Tethering USB Sony α7R III ↔ iPad

Documentazione tecnica di un progetto di reverse engineering: problemi incontrati, strade scartate, protocollo verificato

Federico Verondini — fedevero.it | 1 agosto 2026 | Documento didattico, in aggiornamento

---

Questo documento raccoglie il percorso tecnico seguito per collegare una fotocamera Sony α7R III (ILCE-7RM3) direttamente a un iPad via USB-C, senza passare da un computer intermedio, con l'obiettivo di ricevere ogni scatto istantaneamente in un'app di sviluppo fotografico personalizzata. È pensato come materiale didattico per chiunque affronti un problema simile: cosa non ha funzionato, perché, e cosa invece è stato verificato con dati reali.

1. Obiettivo del progetto

Costruire un'app nativa per iPad che si colleghi via USB-C direttamente alla fotocamera, permetta di scattare (da app o dal pulsante fisico), riceva la foto sull'iPad nell'istante dello scatto, e la apra immediatamente in un modulo di sviluppo con controlli personalizzati e anteprima in tempo reale. Vincolo esplicito: nessun computer intermedio durante l'uso reale (l'iPad deve parlare da solo con la camera).

2. Soluzioni valutate e scartate

Prima di arrivare al reverse engineering, sono state esplorate le vie ufficiali e commerciali disponibili. Nessuna soddisfaceva contemporaneamente i tre vincoli: nessun computer intermedio, velocità adeguata per file RAW, e possibilità di integrare un'interfaccia di editing personalizzata.

Soluzione	Esito	Motivo
Sony Creators' App (Wi-Fi ufficiale)	Scartata	Non supporta il modello ILCE-7RM3 nella lista camere compatibili.
Sony Camera Remote SDK (ufficiale)	Scartata	Gira solo su Windows / macOS / Linux. Nessun supporto iPadOS.
Sony Camera Remote Command (protocollo PTP gratuito)	Scartata	Riservato a clienti aziendali (serve P.IVA); la lista modelli include ILCE-7RM3A ma non ILCE-7RM3 originale.
Capture One per iPad	Scartata	Soluzione commerciale valida ma a pagamento, con interfaccia non personalizzabile.
Shutter+ (shutter.dev)	Tenuta come fallback	Confermata compatibile con α7R III, economica (65€ una tantum), ma non integrabile con un editor custom.
Tethering Wi-Fi custom (PTP-IP)	Scartata	Tecnicamente fattibile ma trasferimento RAW troppo lento (~40s/foto): la camera supporta solo banda 2.4GHz.
USB-C + protocollo reverse-ingegnerizzato	Scelta adottata	Unica via che rispetta tutti i vincoli. Richiede scoprire il protocollo proprietario Sony.

Nota su modello camera: «α7R III» nel commercio corrisponde al codice interno ILCE-7RM3 (2017). Non va confuso con la ILCE-7RM3A (α7R IIIA, 2021), che è l'unica delle due coperta dal programma corporate Sony.

3. L'approccio scelto: USB + ImageCaptureCore

iPadOS espone ImageCaptureCore, un framework pubblico Apple (disponibile da iOS 13) che permette a un'app di terze parti di rilevare una fotocamera collegata via USB e inviarle comandi PTP grezzi tramite il metodo `ICCameraDevice.requestSendPTPCommand(_:outData:completion:)`. Non è un'ipotesi: app come Shutter+, CamRanger e onOne DSLR Camera Remote lo usano già in produzione per controllare camere Sony, Canon e Fujifilm da iPad.

Il framework fornisce il canale di comunicazione, ma non conosce i comandi specifici Sony: quelli sono proprietari e non documentati pubblicamente per il modello in questione (vedi sezione 2). Serviva quindi scoprire il linguaggio esatto che la camera si aspetta.

Nota tecnica: `requestSendPTPCommand` *richiede la chiave* `NSCameraUsageDescription` *in Info.plist, assente la quale iPadOS blocca ogni comando PTP con un errore di autorizzazione.*

3.1 Fonti community consultate per un primo orientamento

• SonyAlphaUSB (github.com/pixeltris/SonyAlphaUSB) — testato esplicitamente su Sony A7 III, stessa generazione della α7R III. Fonte degli opcode iniziali.

• libgphoto2 (camlibs/ptp2) — libreria open source con supporto Sony parziale, reverse-ingegnerizzato nel tempo dalla community gphoto.

• Alpha-fairy (github.com/frank26080115/alpha-fairy) — telecomando Wi-Fi open source per camere Sony, con documentazione dettagliata del processo di reverse engineering via Wireshark.

Queste fonti hanno dato un punto di partenza plausibile ma non certezze: gli opcode e il formato esatto variano leggermente tra modelli e generazioni di firmware. Da qui la necessità di una verifica diretta sul modello specifico.

4. Metodologia: cattura del traffico USB reale

Invece di procedere per tentativi sulla camera reale (rischioso e lento), si è scelto di intercettare il traffico USB generato dal software ufficiale Sony (Imaging Edge Desktop, modulo Remote) mentre comunica con la camera durante uno scatto remoto. Il traffico osservato è per definizione corretto, perché generato dall'app ufficiale.

4.1 Setup (macOS)

• Disattivazione temporanea del SIP (System Integrity Protection): avvio in Recovery Mode, Terminale → `csrutil disable`, riavvio. Necessario perché macOS blocca la cattura USB a livello kernel di default.

• Attivazione dell'interfaccia di cattura USB: `sudo ifconfig XHC20 up` (nome dell'interfaccia legato al controller USB, individuato con `ifconfig -a`).

• Wireshark avviato in cattura sull'interfaccia USB individuata, camera collegata via cavo USB-C (non Micro-USB — la α7R III ha entrambe le porte, solo la USB-C è pensata per il tethering).

• Sulla camera: MENU → Rete → Ctrl w/ Smartphone → Off (altrimenti interferisce con la modalità PC Remote); poi MENU → Setup → USB Connection → PC Remote.

• Scatto eseguito dal pulsante dentro Imaging Edge Desktop mentre Wireshark registra; cattura salvata in formato .pcapng.

• SIP riattivato (`csrutil enable`) a cattura conclusa.

4.2 Analisi della cattura

Il file catturato (~104MB, formato DLT_USB_DARWIN, oltre 37.000 pacchetti) è stato analizzato programmaticamente in Python con la libreria `dpkt` (tshark/Wireshark CLI non installabili nell'ambiente di analisi usato). Lo script ha scansionato ogni pacchetto alla ricerca di blocchi PTP riconoscibili, confrontando ogni possibile offset di byte con la struttura attesa di un container PTP standard.

```
# logica di scansione semplificata
for offset in range(len(packet)):
tipo = uint16_le(packet, offset+4) # 1=comando 2=dati 3=risposta
if tipo in (1, 2, 3):
lunghezza = uint32_le(packet, offset)
codice = uint16_le(packet, offset+6)
txn_id = uint32_le(packet, offset+8)
payload = packet[offset+12 : offset+lunghezza]
```

5. Risultati confermati dalla cattura reale

Questi dati non sono ipotesi: sono stati letti byte per byte dal traffico reale tra Imaging Edge Desktop e la α7R III.

5.1 Formato del container PTP

Confermato l'header standard ISO15740/PIMA a 12 byte, little-endian:

Campo	Dimensione	Note
Lunghezza totale	`4 byte`	Include l'header stesso
Tipo blocco	`2 byte`	1=Comando, 2=Dati, 3=Risposta
Opcode / codice	`2 byte`	Operazione o codice risposta
Transaction ID	`4 byte`	Incrementale, deve combaciare tra comando/dati/risposta
Payload	`variabile`	Parametri (comando) o valore (dati)

Risposta di successo standard osservata: codice 0x2001 (PTP_RC_OK).

5.2 Scoperta chiave: opcode dello scatto

Ipotesi iniziale (da SonyAlphaUSB): lo scatto/AF avrebbero usato l'opcode 0x9205 ("SubSetting"). Smentita dalla cattura: su questa α7R III, sia lo scatto sia l'AF passano per l'opcode 0x9207, lo stesso usato per le altre impostazioni camera. Inoltre ogni comando è risultato essere a due fasi distinte, non una sola come inizialmente ipotizzato: un blocco Comando contenente solo il codice della proprietà da modificare, seguito da un blocco Dati separato (stesso opcode e transaction ID) contenente il nuovo valore.

5.3 Sequenza esatta di uno scatto completo

#	Proprietà	Valore	Significato
1	`0xD2C1 (HalfPressShutter)`	2	Premi — avvia autofocus
2	`0xD2C7 (proprietà associata)`	2	"Arm" collegato all'AF (non documentata da Sony)
–	–	–	~140ms di autofocus (polling live view in corso)
3	`0xD2C2 (CapturePhoto)`	1	Scatto vero
4	`0xD2C1 (HalfPressShutter)`	1	Rilascio

Dopo il passo 4, la camera genera un evento seguito da `GetObjectInfo` / `GetObject` (opcode standard PTP 0x1008 / 0x1009) per recuperare il file nuovo. Nella cattura osservata: file `DSC09999.JPG`, formato PTP 0x3801 (EXIF/JPEG). Il nome file si legge da una stringa UTF-16LE con prefisso di lunghezza dentro la risposta di GetObjectInfo.

Non osservato nella finestra catturata: un handshake esplicito "Connect" (opcode 0x9201). Il traffico Sony (polling 0x9209) risultava già attivo dal secondo pacchetto della cattura — non è chiaro se l'handshake sia avvenuto prima dell'inizio della cattura o se non sia più necessario con OpenSession PTP standard (0x1002).

6. Punti ancora aperti

Un solo punto tecnico resta da verificare direttamente su iPad reale: come `ImageCaptureCore` gestisce le due fasi (Comando + Dati) di un singolo comando `requestSendPTPCommand`. Due ipotesi in campo:

- Il parametro `outData` viene interpretato automaticamente come la fase Dati da inviare subito dopo il Comando (approccio attualmente implementato nel codice).
- Serve invocare il metodo due volte in sequenza: una per il blocco Comando, una per il blocco Dati costruito manualmente con lo stesso schema del container.

Tutto il resto del protocollo (opcode, sequenza, valori) è verificato empiricamente e non richiede ulteriori ipotesi.

7. Lezioni per chi affronta un problema simile

- Verificare sempre il modello esatto della camera, non solo la famiglia. Sigle molto simili (ILCE-7RM3 vs ILCE-7RM3A) possono avere copertura completamente diversa nei programmi ufficiali.
- Le fonti community sono un punto di partenza, non una certezza. Gli opcode differiscono anche tra modelli della stessa generazione: quello che vale per una A7 III non è garantito identico su una α7R III.
- Il modo più affidabile per scoprire un protocollo proprietario è intercettare il software ufficiale che già lo parla, non indovinare comandi sulla camera reale.
- Wi-Fi e USB non sono intercambiabili solo per comodità: stesso protocollo logico, ma limiti fisici (banda, velocità) possono rendere una delle due strade impraticabile a prescindere dalla correttezza del codice.
- Verificare le API della piattaforma prima di assumere che qualcosa sia impossibile. iPadOS sembrava non offrire alcuna via per parlare con una DSLR via USB; in realtà `ImageCaptureCore` lo permette da anni, semplicemente non è ovvio né ben pubblicizzato.

8. Fonti e riferimenti

- SonyAlphaUSB — github.com/pixeltris/SonyAlphaUSB
- libgphoto2, camlibs/ptp2 — github.com/gphoto/libgphoto2
- alpha-fairy — github.com/frank26080115/alpha-fairy
- Apple Developer Documentation — ImageCaptureCore / ICCameraDevice
- Sony Camera Remote SDK & Camera Remote Command — support.d-imaging.sony.co.jp
- Shutter+ — shutter.dev (soluzione commerciale di riferimento/fallback)

Documento in aggiornamento. La versione corrente riflette lo stato del progetto al 1 agosto 2026, dopo la prima cattura USB riuscita. La prossima revisione includerà l'esito del test su iPad fisico.