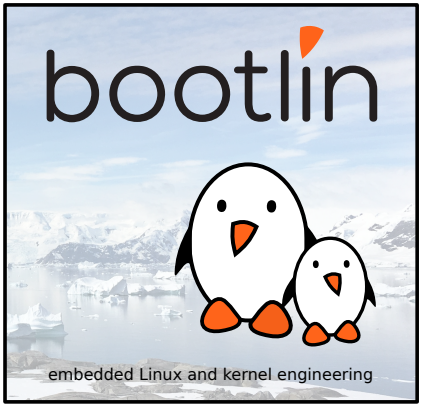
Using Visual Studio Code for Embedded Linux Development

Michael Opdenacker
michael@bootlin.com



Michael Opdenacker

- Founder and Embedded Linux engineer at Bootlin:
  - Embedded Linux engineering company
  - Specialized in low level development: kernel and bootloader, embedded Linux build systems, boot time reduction, secure booting, graphics layers...
  - Contributing to the community as much as possible (code, experience sharing, free training materials)
- Current maintainer of the Elixir Cross Referencer indexing the source code of Linux, U-Boot, BusyBox... (https://elixir.bootlin.com)
- Interested in discovering new tools and sharing the experience with the community.
- So far, only used Microsoft tools with the purpose of replacing them!

In the Stack Overflow 2019 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool, with 50.7% of 87,317 respondents claiming to use it (Wikipedia)

Disclaimer and goals

- I'm not a Visual Studio Code guru!
- After hearing about VS Code from many Bootlin customers, I wanted to do my own research on it and share it with you.
- The main focus of this research is to find out to what extent VS Code can help with embedded Linux development, and how it compares to the Elixir Cross Referencer in terms of code browsing.
- Please share your experience while this pre-recorded talk is played, by posting comments and questions. I'll be available during and after the talk.

Visual Studio Code

https://code.visualstudio.com/

- *Visual Studio Code is a* **free** *source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.* (Wikipedia)
- First release in 2015
- Over 16,000 extensions developed by the community
- **Not a free software / open-source project**, but based on the *vscode* open-source project which is MIT licensed: https://github.com/microsoft/vscode

Image source (Wikipedia): https://frama.link/GoHrYMy5

Issues in Visual Studio Code

- Telemetry:
  - Visual Studio Code collects usage data and sends it to Microsoft. The *Microsoft Privacy Statement* is about all Microsoft products and doesn't reveal details about such data collection.
  - But can can be disabled in user settings (look for `telemetry`), and the source code is available.
- Extensions marketplace:
  - Another issue is that Microsoft prohibits non Visual Studio applications from downloading binaries from their extensions marketplace.
  - See section 1.b. of their Marketplace terms
  - This is disappointing as most extensions are open-source and not developed by Microsoft.
  - As a consequence, extension owners have to push their extensions to other extension registries too.

An alternative: VSCodium

https://vscodium.com/

- A 100 % open-source build of Visual Studio Code, with telemetry disabled
- Strengths:
  - Very frequent updates, staying in sync with VS Code releases
- Limitations:
  - Doesn't have all of VS Code extensions (cannot use the same marketplace). Missing extensions: *DeviceTree*, *kconfig*, *Embedded Linux Kernel Dev.*
  - In particular, C/C++ support is built-in and doesn't work out of the box. See my bug report.
- Project definitely worth supporting, but we will stick to VS Code for the moment.

Image source (VSCodium project): https://frama.link/0onEnRzS

Other related solutions

Theia IDE (https://theia-ide.org/)

- Directly supports VS Code extensions
- But a different project with its own architecture, more modular and allowing for more customizations. Designed from the ground up to run on both cloud and desktop.
- Developed and hosted by a vendor-neutral open-source foundation (Eclipse Foundation)
- Adopted in many IDEs: new Arduino *PRO IDE*, ARM *mbed*, *Eclipse Che*, SAP's web IDE...

Eclipse OpenVSX (http://open-vsx.org/)

- Vendor-neutral open-source alternative to the Visual Studio Marketplace

See the very clear article on https://frama.link/MYGsYuXg for details.

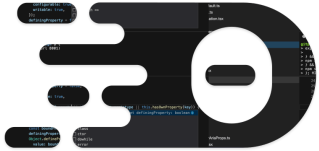


Image source (Theia project): (https://theia-ide.org/)

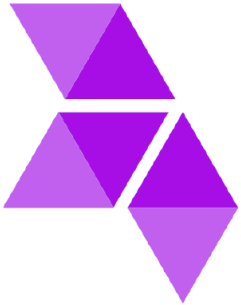


Image source: https://github.com/open-vsx

Quick Visual Studio Code glossary

For words you will see in the interface...

- **Intellisense** : Code Completion Tool built into Microsoft Visual Studio. See https://code.visualstudio.com/docs/editor/intellisense
- **Emmet**: The essential toolkit for web-developers
- **Language Server Protocol**: open protocol for communication between code editors and IDEs. See https://en.wikipedia.org/wiki/Language_Server_Protocol

Demo: Visual Studio Code setup

- Installation on Ubuntu
- Quick interface overview
- Disable telemetry
- Most important shortcuts

Demo: cpptools extension

- For C and C++
- License: proprietary, but based on `https://github.com/microsoft/vscode-cpptools/` (MIT)
- Open the Linux kernel source directory
- Looking up identifiers
- Expansion of defines
- Typing code and autocompletion

Demo: VIM extension

- https://github.com/VSCodeVim/Vim
- License: MIT
- Easy to enable, all commands seem to work

Note: other extensions exist for other editors: Emacs, Atom...

Demo: Checkpatch Lint extension

- https://github.com/idanpa/vscode-checkpatch
- License: MIT
- Need to install `checkpatch.pl` on your system
- Very useful to create code that's compliant with the Linux kernel coding style and rules right from the start.

Demo: Kconfig syntax highlighting

Kconfig syntax highlighting

- https://github.com/luveti/kconfig-vscode
- License: MIT
- Just for syntax highlighting in Linux kernel and Buildroot configuration parameter definitions. No symbol lookup.
- Comparing to Elixir capabilities

Demo: Embedded Linux Kernel Dev extension

- https://github.com/microhobby/linuxkerneldev
- License: MIT
- Symbol autocompletion, function and symbol navigation, for C, Kconfig, defconfig, .config and device tree files.
- Also adds some automation to match device tree compatibles and open their respective driver or documentation files.
- Based on `universal-ctags` but less advanced than Elixir

Demo: GitLens

- https://github.com/eamodio/vscode-gitlens.git
- License: MIT
- Git blame feature

Demo: Linux kernel cross-compiling

- Setting the environment in a terminal
- Using the terminal to configure and build the kernel.
- Didn't find much added value here. cpptools could help using cross-toolchains.

Demo: cross-compiling a simple C program

- How to specify the use of a cross-compiler
- How build tasks are defined

Demo: CMake Tools

From Microsoft

- https://github.com/microsoft/vscode-cmake-tools
- License: MIT
- Can generate a template project for you
- Nice feature: capability to detect cross-toolchains (*Kits*), but partially broken.
- Support for multiple compiling profiles (production, debug, size or speed optimizations...)
- Limitation: example code and CMake files only for C++ (no C, no Rust...)

Demo: remote debugging with gdbserver

- Implementing a script to cross-compile code, deploy the generated executable on the target through SSH, and start it remotely through `gdbserver`.
- Customizing VSCode for the use of a remote debugger.
- Using the remote debugger: inserting breakpoints, step by step execution, reading variables, analyzing a segmentation fault....

Thanks to Karel Vermeiren's article which really helped to make this work.

VS Code and extensions vs Elixir

In terms of code browsing, Elixir Cross Referencer wins for:

- C/C++, Makefiles, Device Tree: provides links to included files
- Device Tree: finding driver bindings
- Kconfig language support

... but VS Code wins for:

- Exposing Git information (Git blame...)
- In place information exposure (symbol prototypes, expansion of defines)

But of course, VS Code offers much more than code browsing!

Visual Studio Code limitations

- No way to filter extensions by license (open-source or proprietary). Same issue on Android by the way!
- cpptools: observed clogging the CPU (or I/O) looking for the declaration of some symbols. Had to close VS Code to end this. Quite rare though.
- Current lack of extensions for embedded Linux applications and tools: cross-compiling toolchains, Valgrind (except output syntax highlighting), strace, ltrace, Yocto Project, Buildroot, nothing for kernel modules, no serial port emulators...
- Currently, VS Code and extensions seem better suited for microcontroller/RTOS work than for embedded Linux.

Conclusions: Visual Studio Code and Embedded Linux

- Visual Studio Code still misses important features for embedded Linux development.
- However, Visual Studio Code is very flexible, easy to extend, and has a great potential for further improvements.
- Great solution for debugging programs, even remotely
- It makes me feel like contributing new extensions, for example to support Bootlin cross-compiling toolchains. This will also benefit to more open platforms (VSCodium, Theia...).

Questions? Suggestions? Comments?

Michael Opdenacker
michael@bootlin.com

https://bootlin.com/pub/conferences/2020/elce