storing, and visualizing data from connected devices.
- Firebase Console: Google's cloud service offering a real-time database, authentication, and hosting for IoT backend needs.
- Node-RED: A visual flow-based programming tool for creating IoT workflows and dashboards without coding complexity.
- TIG Stack Components:
  Telegraf: An agent for collecting and processing time-series data.
  InfluxDB: A time-series optimized database for storing sensor data.
  Grafana: A dashboard platform to visualize and analyze IoT data effectively.
- Supporting Libraries and Tools: MQTT client libraries for ESP32, HTTP client libraries, and SDKs compatible with ESP32 and relevant platforms.

This software setup targets versatility and scalability, enabling development from basic sensor data visualization up to advanced real-time monitoring and control systems.

2. Setting Up the Tektork IoT Kit Development Environment

Refer the Document tilted
a) Tektork-Kit-ArduinoIDE-Driver-Installation-Notes.pdf
b) Tektork-IoT-Board-Hardware-Manual-V1.0.pdf

3. Node-RED with ESP32

3.1. Installing Node-RED(Windows/Linux/Docker)

Windows:

- Use the Node.js installer (from nodejs.org), then install Node-RED globally via:
  - npm install -g --unsafe-perm node-red
- Start Node-RED in CMD:
  - node-red
- Access the Node-RED editor at: http://localhost:1880.

Linux:

- Install Node.js and npm using package manager.
- Run the same npm install -g command as above.
- Start with node-red and access via browser.

Docker:

- Pull the official Docker image:
  - docker pull nodered/node-red
- Run Node-RED in a container:
  - docker run -it -p 1880:1880 --name mynodered
  - nodered/node-red
- The editor is accessible on port 1880 of your host.

3.2. Setting Up MQTT Nodes in Node-RED

- Click the hamburger menu, then “Manage palette” > “Install,” and search for “node-red-dashboard” and “node-red-contrib-mqtt-broker.”
- Drag MQTT-in and MQTT-out nodes to the flow.
- Double-click a node, add new MQTT broker settings:
  - Server: broker.hivemq.com (or your chosen broker)
  - Port: 1883
- Set MQTT topics to match those in your ESP32 code (e.g. esp32pot, esp32ldr for sensors, esp32led1, esp32led2 for LEDs).
- Wire input nodes to display/processing nodes and output nodes to dashboard controls.

3.3. Creating Dashboards(Switches,Gauges,Graphs)

- Install “node-red-dashboard” if not already done.
- Use “ui_switch” for LED control, with on/off as payloads, topic mapped to esp32led1 or esp32led2.
- Add “ui_gauge” nodes for real-time visualization of potentiometer and LDR sensor values. Feed them with data from “mqtt in” nodes for esp32pot and esp32ldr.
- Add “ui_chart” to log and analyze trends over time, connecting the data lines from the respective MQTT input nodes.
- Groups and tabs (dashboard layout) can be adjusted in dashboard sidebar for organized visual flow.

3.4. ESP32 Data Visualization on Node-RED Dashboard

- With MQTT communication set up, sensor values published from the ESP32 (e.g. analogRead on pins 36 and 39) show up live on gauges and charts.
- Toggling a dashboard switch sends a payload through MQTT, and the ESP32 reacts (turning LEDs on/off on pins 33 and 4).
- All sensor activities and actuator statuses are logged in real-time, accessible and interactable via any web browser at http://<node-red-ip>:1880/ui.
- Node-RED’s built-in charting and dashboard components enable responsive IoT visualizations and controls without extra code.

3.5. Working Model Demonstration(ESP32 to Node-RED to Dashboard)

- ESP32 reads potentiometer and LDR (GPIO 36, 39), publishes values to MQTT topics (esp32pot, esp32ldr) every second.

- Node-RED receives the data, displays it on live dashboard gauges & charts, and logs value history.
- LED controls on the dashboard send on/off commands via MQTT (esp32led1, esp32led2), instantly switching physical LEDs connected to the ESP32.
- This setup provides real-time remote monitoring and manual control over all project peripherals from any device running a browser, making it suitable for presentations, labs, or production IoT deployments.