📄 Wildlife Camera Trap Maker Kit — Design Brief (v0.1)

🐾 Product Name (TBD)

Autonomous Wildlife Camera Trap Kit

🎯 Product Vision

A **fully autonomous**, **field-deployable**, motion-activated wildlife camera trap designed for conservationists, researchers, and nature enthusiasts. It operates entirely **offline**, supports **mobile proximity-based offloading**, and requires **no user interaction** in the field beyond walking near it.

Designed for:

* Remote, off-grid locations
* Long-term wildlife monitoring
* Scientific data collection
* Low disturbance, low maintenance deployments

🧭 Design Philosophy

* **No human disturbance**: No physical contact required after deployment
* **No network dependence**: Never connects to the internet or cloud
* **Zero UI field operation**: Haptic-only mobile feedback; no on-device screens or LEDs
* **Modular and fault-tolerant**: Power systems (battery/solar) and file handling are designed to preserve data above all
* **Autonomous, energy-conscious design**: Runs in deep sleep, wakes on motion or proximity

---

📋 Feature Prioritization (MoSCoW Method)

Feature	Priority	Notes
Motion-triggered capture	**Must Have**	Core trap behavior
SD card storage	**Must Have**	Local media archive
Battery power	**Must Have**	Baseline power source
Weatherproofing	**Must Have**	Outdoor operation
Mobile app (offline transfer + config)	**Must Have**	Field access + control
GPS tagging (via phone)	**Must Have**	Accurate observation data
IR night vision	**Should Have**	Important but not blocking MVP
Solar charging	**Should Have**	Power extender, not critical
Web app	**Won’t Have**	No network dependency allowed
Image classification (cloud/AI)	**Won’t Have (unless local)**	Local-only someday
Livestreaming	**Won’t Have**	Incompatible with autonomy goals

---

🔋 Power Strategy

* **Primary:** Li-ion battery (internal or external)
* **Optional secondary:** Solar trickle charging
* **Management:**

  * Battery voltage monitoring (fuel gauge or ADC)
  * Configurable behavior profiles based on battery level
  * Wi-Fi disabled at low power
* **Failsafe principle:** Camera prioritizes saving captured media; powers down modules selectively

---

📶 Offload + Interaction Strategy

Discovery:

* Phone passively scans via **BLE**
* Camera advertises presence periodically (e.g. every 30s)

Data Transfer Modes:

* **BLE (low-power)**: Default; slower
* **Wi-Fi (faster)**: Optional; enabled via config
* **Hybrid**: BLE triggers proximity + transfer decision; device activates Wi-Fi if battery allows

Haptic Feedback:

* 1 vibration = offload started
* 2 vibrations = offload complete
* 3 vibrations = error

Config Sync:

* Settings set in phone app → sent automatically upon proximity

---

📱 Mobile App Responsibilities

* Configure camera settings (resolution, IR, power mode, offload mode, etc.)
* Detect camera automatically when in range
* Push settings + fetch media without user input
* Tag each file with phone GPS + time
* Store and organize photos/videos
* Work offline 100% (no network dependencies)

🧰 Hardware Architecture (Draft)

Component	Choice
MCU	**ESP32-S3**
Camera	**OV2640** (initially)
Storage	MicroSD card
Power	Li-ion battery + optional solar (via TP4056 or similar)
Motion detection	PIR sensor
Enclosure	Weatherproof, IR-transparent window
Comms	BLE + Wi-Fi (softAP or direct)
Power Monitoring	Voltage divider or I2C fuel gauge

📌 Design Decisions Made

✅ Use **BLE-first** with optional **Wi-Fi handoff** for faster transfers
✅ GPS tagging will happen via **phone** at time of offload
✅ App will be the **only configuration surface** — no buttons, screens, or cloud
✅ Adopt **deep sleep with wake-on-timer or motion**
✅ Use haptics-only feedback (no visual or sound signals)
✅ Modular power design with solar as optional secondary source
✅ ESP32-S3 selected as MCU platform

🔧 Open Questions / Future Work

* Final camera sensor selection (OV5640 or thermal cam later?)
* Mounting system and anti-tamper mechanisms
* Advanced power tracking (e.g. max power point tracking solar controller?)
* Config schema storage (on SD or in EEPROM?)
* Firmware OTA (only via SD card? or app-assisted?)
* Researcher dashboard app (non-headless mode)

💾 Save + Export

You can now:

* Copy/paste this into a local file (Markdown, DOCX, or PDF)
* Re-import it into ChatGPT anytime to resume
* Build off it in any documentation tool

Would you like me to:

* Format this as a downloadable **Markdown**, **PDF**, or **Google Doc**?
* Email-style text export?
* Help create a Git repo or Notion doc for this?

Let me know how you'd like to store or work with it next.