Learning BLE and Rust with AI Assistance

Author: Brian Beattie
Last Updated: Dec 1, 2025

History

I wanted to learn BLE and Rust. To this end I got a PineTime watch and a micro:bit + joystick:bit devkit. I started with the micro:bit + joystick:bit by developing a build environment with some example code. Since I had had some experience with using AI as part of the development I decided to use AI as part of the process of developing the build environment and example development. I will be using AI when I am probably talking about Large Language Models(LLM).

> Google says: “Think of AI as the entire concept of smart machines, while an LLM is a powerful tool within that field, like one that powers chatbots and content creation. LLMs are also a subset of **generative AI**, which is a form of AI that can create new content, including text (from LLMs), images, and audio”.

The discussion around AI includes some extreme positions, an ad tells me “AI can develop a webpage in 90 seconds” and we hear reports (accurate or exaggerated) of major corporations replacing development teams with AI. In response some reject the value of AI. I myself have found some value in AI but was skeptical of the more extreme claims.

My view going into this effort was that AI could be a useful tool as part of the evolution of Computer Science, if you will excuse a disordered somewhat arbitrary list, from plugboards to stored program binaries to assembler to FORTRAN and high level languages to objective oriented to Functional vs Declarative languages and IDEs. I expect AI may just be the latest stage in the development of software process paradigms.

The work to this point has been kept as a github repository, docker-microbit-v2.

Working Environment

Being a Linux(debian) user this work was done on a Laptop running Debian Bookworm. I started by developing a docker container setting up a development environment. This container, the construction and invocation of which is part of the github repo mentioned above, was developed with the help of AI, beginning the process of learning what AI could do. The reason for Docker was to provide isolation from the host machine, avoiding conflicts with existing tools and ensuring a stable, reproducible build environment that could be easily shared or recreated.

Process

As I said, I had decided to use AI to help with the process. I started with ChatGPT building a docker container for the micro:bit development board using the *Rust Embedded Discovery Book* and the github code repository associated with it. I also asked ChatGPT to

generate some examples based on that code.

AI Coding

I started with ChatGPT just because that was the one that was getting most of the press. What I started with was:

```
using https://docs.rust-embedded.org/discovery-mb2, in ~/projects/rust-
microbit/docker create an interactive docker container (with the ubuntu
24) to build and develop examples (Dockerfile in docker), using Ubuntu
24.04 as the base image

Chat GPT generated:
~/projects/rust-microbit/
├── docker/
│   ├── Dockerfile
│   └── run.sh
└── examples/
    └── 06-hello-world/
```

At this point I tried building the example and getting ChatGPT to extend the example.

I learned a number of things.

1. It took AI coders a large number of passes to produce code that would compile and more to get code that would run. In fact ChatGPT seemed to end up running in circles so that I abandoned it and tried other AIs.
2. I tried other engines some that stood out were Copilot, Google Gemini and Claude Code.
3. I finally found that *Claude Code* worked best for me, in a Linux, Rust and micro:bit BLE project. (other AI chat tools may work better for other languages or environments)

The AI used in the remainder of this project and document was *Claude*.

As stated there have been claims that AI could produce complete solutions. I did not find this to be the case for the classes of problems I'm interested in. What I found was that AI was useful in managing the boilerplate and initial code and other issues I would classify as *bookkeeping*. Another thing I found, that I was not expecting, was when searching the internet AI searches tended to be broader both in where the searches looked and in finding related topics, for instance AI found that the Senville mini-split and WiFi interface were actually re-badged **Midea** units and that github repositories for Midea existed. Neither of these facts had been discovered using google searches.

In investigating AI coding I worked on two problems. The first was creating something in Rust on the micro:bit and joystick:bit to learn Rust and BLE. The second was creating a tool to control my mini-split for which I had a WiFi interface, no Rust or BLE.

Senville WiFi

In the process of the second problem I granted AI pretty much a free hand. One surprise, as I said, was when AI found the original manufacturer of my HW and internet and github content supporting that HW. This was followed by the iterative process of producing code

that built and then ran. This generated some Python code with both a CLI tool and a Web server. There were a number of problems where code did not work due to unexpected behaviors and idiosyncrasies of the local environment. While the result does function and is sufficient for my own use it is by no means a polished application.

There are things that feel unfinished to me such as presentation and conversion of data. For one, in the Web tool current status mode is displayed as a number, instead of the text “Heat” or “Cool” or such. More importantly the natural temperature unit is Celsius and when using Fahrenheit exact values are not possible and setting and displaying the temperature can be confusing and often entering a whole number temperature F does not produce the desired result. In trying to get AI to fix these problems I found the process was not straightforward, or even successful. I suspect that manipulating the code would be the most effective way to handle these issues.

As an aside; While I was editing this text, the Web tool disconnected from the mini-split and is reporting Communications errors.

I may try to see how much work it takes to get AI to polish this tool.

Rust HID joystick

The first problem had two goals: primarily, to develop a base from which to explore and learn Rust and BLE; and secondarily, to explore and demonstrate AI's capabilities for embedded systems development. While AI could certainly complete a full HID joystick implementation, AI would likely be overkill for such a bounded task, given the constrained hardware and well-documented protocol. This was the aim of this project. I plan to use it as a foundation for hands-on Rust and BLE coding, extending functionality in directions to be determined.

> **NB** To give credit; Claude helped me with the wording of the above paragraph. I’ve been using it to proofread my work.

Reflections

Discussions of AI that I have seen have tended to extremes: some speak of AI changing software development completely, replacing software engineers, while others dismiss AI's value entirely. In this heated environment, I set about gaining experience with AI to develop a reasoned, rational position.

The experience I have developed here has led me to a number of conclusions. My first was that AI is not going to replace software engineers, certainly in the near term. My second was that AI coding is, or will be just the latest in the evolution of software development paradigms. While the history of software development is no guarantee of the future, it shows improvements in productivity and increases in the population of software developers. To put it concisely: AI is a tool that will reshape the work, but the profession itself isn't going away.

Cautions

I used Claude to proofread this document and provide suggestions. This revealed limitations: Claude caught spelling errors but missed grammatical issues like missing

prepositions. It also didn't flag problems with consistency, continuity, or organization, in fact if I had had a professional proofreader/editor I would expect that their suggestions would produce a significantly different and better document. This may reflect my limitations rather than the tool's flaws, but it illustrates a fundamental point: AI isn't human and doesn't think like one.

Final Conclusion

AI can manage the boilerplate, initial code and build setup and even a significant amount of the body of a solution, but as seen in the Senville project and this document the final polish requires more than AI. It may be a corollary of the 90/90 rule (Tom Cargill, Bell Labs): “The first 90 percent of the code accounts for the first 90 percent of the development time. The remaining 10 percent of the code accounts for the other 90 percent of the development time.” Perhaps with AI, it handles the first 90% while the programmer handles the other 90%—though hopefully the actual ratio is more favorable.