POSTER: Simplifying the Networking of Wireless Embedded Systems using a Large Language Model

Pramuka Medaranga*
National University of Singapore
pramukas@comp.nus.edu.sg

Dhairya Shah*
National University of Singapore
dhairya@nus.edu.sg

Savitha Viswanadh Kandala*
National University of Singapore
kandalaviswanadh@gmail.com

Ambuj Varshney
National University of Singapore
ambujv@nus.edu.sg

ABSTRACT

Wireless embedded systems have seen rapid growth. Nonetheless, the further growth of these systems is now threatened due to the complexities of programming, networking, and deploying them. Developing embedded systems today is a challenging task primarily due to the vast diversity in radio standards, network stacks, microcontrollers, sensors, and other parts of the embedded system ecosystem. This complexity makes it a task suited only for experts, thus hindering the broader adoption of embedded systems.

In this early work, we tackle a particular aspect of this challenge related to the complexities of wireless communication. Specifically, we explore the hypothesis that the emergent properties of large language models could help mitigate the complexities of programming the wireless communication stack. We adopt a complex radio transmitter design based on a backscatter mechanism and explore the generation of the transmitter's logic using state-of-the-art language models. Our exploration leads to insights that, with appropriate prompting, today's state-of-the-art large language models are already capable of generating complex modulation and backscatter logic. This finding warrants further efforts to design language model generated radio transmitters to simplify embedded systems.

CCS CONCEPTS

• **Hardware** → *Networking hardware*; • **Computing methodologies** → *Natural language processing*;

KEYWORDS

Embedded systems, networking, Large language models

ACM Reference Format:
Pramuka Medaranga, Dhairya Shah, Savitha Viswanadh Kandala, and Ambuj Varshney. 2024. POSTER: Simplifying the Networking of Wireless Embedded Systems using a Large Language Model. In *ACM SIGCOMM 2024 Conference (ACM SIGCOMM Posters and Demos '24), August 4–8, 2024, Sydney, NSW, Australia.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3672202.3673752

*These authors contributed equally to this work and are co-primary authors.

ACM SIGCOMM '24, August 4–8, 2024, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0717-9/24/08.
https://doi.org/10.1145/3672202.3673752

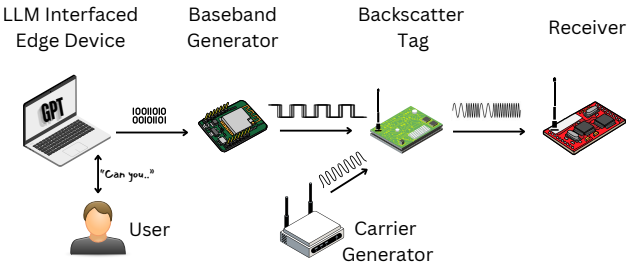


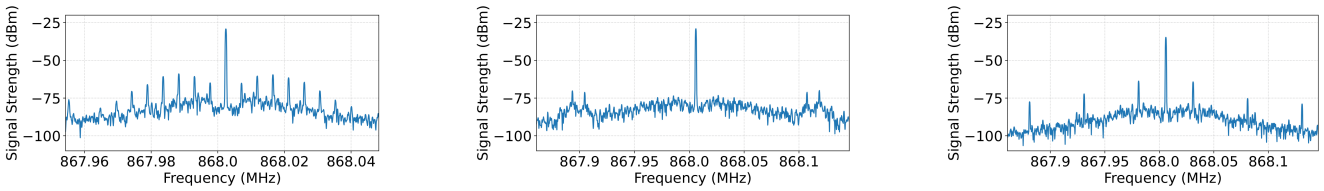
Figure 1: *Overview of the system.* An end-user provides input in natural language to an LLM, specifying details such as modulation scheme, intermediate frequency, bitrate, and the embedded platform. The end-user may also provide the datasheet to the LLM. The LLM generates relevant code for instantiating a radio transmitter. The radio waves generated by the transmitter are observed using a spectrum analyzer.

1 INTRODUCTION

The rapid growth in the number of embedded systems is hindered by challenges related to programming, networking and deploying them. Today, a huge diversity exists in the various components that make up a embedded system [5, 8], including programming languages, frameworks, and networking stacks. Consequently, even experts face significant challenges [2] in learning the intricacies of the embedded systems, and deploying them in the real-world.

If we consider the specific case of wireless communication, there are numerous wireless standards with details extensively covered in voluminous manuals. At a lower level of abstraction, these standards use radio transceivers. Hundreds of radio transceivers from various manufacturers exist, each with varied complexities in interfacing them to a microcontroller. The specific details are spelled out in the datasheet, often spanning hundreds of pages [9].

The past several years have seen significant interest in large language models (LLMs). Based on the transformer architecture [13], these models are now demonstrating capabilities that enable numerous applications. LLMs are now being applied in various contexts, including controlling robots [3], exhibiting agentic behavior for controlling tools [7], and assisting in writing code for software development [2]. With the scaling of these models and their emergent capabilities, there is a sincere debate about whether these models demonstrate a level of reasoning and understanding that could lead to general intelligence [1]. However, the use of LLMs to help with programming of the wireless communication stacks remains largely unexplored. Due to the complexity of programming wireless stacks,



(a) 2-FSK (d = 5 kHz, IF = 15 kHz) **(b) 2-FSK (d = 35 kHz, IF = 78 kHz)** **(c) OOK**

Figure 2: Spectrum of signals emitted by a language model generated backscatter transmitter. We vary modulation , frequency deviations (d), intermediate frequencies (IF) at symbol rate of 10 kbps by providing natural language prompts to a LLM.

the ability of LLMs to simplify the process of programming protocols and writing code for low-level transmitters can significantly help to streamline the task of networking embedded systems. We focus on this particular aspect and examine the ability of LLMs to generate logic for backscatter transmitters [4, 6, 10–12].

Backscatter enables transmissions through reflections or absorption of ambient wireless signals instead of locally generating higher frequency carrier signal [4, 6, 10]. This mechanism allows for low-power transmissions compared to commodity transmitters. In a typical backscatter system, a microcontroller generates digital signals that toggle a switch controlling an antenna. This causes the antenna to reflect or absorb ambient signals according to the baseband signal, generating the reflected backscattered signals.

Despite the benefits, the backscatter mechanism has seen limited adoption due to the complexity of implementation. For example, implementing this mechanism on a microcontroller requires careful and precise control of a general-purpose input/output (GPIO) pin according to the baseband signal. Generating the baseband signal involves using appropriate timers, understanding the intricacies of the modulation scheme, microcontroller architecture, wireless standards, and managing other intricate protocol details like sequences for preamble and error check mechanisms. These complexities motivate us to use backscatter as an appropriate use case to test the hypothesis that LLMs are now sufficiently capable of generating complex logic to support wireless networking in embedded systems.

2 DESIGN

Prompt

Generate a baseband code for ESP32 using the Arduino framework to perform 2-FSK modulation by toggling the output voltage state on a digital GPIO pin.

The system consists of a software defined radio (SDR), an embedded platform, a backscatter frontend, an edge computer interacting with an LLM, and a receiver that tracks the backscattered signal. We provide an overview of the system in Figure 1. The end-user provides prompts to an LLM at the edge device. We do not fine-tune the model or provide any other external documentation like datasheets, allowing us to understand the baseline capabilities of LLM. Next, the generated code is used to program the embedded platform. Specifically, we ask the LLM to generate code for Frequency Shift Keying (FSK) and On-off Keying (OOK) as modulation through the shown prompts. We generate a carrier signal using SDR. The embedded platform then toggles the backscatter frontend, reflecting or absorbing the carrier signal, and we observe the backscattered signal using a spectrum analyzer as a receiver.

Implementation. We use OpenAI, GPT-4o as the LLM. For the embedded platform, we used an ESP32-S3-Devkit. We tracked the signal using a Signalhound BB60C spectrum analyzer.

3 EVALUATION

We observed that the particular LLM is capable of generating complex code to implement modulation schemes such as 2-FSK and OOK. Figure 2 shows the spectrum plot for transmissions from the language model-designed backscatter transmitter.

Subsequent Prompt

Refactor and regenerate the above code as per the following instructions: 1. Set the modulation parameters: Intermediate Frequency (IF) = 15 kHz*; Standard Deviation (d) =* 5 kHz*; and Symbol Rate =* 10 kbps *2. The observed modulation parameters differ from those expected. Make use of the precise timer peripheral present on the ESP32-S3-Devkit.*

Observation 1. Providing detailed prompts with information regarding intermediate frequency, bitrate, and other modulation parameters helps LLM to generate accurate code. It prevents LLMs from hallucinating critical baseband information.

Observation 2. Specifying hardware details such as peripherals helps generate accurate code. For example, we found deviations between the expected intermediate frequency and the observed frequency. This was due to the LLM ignoring the limitations of the ESP32's clock. We corrected this behavior by specifying the use of timers for baseband logic as a prompt, leading the LLM to generate code that precisely controlled the GPIO.

Observation 3. Using less common embedded platforms or complex modulation schemes requires additional prompting cycles. It may also require providing a detailed PDF with the datasheet to the LLM, as it may lack the ability to generalize from its dataset.

4 CONCLUSION

We presented the first study exploring the feasibility of using an LLM for designing a radio transmitter. We will explore fine-tuning an LLM with information for supporting complex protocols and expanding the study to a broader range of embedded platforms.

Acknowledgement. We thank the anonymous reviewers. This work is funded by a grant from NUS-NCS Joint Center (A-0008542-02-00) and is also partially supported by a startup grant (A-8000277-00-00), a MoE Tier 1 grant (A-8001661-00-00), and an unrestricted gift from Google through their Research Scholar Program (A-8002307-00-00), hosted at National University of Singapore.

REFERENCES

[1] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. (2023). arXiv:cs.CL/2303.12712

[2] Zachary Englhardt, Richard Li, Dilini Nissanka, Zhihan Zhang, Girish Narayanswamy, Joseph Breda, Xin Liu, Shwetak Patel, and Vikram Iyer. 2024. Exploring and Characterizing Large Language Models for Embedded System Development and Debugging. In *Extended Abstracts of the 2024 CHI Conference on Human Factors in Computing Systems (CHI EA '24)*. Association for Computing Machinery, New York, NY, USA, Article 150, 9 pages. https://doi.org/10.1145/3613905.3650764

[3] Yafei Hu, Quanting Xie, Vidhi Jain, Jonathan Francis, Jay Patrikar, Nikhil Keetha, Seungchan Kim, Yaqi Xie, Tianyi Zhang, Shibo Zhao, Yu Quan Chong, Chen Wang, Katia Sycara, Matthew Johnson-Roberson, Dhruv Batra, Xiaolong Wang, Sebastian Scherer, Zsolt Kira, Fei Xia, and Yonatan Bisk. 2023. Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis. (2023). arXiv:2312.08782

[4] Bryce Kellogg, Vamsi Talla, Joshua R. Smith, and Shyamnath Gollakot. 2017. PASSIVE WI-FI: Bringing Low Power to Wi-Fi Transmissions. *GetMobile: Mobile Comp. and Comm.* 20, 3 (jan 2017), 38–41. https://doi.org/10.1145/3036699.3036711

[5] Richard Lin, Rohit Ramesh, Parth Nitin Pandhare, Kai Jun Tay, Prabal Dutta, Bjoern Hartmann, and Ankur Mehta. 2024. Design Space Exploration for Board-level Circuits: Exploring Alternatives in Component-based Design. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 338, 14 pages. https://doi.org/10.1145/3613904.3642009

[6] Vincent Liu, Aaron Parks, Vamsi Talla, Shyamnath Gollakota, David Wetherall, and Joshua R. Smith. 2013. Ambient backscatter: wireless communication out of thin air. *SIGCOMM Comput. Commun. Rev.* 43, 4 (aug 2013), 39–50. https://doi.org/10.1145/2534169.2486015

[7] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large Language Model Connected with Massive APIs. (2023). arXiv:2305.15334

[8] Rohit Ramesh and Prabal Dutta. 2016. EMBEDDED DEVELOPMENT TOOLS REVISITED: Verification and Generation from the Top Down. *GetMobile: Mobile Comp. and Comm.* 20, 2 (oct 2016), 5–10. https://doi.org/10.1145/3009808.3009810

[9] Texas Instruments. CC1310 launchpad. 2019. Accessed: 15-02-2019. http://www.ti.com/tool/launchxl-cc1310. (2019).

[10] Ambuj Varshney, Carlos Pérez-Penichet, Christian Rohner, and Thiemo Voigt. 2017. LoRea: A Backscatter Architecture that Achieves a Long Communication Range *(SenSys '17)*. Association for Computing Machinery, New York, NY, USA, Article 50, 2 pages. https://doi.org/10.1145/3131672.3136996

[11] Ambuj Varshney, Andreas Soleiman, and Thiemo Voigt. 2019. TunnelScatter: Low Power Communication for Sensor Tags using Tunnel Diodes. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, Article 50, 17 pages. https://doi.org/10.1145/3300061.3345451

[12] Ambuj Varshney, Wenqing Yan, and Prabal Dutta. 2022. Judo: addressing the energy asymmetry of wireless embedded systems through tunnel diode based wireless transmitters. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '22)*. Association for Computing Machinery, New York, NY, USA, 273–286. https://doi.org/10.1145/3498361.3538923

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. (2023). arXiv:1706.03762