engineering proceedings

MDPI

Proceeding Paper

Prototyping Bespoke Sensor Industrial Internet-of-Things (IIoT) Systems for Small and Medium Enterprises (SMEs) †

Nik Petrov [1], Tim Mulroy [2] and Alexander N. Kalashnikov [2,*]

[1] Data Science and Technical Services Department, Safefood360, Dublin, Irlande; npetrov.ie@gmail.com
[2] Department Engineering and Maths, Sheffield Hallam University, Sheffield S1 1WB, UK; t.j.mulroy@shu.ac.uk
* Correspondence: a.kalashnikov@shu.ac.uk
† Presented at the 10th International Electronic Conference on Sensors and Applications (ECSA-10), 15–30 November 2023; Available online: https://ecsa-10.sciforum.net/.

Abstract: The paper aims to share our experiences gained from working on multiple industrial-academic collaborative projects within the Digital Innovation for Growth (DIfG) regional program. This initiative provided academic expertise to low-resource SMEs. The projects primarily revolved around measuring various process or structural health variables. The subsequent wireless reporting of these results to an online dashboard and generating alert messages when variables exceeded predefined thresholds were central to our work. Due to the diverse nature of our partners' requirements, there was no one-size-fits-all solution for the considered use cases. We will delve into our utilization and insights regarding various IoT-related tools and technologies. These include ESP32 WiFi-enabled microcontrollers, WiFi Manager, NTP time service, watchdog timers, Adafruit IO dashboards, the Twilio SMS gateway, as well as LoRa modules and networks such as TNT and Helium. By effectively combining these tools and technologies, we successfully completed prototypes that enabled testing of the devices on-site.

Keywords: sensor IIoT; IIoT prototyping; IIoT for SMEs; IIoT low power; IIoT SMS alerts; IIoT LoRa communication; IIoT reliability

Citation: Petrov, N.; Mulroy, T.; Kalashnikov, A.N. Prototyping Bespoke Sensor Industrial Internet-of-Things (IIoT) Systems for Small and Medium Enterprises (SMEs). *Eng. Proc.* **2023**, *56*, x. https://doi.org/10.3390/xxxxx

Academic Editor(s): Name

Published: 15 November 2023

Copyright: © 2023 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Advancements in low-cost WiFi-enabled microcontrollers (MCU) and cloud infrastructure have generated a growing interest in the Industrial Internet of Things (IIoT) among SMEs. This interest primarily stems from the potential benefits of storing and visualizing sensor data in the cloud, thereby eliminating the need to maintain and transfer data from local data loggers.

However, many SMEs lack the capability to independently develop an IIoT system and are uncertain about the financial feasibility of funding external development. In such cases, collaborative projects between industry and academia, supported by government or regional sources, emerge as a viable option for prototyping IIoT systems.

This paper presents our observations and findings based on several sensor IIoT prototyping projects, i.a., conducted under the auspices of the Digital Innovation for Growth (DIfG) programme [1]. Two projects were concerned with temperature sensing, and one with sensing impacts.

IIoT technology has garnered significant interest from industrial partners, who have shown a willingness to engage in basic maintenance tasks for their sensor-based IIoT systems, such as on site commissioning and Arduino script uploads. Their main priorities include cost predictability for both initial implementation and ongoing maintenance, utilization of readily available off-the-shelf components for spares and replacements, and suitability of the prototype for operation within their industrial environment.

Developers, on the other hand, emphasize the importance of having access to highly functional components that can be easily assembled. They also seek low-cost or free options for related services, accompanied by example projects, application notes, and similar resources to aid in their development process.

In Section 2, we delve into the fundamental setup that proved to be adequate for all three projects. Section 3 highlights firmware features added for usability and reliability. The utilization of text messages to deliver alerts and alarms to users is addressed in Section 4. Section 5 focuses on achieving operation off WiFi. Finally, Section 6 provides a summary and conclusion of the paper.

2. The Basic Sensor IIoT System Design

An IIoT sensor system can be seen as a convenient alternative to a conventional data logger. Instead of simply recording measurements at set time intervals and storing them in non-volatile memory to prevent data loss during power outages or accidental shutdowns, this system transmits each data point to the cloud. This enables reliable long-term storage and archiving of the data, as well as instant access from any connected device for checking or trend analysis, assuming server-side visualization is available. By utilizing this use case, it becomes possible to significantly reduce power consumption by allowing the sensor and communication link to enter a sleep mode after processing each newly acquired data point.

From a hardware perspective, it is essential for a prototype of this system to be enclosed in a secure casing with robust connections and the necessary IP rating for ingress protection. To simplify the process of commissioning, maintenance, and relocation, it is generally preferable for the prototype to be powered by a battery. For battery-powered prototypes, it is important to utilise the deep sleep MCU mode, which has certain restrictions compared to the light sleep mode more suitable for mains-powered devices. Additionally, the devices should have built-in battery charging and protection circuits. An ideal prototyping ecosystem should also offer a wide range of well-documented link and sensor modules that can be easily attached or detached. However, off-the-shelf ecosystems with such capabilities are limited in availability. In our case, we developed the prototypes using the M5StickC ESP-32 based device, which features a built-in 120 mAh battery in the standard configuration, the option to add an additional 186,50 2200 mAh battery, and a selection of dedicated HAT and general-purpose Grove-interfaced modules [2]. This hardware only operates in the 2.4 GHz WiFi band but it was not an issue for our projects (Figure 1).

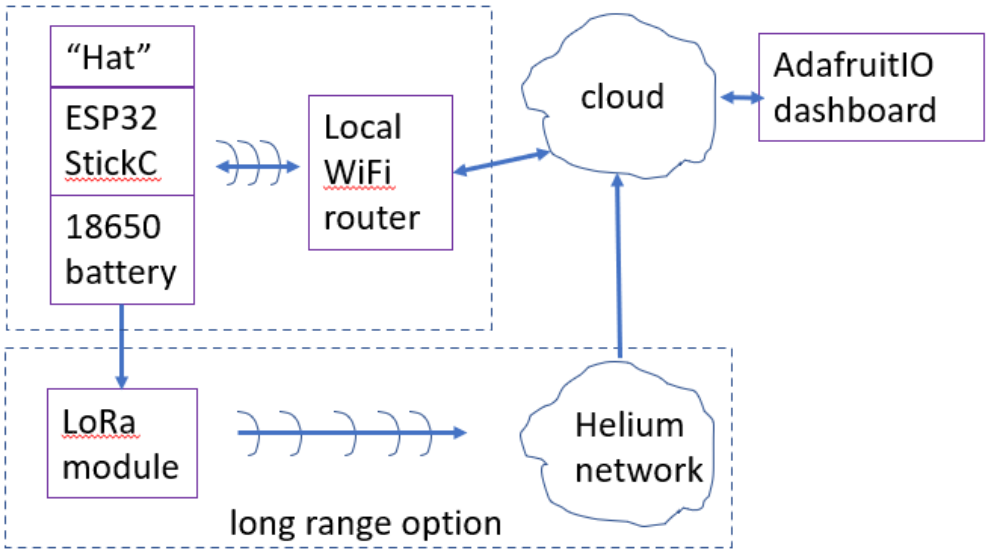

Figure 1. Block diagram of the hardware setup and wireless connections that were fully prototyped.

To complete the WiFi-capable hardware setup, it is necessary to integrate a cloud backend. The small and medium-sized enterprises (SMEs) we collaborated with did not find dedicated company servers or large-scale IoT servers designed for diverse, high-volume industrial clients appealing. These servers often require configuration and administration and lack user-friendly dashboard builders. After careful evaluation, we opted for AdafruitIO as our cloud provider. AdafruitIO stood out due to its transparent pricing structure, extensive documentation, and the availability of a free tier for experimentation. An example of the developed user front end (dashboard) and developer's back end (feed) are presented side-to-side in Figure 2. Additional factors influencing our decision were discussed in [4], Section 5, where another dashboard is presented.

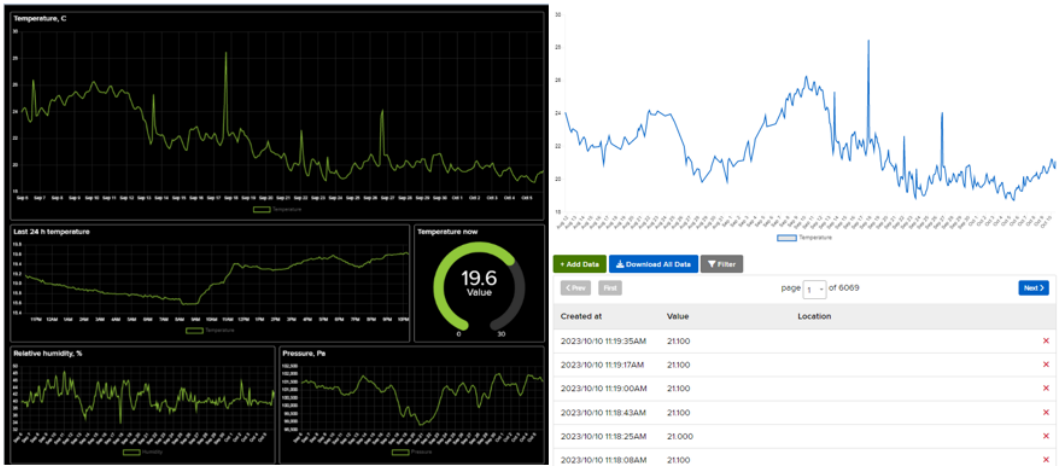

Figure 2. An example of a dashboard (user's front end, **left**) and a feed (developer's back end, **right**) on Adafruit IO.

In summary, the prototyped sensor IIoT systems follow a measure-communicate-sleep cycle using the M5Stick ecosystem and leverage the AdafruitIO cloud server. Considering that the estimated current consumption of the M5Stick in deep sleep mode is around 2–3 mA [5], line 36, it has the estimated capability to run on a fully charged additional 18650 battery for around one month. This assumes that the measurement-communication phase of the cycle, which consumes more than 100 mA, occurs infrequently, such as once per hour for a very short period of time.

3. Firmware Features Added for Usability and Reliability

The initial development of the firmware involved using the Arduino IDE and AdafruitIO library [6]. However, at a later stage, the library ceased to function, leading to the development and sharing of a POST-based workaround [7]. This workaround was implemented in all three projects, and we encountered no data loss or communication issues.

Occasionally, the MCU would hang up when connecting to the WiFi access point, but due to the intermittent nature of this fault, we chose not to fully investigate it. Instead, we implemented a watchdog timer that resets the MCU after set time. If the connection after a watchdog reset is re-established, the code proceeds to the loop function, where the watchdog timer is cleared.

To improve user-friendliness, several significant additions were deemed highly beneficial:

- The use of WiFi manager [8] was implemented to eliminate the necessity for embedding WiFi credentials into the code. This approach offers flexibility during installation, streamlines the process of reconnecting in case of password changes, and permits the use of devices at different locations without requiring reprogramming;
- It was observed that the device's built-in timer, responsible for awakening it from sleep mode, exhibited an inaccuracy on the order of approximately 4 min over a 24-

h period, which proved to be unsatisfactory for our partners. To address this issue, we implemented a solution involving the calculation of the time remaining until the next wake-up event, following the acquisition of precise time readings from the network time protocol (NTP) server. Subsequent to this adjustment, the readings were obtained with an acceptable margin of uncertainty, typically within a few seconds;

- Welcome messages and connection status notifications were presented to assure users of the system's correct functioning, particularly given that the screen was subsequently powered off to conserve energy while in sleep mode.

4. Communicating Alarms/Alerts

While transitioning data logging to an online platform is valuable, it is important to note that, in the context of two out of three IIoT systems, generating distinct messages was necessary when readings fell outside predefined limits. Adafruit IO offers a notification service for account holders, allowing them to receive alerts via either email (accessible for trial accounts) or SMS (requires an AdafruitIO+ paid account), as depicted in Figure 3:

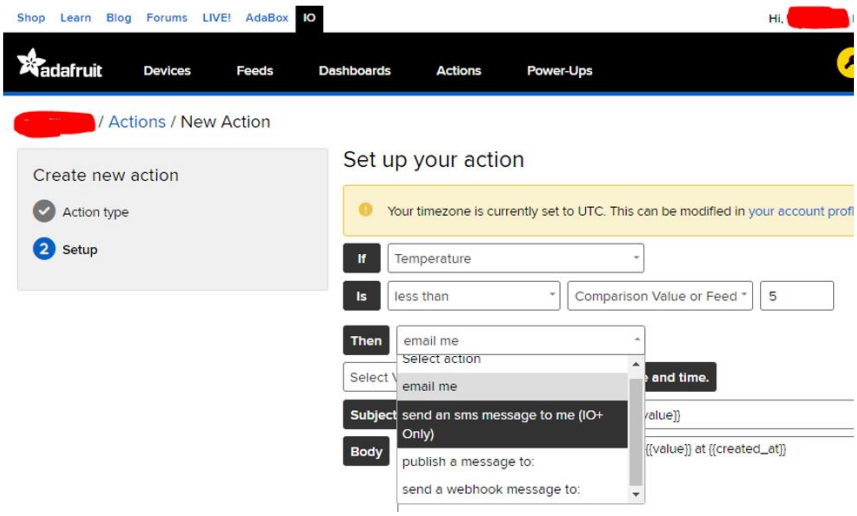

Figure 3. Setting notifications for a reading getting out of set bounds on AdafruitIO.

Email messages can also be sent using the Arduino reference library EmailSender [9].

Another possibility of notifying the end user of something requiring attention is by sending an SMS to their phone number. There are several internet-to-SMS service providers available, we used Twilio in our developments because of their international availability, good number of examples for various programming languages and a possibility to try it for free using sign up credit. We used an Arduino library [10] following a detailed tutorial [11]. The Twilio messages were reliably delivered inside both the UK and Ireland but we could not manage operating it cross-border. Figure 4 presents a console's output and some of the messages received using this service:

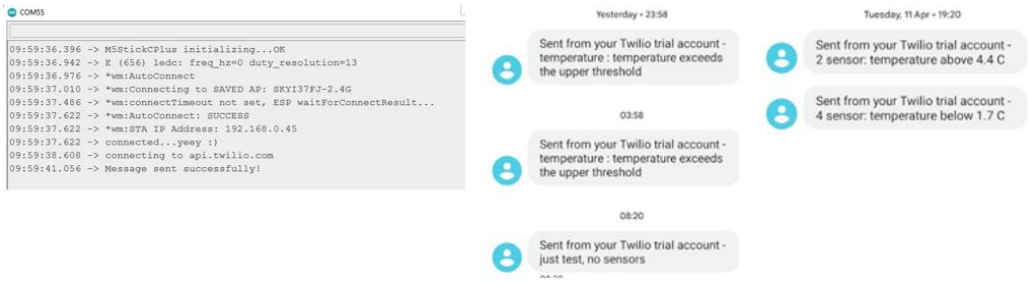

Figure 4. Operating SMS using Arduino library (console output on the left, examples of received messages on the right).

5. Operating IoT Devices off WiFi

While WiFi networks are prevalent at most sites, certain end users might not have access to them or may be reluctant to share their login information due to the associated inconvenience and security considerations. An alternative approach involves the use of wireless modem modules to connect to existing mobile networks with extensive coverage. However, fully-featured LTE modems tend to be relatively expensive and power-hungry, making them less suitable for IIoT purposes. On the other hand, NB-IoT modems, better suited for these applications, are not yet widely supported, with only one UK provider offering them at the time of this writing. Additionally, managing SIM cards (acquisition, registration, billing, security, etc.) poses additional burdens.

Consequently, we initiated an exploration of commercially available radio modules engineered for extended-range communication. Despite the seemingly favorable specifications evident in datasheets and select online demonstrations, we confronted formidable challenges when striving to achieve communication distances surpassing 100 m within a lightly urbanized setting. This was particularly pronounced during our utilization of peer-to-peer module configurations, inclusive of both nRF24L01 + modules, with and without RF amplification, and some LoRa modules.

An effective approach entailed establishing connections with a publicly available network characterized by extensive coverage. Our evaluation encompassed both the LoRa-based The Things Network (TTN), which offers open access, and the subscription-based Helium. It is noteworthy that both of these networks adhere to a decentralized model, relying on nodes privately procured and managed by network associates. Helium was further explored by us because of substantially wider coverage at reasonable cost.

To investigate this possibility, we employed an Arduino library tailored for the LoRa-E5 Grove-connected modules [12] in accordance with a comprehensive tutorial [13]. During our coverage and reliability assessment, the module was woken up and transmitted data through three train rides and two taxi journeys. It transmitted two bytes, deliberately incremented by 5 at each wake-up cycle. While the data flow from the Helium console to a Google spreadsheet did exhibit occasional interruptions and resets, it ultimately affirmed the network's overall resilience as shown in Figure 5.

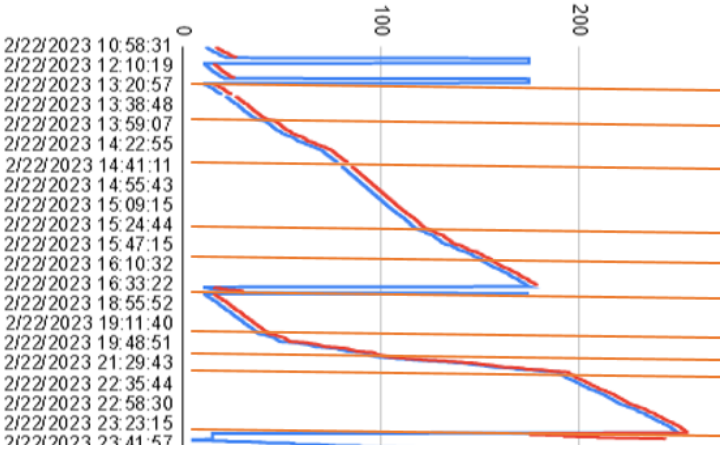

Figure 5. Operating SMS using Arduino library (console output on the left, examples of received messages on the right).

The Helium network functions on a package-oriented data transmission system, where users are billed according to the volume of packages received by the network's gateways. Quite often, the very same data packet is captured simultaneously by multiple gateways. Thanks to reasonably low packet charges, the resulting redundancy does not place a substantial financial burden on users. However, it's important to note that the

network enforces a limitation on packet size, capping it at a maximum of 24 bytes. While this constraint might seem restrictive, it is, in fact, a suitable fit for many IIoT applications.

6. Summary and Conclusions

We have elucidated the diverse methods employed in the successful prototyping of customized IIoT sensor systems tailored to resource-constrained small and medium enterprises (SMEs). Our choice of the M5Stick hardware ecosystem, renowned for its modularity, the presence of enclosed displays, and efficient battery and charging circuits, proved instrumental. In order to assess the data reporting capabilities over extended distances, we conducted an evaluation of LoRa-E5 within the decentralized Helium network, thereby affirming the network's extensive coverage and dependable performance.

Additionally, the Arduino software ecosystem played a pivotal role by facilitating rapid integration and evaluation of diverse firmware options. This process contributed significantly to the enhancement of both reliability and usability in the prototypes. It is important to note that many of the lessons we have learned and reported hold universal relevance and can be applied to the development of similar IIoT systems with confidence.

Author Contributions: Conceptualization, N.P. and A.N.K.; methodology, N.P. and A.N.K.; software, A.N.K.; validation, N.P.; formal analysis, N.P.; investigation, T.M.; resources, N.P.; data curation, N.P.; writing—original draft preparation, A.N.K.; writing—review and editing, N.P., T.M. and A.N.K.; visualization, A.N.K.; supervision, N.P. and T.M.; project administration, N.P. and T.M.; funding acquisition, n/a. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement:

Informed Consent Statement:

Data Availability Statement: We did not upload the obtained data to a public repository because of the amount of work involved in properly describing and labelling these. However we are happy to supply the reported data on a request.

Acknowledgments: The authors gratefully acknowledge support from the Digital Innovation for Growth (DIfG) programme.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Digital Innovation for Growth Welcome Page. Available online: https://www.shu.ac.uk/business/start-your-business/digital-innovation-for-growth (accessed on 30 September 2023).
2. M5Stick Product Page. Available online: https://shop.m5stack.com/products/stick-c (accessed on 30 September 2023).
3. AdafruitIO. Available online: https://learn.adafruit.com/welcome-to-adafruit-io (accessed on 30 September 2023).
4. Elyounsi, A.; Kalashnikov, A.N. Predictive IoT Temperature Sensor. *Eng. Proc.* **2022**, *27*, 55. https://doi.org/10.3390/ecsa-9-13337.
5. Example Code for Sleep Modes on M5Stick. Available online: https://github.com/m5stack/M5StickC/blob/master/examples/Advanced/AXP192/sleep/sleep.ino (accessed on 30 September 2023).
6. Adafruit IO Arduino Library. Available online: https://github.com/adafruit/Adafruit_IO_Arduino (accessed on 30 September 2023).
7. Using POST to Add Data Points to a Feed. Available online: https://forums.adafruit.com/viewtopic.php?f=56&t=177055 (accessed on 30 September 2023).
8. WiFiManager Code Page. Available online: https://github.com/tzapu/WiFiManager (accessed on 30 September 2023).
9. EmailSender, Arduino Refernce Library. Available online: https://www.arduino.cc/reference/en/libraries/emailsender/ (accessed on 30 September 2023).
10. Demuri, A. Twilio-esp32-client. Available online: https://github.com/ademuri/twilio-esp32-client (accessed on 30 September 2023).
11. Send SMS with the ESP32 (Twilio). Available online: https://randomnerdtutorials.com/send-sms-esp32-twilio/ (accessed on 30 September 2023).
12. Helium-E5-DHT22. Available online: https://gist.github.com/NorHairil/808ec64b1d4eac3f4b6f286a9392abce (accessed on 30 September 2023).

13. Mutalib, H. Sending Data to Helium Console using Grove LoRa-E5. Available online: https://www.cytron.io/tutorial/sending-data-to-helium-console-using-grove-lora-e5 (accessed on 30 September 2023).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.