

Zephyr & Visual Studio Code: **How to Develop Zephyr Apps with a Modern, Visual IDE**



Today's Talk

- Why Visual Studio Code (VS Code) is so popular
- Current state of VS Code for embedded development
- Digging into VS Code
- Live demo
- Future topics to explore

Jonathan Beri

@beriberikix, hachyderm.io/@jberi

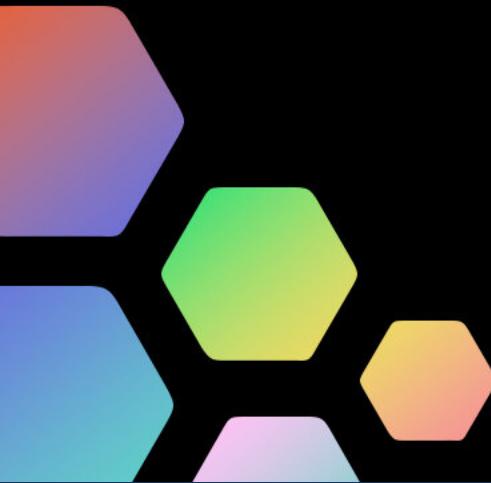
Founder & CEO at Golioth

- Career building developer platforms
- Decade in IoT; Nest, Particle, WeWork
- Working with Zephyr since 2016

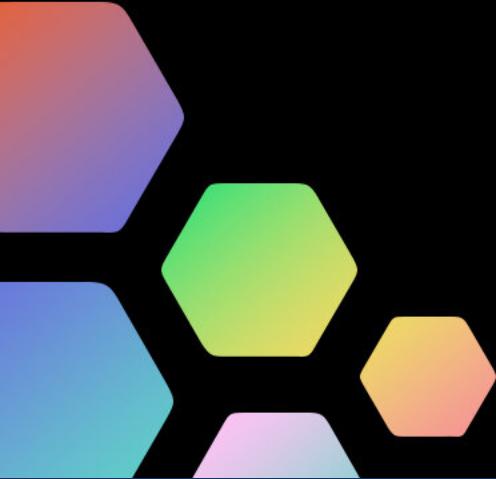


What is Golioth?

- Golioth is an IoT cloud company.
- We make it easy for hardware engineers to connect their sensors and devices to the web without needing to be a cloud expert.
- We use Zephyr to support the widest range of hardware possible.
- We build reference designs with Zephyr and regularly build things using the ecosystem.

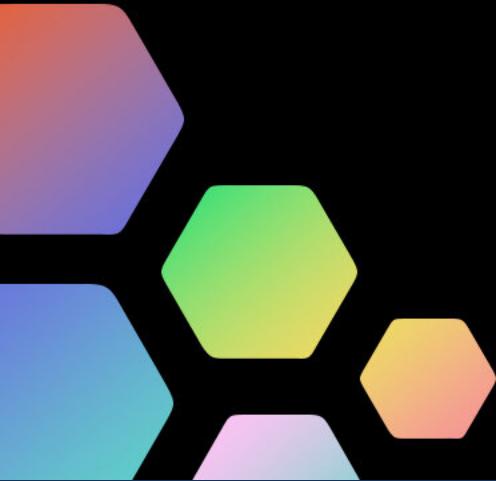


VS Code is Popular



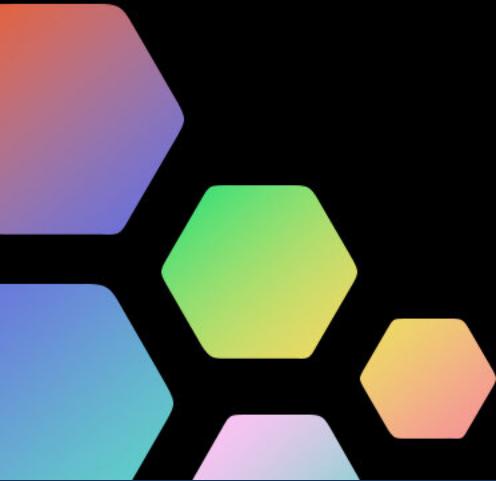
Quick poll: VS Code users in the room

- Who has tried VS Code?



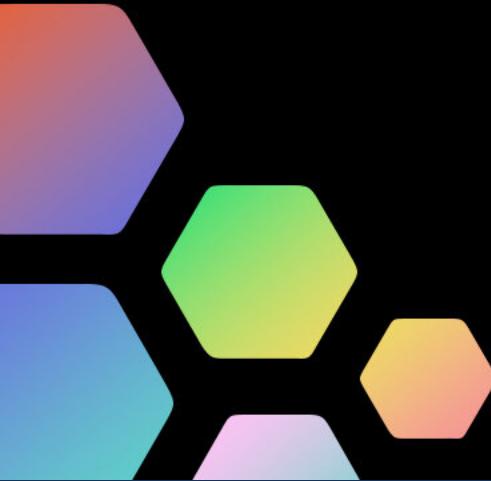
Quick poll: VS Code users in the room

- Who has tried VS Code?
- Who uses VS Code today for non-embedded? Ex. Web, Python, etc?

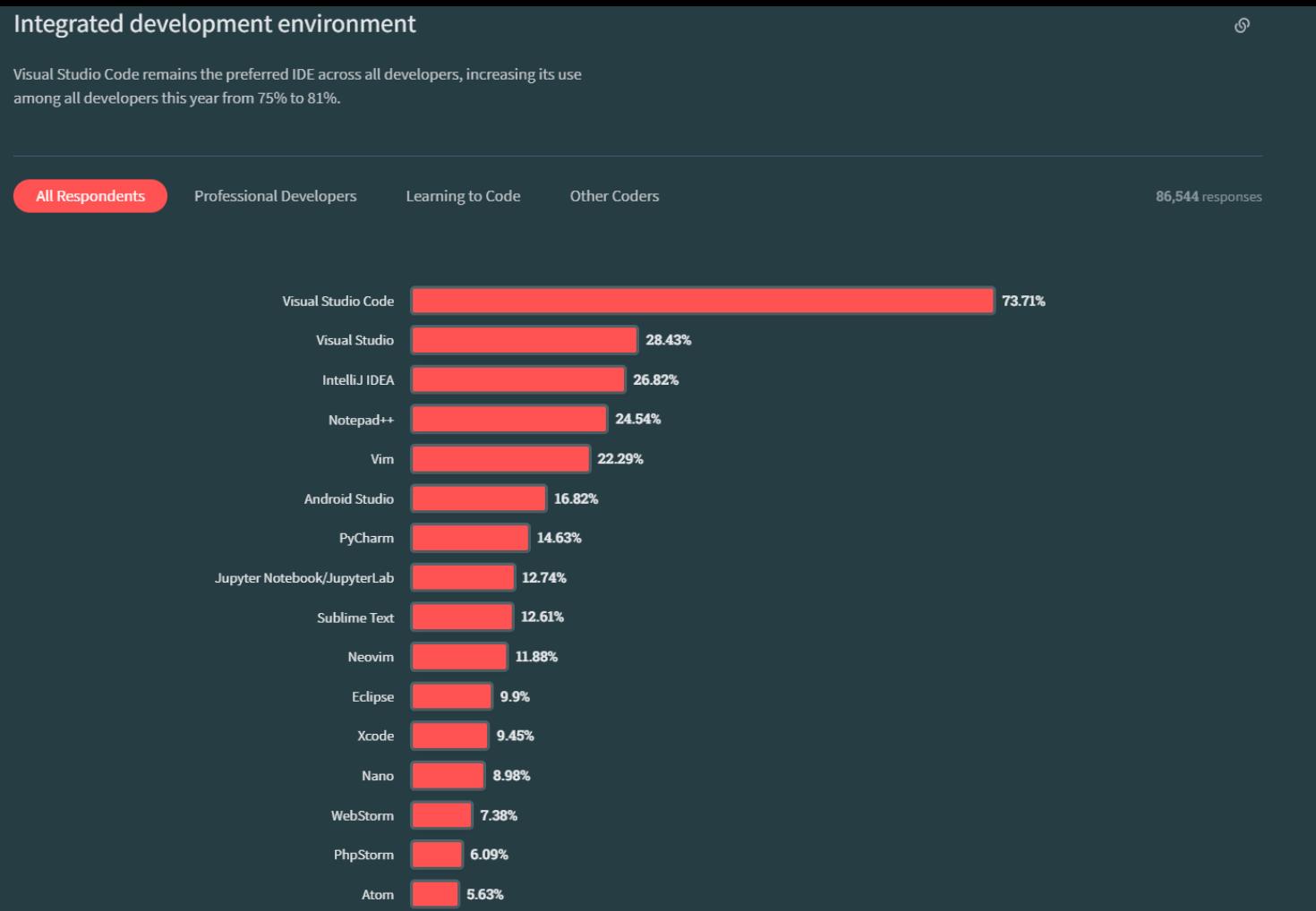


Quick poll: VS Code users in the room

- Who has tried VS Code?
- Who uses VS Code today for non-embedded? Ex. Web, Python, etc?
- Who has tried to use VS Code for embedded?

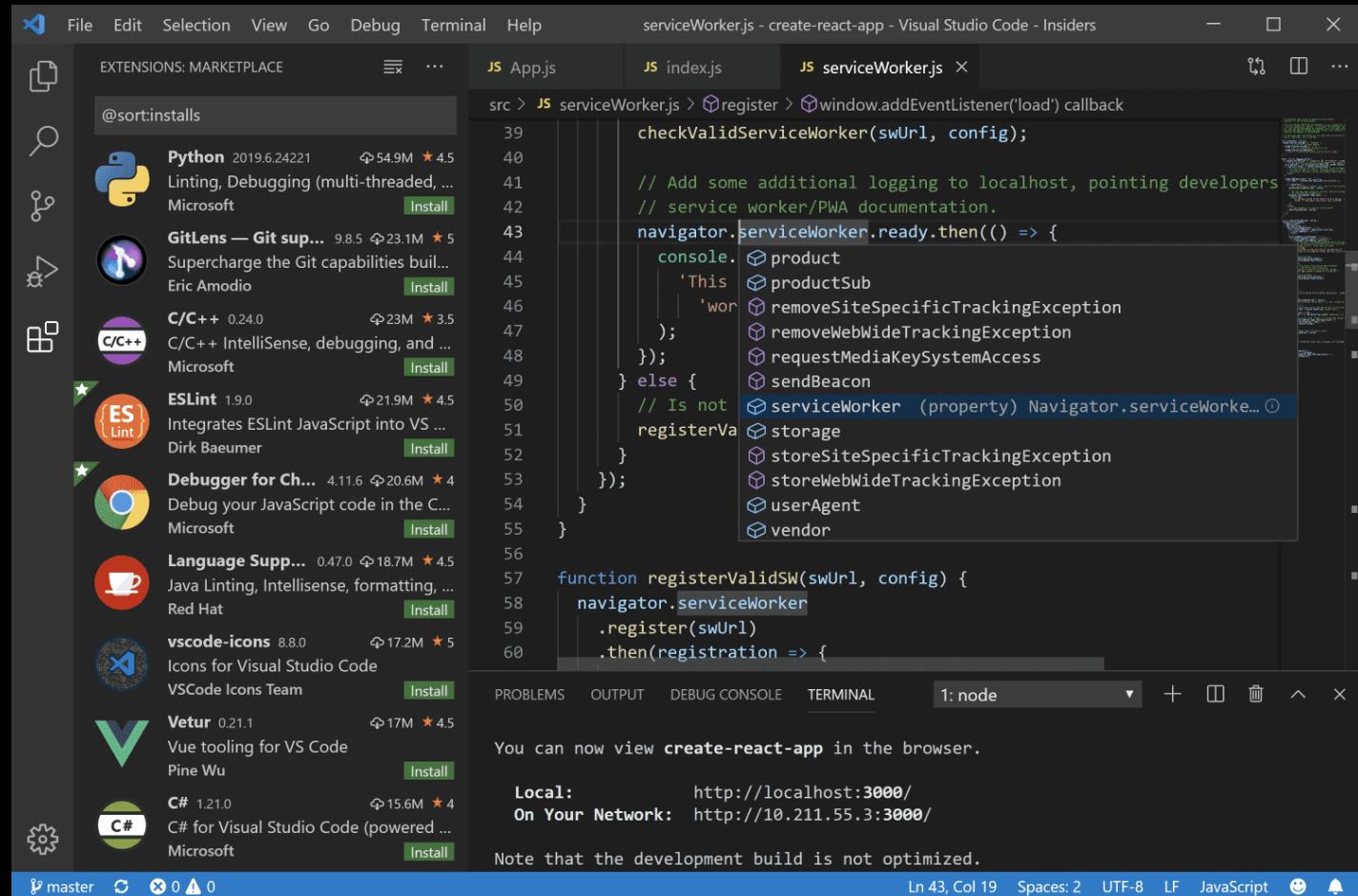


Globally ranked #1 "IDE" among devs



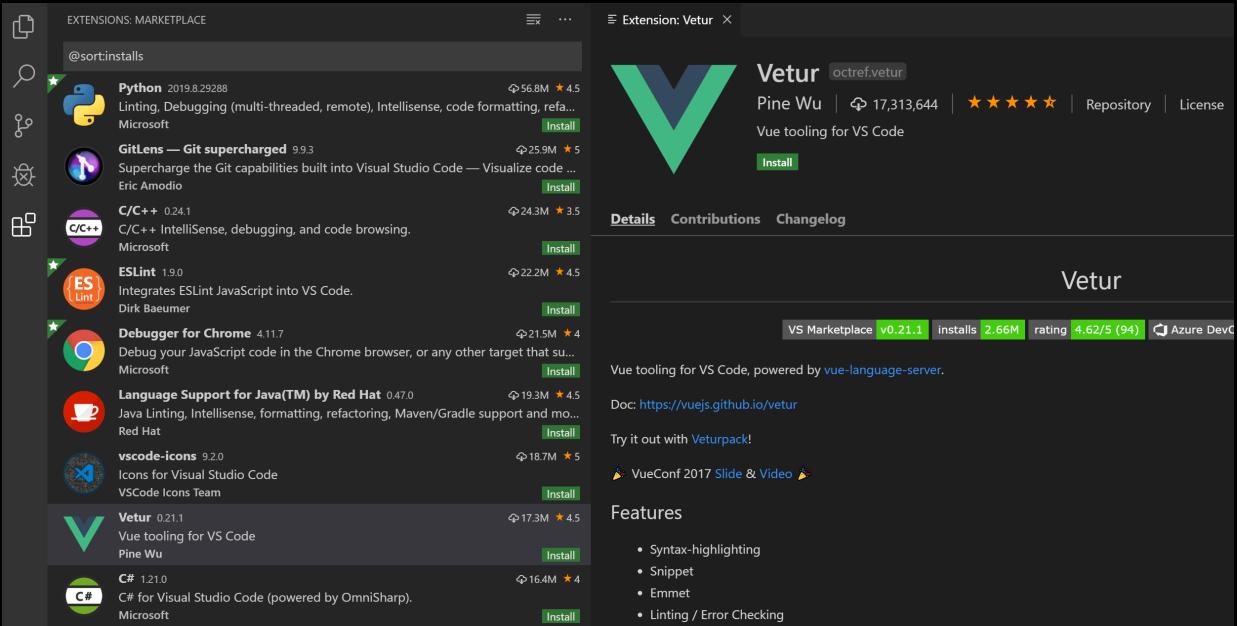
Why is it so popular?

- Free
 - Fast
 - Cross platform
 - Highly customizable
 - Highly configurable
 - IDE feels



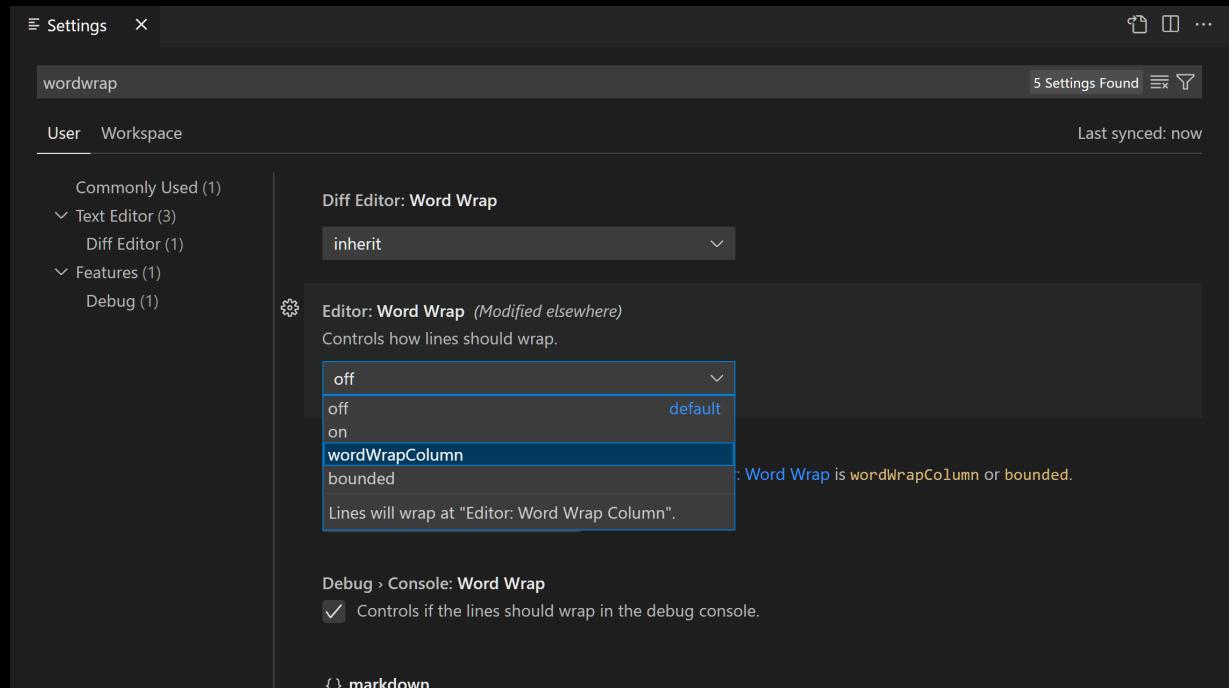
Customizable

- Extensions for most things
 - Programming Langs
 - Themes
 - Debuggers
 - Keymaps
 - Formatters
 - Linters
 - ...much more (180k+)



Configurable

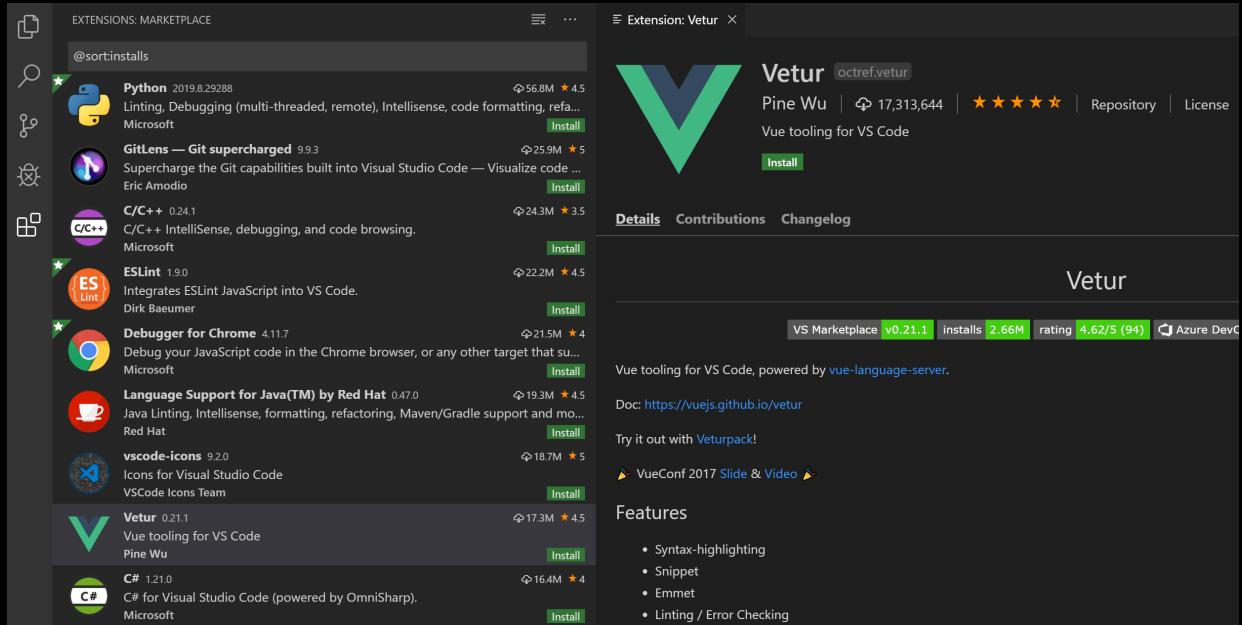
- GUI or JSON
- Settings for everything:
 - Appearance
 - Behavior
 - Extensions
- Global or per project



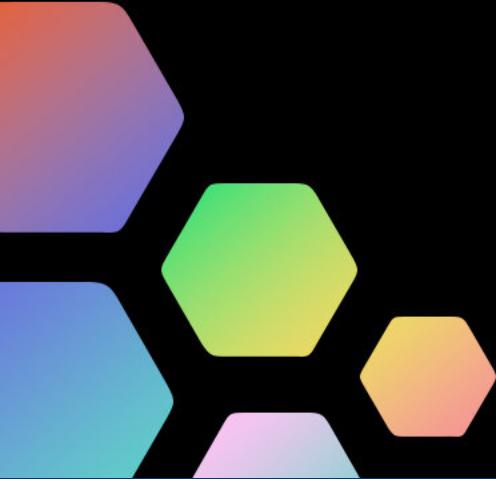
Understanding settings is key to unlocking all the benefits of VS Code!

IDE-like features

- IntelliSense
- Code Navigation
- Refactoring
- Debugging
- Tasks
- Source Control
- Terminal
- Snippets



VS Code for Embedded



Vendor adoption



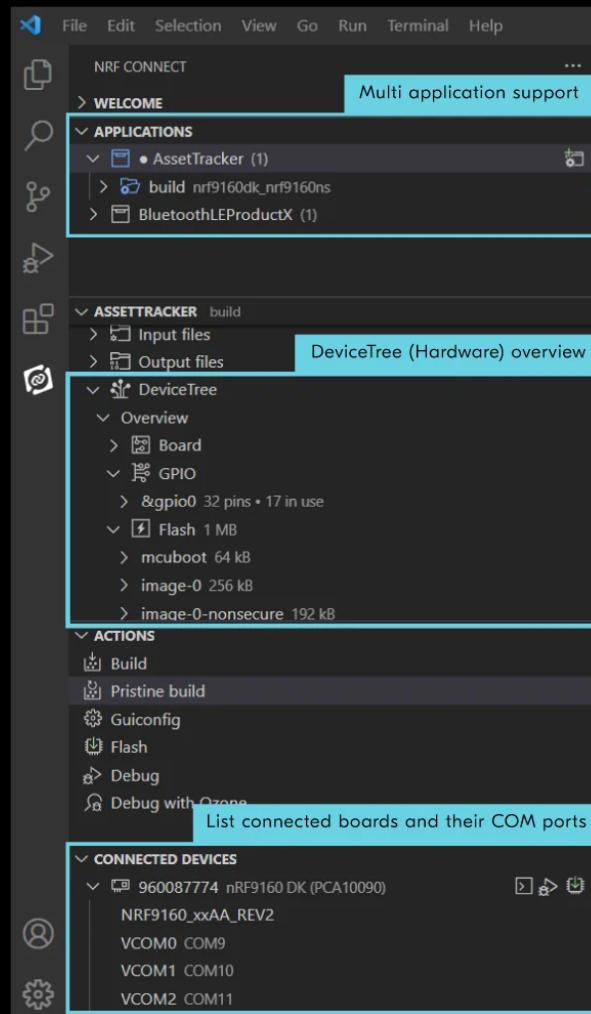
?



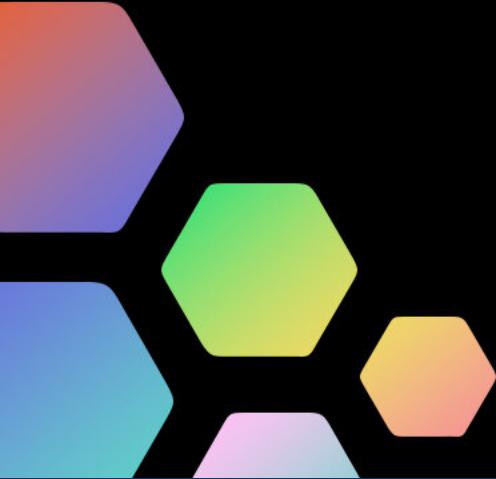
Targetting Zephyr

- Platformio
- nRF Connect from Nordic
- Circuit Dojo
 - Check out Jared's talk after this!
- Coming soon: MCUXpresso from NXP
- DIY

DIY is the focus of this talk



Digging into VS Code

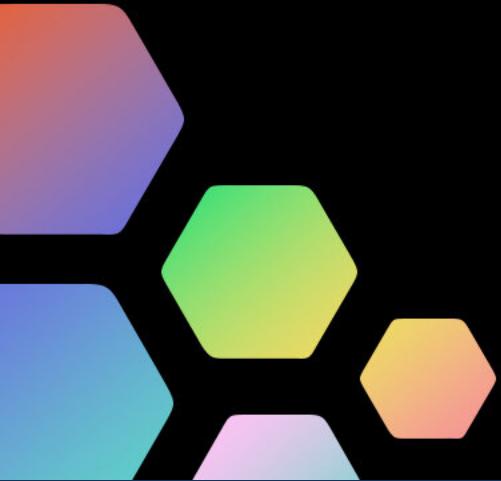


Ideal embedded workflow in VS Code

- "Smart coding"
- Build, Flash, Test & Debug easily
- Reproducible environments

Rest of this talk

- Extensions for embedded
- Configuring VS Code
- Live demo
- Future topics

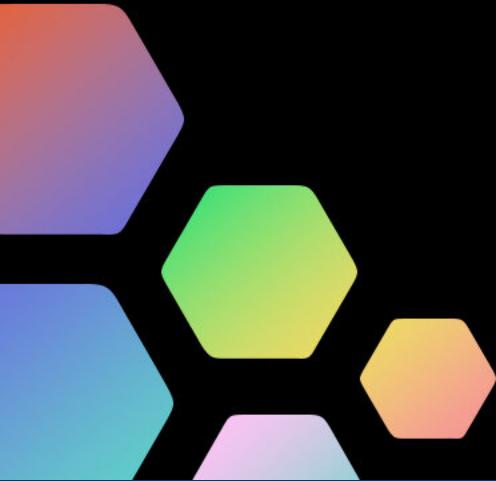


Extensions for Embedded



Types of extensions

- General coding
- Embedded (like Debugging)
- Microsoft vs community vs vendor



C/C++

- IntelliSense
- Syntax Highlighting
- General GDB debugging
- C/C++ Extension Pack
 - includes additional extensions

A screenshot of the VS Code interface showing a code editor window titled "helloworld.cpp". The code is a simple C++ program that includes `<iostream>`, `<vector>`, and `<string>`, and defines a `main()` function. Inside `main()`, there is a declaration of a `vector<string>` named `msg` containing several strings. The cursor is at the end of `msg.`. A code completion dropdown menu is open, listing various member functions of `vector`, such as `assign`, `at`, `back`, `begin`, `capacity`, `cbegin`, `cend`, `clear`, `crbegin`, `crend`, `data`, and `emplace`. The `assign` function is highlighted. To the right of the dropdown, the function signature is shown as `void std::vector<std::cxx11::string>::assign(std::size_t __n, const std::cxx11::string &__val)`, along with a brief description: "@brief Assigns a given value to a %vector. @param __n Number of elements to be assigned. @param __val Value to be assigned." Below this, it says "+2 overloads" and provides a detailed description of the function's purpose.

```
helloworld.cpp ×
helloworld.cpp > main()
1 #include <iostream>
2 #include <vector>
3 #include <string>
4
5 using namespace std;
6
7 int main()
8 {
9     vector<string> msg{"Hello", "C++", "World", "from", "VS Code!", "and the C++ extension!"};
10    msg.|
```

for assign at back begin capacity cbegin cend clear crbegin crend data emplace

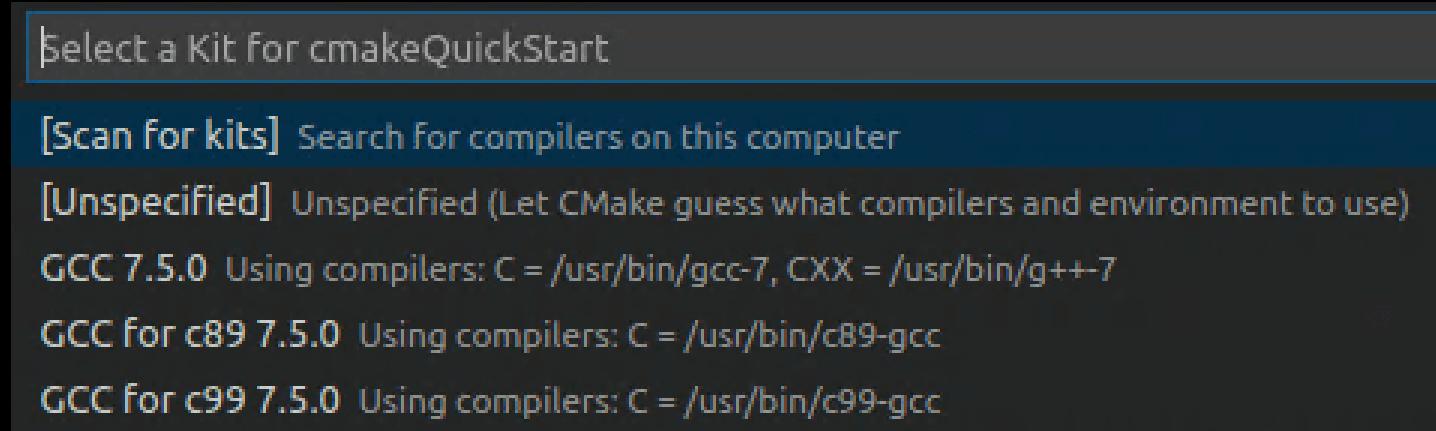
```
void std::vector<std::cxx11::string>::assign(std::size_t __n, const std::cxx11::string &__val)
+2 overloads

@brief Assigns a given value to a %vector.
@param __n Number of elements to be assigned.
@param __val Value to be assigned.

This function fills a %vector with @a __n copies of the given
```

CMake

- Configure, build & debug CMake
- Includes presets (but not for embedded)
- Included in C/C++ Pack



Python

- All the Python things
- Mostly focused on data science and Python web devs

A screenshot of the Visual Studio Code interface. The title bar reads "hello.py - hello - Visual Studio Code". The menu bar includes "Run", "Terminal", and "Help". In the center, there are two tabs: "hello.py" and "hello.py > ...". The main editor area displays the following Python code:

```
1 greeting = 'Hello World!'
```

Debugging: Cortex Debug

- First and popular open source debugger for embedded
 - OpenOCD, JLink, others
 - RTOS aware
 - Advanced features like ITM graphing

The screenshot shows the Eclipse IDE interface for the STM32 Trace Example project. The left panel contains the DEBUG perspective with the following details:

- VARIABLES**:
 - Local:
 - itmdir: 406695062
 - itmvalue: 189457419
 - itmdir2: 1799478117
 - itmval2: 3815390123
- WATCH**: globalCounter = 0
- CALL STACK**: PAUSED ON STEP
main@0x000003e trace_example.c 186
- BREAKPOINTS**:
 - trace_example.c 173
 - trace_example.c 182
 - trace_example.c 207
- COREX REGISTER**:
 - R2 = 0x00000001
 - r3 = 0x40000000
 - r4 = 0x0800004d6
 - r5 = 0x20000051a
 - r6 = 0x000000020
 - r7 = 0x20001f8
- COREX PERIPHERALS**:
 - TIM2 [0x40000000]
 - TIM2 [0x40000000]

The code editor shows `trace_example.c` with the following content:

```
168     TIM2->SR = 0; // Clear interrupt flag
169 }
170
171 int main(void)
172 {
173     globalCounter = 0;
174     configure_tracing();
175     configure_watchpoint();
176
177     RCC->APB2ENR |= RCC_APB2ENR_TIM2EN;
178     GPIOC->CRH = 0x44444433; // GPIOC 8 and 9 as output (STM32F1 discovery leds)
179
180     RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;
181     NVIC_EnableIRQ(TIM2_IRQn);
182
183     TIM2->ARR = 50000;
184     TIM2->DIER = 1;
185     TIM2->CRN = 1;
186
187     unsigned int itmdir = 1;
188     unsigned int itmvalue = 1;
189     unsigned int itmdir2 = 1;
190     unsigned int itmval2 = 1;
191
192     ITM_Print(0, "Boot");
193     ITM_SendValue(6, -1);
194     ITM_SendValue(6, -1000);
195
196     globalCounter = 0;
197
198     for (;;)
199     {
200         delay();
201         GPIOC->BSRR = (1 << 8);
202         delay();
203         GPIOC->BSRR = (1 << 8);
204         // ITM_Print(0, "PORT0");
205     }
206 }
```

The right side of the interface shows the CDT Indexer and Project Explorer.

Debugging: Embedded Tools from Microsoft

- Newer debugger from C++ team
- RTOS aware
- Early Zephyr support

The screenshot shows a Visual Studio Code window for a Zephyr RTOS application named "board_init.c". The code editor displays the following snippet:

```
    _weak void button_a_callback()
{
    WiFi_LED_ON();
    Azure_LED_ON();
    User_LED_ON();
    if (BUTTON_A_IS_PRESSED)
    {
        val += 32;
        if (val > 2047)
            val = 2047;
        RGB_LED_SET_R(val);
        RGB_LED_SET_G(val);
        RGB_LED_SET_B(val);
    }
}

_weak void button_b_callback()
{
    WiFi_LED_OFF();
    Azure_LED_OFF();
    User_LED_OFF();
    if (BUTTON_B_IS_PRESSED)
    {
        val -- 32;
        if (val < 0)
            val = 0;
        RGB_LED_SET_R(val);
        RGB_LED_SET_G(val);
        RGB_LED_SET_B(val);
    }
}

void EXTI4_IRQHandler(void)
{
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_4);
}
```

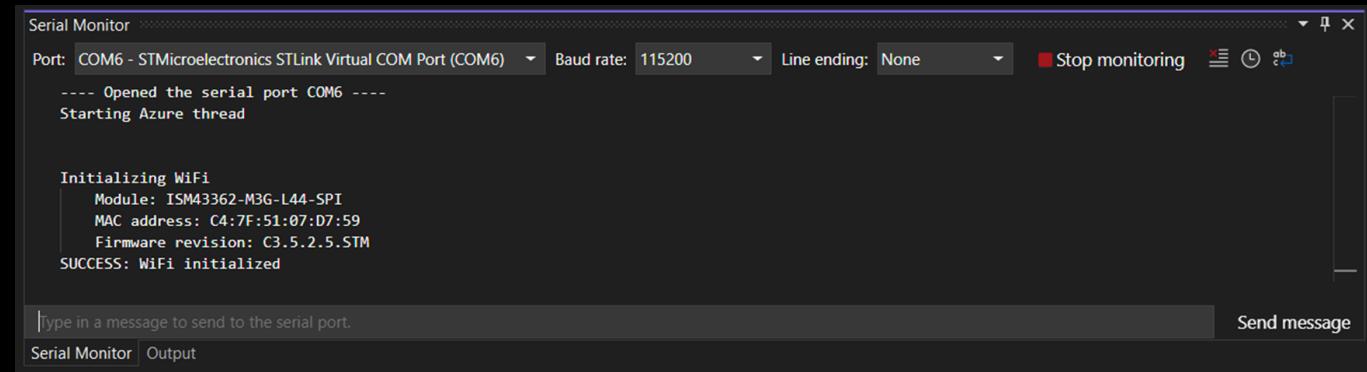
The sidebar on the left includes:

- REGISTER VIEWER: Shows memory locations like CCR1, CCR2, CCR3, CCR4, CR1, CR2, DCR, DIER, DMAR, EGR, PSC, and SNMR.
- VARIABLES: Locals and Registers.
- WATCH: A list of variables being monitored.
- CALL STACK: Paused on step at line 336, showing the stack trace from button_a_callback() down to the interrupt handler.

The bottom status bar shows the file is "PAUSED ON STEP" at line 336, column 1. The status bar also includes icons for master, Launch (AZ3166), Build, Targets In Preset, and No Test Preset Selected.

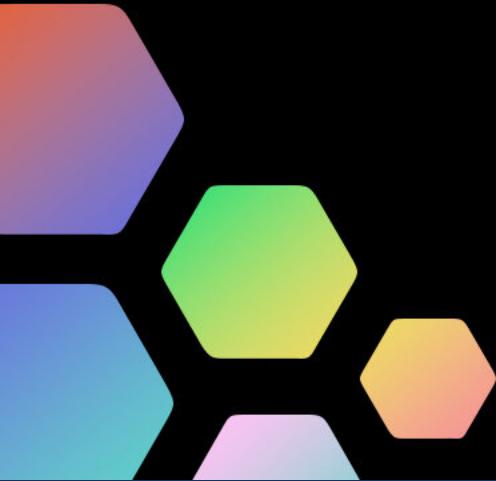
Serial Monitor from Microsoft

- Integrated into terminal
- Multiple instances
- Consistent across OS



Optional, General

- Python Environment Manager
- YAML, RST
- Github Pull Requests & Github Repos
- Remote Development pack
- Docker

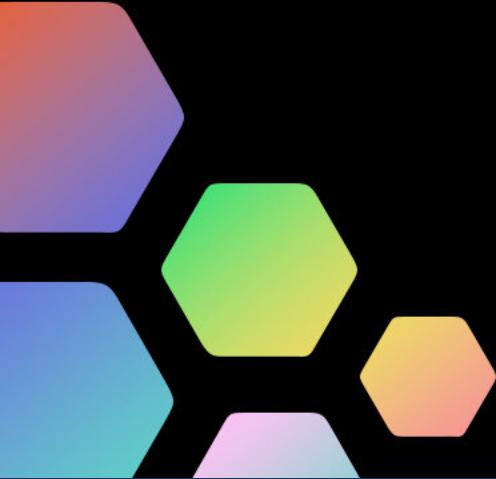


Configuring VS Code



How VS Code does configurations

- GUI or JSON
- Special JSON file names
- Special file location
- Automatically loaded
- Autocompletion



settings.json

```
{  
    // Hush CMake  
    "cmake.configureOnOpen": false,  
  
    // IntelliSense  
    "C_Cpp.default.compilerPath": "${userHome}/zephyr-sdk-0.16.1/arm-zephyr-eabi/bin/arm-zephyr-eabi-  
    gcc.exe",  
    "C_Cpp.default.compileCommands": "${workspaceFolder}/build/compile_commands.json",  
  
    // File Associations  
    "files.associations": {}  
}
```

tasks.json



```
{  
    "version": "2.0.0",  
    "tasks": [  
        {  
            "label": "West Build",  
            "type": "shell",  
            "group": {  
                "kind": "build",  
                "isDefault": true  
            },  
            "linux": {  
                "command": "${userHome}/zephyrproject/.venv/bin/west"  
            },  
            "windows": {  
                "command": "${userHome}/zephyrproject/.venv/Scripts/west.exe"  
            },  
            "osx": {  
                "command": "${userHome}/zephyrproject/.venv/bin/west"  
            },  
            "args": [  
                "build",  
                "-p",  
                "auto",  
                "-b",  
                "nrf52840dk_nrf52840"  
            ],  
            "problemMatcher": [  
                "$gcc"  
            ]  
        }  
    ]  
}
```

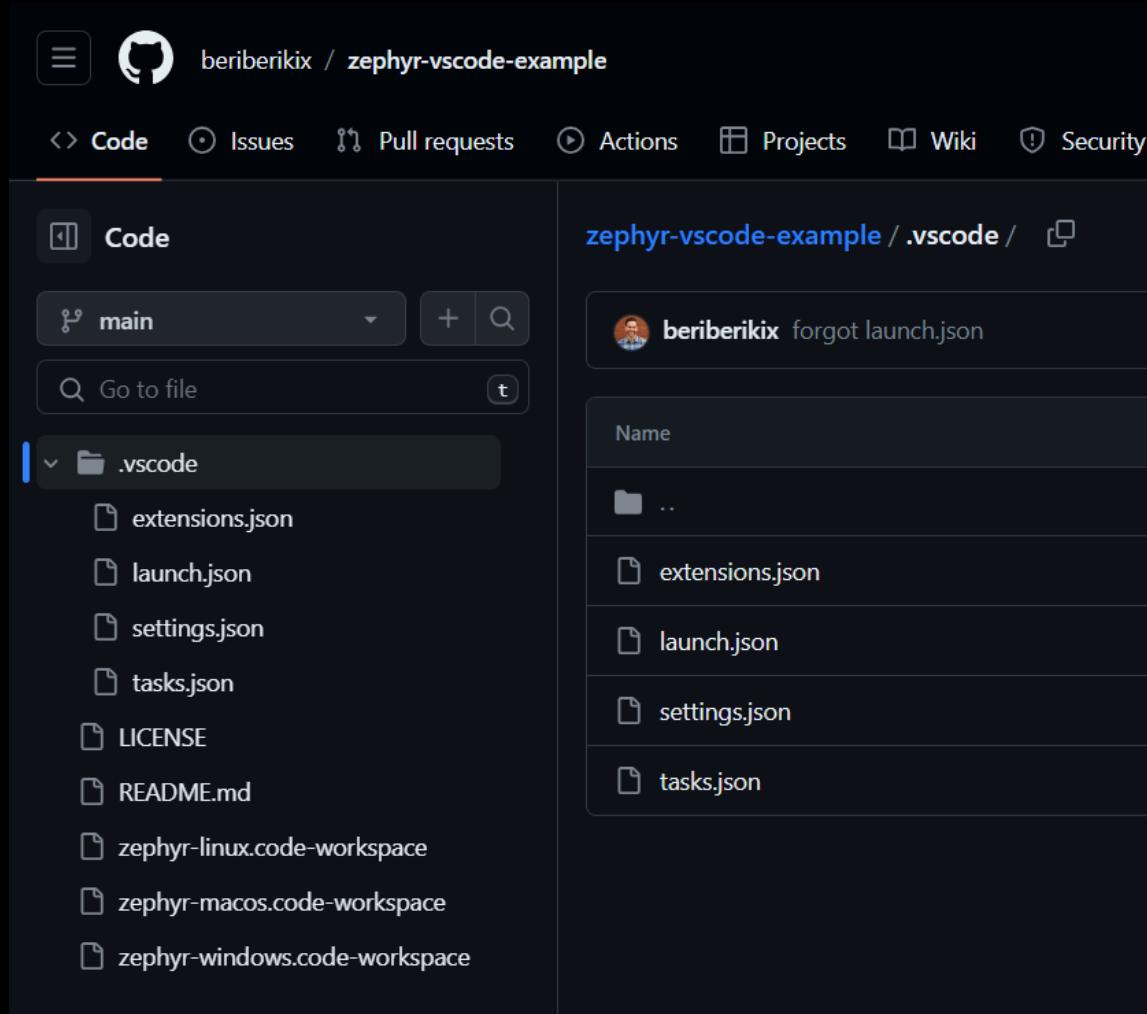
launch.json

```
{  
    "version": "0.2.0",  
    "configurations": [  
        {  
            "name": "Launch",  
            "device": "nRF52840_xxAA",  
            "cwd": "${workspaceFolder}",  
            "executable": "build/zephyr/zephyr.elf",  
            "request": "launch",  
            "type": "cortex-debug",  
            "runToEntryPoint": "main",  
            "serverType": "jlink",  
            "gdbPath": "${userHome}/zephyr-sdk-0.16.1/arm-zephyr-eabi/bin/arm-zephyr-eabi-gdb",  
            "preLaunchTask": "West Build"  
        }  
    ]  
}
```

extensions.json

```
{  
  "recommendations": [  
    "ms-vscode.cpptools-extension-pack",  
    "ms-python.python",  
    "ms-vscode.vscode-embedded-tools",  
    "ms-vscode.vscode-serial-monitor",  
    "marus25.cortex-debug",  
    "donjayamanne.python-environment-manager"  
  ]  
}
```

.vscode folder



Workspaces & .code-workspace

- VS Code "project file"
- Combine .vscode files
- Portable, shareable
- Git-friendly

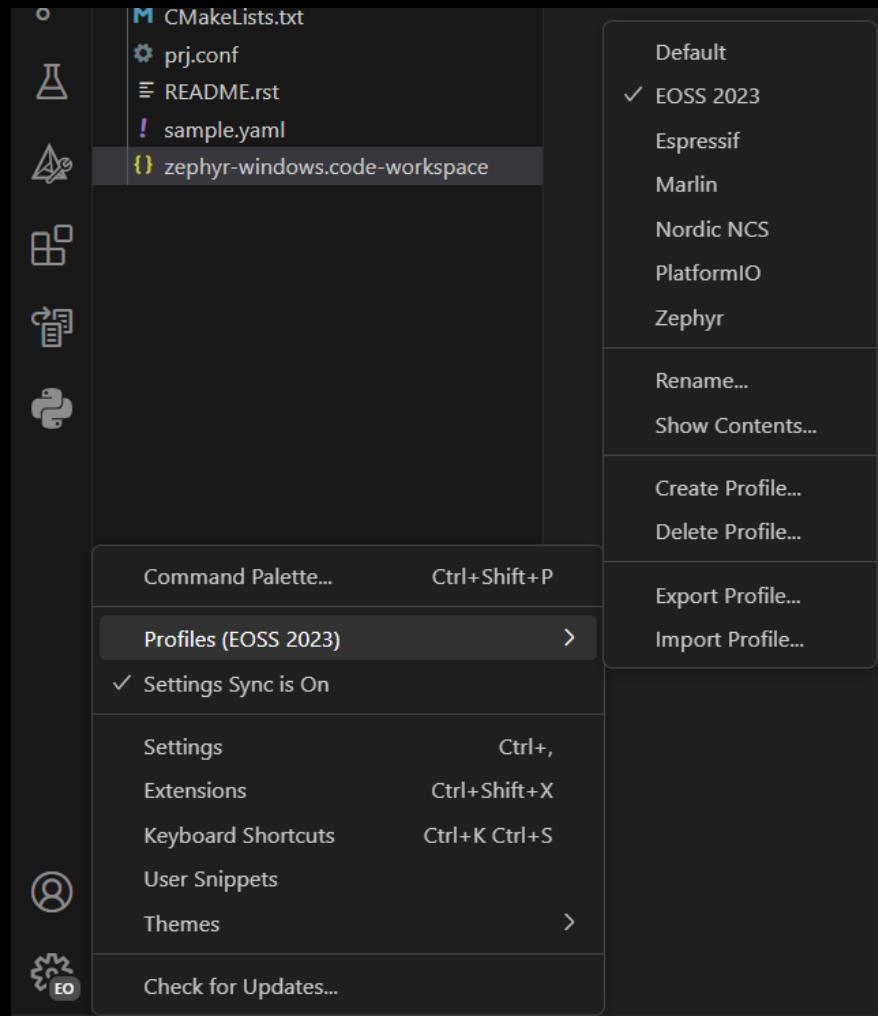


A screenshot of a code editor showing a .code-workspace file. The file contains JSON configuration for a C/C++ project. It includes settings for folders, CMake, IntelliSense (compiler and commands), file associations, tasks, launch configurations, and extensions. The code uses color-coded syntax highlighting for different language elements.

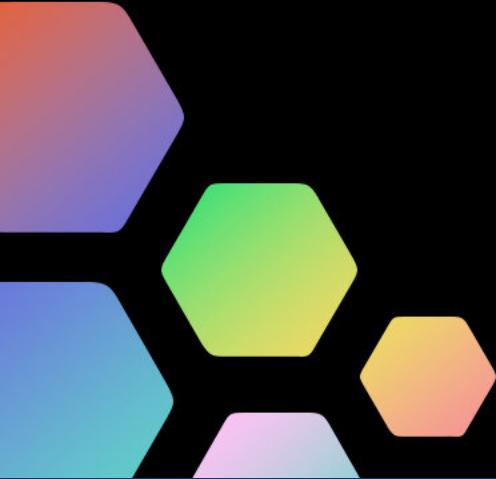
```
{  
  "folders": [  
    {  
      "path": "."  
    }  
  ],  
  "settings": {  
    // Hush CMake  
    "cmake.configureOnOpen": false,  
  
    // IntelliSense  
    "C_Cpp.default.compilerPath": "${userHome}/zephyr-sdk-0.16.1/arm-zephyr-eabi/bin/arm-zephyr-eabi-gcc",  
    "C_Cpp.default.compileCommands": "${workspaceFolder}/build/compile_commands.json",  
  
    // File Associations  
    "files.associations": {}  
  },  
  "tasks": {},  
  "launch": {},  
  "extensions": {}  
}
```

Profiles

- VS Code "environment file"
- Editor-wide configuration
- Also portable, shareable & Git

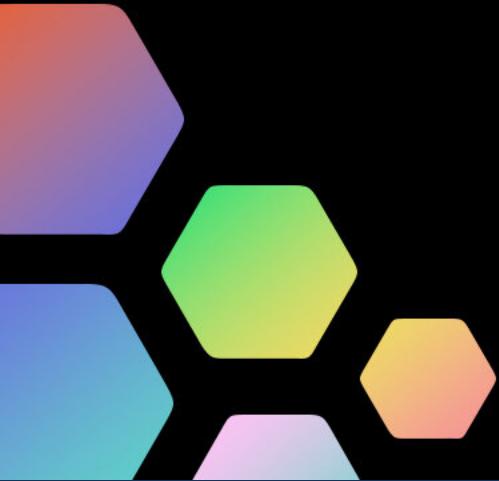


Live Demo

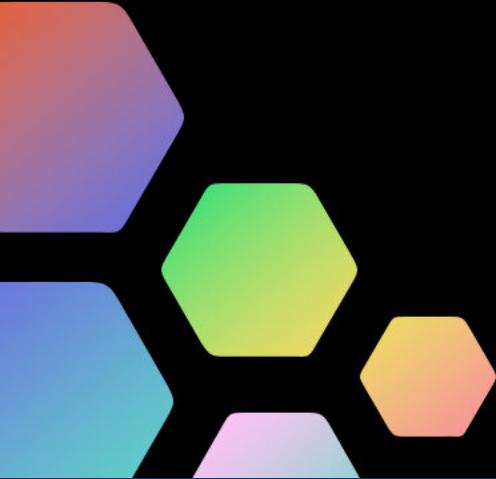


What we saw

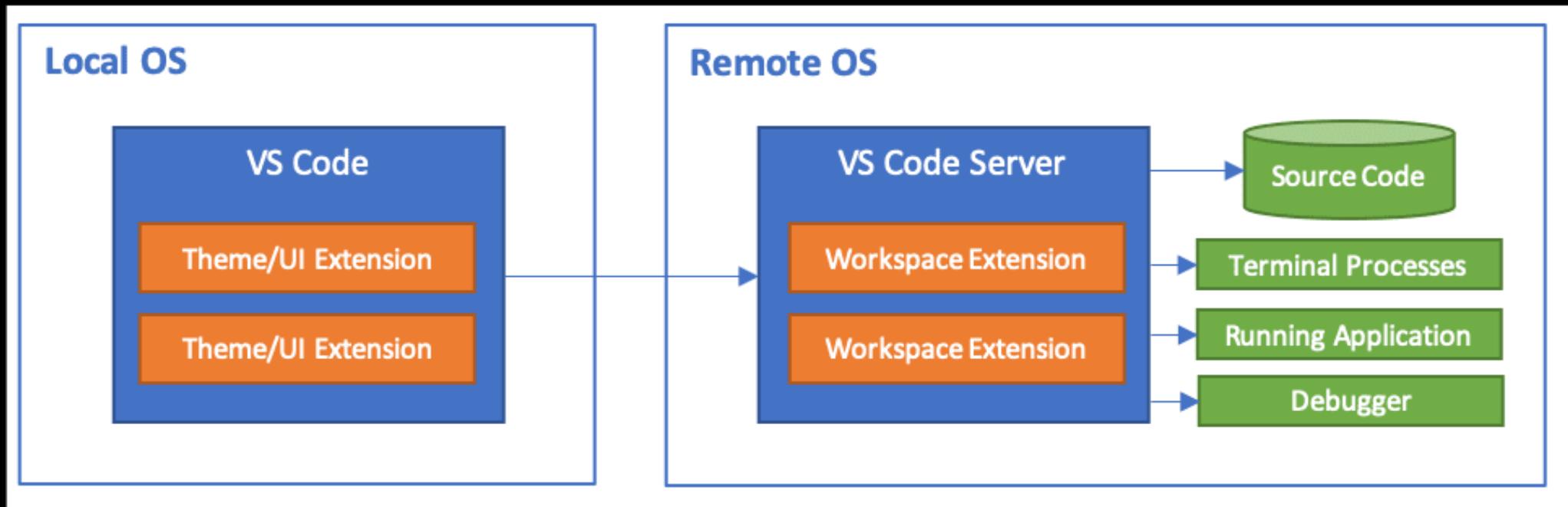
- Preset settings & recommended extensions
- Tasks for calling west to build, flash & debug
- Using Workspaces
- IntelliSense
- <https://github.com/beriberikix/zephyr-vscode-example>



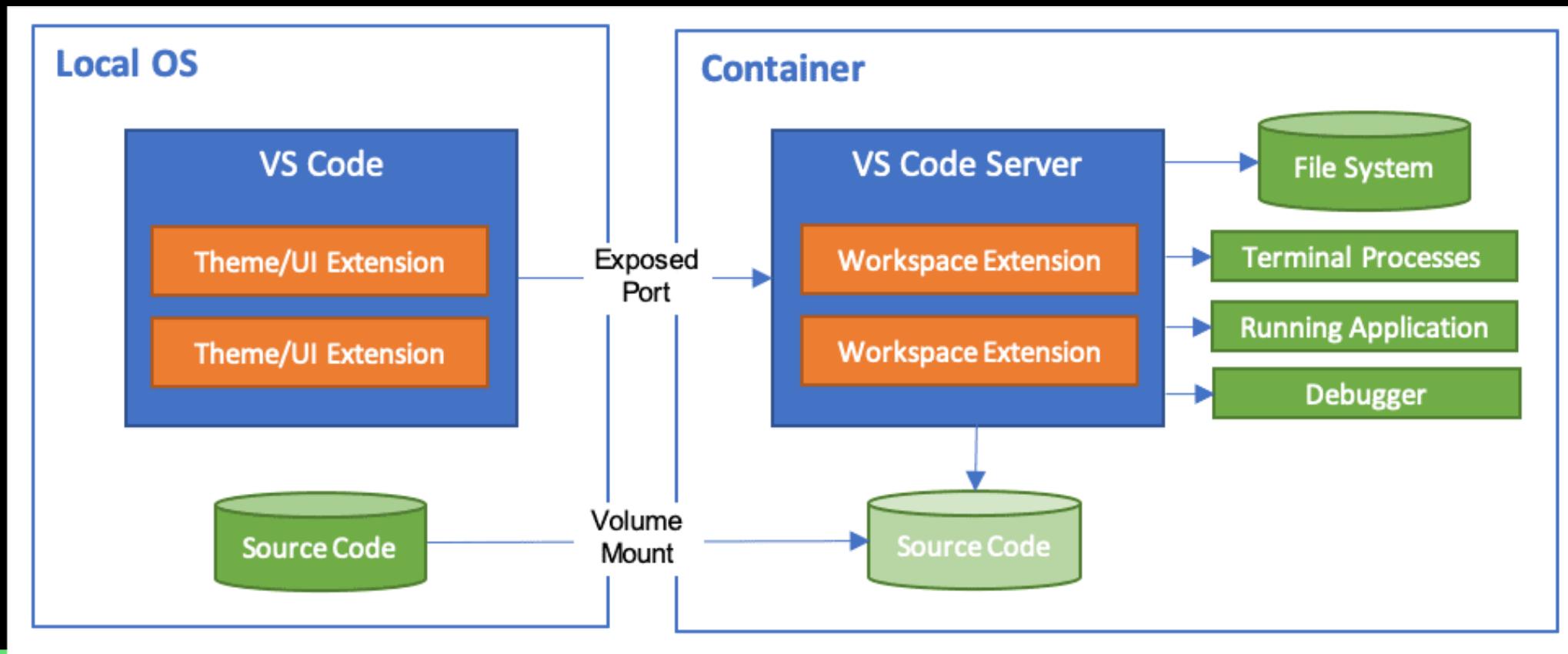
Future Topics



Remote Development



Devcontainers & Docker



GitHub Codespace

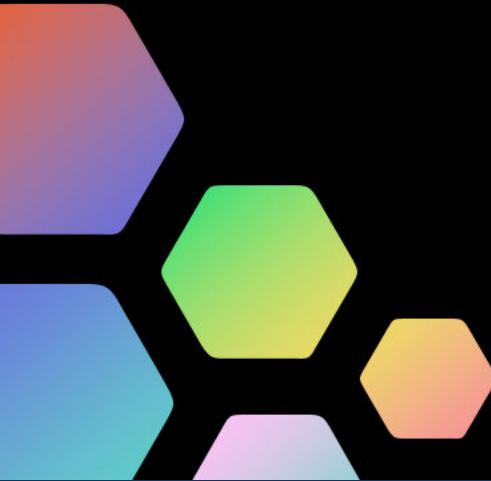


Wrapping Up



Lessons

- VS Code is nearly infinitely configurable
 - Figuring what to configure is half the battle
- You can use it for embedded, today
- Use vendor extensions!



Special Thanks

- zmk.dev community - awesome docs
- Marc Goodner, C++ @ Microsoft



Thank you for attending!

- Come check out the Golioth booth (#41) to see hardware in action!
- Main site: golioth.io
- Golioth Blog: blog.golioth.io
- Golioth Forum: forum.golioth.io
- With the Golioth Dev Tier, your first 50 devices are free!

