CS131: Internet of Things

Phase 2: Initial Deployment Report

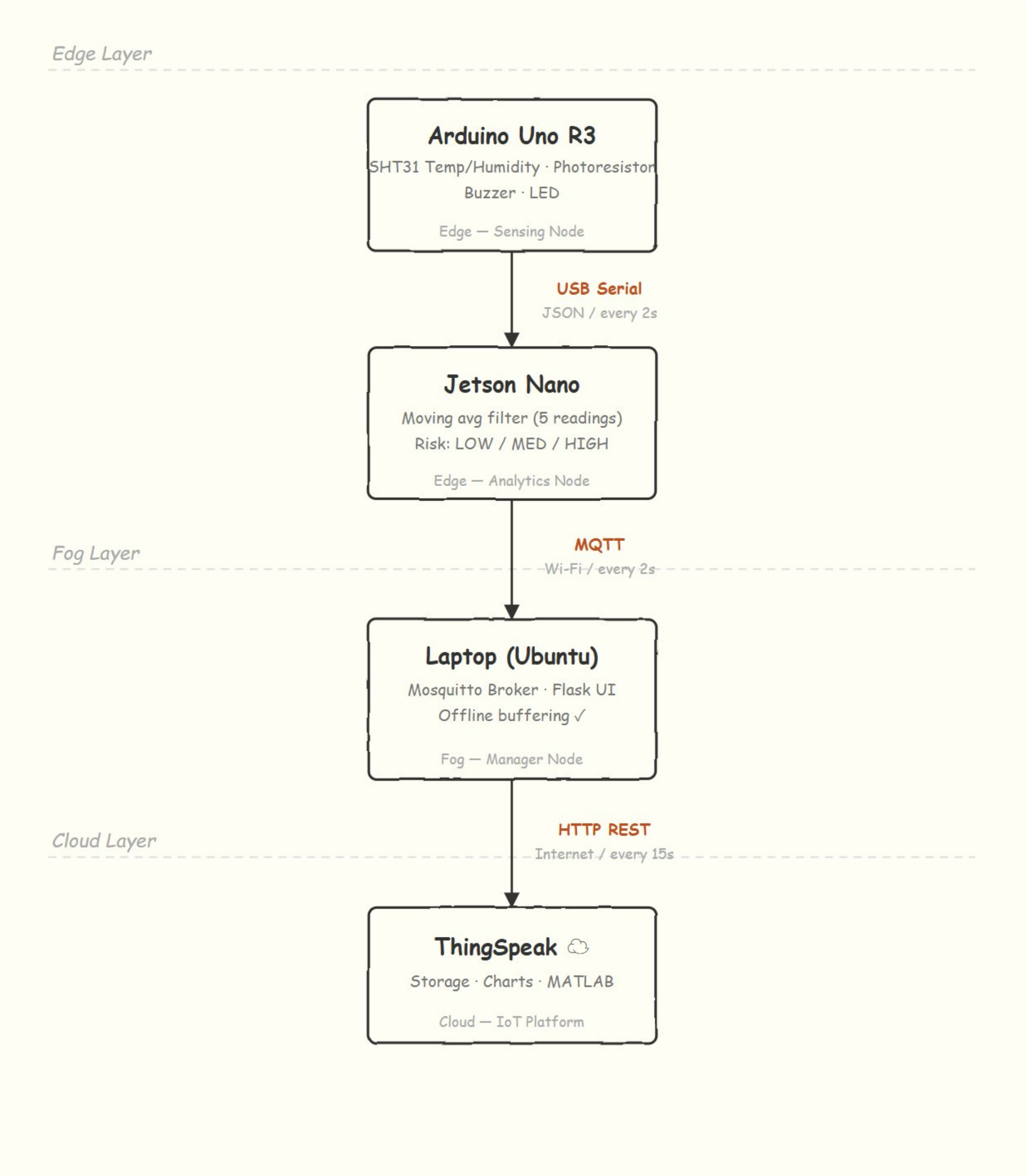
Cold-Chain / Medical Storage Monitoring System

Team 21

Ryan Yang (ryang097)

Pengzhen Lin (ID: 862425101)

GitHub: https://github.com/Linppz/CS131_Project


Edge Layer
Arduino Uno R3
SHT31 Temp/Humidity · Photoresistor
Buzzer · LED
Edge — Sensing Node
USB Serial
JSON / every 2s
Jetson Nano
Moving avg filter (5 readings)
Risk: LOW / MED / HIGH
Edge — Analytics Node
Fog Layer
MQTT
Wi-Fi / every 2s
Laptop (Ubuntu)
Mosquitto Broker · Flask UI
Offline buffering ✓
Fog — Manager Node
Cloud Layer
HTTP REST
Internet / every 15s
ThingSpeak ☁
Storage · Charts · MATLAB
Cloud — IoT Platform

1. Network Diagram

Our system uses a three-layer architecture: Edge, Fog, and Cloud. The diagram below shows how each device is connected and what it does.

Layer	Device	Connects To	What It Does
Edge (Sensing)	Arduino Uno R3	Jetson Nano via USB cable	Reads temperature, humidity, and light every 2 seconds
Edge (Analytics)	Jetson Nano	Laptop via Wi-Fi	Analyzes sensor data and calculates risk level
Fog	Laptop	ThingSpeak via Internet	Shows dashboard, stores data locally if offline
Cloud	ThingSpeak	Accessed from Fog via HTTP	Stores history, shows graphs online

How devices are connected:

- **Arduino → Jetson Nano:** Connected with a USB cable. Arduino sends sensor readings as text messages through this cable every 2 seconds.
- **Jetson Nano → Laptop:** Both connected to the same Wi-Fi network. Jetson sends processed data to the laptop using MQTT, a lightweight messaging system commonly used in IoT.
- **Laptop → ThingSpeak:** The laptop uploads data to ThingSpeak (a free IoT cloud platform) over the internet using simple HTTP requests.

2. Device Catalog

Below is a list of all hardware used in our system.

Device	Role	Key Specs	Sensors / Peripherals
Arduino Uno R3	Sensor Node	ATmega328P microcontroller, 16 MHz, 32 KB storage, 2 KB RAM	SHT31 (temperature & humidity via I2C), Photoresistor (light/door detection), Buzzer, LED
NVIDIA Jetson Nano	Analytics Node	Quad-core ARM CPU, 4 GB RAM, 128-core GPU	None (receives data from Arduino via USB)
Laptop (Ubuntu)	Fog Manager	Personal laptop running Ubuntu, connected to Wi-Fi	None
ThingSpeak	Cloud Service	Free IoT cloud platform by MathWorks	None (web service)

Additional components:

- Breadboard and jumper wires for connecting sensors to Arduino
- USB cable (Arduino to Jetson Nano)

3. Network Operation Report

3.1 Challenges We Encountered

Challenge 1: Sensor giving wrong readings sometimes

The SHT31 temperature sensor occasionally produced slight fluctuations in readings. While the SHT31 is more accurate than cheaper alternatives, we still implemented a simple averaging filter on the Jetson Nano that takes the average of the last 5 readings before making any risk decisions. This smoothing removes minor noise while still detecting real temperature changes quickly.

Challenge 2: Getting Arduino and Jetson Nano to talk to each other

At first, the Jetson Nano would sometimes receive garbled or incomplete messages from the Arduino over USB. We solved this by having the Arduino send data in a simple JSON format (e.g., {"temp": 4.2, "humidity": 45.1, "light": 312}) with a newline at the end of each message. The Jetson reads line by line and discards any message that can't be parsed as valid JSON.

Challenge 3: Setting up MQTT

Neither of us had used MQTT before, so it took some time to understand how it works. We installed Mosquitto (an MQTT broker) on the laptop and used the paho-mqtt Python library on the Jetson Nano to publish messages. After following a few online tutorials, we got it working. The key concept is that the Jetson publishes data to a "topic" (like a channel), and the laptop subscribes to that topic to receive the data.

Challenge 4: Uploading data to ThingSpeak

ThingSpeak has a rate limit of one update every 15 seconds on the free plan. Since our sensor reads every 2 seconds, we had to batch the readings on the laptop and only send an average value to ThingSpeak every 15 seconds. This was a simple fix but something we didn't expect initially.

3.2 Services We Are Using

Service	What It Does
Mosquitto	A free, lightweight MQTT broker that runs on our laptop. It acts as a middleman: the Jetson Nano sends data to Mosquitto, and our dashboard reads data from Mosquitto. This way the sender and receiver don't need to know about each other directly.
Flask (Python)	A simple Python web framework. We use it to run a local web dashboard on the laptop that shows the current temperature, risk level, and recent readings in a chart.
ThingSpeak	A free IoT cloud platform by MathWorks. We send sensor data to ThingSpeak via its REST API, and it automatically stores the data and generates time-series graphs that we can view from any browser.
Arduino IDE	Used to write and upload code to the Arduino board. The Arduino code reads the SHT31 sensor over I2C and the photoresistor, then sends JSON data over USB serial.
Python (Jetson Nano)	Python scripts on the Jetson Nano read serial data from Arduino, smooth the readings, compute a risk level (LOW/MEDIUM/HIGH), and publish results via MQTT.

3.3 Quality Attributes

Security

- The MQTT broker (Mosquitto) only accepts connections from devices on our local Wi-Fi network, so no one outside can send fake data.
- ThingSpeak channels are configured as private, meaning only we can view the data with our API key.
- The Arduino and Jetson Nano are connected by a physical USB cable, which cannot be intercepted remotely.

Privacy

- No personal information is collected. The only data transmitted is temperature, humidity, and light values with timestamps.
- ThingSpeak data is stored under our private account and is not shared publicly.

Reliability

- The most important feature of our system is that it keeps working even without internet. The Arduino reads sensors and the Jetson Nano analyzes data locally — no cloud needed for the core monitoring function.
- If the internet goes down, the laptop stores sensor data in a local file. When the internet comes back, it uploads the buffered data to ThingSpeak.
- The Arduino has a built-in hard limit: if temperature exceeds a dangerous threshold (e.g., above 8°C for vaccines), it immediately triggers the buzzer and LED, regardless of what the Jetson or laptop are doing.

3.4 How Devices Communicate

Our system uses three communication methods:

1. USB Serial (Arduino → Jetson Nano)

The simplest connection. Arduino sends a line of text (JSON format) through the USB cable every 2 seconds. The Jetson reads it using Python’s serial library. This is a direct one-to-one connection — no network needed.

2. MQTT (Jetson Nano → Laptop)

MQTT is a messaging system designed for IoT. It works like a group chat: the Jetson Nano “publishes” data to a topic (e.g., coldchain/temperature), and the laptop “subscribes” to that topic to receive it. The Mosquitto broker on the laptop manages this. We chose MQTT because it’s lightweight and widely used in IoT projects.

3. HTTP REST API (Laptop → ThingSpeak)

The laptop sends data to ThingSpeak using a simple HTTP GET request with the sensor values as URL parameters. This is the same kind of request your browser makes when you visit a website. ThingSpeak stores the data and makes it available as charts.

From → To	Connection	Method	How Often
Arduino → Jetson	USB cable	Serial (text/JSON)	Every 2 seconds
Jetson → Laptop	Wi-Fi (same network)	MQTT	Every 2 seconds

Laptop → ThingSpeak	Internet	HTTP GET request	Every 15 seconds

4. Remaining Tasks and Plans

The following tasks are planned for the next phase:

Dashboard Improvements

- Add a temperature history chart to the local Flask dashboard so we can see trends over time.
- Add color-coded risk indicators (green/yellow/red) for easy visual monitoring.

Fault Tolerance Demo

- Prepare a live demo where we disconnect the internet while the system is running, show that local monitoring and alerts continue working, then reconnect and show the buffered data uploading to ThingSpeak.

Better Risk Calculation

- Improve the risk algorithm on the Jetson Nano to consider not just the current temperature, but how long it has been above a safe threshold (cumulative exposure tracking).

ThingSpeak Analytics

- Use ThingSpeak's built-in MATLAB analysis to run simple predictions, such as estimating how long until the storage becomes unsafe based on current temperature trends.

Code Cleanup

- Clean up our GitHub repository, add a README with setup instructions, and document the code so others can reproduce our system.