بسم الله الرحمن الرحیم : ساختار پیشنهادی و اولیه پروژه

```
user_code_folder/
├── CMakeLists.txt
│
├── include/
│   └── user_code/
│       ├── user_code.h              # هدر جامع (اختیاری)
│       ├── app_api.h                # API عمومی برنامه
│       ├── event_bus.h              # API event bus
│       ├── event_ids.h              # تعریف تمام event IDs
│       ├── module_ids.h             # تعریف module IDs
│       ├── drivers_public.h         # API عمومی درایورها
│       ├── network_service.h        # API سرویس شبکه
│       ├── hardware_service.h       # API سرویس سخت‌افزار
│       └── app_events.h             # رویدادهای اختصاصی برنامه
│
├── app/                             # سطح 5 (بالاترین)
│   ├── app_main.c                   # تابع اصلی برنامه (Task)
│   ├── app_main.h                   # هدر app_main
│   ├── app_state_machine.c          # ماشین حالت برنامه
│   ├── app_state_machine.h          # هدر ماشین حالت
│   └── app_events.c                 # پردازش رویدادهای برنامه
│
├── services/                        # سطح 4
│   ├── network_service.c            # سرویس شبکه یکپارچه (Task)
│   ├── network_service.h            # هدر سرویس شبکه
```

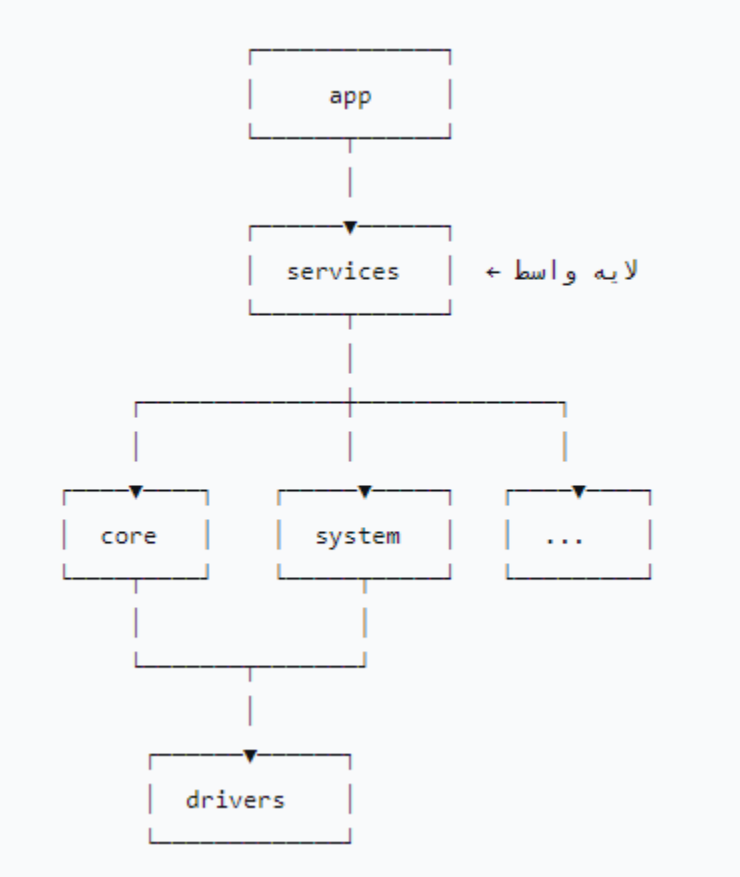
```
│   ├── network_internal.h          # توابع داخلی سرویس شبکه
│   ├── hardware_service.c          # سرویس سخت‌افزار یکپارچه (Task)
│   ├── hardware_service.h          # هدر سرویس سخت‌افزار
│   └── hardware_internal.h         # توابع داخلی سرویس سخت‌افزار
│
├── core/                           # سطح 3&2 (همسطح system)
│   ├── event_bus/
│   │   ├── event_bus.c             # پیاده‌سازی event bus
│   │   ├── event_bus.h             # هدر event bus
│   │   └── event_bus_internal.h    # توابع داخلی event bus
│   ├── log.c                       # سیستم لاگ
│   ├── log.h                       # هدر لاگ
│   ├── timers.c                    # سیستم تایمر
│   ├── timers.h                    # هدر تایمر
│   └── utils/
│       ├── utils.c                 # توابع کمکی عمومی
│       ├── utils.h                 # هدر توابع کمکی
│       ├── ring_buffer.c           # بافر حلقوی
│       ├── ring_buffer.h           # هدر بافر حلقوی
│       ├── crc.c                   # محاسبات CRC
│       └── crc.h                   # هدر CRC
│
├── system/                         # سطح 3&2 (همسطح core)
│   ├── sys_init.c                  # مقداردهی اولیه سیستم
│   ├── sys_init.h                  # هدر sys_init
│   ├── sys_config.c                # کانفیگ سیستم
│   ├── sys_config.h                # هدر sys_config
│   ├── rtos_wrapper.c              # توابع wrapper برای RTOS
│   ├── rtos_wrapper.h              # هدر rtos_wrapper
│   ├── power_manager.c             # مدیریت مصرف انرژی (Task)
│   └── power_manager.h             # هدر power_manager
│
```

```
└── drivers/                    # سطح 1 (پایین‌ترین)
    ├── uart/
    │   ├── uart_driver.c       # درایور UART
    │   └── uart_driver.h       # هدر درایور UART
    └── gpio/
        ├── gpio_driver.c       # درایور GPIO
        └── gpio_driver.h       # هدر درایور GPIO
```

سلسله مراتب دسترسی:

معماری صحیح با core و system همسطح:

- توجه شود که لایه های همسطح مجاز به استفاده از هم نیستند و اگر ارتباطی نیاز شد ، باید تحت کنترل لایه بالایی خودشان (که علاوه بر نقش خود ،نقش لایه واسط برای اینان ایفا میکند) باشد.



توضیحات کلی:

کدام فایل‌ها Task دارند

کدام ماژول Event Publish می‌کند

Dependencyبین سرویس‌ها چیست

1. فایل‌هایی که Task دارند:

فایل	نام Task	اولویت	وظیفه	پریود
app/app_main.c	`app_task`	پایین (2)	منطق اصلی برنامه، ماشین حالت، پردازش داده	Event-driven
services/network_service.c	`network_task`	متوسط (4)	مدیریت یکپارچه MQTT، HTTP، OTA، Net	100ms
services/hardware_service.c	`hardware_task`	بالا (5)	جمع‌آوری داده GPS، UART، GPIO، Device Manager	10ms
system/power_manager.c	`system_task`	خیلی بالا (7)	مانیتورینگ سیستم، مدیریت Sleep/Wake	500ms

2. ماژول‌هایEvent Publisher (ارسال‌کننده رویداد)

ماژول	فایل	رویدادهای منتشر شده	هدف	Destination Task

Hardware Service	`services/hardware_service.c`	`EVENT_GPS_FIX`
		`EVENT_GPS_LOCATION`
		`EVENT_UART_DATA`
		`EVENT_BUTTON_PRESS`
		`EVENT_DEVICE_ATTACHED`
			موقعیت مکانی
			داده سریال
			رویداد دکمه
			اتصال دستگاه
				→ App Task

Network Service	`services/network_service.c`	`EVENT_NET_CONNECTED`
		`EVENT_NET_DISCONNECTED`
		`EVENT_MQTT_DATA`
		`EVENT_HTTP_RESPONSE`
		`EVENT_OTA_PROGRESS`
		`EVENT_OTA_COMPLETE`
			وضعیت شبکه
			داده MQTT
			پاسخ HTTP
			پیشرفت OTA
				→ App Task

Power Manager	`system/power_manager.c`	`EVENT_BATTERY_LOW`
		`EVENT_SYSTEM_SLEEP`
		`EVENT_SYSTEM_WAKE`
		`EVENT_WATCHDOG`
			وضعیت باتری
			خواب سیستم
			بیداری سیستم
			ریست واتچ داگ
				→ System Task
				→ App Task

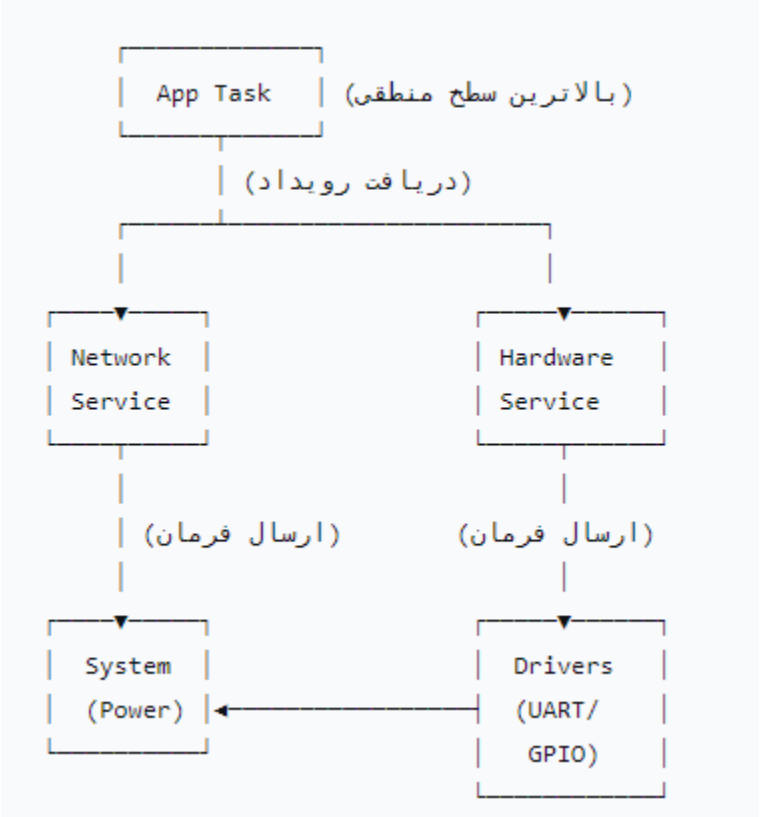
App Main	`app/app_main.c`	`EVENT_STATE_CHANGED`
		`EVENT_SYSTEM_ERROR`
		`EVENT_DATA_READY`
			تغییر وضعیت
			خطای سیستم
			داده آماده پردازش
				→ (خودش)

3. ماژول‌های Event Subscribe (ارسال‌کننده رویداد)

```
app/app_main.c  →  تنها دریافت‌کننده اصلی رویدادها
```

4. Dependency بین سرویس ها (جدول وابستگی)

جدول وابستگی ساده:



جدول وابستگی کامل:

جدول وابستگی کامل (برای EC200U):

سرویس (فایل)	وابسته به (Dependency)	نوع وابستگی	روش ارتباط	توضیح
network_service.c	`drivers/uart/uart_driver.h`	قوی (Strong)	تابع Call	نیاز به UART برای PPP/4G
	`system/power_manager.h`	متوسط (Medium)	Event	اطلاع از وضعیت خواب/بیداری
	`core/event_bus.h`	ضعیف (Weak)	Event Publish	ارسال وضعیت شبکه
	`core/log.h`	ضعیف (Weak)	تابع Call	ثبت خطاها
hardware_service.c	`drivers/uart/uart_driver.h`	قوی (Strong)	تابع Call	خواندن GPS از UART
	`drivers/gpio/gpio_driver.h`	قوی (Strong)	تابع Call	خواندن سنسورها/دکمه
	`core/event_bus.h`	متوسط (Medium)	Event Publish	ارسال داده GPS
	`core/timers.h`	ضعیف (Weak)	تابع Call	تایم‌اوت‌ها
	`core/log.h`	ضعیف (Weak)	تابع Call	ثبت خطاها
power_manager.c	`drivers/gpio/gpio_driver.h`	قوی (Strong)	تابع Call	کنترل پین‌های Power
	`core/event_bus.h`	ضعیف (Weak)	Event Publish	اعلام خواب/بیداری
	`core/timers.h`	متوسط (Medium)	تابع Call	تایمر بیداری
app_main.c	`core/event_bus.h`	قوی (Strong)	Event Subscribe	دریافت همه رویدادها
	`services/network_service.h`	متوسط (Medium)	فرمان از طریق Queue	ارسال فرمان (MQTT/HTTP)
	`services/hardware_service.h`	متوسط (Medium)	فرمان از طریق Queue	فعال/غیرفعال کردن GPS
	`system/power_manager.h`	ضعیف (Weak)	تابع Call	درخواست Sleep
	`core/timers.h`	ضعیف (Weak)	تابع Call	تایمرهای برنامه

5. ممنوعیت‌های وابستگی (Dependency Rules)

```
// ❌ این وابستگی‌ها ممنوع است:

// 1. Services نباید به App وابسته باشد
network_service.c  →  app_main.h  // ممنوع

// 2. Drivers نباید به هیچ ماژول دیگری غیر از SDK وابسته باشد
gpio_driver.c  →  event_bus.h    // ممنوع
uart_driver.c  →  network_service.h  // ممنوع

// 3. Core نباید به Services وابسته باشد
event_bus.c  →  network_service.h  // ممنوع

// 4. System نباید به Services وابسته باشد
power_manager.c  →  network_service.h  // ممنوع

// 5. وابستگی حلقوی (بخصوص بین Network و Hardware)
network_service.c  →  hardware_service.h  // ممنوع
hardware_service.c  →  network_service.h  // ممنوع
```

6. روش صحیح ارتباط بین سرویس‌ها

```
// ✅ روش صحیح: ارتباط از طریق Event Bus و Queue

// 1. Hardware Service ارسال داده GPS
// services/hardware_service.c
gps_data_t gps = { .lat = 35.68, .lon = 51.42 };
event_publish(EVENT_GPS_FIX, &gps);  // ارسال به Event Bus

// 2. App Task دریافت رویداد
// app/app_main.c
static void on_gps_fix(void *data) {
    gps_data_t *gps = (gps_data_t*)data;
    process_gps_data(gps);
}
event_subscribe(EVENT_GPS_FIX, on_gps_fix);

// 3. App Task ارسال فرمان به Network Service
// app/app_main.c
network_cmd_t cmd = {
    .type = CMD_MQTT_PUBLISH,
    .topic = "sensor/data",
    .payload = &sensor_data
};
ql_rtos_queue_send(network_queue, &cmd, 0);

// 4. Network Service دریافت فرمان
// services/network_service.c
ql_rtos_queue_recv(network_queue, &cmd, OS_WAIT_FOREVER);
if (cmd.type == CMD_MQTT_PUBLISH) {
    mqtt_publish(cmd.topic, cmd.payload);
}
```

7. خلاصه نهایی

نوع	تعداد	لیست
Tasks	4	`app_task`, `network_task`, `hardware_task`, `system_task`
Event Publishers	4	Hardware, Network, Power Manager, App
Event Subscribers	1	فقط App Task (مرکزی)
Strong Dependencies	3	Network→UART, Hardware→UART/GPIO, Power→GPIO
Weak Dependencies	5	از طریق Event Bus و Queue

توضیحات مختصر هر فولدر و فایل

پوشه `include/user_code/` (هدرهای عمومی پروژه)

فایل	توضیح مختصر
`user_code.h`	هدر جامع که همه هدرهای دیگر را شامل می‌شود (اختیاری)
`app_api.h`	توابع عمومی که برنامه در اختیار ماژول‌های دیگر قرار می‌دهد
`event_bus.h`	توابع مربوط به event bus: `event_publish()`, `event_subscribe()`
`event_ids.h`	تعریف تمام شناسه‌های رویدادها (`EVENT_GPS_FIX`, `EVENT_NET_UP`, ...)
`module_ids.h`	تعریف شناسه ماژول‌ها برای لاگ و دیباگ (`MODULE_APP`, `MODULE_NETWORK`, ...)
`drivers_public.h`	هدر یکپارچه برای دسترسی به تمام درایورها
`network_service.h`	توابع عمومی سرویس شبکه: `network_init()`, `mqtt_publish()`, ...
`hardware_service.h`	توابع عمومی سرویس سخت‌افزار: `gnss_start()`, `gnss_stop()`, ...
`app_events.h`	تعریف رویدادهای اختصاصی برنامه و تابع‌های پردازش

پوشه `app/` (لایه برنامه کاربردی - سطح 4)

فایل	توضیح مختصر	دارد؟ Task
`app_main.c`	**ورودی اصلی برنامه**. راه‌اندازی سیستم، ایجاد تسک‌ها، حلقه اصلی	✅ `app_task`
`app_main.h`	اعلان توابع اصلی برنامه	❌
`app_state_machine.c`	**ماشین حالت برنامه**. مدیریت وضعیت‌ها: `STATE_INIT`, `STATE_RUNNING`, `STATE_SLEEP`, `STATE_ERROR`	❌
`app_state_machine.h`	اعلان توابع ماشین حالت	❌
`app_events.c`	**پردازشگر رویدادها**. تابع `process_event()` که رویدادهای دریافتی را پردازش می‌کند	❌

پوشه `services/` (لایه سرویس‌ها - سطح 3)

فایل	توضیح مختصر	دارد؟ Task
`network_service.c`	**سرویس یکپارچه شبکه**. مدیریت: 4G/LTE، MQTT، HTTP، OTA. شامل `network_task`	✅ `network_task`
`network_service.h`	اعلان توابع عمومی سرویس شبکه	❌
`network_internal.h`	توابع داخلی سرویس شبکه (برای استفاده داخل ماژول)	❌
`hardware_service.c`	**سرویس یکپارچه سخت‌افزار**: GNSS/GPS، UART، GPIO، Device Manager. مدیریت	✅ `hardware_task`
`hardware_service.h`	اعلان توابع عمومی سرویس سخت‌افزار	❌
`hardware_internal.h`	توابع داخلی سرویس سخت‌افزار (پارسر GPS،管理等)	❌

پوشه `core/` (هسته اصلی - سطح 2)

فایل	توضیح مختصر	Task دارد؟
`event_bus/`		
`event_bus.c`	**سیستم انتشار/اشتراک رویداد.** پیاده‌سازی `publish/subscribe` با استفاده از Queue	❌
`event_bus.h`	اعلان توابع event bus	❌
`event_bus_internal.h`	ساختارهای داخلی event bus (لیست مشترکین، صف رویدادها)	❌
`utils/`		
`utils.c`	توابع کمکی عمومی: `string_to_int()`، `int_to_string()`، `memcpy_safe()`	❌
`utils.h`	اعلان توابع کمکی	❌
`ring_buffer.c`	**بافر حلقوی (FIFO).** برای ذخیره موقت داده‌های GPS و UART	❌
`ring_buffer.h`	اعلان توابع بافر حلقوی	❌
`crc.c`	**محاسبات CRC.** CRC16، CRC32 برای بررسی یکپارچگی داده	❌
`crc.h`	اعلان توابع CRC	❌
سایر		
`log.c`	**سیستم لاگینگ.** لاگ با سطوح: ERROR, WARN, INFO, DEBUG. خروجی روی UART	❌
`log.h`	اعلان توابع لاگ (`LOG_ERROR`، `LOG_INFO`، ...)	❌
`timers.c`	**سیستم تایمر.** تایمرهای نرم‌افزاری یکبارمصرف و دوره‌ای	❌
`timers.h`	اعلان توابع تایمر (`timer_start()`، `timer_stop()`، `timer_expired()`)	❌

پوشه `system/` (لایه سیستمی - سطح 2)

فایل	توضیح مختصر	Task دارد؟
`sys_init.c`	**مقداردهی اولیه سیستم**: ترتیب راه‌اندازی GPIO → UART → Event Bus → Timers → Services	❌
`sys_init.h`	اعلان توابع مقداردهی (`system_early_init()`, `system_late_init()`)	❌
`sys_config.c`	**پارامترهای کانفیگ سیستم**: سرعت UART، پین‌های GPIO، آدرس سرور MQTT	❌
`sys_config.h`	اعلان متغیرهای کانفیگ (`UART_BAUD_RATE`, `MQTT_BROKER_URL`)	❌
`rtos_wrapper.c`	**لایه انتزاعی RTOS**: توابع wrapper برای ایجاد تسک، Queue، Semaphore (مختلف RTOS قابلیت پورت به)	❌
`rtos_wrapper.h`	اعلان توابع RTOS wrapper	❌
`power_manager.c`	**مدیریت مصرف انرژی**: کنترل Sleep/Wake، مانیتورینگ باتری، Watchdog	✅ `system_task`
`power_manager.h`	اعلان توابع مدیریت انرژی (`system_sleep()`, `system_wakeup()`, `get_battery_level()`)	❌

پوشه `drivers/` (لایه درایورها - سطح 1 - پایین‌ترین سطح)

فایل	توضیح مختصر	دارد؟ Task
`uart/`		
`uart_driver.c`	**درایور UART.** توابع: `uart_init()` , `uart_send()` . مدیریت بافر و DMA `uart_receive()`	❌
`uart_driver.h`	اعلان توابع UART (با پشتیبانی از UART1, UART2, UART3)	❌
`gpio/`		
`gpio_driver.c`	**درایور GPIO.** توابع: `gpio_init()` , `gpio_set()` , `gpio_get()` , `gpio_interrupt_enable()`	❌
`gpio_driver.h`	اعلان توابع GPIO (پشتیبانی از 32 پین، حالت‌های INPUT/OUTPUT/INT)	❌

فایل ریشه (Root)

فایل	توضیح مختصر
`CMakeLists.txt`	**اسکریپت کامپایل.** تعریف فایل‌های منبع، مسیرهای هدر، کتابخانه‌های SDK، گزینه‌های کامپایلر

جدول خلاصه بر اساس مسئولیت

مسئولیت	فایل‌ها
مقداردهی اولیه	`sys_init.c`, `sys_config.c`
مدیریت انرژی	`power_manager.c`
ارتباطات شبکه	`network_service.c` (MQTT, HTTP, OTA)
داده‌های سخت‌افزاری	`hardware_service.c` (GPS, UART, GPIO, دستگاه‌ها)
منطق برنامه	`app_main.c`, `app_state_machine.c`, `app_events.c`
ارتباط داخل سیستمی	`event_bus.c`
لاگ و خطایابی	`log.c`
زمان‌بندی	`timers.c`
توابع کمکی	`utils.c`, `ring_buffer.c`, `crc.c`
لایه انتزاعی RTOS	`rtos_wrapper.c`
دسترسی به سخت‌افزار	`uart_driver.c`, `gpio_driver.c`

ترتیب فراخوانی در زمان بوت

text

Copy Download

```
1. system/sys_init.c      → system_early_init()
   ├── drivers/gpio/gpio_driver.c    → gpio_init()
   ├── drivers/uart/uart_driver.c    → uart_init()
   └── core/log.c                    → log_init()

2. system/sys_init.c      → system_late_init()
   ├── core/event_bus/event_bus.c     → event_bus_init()
   ├── core/timers.c                  → timers_init()
   ├── services/hardware_service.c    → hardware_service_init()
   ├── services/network_service.c     → network_service_init()
   └── system/power_manager.c         → power_manager_init()

3. app/app_main.c         → app_task ایجاد میشود
   └── app/app_state_machine.c        → state_machine_start()
```

توابعی که فقط یکبار در زمان بالا اومدن دستگاه بیاد اجرا بشن و نیازی به اجرای مداوم آنها نیست در RTOS کجا باید باشند:

- به عنوان تابع معمولی (قبل از ایجاد تسک‌ها) اجرا شوند. در فایل های آموزشی قبلی توضیح دادم که کدام تابع از sdk وظیفه شروع تسک ها (tasks) را به عهده دارد، قبل از اجرا تسک های خودمون ، توابعی که نیاز به یکبار اجرا دارند را اجرا و بعد تسک های خودمون رو شروع میکنیم.