Resource-Aware Design of an IoT Node for Use in Remote Industrial and Hazardous Areas*

Petar Rajković [1], *, Milan Vesković [2], Dejan Aleksić [3] and Dragan Janković [1]

[1] University of Niš, Faculty of Electronic Engineering
Aleksandra Medvedeva 4, 18104 Niš, Serbia
petar.rajkovic@elfak.ni.ac.rs,
dragan.jankovic@elfak.ni.ac.rs
[2] Faculty of Technical Sciences Čačak
Svetog Save 65, 32102 Čačak, Serbia
milan.veskovic@ftn.kg.ac.rs
[3] University of Nis, Faculty of Sciences and Mathematics, Department of Physics
Višegradska 33, PO BOX 224, 18106 Niš, Serbia
alexa@pmf.ni.ac.rs

Abstract: As the Internet of Things (IoT) nodes become one of the cornerstones of Industry 4.0, they tend to be incorporated into every aspect of production automation. This paper addresses the challenge of designing low-power IoT nodes based on standardized components for deployment in remote, off-grid, industrial, and hazardous environments where energy efficiency and autonomy are critical. The proposed design integrates hardware-software co-design, replacing standard hardware setup with energy-efficient components, solar-powered batteries, and dynamic working modes to reduce energy consumption. Software elements were designed with the possibility of over-the-air updates and reconfiguration. Next, battery charging routines are optimized, and the node is integrated into a cloud-based digital twin with centralized control over the complete operation cycle. The proposed node architecture achieves an energy reduction of up to 50% and, in some configurations, reduces consumption by up to one-tenth compared to conventional designs. The additional result is a set of design recommendations when the standard components must be adapted for harsh environments.

Keywords: internet of things, resource awareness, industry 4.0, hardware-software codesign

1. Introduction and Background

The IoT represents a world of relatively small devices connected to networks that can capture, use, and exchange data [1]. This emerging paradigm has spread over business integration [2] and industrial automation in recent years. It created benefits for smart manufacturing [3] and Industry 4.0 [4], fueling the advances considered the new industrial revolution. Integrating IoT devices with increased computing power brought benefits not envisioned a decade ago [5]. Installing such devices to the production lines initially facilitates the data exchange with control systems. As a primary consequence, the reaction of the complete production systems becomes faster, better, and more accurate. With more extensive and detailed data sets, the production enterprises could initiate the changes in the planning process and give an additional plus to the production [6].

---

* This manuscript is an extended version of the papers published in proceedings of the CERCIRAS 2023 workshop and SQAMIA 2023 conference

Our research group has designed software components for different manufacturing systems for over a decade. The research has been focused on solutions targeting the planning [7], execution [8], development [9], and deployment [10] of the software for industrial systems at various levels according to ISA-95 (ISA – International Society of Automation) standards [11]. The requirements and challenges vary from level to level, but operational efficiency is a must. The research presented in this paper focuses on ISA-95 levels 0 and 1. Levels 0 and 1 consist of sensor networks, actuators, and other devices that bring data to IoT nodes. Such nodes sometimes operate in complex and demanding environments, aiming to be self-sustainable as much as possible. In such a case, the design must consider that the device will run in harsh exploitation using minimal power and network resources.

Figure 1 Developed IoT node before sealing in the safety *Ex e* casing

Industrial hazardous areas, such as processing refineries, are the parts of the plants and industrial facilities where the environmental effects could permanently damage human health or threaten safety by emitting harmful gases or chemicals and where small parameter changes could cause an explosion [14]. This environment implies that any foreign object brought in (such as sensors and IoT nodes) must be designed with minimal environmental impact. In this light, any additional wiring and connection to different pieces of equipment is a source of high potential risk. Safety standards [15] imply that equipment must be packed into *Ex e* enclosures (Figure 1). The complete node and all communication devices, batteries, and charge controllers should be in the verified casings (ideally in the same casing), and the node's building and operational costs should be the lowest possible.

The IoT nodes are considered to communicate with Edge computers. Since the communication between the IoT and Edge layer must be set up and maintained, selecting the wireless network will avoid additional wiring. This connection is also essential to make remote OTA (Over-the-air) configuration and management highly efficient.

An effective wireless connection is especially needed for mobile nodes in vehicles that carry dangerous or explosive materials. These vehicles need constant monitoring of the transported substance. Unlike stationary devices, mobile devices' location must be monitored in addition to all the standard values. It is essential to note that Edge computers in such a scenario are usually not in the same network or physically close, so the proper communication protocol must be defined or chosen.

To meet the requirement for such a node, we started the research that resulted in a new architecture. The architecture employs all the benefits of the IoT concepts, supported by general resource awareness. Initial results are presented as conference papers [12] and [13], and this work represents their direct extension. The focus of the work [12] was on the battery charging routines and hardware design that examines energy consumption in different working nodes. The result is the hardware setup, which should allow the IoT system to work for as long as possible.

Another founding block for this research is the modular software development approach, which was initially described in the paper [13]. Necessary changes in hardware design must be followed with new approaches in software development to make the complete system effective. Description of the IoT node's software platform, the routines for transitions to sleep mode, node update, and configuration steps are included from [13] and extended to make the complete picture of the developed IoT node.

Besides many custom-built solutions in the market and the literature, the main requirement was to stay with the widely used components, which are easily affordable worldwide and backed up by comprehensive support communities. Many existing (entirely off-grid) designs were built on high-end components that are either too expensive, not easily replaceable, or without a broad enough support network. Having in mind maintainability, together with the focus on low energy consumption, the following main design goals are formulated:

- Base the design on the standard components proven in the industrial environment to reuse standardized solutions and increase maintainability
- Identify the top energy consumers within the standard IoT node and replace them with the appropriate external components. In this way, energy consumption should be reduced, and the maintainability level should remain the same
- Introduce redundancy for the critical elements of the design. This will increase the system's availability and general readiness (such as transmission modules and sensors)
- Introduce new working modes for the existing IoT component – to improve system readiness and reduce energy consumption
- Include battery charging strategies as described in [12]
- Create an easily adaptable software model that will allow node behavior change without installation or restart – to improve both maintainability and energy consumption
- Support runtime changes of the working modes and make the system highly responsive to the update requests
- Integrate the node into the digital twin to make the complete system more controllable

All the requirements align with designing the IoT node with a higher readiness, better maintainability, higher stability, and lower energy consumption. This paper presents the results achieved in line with these guidelines. Section 2 represents a review of the research whose concepts were adopted and updated during the development of the IoT node. After that, hardware and software designs and energy management are elaborated in the materials and methods sections. In the section Results, measured and estimated values are compared with the expected energy levels suggested by the default designs to document used components. Ultimately, all benefits, challenges, and suggestions for further research are pointed out.

2. Related Work

This research aims to define the energy and process-efficient IoT node that should work in hazardous areas with contradictory requirements by exploring advances in hardware and software. Analyzing energy usage, the IoT node spends some power during standby, some when collecting data, some when processing them, and finally, when transmitting to the Edge level. Since energy reduction could be achieved in every step, we checked many studies to create a promising approach for the overall node design.

Study [16] advocates using power-saving modes and introducing execution cycles with multiple sleep modes. This approach is constantly evolving, and [17] further introduces a complex model of sleep states where each is used in separate process steps. In [18], the advantages of decentralized IoT architecture were pointed out. We considered this concept when developing general-purpose nodes that can perform different roles by employing different software setups. This is in line with the recent findings presented in [19] where one of the main recommendations is to create IoT networks based on the lowest possible number of node types and processes.

Looking at the software side of the design, we focused on two main aspects: building a highly adaptable software model that could be easily extended and employing control mechanisms that could reconfigure real-time execution by changing the control flags. We accepted the idea behind the task allocation algorithm to reduce the time required to process the high workload in IoT [20]. The control process sets up a set of activation flags that activate only necessary parts of the processing loop in specific loops. The same principle was used when we tried to optimize the data processing routine and the size of synchronization queues in runtime.

The study [21] brought the dependency inversion principle when the driver routines for new sensors are developed. With this approach, the data collection part of the program could be developed faster and with significantly fewer changes in the complete system. Since our IoT node tends to be as general as possible, this approach enables flexibility when integrating new sensors. The mentioned work brings a complete energy-efficient framework based on several more design concepts, which could be obtained only up to some portion due to different programming paradigms used in the current software design of the suggested solution.

The contribution of the previous study is also by raising awareness of general energy consumption reduction through software design. The research [22] initially raised the attention of so-called energy bugs and hotspots resulting from the software design and scheduled task execution. Further research from the same authors [23] provides a deeper analysis of inter- and intra-task energy hotspots, with use cases and guidelines for minimizing their impact. The suggestions are integrated into the primary execution model and battery charging algorithms, similarly as suggested in [24]. They have been implemented in the presented solution by decoupling the data collection and processing from the data transmission routine.

Data transmission is critical since it uses a sizable part of the energy. The studies [25, 26] give us an insight into the expected power consumption modes for the data transmission phase when different scenarios and technologies are deployed. The general suggestion is to keep transmission devices in the lowest possible energy regime as long as possible. In the ideal case, the suggestion is to keep transmission equipment in sleep mode for more than 99% of the time, regardless of the technology used.

For primary data transmission technology, LoRaWAN (Long Range Wide Area Networking) was a choice for our solution due to a higher transmission range and a longer battery lifespan compared with similar technologies [27]. LoRaWAN is usually not the first choice for the data transmission mechanism. Bluetooth-enabled devices are considered a standard solution, but their limited range could not be used as communication components in the expected exploitation conditions. However, "design principles for selecting hardware components subject to varying environmental conditions and application requirements" are inherited from [28]. An excellent example of the usage of LoRaWAN technology is presented in [29]. It describes the IoT node used in water management systems. The presented node works outdoors and has proven to use LoRa (Long-Range) technologies for its reliability and excellent power consumption rate.

The IoT nodes are intended to work as a part of a more comprehensive system, and it is necessary to define the environment that would allow fast recovery when the IoT node needs to get refreshed or reconfigured. Firstly, the set of recommendations for the software update processes in different IoT levels has been defined [7]. It was followed by the establishment of a digital twin structure, which was recognized as a need to support development and testing and later support when the system was in active usage [10]. During the research, dark launch expanded with feature flag deployment, which looked interesting, with the possibility of a broader application [30]. It was based on the concept that specific software features were enabled or disabled based on the value of the corresponding flags. The feature would be active only when the flag was set. The flag could be set or reset through the external interface, and the software behavior could be changed without restarting or reinstalling. Based on the feature flags approach, we designed the ESP32 node's main loop and all other software tasks.

The paper [31] describes a highly scalable solution that organizes IoT nodes for monitoring hazardous areas. It envisions a case where the set of static IoT nodes is active simultaneously with the set of mobile nodes and where the network can perform self-healing up to some point. The next crucial point in the research [31] is an effective alarming process. The research defines the concept of "smart alerting for potential hazard avoidance." The design rules and the algorithms for raising alarms were adapted when system parts reported problematic values, switched to backup routines, or stopped responding.

IoT nodes based on ESP32 microcontrollers whose communication part is based on MQTT (Message Queuing Telemetry Transport) protocol are proven as a choice that could support heavy computational requests. The research presented in [32] demonstrates the usage of such a combination in the system dedicated to monitoring self-generated energy during trading activities based on the Ethereum blockchain, which makes it applicable for sensor network support.

The security in such systems is not at the highest possible level, and future work will focus on this. Currently, the developed system relies on the standard security features integrated into used components and protocols. According to [33], this is assumed to be a potential security concern. Compared to other computing devices, IoT nodes have lower processing power, so specialized countermeasures against network attacks should be designed [33]. Furthermore, the research presented in [34] explains all negative aspects of the MQTT-SN (Message Queuing Telemetry Transport for Sensor Networks) protocol in detail.

When it comes to energy management, the second part is charging strategies. In [35], the authors discussed traditional charging control methods, such as constant current, voltage, pulse charging, and software-enabled battery management systems. We used some principles of fuzzy logic charging as the extension of standard threshold-based charging, such as an adaptive standard low threshold. The approach presented in [35] that we found interesting is the predictive control model of energy storage systems. The study presented in [36] explains 26 different battery charging strategies. This was important to us since it explicitly focused on the charging characteristics of Li-ion batteries. It comprehensively explains controlled features, cut-off conditions, and observed parameters. The suggested multi-step-ahead predictions based on accumulated parameter values would help determine the right time to start charging. This approach was a base for our alarm-based and controlled charging scenario.

With the anticipated growth of battery management systems by more than 50% annually until 2030 [37], this research area is considered highly important and with the expected high-level improvements. This research also indicates the importance of machine learning and building an adaptive battery management system that should consider multiple parameters for their operations.

Table 1 The main features of similar solutions from literature

Feature	**WaterGrid-Sense [29]**	**E-Nose application [38]**	**Fire detection [39]**	**Presented solution**
Transmission protocols	LoRaWAN	LoRaWAN	LoRaWAN, GPRS optionally Wi-Fi and Bluetooth	LoRaWAN, GPRS optionally Wi-Fi and Bluetooth
LoRaWAN protocol class	A	Probably B, based on the model	Probably B, based on the model	C
Sensors	Fixed package of two sensors	N-IGSS sensor node	Maximum 4 per device, various	Maximum 4 per device, various
Processing unit	Microchip, non-specified model	ESP32	ESP32	ESP32
Battery	3.7V, 1000mAh	Battery, non-specified	Battery, non-specified	3.6V, 3500mAh
Battery charging	Whenever sunlight detected	Not Implemented	Not Implemented	Adaptive charging algorithm
Power option	External solar panel	Possible installation of solar panel	Possible installation of solar panel	Integrated or external solar panel

Looking at the literature, many IoT-based solutions based on a single node can be found. The most similar that we could identify are Water-Grid Sense [29], E-Nose application to detect pollution hazards [38], and forest fire detection system [39] (Table 1). All these solutions are based on LoRaWAN as the primary communication channel. The fire detection system and our solution include a GPRS module as the backup channel. Forest fire detection solutions anticipate higher energy consumption due to the higher usage rate of GPRS; thus, they work at a much higher voltage level than others. E-nose and fire detection applications did not focus on effective battery management but higher-volume data usage. Regarding dimension, Water-Grid Sense is the smallest device, but it uses a fixed package of two sensors optimized for low consumption. It encloses a smaller battery and, as with our system, comes with a charging module. The difference in favor of our solution is that we use an adaptive charging algorithm that ensures longer battery life. At the same time, Water-Grid Sense charges the battery whenever sunlight is detected. The option of the external solar panel is available in all solutions. Water-Grid sense theoretically could use an internal solar panel as our solution, but currently, this is impossible since their casing is the smallest possible.

To create an energy-efficient IoT node dedicated to the specific setup, we had to support a complex co-design, including hardware elements, execution mode adaptation, new software design and update principles, and the definition of an adaptive battery charging approach. Referenced work exposed brilliant ideas but primarily focused on a single area of interest. At the same time, we aimed to combine all available techniques to make the IoT node as energy-efficient as possible.

3. Hardware Design

As the introduction summarized, the main direction of the design process was to create an IoT node based on standardized and worldwide available hardware components. The solution should be solar-powered, battery-based, and equipped with some wireless data emission device to integrate with higher levels. To reduce energy consumption, the IoT system should be based on a hardware platform that enables active and hibernate/sleep mode work. The node must be able to alternate working modes periodically or as the result of specific signals. In the active mode, it should periodically check sensors, read and process sensor data, and then send the retrieved values to the upper level. Further, the selected components must have enough processing power, a standardized operating system, and data storage capacity to integrate into the digital twin and enable remote diagnostics and control.

3.1. Hardware Components

The market offers several microcontrollers that could act as the core for the IoT nodes. Considering previous requirements, as the base component for the designed IoT node, the ESP32-WROOM-32 SoC module has been chosen [41]. It is widely used in industrial environments, and its modular design (Figure 2) supports work in different operation modes defined by the states of internal components (Table 2). Its processing unit consists of two central ESP32 cores and an ultra-low-power coprocessor (ULP coprocessor), which controls work in sleep mode. The ULP coprocessor is further supported with a real-time clock memory (RTC memory), primarily used for saving and keeping values during sleep mode. This memory allows active sensor data collection while two execution cores are inactive. The connectivity part of ESP32 consists of the wireless radio, Wi-Fi, and Bluetooth modules. For our design, integrated network modules were not adequate. To make ESP32 usable in the off-grid setup, these modules should be based on protocols with a much higher communication range, such as LoRaWAN and GSM (Global System for Mobile Communications). Integrated Wi-Fi and Bluetooth could be used in a production plant environment, but when it comes to the range and energy usage, they are not appropriate for remote areas. To keep the data exchange secure, ESP32 has integrated IEEE 802.11 standard security features, secure boot flash encryption, and essential power management to ensure the component's sleep mode activity. These basic features ensure enough security to be integrated with digital twins and to be updated OTA.

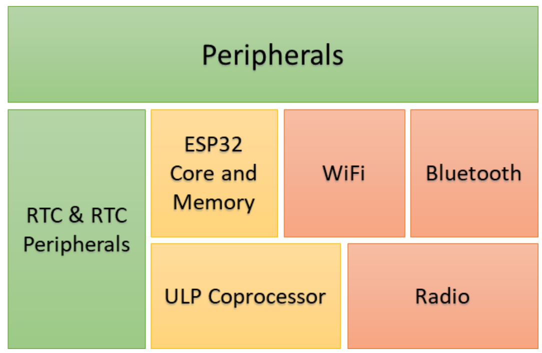

Figure 2 ESP 32 - main building blocks

Table 2 ESP32 – comparison of active components in standard modes

Component	Active mode	Modem sleep	Light sleep	Deep sleep	Hibernation
ESP 32 cores	+	+	paused		
RTC memory	+	+	+	+	+
ULP Coprocessor	+	+	+	+	
Radio, Wi-Fi, and Bluetooth	+				

Alongside network communication components, ESP32 offers a powerful peripheral interface set that supports data collection from other hardware devices and sensors. Two interfaces are supported in this category: I2C and RS485. ESP32 natively supports I2C and comes with dedicated pins and communication routines. RS485 is a bit more critical for communication and usage in hazardous areas. It is a protocol that supports asynchronous serial communication with multiple devices and is suitable for industrial environments since it can connect to 32 devices with a cable 1200m long. It is less prone to electrical noise.

Aside from ESP32, a few more components were necessary to complete the IoT node. The protected lithium-ion battery of type 18650, with a capacity of 3500mAh and working on 3.6V, was chosen. The battery is supplemented with a charge controller and an adequate solar panel. Supporting the battery charging process is critical for such nodes, so the chosen solar panels must be strong enough to enable successful recharge.

The complete hardware design – ESP32, battery, GSM unit, LoRaWAN module, charger, and optional solar panel- are combined as a single device and enclosed in the proper casing, certified for use in hazardous areas (Figure 3). Since the GSM and LoRaWAN modules are used because of their range, the choice of ESP32 microcontroller was a bit challenging. In the market, many similar devices, including support for I2C and

RS485, could be considered good candidates for the base component. Table 3 shows a brief comparison of their most essential features.

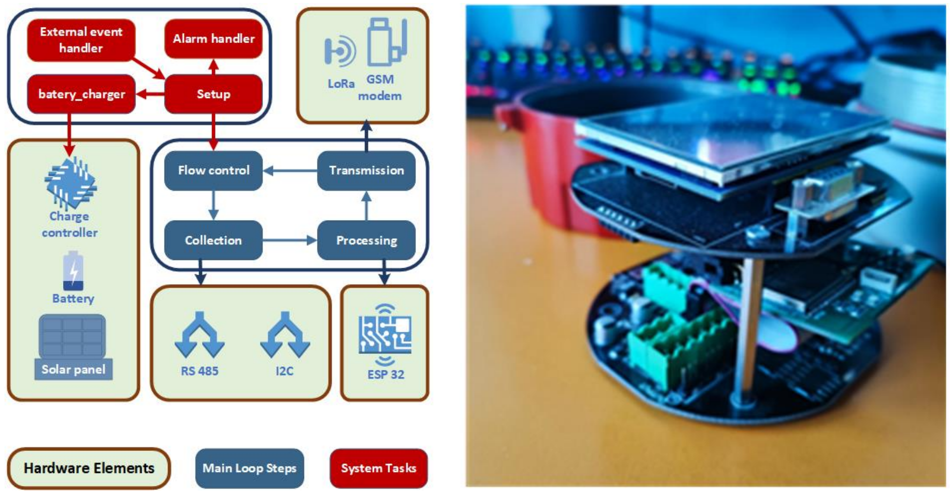

Figure 3 IoT node for hazardous areas – left [13]: schematic display with interaction between software and hardware elements, right: the look of the assembled device

Table 3 Comparison of ESP32 and similar microcontrollers (extracted from [43])

Controller	Clock Speed (MHz)	Flash Memory (MB)	Maximal Operating Voltage	Price ratio (against ESP32)
ESP32	240	4	3.6	1
Raspberry Pi Pico	133	2	5.5	1
STM32	480	2	3.6	3
Arduino Nano	16	0.03	5	2
Teensy	600	8	5	3.5
nRF52840	64	2	3.6	2

ESP32 is one of the cheapest chipsets in the market and offers worldwide support with a strong and responsive community. There are faster components like STM32 and Teensy, but they are more expensive. ESP32 is second best in memory capacity and third in the clock speed category, but it is the cheapest and works at the lowest voltage level. In that light, it is also one of the components with the lowest energy consumption. The advantage of Raspberry Pi, STM32, Teensy, and nRF52840 is that the ARM architecture offers the base for more advanced software and hardware platforms, but with the current setup, taking into consideration all the mentioned aspects (speed, data capacity, energy consumption, and support community), ESP32 has been considered as the optimal choice.

3.2. Working Modes

The mode when all components are running is considered active, while all the other modes are considered sleep modes (Figure 4). In active mode, the controller has maximal processing power, and all communication means are active. Consequently, it uses the most possible amount of energy and should be rarely used in configurations when energy efficiency is the primary goal.

Figure 4 Comparison of active elements in ESP32 standard working modes and Controlled Active Mode

Each sleep mode has a distinct set of active components. In modem sleep, peripherals and communication elements are disabled, while core and memory are active with the ULP processor and RTC and RTC peripherals. Modem sleep is used when the node actively collects sensor data and processes them locally without uploading them over the network. This mode had the potential for standard use but was not adopted because no external control was possible. The light sleep mode is designed to spare more energy since the core and memory are paused. It allows fast wake-up upon the signal's arrival or after the timer has elapsed. Its intended use is when the node only collects data from the sensor array.

Deep sleep and hibernate modes are intended for use when a node is in the state when waiting for the following command but with the ability to change its state as fast as possible. In deep sleep mode, RTC parts and ULP coprocessors are only active, waiting for the signals from the sensors. In hibernate mode, RTC is the only part that stays active. So, in hibernate mode, everything is shut down in the node, and the node will wake up only after a predefined time.

The working modes described are native to ESP32, and switching between them is fully supported. Since the device spares significantly more energy when in active mode, keeping the active mode as short as possible and switching between appropriate sleep modes when necessary is essential. Keeping the node in the lowest sleep mode will significantly reduce energy use.

However, for our implementation, we needed to slightly modify the mode system and introduce a new working mode – the so-called controlled active module (CAM). CAM is intended to replace active mode, modem sleep, and light sleep mode. The main idea is to switch off the complete network communication subset in ESP32 since they are not used. At the same time, the peripherals block will be kept active, allowing communication using external components and enabling the node to communicate with other pieces of software. The activity of processing cores could be controlled through the software routines, enabling the fast change of the state of active components. With this approach, the node will have processing cores active for more time compared to the default active mode for the same amount of energy.

3.3. Communication Channels

As stated before, the ESP32's communication channels had to be disabled because of limited range and high energy consumption and replaced by LoRaWAN and GSM modules. Considering all the previously described criteria, the LoRaWAN was the best fit for the design. It defines the communication on the network level and supports the protocol, which runs on the physical level and provides data exchange over long distances. Overall, the LoRaWAN technology stack positively impacts the battery lifecycle, network capacity, quality of service, safety, and security. It ensures stable bidirectional low-speed communication between mobile devices and offers the possibility to develop specialized and localized services. The data transfer speed is between 0.3 and 50kbps, which is assumed to be a compromise between the connection range and the maximum message length [44].

The main drawback is that the communication under the LoRaWAN protocol does not support data exchange between IoT nodes or other terminal devices. It supports communication between IoT nodes and LoRa gateway devices and vice versa. In LoRaWAN networks, it is possible to have three categories of node devices: A, B, and C. Only class C, or bidirectional end devices, has been considered for the presented node design. After every data package has been sent, the class C device has two short message receive time windows.

Since the IoT nodes run in off-grid areas, they must have a backup communication channel. When the LoRa channel gets interrupted or out of use, the node must be able to continue sending collected data. The backup channel was realized on a SIM-based (Subscriber Identification Module) GPRS/UMTS (General Packet Radio Service/Universal Mobile Telecommunications System) connection.

The system automatically switches to backup communication when the primary channel gets disconnected. Communication in the backup channel is much more expensive since it requires a billable connection via a mobile network operator. The added cost is related to energy consumption. The GPRS/UMTS module uses more energy for its work than the LoRa devices. For this reason, the switch to the backup communication channel is the automatic switch to the alarm state. If the main channel becomes operative again, the system automatically switches back to the LoRa connection and returns to normal operation mode.

4. Software Design

The software component of the IoT node design is developed on top of the FreeRTOS [47] operating system. It is compatible with and supported by an ESP32 microcontroller. Its main advantage is that it fully supports multitasking, catering to the latest requirements of IoT devices.

4.1. Software Processes

The software implementation of ESP32-based nodes is designed around the main task: the core revolving routine. It could call other tasks for execution, and their number is not limited. Additional tasks can either be controlled by the main task or triggered in response to specific environmental signals. The main task consists of five steps (Figure 5), where each step calls specific tasks:

- **Flow control** is responsible for reading configurations and setting up process flags and parameters, making the main loop go only through the necessary steps.
- **Setup** facilitates the configuration of control flags and enables or disables specific aspects of the system. It is responsible for switching between execution nodes, managing the update process, and reporting data back to the digital twin
- The **collection** step manages communication with sensors and retrieves measured data.
- **Processing** is where collected data are verified and packed into synchronization objects. The created objects are then placed into synchronization queues and prepared for transmission.
- **Transmission** is when prepared synchronization objects are dequeued and sent through the network using appropriate communication.

Various tasks are implemented in every step to facilitate the IoT node's operation. These tasks fall into three main categories: setup and maintenance (indicated by red graphic elements in Figure 5), data processing (light blue elements), sensor communication (green elements), and data transmission tasks (amber elements). Namely, as explained in detail in [13]:

- The **all_param** task encompasses a set of routines and data structures responsible for managing system setup parameters.
- The **battery_charger** task monitors the battery level and controls the charging procedure, ensuring the IoT node maintains sufficient power for uninterrupted operation.
- The **external event handler** is the gateway for controlling the external network. It is responsible for receiving and processing commands from the cloud or

other controlling devices and forcing processes such as OTA updates, immediate battery charging, or a change of the execution mode.

- The **alarm handler** raises alarms when specific parameters reach predefined critical values. As a result of its action, the node could go to the hibernate node, or communication with a faulted external device could be terminated.
- **I2C_comm** and **RS485_comm** facilitate data exchange between the IoT node and connected sensors using one of the protocols. They ensure efficient communication and promptly support exchange routines.
- The **GPS_comm** task handles communication with the GPS (Global Positioning System) module. Accurate device positioning is crucial when the node is installed on a moving object, such as a barge transporting crude oil in rivers.
- Processing step runs **data_pack** and **telemetry_pack** processes. They are responsible for packing sensor readings (data_pack) or node's status parameters (telemetry_pack) into synchronization objects.
- The **MQTT_SN_comm** task manages the synchronization queue's capacity and occupancy. It coordinates write processes from data producers and read processes from data consumer tasks.
- **LoRa_comm** task supervises communication between the IoT node and the Edge computer using the LoRaWAN protocol.
- **GSM_comm** task oversees the backup communication channel between the IoT node and the Edge computer.

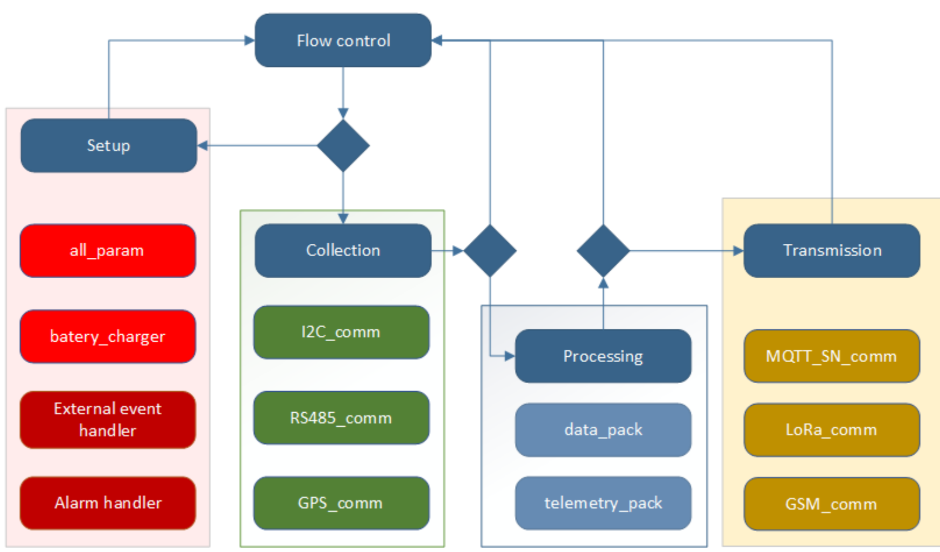

Figure 5 Main loop and support tasks running in the realized IoT node (as in [13])

4.2. Message Protocols

Devices at the Edge level are considered much more potent than IoT nodes and can run more advanced software and communication equipment. This led to choosing the correct communication protocol focused on data delivered to the consuming Edge devices not by their network addresses but as a function of their contents and interests.

The IoT node and Edge layer communication is realized using the MQTT-SN (MQTT for Sensor Networks) protocol (Figure 6). It is a sub-variant of MQTT modified for the wireless communication environment, characterized by low bandwidth, high link failures, and short message length [46]. Since MQTT-SN is perfected for low-cost, battery-operated devices with limited processing and storage resources, it could fully support the IoT node's hibernate mode and the LoRaWAN class C protocol.

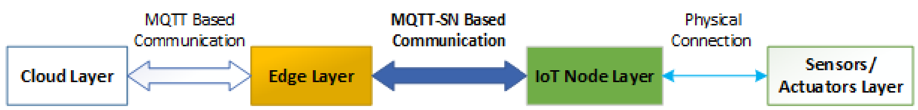

Figure 6 Place of IoT nodes in broader ISA-95 technology stack and data exchange means between layers (as introduced in [14])

The connection between Edge and upper levels could be fulfilled using MQTT, which is an open and lightweight publish/subscribe protocol designed specifically for machine-to-machine and mobile applications [45]. The MQTT protocol is adequate since a stable wired connection connects the Edge and cloud levels. Since variants of the same protocol are used across the entire system, the whole structure has certain advantages in system response to hazardous events, overall system reliability, data security, traffic reduction in the Edge-client connection, and the background for introducing digital twins.

4.3. Task Synchronization Mechanism

The management of configuration parameters within the FreeRTOS environment relies on established and widely recognized mechanisms. Specifically, semaphores regulate access to shared resources and effectively facilitate data exchange among tasks. To improve efficiency, the IoT node uses internal synchronization queues (set up as the internal variables in all_param tasks) between collection and processing and between processing and transmission steps. This way, steps that consume less energy could be performed several times before the next step, which consumes more energy, would run. With this approach, energy consumption in controlled active mode could be further reduced.

As previously elucidated, the primary objective of the IoT node centers around capturing data from sensors via RS485 or I2C interfaces. Periodic data retrieval occurs concurrently through the RS485_comm and I2C_comm tasks. These tasks write data to the same message queue, guarded by semaphore. Consequently, data processing could remain dormant until the queue is filled up and only switch to an active state. Once the buffer contains enough data, the loop task proceeds with data validation and processing. The processed values are then written in the message queue for transmission to the edge level.

This process is supported by I2C_comm and RS485_comm tasks. They execute concurrently and write the values they read from sensors to the same message queue. At the same time, task MQTT_SN_comm reads the items from the queue and prepares them to be sent to the cloud (Figure 7). Using the three tasks mentioned, the semaphore approach avoids eventual read/write hazards during concurrent access to the mqtt_msg queue. Every task that should access the message queue waits until it is free and only enters the critical section. The task releases the message queue when the read or write is done, and the next task can access it.

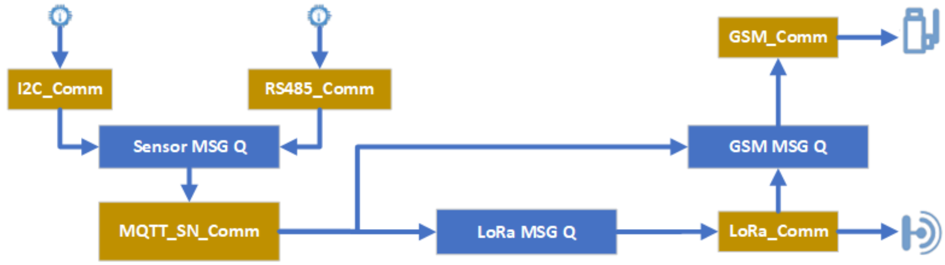

Figure 7 Data flow from sensors to transmission elements through message queues and processing tasks

Message queues are also used for data transmission, one for LoRaWAN and another for the GSM module. The LoRa message queue does not need synchronization since each data producer has only one data producer and consumer. On the other hand, the message queue dedicated to the GSM module must be synchronized in the same way as the message queue used for data collection from the sensors. It can receive data directly from the processing step or data that failed to be sent using LoRa_comm.

5. Battery Charging Routines

An ideal energy consumption scenario involves standardized functionalities that maintain consistent energy usage levels over an extended period. However, practical constraints often prevent such ideal conditions [36]. As previously discussed, different data transmission devices exhibit significant variations in energy consumption. For instance, scenarios involving updates or lost connections to sensor devices result in increased energy usage beyond the baseline. Furthermore, distinct active and sleep modes consume varying amounts of energy depending on the volume of workload nodes have to perform. Also, transitions between modes can trigger consumption peaks if specific initialization procedures are required. As outlined earlier, energy usage during node operation depends on the working mode and the frequency of necessary actions.

When evaluating data usage across the three phases of the node's cycles, data processing and data collection use a similar amount of energy. Compared to data transmission, data collection and processing use much less energy. Data transmission modules exhibit substantial differences in range, speed, and data package volume, but in any case, data transmission remains the most demanding energy task [37-40]. The battery's energy level should always be adequate to ensure proper node operation fitness. For this reason, a separate set of routines is developed and integrated into the IoT node's software model. It is intended to drive the charge controller and execute chosen charging strategies.

5.1. Automatic Charging

The charging process periodically checks the battery's energy level in the automatic charging mode. It starts if it reaches a standard low battery level (SL). The node continues its operation while the battery is charging, and when it reaches a standard high level (SH), the charging process stops. The charging controller is a separate component and does not affect the work of any other IoT node element. This approach could be problematic when the node's charging routine depends on solar power. Sunlight is available at most 50% of the time, and the periods of active sunlight are not constant. Furthermore, the effect of the other natural elements and construction properties of the device could reduce the period of sunlight exposure.

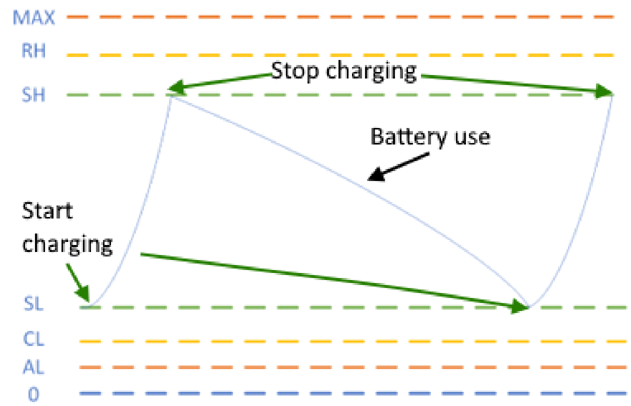

Figure 8 Ideal consumption setup with automatic charging mode

Whenever the charging controller starts or stops the charging process, it sends this information to the edge level using the telemetry call with a timestamp. These data are collected at higher levels and used to analyze node functionality and act as a base for future improved charging modes. They could also be used to identify malfunctions early. The default charging process, if applied constantly, is envisioned to ensure longer battery life. The best use case for most available battery types is if their power level varies between SL and SH thresholds, following the process as presented in Figure 8 .

5.2. Alarm-Based and Controlled Charging

An automatic charging scenario is not always possible. First, it could be triggered at night or when the sunlight is not bright enough. Then, the solar panel will not generate enough power to raise the battery's energy level. When charging starts, but the energy level is still going down, the alarm signal from the IoT node will trigger. The signal will be received and registered at the edge level. Since the charging controller frequently reads the battery's energy level, it could continue to trigger alarms that indicate that the energy level is still reducing despite initiating the charging process (Figure 9, block “Report charging issue”). If the energy level continues to reduce, it will eventually reach CL (Charging Required Level). At that moment, the IoT node will send a higher priority alarm to the Edge computer and reconfigure its operation strategy by reducing the number of data transmission operations. If the battery level continues to degrade, after some time, it will reach the alarm low (AL) threshold (Figure 10, left). This is considered the highest-level alarm, and the node will stop all its operations and switch to hibernation sleep mode. Up to that time, based on the data received in the Edge and then forwarded to the cloud level, the operation engineers could decide what to do with the affected IoT node.

One of the simplest ways to prevent this situation is to enable the calculation of the energy use depending on the time of the day and the introduction of an additional method that will check if the charging process should start (Figure 9, block "start charging," line 28). SL would be increased by some percentage (like 10 or 20%). In this case, the charging routine will check the remaining time until sunset and the increased SL. If the energy level falls to SL+10% and the remaining period of the day is, i.e., 10% sunlight, the charging process will start immediately. This simple and effective approach allows for additional charging periods with the lowest possible effect on battery life. The problem with such an approach is that the node must have daily information about sunrise and sunset and run more complex checks.

```
01    while (flag_charging_active)
02    {
03        decimal batteryLevel = ChargingController.ReadBatteryLevel();
04
05        if (flag_charging_running                                  Report charging issue
06            && batteryLevel < previousBatteryLevel)
07        {
08            Telemetry.NotifyEvent("DrainWhileCharging", DateTime.Now, Alarm.Amber);
09
10            if (batteryLevel < CL)
11            {
12                Telemetry.NotifyEvent("DrainWhileCharging", DateTime.Now, Alarm.Red);
13                Reconfigure(reduction * TransmissionRate);
14            }
15
16            if (batteryLevel < AL)
17            {
18                Telemetry.NotifyEvent("DrainWhileCharging", DateTime.Now, Alarm.Stop);
19                ChangeMode(Modes.Hibernate);
20            }
21            continue;
22        }
23
24        if (flag_forced_charging
25            ||                                                 Start charging
26                (!flag_charging_running
27                && (batteryLevel <= SL
28                    || ShouldStartCharging(batteryLevel, DateTime.Now)
29                    )
30                )
31            )
32        {
33            ChargingController.StartCharging();
34            flag_charging_running = true;
35            Telemetry.NotifyEvent("StartCharging", DateTime.Now);
36        }
37
38        if ((flag_charging_running                             Stop charging
39            && batteryLevel >= SH)
40            || (flag_forced_charging
41                && batteryLevel > RH))
42        {
43            ChargingController.StopCharging();
44            flag_charging_running = false;
45            Telemetry.NotifyEvent("StopCharging", DateTime.Now);
46        }
47
48        previousBatteryLevel = batteryLevel;
49    }
50 }
```

Figure 9 Charging controller routine incorporating alarm-based and controlled charging (pseudocode)

The charging controller's next operation mode is the controlled mode. This mode is initiated from the edge level and intended to instantly trigger the charging process. Regardless of the current battery level, the charging process will start immediately when the control signal is received and the *flag_forced_charging* is set.

The mentioned control signal is followed by the requested high level (RH in the further text); the battery will be charged until the requested level is reached, regardless of the value set for SH (Figure 10, suitable; Figure 9, block "stop charging," line 41). This process does not change the SH level but is omitted during a single charging run. When the battery level reaches RH, the charging process stops, and the node returns to the alarm-based mode. The battery could lose power in the controlled charging mode, as in the automated charging mode. In this case, the same alarm procedure will run. Eventually, the charging controller could be disabled by setting *flag_charging_active* to *false*. This happens regularly when the IoT node is connected to the power grid, but this situation is outside the scope of our paper.

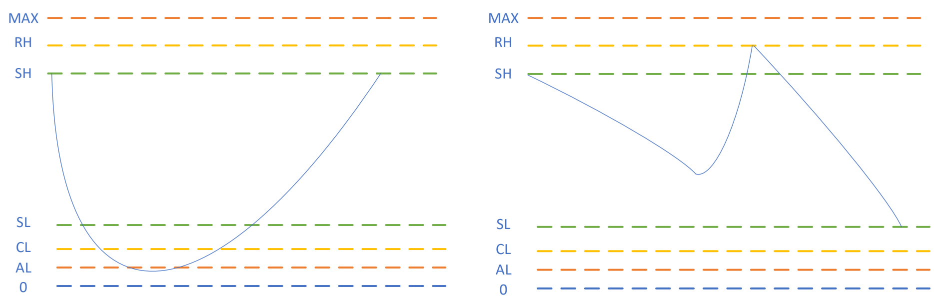

Figure 10 Battery recharge after the intensive drain (left) and the battery charging in controlled mode (right)

5.3. Short Term Improvements

As explained, making the charging process more adaptive and efficient is essential. Considering that the transition to controlled charging mode with the predefined RH could be triggered from the higher levels at any time, bringing a dose of safety, the process will be automated to ensure less frequent (ideally never-happening) situations when the IoT node goes to the alarm state. The charging controller regularly reads the battery status and uploads (and stores locally) these data for further analysis. The average energy consumption per hour (ACH) is calculated based on this. Since the node reads data from the sensors during standard periods, the actual energy consumption could be an additional input for deciding when to start charging.

The next improvement will be for the method running in the node that decides when to start charging (Figure 9, block "start charging," line 28). The update method will calculate the sum of SL and the value resulting from multiplying ACH by the number of hours until sunrise. If this sum is higher than the battery's current energy, the charging process could start immediately, significantly reducing the risk of the transition to the alarm state. Further improvements would include the weather report and checking if the potential period with less sunlight is ahead. This way, the charging process could run up to a higher threshold than SH, bringing the battery a higher operational period. It is important to note that the charging frequency depends on the battery capacity and the effectiveness of the solar panel and the charging component. With the standardized working mode, with two RS485 sensors attached and a LoRa module used for data transmission, our node will need one charge weekly or bi-weekly. This period is long, and the weather could change several times. Also, if there is a need to use more expensive energy, GPRS communication channel energy will be drained much faster. Thus, the possibility to react fast and run charging is a necessity.

6. Results

The proposed solution is based on the ESP32 series of devices with added communication and power supply components (Figure 11). The node is designed to be robust from the physical perspective, with easily reconfigurable hardware execution modes, and flexible from the software design point of view. Operationally, it should run using the lowest possible amount of energy while acquiring data from different interfaces. Since the system has not only the ESP32 but also other components, the measurement must be done in correlation with the entire system, not only the processor itself. The overall power consumption combines the consumption of sensors (the setup with two RS485 inductive distance sensors with a maximal 10Hz measuring rate), ESP32, and internal and external communication modules. The usual test setup was with 100 execution setup daily.

The measurement has been performed in the laboratory and simulated field conditions. The measured objective was the water level in the water tanks. We tested energy consumption in the laboratory with regulated temperature settings. In the simulated field conditions, we mainly tested battery charging routines. Simulated field conditions were performed at the rooftop of the Faculty of Sciences, Niš, Serbia, where solar exposure is somewhat average for Southern Europe – between 1.5 kWh/m^2 in January and 6.5 kWh/m^2 in July [48]. Since solar panels are usually certified for 1kWh/m2, the node is usually charged with the nominal current. The node ran constant readings from the sensors while the data processing and transmission frequency were controlled from the Edge computer. The node is automatically reconfigured when the battery level reaches critical values. Digital multimeters GDM-8255A [49] were used as measuring equipment in the laboratory, and UNI-T UT71C [50] for the fieldwork.

6.1. ESP32 Default Energy Levels

The default energy consumption data can be found in the related product datasheet [41]. The consumption analysis started with the measurement for the node based entirely on ESP32, where its internal communication modules are used. The software part is equal in this and the setup with the external communication modules, so the execution mode is assumed to be constant in the system. Internal modules are used only for the testbench since they are unsuitable for remote areas.

The values shown in Table 4 represent standard energy consumption levels measured in laboratory conditions and vary by some percentage compared to the values from the producer data sheet. Furthermore, some additional differences could be introduced due to the influence of connected sensors. In the examined case, the node was connected to different RS485-based sensors (Figure 11).

Table 4 Expected values for energy consumption in ESP32-based nodes [42].

Power mode	Description	Typical power consumption
Power off	CHIP_PU is set to a low level; the chip is powered off	0.1 μA
Hibernation	RTC timer only	5 μA
Deep sleep	From only RTC timer + RTC memory to ULP co-processor is powered on	10 – 150 μA
Light sleep	ESP32 core is paused	0.8 mA
Modem sleep	ESP32 core is powered	Slow speed:2-4 mA Normal speed: 20-25 mA Max speed: 30-50 mA
Active (RF working)	Receive - Transmit BT/BLE Transmit 802.11g Transmit 802.11b, OFDM 54 Mbps Transmit 802.11.b, DSSS 1 Mbps	95-130 mA 180 mA 190 mA 240 mA

6.2. Measured Values

As mentioned in the introduction, the opposing requirements for the designed nodes are that they should be as ready as possible and use the lowest possible amount of energy. In an important event, the node must immediately wake up, raise an alarm, and take the necessary action. Deactivating the data transmission part is how to keep the ESP32 active but use less power. This will not affect data processing and sensor connectivity, but the consumption will be lower in CAM mode, as defined in 3.2. With the new working mode, the node will be active in remote areas with lower power consumption compared with standard active mode and modem sleep. The complete execution setup includes switching between sleep modes and the CAM mode.

Figure 11 Finalized IoT node with one RS485-based sensor attached

As seen from Table 5, if the standard active mode were used, the lowest possible consumption would be at least 100 mA. The power consumption in CAM mode was up to 36 mA, while the modem sleep with active processing cores worked between 45 and 50 mA. This means that CAM mode could successfully replace parts of the processing routine where both active and modem sleep modes are running. The measured values for modes with active processing outdoors were close to lab measurement, with a difference of not more than 10%.

Table 5 Comparison of measured values for the IoT consumption (Setup A – improved design with CAM and external communication modules, Setup B – design relying only on ESP32 internal modes and modules)

Process	**Operation setup**	**Setup A lab (mA)**	**Setup A field (mA)**	**Setup B lab (mA)**	**Setup B field (mA)**
Light sleep + Sensors	Light sleep ESP32 core is paused	7.5	8.4	7.8	8.5
Data processing only (active mode)	Setup A – CAM Setup B – Active mode	32	36	>100	>100
Data processing only (modem sleep)	Setup A – CAM Setup – Modem sleep	32	36	50	50
Collection + Processing	CAM/Active mode + 2 RS485 Each RS485 < 20 mA	69	72	149	160
Transmission only (worst case)	Setup A: GSM Setup B: Wi-Fi DSSS	480	412	270	290
Full cycle (standard case)	Setup A: CAM + Sensors+ LoRaWAN Setup B: Active Mode + Sensors + Wi-Fi	98	104	200	200
Full cycle (worst case)	Setup A: CAM + Sensors+ GSM Setup B: Active + Sensors + Wi-Fi DSSS	560	524	430	460

The subsequent measurement is to connect sensors and measure the energy spent for data collection and processing at once. The sensors are connected to ESP32 through the RS485 interface. In this case, the total measured power consumption in CAM mode is 69 to 72 mA. The active components are ESP32 and two RS485 sensor arrays, whose consumption level is a maximum of 20 mA per sensor. In this case, the computed consumption was 36 + 2x20 = 76 mA. Still, the measured values remained around 70 mA in the laboratory and just above this level in simulated field conditions (72 average, 78 mA max). Compared to standard ESP32 active mode, the difference is significant, where consumption is usually at 150-160 mA but could hit 200 mA if unoptimized software loops are used.

The collection-only scenario was checked when the ESP32 was put into light sleep mode. The node in light sleep mode with attached sensors uses around 8 mA regardless of the scenario. The measurement in field conditions shows an average energy need of less than 10% more. In the period when the node needs to perform data collection periodically, light sleep mode is the logical choice. The ESP32 core and memory will be paused, but with RTC components active, the node can react to requests. The consumption in light sleep mode is as low as 7.5 mA with a peak value of 8.5. The consumption of the ESP32 itself is about one mA (0.8 mA as per documentation), but, simultaneously, the battery should also power sensors on stand-by, thus the difference.

The following important measurement is the consumption level when all cycle elements run – data collection, processing, and transmission. In a setup with only ESP32 components as the transmission device, the Wi-Fi in SoftAP (software-enabled access point) or STA (station) mode is enabled. In this case, the total consumption reaches 200 mA (compared with 190 mA from documentation). The usage is at the expected level, yet another argument for using the CAM is against using the full active mode as much as possible. So, from the calculation, it could be concluded that the communication part of the ESP32, in the measured case, uses energy equivalent to 110 mA.

LoRaWAN is the communication carrier for complete cycle measurement with CAM mode. Specifically, as the communication part of the LoRaWAN module, SX1268 [51] was installed. It uses 22 mA for data transmission and five mA for data reception. As mentioned, the LoRa works in class C since the node must operate in active and sleep modes. The measured value for the LoRa communication, when data are taken from the message queue and emitted, is at the level of 28 mA for transmission and 6.4 mA for reception. The overall energy used when the complete cycle is active with the LoRa part is around 100 mA, significantly under 200 mA, measured if Wi-Fi was running (Figure 12).

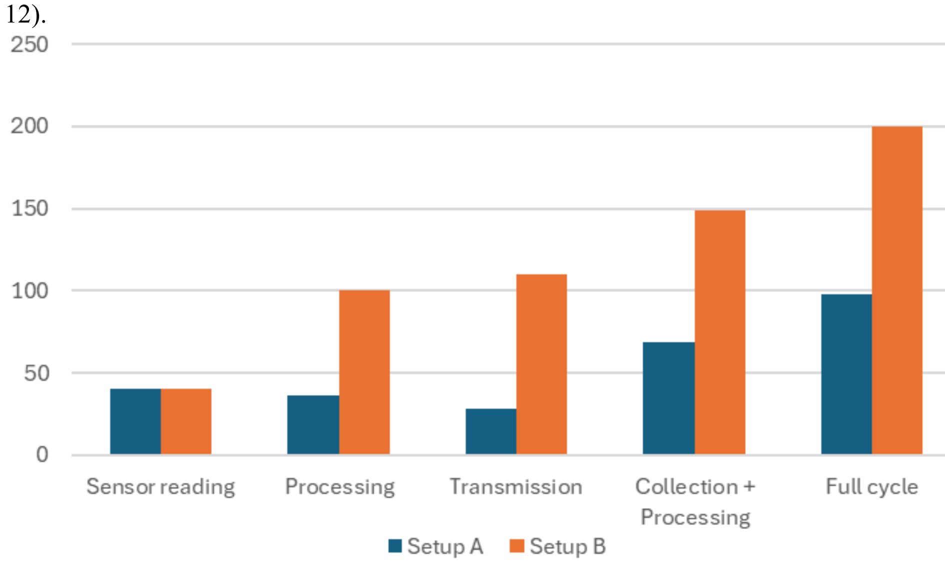

Figure 12 Comparison of energy consumption for proposed (Setup A) and standard (Setup B) configurations

In the case of regular use, the LoRa is more efficient than internal communication modules. In urgent cases, the system needs communication to contact the device outside the internal network. LoRaWAN or integrated Wi-Fi and Bluetooth will not be helpful when the communication is broken down. The GSM module is introduced to manage such an event. The consumption of the GSM module is significantly higher than anything else, and the maximal measured level in field condition was 412 mA (345 mA as in specification) when active and 21 mA when idle (19 mA as in specification). Measured values in the lab were higher (around 480 mA) because connection establishing takes longer. The used GSM module is SIM800H [52] with GPRS data mode (1Rx, 4Tx) on EGSM900.

Measured values are higher than specified but in the acceptable ratio. Setting up the connection could be the critical point in both LoRaWAN and GPRS data modules. It could take some time to execute, and the power consumption could be high during that period. The average of the GPRS module was 580 mA, while the theoretical peak could reach even 2000 mA. This fact is one of the reasons why introducing message queues and reducing the number of data transmission calls (when possible) is also essential.

When checking the complete cycle consumption with GSM, the average values are much higher than in any other setup. It was up to 560 mA in the lab, while outside reaches almost 530. Compared to GPS, the energy used in configuration with Wi-Fi running in DSSS mode was not more than 460. This is the only category where process-level updates do not bring benefits since the transmission part uses way higher amount of energy. This case clearly shows the importance of message queues and reducing transmission calls. The transmission mode could be adjusted to shrink the drawback of GPRS data module usage. Since the GPRS could manage a higher data volume, the system could decrease the number of transmissions and thus reduce overall energy consumption.

6.3. Consumption Analysis for Different Execution Modes

Measuring the energy consumption for the different elements of the IoT node offers a realistic overview of the energy consumption reduction rate. These values could also estimate energy consumption for various system configurations. By employing buffers, the number of data processing and transmitting operations would be reduced, positively impacting the consumed energy level. Table 6 and Figure 13 show proposed energy-saving configurations and maximal measured values for every step in the process that will be used for estimate. In this case, the measurements have been done only in the laboratory.

Table 6 Maximal measured values (in mA) for every step in the node operation

System configuration	Sensor reading	Sleep1	Processing	Sleep2	Transmission	Sleep
A + LoRaWAN	40	-	36	-	28	8
A + Wi-Fi	40	-	36	-	110	8
A + GPRS	40	-	36	-	412	8
B + LoRaWAN	40	-	36	8	28	8
B+ Wi-Fi	40	-	36	8	110	8
B + GPRS	40	-	36	8	412	8
C + LoRaWAN	40	8	36	8	28	8
C + Wi-Fi	40	8	36	8	110	8
C + GPRS	40	8	36	8	412	8

The execution modes are named A, B, and C. The difference is in the usage of message buffers. In execution mode A, there are no buffers. Each data collection is followed by data processing and transmission. Operation mode B introduced a buffer before data transmission. This means the node will read the data, process them, and put them into the queue. Data will be sent to the Edge level when the queue is full. Execution mode C is the update of mode B and brings an additional buffer between data collection and processing.

The maximal measured value for the sensor reading segment was close to 40mA, which was used as the estimation value. For the processing part, the baseline value of 36mA was considered, while all sleep modes were calculated as having the top consumption level of 8mA. Transmission rates were acquired as 28mA for the LoRaWAN module, 110 for Wi-Fi, and 412 for the GPRS external module.

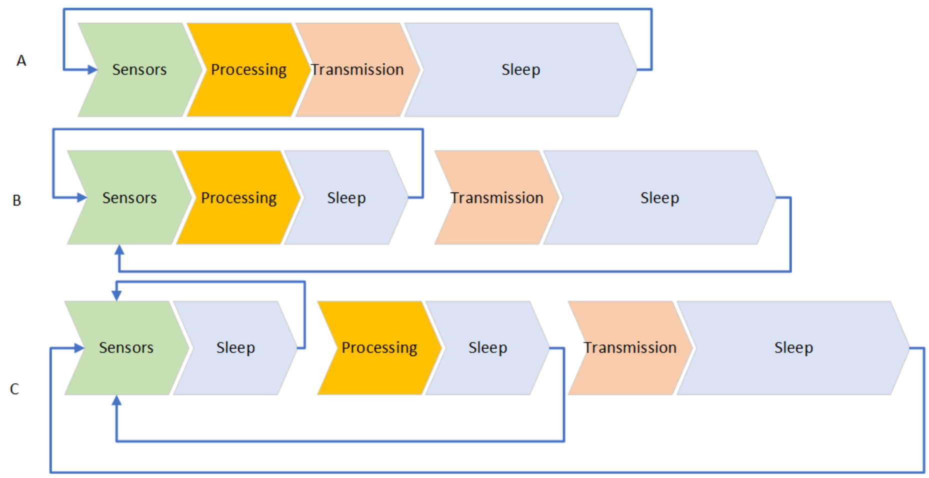

Figure 13 Different configuration variants supported by IoT node, derived from general state-based energy consumption model

The primary operation mode (Figure 13, A) is the sequence read-process-transmit followed by the sleep period. Depending on the current process or state of the overall system, the node could go either in the CAM or light sleep mode. This way, the node does not need to store any data locally and can go to sleep mode at the lowest cost possible.

Since the part of the process that consumes a considerable amount of energy is the transmission part, introducing a buffer before sending data to the Edge level brings the best gain. The node would wake up periodically, read sensor data, process them, and store them in the internal buffer (Figure 13, B). This will reduce the number of data transmissions every cycle. This is especially important when using the GPRS module since its connection setting-up part could quickly drain the battery. Note the difference in setup A with GPRS when the measured value of 61600 mA was much greater than the estimated 49600. It is partly due to indoor conditions, but the consumption is significant. More than five times compared with LoRaWAN and about 2.5 times with Wi-Fi.

With the buffer introduced between the data collection and data processing parts (Figure 13, C) sensors will read data periodically, pump them to the message queue, and the system will transit to sleep mode. After several iterations, the processing part will get activated. It will take the data from the queue, process it, and then store it in the queue before transmission. Data transmission will run when enough data gets stored in the second queue.

The analysis was based on 100 complete work cycles to provide a more comprehensive overview of the proposed solution's expected effect. The energy usage was lowest when the configuration variant C was applied, and the LoRaWAN was used as the communication module. The worst case from the energy consumption point of view was when strategy A was applied, and the GPRS was used for data transmission.

A comparison between these three variants is shown in Table 7. The estimate was calculated on the base of 100 sensor reading cycles. Comparing one variant, it is evident that the lowest consumption is in configuration with the LoRaWAN as a transmitting device. The difference is more significant in variant A than in B and C. The number of total transmissions is in direct proportion to the energy use, so the best effect is with the default operation mode. In variant A, the system with the LoRaWAN uses slightly above one-half of the energy used by the system with the ESP32 native Wi-Fi (50.64%). The energy usage is the highest with the configurations with the GPRS transmitter. Variant A uses more than five times more energy than the configuration with the LoRaWAN and more than 2.5 times more than the native Wi-Fi transmitter.

Table 7 Effects of proposed node configuration variants equivalent to 100 cycles

Configuration variant	Communication module	Estimated (mA)	Measured (mA)	Transmission count	Comparison with native setup (Wi-Fi)	Comparison with native variant (A)
A	LoRaWAN	11200	11800	100	50.64%	100%
A	ESP32 Radio	19400	23300	100	100%	100%
A	GPRS	49600	61600	100	264.38%	100%
B	LoRaWAN	8760	8820	10	88.47%	74.75%
B	ESP32 Radio	9580	9970	10	100%	42.79%
B	GPRS	12600	13800	10	138.42%	22.40%
C	LoRaWAN	5276	5282	1	97.87%	44.76%
C	ESP32 Radio	5358	5397	1	100%	23.16%
C	GPRS	5660	5780	1	107.09%	9.38%

Variants B and C have the most significant effect when the GPRS is used. Since the amount of time required for data acquisition is always the same, the number of data transmissions in variant B is reduced. In contrast, in variant C, further reductions are achieved by joining the processing part for 10 data acquisitions. In that way, in variant B, the data are transmitted only ten times for 100 reading cycles, and in variant C, only once. Variant C brings the most minor differences between configurations with different communication modules. It is on the level of 10% (107.09% vs 97.87%). For variant B, this difference is almost 50% (138.42% vs 88.47%). In variant C, the configuration with the GPRS uses less than one-tenth (9.38%) of energy compared to variant A. For the Wi-Fi as the transmitting module, the energy usage is reduced to a quarter (23.16%), and for the LoRaWAN-based configuration, it is close to half (44.76%).

This proves that buffer use is effective whenever possible, which means that the delay of transmitted data is not problematic for the entire system's efficiency in every case. By adjusting the count of cycles in the digital twin and pushing the update to the end node, the energy consumption could be adjusted in the node without physical access.

7. Discussion and Future Work

The primary purpose of the proposed system is to run in a remote and hazardous area as efficiently as possible. The system must operate on batteries and use every opportunity to reduce energy usage. To achieve this goal, the following set of improvements was realized over the standardized ESP32-based IoT node:

- The new active working mode will be introduced by disabling modules that consume high energy values.
- Define the transition to the adequate sleep mode, depending on the node’s usage cycle stage.
- Add external communication components that are more suitable for the expected use and have lower energy consumption.
- Enable redundancy whenever possible to make the system more dependable.
- Create an adaptive software model that will allow easy reconfiguration of the system’s working mode without needing restart or hardware replacement.
- Introduce data buffers between system segments and make the operation of the more significant energy consumers less frequent.

Having in mind the requested purpose, the designed IoT node must be not only energy efficient but also highly dependable. It should be able to supervise various errors, failures, and technical problems adequately. Hardware and software design modifications were implemented during the proposed node's work. Hardware-level interventions are mostly related to the installation of redundant parts – both sensors and communication lines. In that sense, the IoT node has two I2C and two RS485 communication channels, while the transmitting device based on the LoRaWAN is backed up with the GPRS module.

Regarding future improvement, the widest open point is data security. ESP32 runs with integrated IEEE 802.11 security for IoT nodes, but it has been proven that this level is not enough in every case. So, improvements in this area would be one of the future research directions. For the moment, an additional security measure is that access to IoT nodes is possible only through the Edge level or, in exceptional cases, through a device that has an authentication token provided.

The effect of the implemented updates is presented in Table 5. The node's power consumption is closer to modem sleep than active mode. This is expected since the communication part uses a massive portion of energy. With sensors enabled, measured consumption is around 70 mA, which is between one-half and one-third of the consumption when the ESP32 is active. When the complete system is operational, the consumption of the designed IoT node is about one-half compared to the node running on the ESP32 in fully active mode (98 mA vs 200 mA).

Improvements to the rest of the system are made at the software level. The crucial point was the implementation of setup routines that could directly influence the behavior of the main loop and change the execution variant of the node only by setting the feature flags. The control over these processes was moved to the cloud to create a digital twin. From this point, the updates could be directly passed down to the IoT nodes through the Edge computer. In that way, the control is centralized, and the status of each node will be successfully kept on the cloud.

Thanks to this feature, the node can easily switch operation modes and return to a more energy-efficient configuration. In variant A (Figure 13), the node runs the collection-processing-transmitting sequence followed by the sleep period. In this mode, there is no need to store the collected data locally since they are once uploaded to a higher level. This mode uses the highest energy value but ensures the exact data reporting process.

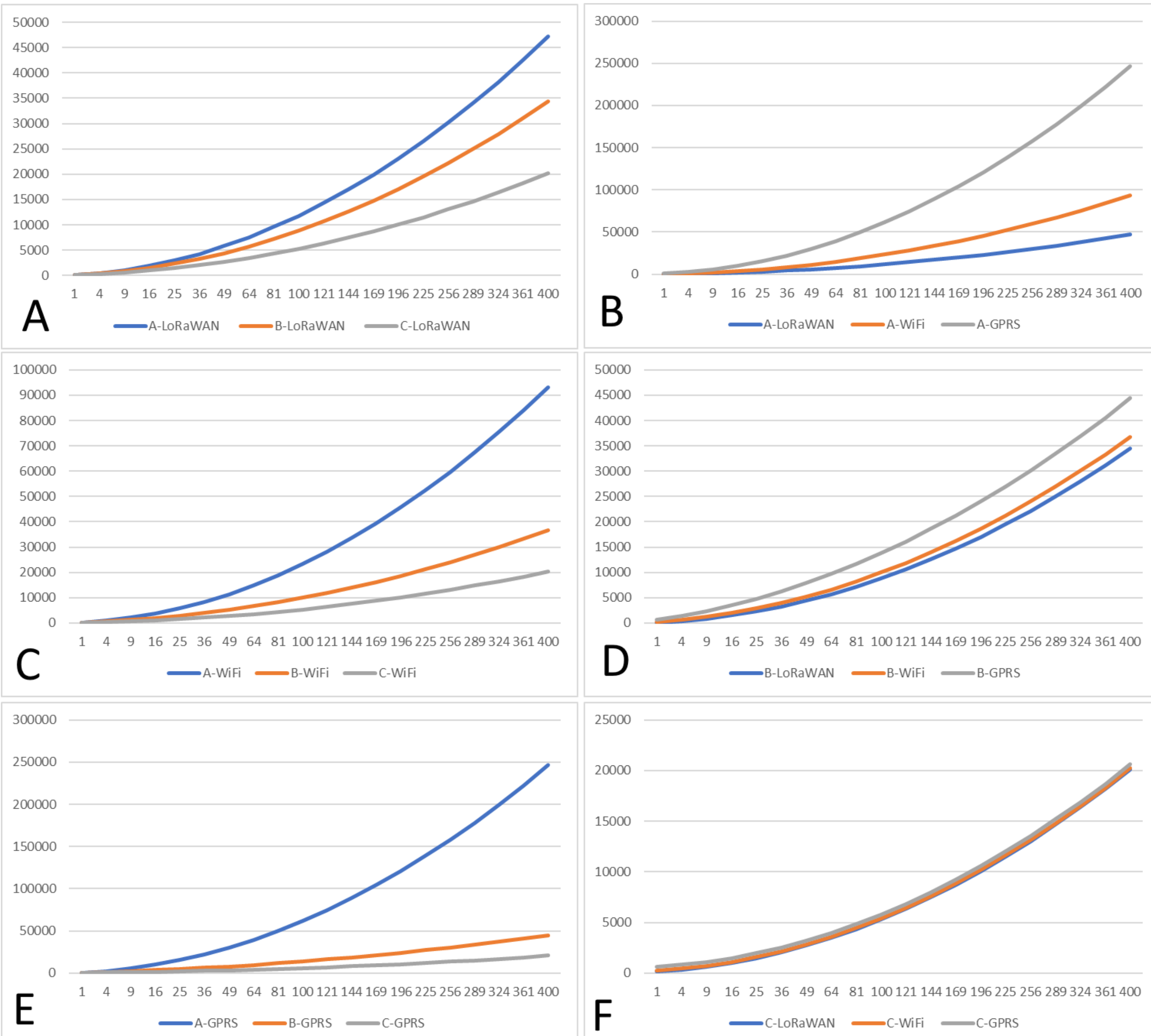

Figure 14 Comparison of energy consumption across a different combination of variants and communication devices (The X-axis represents the number of data collection events from sensors, and the Y-axis is energy consumption in mA)

Configuration variant B is intended to reduce the number of data transmissions, but it cannot be used in every case. It could be used only when the acceptable delay between data retrieval and transmission is long enough. The highest gain of this approach is when the GPRS data transmission method must be used since it consumes a significant amount of energy while setting up a connection to the network.

Configuration variant C is the best solution from the point of view of energy consumption, but it brings additional limitations. First, the time until data are uploaded to the Edge level is even higher. Second, since the data processing part does not follow every data collection, there is some risk that potentially wrong values could be discovered later than in cases B and C.

In the end, sometimes, IoT nodes must be on constant alert and run actively as much as possible. Since the consumption in fully active nodes is far from acceptable, one solution for the ESP32-based systems is the introduction of CAM when only radio, Wi-Fi, and Bluetooth are disabled. In that way, the system could stay in an active state longer and use less energy. The working mode would be the most like configuration variant B in this case.

As can be seen, each of the three working modes has advantages and disadvantages, and the operation mode would probably need to be adjusted during the node's life cycle. The possibility of changing the node behavior through the software interface would help in this case. The use of the mentioned digital twin is crucially important here. The end user could adjust node behavior in the digital twin, run the simulations on data transfer and energy consumption, and then push the change to the actual node.

Figure 14 Compares energy consumption with different operation modes and communication modules enabled. Subfigures A, C, and E (of Figure 14) show the effect of buffering when the same transmission module is used. The energy use is the highest in the case without buffering (configuration A). When the pre-transmit buffer is included (scenario B), energy is reduced up to some point, and with the second buffer, the reduction is more significant. Scenario C with LoRaWAN is at an energy usage level of 42.63% compared to scenario A with the same communication module (20122 mA vs 47200 mA). The difference between scenarios A and C with the integrated Wi-Fi module is 21.71% (20237 mA vs 93200 mA). The biggest gain is with GPRS, where the energy needed for scenario C is only 8.36% (20620 mA vs 246400 mA).

When comparing the same operating scenario against different communication modules (Figure 14 – B, D, and F), the most significant difference is for scenario A. The introduction of a buffer would close the gaps. For scenario C, the power usage with the GPRS module is less than 3% higher than with LoRaWAN.

This result is promising for implementing the nodes running in an off-grid regime. When they operate in near real-time with the most effective configuration (scenario A and with LoRaWAN), the node uses a predictable amount of energy. The battery could last several more days without recharging than the design based only on ESP32. The node must adapt its behavior if the external conditions worsen or the LoRaWAN module stops working correctly. So, it should switch to more energy-consuming communication devices, such as the GPRS. With the consumption estimate, the node could calculate the remaining energy and raise the appropriate alarm. Depending on the battery charging rate, buffering could be turned on, and the message queue size could be adjusted. In this way, the node could reduce energy consumption on the cost of near real-time reporting.

In the cloud system, in the database layer, each IoT node has been represented by the configuration data sequence. These data are sensor addresses, retrieval and retention period, boundary (minimal and maximal), or set of accepted values. The copy of all these data is then moved to the memory of the IoT node connected to specific sensors. In this way, every IoT node is fully aware of all connected sensors and their behavior. In this situation, verifying the sensor or connection line failure is more accessible. The most common conditions are when the IoT receives data from a sensor in an irregular interval, with values out of bounds, or when no response from the sensor can be detected. The response to all the mentioned scenarios could be predefined in the IoT node software, making the system reaction faster and more predictable. Also, the software change, if needed, is a much easier task in IoT than at the sensor network level.

7.1. Comparison with Industrial Standard Solution

Since IoT is an essential element of the Industry 4.0 landscape, many successful solutions are available. During the development process, we designed our solution based on our experience with Cassia [53], Aegex [54], and BARTEC [55], and with special requirements faced in hazardous and remote areas for the device with low build, maintenance, and operational costs (Table 8).

The usual approach for hazardous areas is gateway-centric architecture. This means that the complete system consists of multiple devices, some of which are sensors, some of which are concentration nodes, and some of which are gateways. Such approaches bring robust and very potent solutions, but from an explorational point of view, they are more convenient for more extensive facilities with constant human presence. The gateway-centric approach comes with dedicated on-site supporting hardware. The three IoT systems have their own hardware devices for monitoring and maintenance. Our solution could be monitored by any device with LoRaWAN connectivity, authorized through our cloud, and installed with dedicated software. Another advantage of gateway-centric architecture is the possibility of extending the system over the API, while the presented solution only supports application-level software updates. Our solution has been developed to work in IoT-centric mode, where only one type of node plays a leading role in data collection, aggregation, and transmission processes.

Regarding connectivity and supported sensors, Aegex and BARTEC support manufacturer-specific sensors as separate devices that could be added to a network plug-and-play manner using LAN, Bluetooth, or Wi-Fi. At the same time, Cassia's solution relies only on Bluetooth for connection. On the other hand, our solution works on a bit lower level, offering I2C and RS485 connectivity for any low-level sensor with such possibility. Our solution allows connecting to 4 sensors, the same as the Aegex solution. Aegex solution would need a gateway for each IoT node, while our solution gateway node is unnecessary.

The most similar solution to our node is BARTEC HY LOG. It is a complete system in one enclosure dedicated to monitoring the quantity of hydrogen. This device also supports GSM connectivity and GPS tracking by default, but it is committed to only one task. Like our IoT node, it has an incorporated solar panel and can run independently from a wired power supply. Other systems support integration with GSM, GPS, and solar-powered battery power supplies, but only through external devices, which makes the system much more extensive and complex for installation.

The proposed solution is a complete system in one device, intended to work without human intervention and with the possibility of connecting to any sensor running supported connection interfaces. It offers software-level flexibility, which means that the nodes in the same network can perform different tasks. Since all nodes are equal, maintenance constantly replaces a malfunctioned device with a new one and initiates the OTA setup.

Table 8 The main features of similar industrial solutions

Feature	Cassia [53]	Aegex [54]	BARTEC [55]	Presented solution
Architecture type	Gateway-centric	Gateway-centric and IoT-centric	Gateway-centric and partly IoT-centric	IoT-centric
On-site hardware support	Cassia IoT Access Controller with Bluetooth plug-and-play	Custom-built, intrinsically safe tablet device, Wi-Fi connected	Custom-build Android-base smartphone	None specific, but any device supporting LoRaWAN standard
Software extensibility	Application-level API level	Application-level API level	Application-level API level	Application-level
Sensor connectivity	Separate sensors with Bluetooth connectivity	Specific supported sensors Plug-and-Play (LAN, Bluetooth, Wi-Fi)	Specific supported sensors Plug-and-Play (LAN, Bluetooth, Wi-Fi)	Any sensor able to connect I2C or RS485 Software level adaptation
Number of sensors per device	Practically unlimited	8 sensors per gateway or 4 per endpoint device	Practically unlimited, 1 for BARTEC HY LOG	4 per device
GSM module	External	Integrated	External, except BARTEC HY LOG	Integrated
GPS module	External	Integrated	External, except BARTEC HY LOG	Integrated
Power option	AC or DC with battery backup	AC or DC with battery backup External solar system	AC, Replaceable battery or solar for BARTEC HYLOG	Integrated or external solar system

7.2. Reliability Analysis and Next Steps

Future work will enhance IoT nodes by employing redundancy and reliability improvement schemes, such as failure partners. In this way, nodes will be able to cover more scenarios that are outside their current niche. Currently, redundancy is supported on a sensor level. A single IoT node can monitor multiple sensor devices of the same type (usually two), and they can act as failure partners. In this scenario, the operation node uses one sensor until its return values are within a predefined range. When the sensor returns unbalanced or out of the predefined range values, the IoT node will raise the alarm and switch to the backup sensor. This complete control is done on the software level. It is worth mentioning that such an approach will result in lower energy consumption but with lower flexibility.

The update of the failure partner scenario at the sensor level will be the approach when both sensors are active simultaneously. In this case, the IoT node compares results, and when one of them starts generating invalid values, the IoT node completely switches to the one that functions correctly. The sensor in a failure state could then be shut down, and an adequate alarm could be generated. When the sensor malfunction gets repaired or replaced, it will send the notification signal to the IoT node, which will start the recovery procedure. This approach does not guarantee 100% reliability since there is always a chance that both sensors could go to the failure state. In this case, the system will react by raising the highest priority alarm. The same type of alarm will also be raised when the sensor gets to an error state, but no redundancy device is installed. When only one sensor is present and it fails, the situation is beyond software-directed recovery, and physical intervention must be done. This update will also be entirely on a software level.

One of the limits is the possibility of replacing the processing and communication modules. They are in the device casing, so any repair or replacement action would require node disconnection and replacement. For this reason, introducing redundant IoT nodes will be one of the possible solutions. Another possibility for improvement would be reconfiguring the complete network by introducing different IoT nodes with different roles. When one would be used only for data collection, the others could be used for data processing and transmission. This way, the system would be more robust and reliable but at a higher maintenance cost since more nodes must be employed and more software variants must be maintained. Such an improvement would move the architecture towards a gateway-centric model, but with all nodes running the same hardware.

The introduction of redundant IoT nodes is the solution to handle cases with hardware errors. In a configuration with two IoT nodes, both have an equal structure and have the same software installed. One of them acts as a master, and the other one is a slave. The configuration with master and slave IoT nodes is a shift away from IoT-centric design since both nodes must be connected to the same set of sensors over the communication line. This would result in more expensive solutions and a significant shift to gateway-centric architecture. Compared to redundant partner design, the difference is that only the master can trigger data exchange with the Edge level. At the same time, the slave will only listen to the traffic and receive the data sent by the sensors. In this situation, the master IoT node is active, and the slave is in the so-called sniffer mode. When the IoT node is in the sniffer node, it sends no data to higher levels (Edge computer).

When the slave node does not receive the keep-alive message for the predefined period, it will try to connect to the master node (ping). If there is no response from the master node, the slave will switch to the active (master) mode. At that moment, the former slave IoT node will take over the complete functionality of the former master and set up all the functions needed for the sensor and Edge layers. This procedure will be executed without human intervention, and when such an incident happens, the new master node will send a high-level alarm to the Edge layer. Also, regarding software updates or hardware replacements, one node could be shut down for updates while the other will continue to collect measurements. Research in this direction would also switch the deployment paradigm to gateway-centric design, bringing higher reliability but at a higher maintenance cost. With such an update, the solution will be more suitable for more extensive deployments and leave the niche it currently holds. Expanding communication to higher levels will focus on security. Currently, both ESP32 and additional communication modules support basic 802.11 security standards. Since this could be easily broken, one of the focuses for the next phase will be the acquisition of advanced security protocols for IoT devices.

The presented research was focused on the design of the single node. In terms of scalability, it is equal to the scalability of its building blocks. The most important feature of the design is the possibility of integrating the IoT node into broader systems. The node can communicate with the environment using two channels (LoRaWAN and GSM) and, optionally, two channels that come as part of ESP32 (Wi-Fi and Bluetooth). The proposed IoT nodes could theoretically cover unlimited sensing devices by participating in the more comprehensive network. Each IoT node could connect to RS485 and I2C and transmit data to the Edge level. Using the MQTT-SN protocol, the designed IoT node can connect to every system that supports such communication.

Improvements in the battery charging algorithm would be necessary for future design improvements. As the first step, we introduced externally controlled charging, which could be triggered from the Cloud or Edge level and force the IoT node to start to charge the battery. Next, we replaced simple threshold-based charging with an improved process that considers the current battery level, the estimated energy consumption, and the time until the next sunrise. The focus is currently on defining the method based on the improved techniques and machine learning to define autonomous models, which will ensure, if possible, IoT node operation in the off-grid environment.

8. Conclusions

The paper introduces a novel combination of energy-efficient hardware selection and adaptive software control to manage power consumption autonomously. Multiple limitation factors, such as casing design, cost, and the worldwide availability of used components, drove the design request. The starting point was a solely used ESP32, and during the development, the inefficient hardware elements were replaced, and an autonomous power supply system was integrated. This was a challenge because used components were often designed to run in factory conditions without power or connectivity limitations. Thanks to the advanced operating system of the ESP32 node, further improvements were made through the set of software implementations and updates, including the definition of the optimized working mode. By integrating hardware and software optimizations, this work improves upon traditional IoT designs for Industry 4.0, offering enhanced efficiency for deployment in remote and hazardous environments. This research was conducted in parallel with investigating diverse deployment strategies for client software across various ISA-95 layers. Throughout this process, the node was integrated into a digital twin structure in the cloud, and the possibility of the software OTA update and monitoring was enabled. Overall, all software design and hardware configuration optimizations aimed to enhance energy efficiency (Table 9), and this goal was achieved by:

- Implementing different battery charging routines to maximize energy collection effectiveness. Since the standard battery charging routine triggers relatively rarely (once a week or bi-weekly), automatic charging could start at night or in bad weather, resulting in no energy gain. To suppress this, a controlled charging mode, initiated from the Edge level, was implemented, which could trigger battery charge on demand, by a predefined schedule, or based on the weather forecast.
- Utilizing external low-power communication components. The LoRaWAN component for real-time transmission reduces energy use by nearly half (50.64%).
- Defining a new controlled active mode optimized for the anticipated use. The new mode with the communication part disabled utilizes 72% of the energy used

in comparable modem sleep mode (36 mA vs. 50 mA) and only 40% of the power that would model sleep mode with active sensors (69 mA vs. 149-200 mA) would use. A similar ratio applies when sensors and the LoRaWAN module are active – 98 mA vs. 200 mA when ESP32 is in standard active mode with sensors enabled.

- Implementing adaptive software that ensures seamless transitions between active and sleep modes. Based on the required measurement, processing, and transmission frequencies, the controlling software will decide when to switch the active components off and reduce energy consumption.
- Integration into digital twin that allows early warning mechanisms and OTA updates. The frequency of transmission of node health parameters to digital twin could be configured, but their size is the equivalent of a single packet containing data collected from sensors. Usually, it is enough to run such a telemetry for once after 1000 data collection cycles. The additional energy consumption caused by such a process would be less than 0.1%.
- Using message buffers to reduce the number of data transmissions. For the most common scenario with LoRaWAN, using a buffer of size ten will result in an energy reduction of 25%, while using a buffer of size 100 will result in a reduction of up to 55%. When a message buffer of size 100 is used, the total energy consumption will be very close regardless of the transmission module used.

The more notable gain is when GPRS is used for transmission. If a buffer of only ten messages were used, only 22.40% of the initially required energy would be used. In contrast, with a buffer size of 100, the consumption will be reduced to 9.38%. Notably, this approach introduces a trade-off: while it reduces energy usage, reporting to the Edge layer will be less frequent.

Table 9 Energy-saving enhancements

Update	**Compared element**	**Energy Reduction**
CAM Mode	ESP32 Light Sleep	20 – 30%
CAM Mode	ESP32 Active Mode	45 – 55%
CAM + Sensors	Sensor reading and ESP32 processing in active mode	50 – 70%
LoRaWAN	ESP32 integrated Wi-Fi	50%
Transmission buffer of size 100	Immediate transmission upon processing. The used energy is nearly equal regardless of the transmission device	55 – 90%

Continued improvement efforts are directed toward enhancing system reliability, fault tolerance, information security, and overall system readiness and availability. As a preliminary step, we envision enhancing reliability by introducing additional redundancy at the IoT level, bolstering robustness and error resilience. Further improvements to the battery charging subsystem will also run in parallel with ongoing node development, aiming to extend battery life and mitigate the risk of power depletion. An ancillary outcome of this research is a set of design recommendations formulated during the enhancement process:

- **Standardized Components**: Adhere to proven standardized components that have demonstrated reliability in real-world conditions.
- **Module Disabling and Replacement**: Permanently disable or replace modules that fail to meet performance expectations.
- **Feature Flags for Dark Mode**: Introduce feature flags to enable dark mode in regular software operations (not exclusively for software updates).
- **Message Queues and Buffering**: External management of message queues and buffering must be employed to adapt the node's operation dynamically.
- **Integration with Digital Twins**: Enable permanent monitoring by integrating IoT nodes with digital twins.

While the presented node operates within a specific industrial context, the solutions it embodies transcend disciplinary boundaries. Authors must remain receptive to diverse concepts, regardless of their research origins. This study underscores the ongoing need to continually enhance energy-efficient component usage, evaluating and incorporating solutions as they prove sufficient.

Acknowledgments: This work has been supported by the cost action CA 19135 CERCIRAS (Connecting Education and Research Communities for an Innovative Resource Aware Society). This work has been funded by the Ministry of Education, Science, and Technological Development of the Republic of Serbia, grant number 451-03-68/2022-14/ 200102

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the study's design, data collection, analysis, interpretation, manuscript writing, or decision to publish the results.

Nomenclature

Acronym	Description
ACH	Average energy Consumption per Hour
AL	Alarm Low energy level in battery.
CAM	Controlled Active Mode
CL	Charging required Level
ESP32	Low-power microcontrollers are widely used in IoT applications.
Ex e	The class of device enclosure constructed and certified as explosion-protected according to the Increased Safety standard.
FreeRTOS	Free Real Time Operation System. Operation system native to ESP32 controller
GPRS	General Packet Radio Service, data transfer standard for mobile networks
GPS	Global Positioning System. Satellite-based radio navigation system.
GSM	Global System for Mobile communications, standard for mobile networks
I2C	Inter-Integrated Circuit. Serial communication bus used to attach lower speed sensors
IEEE	Institute of Electrical and Electronics Engineers

IoT	Internet of Things
ISA-95	Standard from the International Society of Automation for developing an automated interface between enterprise and control systems.
LoRa	Low Radiation. Network protocol to wirelessly connect battery-powered devices.
MQTT	Message Queuing Telemetry Transport protocol
MQTT-SN	Message Queuing Telemetry Transport for Sensor Networks protocol
OTA	Over-The-Air. Update to an embedded system that is delivered through a wireless network
RH	Requested High level. Battery level where charging should stop.
RS485	Recommended Standard #485. The standard for serial communication between devices
RTC	Real-Time Clock
SH	Standard High battery level
SIM	Subscriber Identification Module. The card is used to enable mobile communication for devices.
SL	Standard Low Battery Level
ULP	Ultra-Low Power. Processing unit optimized for low energy consumption.
UMTS	Universal Mobile Telecommunication System. Cellular system for network based on GSM

References

1. Kumar, S., Tiwari, P., & Zymbler, M. (2019). Internet of Things is a revolutionary approach for future technology enhancement: a review. Journal of Big data, 6(1), 1-21.
2. Paiola, M., & Gebauer, H. (2020). Internet of things technologies, digital servitization and business model innovation in BtoB manufacturing firms. Industrial Marketing Management, 89, 245-264.
3. Qu, Y. J., X. G. Ming, Z. W. Liu, X. Y. Zhang, and Z. T. Hou. "Smart manufacturing systems: state of the art and future trends." The International Journal of Advanced Manufacturing Technology 103 (2019): 3751-3768.
4. Aheleroff, S., Xu, X., Lu, Y., Aristizabal, M., Velásquez, J. P., Joa, B., & Valencia, Y. (2020). IoT-enabled smart appliances under industry 4.0: A case study. Advanced engineering informatics, 43, 101043.
5. Brous, P., Janssen, M., & Herder, P. (2020). The dual effects of the Internet of Things (IoT): A systematic review of the benefits and risks of IoT adoption by organizations. International Journal of Information Management, 51, 101952.
6. Phuyal, S., Bista, D., & Bista, R. (2020). Challenges, opportunities, and future directions of smart manufacturing: a state of art review. Sustainable Futures, 2, 100023.
7. Aleksić, D. S., Janković, D. S., & Stoimenov, L. V. (2012). A case study on the object-oriented framework for modeling product families with the dominant variation of the topology in the one-of-a-kind production. The International Journal of Advanced Manufacturing Technology, 59, 397-412.
8. Aleksic, D. S., Jankovic, D. S., & Rajkovic, P. (2017). Product configurators in SME one-of-a-kind production with the dominant variation of the topology in a hybrid manufacturing cloud. The International Journal of Advanced Manufacturing Technology, 92, 2145-2167.

9. Rajković, P.; Aleksić, D.; Djordjević, A.; Janković, D. Hybrid Software Deployment Strategy for Complex Industrial Systems. Electronics 2022, 11, 2186. https://doi.org/10.3390/electronics11142186
10. Rajković, P., Aleksić, D., Janković, D., Milenković, A., & Đorđević, A. (2021, September). Resource Awareness in Complex Industrial Systems–A Strategy for Software Updates. In Proceedings of the First Workshop on Connecting Education and Research Communities for an Innovative Resource Aware Society (CERCIRAS), Novi Sad, Serbia (Vol. 2).
11. ISA-95 standard page, available online: https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95, last accessed on February 26^{th}, 2023
12. Rajković, P., Aleksić, D., Janković, D. (2024). The Implementation of Battery Charging Strategy for IoT Nodes. In: Zeinalipour, D., et al. Euro-Par 2023: Parallel Processing Workshops. Euro-Par 2023. Lecture Notes in Computer Science, vol 14352. Springer, Cham. https://doi.org/10.1007/978-3-031-48803-0_4
13. P. Rajković, A. Djordjević, D.Aleksić, D. Janković, Usage of Modular Software Development for IoT Nodes— A Case Study, Proceedings of the Tenth Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications SQAMIA 2023, Bratislava, Slovakia, September 2023, pp. 114-125, (https://ceur-ws.org/Vol-3588/p11.pdf)
14. Guidelines for integrated risk assessment and management in large industrial areas, https://www-pub.iaea.org/MTCD/publications/PDF/te_994_prn.pdf, last accessed on April 22^{nd}, 2023
15. Increase safety Ex e standards, available online: https://www.nsw.gov.au/testsafe/electrical/explosive-atmosphere/increased-safety, last accessed on February 26^{th}, 2023
16. Andres-Maldonado, P., Lauridsen, M., Ameigeiras, P., & Lopez-Soler, J. M. (2019). Analytical modeling and experimental validation of NB-IoT device energy consumption. IEEE Internet of Things Journal, 6(3), 5691-5701.
17. Anbazhagan, S., & Mugelan, R. K. (2024). Energy efficiency optimization of NB-IoT using integrated Proxy & ERAI technique. Results in Engineering, 23, 102419. https://doi.org/10.1016/j.rineng.2024.102419
18. Mocnej, J., Miškuf, M., Papcun, P., & Zolotová, I. (2018). Impact of edge computing paradigm on energy consumption in IoT. IFAC-PapersOnLine, 51(6), 162-167.
19. Monteil, T. (2023). Integration of green aspect inside internet of things standard. In 2023 congress in computer science, computer engineering, & applied computing (CSCE). IEEE. https://doi.org/10.1109/csce60160.2023.00289
20. Dos Anjos, J. C., Gross, J. L., Matteussi, K. J., González, G. V., Leithardt, V. R., & Geyer, C. F. (2021). An algorithm to minimize energy consumption and elapsed time for IoT workloads in a hybrid architecture. Sensors, 21(9), 2914.
21. Uelschen, M.; Schaarschmidt, M. (2022). Software Design of Energy-Aware Peripheral Control for Sustainable Internet-of-Things Devices. In Proceedings of the 55th Hawaii International Conference on System Sciences, Maui, HI, USA, 4–7 January 2022.
22. Shekarisaz, M., Thiele, L., & Kargahi, M. (2021). Automatic energy-hotspot detection and elimination in real-time deeply embedded systems. In 2021 IEEE Real-Time Systems Symposium (RTSS). IEEE. https://doi.org/10.1109/rtss52674.2021.00020
23. Shekarisaz, M., Kargahi, M., & Thiele, L. (2024). Inter-Task Energy-Hotspot Elimination in Fixed-Priority Real-Time Embedded Systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1. https://doi.org/10.1109/tcad.2024.3372447
24. Schaarschmidt, M., Uelschen, M., & Pulvermüller, E. (2022). Hunting energy bugs in embedded systems: A software-model-in-the-loop approach. Electronics, 11(13), 1937. https://doi.org/10.3390/electronics11131937
25. Bouguera, T.; Diouris, J.-F.; Chaillout, J.-J.; Jaouadi, R.; Andrieux, G. Energy Consumption Model for Sensor Nodes Based on LoRa and LoRaWAN. Sensors 2018, 18, 2104. https://doi.org/10.3390/s18072104
26. Rajab, H., Cinkler, T., & Bouguera, T. (2021). Evaluation of Energy Consumption of LPWAN Technologies, available at Research Square, DOI: 10.21203/rs.3.rs-343897/v1

27. Al-Kashoash, H. A., & Kemp, A. H. (2016). Comparison of 6LoWPAN and LPWAN for the Internet of Things. Australian Journal of Electrical and Electronics Engineering, 13(4), 268-274. DOI: 10.1080/1448837X.2017.1409920
28. Jeon, K. E., She, J., Xue, J., Kim, S. H., & Park, S. (2019). luXbeacon—A batteryless beacon for green IoT: Design, modeling, and field tests. IEEE Internet of Things Journal, 6(3), 5001-5012.
29. Khutsoane, O., Isong, B., Gasela, N., & Abu-Mahfouz, A. M. (2019). Watergrid-sense: A lora-based sensor node for industrial iot applications. IEEE Sensors Journal, 20(5), 2722-2729.
30. Fowler, M. DarkLaunching, April 2020.
31. Kanan, R., Elhassan, O., & Bensalem, R. (2018). An IoT-based autonomous system for workers' safety in construction sites with real-time alarming, monitoring, and positioning strategies. Automation in Construction, 88, 73-86. DOI: 10.1016/j.autcon.2017.12.033
32. Baig, M. J. A., Iqbal, M. T., Jamil, M., & Khan, J. (2021). Design and implementation of an open-Source IoT and blockchain-based peer-to-peer energy trading platform using ESP32-S2, Node-Red and, MQTT protocol. Energy reports, 7, 5733-5746.
33. Mcginthy, J. M., & Michaels, A. J. (2019). Secure industrial Internet of Things critical infrastructure node design. IEEE Internet of Things Journal, 6(5), 8021-8037.
34. Roldán-Gómez, J., Carrillo-Mondéjar, J., Castelo Gómez, J. M., & Ruiz-Villafranca, S. (2022). Security Analysis of the MQTT-SN Protocol for the Internet of Things. Applied Sciences, 12(21), 10991.
35. Banguero, E., Correcher, A., Pérez-Navarro, Á., Morant, F., & Aristizabal, A. (2018). A review on battery charging and discharging control
36. Bose, B., Garg, A., Panigrahi, B. K., & Kim, J. (2022). Study on Li-ion battery fast charging strategies: Review, challenges, and proposed charging framework. Journal of Energy Storage, 55, 105507.
37. Battery Management System Market Research Report: By Battery Type, Connectivity, Topology, Vertical—Global Industry Analysis and Forecast to 2030—Global Industry Analysis and Demand Forecast to 2030.
38. Kumar, K., Chaudhri, S. N., Rajput, N. S., Shvetsov, A. V., Sahal, R., & Alsamhi, S. H. (2023). An iot-enabled e-nose for remote detection and monitoring of airborne pollution hazards using lora network protocol. Sensors, 23(10), 4885. https://doi.org/10.3390/s23104885
39. Muralidhar, T. V., Sandeep, V. V. S., Manohar, P., Krishna, M. L., Ruthvik, K., & Bagwari, S. (2024). An iot based real time forest fire detection & alerting system using lora communication. In 2024 11th international conference on signal processing and integrated networks (SPIN). IEEE. https://doi.org/10.1109/spin60856.2024.10512122
40. Ensworth, J. F., & Reynolds, M. S. (2017). BLE-backscatter: Ultralow-power IoT nodes compatible with Bluetooth 4.0 low energy (BLE) smartphones and tablets. IEEE Transactions on Microwave Theory and Techniques, 65(9), 3360-3368. OI: 10.1109/TMTT.2017.2687866
41. ESP32-WROOM-32 Datasheet, available online: https://cdn-shop.adafruit.com/product-files/3320/3320_module_datasheet.pdf last accessed on February 25th, 2023
42. ESP32 Series Datasheet, available online: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf last accessed on February 25th, 2023
43. ESP32 alternatives - finding the best microcontroller for your project needs. (n.d.). Espboards.dev. available online, https://www.espboards.dev/blog/esp32-alternatives/ last accessed on September 20th, 2024
44. Sundaram, J. P. S., Du, W., & Zhao, Z. (2019). A survey on lora networking: Research problems, current solutions, and open issues. IEEE Communications Surveys & Tutorials, 22(1), 371-388. DOI: 10.1109/COMST.2019.2949598
45. Stanford-Clark, A., & Truong, H. L. (2013). Mqtt for sensor networks (mqtt-sn) protocol specification. International business machines (IBM) Corporation version, 1(2), 1-28.

46. Jia, K., Xiao, J., Fan, S., & He, G. (2018). A mqtt/mqtt-sn-based user energy management system for automated residential demand response: Formal verification and cyber-physical performance evaluation. Applied Sciences, 8(7), 1035. DOI: doi.org/10.3390/app8071035
47. FreeRTOS resource page, available online: https://www.freertos.org/, last accessed on February 26th, 2023
48. Potić, I., Golić, R., & Joksimović, T. (2016). Analysis of insolation potential of Knjaževac Municipality (Serbia) using multi-criteria approach. Renewable and Sustainable Energy Reviews, 56, 235–245. https://doi.org/10.1016/j.rser.2015.11.056
49. GDM-8255A Dual Display Digital Multimeter Factsheet, available online: https://www.gwinstek.com/en-global/products/detail/GDM-8255A, last accessed on February 25th, 2023
50. UT71C digital multimeter, available online: https://meters.uni-trend.com/product/ut71-series/, last accessed on February 25th, 2023
51. LoRaWAN module SimTech SX1268 Factsheet, available online: https://www.semtech.com/products/wireless-rf/lora-connect/sx1268, last accessed on February 25th, 2023
52. SimCom SIM800H GSM module resource page, available online: https://datasheetspdf.com/pdf/823439/SIMCom/SIM800H/1, last accessed on February 25th, 2023
53. Cassia Networks. Industrial IoT Products and Solutions, available online: https://www.cassianetworks.com/bluetooth-iot-solutions/industrial-iot/, last accessed September 22nd, 2024.
54. Ventulett, T. Aegex IoT Platform for Hazardous Locations, available online: https://aegex.com/images/uploads/Aegex_IoT_Platform_For_Hazardous_Locations_FINAL-1.pdf /, last accessed September 22nd, 2024.
55. Industrial Internet of Things for hazardous areas: potential for the optimisation of existing plants Whitepaper, available online: https://bartec.com/fileadmin/2-Products_and_Solutions/2-5-Smart_Factories/ACS_Whitepaper_EN.pdf, last accessed September 22nd, 2024.