ESP32 GPS Photo Uploader

NB-IoT Camera System with GNSS, Deep Sleep, and Power Management

Department of Physics
Czech Technical University in Prague

June 30, 2025

Contents

1 Introduction

This document describes an ESP32-based IoT camera system that leverages NB-IoT technology for wireless communication, incorporating GNSS (Global Navigation Satellite System) for precise positioning, a high-quality camera for image capture, and comprehensive power management. The device is designed with power efficiency as its primary concern, employing various deep sleep strategies and intelligent power management techniques.

The system captures photos using an OV2640 camera module and transmits them over TCP connections, while simultaneously sending sensor data and GPS coordinates via UDP in Telegraf line protocol format. The device includes temperature and humidity monitoring, battery level tracking, and an OLED display for status information.

Key Features

- **NB-IoT Communication**: Cellular connectivity for reliable data transmission
- **High-Quality Camera**: OV2640 camera with configurable resolution and quality
- **GNSS Positioning**: Accurate location tracking with GPS/GLONASS support
- **Advanced Power Management**: Deep sleep modes and power cycling
- **OLED Display**: Visual status information and debugging
- **Environmental Monitoring**: Temperature and humidity sensing
- **Battery Monitoring**: Real-time battery status reporting
- **Dual Protocol Support**: TCP for photos, UDP for sensor data
- **Night Mode**: Extended sleep during night hours
- **DNS-Based Configuration**: Flexible server addressing

The device is particularly suited for applications requiring periodic photo capture and transmission with minimal power consumption, such as environmental monitoring, security surveillance, and smart city infrastructure. Its ability to operate on battery power for extended periods makes it ideal for deployment in remote or hard-to-access locations.

2 Hardware Components

The camera system integrates several hardware components, each serving a specific function in the overall system:

The following figures show the main schematic sections of the NB-IoT camera system:

- FireBeetle 2 Board ESP32-S3 microcontroller and connectors
- NB-IoT BC95-G modem and supporting circuitry
- Solar charging and power management

2.1 ESP32-S3 Microcontroller

The ESP32-S3 is a powerful, dual-core microcontroller with integrated Wi-Fi and Bluetooth capabilities. For this application, we specifically use features including:

Component	Model	Function
Microcontroller	ESP32-S3	Main processing unit with integrated Wi-Fi and Bluetooth (disabled for power saving)
Camera Module	OV2640	High-quality image capture with configurable resolution
GNSS Module	DFRobot GNSS	Provides precise geographical coordinates
NB-IoT Modem	Quectel BC95	Enables cellular connectivity for data transmission
Power Management	AXP313A	Manages camera power, battery monitoring, and power optimization
OLED Display	LD7032 60x32	Shows device status and debug information
Environmental Sensor	AHT20	Monitors temperature and humidity
Battery	Li-ion/LiPo	Power source with voltage divider monitoring

Table 1: Hardware components of the NB-IoT camera system

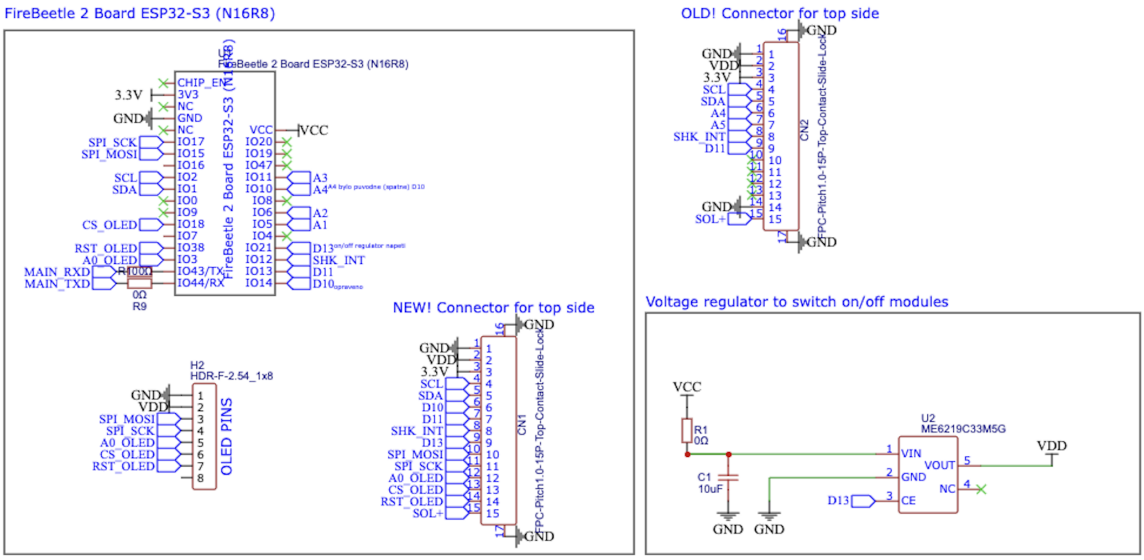

Figure 1: Schematic: FireBeetle 2 Board ESP32-S3 microcontroller and connectors.

- Low deep-sleep current consumption
- Multiple sleep modes (light sleep, deep sleep, and hibernation)
- Programmable I/O pins with wake-up capability
- PSRAM support for high-quality image processing
- Hardware acceleration for cryptographic operations

2.2 OV2640 Camera Module

The OV2640 camera module provides high-quality image capture capabilities:

- Resolution up to 1600x1200 pixels
- Configurable JPEG quality settings
- Automatic exposure and white balance control

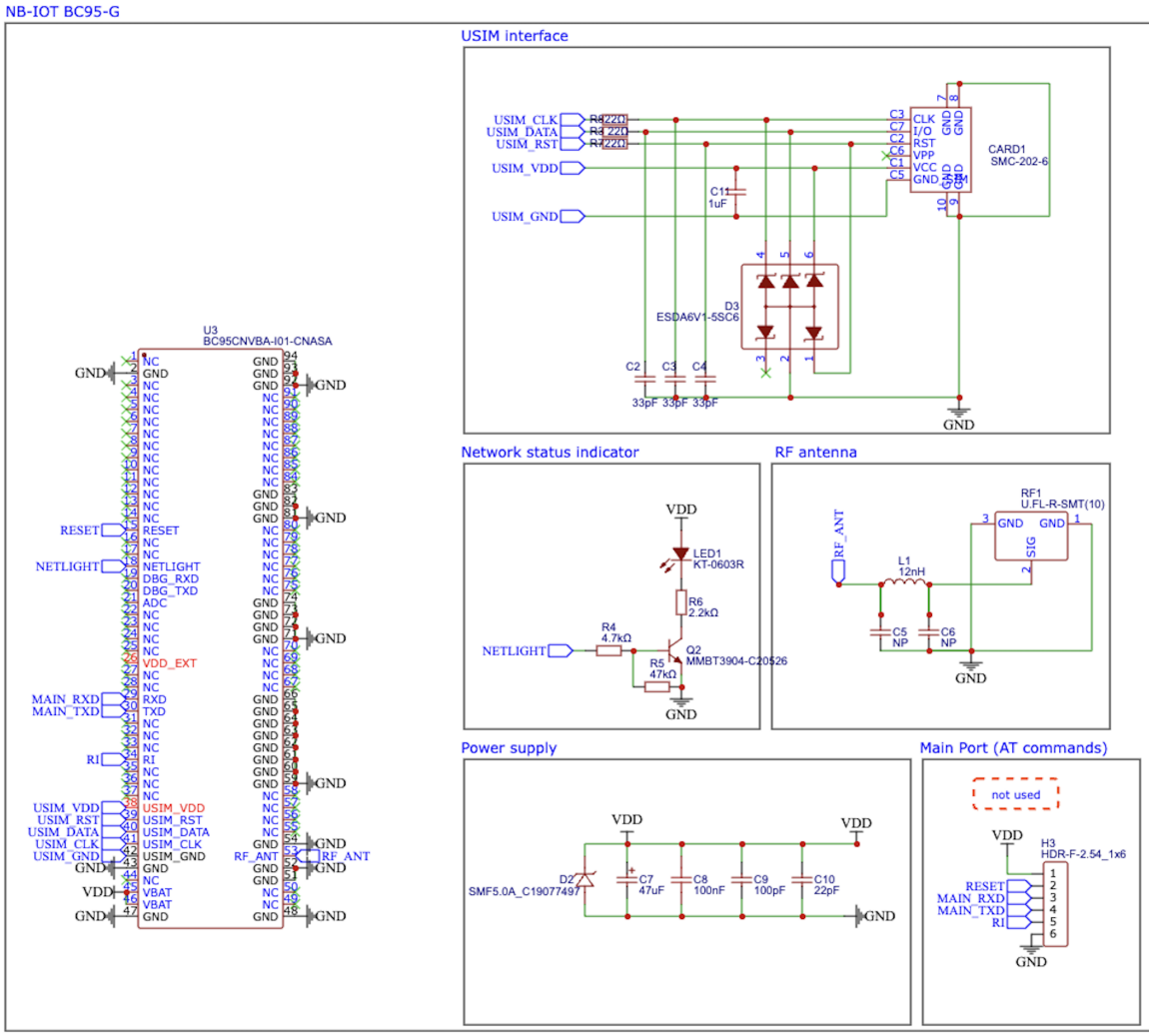

Figure 2: Schematic: NB-IoT BC95-G modem and supporting circuitry.

- SCCB (I2C) interface for configuration
- 8-bit parallel data interface

2.3 NB-IoT Communication

The device uses a Quectel BC95 NB-IoT modem to communicate over cellular networks. NB-IoT offers several advantages for IoT applications:

- Wide coverage area with cellular infrastructure
- Low power consumption compared to traditional cellular
- Reliable connectivity in challenging environments
- Support for both TCP and UDP protocols
- Cost-effective data transmission

2.4 GNSS Module

The GNSS module provides precise geographical positioning by receiving signals from multiple satellite constellations (GPS, GLONASS). Key features include:

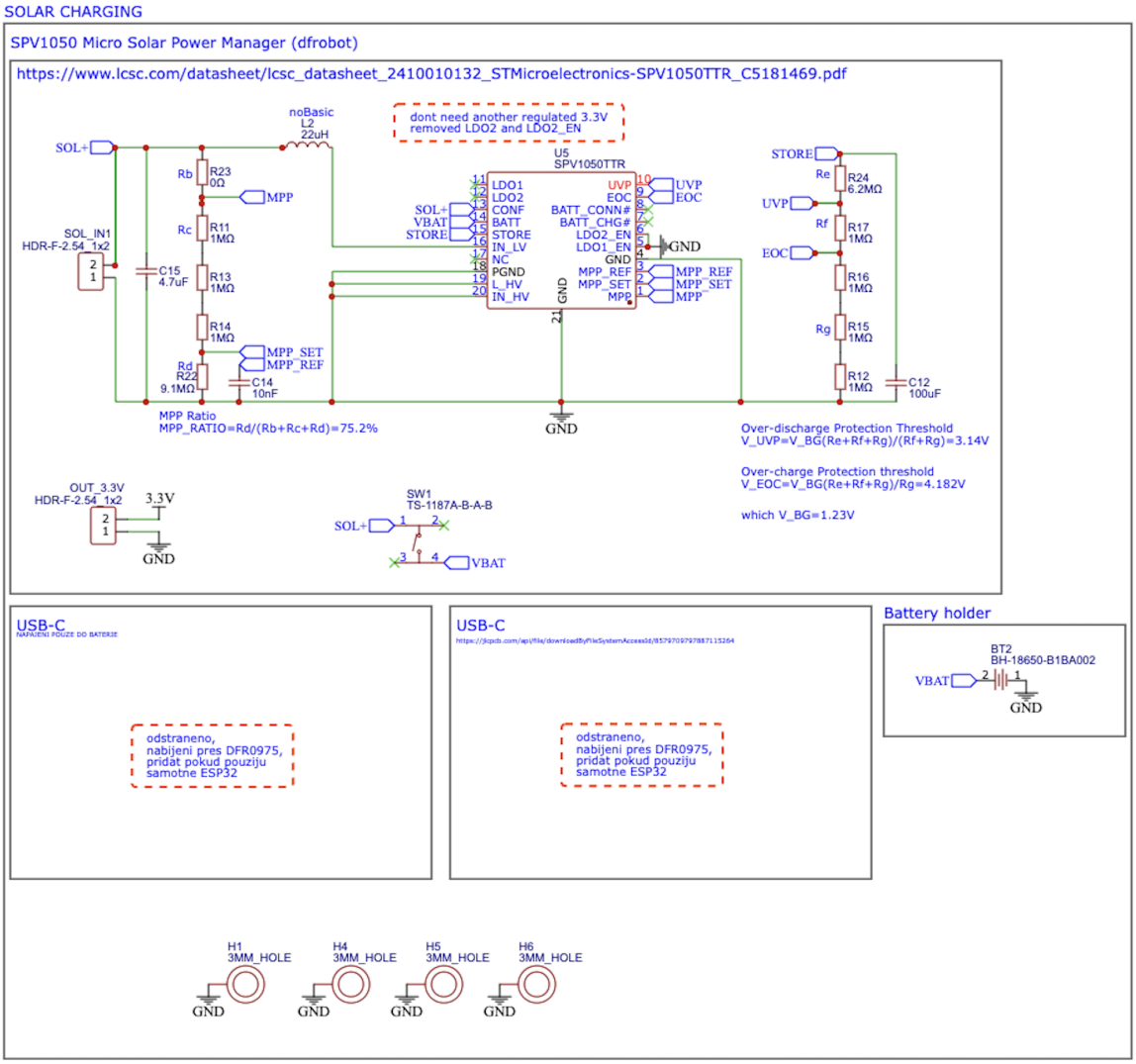

Figure 3: Schematic: Solar charging and power management section.

- Support for multiple satellite systems
- Configurable power modes
- I2C interface for communication with the ESP32
- Position accuracy to within a few meters
- Speed and course over ground information

2.5 Power Management System

The AXP313A power management IC provides comprehensive power control:

- **Camera Power Control**: Dedicated power rail for the camera module
- **Battery Monitoring**: Real-time voltage and percentage monitoring
- **Power Optimization**: Automatic power cycling and sleep management
- **Voltage Regulation**: Multiple regulated outputs for different components

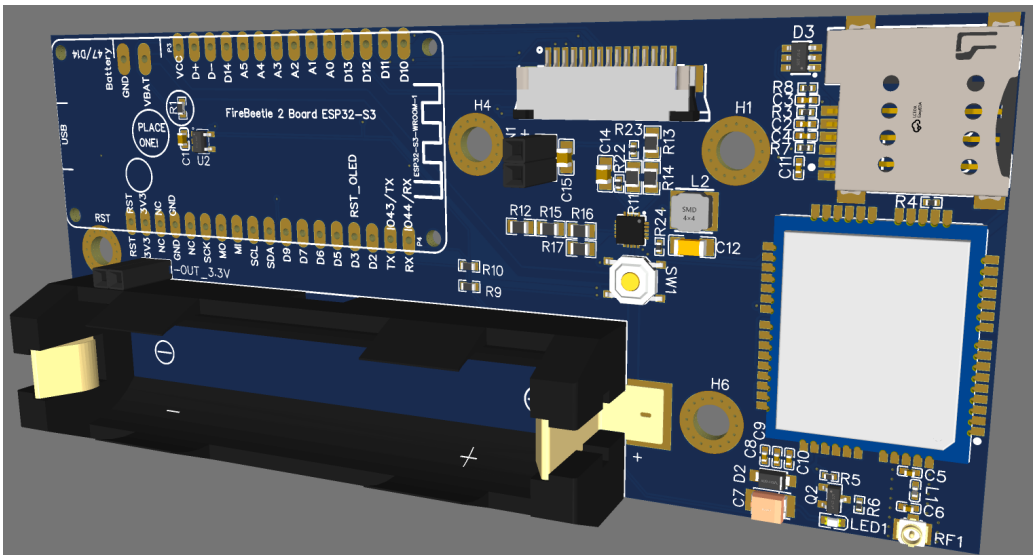

Figure 4: 3D model of the main NB-IoT camera PCB board.

3 Software Architecture

The firmware is structured around power-efficient operation cycles, with careful management of active and sleep states. The main operational cycle consists of:

1. Wake up (from timer or external interrupt)
2. Initialize sensors and modules
3. Acquire GNSS data (if enabled)
4. Capture photo (if enabled)
5. Transmit data via NB-IoT
6. Enter deep sleep mode

3.1 Operational Modes

The device supports configurable operational modes:

Camera Mode Captures and transmits photos over TCP with start/end markers

Sensor Mode Transmits environmental and GPS data over UDP

Combined Mode Both photo capture and sensor data transmission

Night Mode Extended sleep periods during night hours

3.2 Sleep Management

Power efficiency is achieved through sophisticated sleep management:

- **Deep Sleep**: The ESP32 and peripheral components are powered down between transmissions
- **Adaptive Sleep Duration**: Sleep times can be adjusted based on time of day

Figure 5: NB-IoT unit on the right, placed in a demolition container.

- **RTC Memory**: Critical state information is preserved in RTC memory
- **Wake-up Sources**: The device can wake from timer expiration
- **Component Power Cycling**: Individual components are powered down when not in use

4 Data Communication

4.1 NB-IoT Integration

The device uses AT commands to communicate with the Quectel BC95 modem:

1. Initialize the modem with proper configuration
2. Register on the NB-IoT network
3. Establish TCP/UDP connections as needed
4. Transmit data with retry mechanisms
5. Close connections and enter sleep mode

4.2 TCP Photo Transmission

Photos are transmitted over TCP with special markers for reliable delivery:

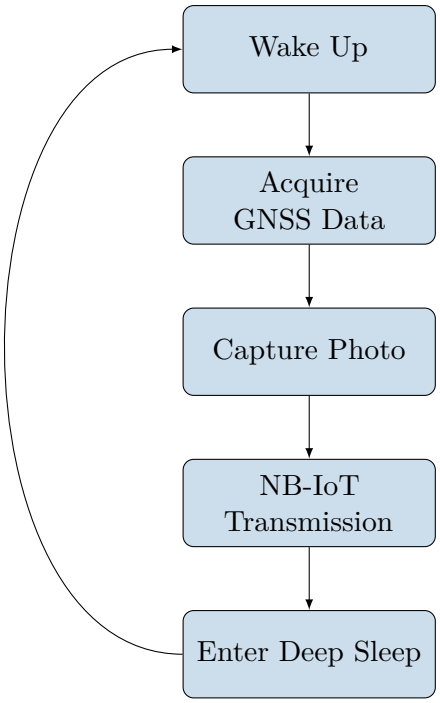

Figure 6: Operational cycle of the NB-IoT camera system

Component	Size	Description
Start Marker	4 bytes	[0x00, 0x00, 0x00, 0x00]
Photo Data	Variable	JPEG image data in chunks
End Marker	4 bytes	[0xFF, 0xFF, 0xFF, 0xFF]

Table 2: TCP photo transmission format

4.3 UDP Sensor Data Format

Sensor and GPS data is transmitted in Telegraf line protocol format:

```
gps_data,device=esp32 gpsFIX=true,temperature=23.1,humidity=45.2,battery=42
```

Listing 1: UDP Data Format

The data includes:

- GPS fix status (true/false)
- Temperature in degrees Celsius
- Humidity percentage
- Battery percentage

5 Power Optimization

Several techniques are employed to minimize power consumption:

5.1 Hardware Power Management

- **Power Gating**: The camera and GNSS modules are completely powered down when not in use
- **Modem Sleep**: The NB-IoT modem enters sleep mode between transmissions
- **WiFi/Bluetooth Disable**: These radios are disabled as they're not needed
- **Voltage Regulation**: Efficient power conversion and distribution

5.2 Software Power Management

- **Adaptive GNSS Timeout**: The device limits how long it will wait for a GNSS fix
- **Modem Sleep**: The ESP32 is configured to enter modem sleep to reduce power
- **Night-time Scheduling**: Longer sleep periods are used during night hours
- **Component Initialization**: Only initialize components when needed

5.3 Measured Power Consumption

The device's power consumption varies significantly across different operational phases:

Mode	Current	Duration	Energy per Cycle
Deep Sleep	15 µA	3600 s	54 mC
GNSS Acquisition	70 mA	10-120 s	700-8400 mC
Camera Capture	150 mA	5-10 s	750-1500 mC
NB-IoT TX	200 mA	2-5 s	400-1000 mC
Processing	50 mA	3 s	150 mC

Table 3: Power consumption in different operational states

6 Data Integration

The device integrates with backend systems through multiple pathways:

6.1 NB-IoT Network

Data transmitted by the device is first received by NB-IoT base stations and forwarded to the cellular network infrastructure, which then routes the data to the configured server.

6.2 Node-RED Integration

The device's data is processed using Node-RED flows that:

1. Decode the TCP photo data and reconstruct complete images
2. Parse UDP sensor data in Telegraf line protocol format
3. Extract relevant fields (GPS coordinates, sensor readings, battery status)
4. Forward data to storage or visualization systems

6.3 Server Configuration

The device connects to a server at `www.iot-magic.com` using:

- **TCP Port 8009**: For photo transmission
- **UDP Port 8094**: For sensor data
- **DNS Resolution**: Flexible server addressing without hardcoded IPs

7 Setup and Configuration

7.1 Hardware Setup

To configure the hardware:

1. Connect the OV2640 camera to the designated GPIO pins
2. Connect the GNSS module to the correct I2C pins
3. Connect the BC95 modem using UART2 (pins 43/44)
4. Connect the AHT20 sensor to I2C
5. Connect the OLED display to the designated SDA/SCL pins
6. Connect the AXP313A power management IC
7. Connect the battery with voltage divider for monitoring

7.2 NB-IoT Configuration

The device requires proper NB-IoT network configuration:

```
// Server configuration
const String SERVER_IP = "www.iot-magic.com";
const int TCP_PORT = 8009;
const int UDP_PORT = 8094;

// Modem configuration
#define BC95_RX_PIN 44
#define BC95_TX_PIN 43
```

Listing 2: NB-IoT Configuration

7.3 Firmware Configuration

Key firmware parameters that can be adjusted:

- **Sleep Duration**: The default sleep time between transmissions (3600 seconds)
- **Camera Quality**: JPEG quality and resolution settings
- **GNSS Timeout**: How long to attempt to acquire a GNSS fix (120 seconds)
- **Night-time Parameters**: Configure different behavior during night hours
- **Retry Mechanisms**: Number of retry attempts for failed transmissions

8 Applications and Use Cases

The NB-IoT camera system is versatile and can be applied to various domains:

8.1 Environmental Monitoring

Capture images of environmental conditions while collecting sensor data for comprehensive monitoring of remote locations.

8.2 Security and Surveillance

Deploy units in areas requiring periodic visual monitoring with minimal power requirements and cellular connectivity.

8.3 Smart City Infrastructure

Monitor fixed infrastructure assets while providing visual context and location information.

8.4 Research and Education

The device serves as an excellent platform for academic research and educational purposes, demonstrating IoT principles, power optimization techniques, and image processing.

9 Future Development

Several enhancements are planned or possible for future iterations:

- **Additional Sensors**: Integration of more environmental sensors
- **Image Processing**: Local image analysis and compression
- **Enhanced Security**: Implementation of additional security measures
- **Remote Configuration**: Adding the ability to configure device parameters via downlink
- **Multi-Camera Support**: Support for multiple camera modules

10 Documentation and Resources

The project includes comprehensive documentation:

- **LaTeX Chart**: See `documents/capacity_chart.tex` for professional charts
- **Datasheets**: See `datasheets/` for hardware reference manuals
- **Node-RED Flows**: Server-side processing flows for TCP/UDP data
- **Real-time Dashboard**: Available at `rm.fsv.cvut.cz`

11 Example Data and Visualization

Figure **??** shows an example photo taken from the IoT unit, demonstrating the perspective and coverage of the camera when the container is empty.

The data collected by the NB-IoT camera system is visualized and managed through a web dashboard. Figure **??** shows a screenshot of the dashboard, which displays real-time sensor data, fill level, battery status, and location. The dashboard is accessible at https://rm.fsv.cvut.cz/container/5/.

Figure 7: Example photo taken from the IoT unit camera, showing an empty demolition container.

12 Conclusion

The current ESP32-based NB-IoT camera system demonstrates a robust platform for remote image and sensor data collection in challenging environments. While the present design uses a FireBeetle breakout board for the ESP32-S3, it is important to note that this configuration results in higher power consumption, particularly due to the camera's connection through the breakout. This limitation will be addressed in the next hardware iteration, where the ESP32-S3 will be placed directly on a custom-designed PCB. Prototypes of this new design have already been tested successfully, promising significant improvements in power efficiency and integration.

The primary purpose of the current unit is to collect a diverse dataset of images from the field. These images will be used to develop and train machine learning models for automated image analysis. In future versions, the trained model will be deployed directly on the device, enabling real-time, on-device inference. This approach will drastically reduce the need to transmit large volumes of image data, saving both bandwidth and energy. The device will be able to provide users with actionable information, such as container fill level and material classification, and transmit only images deemed important by the operator—such as those indicating a violation of allowed materials or other critical events.

This evolution will transform the NB-IoT camera system from a data collection tool into an intelligent edge device, capable of delivering timely and relevant insights while maintaining ultra-low power operation. The flexible architecture and ongoing hardware improvements ensure that the system can adapt to new requirements and continue to provide value in a wide range of smart monitoring applications.

For more information, code examples, and real-time data visualization, please visit:
rm.fsv.cvut.cz

References

[1] Espressif Systems, "ESP32-S3 Series Datasheet," 2021.

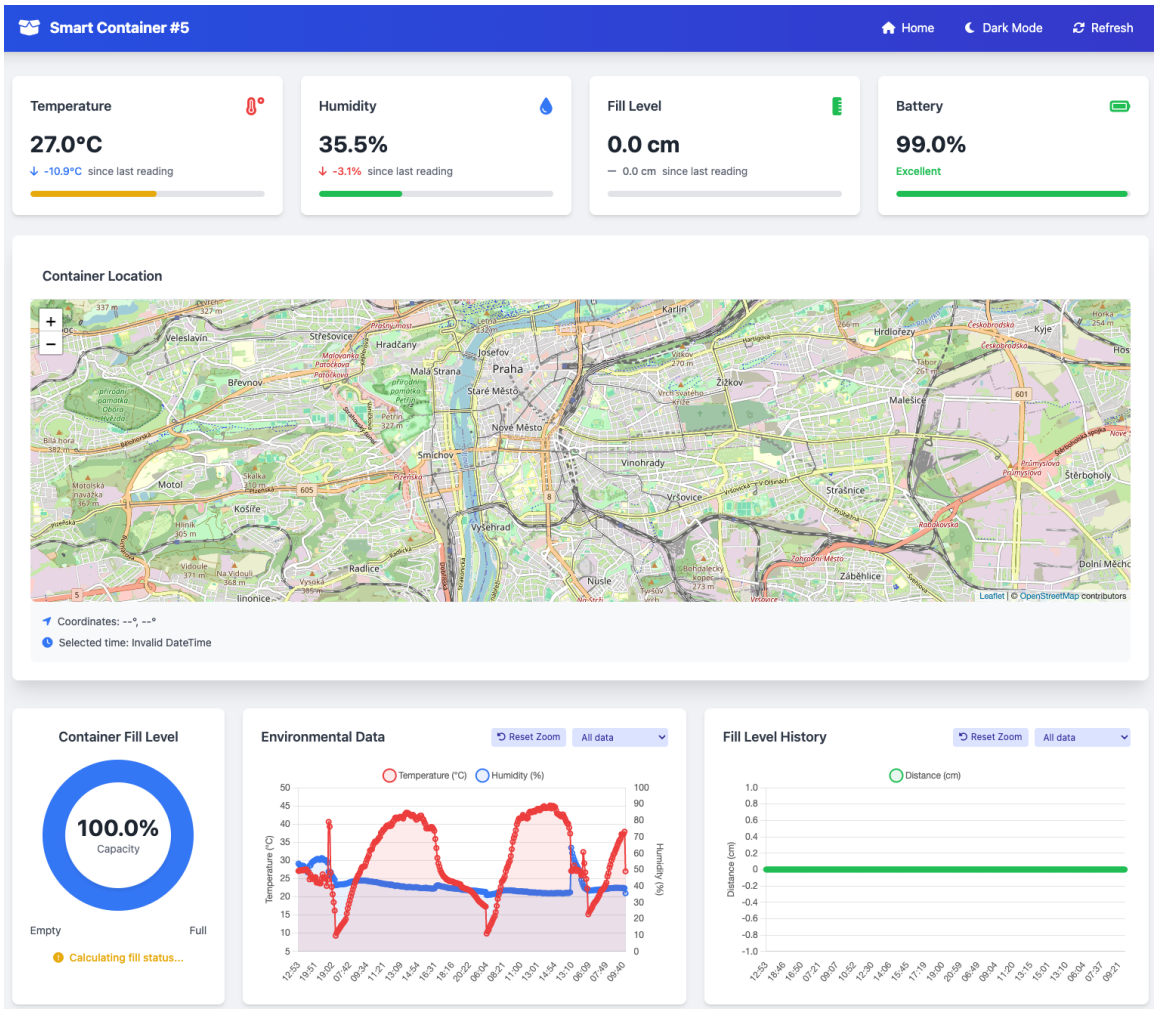

Figure 8: Screenshot of the smart container web dashboard at https://rm.fsv.cvut.cz/container/5/, showing real-time data and analytics.

[2] Quectel, "BC95-G Hardware Design," 2020.

[3] OmniVision, "OV2640 Camera Module Datasheet," 2019.

[4] X-Powers, "AXP313A Power Management IC Datasheet," 2020.

[5] DFRobot, "GNSS Module Documentation," 2020.

Appendix: IoT Server Setup Summary (Node-RED, InfluxDB 2.x, Grafana, UDP/TCP)

The following components were successfully configured and validated on the Google Cloud VM instance:

- **Node-RED** running and listening on custom ports
- **InfluxDB 3 removed**, InfluxDB 2.x installed and initialized with web UI
- **Grafana** installed and connected to InfluxDB 2.x
- Custom **Node-RED flow** parsing UDP payloads into InfluxDB-ready format
- Successfully tested **TCP and UDP connectivity** on ports 8009 and 8094

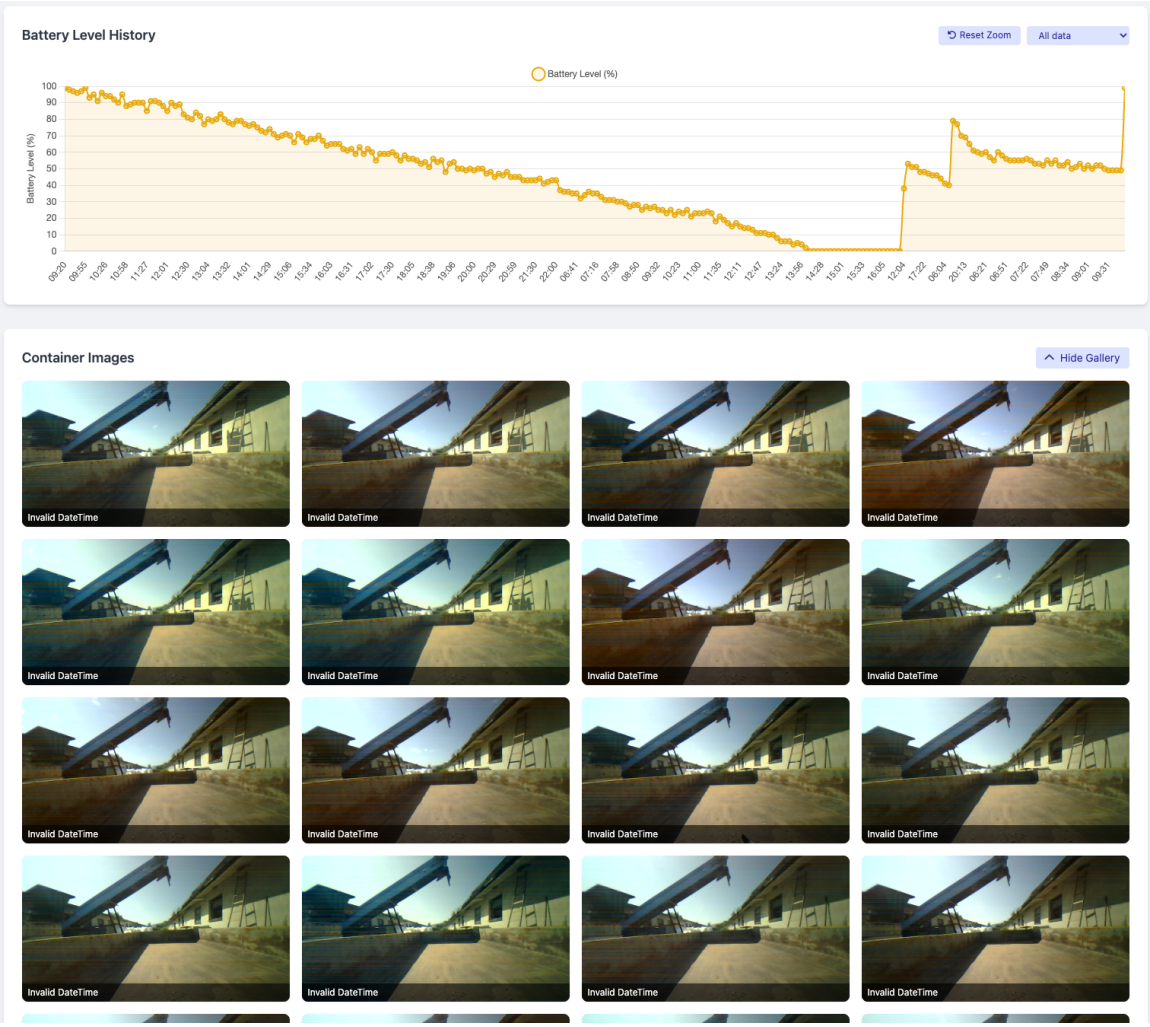

Figure 9: Another view of the smart container web dashboard, showing battery monitoring and photo gallery features.

Installed Services

Node-RED

- Installed with `npm install -g -unsafe-perm node-red`
- Run as a persistent service using PM2:

```
pm2 start $(which node-red) -- -v
pm2 save
pm2 startup
```

- Configured to accept:
  - TCP connections on port `8009`
  - UDP messages on port `8094`

InfluxDB 2.x

- Removed InfluxDB 3.x
- Installed 2.x via:

```
curl -s https://repos.influxdata.com/influxdata-archive_compat.key
    | \
  gpg --dearmor | sudo tee /usr/share/keyrings/influxdb-archive-
     keyring.gpg > /dev/null

echo "deb [signed-by=/usr/share/keyrings/influxdb-archive-keyring.
    gpg] \
  https://repos.influxdata.com/debian stable main" | \
  sudo tee /etc/apt/sources.list.d/influxdb.list

sudo apt update
sudo apt install influxdb2
```

- Web UI available at `http://iot-magic.com:8086`
- Created:
  - Bucket: `container`
  - Organization: `iot`
  - Token: `iot-write-token`

Grafana

- Installed via:

```
curl -fsSL https://apt.grafana.com/gpg.key | \
  sudo gpg --dearmor -o /etc/apt/keyrings/grafana.gpg

echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] \
  https://apt.grafana.com stable main" | \
  sudo tee /etc/apt/sources.list.d/grafana.list

sudo apt update
sudo apt install grafana
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
```

- Access at `http://iot-magic.com:3000`
- Connected to InfluxDB using token and Flux queries
- Implemented:
  - State timeline panel for `wakeUpReasonText`
  - Geomap panel with `latitude`, `longitude`, `RSSI`
  - Highlighted latest known device position using `last()` query

UDP Integration and Message Format

UDP Payload Format

Node-RED receives messages like:

```
sensor temperature=22.5,humidity=65,latitude=50.0875,longitude=14.4213,
    battery=89
```

Node-RED Function Logic

- Parses UDP `msg.payload` (as Buffer)
- Splits key=value fields
- Creates structured JSON for InfluxDB:

```
{
  temperature: 22.5,
  humidity: 65,
  distance: 0,
  latitude: 50.0875,
  longitude: 14.4213,
  container_id: 5,
  image_status: 1,
  battery_status: 89,
  gps_fix_time: 0,
  wake_up_reason: 0
}
```

Connectivity Tests

Local Port Checks

```
sudo apt install lsof
sudo lsof -i -n -P | grep -E '8009|8094'
```

Local UDP/TCP Sending

```
echo "hello tcp" | nc 127.0.0.1 8009
echo "hello udp" | nc -u 127.0.0.1 8094
```

Remote UDP Sending Example

```
echo "sensor temperature=22.5,humidity=65,latitude=50.0875,longitude
   =14.4213,battery=89" \
  | nc -u <your-vm-ip> 8094
```

ESP32 Wake-Up Reason Codes

Used for mapping `wakeUpReason` integers:

Code	Meaning
0	ESP_SLEEP_WAKEUP_UNDEFINED
1	ESP_SLEEP_WAKEUP_EXT0 (RTC_IO)
2	ESP_SLEEP_WAKEUP_EXT1 (RTC_CNTL)
3	ESP_SLEEP_WAKEUP_TIMER
4	ESP_SLEEP_WAKEUP_TOUCHPAD
5	ESP_SLEEP_WAKEUP_ULP
6	ESP_SLEEP_WAKEUP_GPIO
7	ESP_SLEEP_WAKEUP_UART
8	ESP_SLEEP_WAKEUP_WIFI
9	ESP_SLEEP_WAKEUP_COCPU
10	ESP_SLEEP_WAKEUP_COCPU_TRAP
11	ESP_SLEEP_WAKEUP_BT

Table 4: ESP32 wake-up reason codes