StillCold — Week 6 Progress

BLE Integration — Live environmental data over Bluetooth

A brief update on where the project stands

What Week 6 Was About

Goal: Complete the BLE exposure milestone for Sprint 1 by making live environmental data accessible over Bluetooth Low Energy.

- Expose temperature and humidity readings through a BLE service.
- Make the data readable by nearby devices (like phones with nRF Connect).
- Ensure the system remains stable and discoverable after connections.
- Validate that BLE values match the internal sensor readings exactly.

This week completed the final link in the Sprint 1 data pipeline: Sensor → Arduino Nano → ESP32-C6 → BLE → External Device

What Was Achieved

By the end of Week 6:

- **BLE server is operational** — The ESP32-C6 advertises as "StillCold" and remains discoverable.
- **Custom BLE service implemented** — A dedicated service with unique UUID exposes environmental data.
- **Two readable characteristics** — Temperature and humidity are both accessible as separate BLE characteristics.
- **Real-time data synchronization** — BLE values update immediately when new sensor readings arrive.
- **Stable reconnection behavior** — The device automatically resumes advertising after disconnection, supporting multiple connect/disconnect cycles.

Week 6 Photos

In the same folder as this presentation file, you'll find images which are screenshots from the nRF Connect app from my personal client, showing each step in my progress

Value under Unknown Characteristic shows the transmitted value (in this case temperature in C & humidity)

The Complete Data Pipeline

The StillCold system now operates end-to-end:

1. **HTU21D sensor** measures temperature and humidity.
2. **Arduino Nano** formats readings as structured text: `T=18.59,H=39.22`
3. **ESP32-C6** receives and parses the UART data.
4. **BLE service** exposes the latest values through readable characteristics.
5. **External device** (nRF Connect or future Flutter app) can read live data.

All steps happen automatically, creating a seamless flow from sensor measurement to wireless access.

BLE Architecture Details

Device Configuration:

- BLE Device Name: `StillCold`
- Custom Service UUID: `12345678-1234-1234-1234-1234567890ab`

Characteristics:

- **Temperature** (UUID: `abcd1234-5678-1234-5678-abcdef123456`)
  - Returns latest temperature as string (e.g., "18.59")
  - Updates immediately after successful UART parse
- **Humidity** (UUID: `abcd5678-1234-5678-1234-abcdef654321`)
  - Returns latest humidity as string (e.g., "39.22")
  - Updates immediately after successful UART parse

How BLE Lifecycle Works

The system handles BLE connections gracefully:

On Connect:

- Device sets connection state flag.
- Logs connection event to Serial Monitor.
- Characteristics remain readable throughout the connection.

On Disconnect:

- Device automatically restarts advertising.
- Remains discoverable immediately after disconnect.
- No manual reset required.

This ensures the device never becomes "invisible" after a client disconnects, supporting reliable reconnection.

Validation Testing Performed

The following tests confirmed system reliability:

- **Advertising Stability** — Device advertises continuously and reappears after power cycle.
- **Characteristic Validation** — Temperature and humidity both readable; values match Serial output exactly.
- **Reconnect Stress Test** — 10+ connect/disconnect cycles performed without crashes or resets.
- **Extended Idle Test** — Device left running for extended period; successful connection after idle with accurate values.
- **Environmental Variation Test** — Sensor warmed/cooled; both Serial and BLE reflected changes correctly.

All tests passed, confirming the system is stable and ready for mobile app integration.

Assumptions We Validated

- **BLE can expose live sensor data reliably** — The ESP32-C6 successfully updates characteristic values in real-time as new readings arrive.
- **Two-characteristic design works** — Separating temperature and humidity into distinct characteristics provides clear, focused access to each value.
- **Simple text format is sufficient** — String values with 2 decimal precision are readable and match internal storage format.
- **Advertising lifecycle can be automated** — Custom callback handlers ensure the device remains discoverable without manual intervention.

What We Corrected or Clarified

- **Advertising must restart after disconnect** — Without explicit restart logic, the device would become invisible after a client disconnected. A custom `BLEServerCallbacks` class ensures advertising resumes automatically.
- **Characteristic values must update dynamically** — Values are updated via `setValue()` immediately after UART parsing, ensuring BLE clients always see the latest readings.
- **Connection state tracking is important** — A `deviceConnected` flag helps manage advertising lifecycle and prevents conflicts during active connections.

These implementation details ensure reliable, production-ready BLE behavior.

Ready for the Next Step

Current state:

The StillCold prototype now has a **complete end-to-end data pipeline**:

Sensor → Sensing board → UART transfer → Communication board → BLE exposure → External device

- Data flows from sensor to BLE reliably.
- Values are accessible wirelessly without opening the monitored environment.
- System architecture remains modular and maintainable.
- No Wi-Fi or external infrastructure required.

Next:

Begin Flutter BLE client groundwork (Sprint 1 Week 4) to create a mobile application that can connect to StillCold and display live environmental data.

Summary

Area	Status
BLE server implementation	Complete; device advertises as "StillCold"
Custom service & characteristics	Implemented; temperature and humidity both readable
Real-time data synchronization	Working; BLE values update immediately with sensor readings
Connection lifecycle	Stable; automatic advertising restart after disconnect
End-to-end validation	Verified; BLE values match Serial output exactly
Sprint 1 BLE milestone	**COMPLETE**

StillCold now operates as a functional embedded environmental monitoring prototype with wireless data access.

Thank you

StillCold — *Environmental monitoring without opening the door*

Questions?