The urdflib Library for MicroPython: Manipulating RDF on Constrained Devices

Mohsen Hadavi[1,*], Maxime Lefrançois[2]

[1]Mines Saint-Étienne, 42023 Saint-Étienne France
[2]Mines Saint-Étienne, Univ Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158 LIMOS, F - 42023 Saint-Étienne France

Abstract

This demonstration paper introduces the urdflib library for MicroPython, which facilitates the development of RDF manipulation programs for embedded devices that run MicroPython, and additionally ensures the API is compatible with the widely used RDFLib Python library. The library is openly available on GitHub, a demo is presented, and the performances of the library are evaluated on the Unix and the ESP32 ports of MicroPython. urdflib performs better than RDFLib on the Unix port.

Keywords

Semantic Web of Things, MicroPython, Programming the Semantic Web, Internet of Things

1. Introduction

In recent years, the world of embedded systems and microcontroller programming has witnessed a significant transformation, with an ever-increasing demand for efficient and versatile solutions. As electronic devices continue to permeate every aspect of modern life, the need for simplified and accessible programming tools becomes more pronounced. MicroPython [1], an implementation of Python tailored for microcontrollers, has emerged as a game-changer in this domain, bridging the gap between traditional embedded systems programming and the world of high-level languages.

It has been proven that Python (and its variant for microcontrollers MicroPython) is easier to learn and use than C [2]. Recent efforts to define compressed protocols and syntaxes for the Web contribute to the vision of a *Semantic Web of Things*, where CoAP is used instead of HTTP, and compressed RDF syntaxes such as CBOR-LD are employed. New compression mechanisms such as SCHC [3] help bring IPv6, UDP, CoAP, and CBOR to LPWAN (Low-Power Wide Area Network) and LoWPAN (Low-Power Wireless Personal Area Networks) networks.

Typically, embedded programming is done in C/C++, potentially using the Arduino framework. Sord[1] and Serd[2] are lightweight C libraries to work with RDF data and store RDF

ISWC 2023 Posters and Demos: 22nd International Semantic Web Conference, November 6–10, 2023, Athens, Greece
*Corresponding author.
moh3nhadavi@gmail.com (M. Hadavi); maxime.lefrancois@emse.fr (M. Lefrançois)
0000-0001-9814-8991 (M. Lefrançois)

[1]https://github.com/drobilla/sord
[2]https://github.com/drobilla/serd

statements in memory, and are a perfect fit for high-performance or resource-limited applications.

In this demonstration paper, we contribute to the vision of the Semantic Web of Things by providing a simple and accessible, yet lightweight library for RDF processing, based on MicroPython. Our library, named urdflib, uses Sord and Serd as the backend, and provides an API that conforms to the widely used RDFLib Python library [4], in such a way that simple RDFLib programs could be adapted to constrained devices with a simple change of the module import declaration:

```
import urdflib as rdflib
```

In the following sections, we will thoroughly examine the critical components of our research. Section 2 will extensively cover the design and implementation decisions made for urdflib, along with an availability statement. Section 3 will provide an initial assessment of the library, emphasizing its compatibility with Unix and ESP32 ports of MicroPython. Moving on to Section 4, we will investigate other relevant research in this domain and underscore the unique attributes of urdflib when compared to these works. Lastly, Section 5 will be dedicated to summarizing our findings, drawing conclusions, and discussing the resulting implications.

2. Design, Implementation, Availability, Demonstration

Compatibility with RDFLib. urdflib is intended to provide users with a significant interpretation of RDFLib, ensuring both ease of use and compatibility with existing code. This goal was achieved by running tests using the RDFLib repository, thus allowing users to stick to the RDFLib APIs. However, it is necessary to acknowledge that the underlying implementation in the urdflib module is written in the C language. As of today, urdflib implements a coherent subset of the RDFLib API, leaving out namespaces, syntax parsers and serializers, and other storage and querying plugins.

Integration with the MicroPython Firmware. Three methods for creating a module in MicroPython are available: Core module, external module, and port module. The implementation of urdflib utilizes the Sord, Serd, and Zix C libraries, which required modifications in their code. As a result, these libraries were merged into a single repository and added to MicroPython as an external module. Consequently, urdflib now operates as a core module, utilizing Serd and Sord to establish an API that facilitates RDF manipulation on resource-constrained devices. Two MicroPython ports have been tested successfully: the Unix and the ESP32 ports.

Decoupling with the C backend library. The dependency of urdflib to Serd is only scoped to the separate middleware folder. To migrate to a new RDF C library backend, one would only need to change the files in this folder.

Resource availability statement. The source code for urdflib is available from GitHub[3] under the open MIT license. urdflib is registered on Zenodo, with a permanent DOI and canonical

[3] https://github.com/moh3nhadavi/micropython-urdflib, or https://gitlab.com/coswot/micropython-urdflib

citation.[4] Additionally, comprehensive documentation and practical examples illustrating how to create MicroPython modules like urdflib can be found on GitHub[5].

Demonstration A demonstration of the library running on an ESP32 via REPL (Read, Evaluate, Print, Loop), is available online.[6]

3. Performance Evaluation

When the urdflib module is incorporated into MicroPython, it only adds 27 KB to the firmware size for the ESP32 port, which represents $\tilde{1}.5$ % of the 1.5 MB original MicroPython firmware for ESP32. The approximate memory overhead during firmware writing is 2 KB.

Table 1 summarizes the results of an experiment conducted to assess the process of adding triples on three different platforms, having different resource capabilities. The time taken and memory allocations for operations were measured. The platforms used in the experiment were: (**py-rdflib**) A MacBook Air with RDFLib package installed, a 1,6GHz Dual-Core Intel Core i5 processor, with 16GB 2133 MHz LPDDR3 memory; (**mpy-unix-urdflib**) The same MacBook Air running the MicroPython Unix port with the urdflib module; and (**mpy-esp32-urdflib**) An ESP32 device with a memory capability of 128KB, running MicroPython with the urdflib module. As illustrated in Table 1, urdflib demonstrates significantly higher speed and lower memory usage than RDFLib. Step-by-step instructions to compile the code and reproduce the experiments are available online.[7]

Table 1
Performance Evaluation

Platform	# triples	Initial Time (us)	Adding Time (us)	Memory Allocation (B)
py-rdflib	100	173302	6544	289640
py-rdflib	1000	180661	65505	2319151
py-rdflib	10000	180499	449035	22944084
mpy-unix-urdflib	100	17	1222	60160
mpy-unix-urdflib	1000	20	15857	578560
mpy-unix-urdflib	10000	32	115643	1909248
mpy-esp32-urdflib	50	12276	90363	18880
mpy-esp32-urdflib	100	10924	179512	34880
mpy-esp32-urdflib	180	8001	330541	60480

4. Related Work

This section lists some related work from the literature on RDF for the IoT (Internet of Things). RDF4Led [5] is an RDF engine for edge devices. It is compared against Virtuoso and Jena TDB

[4] https://zenodo.org/record/8342624
[5] https://github.com/moh3nhadavi/micropython-usermod/tree/update
[6] https://ci.mines-stetienne.fr/urdflib/demo
[7] https://github.com/moh3nhadavi/micropython-urdflib-iswc2023

on three types of hardware with 256-512 MB of RAM. The constrained device urdflib targets are microcontrollers such as the ESP32 with just a few MB of RAM. Cowl [6] is a lightweight implementation of OWL 2 that aims to target devices with severe processing and memory limitations. A similar work as ours could be led to use it as the backed for a lightweight version of a Python implementation of OWL, such as Owlready2[8]. LiRoT [7] use Serd and Sord to propose a lightweight incremental reasoner that can be embedded in constrained objects, so that reasoning on them in a fog architecture becomes possible. It demonstrates lower reasoning time for small numbers of triples, which makes it suitable for the Semantic Web of Things. Regarding the lightweight RDF syntaxes, [8] proposes to use a CBOR equivalent of JSON-LD as a lightweight syntax for RDF. RDF/CBOR is another proposed syntax for RDF based on CBOR[9]. [9] compares HDT and CBOR for exchanging RDF.

5. Conclusion and Future Work

The urdflib module facilitates the development of RDF manipulation programs for embedded devices that run MicroPython and additionally ensures the API is compatible with the widely used RDFLib Python library. urdflib may have an impact in contributing to the adoption of Semantic Web technologies by easing its use for IoT learners and hackers.

While this library represents a significant advancement in working with RDF on constrained devices, there are further steps to be taken in the future. These include: (i) use compact data structure for storing IRIs, (ii) use native datatypes for storing literals instead of their lexical form, (iii) implement serializers and parsers for compressed syntaxes that minimize memory usage because most RDF syntaxes are verbose and not suitable for IoT communication scenarios, and (iv) provide API for lightweight reasoner LiRoT or other features due to the intended goals.

Acknowledgments

This work is supported by grant ANR-19-CE23-0012 from Agence Nationale de la Recherche, France, for project CoSWoT [10].

References

[1] D. George, Micropython-python for microcontrollers, 2014.

[2] H. Fangohr, A comparison of c, matlab, and python as teaching languages in engineering, in: Computational Science-ICCS 2004: 4th International Conference, Kraków, Poland, June 6-9, 2004, Proceedings, Part IV 4, Springer, 2004, pp. 1210–1217.

[3] A. Minaburo, L. Toutain, C. Gomez, D. Barthel, J.-C. Zúñiga, Rfc 8724: Schc: Generic framework for static context header compression and fragmentation, 2020.

[4] D. Krech, G. A. Grimnes, G. Higgins, J. Hees, I. Aucamp, N. Lindström, N. Arndt, A. Sommer, E. Chuc, I. Herman, A. Nelson, J. McCusker, T. Gillespie, T. Kluyver, F. Ludwig, P.-A.

[8] https://pypi.org/project/Owlready2/

[9] https://openengiadina.codeberg.page/rdf-cbor/

[10] https://coswot.gitlab.io/

Champin, M. Watts, U. Holzer, E. Summers, W. Morriss, D. Winston, D. Perttula, F. Kovacevic, R. Chateauneu, H. Solbrig, B. Cogrel, V. Stuart, Rdflib/rdflib: Rdflib 6.3.2, 2023. URL: https://doi.org/10.5281/zenodo.7771749. doi:10.5281/zenodo.7771749.

[5] A. Le-Tuan, C. Hayes, M. Wylot, D. Le-Phuoc, Rdf4led: An rdf engine for lightweight edge devices, in: Proceedings of the 8th International Conference on the Internet of Things, 2018, pp. 1–8.

[6] I. Bilenchi, F. Scioscia, M. Ruta, Cowl: A lightweight owl library for the semantic web of everything, in: International Conference on Web Engineering, Springer, 2022, pp. 100–112.

[7] A. Bento, L. Médini, K. Singh, F. Laforest, Do arduinos dream of efficient reasoners?, in: European Semantic Web Conference, Springer, 2022, pp. 289–304.

[8] V. Charpenay, S. Käbisch, H. Kosch, Towards a binary object notation for rdf, in: The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15, Springer, 2018, pp. 97–111.

[9] K. Sahlmann, F. Mikolajczak, B. Schnor, Interoperability in the iot–an evaluation of the semantic-based approach, arXiv preprint arXiv:2203.14585 (2022).