Integration of Tektork IoT kit with GoogleSheets

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. Google Sheets Integration

3.1. Introduction to Google Apps Script and Webhooks

Google Apps Script is a cloud-based scripting platform for light-weight application development in the Google Workspace environment, allowing automation and integration of Google Sheets with external systems.

- Webhooks are HTTP callbacks that enable ESP32 (or any IoT device) to trigger a Google Apps Script through a simple web request (GET or POST).
- By deploying Apps Script as a web app, Sheets can receive data from ESP32 sensor readings, providing real-time cloud data logging.

3.2. ESP32 Data Logging using Google Sheets API

The integration is established by creating a Google Apps Script bound to a Sheets file and deploying it as a web app.

The Script provides a URL endpoint. ESP32 sends data via HTTP request. The Script captures timestamp, potentiometer, and LDR readings, and appends them as new rows in the spreadsheet.
Supports both POST (with JSON) and GET (with URL parameters) request styles for data logging:

```
function doPost(e) {
  try {
    var sheet = SpreadsheetApp.getActiveSheet();
    var data = JSON.parse(e.postData.contents);
    var timestamp = new Date();
    sheet.appendRow([timestamp, data.pot, data.ldr]);
    return ContentService.createTextOutput(JSON.stringify({status:"success"}))
      .setMimeType(ContentService.MimeType.JSON);
  } catch (error) {
    return ContentService.createTextOutput(JSON.stringify({status:"error", message:
error.toString()}))
      .setMimeType(ContentService.MimeType.JSON);
  }
}
```

3.3. Sending Sensor Data from ESP32 to Google Sheets

The ESP32 reads analog values, then uses the WiFi and HTTPClient libraries to make a web request to the deployed Google Apps Script web app:

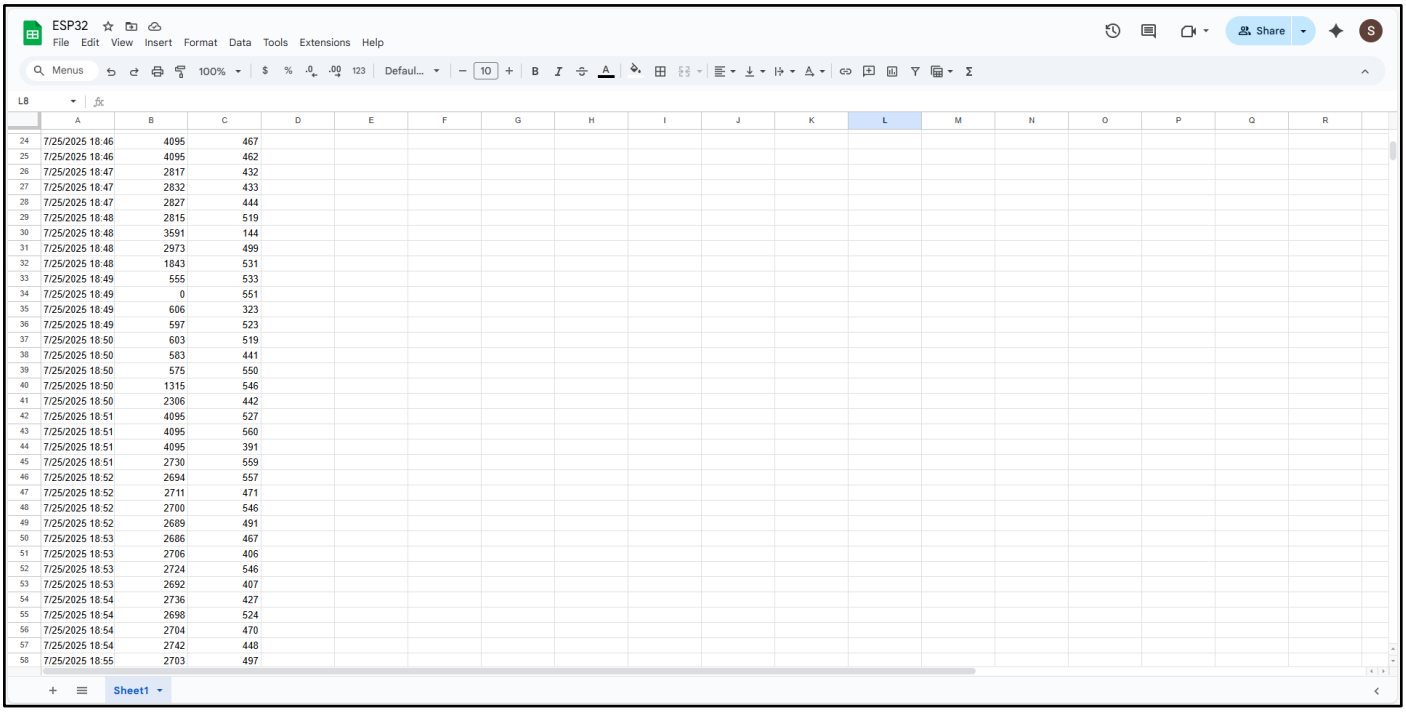
```
#include <WiFi.h>
#include <HTTPClient.h>
const char* ssid = "S23FE";
const char* password = "Sanjey123@";
const char* googlescriptid = "AKfycbw6Oecf6kymGD27SO1jWXdTTAFgPQO5C2Cm0q27-QHEkV8S-
rXZCm7GPYPJOpZGFS1Kg";

const int potPin = 36;   // Potentiometer pin
const int ldrPin = 39;   // LDR pin

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  pinMode(potPin, INPUT);
  pinMode(ldrPin, INPUT);
  while (WiFi.status() != WL_CONNECTED) delay(1000);
}

void loop() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    int potValue = analogRead(potPin);
    int ldrValue = analogRead(ldrPin);
    String url = String("https://script.google.com/macros/s/") + googlescriptid +
"/exec?pot=" + String(potValue) + "&ldr=" + String(ldrValue);
    http.begin(url);
    int httpCode = http.GET();
    http.end();
```

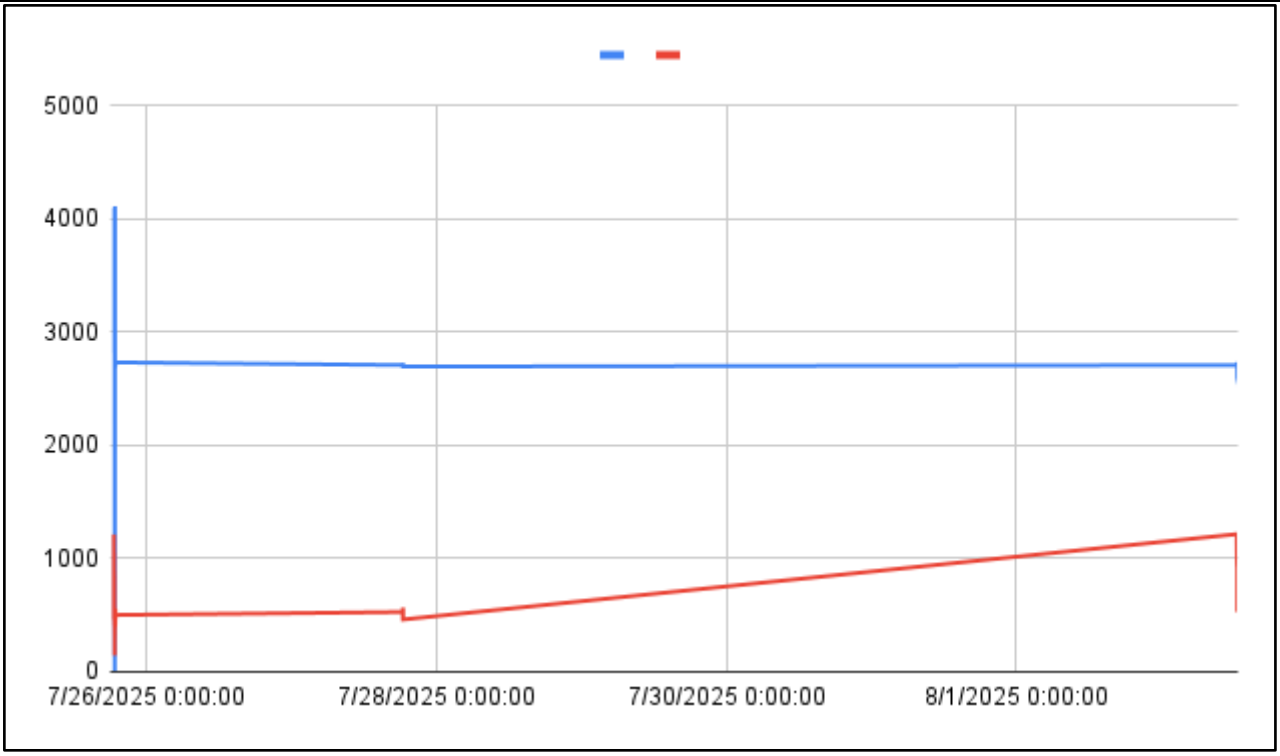
```
  }
  delay(10000); // Send data every 10 seconds
}
```



	A	B	C
24	7/25/2025 18:46	4095	467
25	7/25/2025 18:46	4095	462
26	7/25/2025 18:47	2817	432
27	7/25/2025 18:47	2832	433
28	7/25/2025 18:47	2827	444
29	7/25/2025 18:48	2815	519
30	7/25/2025 18:48	3591	144
31	7/25/2025 18:48	2973	499
32	7/25/2025 18:48	1843	531
33	7/25/2025 18:49	555	533
34	7/25/2025 18:49	0	551
35	7/25/2025 18:49	606	323
36	7/25/2025 18:49	597	523
37	7/25/2025 18:50	603	519
38	7/25/2025 18:50	583	441
39	7/25/2025 18:50	575	550
40	7/25/2025 18:50	1315	546
41	7/25/2025 18:50	2306	442
42	7/25/2025 18:51	4095	527
43	7/25/2025 18:51	4095	560
44	7/25/2025 18:51	4095	391
45	7/25/2025 18:51	2730	559
46	7/25/2025 18:52	2694	557
47	7/25/2025 18:52	2711	471
48	7/25/2025 18:52	2700	546
49	7/25/2025 18:52	2689	491
50	7/25/2025 18:53	2686	467
51	7/25/2025 18:53	2706	406
52	7/25/2025 18:53	2724	546
53	7/25/2025 18:53	2692	407
54	7/25/2025 18:54	2736	427
55	7/25/2025 18:54	2698	524
56	7/25/2025 18:54	2704	470
57	7/25/2025 18:54	2742	448
58	7/25/2025 18:55	2703	497

3.4. Live Graphs and Logs in Google Sheets

- Google Sheets automatically logs every received data row with timestamp, POT, and LDR values.
- Use built-in chart functions to create live graphs:
  - Select the logged columns (A: timestamp, B: POT, C: LDR)
  - Click "Insert" > "Chart", choose "Line chart" or "Scatter chart"
- As new data is appended by the Apps Script, charts update in real-time without any manual refresh needed.
- This enables continuous monitoring, quick pattern detection, and easy sharing with collaborators for IoT analytics.



3.5. Working Model Demonstration(ESP32 to Sheets Logging)

- Deployed Scenario: ESP32 reads POT (GPIO 36) and LDR (GPIO 39), pushes each synchronized reading to Google Sheets via HTTP.
- Validation: Instantly view new log entries and chart updates in Sheets as the ESP32 runs.
- Debugging: Check serial monitor for HTTP response codes to verify delivery; inspect the “Status” column in the Script output for troubleshooting.
- Applications: This flow allows long-term archival, cloud visualization, and alerting setups—all inside the Google Sheets environment.