Adaptive Low-Power IoT Protocol

From IoT SoC to IoT Platform

Przemyslaw Bakowski, Benoît Parrein - LS2N

Introduction

This work presents the development of an **Adaptive Low-Power IoT Protocol** implemented on dedicated IoT platforms. These platforms integrate **Single Board Computers** (SBCs) and IoT development kits (**DevKits**).The available SBCs include a RISC-V (SpacemiT-X60) board and an x86 (Intel-N100) board. The IoT DevKits incorporate the ESP32-C3 IoT SoC, which provides the necessary components for developing the Adaptive Low-Power IoT Protocol. This SoC features integrated Wi-Fi and MAC-Wi-Fi communication capabilities. Additionally, the IoT DevKits include **LoRa** modems.

Using these communication modules, we build three types of IoT terminals: **Direct Terminals**, **Close Terminals**, and **Remote Terminals**, along with the necessary routers and gateways. The operation of the Adaptive Low-Power Protocol focuses on reducing power (current) consumption in terminals that control various types of sensors. This is achieved through the use of multiple functional modes and different types of memory units. The use of **deepsleep** mode, with a current consumption of approximately 20 µA, is a key mechanism for reducing overall power consumption. The longer the low-power stage (deepsleep mode) lasts, the lower the terminal's total energy usage. During the high-power stage, the terminal may or may not transmit a data packet. Given the relatively high current consumption associated with the transmission phase, it is essential to minimize its activation whenever possible. This behavior is managed by analyzing the difference between the last transmitted sensor value and the current reading, a delta meta-variable. To implement these mechanisms and build the Adaptive Low-Power Protocol, we leverage various types of memory units, including main SRAM, low-power SRAM, internal non-volatile storage (NVS), and external EEPROM blocks.

Cycles, stages and phases – from simple cyclical to adaptive mode

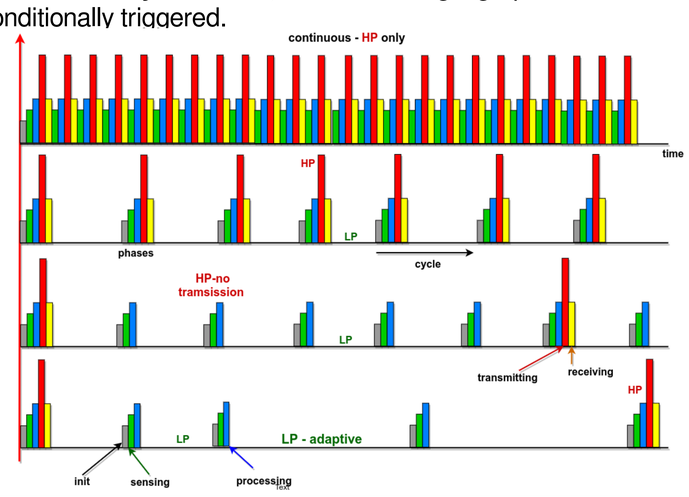
The following diagram illustrates the main operational modes (without interruption) of a terminal node:

Continuous – Operates only in the high-power stage, kept as short as possible.

Cyclical – A regular cycle alternating between high-power and low-power modes; includes a transmission phase in each cycle.

Cyclical/Conditioned – A regular cycle with high-power and low-power modes; the transmission phase occurs only under certain conditions.

Adaptive – An adaptive cycle based on a **delta** value and **cycle factor**, with alternating high-power and low-power modes; the transmission phase is conditionally triggered.



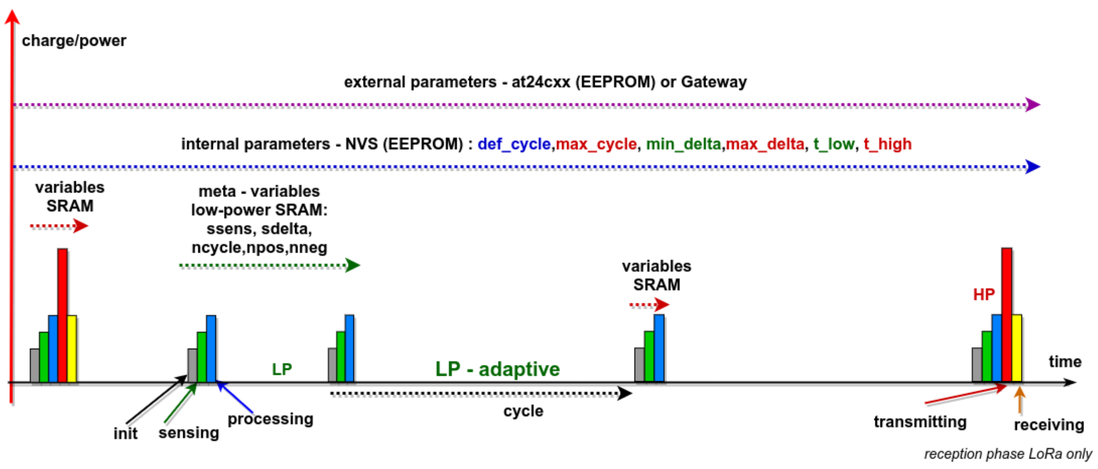
Adaptive mode : parameters, meta-variables, variables

The implementation of the adaptive mode requires the use of the deep sleep state and various types of memory to store and manage parameters, metadata, and data:

Parameters – Externally provided, modifiable constant values used to control the evolution of meta-variables.

Meta-variables – Internal variables maintained during the SoC's deep sleep state.

Variables – Data acquired from sensors, processed, and transmitted during the high-power stage.



During the high-power stage, all types of data—parameters, meta-variables, and variables—are used to read sensor values, transmit data packets, evaluate whether transmission is necessary, update the cycle factor, and adjust the delta value.

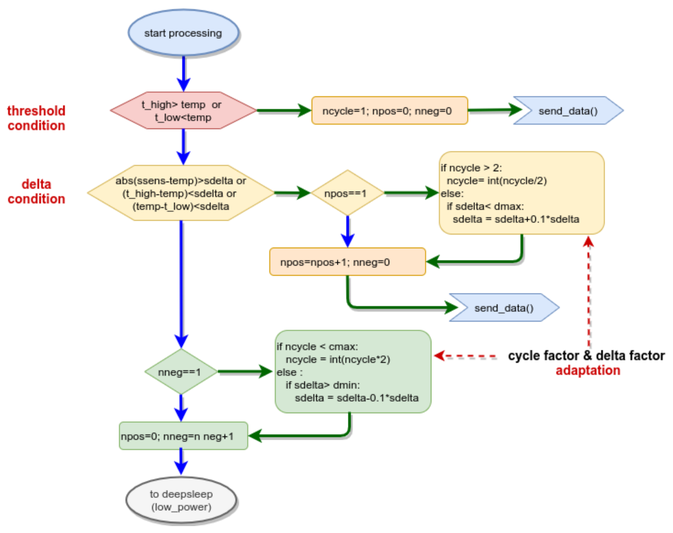
The key objective is to **maximize the duration of the next low-power cycle** and **minimize the use of the transmission phase**.

Adaptive mode : control flow

During the high-power stage, conditional operations (written in µPython) are executed to first evaluate whether there is a need to send an "urgent" data packet. This decision is based on **threshold** values (t_high, t_low) provided as parameters.

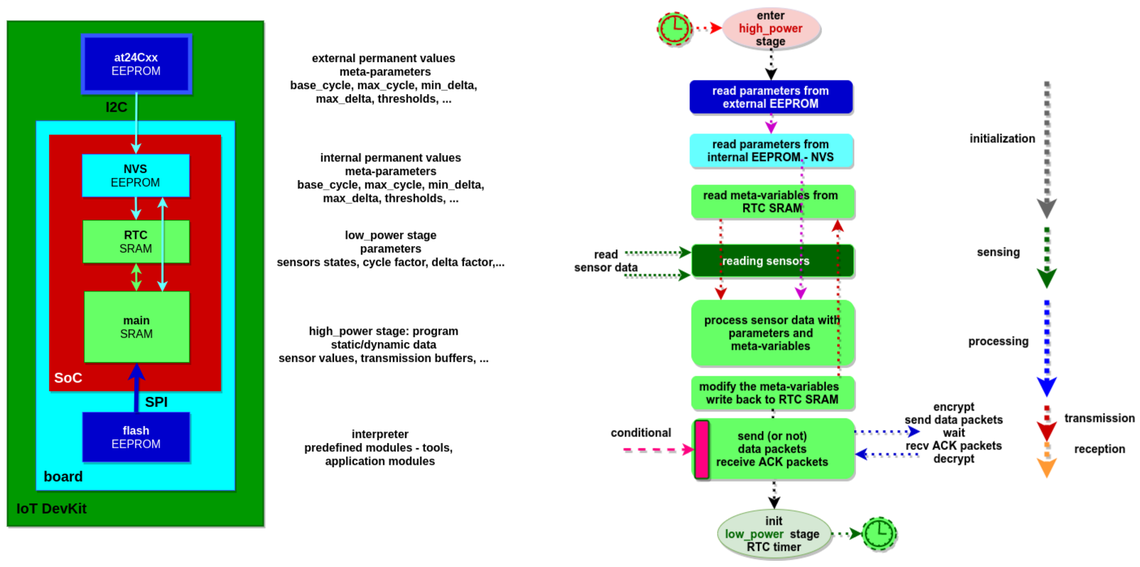
The next step involves evaluating the difference (**delta**) between the last transmitted sensor value and the current reading. The last sent value is stored in **non-volatile storage** (NVS). This condition also considers the proximity of the current value to the defined thresholds.

If both conditions are not met, no data packet is transmitted. In all cases, the new cycle factor and, if necessary, an updated delta value are computed. These values are then stored in NVS memory.



Implementation aspects

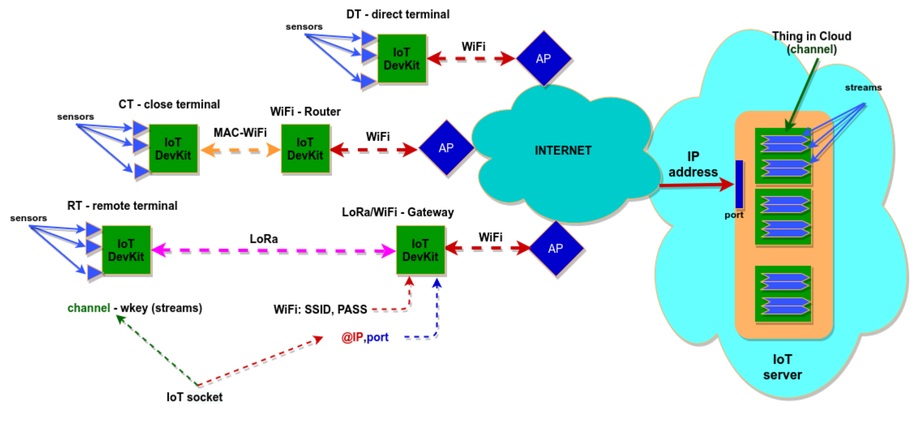
The implementation of the adaptive low-power protocol involves several types of memory units as shown below.



The figure on the right illustrates the execution process of the high-power stage, highlighting the use of various data types and memory units. The process begins with the registered parameters and meta-variables, which are used to process the data and generate updated meta-variable values.

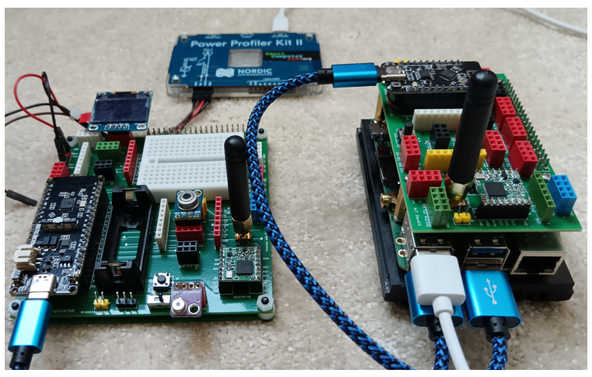
Direct, Close and Remote terminals - implementation aspects

The same type of "adaptive" processing is performed in Direct, Close, and Remote Terminals. However, Remote Terminals also include a reception phase and use an **AES key** to secure transmissions over LoRa links. The following figure illustrates how the "**things**" (DevKits) and their associated **sensors** are **mapped** to **channels** and **data streams** implemented on IoT servers.

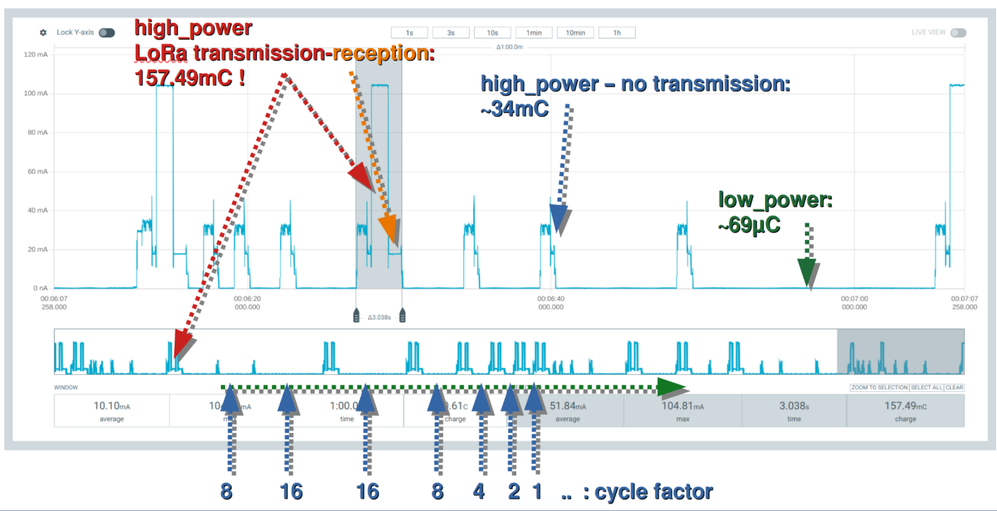


Implementation results

The Adaptive Low-Power Protocol is implemented on an IoT platform comprising an SBC (x86-N100) and an IoT DevKit. Programming is performed ***in situ*** on the IoT DevKit using µPython and the Thonny IDE.



The PPK II power profiler is used to analyze current (charge) consumption across different types of terminals and sensors, as well as under various parameter configurations. The following PPK diagram for a **Remote Terminal** (LoRa: SF=11, SB=125 kHz, CR=8) shows **current consumption** across different operational stages, along with the evolution of the meta-variables (**cycle factor**).



Conclusion

Modern IoT SoCs offer a wide range of components and features that enable the development of energy-efficient IoT protocols. Leveraging these capabilities, we have developed an Adaptive Low-Power IoT Protocol which, according to initial tests, significantly reduces current consumption across various types of IoT terminals.

However, extensive and time-consuming testing with different sensor types and configuration parameters is still required to fully assess and validate the protocol's efficiency.

Acknowledgements

We would like to thank our colleagues, as well as our MSc and PhD students, for their valuable feedback on the various courses and numerous projects based on our IoT DevKit platforms.

References

[1] Bakowski (P.), Parrein (B.) – Low-Power IoT Architectures , (poster) Journées LPWAN, Grenoble, 2023
[2] Bakowski (P.) – Programming, Modeling and IoT development on RISC-V architectures (poster) – "RISC-V Summit Europe". Paris, 2025
[3] Bakowski (P.) – Very Low-Power IoT Architectures , "COMPASS" , Nantes, 2024