LTE-M Communication for Low-Powered IIoT: An Experimental Performance Study

Eddy Bajic
Research Centre for Automatic Control of Nancy, CNRS 7039
Campus Sciences, 54506 Vandoeuvre-lès-Nancy, France
e-mail: eddy.bajic@univ-lorraine.fr

Kais Mekki
OKKO SAS, 107 rue Saint Jean, 57510 Remering-lès-Puttelange, France
e-mail: kais.mekki@okko-france.com

Clement Rup
Research Centre for Automatic Control of Nancy, CNRS 7039
Campus Sciences, 54506 Vandoeuvre-lès-Nancy, France
e-mail: clement.rup@univ-lorraine.fr

***Abstract*—This paper presents an experimental study of the communication performance of the Long-Term Evolution for Machines (LTE-M) network for the Industrial Internet of Things (IIoT). We conducted an in-depth literature review on the communication networks used by IIoT devices, based on criteria such as transportable data size, throughput, connectivity, latency, transmission energy budget and cost. Next, we experimented with LTE-M communication for a prototype energy-constrained IoT device. Using the microcontroller and modem, we were able to establish TCP requests (send and receive), as well as HTTP requests (POST and GET). The results of our study show that LTE-M communication offers significant advantages in terms of throughput, latency and energy performance compared with other existing Low Power Wide Area Networks (LPWAN), making it particularly well suited to the needs of IIoT applications. This experimental work opens a wide range of prospects for setting up efficient, high-performance IIoT solutions in industrial environments.**

Keywords—Industrial Internet of Things; Low Power Wide Area Network; LTE-M; Experimental study.

I. Introduction

The Industrial Internet of Things (IIoT) is a constantly evolving field, with rapid growth in the number of IoT devices integrated into multi-tier communication architectures to collect and process mass industrial data. Communication networks play a crucial role in this evolution, offering solutions to meet the needs of different industrial scenarios.

Low Power Wide Area Networks (LPWAN) are key communication technologies that are chosen based on specific criteria such as the size of transportable data, throughput, connectivity, latency, the energy budget for transmissions and cost [1]. Sigfox, LoRaWAN, LTE-M, and NB-IoT are the main existing LPWAN networks. Currently in France, the Sigfox and LoRaWAN networks have demonstrated their inefficiency in terms of connectivity/coverage for large-scale deployment, particularly in rural areas [2][3]. For example, Objenious, Bouygues Telecom's IoT subsidiary, has shut down its LoRaWAN network [4]. The LTE-M network, on the other hand, is currently being deployed and marketed on a larger scale than the NB-IoT network [5].

In this context, our experimental study focuses on the communication performance of the LTE-M network. To do this, we first carried out an in-depth literature review on LPWAN for IoT communications, based on the available scientific literature. Then, we set up an experiment to test and evaluate the performance of LTE-M communication for an energy-constrained prototype IoT terminal.

In view of our study of the existing literature, we consider that this work represents an initial contribution detailing the implementation of this type of experiment. It details the sequence of AT commands to be used to operate an LTE-M modem. This sequence is not detailed anywhere on the Internet, even in the modem manufacturers' documentation. Hence, this work represents a tutorial and a reference document for future projects and opens new prospects for the implementation of effective, high-performance IoT solutions in industrial environments.

The rest of the paper is organized as follows. First, the state of the art is presented. Then, the test carried out of LTE-M network is detailed. Finally, experimental results are discussed.

II. State of the Art

In the following, we present the Industrial Internet of Things domain, its emerging technologies, and the technical aspects of its wireless communication solutions.

A. *Internet of Things*

The Internet of Things (IoT) describes the network of physical terminals, or "objects", that embed sensors, actuators, software, and other technologies to connect to servers on the Internet (Cloud servers) and exchange data with them. These terminals range from simple domestic devices to highly complex industrial tools [6]. It is thanks to several different technologies that the IoT was able to emerge. Technologies, such as advances in connectivity, with the proliferation of network protocols that have made it easier to connect sensors to the Cloud and to other 'objects', making data transfer more efficient; low-cost, low-power sensor technologies that have made the field more accessible; advances in Machine Learning and analytics; and access to vast quantities of diverse data stored in the Cloud, enabling businesses to obtain information more quickly [7].

B. *Industrial Internet of Things*

The Industrial Internet of Things (IIoT) refers to the application of IoT technology in an industrial context, especially for the instrumentation and control of sensors and terminals using Cloud technologies. To transfer all this data between objects, networks have had to be set up that are adapted not only to the objects themselves, but also to the quantity of data to be sent and the industrial environment in which the objects operate. The IIoT generally uses a two- or three-tier Device-Edge-Cloud network communication architecture to collect and process massive industrial data [8][9]. Moreover, LPWAN are increasingly used for industrial IoT applications because of their ability to provide low-cost, low-power connectivity. The characteristics of these networks are therefore very important when choosing the appropriate network for an industrial IoT application.

C. *IoT networks*

With the rapid advance of wireless communication technologies, choosing the right network for IoT devices can seem complicated. However, for an IoT system architect, it is essential to consider the specifications and constraints associated with the application when choosing the most suitable network. The choice of network for an IoT device must be based on a careful assessment of these specifications and constraints. There are four main criteria for choosing a network:
- The scope
- Data transmission rate
- Energy consumption
- The cost of deployment

Range is determined by the maximum distance data can be transmitted between devices and access points, while data transmission rate is the speed at which data can be transmitted. Power consumption is crucial for battery powered IoT devices, as it determines the autonomy of the devices. Also, the cost of deployment is an important factor to consider when deciding to use a specific network.

D. *LPWAN*

Low Power Wide Area Networks (LPWAN) characterizes, as its name suggests, all networks with a long range and low energy consumption. By reducing the data rate, data can be sent over greater distances while maintaining low power consumption [10]. When we talk about long distance for this type of network, we are talking about an order of magnitude of a few kilometers, with a device autonomy of up to several years and a bandwidth of around several hundred kbits/sec. The various LPWAN implementations differ in terms of throughput and frequency bands used. Nevertheless, these implementations can be classified into two categories: cellular LPWAN and non-cellular LPWAN.

1) *Non-cellular LPWAN:*

SigFox: French start-up SigFox was the first to make LPWAN technology popular. In the early 2010s, the company developed the SigFox network, which is based on patented Ultra Narrow Band (UNB) technology and uses unlicensed Industrial, Scientific, and Medical (ISM) frequency bands: 868 MHz in Europe, 915 MHz in North America and 433 MHz in Asia. SigFox antennas have a range of between 3 and 10 kilometers in dense areas and up to 50 kilometers in areas with few obstacles [6]. However, it should be noted that the SigFox network covers the whole of France, with 2,000 antennas deployed throughout the country. Internationally, the network covers 71 other countries in Europe and around the world, including 21 with global coverage.

Long Range Wide Area Network (LoRaWAN): LoRaWAN provides two-way data transmission at very low bit rates and very low energy consumption between objects and gateways. It uses a Chirp Spread Spectrum (CSS) modulation called LoRa. Like SigFox, this modulation is used mainly in frequency bands without an ISM license. The use of CSS modulation for the Internet of Things was patented by Cycléo, a French company that was later acquired by Semtech in 2012 [8]. On average, this modulation allows up to 5 kilometers between a gateway and an object in urban areas and 15 kilometers in rural areas.

2) *Cellular LPWAN:*

Narrowband Internet of Things (NB-IoT): NB-IoT antennas have a range of up to one kilometer in urban areas and 10 kilometers in rural areas. Within these ranges, NB-IoT provides bidirectional transmission at data rates of between 20 and 250 Kbps. This low data rate has led to a significant reduction in the energy consumption of IoT terminals, making them more suitable for battery-powered remote monitoring applications. This low data rate has also reduced the complexity and therefore the cost of deployment.

Long-Term Evolution for Machines (LTE-M): LTE-M signals have a range of up to 400 meters in urban areas and around 8 kilometers in rural areas. In the same way as for NB-IoT, the reduction in the data transmission rate has made it possible to reduce the energy consumption of the terminals deployed, thus opening the field of IoT applications to use cases that were previously reserved solely for non-cellular LPWAN.

Unlike NB-IoT, LTE-M enables data to be transmitted at lower data rates, in the order of ten bits per second, and at data rates of around 1 Mb per second. As a result, LTE-M covers a wider range of applications than NB-IoT [5] [11].

III. TESTING LTE-M NETWORK

As discussed before, LTE-M is currently being deployed and marketed on a larger scale than others IoT network. Thus, this section aims to test and evaluate the performance of this technology for an energy-constrained prototype IoT terminal.

A. *Hardware*

To carry out the implementation and various tests, we used an IoT development board designed by our project partner OKKO. This board incorporates a compact, high-performance ESP32-S2 microcontroller, specially designed for IoT applications. It was a wise choice because of its small size, advanced features, and low power consumption. It also incorporates a Sierra Electronics HL7800 modem, a wireless communication module that uses LTE-M and NB-IoT networks to provide low-speed connectivity and low power consumption, making it an ideal choice for IoT applications [12]. The modem also offers an integrated GPS module for location based IoT applications. This board offers several I2C connectors for connecting ambient sensors. We used an IoT SIM card from the multi-operator Thingsmobile [13]. Figures 1, 2 and 3 illustrate the hardware used.

B. *Problem analysis*

The main objective of our project was to develop a system capable of efficiently transmitting the data collected by various sensors to a remote server. To achieve this, we used the LTE-M modem to establish an Internet connection and send the data via TCP and HTTP requests. The card's operation had to ensure very low energy consumption. We therefore implemented a strategy involving a sequence of actions followed by a prolonged sleep mode, to save energy during periods of inactivity. This approach considerably extended the system's autonomy, while guaranteeing reliable and efficient transmission of the data collected.

The ESP32-S2 microcontroller used for this project is a customized version which presents a few differences from the standard models, particularly in terms of the number and limitation of programmable pins. This particularity meant that our codes had to be adapted to take account of this constraint. In addition, we have found that the command return terminal can sometimes have uncontrollable character inversions or deletions. We had to deal with this problem when implementing certain functionalities, such as setting up a monitoring and connection recovery system in the event of signal loss. Despite these constraints, we succeeded in overcoming these difficulties by carefully analyzing the feedback from the terminal and adapting our codes to take account of it. We thus worked to achieve two key objectives: the transmission of data via TCP requests, then the sending of HTTP requests to an external Cloud server.

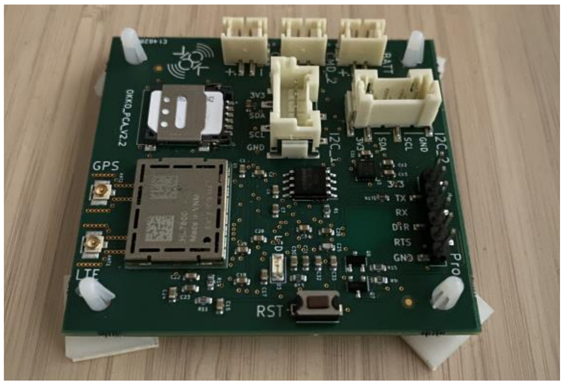


Figure 1. The OKKO board: ESP32-S2 microcontroller + HL7800 LTE-M modem + I2C connectors.

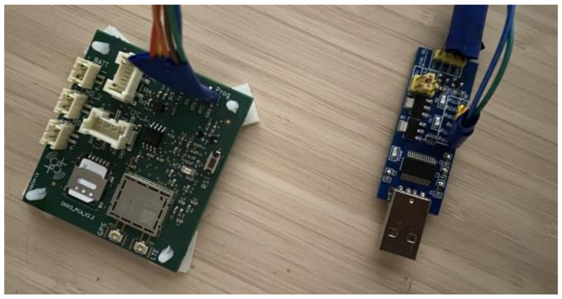


Figure 2. Assembly for uploading Arduino code to the OKKO board.

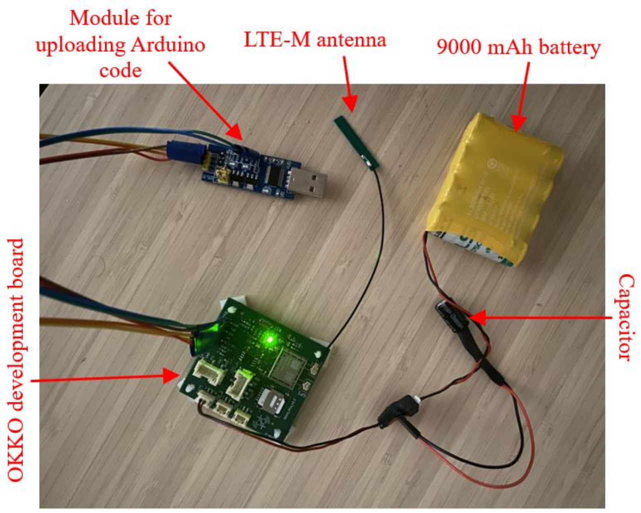


Figure 3. Complete assembly for battery operation.

C. *Study of the manufacturer's documentation*

To understand how the ESP32-S2 microcontroller works, we studied its documentation and carried out tests by modifying the values of the pins on the electronic board. This enabled us to control the flashing of the LED on the board according to our needs. This step was crucial for the rest of the project, as it enabled us to create a reliable visual feedback method for future communication tests with the LTE-M modem. We also explored the different libraries available for communicating with this HL7800 modem [12], testing several of them to determine the one best suited to our needs.

We then consulted the modem's documentation in full. The ESP32-S2 microcontroller can communicate directly with the HL7800 modem using the AT commands listed in the documentation. You can find explanations of the possible commands and even examples of instruction strings to perform certain tasks. As our focus is on TCP and HTTP requests, we have concentrated on these areas in the documentation.

Each of these commands can be used in different ways, and most functions have three versions:
- A read mode specified by adding a ? to the command name, returns the status of parameters related to the command.

- A write mode (adding an = to the command name) which allows these parameters to be modified and the command to be executed.
- A test mode that returns a list of possible values for each parameter.

We have seen that the LTE-M modem has two main modes and, depending on what is sent to it, one or the other will be activated:
- The main mode, known as command mode, is the one the modem should be in when you want to send it an AT command.
- The given mode is the one the modem is in when, for example, it is waiting to receive data to transmit. This can be seen when connecting to a TCP server, where the mode is activated to send data to the server.

Once these tasks had been completed, we concluded that it was necessary to have routines that could be executed after the microcontroller and modem had been initialized. The SIM card in the LTE-M modem must first be setup to establish a connection with the cellular network. AT commands are used for this initialization step. These commands allow the modem to be configured and controlled via a serial command interface. When the SIM card is initialized, it is also possible to configure the cellular network parameters such as the Access Point Name (APN), username and password, which are required to connect to the cellular network. Once the SIM card has been initialized, the modem can then be configured to establish a TCP or HTTP connection with a remote server using the relevant AT commands. To test the capabilities of our system, we set up and programmed two servers in Python, as shown in Figure 4: one capable of receiving HTTP requests and the other TCP requests. This set-up enabled us to test the two modes of communication and validate that they worked properly. According to the modem's documentation, sending HTTP requires an extra step than sending TCP, because the fields needed for the HTTP request must be placed. We therefore started with the TCP request using the following AT commands: AT-KTCPCFG, AT-KTCPSND, ATKTCPRCV and AT-KTCPCLOSE. For HTTP requests, we used the following AT commands: AT-KHTTPCFG, AT-KHTTPGET, AT-KHTTPPOST, ATKHTTPHEADER and AT-KHTTPCLOSE.

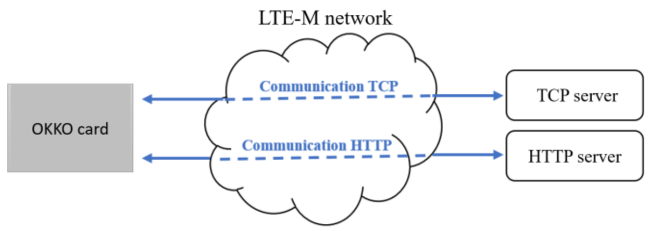
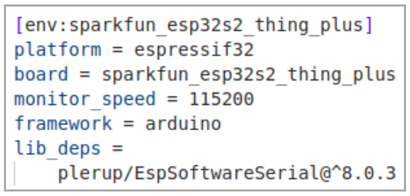


Figure 4. Communication between the card and the servers via the LTE-M network.

D. Implementation

To implement our project, we used the Visual Studio Code editor with the Arduino PlatformIO extension to compile and send our code to our microcontroller, which enabled us to work efficiently and save time in the development process. As with any PlatformIO project, we defined the specifications for the board and the Framework in a "platform.ini" file, as shown in Figure 5. We set the data transmission speed from the serial port to the monitor, which helps display the modem responses, to 115200 baud. The choice of this speed depends on the microcontroller used, in our case an ESP32-S2 derivative. The EspSoftwareSerial library is included in PlatformIO and provides the various instructions for output to the monitor. It was necessary to import it when working on the Visual Studio Code environment.

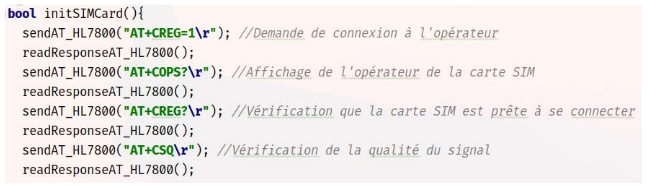


```
[env:sparkfun_esp32s2_thing_plus]
platform = espressif32
board = sparkfun_esp32s2_thing_plus
monitor_speed = 115200
framework = arduino
lib_deps =
    plerup/EspSoftwareSerial@^8.0.3
```

Figure 5. Extract from the platform.ini file.

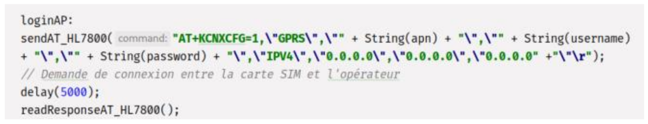
To initialize the SIM card, several AT commands are required. Firstly, the AT+CREG= command is used to start the operator search and connection to the chosen operator. Two checks are then carried out:
- AT+COPS? command ensures that the modem has chosen the right operator.
- AT+CREG? command shows the status of the operator search and ensures that it has been assigned to the correct operator.

The quality of the data rate (in dB) is then read using AT+CSQ to check that the antenna is operating correctly. This sequence of commands is illustrated in Figure 6. Then, as shown in Figure 7, the AT+KCNXCFG= command tells the modem which parameters to use to connect to the operator's network.



```
bool initSIMCard(){
  sendAT_HL7800("AT+CREG=1\r"); //Demande de connexion à l'opérateur
  readResponseAT_HL7800();
  sendAT_HL7800("AT+COPS?\r"); //Affichage de l'opérateur de la carte SIM
  readResponseAT_HL7800();
  sendAT_HL7800("AT+CREG?\r"); //Vérification que la carte SIM est prête à se connecter
  readResponseAT_HL7800();
  sendAT_HL7800("AT+CSQ\r"); //Vérification de la qualité du signal
  readResponseAT_HL7800();
```

Figure 6. initSIM function - part 1.

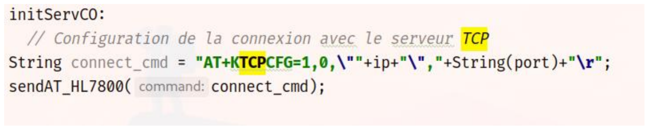


```
loginAP:
sendAT_HL7800( command: "AT+KCNXCFG=1,\"GPRS\",\"" + String(apn) + "\",\"" + String(username)
+ "\",\"" + String(password) + "\",\"IPV4\",\"0.0.0.0\",\"0.0.0.0\",\"0.0.0.0" +"\"\r");
// Demande de connexion entre la carte SIM et l'opérateur
delay(5000);
readResponseAT_HL7800();
```

Figure 7. initSIM function - part 2.

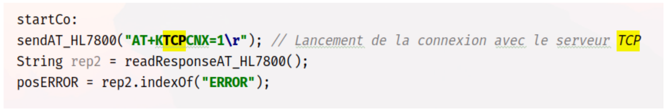
1) Sending a TCP message:

As shown in Figure 8, the AT+KTCPCFG= command allows the modem to specify the IP address and port of the TCP server to connect to. Then, in Figure 9, the AT+KTCPCNX= command enables the LTE-M modem to launch the TCP connection with the server.



```
initServCO:
  // Configuration de la connexion avec le serveur TCP
String connect_cmd = "AT+KTCPCFG=1,0,\""+ip+"\","+String(port)+"\r";
sendAT_HL7800( command: connect_cmd);
```

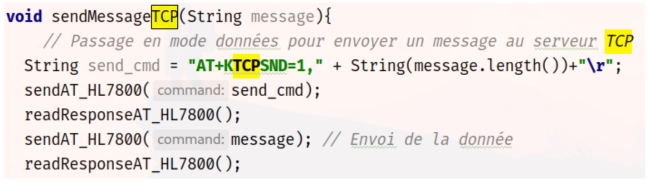
Figure 8.   initSIM function - part 3.



```
startCo:
sendAT_HL7800("AT+KTCPCNX=1\r"); // Lancement de la connexion avec le serveur TCP
String rep2 = readResponseAT_HL7800();
posERROR = rep2.indexOf("ERROR");
```

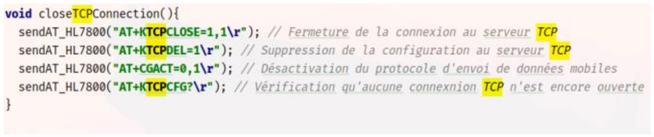
Figure 9.   Extract from the connectTCP function.

The AT+KTCPSND= command switches the modem to data mode and transmits everything it receives until it has reached the maximum size specified in the parameter, or until it returns to command mode. Finally, the TCP connection is closed. These procedures are illustrated in Figures 10 and 11, respectively. This sequence of AT commands allows the connection to the TCP server to be fully executed.



```
void sendMessageTCP(String message){
   // Passage en mode données pour envoyer un message au serveur TCP
  String send_cmd = "AT+KTCPSND=1," + String(message.length())+"\r";
  sendAT_HL7800( command: send_cmd);
  readResponseAT_HL7800();
  sendAT_HL7800( command: message); // Envoi de la donnée
  readResponseAT_HL7800();
```

Figure 10. Extract from the sendMessageTCP function.

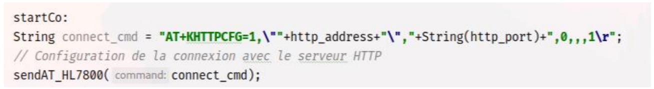


```
void closeTCPConnection(){
  sendAT_HL7800("AT+KTCPCLOSE=1,1\r"); // Fermeture de la connexion au serveur TCP
  sendAT_HL7800("AT+KTCPDEL=1\r"); // Suppression de la configuration au serveur TCP
  sendAT_HL7800("AT+CGACT=0,1\r"); // Désactivation du protocole d'envoi de données mobiles
  sendAT_HL7800("AT+KTCPCFG?\r"); // Vérification qu'aucune connexnion TCP n'est encore ouverte
}
```

Figure 11. Extract from the closeTCPConnection function.

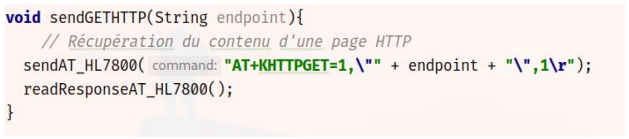
2) HTTP GET and HTTP POST exchanges between the card and the server:

As shown in Figure 12, the AT+KHTTPCFG= command allows the modem to specify the IP address and port of the HTTP server to connect to. Then, in Figure 13, the AT+KHTTPGET= command is used to make a GET request to the specified address of the HTTP server to which the modem is connected. This retrieves the data from the server.



```
startCo:
String connect_cmd = "AT+KHTTPCFG=1,\""+http_address+"\","+String(http_port)+",0,,,1\r";
// Configuration de la connexion avec le serveur HTTP
sendAT_HL7800( command: connect_cmd);
```

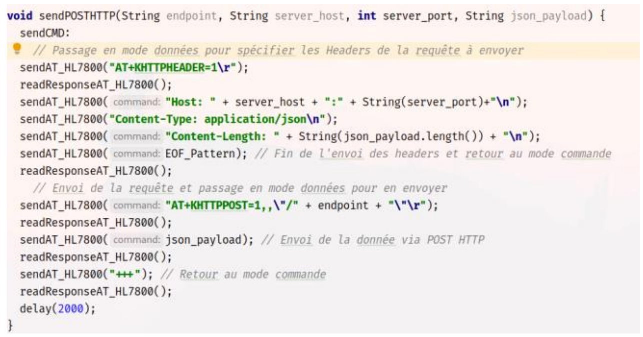
Figure 12. Extract from the connectHTTP function.



```
void sendGETHTTP(String endpoint){
   // Récupération du contenu d'une page HTTP
  sendAT_HL7800( command: "AT+KHTTPGET=1,\"" + endpoint + "\",1\r");
  readResponseAT_HL7800();
}
```

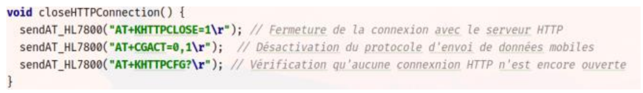
Figure 13. Extract from the sendGETHTTP function.

The AT+KHTTPHEADER= command is used to specify the headers of the HTTP POST request to be sent. For example, we specify the type of data we are sending and its size (in our case the data type is JSON). Then we use AT+KHTTPPOST= to send the HTTP POST message to the specified server address. As a result, the modem switches to data mode. We give it the data to send to the server. The "+++" then allows the modem to return to command mode, as shown in Figure 14. Finally, we close the HTTP connection (see Figure 15). This sequence of AT commands enables the HTTP GET and POST connection to the server to be fully executed.



```
void sendPOSTHTTP(String endpoint, String server_host, int server_port, String json_payload) {
  sendCMD:
   // Passage en mode données pour spécifier les Headers de la requête à envoyer
  sendAT_HL7800("AT+KHTTPHEADER=1\r");
  readResponseAT_HL7800();
  sendAT_HL7800( command: "Host: " + server_host + ":" + String(server_port)+"\n");
  sendAT_HL7800("Content-Type: application/json\n");
  sendAT_HL7800( command: "Content-Length: " + String(json_payload.length()) + "\n");
  sendAT_HL7800( command: EOF_Pattern); // Fin de l'envoi des headers et retour au mode commande
  readResponseAT_HL7800();
   // Envoi de la requête et passage en mode données pour en envoyer
  sendAT_HL7800( command: "AT+KHTTPPOST=1,,\"/" + endpoint + "\"\r");
  readResponseAT_HL7800();
  sendAT_HL7800( command: json_payload); // Envoi de la donnée via POST HTTP
  readResponseAT_HL7800();
  sendAT_HL7800("+++"); // Retour au mode commande
  readResponseAT_HL7800();
  delay(2000);
}
```

Figure 14. Extract from the sendPOSTHTTP function.



```
void closeHTTPConnection() {
  sendAT_HL7800("AT+KHTTPCLOSE=1\r"); // Fermeture de la connexion avec le serveur HTTP
  sendAT_HL7800("AT+CGACT=0,1\r");  // Désactivation du protocole d'envoi de données mobiles
  sendAT_HL7800("AT+KHTTPCFG?\r"); // Vérification qu'aucune connexnion HTTP n'est encore ouverte
}
```

Figure 15. Extract from the closeHTTPConnection function.

E.   Study of energy consumption

Once the program has been uploaded to the OKKO card, the microcontroller initializes the modem. The modem then starts the TCP and HTTP communication described in the previous section. Once transmission is complete, the card goes into Deep Sleep mode until the next transmission request. As discussed earlier in this article, this mode saves battery power during periods of card inactivity.

The energy consumption of this behavior was measured using the OTII tool [14]. Table 1 shows the power consumption of the different execution phases of the card. As this table shows, the power consumption of the OKKO card in TCP mode is lower than in HTTP mode because the latter has more protocol load (header and connection establishment) to establish the connection between the modem and the server. We therefore recommend using the TCP protocol to send data from the card to the server via the LTE-M network. The table also shows the card's consumption in sleep mode. This consumption is equal to 32 micro-amperes, which guarantees long battery life.

Calculations and tests carried out at OKKO indicate a battery life of 5 years for a card that sends a single TCP message every 24 hours (with a 9000 mAh battery).

TABLE I. ENERGY CONSUMPTION OF THE OKKO CARD.

	Modem Initialization		**Communication**		**Sleeping mode**
	Average duration	**Average consumption**	**Average duration**	**Average consumption**	**Average consumption**
TCP	15 s	88 mA	22 s	96.2 mA	32 uA
HTTP	27 s	91 mA	106 s	128 mA	

IV. CONCLUSION

In this paper, we studied the performance of the LTE-M network for the Industrial Internet of Things, based on an in-depth literature review and a practical implementation on a prototype IoT terminal with energy constraints from the OKKO company. Thanks to the ESP32-S2 microcontroller and the LTE-M modem, we have succeeded in establishing TCP communications as well as HTTP communications (POST and GET), which contributes to the implementation of efficient, high-performance IoT solutions in industrial environments. The results show that the LTE-M network is perfectly suited to applications requiring low-speed transmission over a wide deployment area. These results open up a wide range of possibilities for the implementation of autonomous IoT systems in industry.

ACKNOWLEDGMENT

This paper is an exploratory study on IoT as part of the I2RM (21-SIOM-0007-02) project funded by the French National Research Agency.

REFERENCES

[1] E. U. Ogbodo, A. M. Abu-Mahfouz, and A. M. Kurien, “A Survey on 5G and LPWAN-IoT for Improved Smart Cities and Remote Area Applications: From the Aspect of Architecture and Security,” Sensors, vol. 22, pp. 32132-32149, August 2022, doi:10.3390/s22166313.

[2] P. Levchenko, D. Bankov, E. Khorov, and A. Lyakhov, “Performance Comparison of NB-Fi, Sigfox, and LoRaWAN,” Sensors, vol. 22, pp. 1-21, December 2022, doi:10.3390/s22249633.

[3] M. A. M. Almuhaya, W. A. Jabbar, N. Sulaiman, and S. Abdulmalek, “A Survey on LoRaWAN Technology: Recent Trends, Opportunities, Simulation Tools and Future Directions,” Electronics, vol. 11, pp. 1-32, January 2022, doi:10.3390/electronics11010164.

[4] Bouygues Telecom LoRaWAN network shutdown: https://objenious.com/blog/technologie/arret-du-reseau¬lorawan-de-bouygues-telecom/. [retrieved: September, 2023].

[5] S. He, K. Shi, C. Liu, and B. Guo, “Collaborative Sensing in Internet of Things: A Comprehensive Survey,” IEEE Communications Surveys & Tutorials, vol. 24, pp. 1435-1474, June 2022, doi: 10.1109/COMST.2022.3187138.

[6] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, “A comparative study of LPWAN technologies for large-scale IoT deployment,” ICT Express, vol. 5, pp. 1-7, March 2019, doi:10.1016/j.icte.2017.12.005.

[7] J. Wang, M. K. Lim, C. Wang, and M. L. Tseng, “The evolution of the Internet of Things (IoT) over the past 20 years,” Computers & Industrial Engineering, vol. 155, pp. 1395-1413, May 2021, doi:10.1016/j.cie.2021.107174.

[8] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, “Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT,” The IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), March 2018, pp. 197-202, ISBN: 978-1-5386-3227-7.

[9] K. Nahrstedt, H. Li, P. Nguyen, S. Chang, and L. H. Vu, “Internet of Mobile Things: Mobility-Driven Challenges, Designs and Implementations,” The First IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI), April 2016, pp. 25-36, ISBN: 978-1-4673-9948-7.

[10] L. Zemko and P. Čičák, “IoT and LPWAN Networks: Increasing Efficiency by Communication Planning,” The 45th IEEE International Conference on Telecommunications and Signal Processing (TSP), July 2022, pp. 78-85, ISBN: 978-1-6654-6948-7.

[11] C. Westerkamp, A. Grunwald, and M. Schaarschmidt, “LoRaWAN, NB IoT and other radio networks for agricultural applications,” The 26th IEEE ITG-Symposium Mobile Communication - Technologies and Applications, May 2022, pp. 61-67, ISBN:978-3-8007-5873-9.

[12] AT Commands Interface Guide for AirPrime HL78xx: https://source.sierrawireless.com/resources/airprime/software/hl78xx_at_commands_interface_guide/#sthash.2HvUJG26.dpbs. [retrieved: September, 2023].

[13] Thingsmobile, www.thingsmobile.com. [retrieved: September, 2023].

[14] Otii by QOITECH, www.qoitech.com. [retrieved: September, 2023].