

# Tiny oneM2M C Language API

Rafael Pereira<sup>a</sup>

<sup>a</sup>Computer Science and Communications Research Centre, School of Technology and Management, Polytechnic of Leiria, 2411-901 Leiria, Portugal

## ARTICLE INFO

### Keywords:

quadrupole exciton  
polariton  
WGM  
BEC

## ABSTRACT

This template helps you to create a properly formatted L<sup>A</sup>T<sub>E</sub>X manuscript.

\begin{abstract} ... \end{abstract} and \begin{keyword} ... \end{keyword} which contain the abstract and keywords respectively.

Each keyword shall be separated by a \sep command.

## 1. Introduction

The Internet of Things (IoT) has experienced tremendous growth in recent years, with an ever-increasing number of devices being connected and integrated into various applications, such as smart cities, industrial automation, and environmental monitoring. One key aspect enabling this widespread adoption is Machine Type Communication (MTC), which facilitates seamless communication between machines and devices without human intervention. MTC allows for efficient data exchange among heterogeneous devices, leading to better coordination and improved overall system performance.

In this context, interoperability plays a vital role in ensuring the seamless functioning of IoT ecosystems. As the number and diversity of connected devices continue to grow, so does the need for a robust mechanism to manage communication between devices employing different communication protocols, data formats, and standards. MTC gateways serve as an essential bridge between these devices and the network infrastructure, enabling effective data routing and ensuring smooth communication among heterogeneous devices.

Existing MTC gateway implementations predominantly rely on high-level programming languages, which offer the advantages of rapid development cycles and ease of use. However, these high-level languages often come with trade-offs, such as limited control over hardware resources, increased memory overhead, and suboptimal performance. These limitations can pose significant challenges when deploying large-scale IoT systems, where efficient resource utilization, interoperability, and high-throughput data transmission are essential.

In this paper, we present a novel MTC gateway implementation using a low-level programming language to address the limitations of high-level language-based implementations. Our approach is designed to deliver superior performance, reduced memory overhead, and increased control over hardware resources, while maintaining compatibility with a wide range of MTC protocols. By focusing on key challenges associated with MTC gateways, such as protocol interoperability, scalability, and latency, our implementation

aims to provide a robust and efficient solution for the evolving IoT landscape.


**Paper structure** The remainder of this paper is organized as follows: Section 2 provides a comprehensive review of the related work in MTC gateway implementations, discussing various programming languages and architectural choices, as well as the importance of interoperability in IoT systems. Section 3 delves into the specific challenges associated with MTC protocol integration and outlines the design principles of our novel low-level language-based gateway. Section 4 presents the experimental setup and performance evaluation of our proposed solution. Finally, Section 5 concludes the paper and highlights potential future research directions.

## 2. Related work

Machine-Type Communication (MTC) has become an increasingly critical aspect of modern communication systems, driving the transformation of various applications in the ever-evolving Internet of Things (IoT) ecosystem. MTC enables the seamless interaction between a vast array of devices, ranging from wearables and smartphones to industrial equipment and smart home appliances, facilitating the data exchange required for intelligent and automated decision-making processes. With the exponential growth of IoT devices and their stringent requirements for low latency, high reliability, and energy efficiency, the implementation of MTC in low-level languages has emerged as a promising solution to address these challenges.

In this section, we will provide an overview of the current state-of-the-art in MTC gateway implementations, highlighting the advantages and disadvantages of various programming languages and architectural choices. Additionally, we will explore the specific challenges associated with MTC protocol integration and discuss how our novel approach addresses these issues to provide a comprehensive solution for the next generation of IoT deployments.

Rubi et al. [1] propose an IoMT platform with a cloud-based electronic health system that collects data from various sources and relays it through gateways or fog servers. The platform enables data management, visualization of electronic records, and knowledge extraction through big data processing, machine learning, and online analytics processing. Additionally, it offers data sharing services for third-party applications, improving healthcare services and inter-

 rafael.m.pereira@ipleiria.pt (R. Pereira)

ORCID(s): 0000-0001-8313-7253 (R. Pereira)



[https://www.linkedin.com/profile/view?id=](https://www.linkedin.com/profile/view?id=rafaelmendespereira)

'rafaelmendespereira' (R. Pereira)

operability.

The authors in [2] conducted IoTDM service testing for smart city management, focusing on the ecological situation in Saint-Petersburg's central district, using an SDN network infrastructure. They developed a hierarchical model and IoT traffic generator in compliance with oneM2M specifications and implemented HTTP, CoAP, and MQTT protocols. Their findings suggested that HTTP is the most suitable protocol for registering IoT devices to the IoTDM service, while MQTT is optimal for transmitting sensor data. The authors concluded that the SDN approach significantly reduces the RTT parameter, demonstrating its potential in managing IoT traffic in smart city environments.

Soumya et al. [3] explore Fog Computing as a consumer-centric IoT service deployment platform, particularly for connected vehicles and intelligent transportation systems requiring low latency, high mobility, real-time data analytics, and wide geographic coverage. They present an IoT architecture for connected vehicles, integrating Fog Computing platforms into oneM2M standard architecture. The architecture comprises virtual sensing zones, Access Points (RSUs), M2M gateways, and cloud systems. Fog Computing enables consumer-centric services such as M2M data analytics with semantics, discovery of IoT services, and management of connected vehicles. The paper highlights the advantages of Fog Computing, including reduced latency, improved QoS, real-time data analysis, and actuation for superior user experience and consumer-centric IoT products.

This paper [4] presents a cluster-based congestion-mitigating access scheme (CCAS) that addresses the issue of collision and access efficiency in machine-to-machine (M2M) communications. The authors propose a modified spectral clustering algorithm to form clusters based on the location and service requirements of machine-type communication devices (MTCDs), ensuring that devices with similar requirements are grouped together. They also develop an MTCG (machine-type communication gateway) selection algorithm that considers the delay requirements of MTCDs and selects an MTCG with the appropriate forwarding threshold. The proposed CCAS divides the data transmission process into an MTCD-MTCG stage and an MTCG-BS (base station) stage to reduce collision probability and increase the number of successfully received packets, without causing additional average access delays for MTCDs. The paper includes a theoretical analysis of the MTCG aggregation and forwarding process, as well as performance evaluations through simulations.

### 3. Architecture

### 4. Proposed Solution

### 5. Tests

### 6. Test Bed

Machine1	
Hardware	
Model Name	MacBook Pro
Model Identifier	MacBookPro18,1
Chip	Apple M1 Pro
Total Number of Cores	10 (8 performance and 2 efficiency)
RAM Memory	16 GB
Operating System	
Product Name	macOS
Product Version	12.6
Build Version	21G115
Compiler	
GCC Version	Apple clang version 14.0.0 (clang-1400.0.29.102)
Libraries	
SQLite3 Version	3.41.1
cJSON Version	1.7.15

In order to thoroughly evaluate the effectiveness and applicability of our proposed solution, it is crucial to conduct a comprehensive set of test cases. First, we aim to compare the performance of three different scenarios in terms of query time: using the original codebase, the codebase with "expiration time" field as a condition in database queries, and the codebase with "expiration time" field as a condition in database queries when it is indexed. This analysis will provide valuable insights into how the handling of the "expiration time" field impacts system performance and help identify potential optimizations. Second, a comparative analysis of latency for 500 AE resource CRUD operations using the proposed solution and OpenMTC will enable us to gauge the relative efficiency of our approach, highlighting its strengths and areas for improvement. Third, performing a cross-platform evaluation through installation and compilation on Ubuntu and macOS systems ensures that our solution is versatile and can be seamlessly integrated into different operating environments. Finally, testing the compatibility of our solution with minimal hardware, such as the ESP32 case study, will help verify its suitability for resource-constrained scenarios, broadening its potential applications and enabling its adoption in a wider range of use cases.

- 6.1. Assessing the Impact of Varying Expiration Time Field Length on Performance**
- 6.2. Comparative Analysis of Latency for 500 AE CRUD Operations Using the Proposed Solution and OpenMTC**
- 6.3. Cross-Platform Evaluation: Installation and Compilation on Ubuntu and macOS Systems**
- 6.4. Minimal Hardware Compatibility Testing: ESP32 Case Study**

Avaliação do impacto resultante do comprimento do objetivo do campo expiration time

Quanto tempo demora a GET/POST/PUT/DELETE 500 AEs usando a nossa solução e a solução disponível do OpenMTC na mesma máquina (na máquina do MEI-CM) na rede - Saber o tempo de cada um dos 500 e calcular, mínimo, máximo, média, desvio padrão

Instalação e Compilação em máquinas Ubuntu e MacOS (verificar se corre em outros SOs)

? Fazer testes com Hardware mínimo - por exemplo ESP32 ?

## **CRedit authorship contribution statement**

: Conceptualization of this study, Methodology, Software.

## **References**

- [1] Jesús N.S. Rubí and Paulo R.L. Gondim. Iomt platform for pervasive healthcare data aggregation, processing, and sharing based on onem2m and openehr. *Sensors* 2019, Vol. 19, Page 4283, 19:4283, 10 2019.
- [2] Artem Volkov, Abdukodir Khakimov, Ammar Muthanna, Ruslan Kirichek, Andrei Vadyko, and Andrey Koucheryavy. Interaction of the iot traffic generated by a smart city segment with sdn core network. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10372 LNCS:115–126, 2017.
- [3] Soumya Kanti Datta, Christian Bonnet, and Jerome Haerri. Fog computing architecture to enable consumer centric internet of things services. *Proceedings of the International Symposium on Consumer Electronics, ISCE*, 2015-August, 8 2015.
- [4] Liang Liang, Lu Xu, Bin Cao, and Yunjian Jia. A cluster-based congestion-mitigating access scheme for massive m2m communications in internet of things. *IEEE Internet of Things Journal*, 5:2200–2211, 6 2018.



My name is Rafael Pereira, I'm 22 years old, and I'm an MSc Computer Engineering Student and Researcher at Polytechnic of Leiria. I'm extremely curious about the technology world, ambitious for knowledge in this area, I'm dedicated to doing my work. The programming thing always got me, and every day it grows. Today I'm looking for interesting projects, that can make me think and learn every day.