Hardware Documentation

FireStore

In Firestore:

* **Collections**: These are like folders — they group related documents.
* **Documents**: Each document is like a JSON object containing key-value pairs (your sensor data).
* **Document IDs**: Each document needs a unique identifier. If you don’t specify one, Firestore will auto-generate a random ID.

**Do you need to create a collection beforehand?**

No, Firestore will automatically create the collection if it doesn’t exist when you add the first document.  
In the code I shared, the sensorData collection will be created automatically if it doesn’t already exist.

A screen shot of a computer

AI-generated content may be incorrect.

**Timestamp:**

* Uses NTP (Network Time Protocol) to get the current time in UTC format, suitable for Firestore’s timestampValue.

**MQTT and TLS (Transport Layer Security)**

Importance,

Without TLS, MQTT sends data in plain text, meaning:

* Your sensor data (like soil moisture, temperature) can be intercepted.
* Hackers could publish fake data to your topics.
* Unauthorized clients might subscribe and steal sensitive data.

TLS is a cryptographic protocol that secures data transmission by encrypting the communication between clients (like your ESP32) and the MQTT broker (like Mosquitto).

Protects data from eavesdropping by encrypting messages. Even if someone intercepts the data, they can't read it.

Two methods,

* One-way authentication: Only the server (broker) has a certificate, and clients trust the server.
* Two-way (mutual) authentication: Both the broker and clients have certificates more secure but harder to set up.

**Micro SD card for Store and Forward data**

Steps to do,

1. Sensor reads → Save each reading to MicroSD (with a timestamp)
2. Check internet connection (WiFi status)
3. If no WiFi, keep saving locally
4. If WiFi available, read saved data from MicroSD and upload batch by batch to Firestore
5. After successful upload, delete uploaded data from SD card

Module - Micro SD TF Card Reader Module with SPI interface

**SPI** (Serial Peripheral Interface) is a high-speed, full-duplex communication protocol used to transfer data between a microcontroller (like ESP32) and peripheral devices such as:

* SD card modules
* LoRa modules
* Displays (OLED, TFT)
* Sensors

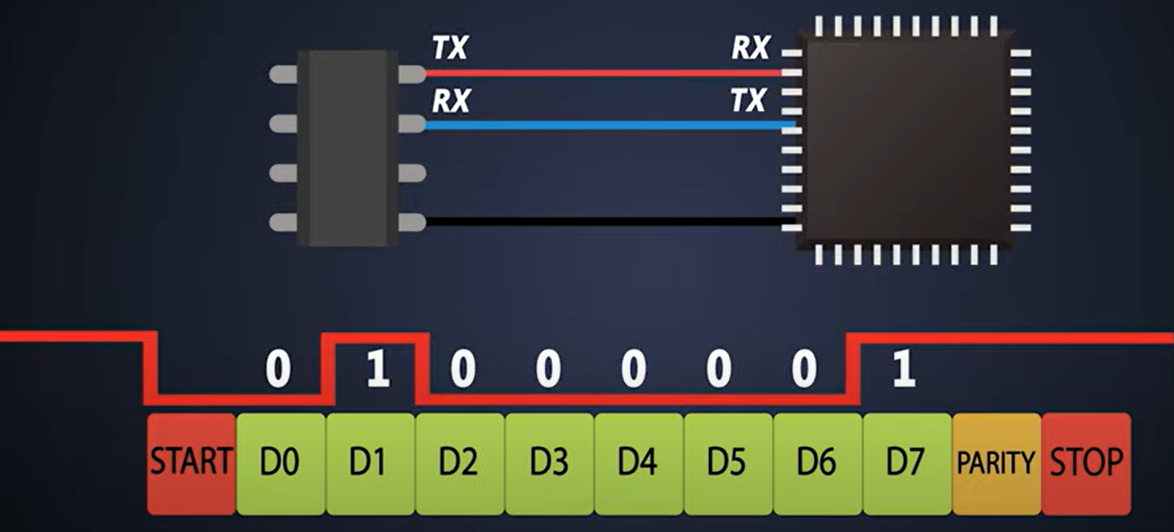
SPI uses 4 main lines,

* MOSI - Master Out Slave In (Data to device)
* MISO - Master In Slave Out (Data from device)
* SCLK - Serial Clock (provided by master)
* CS/SS - Chip Select / Slave Select (activates a device)

**UART (Universal Asynchronous Receiver/Transmitter)**

UART is a hardware protocol for serial communication.

Peer to peer/ Ad hoc topology.



Data frame of UART

**Test Code**

Prompt;

Give me platformIO code for below senario.  
RTC module and sd card module are connected to the esp32. Also DHT22 sensor and GPS sensor are connected to get those sensor data. I want get the sensor data and upload them to firestore with time stamp. If wifi available send the data directly if not write them in sd card with timestamp. After that when wifi available in next time upload them to firestore. After considerable time delete sd card data which are uploaded to firestore.

#include <Arduino.h>

#include <WiFi.h>

#include <Wire.h>

#include <SPI.h>

#include <SD.h>

#include <RTClib.h>

#include <DHT.h>

#include <TinyGPSPlus.h>

#include <Firebase\_ESP\_Client.h>

// ——— CONFIG ——————————————————————————————————————————————

/\* Wi-Fi \*/

const char\* ssid = "YOUR\_WIFI\_SSID";

const char\* password = "YOUR\_WIFI\_PASS";

/\* Firestore \*/

#define API\_KEY "YOUR\_FIREBASE\_API\_KEY"

#define PROJECT\_ID "YOUR\_FIREBASE\_PROJECT\_ID"

#define USER\_EMAIL "YOUR\_FIREBASE\_USER\_EMAIL" // if using email sign-in

#define USER\_PASSWORD "YOUR\_FIREBASE\_USER\_PASS"

FirebaseData fbdo;

FirebaseAuth auth;

FirebaseConfig config;

/\* Pins \*/

#define DHT\_PIN 4

#define DHT\_TYPE DHT22

#define SD\_CS\_PIN 5

#define GPS\_RX 16 // to TX of GPS

#define GPS\_TX 17 // to RX of GPS

// ——— HARDWARE OBJECTS —————————————————————————————————————

RTC\_DS3231 rtc;

DHT dht(DHT\_PIN, DHT\_TYPE);

TinyGPSPlus gps;

HardwareSerial GPS(1);

// ——— SETUP ——————————————————————————————————————————————

void setup() {

Serial.begin(115200);

Wire.begin(); // I2C for RTC

dht.begin();

GPS.begin(9600, SERIAL\_8N1, GPS\_RX, GPS\_TX);

// init SD

SPI.begin(18, 19, 23, SD\_CS\_PIN);

if (!SD.begin(SD\_CS\_PIN)) {

Serial.println("SD init failed!");

}

// init RTC

if (!rtc.begin()) {

Serial.println("DS3231 not found");

}

if (rtc.lostPower()) {

// set to compile time

rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

}

// init Firebase

config.api\_key = API\_KEY;

auth.user.email = USER\_EMAIL;

auth.user.password = USER\_PASSWORD;

config.project\_id = PROJECT\_ID;

Firebase.begin(&config, &auth);

Firebase.reconnectWiFi(true);

// connect Wi-Fi (we’ll check status later)

WiFi.begin(ssid, password);

}

// ——— HELPERS ——————————————————————————————————————————————

// return ISO8601 timestamp

String getTimestamp() {

DateTime now = rtc.now();

char buf[20];

sprintf(buf, "%04u-%02u-%02uT%02u:%02u:%02uZ",

now.year(), now.month(), now.day(),

now.hour(), now.minute(), now.second());

return String(buf);

}

// append a line to /data.csv on SD

void logToSD(const String &line) {

File f = SD.open("/data.csv", FILE\_APPEND);

if (f) {

f.println(line);

f.close();

Serial.println("Logged to SD: " + line);

}

}

// upload one JSON record to Firestore under collection "readings"

bool uploadRecord(const String &ts, float hum, float temp, double lat, double lon) {

String path = String("/readings/") + ts; // document ID = timestamp

FirebaseJson j;

j.set("timestamp", ts);

j.set("humidity", hum);

j.set("temperature", temp);

j.set("latitude", lat);

j.set("longitude", lon);

return Firebase.Firestore.createDocument(&fbdo, PROJECT\_ID, "(default)", "readings", ts.c\_str(), j.raw(), "timestamp");

}

// read all lines, upload, then remove data.csv

void flushSDToCloud() {

if (!SD.exists("/data.csv")) return;

File f = SD.open("/data.csv");

if (!f) return;

while (f.available()) {

String line = f.readStringUntil('\n');

line.trim();

if (line.length() == 0) continue;

// CSV: timestamp,humidity,temperature,lat,lon

auto parts = std::vector<String>();

int idx=0;

while (idx < (int)line.length()) {

int comma = line.indexOf(',', idx);

if (comma == -1) comma = line.length();

parts.push\_back(line.substring(idx, comma));

idx = comma + 1;

}

if (parts.size() == 5) {

uploadRecord(parts[0], parts[1].toFloat(), parts[2].toFloat(),

parts[3].toDouble(), parts[4].toDouble());

}

}

f.close();

SD.remove("/data.csv");

Serial.println("Flushed SD → Firestore");

}

// ——— MAIN LOOP ————————————————————————————————————————————

void loop() {

// read sensors

float h = dht.readHumidity();

float t = dht.readTemperature();

// read GPS

while (GPS.available()) gps.encode(GPS.read());

double lat = gps.location.isValid() ? gps.location.lat() : 0.0;

double lon = gps.location.isValid() ? gps.location.lng() : 0.0;

String ts = getTimestamp();

String record = String(ts) + "," + h + "," + t + "," + lat + "," + lon;

// Wi-Fi check

if (WiFi.status() != WL\_CONNECTED) {

// try reconnect once

WiFi.begin(ssid, password);

delay(5000);

}

if (WiFi.status() == WL\_CONNECTED) {

// upload current

if (uploadRecord(ts, h, t, lat, lon)) {

Serial.println("Uploaded: " + record);

} else {

// fallback to SD

logToSD(record);

}

// then flush backlog

flushSDToCloud();

} else {

// offline -> log only

logToSD(record);

}

delay(60000); // wait 1 minute between readings

}

**Work according to Button Press**

#define BUTTON\_PIN 13 // GPIO pin connected to button

void setup() {

Serial.begin(115200);

pinMode(BUTTON\_PIN, INPUT\_PULLUP); // internal pull-up enabled

}

void loop() {

// Read the button state (LOW means pressed)

if (digitalRead(BUTTON\_PIN) == LOW) {

Serial.println("Button pressed by human!");

// Run your code here

delay(500); // debounce delay

}

}

Not working until comes to ready mode.

Once pressed not working for processing time period. Until ready mode.

If one button pressed other button not working.

Indicate those by a light

Portable (Device 1)

* GPS (on board)
  + 16 (RX2)
  + 17 (TX2)
* NPK
  + 16 (RX2)
  + 17 (TX2)
* Soil Moisture
  + 34
* SD (on board)
  + 5 (SD\_CS)
  + 18 (SCK)
  + 19 (MISO)
  + 23 (MOSI)
* Step down
  + VCC
  + GND
* RTC (on board)
  + 21 (SDA)
  + 22 (SCL)
  + SQW
  + 32K
* Display
  + 21 (SDA)
  + 22 (SCL)
* Power switch
* Two Switches
  + D12
  + D14
* Two LEDs
  + D27
  + D33
* TTL to RS485
  + D32
  + R0 to D25
  + D1 to D26

(16, 17)\*2

(21, 22)\*2

Fixed (Device 2)

* Temperature and Humidity
* Hall sensor
* Light sensor
* 