BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Publicat de
Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Volumul 69 (73), Numărul 1, 2023
Secţia
ELECTROTEHNICĂ. ENERGETICĂ. ELECTRONICĂ
DOI:10.2478/bipie-2023-0004

A STUDY REGARDING POWER CONSUMPTION OF AN IoT NODE FOR IMAGE RETRIEVAL AND ITS OPTIMIZATION

BY

DORU CORNEI[1,*], CRISTIAN FOȘALĂU[1] and LAURA CORNEI[2]

[1]“Gheorghe Asachi” Technical University of Iași,
Faculty of Electrical Engineering, Iași, Romania
[2]“Alexandru Ioan Cuza” University of Iași, Faculty of Computer Science, Iași, Romania

Received: February 18, 2024
Accepted for publication: April 11, 2024

Abstract. This paper analyses the power consumption of an IoT node that captures images and transmits them to a cloud storage. Various implementation versions of the IoT node are proposed, focusing on the utilization of common components to achieve cost-effectiveness and minimize the maintenance requirements. The realised node variants were initially tested in laboratory conditions and then commissioned and deployed outdoors, being located in an orchard where they remained operational during 7 months under different meteorological conditions. The proposed solution utilizes an ESP32 microcontroller and a camera with a resolution of 1.3 Megapixels as a data acquisition node whose energy is assured by a Li-Po battery, charged through a solar panel. Wi-Fi communication was employed to transmit the images to the server. The power consumption of the node was evaluated for different variants of software optimization using the HTTP or the MQTT transmission modes. The instantaneous current consumption of the nodes was measured in laboratory to identify the power consumption for each phase of the program execution.

*Corresponding author; *e-mail*: doru.cornei@student.tuiasi.ro

The experiments revealed that the MQTT transmission mode consumed considerably less power than the HTTP mode. Additional experiments have been performed in which the variation of the battery voltage, the solar panel voltage and the number of daily emissions in the examined period, as well as correlations between these quantities have been analyzed.

Keywords: IoT, camera, power consumption, ESP32, MQTT, HTTP.

1. Introduction

Agriculture, a dynamic and constantly changing domain of human activity, is often affected by limitations related to the lack of resources such as fertile soil, fresh water, high-quality fertilizers and labour force. These limitations have been accentuated in recent years by climate change. Other important agricultural challenges include the desertification of the agricultural land and natural disasters, such as extreme drought or flooding. In this context, the optimal use of resources through smart farming is essential in order to promote the sustainable development of this field. IoT technologies can measure and determine the needs of plants, animals, as well as perform actions, while optimization and prediction algorithms are able to indicate the best solutions for resource utilization.

In the following sections, we will discuss various designs and operational modes that can be used for an IoT node dedicated to the acquisition of images from the agricultural environment. Moreover, we will analyse the power consumption of the created versions and present and test different methods for optimizing them.

The proposed IoT node variants were built and replicated in at least 3 copies, commissioned and deployed outdoors for plant monitoring, operating for a minimum of 7 months each. Problems in the operation of the node versions were observed and the design was corrected, this paper presenting the final variants.

2. Related Work

In this section, we discuss several IoT applications utilized for parameter measuring in the context of smart agriculture. Many of the existing state of the art papers which present image transmission applications do not provide a thorough analysis of the energy consumption.

Palniladevi *et al*. (2023) created a system for measuring humidity, air temperature and soil moisture, using the "Raspberry Pi" and "Arduino UNO" modules. This system maintained the soil moisture using a water pump. The irrigation logic was realized with the help of a Recurrent Neural Network, implemented in the "Raspberry PI" module. The power supply was provided by a series of solar panels using a minimal orientation system. The developed

application displayed the information locally and transmitted it to the cloud. The power consumption of this system was not specified.

Zheng *et al*. (2023) presented a greenhouse control system that accurately monitored the temperature, soil moisture, and the ambient light levels. After collecting the data, the application sent it to a local server (Wang *et al*., 2022), responsible for: processing information, determining commands and transmitting them to the execution elements. However, details such as control mode, execution elements, and the power supply of the system were not mentioned.

Konain *et al*. (2023) presented a study on the implementation of an ESP32 microcontroller-based system for regulating environmental conditions in a mushroom farm. The application employed sensors to monitor air temperature, humidity, soil moisture, and CO_2 gas concentrations with the purpose of managing the irrigation and ventilation processes. The Wi-Fi and HTTP protocols were used for transmitting the data to the cloud. The system was powered by a solar panel, although the power consumption was not analysed.

To maximize the energy obtained from the solar panels, the consumer impedance is usually fine-tuned to attain the Maximum Power Point (MPP). Given the variability in the light intensity in practical scenarios, (Sun *et al*., 2022) proposed an improved variable-step incremental conductance method. This method was used for efficient, rapid and oscillation-free MPP tracking for individual solar panels.

Meng *et al*., (2023) introduced an algorithm that determined the Maximum Power Point for a system consisting of multiple solar panels. The measured parameters included temperature, humidity, light intensity in the area of the solar panels, voltage and the electric current generated by the solar panels. The authors proposed a composite variable step MPP tracking control algorithm which was based on the three-stage variable step incremental conductance method (Liu and Sun, 2018). The method added the Kalman filtering algorithm to pre-process the output signal of the photovoltaic cells and used a new calculation approach to adjust the variable step coefficient.

Correia *et al*. (2022) presented an approach for transmitting images using the LoRaWan protocol. This method employed an ESP32-CAM module, equipped with an OV2640 camera and the Arduino LoRa MKR WAN 1310 module for LoRa transmission. The pictures were retrieved with a resolution of 320x240 pixels and encoded in JPEG format. To transmit the data via LoRa communication, these images were further encoded in Base64 and segmented into packets of maximum 50 bytes. The resulted packets were sent every 20 seconds, therefore the transmission of 3 kilobytes of image data could be realized in approximately 26 minutes. The retrieval of the packets from the LoRaWan server was accomplished using the MQTT protocol through a Node-RED program, responsible for assembling and decoding the images. It is worth

noting that the time required for image transmission was considerable, as the node remained active throughout that duration.

The study performed by (Ching-Chuan Wei *et al*., 2020) presented a comparison of the LoRa image transmission efficiency, focusing on different encoding methods (JPEG and Webp + Base64). The experiment involved the transfer of 200×150 pixels images via LoRa communication covering a distance of 1.5 km, using point-to-point transmission. The hardware setup included a Raspberry Pi 3B module and Semtech SX1276 LoRa chips. The results indicated that Webp + Base64 encoding yielded a smaller file size, thereby reducing the transmission time by approximately fifty percent. However, it is important to note that the Webp + Base6 encoding was resource-intensive in terms of both memory and computation time.

In (Takaaki Kawai, 2021), an image compression scheme, based on a H.264 video encoding approach, was proposed in order to reduce the data rate to be transmitted. Their method transmitted only once, as reference, the photo taken in beginning of the day and then, the rest of the images, as differences from the reference. The chosen microcontroller for implementation was the Raspberry Pi Zero W, selected for its ability to store images and facilitate the computation of the differences. However, just as in the previous case, the solution required significant memory resources and computing time.

Regarding the measurements using IoT nodes in agriculture, a main problem was their power consumption, given that these nodes operated with energy sources independent of the power supply network (e.g., materialized by batteries or solar and wind harvesters). One well-known and widely used energy saving method was the sleep/awake mode of working described in (Jawad *et al*., 2018).

This paper presents an analysis regarding the power consumption of multiple versions of an IoT node employing WiFi data transmission to the server. The node was placed in an orchard in close proximity to the associated residence and was used for taking pictures from the orchard and remotely transmitting them to a cloud storage.

3. Node Construction

The proposed IoT node is power-efficient, resilient, cheap and easy to maintain, meeting the following requirements:

- automatically transmits the images to the storage server;
- ensures periodic (hourly) image retrieval;
- has the following minimal image requirements: colour image, 1280x1024 pixels, equivalent to a resolution of 1.3 Megapixels). Compressed in JPEG format, the images had an average size of 450 kilobytes;
- operates outdoors with no other protections;

- is stand-alone, no maintenance being required;
- is powered from one solar panel or other renewable sources;
- was built using cheap and accessible components.

It is known that the LoRa protocol provides long range, low power data communication, being largely employed in many smart farming applications. Several drawbacks like low speed prevent it to be used in certain application, especially when a Wi-Fi router is closely available. That is why we decided to analyse the power consumption for a Wi-Fi node, useful for those application where such a solution is proved to be useful. A deeper justification of using WiFi data transmission in this case is provided below.

To retrieve and transmit an image, the processor unit must be able to store it. Due to the fact that a large amount of data needed to be transferred per hour, low-rate transmission modes did not represent a feasible solution. For example, LoRaWan could transfer a maximum of 21,534 bytes per hour, significantly less than the average image size of 450 kilobytes. Below, we explain how we calculated the transmission rate for LoRaWan.

The maximum data packet size that can be transferred using the LoRaWan protocol is 222 bytes. Considering the fastest transmission rate DR5 (Data Rate 5), a packet can be sent in 368.9 milliseconds (Avbentem.github.io). Given a 1% duty cycle of emitting and pause time, this message can be transmitted 97 times per hour. Therefore, LoRaWan can transmit a maximum of 21,534 bytes in an hour (97 messages of 222 bytes each).

To solve this issue, we used Wi-Fi communication, as it was the cheapest and the most accessible solution. Wi-Fi is present in most inhabited areas and it is easy to implement and expand. In remote areas, mobile communication can be used to connect the Wi-Fi network to the internet. The equipment required for the Wi-Fi connection is affordable and widely commercially available. Nowadays, devices implement at least the IEEE 802.11g standard, which offers a transmission rate of 54 Mbit/seconds, which is significantly higher compared to the one of LoRaWan.

The power required to transmit data is directly proportional to the size of the data. Therefore, it was more efficient to obtain the compressed image from the camera and then transmit it, taking into account that the camera's power consumption was approximately the same when reading a raw or compressed image. The power consumption of the IoT node in an awake state is determined by the acquisition and the transmission steps and cannot be significantly minimized. The power usage of the node during the sleep state is not dependent of the task realized during the awake state, so it should be low to reduce the node's overall power consumption.

One of the most cheap and straightforward solutions to power a node is by utilizing a solar panel that charges a Li-Polymer battery. We employed this solution and selected a solar panel designed for outdoor use in order to offer resistance to ultraviolet rays and mechanical stresses, including hail.

4. Hardware Considerations

The node structure used in our application is shown in Fig. 1. As it can be seen, the node is composed of: a camera, the processor, the real time clock (RTC) module and a power supply module.

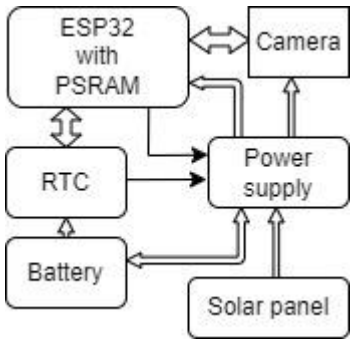

Fig. 1 − Hardware structure of the node.

The used camera, model OV3660, is affordable and widely available. Several features of this device are: 3 Megapixels resolution, automatic image and exposure controls, 1.5 V power supply, active mode power consumption: 28 mA for analogue, 64 mA for digital, 6 mA for I/O, 40 μA standby consumption.

For the choice of the signal processor, we tested several design variants with different processors. One used version was the Raspberry Pi Zero W minicomputer, which didn't meet the low power consumption as it requires in the low power state a current of approximately 20 mA. Another problem was related to the fact that it was sensitive to power supply voltage switching off, due to the reasons explained below. The operating system (Raspbian) is a Unix-like operating system based on Debian, optimised for the Raspberry Pi hardware. The operating system resides on the storage device. When the Raspberry Pi shuts down, the system takes care of unmounting the storage device. Therefore, the power supply cannot be switched off before executing the shutdown procedure as the storage device could get corrupted.

A more convenient solution was to use ESP32 development kits such as ESP32_CAM and ESP_EYE, which had lower power consumption and could be switched off without damaging the program. The program was stored in EEPROM and the microcontroller accessed it exclusively in a read-only mode. The power consumption of the microcontroller was about 3 μA. To obtain and store the image, the microcontroller communicated with the camera, acquired the picture with a resolution of 1920x1080 pixels and transmitted it to the storage server. The used microcontroller had sufficient memory for managing

the image acquisition, storage and transmission operations. The image was retrieved in a compressed JPEG format, the size of it being variable (usually between 0.3 Mbyte and 0.7 Mbyte), depending on the content. The final used microcontroller was the ESP32 with a 4 Mbyte external EEPROM memory for the program and an 8 Mbyte external STATIC RAM for the data.

Even if the power consumption of the microcontroller is low, the ESP32_CAM and ESP_EYE development kits have a high-power usage (in the order of hundreds of microamperes). This elevated power consumption is attributed to the fact that the camera remained powered during the sleep mode.

To fix the issue mentioned above, one solution was to use a real-time clock to reduce the power consumption below 1 μA during idle state, in which only the real-time clock was powered. The real-time clock generated an alarm, whose signal was used to power the microcontroller and the camera assembly. The microcontroller's role involved maintaining power, resetting the clock's alarm, capturing camera data, transmitting it, and programming a new alarm in the clock for future actions after a specified time. This solution came with a problem because the interaction between the real-time clock and the microcontroller must be initialized in order to operate. When the node starts to function for the first time, taking into account that an alarm was not previously set in the real-time clock, the microcontroller cannot be powered. On the other hand, if the microcontroller is not powered, it is not able to program an alarm in the real-time clock.

The power supply for the module must provide the necessary power for the camera and the microcontroller. The ESP32 microcontroller utilized an average of 80 to 150 mA while operating without communication. This power consumption increased towards 300 to 500 mA during periods when the Wi-Fi communication was active. This required a 3.3 V voltage stabilizer that supported a current of 600 mA and had low power consumption in no-load mode. Regarding the solution with the real-time clock, the power consumption in no-load was no longer relevant, but the short-term overcurrent support became important due to the reasons explained below. During idle time, the module (containing the camera and the ESP32 microcontroller) was not powered. Thus, the capacitors on the module were discharged to 0 volts. At the moment of wakeup, they were recharged, generating a peak current.

The energy needed to power the module was given by a Li-Polymer battery. This battery was charged using a solar panel. We evaluated three circuit variants for charging the batteries utilizing solar panels: the BQ25570 circuit (MIKROE-2814 kit), the SPV1050 circuit (DFR0579 kit), and the linear circuit LTC4057.

MIKROE-2814 and DFR0579 are switching chargers for the Li-Po battery which operate efficiently from input voltages higher than 0.6 V and utilize a supercapacitor for energy storage. In comparison to the linear circuit,

they were more expensive and sensitive to the operating conditions and could fail in the presence of condensation.

The module employed in our application was located in a bright area in order to be able to capture images. Because of this, the solar panel was able to generate a sufficient level of voltage and power to utilize a low cost linear battery charging circuit like LTC4057. Consequently, this circuit was used in the last version of the hardware structure of the node.

Finally, the module used in our study was ESP32 PSRAM Timer Camera, commercialized by M5 STACK. Initially, the battery was charged from the voltage of an USB connector. The hardware of the module was modified to use the solar panel for charging the battery.

The real time clock (model BM8563) was directly powered from the battery, having the following features:

- supply voltage between 1.8 V and 5.5 V,
- power consumption 0.25 ÷0.6 μA for 3.0 V power supply
- operating temperature between -40°C and 80°C.

The power supply module contained the following components:

- A voltage stabilizer built using the SY8089AAC circuit, a synchronous step-down DC-DC regulator.
- A 3.7 V Li-Polymer battery, model LP802030, with a capacity of 400 mAh and energy of 1.48 Wh. The battery is equipped with a protection circuit against voltage drop. Its maximum charging current is 200 mA, whereas the maximum continuous discharging current is 400 mA.
- A battery charger built using the linear charging circuit TP4057.
- A monocrystalline solar panel optimized for outdoor use, with a power output of 1W and an open circuit voltage of 8.2V.

A picture of the assembled IoT node is shown in Fig. 2.

Fig. 2 – Picture of the IoT node.

5. Node Communication

The node involving the ESP32 microcontroller used the Wi-Fi protocol with a radio frequency of 2.4 GHz for communication. The connection was done securely to the local router with a SSID and a password.

Two versions of data transmission were utilized, as depicted in Fig. 3:

- a HTTP POST to a script in GOOGLE DRIVE, transmitting images and non-visual information (the node name, the battery and the solar panel voltage). The image was transmitted encoded in Base64, in pieces of 1020 bytes, while the non-visual information was sent using URL parameters. The script decoded the image, created a unique name from the node name and the current date and time and saved it to GOOGLE DRIVE. The script also stored the non-visual data in the GOOGLE SHEETS application in the following way: for each transmission, a line containing the non-visual information, along with a timestamp was created and added to the sheet corresponding to the node name.
- the MQTT protocol, used to transmit data to a local MQTT server (MQTT Broker). The module published messages in threads with the following structure:
    - camera/<NOD ID>/ jpg – for image transmission,
    - camera/<NOD ID>/sensor/battery – for battery voltage,
    - camera/<NOD ID>/sensor/psolar – for solar panel voltage.

  <NOD ID> was the unique identifier of the node. A Python program, that connected to the local MQTT server and subscribed to threads created by the nodes was used to retrieve the transmitted data and store it on a disk.

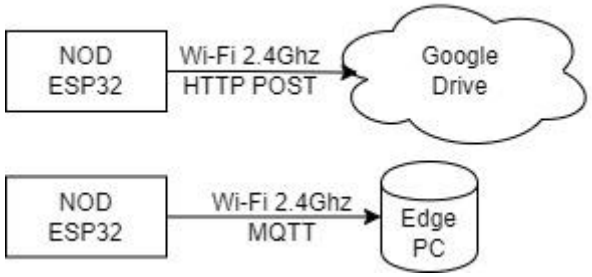

Fig. 3 – The two types of data transmissions used by different node versions.

6. Software

The program execution started when the microcontroller was powered by a real-time clock alarm. The microcontroller turned off its power when the program ended. The program couldn’t store or maintain the variables in the

memory from one run to another. The only way to transmit information between successive program runs was through the EEPROM memory of the microcontroller. The organizational chart of the program has a START point and one or more STOP points. The flowcharts depicting the two communication variants is shown in Fig. 4.

The program was responsible for taking a picture and transmitting it, along with the battery voltage and the effective voltage of the solar panel, to the storage server. The code was written in C language. The manufacturer of the microcontroller provided the ESP-IDF package, which contained the libraries, the source code, the examples, and the compilation and programming utilities. We used this version instead of the ARDUINO package due to the following reasons: the source code of the libraries was included in the project and one could create and utilize tasks and independent processes. The module manufacturer (M5STACK) offered the documentation, the schematic diagram, the software, and clear examples of use.

The initial part of the program was similar for both versions and included initialization, retrieving the communication parameters, reading the battery voltage, making a decision on whether to conserve power or continue transmitting the image, programming the real-time clock, and stopping the power supply. For the MQTT variant, the voltage of the solar panel determined whether to continue with capturing and transmitting the image or not. In the HTTP version, the time and date were read from the real-time clock. A synchronization was performed if the date was not valid or if it was the time of the month when synchronization occurred. Depending on the hour, the decision to capture and transmit the image or not was made.

The node transmitted the data hourly. To ensure the conservation of battery energy, the software in the microcontroller was programmed to limit data transmission during daytime hours, when relevant images could be taken. Attempts were made to optimise the power consumption by making use of the existing elements without introducing additional components. One option was to connect to a Network Time Protocol (NTP) server and retrieve the current date and time from it. We used the retrieved time to schedule the next alarm. During the daytime, the real-time clock alarm was set to go off every hour, while during the night, the next alarm was scheduled at 6:00 AM the following day. This solution generated additional energy consumption for each data transmission. The energy consumption of the node during NTP server reading was over 150 mA because the Wi-Fi communication was enabled. The duration of attempting to read the NTP server was 1 second. On average, the server reading succeeded after 2 to 4 attempts during the day and after 7 to 10 attempts in the evening, due to its response times. To avoid reading the NTP server during each transmission of the node, the date and time from the real-time clock were utilized. Real-time clock synchronization could be performed once a month using the date and time obtained from the NTP server.

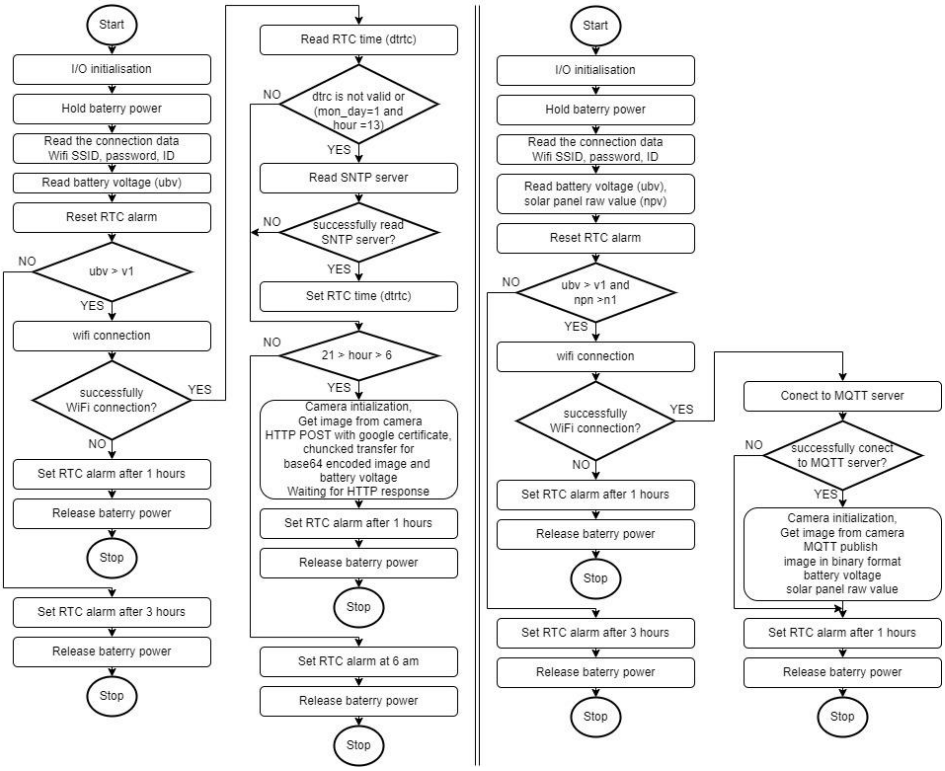

Fig. 4 − Structure of the node's software component.
HTTP variant to left, MQTT variant to right.

The above-mentioned version was implemented in the HTTP communication version. This time-based image capturing solution took images whenever the lighting conditions were very favorable for obtaining high-quality pictures. The issue with this approach was that the ambient light at a particular time differed depending on the day, month and weather conditions, thus affecting the picture quality. Optimization could be achieved by using a sunrise and sunset timetable, but this timetable needed to be adjusted according to the actual location of the node and the date of the daylight-saving time change.

The second option was to use the solar panel voltage, without using the time information, to determine whether there was enough light for taking pictures. This method was implemented in the MQTT communication version. Because it didn't use the time retrieved from the real-time clock or the NTP server, the drawback of this solution was that the microcontroller always set the alarm every hour. The microcontroller would wake up, read the panel voltage, make a decision to transmit or not a picture and, finally, program the real-time clock alarm to go off after one hour. These versions of the node software could

have been optimized, but were left as they were to facilitate easier analysis and tracking of their operation.

The manual start of the node was carried out by powering the microcontroller. The microcontroller set the alarm in the real-time clock and the awake and sleep logic started. That interconnection was interrupted in two cases: when the battery value dropped (and in this case, the clock lost the settings) or when the voltage was too low (in which situation the microcontroller failed to set the alarm in the real-time clock). To avoid those issues, if the voltage on the battery was below a certain threshold, the only task performed by the node involved awaking (triggering the alarm) after 3 hours. This approach aimed to save energy by reducing the power consumption and waiting for the battery to charge from the solar panel.

To avoid losing the settings of the real-time clock, the node's power supply block could be changed by powering the real-time clock from a separate battery, different from the node's battery. However, in this case the connection would be interrupted if at the node's waking up moment (an event generated by the real-time clock alarm), the Li-Po battery wasn't able to provide the necessary energy to the microcontroller to reprogram the alarm in the real-time clock.

7. Node Power Consumption Analysis

The tests were conducted first in the laboratory and next in the field. In the lab, the node was supplied by a 5 V voltage source connected in series with a 1 Ω resistor. This resistor was utilized to measure the instantaneous current of the node through the Analog Discovery2 oscilloscope. For the power consumption measurement, the battery, battery charging circuit and solar panel were removed. The operating voltage for the ESP32 microcontroller and the camera-microcontroller assembly was 3.3 V. The previous references to the power consumption were given in terms of current values. To calculate the power consumption, it was necessary to consider the voltage supplying the module and the specific source method used to achieve the 3.3 V voltage.

Because the power supply in the node was realized using a synchronous step-down regulator (SY8089AAC), the power consumed by the node was slightly dependent on the variation of the supply voltage of the node, while the power consumed by the module was relatively independent of the battery voltage.

Next, we will detail the conditions under which the power consumption tests were performed in the laboratory to compare different data transmission methods:

- The power consumption was measured sequentially for each node variant
- At the time of measurement, we ensured similar conditions, viz.:

- Each node version was placed in the same position (at different times) and was connected to the same Wi-Fi router.
- Each node version transmitted the same color image with a resolution of 1920x1080 pixels. The resulting compressed JPEG image file had a size of about 300 kbytes.

The tests were performed for the following 4 data transmission approaches:

- A node transmitting via MQTT;
- A node transmitting via HTTP;
- A node transmitting via HTTP, executing clock synchronization with the NTP server (shortest time);
- A node transmitting via HTTP, executing clock synchronization with the NTP server such that the communication had multiple retries (longest time).

For the last two data transmission methods, 15 measurements were performed and the shortest and the longest registered times were recorded. The node power consumption was dependent on asynchronous phenomena that could not be controlled or predicted, generated by the communication with the NTP servers and the Google server. The current consumed during sleep was primarily attributed to the power consumption of the real-time clock that depended on both the supply voltage and the temperature. Under conditions of a maximum voltage of 4.2 V and a temperature not exceeding 60°C, the current consumption consistently remained below 0.5 μA. The maximum power consumed was 2.1 μW and the energy consumed by the node in the sleep state for a one-hour duration was less than 2.1 μWh.

The data retrieved by the oscilloscope were also available as a comma separated values (CSV) file. We used these data to calculate the average values of the current over a period of time and its integral value for the entire period. The data processing part was carried out using a program written in Python. The node current consumption for a transmission could be calculated by integrating the current variation.

Fig. 5 presents the current variation for the MQTT transmission, the waveform being captured by an oscilloscope. The current exhibited non-constant behavior with a high frequency of variation, displaying a band of values. However, it was not possible to deduce the average or the integral value of the current consumption from the given figure.

To visualize the average current for the node using the MQTT transmission, we calculated the average values for 100 ms periods, as depicted in Fig. 6. The various stages of the program are marked on the figure and detailed below:

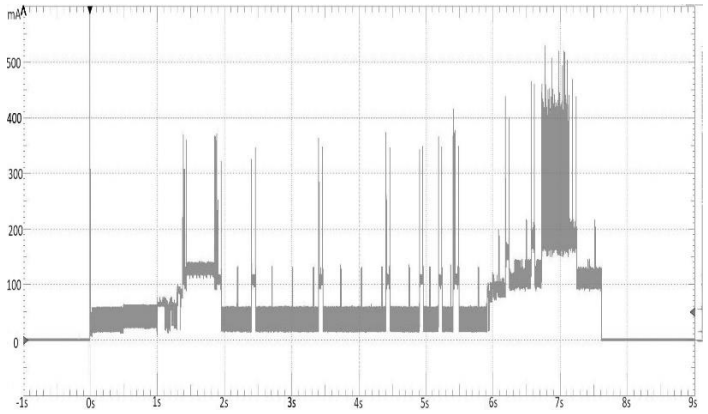

Fig. 5 – The current consumption for the IoT node with the MQTT communication.

A - Powering up the microcontroller, initializing ports and peripherals, reading the real-time clock, and the stored data.

B - Initiation of the connection to the Wi-Fi router was achieved through the use of SSID and password.

C - The program entered a waiting state in which the node was registered in the network and assigned an IP address.

D - Initiation of the connection to the MQTT server occurred, followed by the acquisition of images from the camera.

E - Data transmission to the MQTT server took place.

F - The program awaited the confirmation of the MQTT transmission.

G - The final operations were executed, including the setting of the activation time in the real-time clock.

H - The program termination was initiated, leading to the subsequent power-off of the microcontroller.

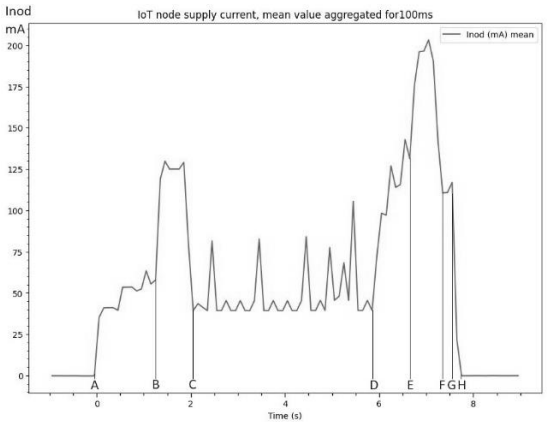

Fig. 6 – The average current consumption per 100 ms for the IoT node with the MQTT communication.

For the HTTP transmission node, the current consumption during various stages of the program was observed, as illustrated in Fig. 7, where:

A - Powering up the microcontroller, initializing ports and peripherals, reading the real-time clock, and the stored data.

B - Initiation of the connection to the Wi-Fi router was achieved through the use of a SSID and a password.

C - The program entered a waiting state in which the node was registered in the network and assigned an IP address.

D - The images were acquired from the camera.

E - The initiation of the connection to the GOOGLE server was produced through the HTTPS protocol.

F - The image was transmitted in batches of 1020 bytes Base64 encoded data to the server.

G - The image transmission was completed, followed by the closure of the connection, with the program waiting for the confirmation from the server.

H - The confirmation from the server was received.

I - The final operations were executed, including the setting of the activation time in the real-time clock.

J – The program termination was initiated, leading to the subsequent power-off of the microcontroller.

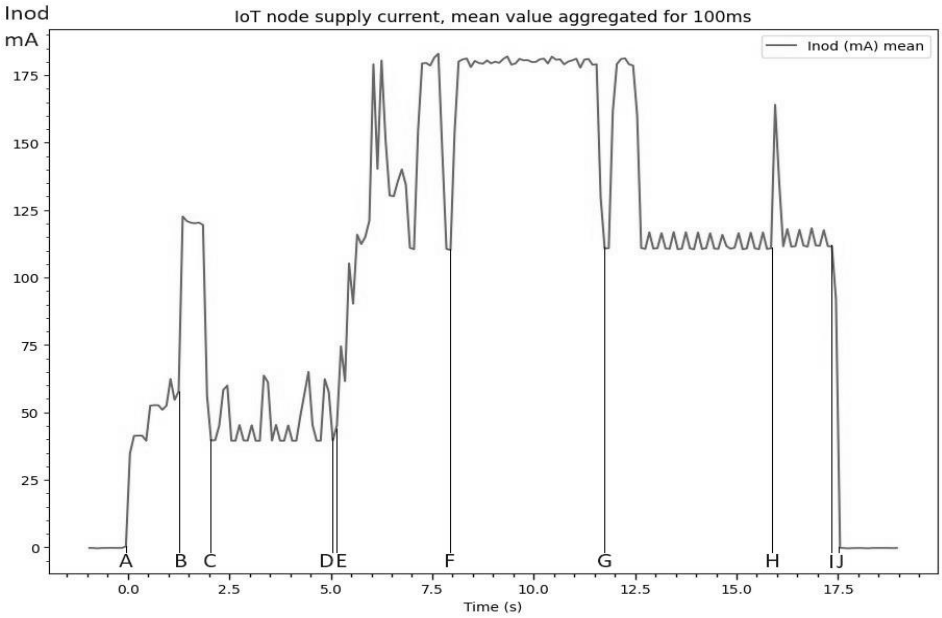

Fig. 7 – Mean value of current consumption per 100 ms for the IoT node using HTTP and NTP communication.

The effective data transmission duration was observed to be approximately 0.5 s for MQTT and 3.4 s for HTTP. This discrepancy arose from the fact that the MQTT transmission was binary and continuous, while the HTTPS transmission involved Base64 encoding (converting binary data to ASCII) and was performed in batches of 1020 bytes. Additionally, it is important to note that the HTTPS protocol employed encryption and security measures through the use of digital certificates.

When computing the energy consumption, we disregarded the voltage drops across the 1 Ω resistor and the voltage fluctuations of the 5 V power supply. In Table 1, the energy consumptions were summarized for the MQTT and HTTPS transmission modes, both with and without connection to the NTP server. A significant lower energy consumption was evident when transmitting images via the MQTT protocol. Additionally, connecting to the NTP server for date and time retrieval consumed between 0.250 mWh and 0.800 mWh.

Table 1

Energy consumption of nods

Communication	Integration of current [mAs]	Energy [mWh]
MQTT	*570.22*	*0.792*
HTTP	*1467.98*	*2.039*
HTTP, NTP (minim)	*1655.52*	*2.299*
HTTP, NTP (maxim)	*2026.61*	*2.815*

7. Battery and Solar Panel Considerations

In this section we are dealing with the power consumption of the node when deployed in the field, i.e. in an orchard. In accordance with the data acquired through the oscilloscope, it could be observed that the current peaks did not exceed 600 mA and the current consumed by the module remained below 200 mA. According to the catalog data for the standard Li-Po battery, a battery of at least 300 mAh met the requirements. We selected a 400 mAh battery to mitigate the progressive degradation of the battery parameters over time and to compensate for the impact of low temperatures.

The battery voltage provided insights into the stored energy within the battery. At the time of the transmission, both the battery voltage and the solar panel voltage were read. The solar panel voltage represents an instantaneous value, providing information about its illumination at the time of reading. In contrast, the battery voltage is an integrative measure dependent on the energy derived from the solar panel.

In Fig. 8, the variation of the battery voltage, solar panel voltage, and the number of acquired data points from April 14, 2023, to October 15, 2023 is depicted for a node with HTTP transmission. The same information for the period May 7 to May 13, 2023, is presented in Fig. 9.

According to Fig. 8, the maximal values of the battery and solar panel occurred during April, May, and a portion of June, diminishing through October. This phenomenon was attributed to the placement of the solar panel within the designated area, in proximity to vegetation. The panel experienced shading effects induced by the flourishing growth of the plants. Specifically, the solar panel was situated beneath the canopy of an apple tree.

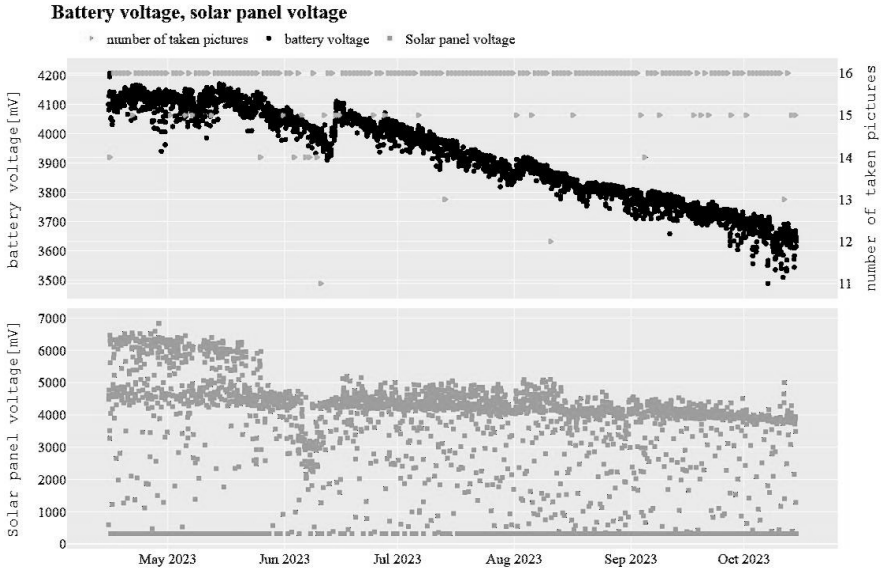

Fig. 8 – Graphs depicting the variation of the battery voltage, the number of taken pictures (upper), and the solar panel voltage (lower) are presented for the node with HTTP transmission.

In the graph illustrated in Fig. 9, the daily emission count (number of captured images) is also displayed. A daily acquisition rate of 16 images can be observed. Sometimes, this value was reduced on certain days, these diminishments exhibiting no correlation with the battery or the solar panel voltages. The absence of recordings may be attributed to other factors, such as the lack of internet connectivity or non-responsive storage servers within the allocated time frame. The emissions around 20:40 on May 7th and 12th were notably absent from the presented graph, most likely due to server non-responsiveness. This phenomenon is also evident in Fig. 9, denoted by markers 1 and 2 on the emission number graph.

As noticed from Fig. 9, the battery voltage exhibited a periodic variation throughout the day, peaking in the morning until midday and subsequently declining towards evening. The solar panel voltage was elevated

during illuminated periods in the day and was diminished during mornings and evenings.

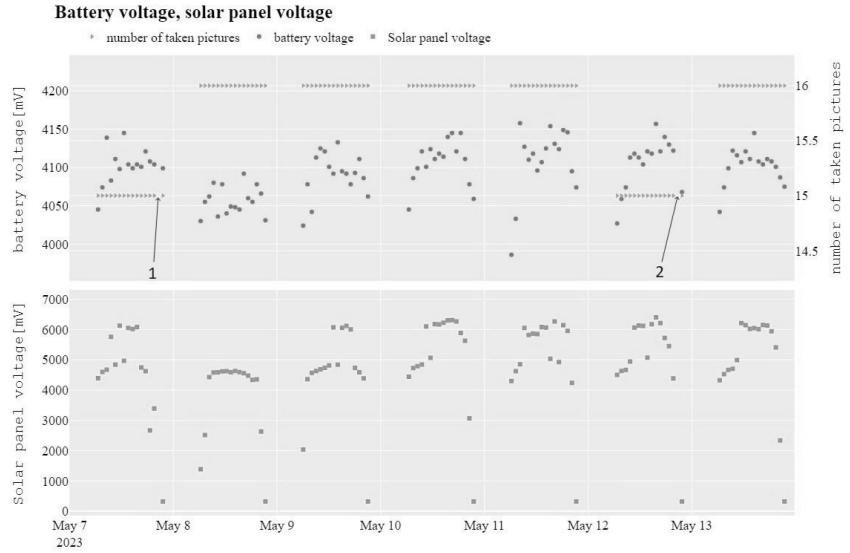

Fig. 9 – Details for the graphics presented in Fig. 8 are provided for seven days from May 7th to May 13th.

The manufacturer provided graphs of the battery voltage variation under no load, based on the battery charge for various temperatures and charging/discharging modes. These graphs could not be used because we could not disconnect the battery for taking measurements, and we did not know whether the node operated in battery charging or discharging mode. The battery voltage reading was performed only once at the program's startup before establishing the Wi-Fi connection. The transmission power consumption depended on the transmitted image and the connection latency to the server.

The node performed 16 image transmissions each 24 hours, from 6 AM to 9 PM. The node's power consumption was determined by the daily transmission and sleep mode power consumption. The daily emission time was at most 25 seconds for one transmission, so for 16 transmissions, the total time was at most 400 seconds. This means that the daily sleep time can be considered to be approximately 24 hours by ignoring the emission time. The power required for these 16 transmissions was approximately 32mW. The power in sleep mode for one day was calculated by multiplying 24 hours by the maximum energy consumption per hour, which was 2.1 μWh. This resulted in 0.0504 mWh. Therefore, the power required for one day was 32.0504 mWh. It was observed that the power in sleep mode was very small compared to the power required for the emissions.

If the battery was not charged by the solar panel, then the node could emit for at most 45 days, assuming the battery was fully charged to the maximum capacity of 1480 mWh. The energy consumed by the node in one day was equal to the energy generated by the solar panel within 3 minutes, provided that it operated at maximum power (voltage above 4.5 V and a current of 170 mA).

In Fig. 10, the graphs measuring the battery and solar panel voltage for seven days in October are depicted. It can be observed that the voltage on the solar panel was mostly below 4 V. The day was short, and the sun set no later than 6:42 PM.

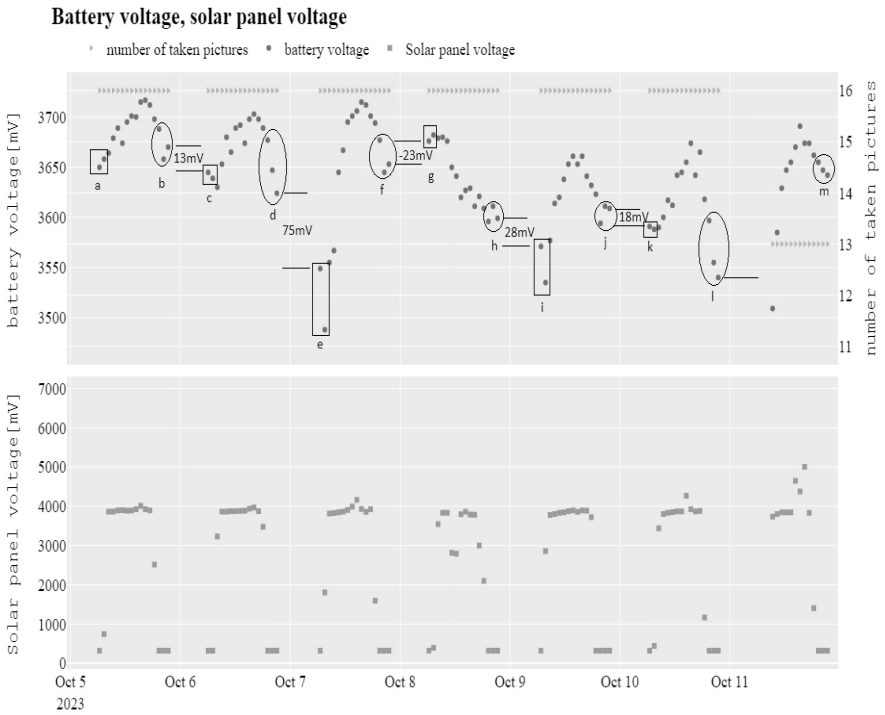

Fig. 10 – Details for the graphics presented in Fig. 8 are provided for seven days from October 5th to October 11th.

Therefore, data transmissions at 7, 8, and 9 PM were made with inconclusive, dark images. It could be assumed that the images had approximately the same length. The first transmission at 6 AM was also inconclusive because the sun rose after 7:12 AM. The areas marked with rectangles and labeled *a, c, e, g, i, k* represent the emissions at 6 AM, having approximately the same image length. It may be observed that the variation in voltage between the emissions at 6 AM and the ones at 7 AM had no correlation with the solar panel voltage.

The areas marked with ovals labeled *b, d, f, h, j, l, m* represent the emissions at 7, 8 and 9 PM. The variation of the battery voltage between these emissions depended on other external factors such as temperature and the server's latency. The horizontal lines marked the voltage of the transmissions at 9 PM and 6 AM from the next day. The voltage difference is indicated between the two lines. On October 11th, the voltage was below the minimum and there was no emission at 6 AM.

In zone *d* the latencies to the servers were high and emissions consumed more energy. The difference of 75 mV represented the voltage drop of the battery for the last emission at 9 PM. During the next morning, the server latency was not lower; in zone *e* there was a pronounced drop in the battery voltage after the emission at 6 AM. The transition from October 7th to October 8th showed that the battery voltage increased instead of decreasing.

Finally, we computed the correlation between the battery voltage and the solar panel voltage within the timeframe spanning from April 15, 2023 to October 15, 2023. Notably, the battery voltage exhibited a better correlation with the solar panel voltage from two hours prior, as shown in Fig. 11. The designated waiting time for a new emission attempt, in the event of minimum voltage, was set at three hours. This temporal interval represented a balanced compromise between data acquisition and the duration required for the battery recharging.

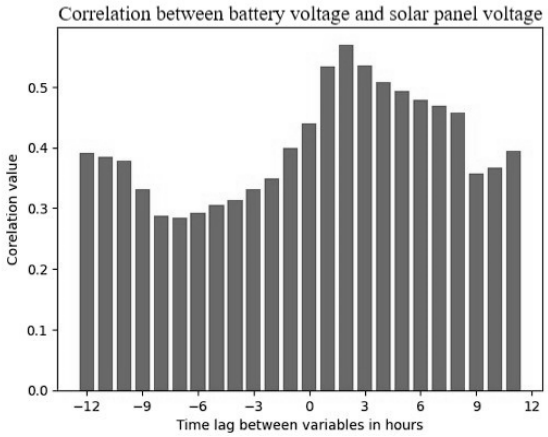

Fig. 11 – Correlation between battery and solar panel voltage values.
The positive time lag means the correlation of the battery voltage with the older solar panel voltage.

The analyzed node was exposed to suboptimal operating conditions. These conditions were maintained in order to analyze the node's behavior. The node ceased emitting after December 6th at 11:36 a.m. The node's battery dropped below the minimum protection value, and the voltage at its terminals

was null. The real-time clock was not powered anymore and did not generate the alarm that started the microcontroller. Even though the battery was recharged in the meantime, the initialization of the node's operation could only be performed manually. In Fig. 12, the irregular node operation due to battery discharge can be observed.

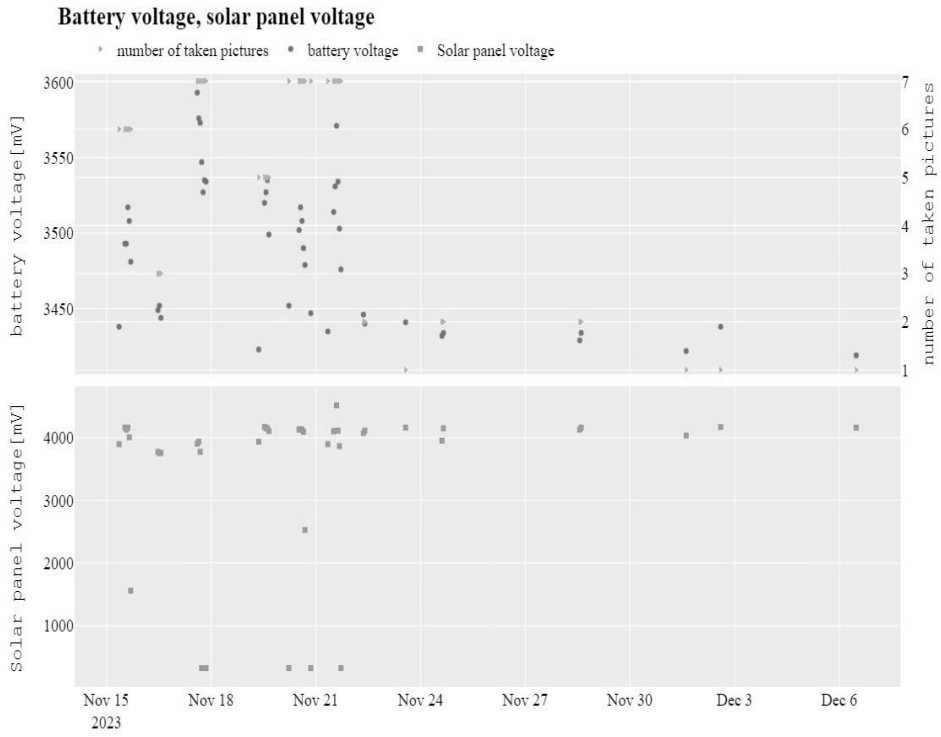

Fig. 12 – Details for the graphics presented in Fig. 8, provided for 21 days from November 15th to December 6th.

The number of emissions varied significantly depending on the days. There were periods during which the node decided not to emit due to the battery's discharge state, as determined by the battery voltage. After December 6th, the node did not emit anymore, and it was not possible to determine until which day the node was still functional. Its cessation of operation was caused by two major factors: low temperatures (which accelerated the battery discharge) and weather conditions such as snow and precipitations (which conduced to the solar panel's malfunction). It can be observed that the maximum voltage value on the battery was around 4 V at that time. This was caused by dirt present on the solar panel's surface. No maintenance intervention was carried out on the node, so the solar panel was not cleaned; it was

intentionally left unmaintained, to study the behavior of the system in diverse real conditions.

8. Conclusions

This paper presented an analysis regarding the power consumption of multiple versions of an IoT node, utilized for taking and transmitting pictures from an orchard. The IoT node was constructed using accessible and cost-effective components. The selection of the components was made with the objective of maintaining an optimal balance between cost and quality, ensuring the ease of the node's assembly. The verification of the functionality took place in a controlled laboratory environment, followed by the relocation of the nodes outdoors to monitor various plants. The extended operation of the node served as the verification of its correct functionality.

The type of battery was chosen to deal with a variation in the current of the node up to 400 mA. The daily energy consumption of the node was approximately 2 mWh per emission and 0.05 mWh in the sleep mode. With 16 emissions per day, the fully charged battery could power the node for up to 45 days without relying on the solar panel charging. To increase the node's working time, the number of emissions could be reduced. The solar panel was theoretically oversized. In practice, due to shading and dirt, the 1 W (5.5 V) panel failed to charge the battery.

The insufficient charging of the node's battery was attributed to the low voltage of the solar panel. A future optimization strategy would consist in using a panel with the same power output but with a higher voltage. A minimum of 4.2 V is necessary in order to utilize a linear charging circuit for the battery.

The energy consumption was not the same for each transmission. The image was compressed into JPEG format, and its size depended on its content. The MQTT transmission's energy consumption rate was 2.5 times lower than the one of the HTTP transmissions, but required an additional component to retrieve and save the image. For an edge solution, the transmission times were stable. In contrast, the cloud transmission was simpler to implement and maintain, but the server access times were not guaranteed.

The power consumption was interesting to track throughout the year, including during winter, to observe the node's behavior in the context of temperature variations. Additionally, in recent times, we encountered situations in which vegetation appeared during winter due to abnormal temperatures.

In terms of operation, the node functioned reliably without requiring periodic maintenance or upkeep.

An idea for future development is to determine the battery's charge status based on the battery voltage, similar to (De Angelis *et al*., 2022).

REFERENCES

Avbentem.github.io, *Airtime calculator for LoRaWan,* https://avbentem.github.io/airtime-calculator/ttn/eu868/222.

Ching-Chuan Wei, Pei-Yi Su, Shu-Ting Chen, *Comparison of the LoRa Image Transmission Efficiency Based on Different Encoding Methods*, International Journal of Information and Electronics Engineering, 10, 1-4 (2020).

Correia Pedro, Gomes Marcela, Martins Gabriel, Panda Renato, *Low Cost LoRaWAN Image Acquisition System for Low Rate Internet of Things Applications*, 2022 2nd International Conference on New Technologies of Information and Communication (NTIC), (2022).

De Angelis Paolo, Carbone Paolo, Santoni Francesco, Vitelli Michele, Ruscitti Luca, *On the Usage of Battery Equivalent Series Resistance for Shuntless Coulomb Counting and SOC Estimation,* Batteries an Open Access Journal from MDPI, Volume 9, Issue 4 April 2023 p. 238 (2023)

Jawad H.M., Nordin R., Gharghan S.K., Jawad A.M., Ismail M., Abu-AlShaeer M.J., *Power Reduction with Sleep/Wake on Redundant Data (SWORD) in a Wireless Sensor Network for Energy-Efficient Precision Agriculture*, Sensors, vol. 18, no. 10, Art. no. 10, oct. 2018 (2018).

Konain Mahvish, Janapati Ravichander, Ahmed Syed Mushtak, Ahmed Mohammad Mustafa, *IoT based Solar-Powered Mushroom Farming for Sustainable Agriculture*, 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), p. 944-948 (2023).

Liu C., Sun Y., *Tracking Strategy of Maximum Power Point Based on Three-Stage Variable Step-Size Incremental Conductance Algorithm*, Laser Optoelectron. Prog 2018, vol. 55, p. 420-426 (2018).

Meng Yang, Chen Zunliang, Cheng Hui, Wang Enpu, Tan Baohua, *An Efficient Variable Step Solar Maximum Power Point Tracking Algorithm*, Energies, vol. 16, p. 1299 (2023).

Palniladevi P., Sabapathi T., Kanth D. Ashwen, Kumar B. Prakash, *IoT Based Smart Agriculture Monitoring System Using Renewable Energy Sources*, 2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN), p. 1-6 (2023).

Sun Chunhu, Ling Jing, Wang Jing, *Research on a novel and improved incremental conductance method*, Scientific Reports, vol. 12, no 1, p. 15700 (2022).

Takaaki Kawai, *Video Slice: Image Compression and Transmission for Agricultural Systems*, Sensors, 21, 3698 (2021).

Wang Enpu, Xiao Lu, Han Xu, Tan Baohua, Luo Lina, *Design of an Agile Training System Based on Wireless Mesh Network*, IEEE Access, vol. 10, p. 84302-84316 (2022).

Wikipedia, *IEEE 802.11*, https://en.wikipedia.org/wiki/IEEE_802.11

Zheng Beitian, Tian Shihao, Rao Caijun, Wang Qinyang, Liang Can, Tan Baohua, *Wireless Smart Greenhouse Management System Based on Multi-sensor of IoT*, 2023 IEEE 3rd International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB), p. 95-99 (2023).

STUDIU PRIVIND CONSUMUL DE ENERGIE AL UNUI NOD IoT CARE ACHIZIȚIONEZĂ IMAGINI ȘI MODALITĂŢI DE OPTIMIZARE A ACESTUIA

(Rezumat)

Lucrarea analizează consumul de energie al unui nod IoT care captează imagini și le transmite către un mediu de stocare în cloud. Sunt propuse diferite versiuni de implementare ale nodului IoT, concentrându-se pe utilizarea componentelor comune pentru a obține eficiența costurilor și a minimiza cerințele de întreținere. Variantele de noduri realizate au fost testate inițial în condiții de laborator și apoi puse în funcțiune și desfășurate în aer liber, fiind amplasate într-o livadă unde au rămas operaționale timp de 7 luni în diferite condiții meteorologice. Soluția propusă folosește un microcontroler ESP32 și o cameră cu o rezoluție de 1,3 Megapixeli ca nod de achiziție de date a cărui energie este asigurată de o baterie Li-Po, încărcată printr-un panou solar. Comunicarea Wi-Fi a fost folosită pentru a transmite imaginile către server. Consumul de energie al nodului a fost evaluat pentru diferite variante de optimizare software folosind modurile de transmisie HTTP sau MQTT. Consumul de curent instantaneu al nodurilor a fost măsurat în laborator pentru a identifica consumul de energie pentru fiecare fază a execuției programului. Experimentele au arătat că modul de transmisie MQTT consuma mult mai puțină energie decât modul HTTP. Au fost efectuate experimente suplimentare în care au fost analizate variația tensiunii bateriei, a tensiunii panoului solar și a numărului de emisii zilnice în perioada examinată, precum și corelațiile dintre aceste mărimi.