## 20×40 mm target with minimal BOM and full support for:

* Native USB
* RFM69HCW (SPI)
* UM980 GNSS (UART)
* Optional buzzer
* Power from USB or 5–8V

## ESP32-C6FH4 Hardware Design Overview

**📌 Core Specs (C6FH4)**

* QFN-48, 5×5 mm
* Native USB-CDC (no external USB-UART needed)
* 4 MB flash integrated
* Runs at 3.3
* No shielding required if Wi-Fi/BLE is unused
* 12 MHz crystal recommended

## Required Components and Connections

**1. Power Supply**

You want to support:

* USB (5V)
* External 5–8V

Recommended:

* MP2338 buck converter (efficient, 5–24 V → 3.3 V)
* OR AP2112K or XC6220 (LDO, 5–6 V → 3.3 V)

🔌 Connect 3.3 V output to:

* VDD3P3\_CPU
* VDD3P3\_RTC
* Decouple with 10 µF + 0.1 µF caps

**2. USB Interface (native USB-CDC)**

Pins:

* GPIO19 → USB D−
* GPIO20 → USB D+

Passives:

* 22 Ω series resistors on D+ and D− (optional if short traces)
* Optional ESD protection (e.g., USBLC6-2SC6)

Connector:

* USB-C (with CC resistors) or micro-USB

**3. Crystal**

* 12 MHz crystal (e.g., ECS-120-20-4X)
* Load caps: 2 × 18–22 pF
* Use close layout with ground guard

**4. RFM69HCW**

Interface: SPI

Pins (example):

* MOSI → GPIO6
* MISO → GPIO5
* SCK → GPIO4
* NSS → GPIO7
* DIO0 → GPIO8 (interrupt)

Use 0.1 µF decoupling cap close to VCC pin.

**5. UM980 GNSS Module**

Interface: UART

Pins (example):

* TXD → GPIO1
* RXD → GPIO2

Baud rate: 115200 (typical)

Level shifting not needed — both run at 3.3 V

**6. Miniature Beeper**

Use GPIO + NPN transistor to drive buzzer:

* GPIO10 → 1kΩ → base of 2N3904
* Emitter → GND
* Collector → Negative side of buzzer

Positive side → 3.3V

Buzzer: 5×5×1.8 mm passive piezo (e.g., CPT-5018C)

**7. Boot/Reset Logic (Optional)**

* GPIO9 → BOOT (pull LOW to enter flash mode)
* EN → reset (pull LOW to reset chip)

You can use RTS/DTR from USB-to-PC to control these via a transistor auto-reset circuit (or omit if using USB-native flashing only).

**8. Programming**

No USB-to-UART chip needed. Flash directly using:

esptool.py --chip esp32c6 --port COMx write\_flash 0x0 firmware.bin

🧠 Flash Memory for Settings

Use ESP32 NVS (Non-Volatile Storage) API:

* Stores key-value data in internal flash
* Reliable across reboots and power loss
* Available in Arduino and ESP-IDF

**✅ Pin Usage Summary (example)**

**Purpose ESP32-C6 Pin**

USB D− GPIO19

USB D+ GPIO20

RFM69 NSS GPIO7

RFM69 DIO0 GPIO8

SPI MOSI GPIO6

SPI MISO GPIO5

SPI SCK GPIO4

UM980 RX GPIO1

UM980 TX GPIO2

Buzzer GPIO10

BOOT (option) GPIO9

EN (reset) EN pin

**🧩 1. USB-C Connection Diagram (Native USB for ESP32-C6)**

**✅ Objective:**

* Use **USB-C** connector to connect ESP32-C6FH4’s **native USB-CDC interface**
* Add **CC pull-down resistors** so hosts detect it as a USB device
* Add optional **ESD protection**

**📌 USB-C Pinout (for USB 2.0 only)**

You only need **USB 2.0 D+/D−**, **GND**, and **VBUS**:

USB-C Receptacle (Front View, PCB side)

┌───────────────┐

| A1 ─ GND | B1 ─ GND

| A2 ─ TX1+ | B2 ─ TX2+

| A3 ─ TX1− | B3 ─ TX2−

| A4 ─ VBUS | B4 ─ VBUS

| A5 ─ CC1 \* | B5 ─ CC2 \*

| A6 ─ D+ →───┐| B6 ─ D+ →──┐

| A7 ─ D− →───┘| B7 ─ D− →──┘

| A8 ─ SBU1 | B8 ─ SBU2

| A9 ─ VBUS | B9 ─ VBUS

| A10─ RX2− | B10─ RX1−

| A11─ RX2+ | B11─ RX1+

| A12─ GND | B12─ GND

└───────────────┘

**🔧 Schematic Snippet**

VBUS ──────> 5V input to MP2338

D+ (A6, B6) ──────> GPIO20 (ESP32-C6)

D− (A7, B7) ──────> GPIO19 (ESP32-C6)

CC1 ───┬── 5.1kΩ ──> GND

└── Optional ESD diode (e.g., USBLC6)

CC2 ───┬── 5.1kΩ ──> GND

└── (same)

GND ───────────────> GND

**⚠️ Notes:**

* Use **one 5.1 kΩ pull-down** on **each CC pin** (CC1, CC2) to signal to the host that your device is a USB peripheral.
* **Do not connect TX/RX or SBU** — those are for USB 3.0 or alternate modes.
* Add 22 Ω resistors in series on D+ and D− (optional, if trace is <1" and well-matched).
* Optional: Add **TVS diode array** (e.g. ESD protection: USBLC6-2SC6 or PESD5V0S1UL).

**⚡ 2. Power Tree with MP2338 (5–8 V input)**

**✅ Requirements:**

* Input: 5–8 V from USB or external
* Output: **3.3 V** to power ESP32-C6, RFM69, UM980, and buzzer

**📌 MP2338 Setup (Simplified)**

| **Pin** | **Connect to** |
| --- | --- |
| **VIN** | 5–8 V input (VBUS or ext) |
| **EN** | Tie to VIN (via 10k pull-up or direct) |
| **SW** | 3.3V output node (to inductor) |
| **FB** | Feedback from VOUT (voltage divider) |
| **GND** | Ground plane |
| **BST** | Bootstrap cap (0.1 µF) between SW and BST |

**🔧 Schematic Snippet**

plaintext

CopyEdit

VIN (USB or ext) ─────┐

|

[CIN] 10µF

|

VIN (MP2338)

|

EN ← VIN

|

+--------+

| MP2338 |

+--------+

|

SW ────┬───→ Inductor 2.2 µH →────┐

| |

[D1] Schottky diode |

| [COUT] 10µF

GND |

VOUT (3.3 V)

↓

┌───────────────────────────────┐

│ Powers: ESP32, RFM69, UM980 │

└───────────────────────────────┘

FB → voltage divider: 3.3V = VREF × (1 + R1/R2)

Use: R1 = 100k, R2 = 100k (to get ~3.3V)

**CIN**: 10–22 µF  
**COUT**: 10–22 µF  
**Inductor**: 2.2–4.7 µH  
**D1**: 1A Schottky (e.g., SS14)

**🧠 Optional Enhancements**

* Add reverse-polarity protection on VIN (Schottky or PFET)
* Add a simple PTC fuse on VBUS
* Use a buck-boost if needing 3.3 V from <5 V sources occasionally

**✅ Combined Summary**

* USB-C with 5.1 k CC resistors and D+/D− → ESP32-C6
* MP2338 drops 5–8 V input to clean 3.3 V for all components
* Minimal parts, <20×40 mm layout achievable
* Fully USB flashable with no buttons, no UART chi