

An Editor Can Do That?

Debugging Assembly Language and
GPU Kernels in Visual Studio Code

JULIA REID



20
21



Welcome to CppCon 2021!

Join #visual_studio channel on CppCon Discord
<https://aka.ms/cppcon/discord>

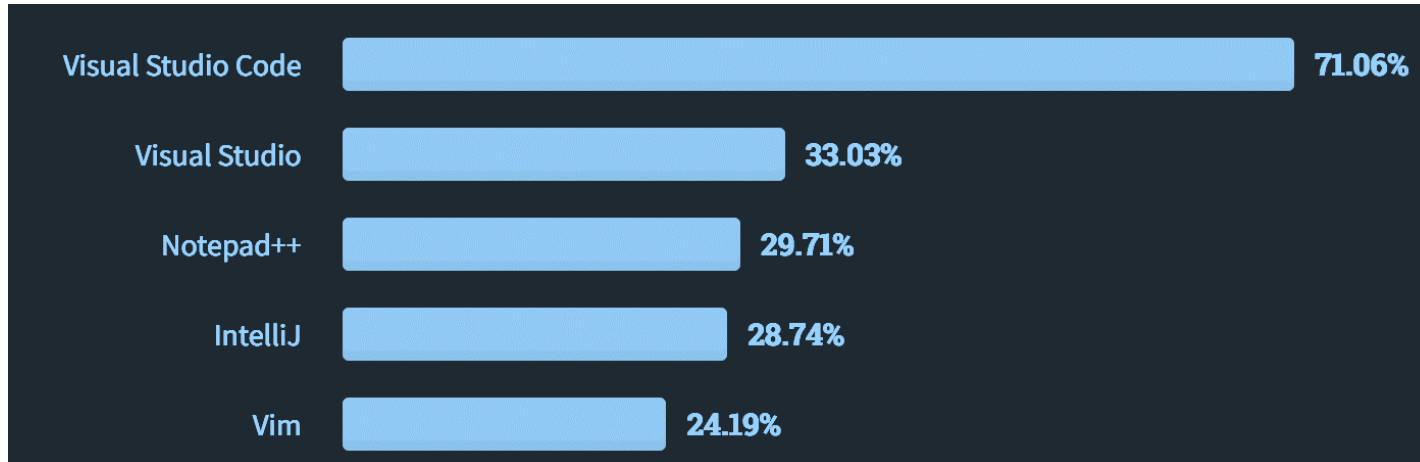
- Meet the Microsoft C++ team
- Ask any questions
- Discuss the latest announcements

Take our survey
<https://aka.ms/cppcon>



Visual Studio Code

#1 most-used code editor

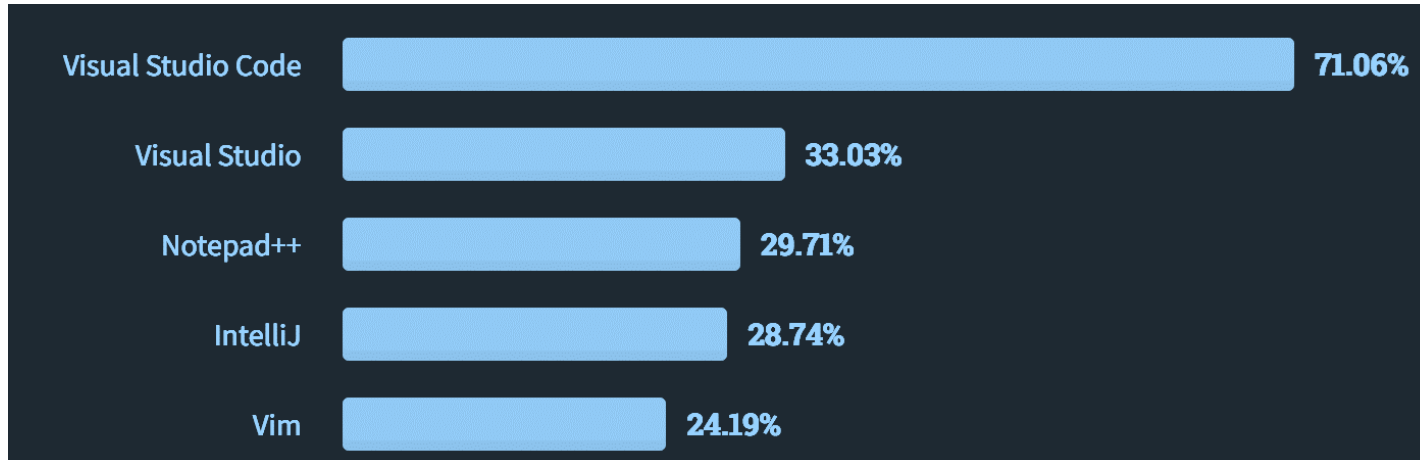


Source: [Stack Overflow Developer Survey 2021](#)



Visual Studio Code

#1 most-used code editor



- Free
- Cross-platform
- Lightweight

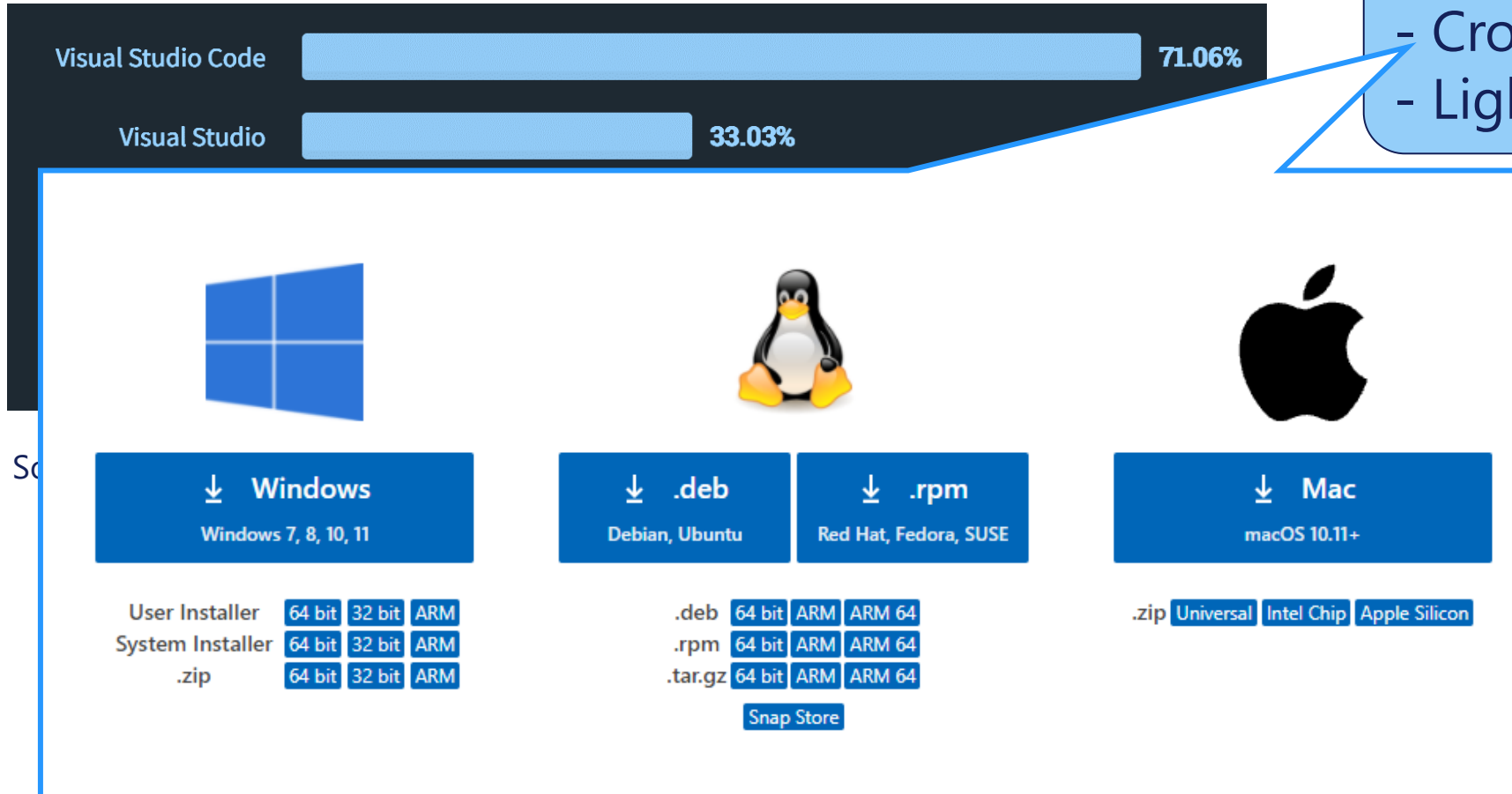
Source: [Stack Overflow Developer Survey 2021](#)



Visual Studio Code

#1 most-used code editor

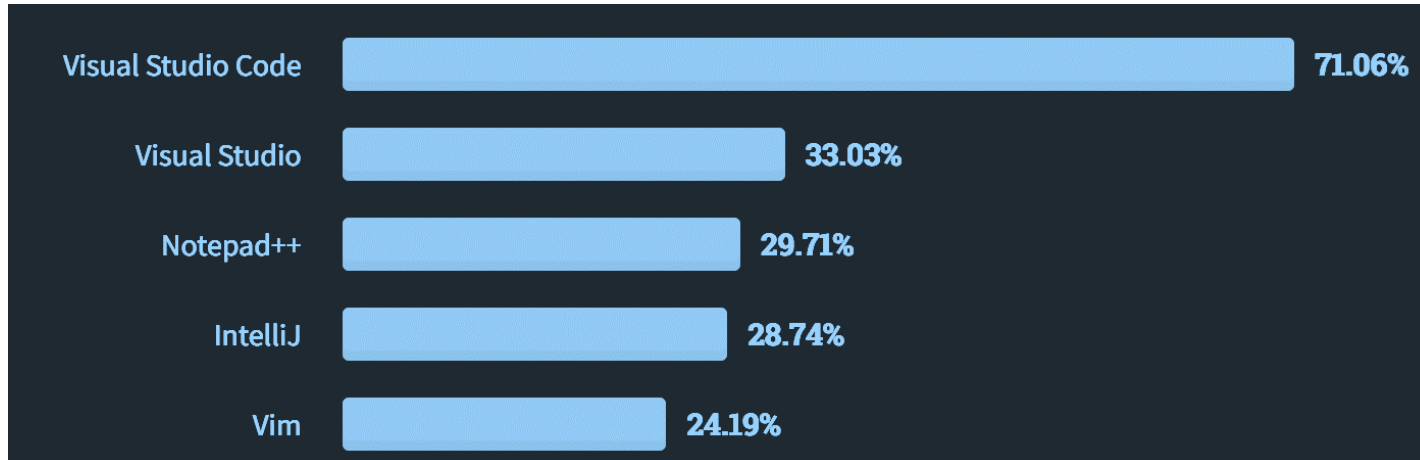
- Free
- Cross-platform
- Lightweight





Visual Studio Code

#1 most-used code editor



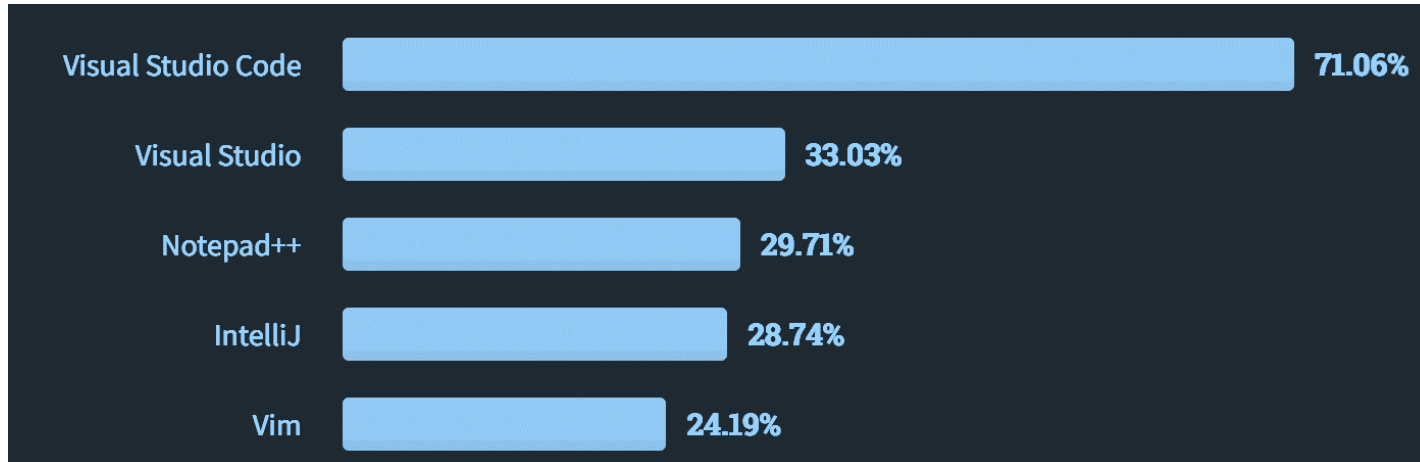
- Free
- Cross-platform
- Lightweight

Source: [Stack Overflow Developer Survey 2021](#)



Visual Studio Code

#1 most-used code editor



Source: [Stack Overflow Developer Survey 2021](#)

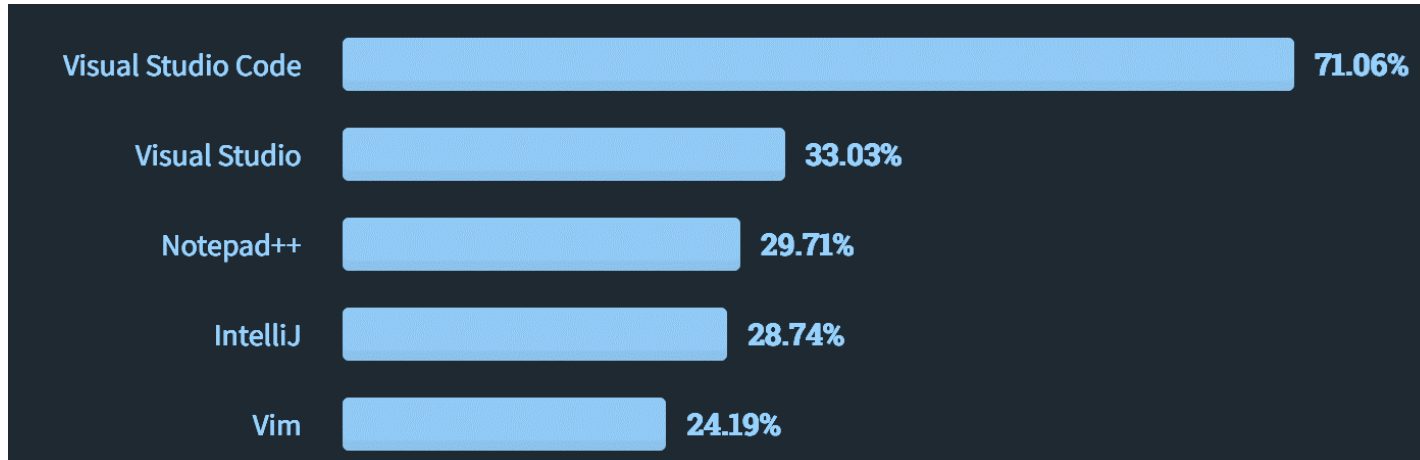
- Free
- Cross-platform
- Lightweight

- WSL
- SSH
- Containers



Visual Studio Code

#1 most-used code editor



Source: [Stack Overflow Developer Survey 2021](#)

- Free
- Cross-platform
- Lightweight

- WSL
- SSH
- Containers

- IntelliSense
- Debugging
- CMake
- Make



Visual Studio Code

What's new?



Visual Studio Code

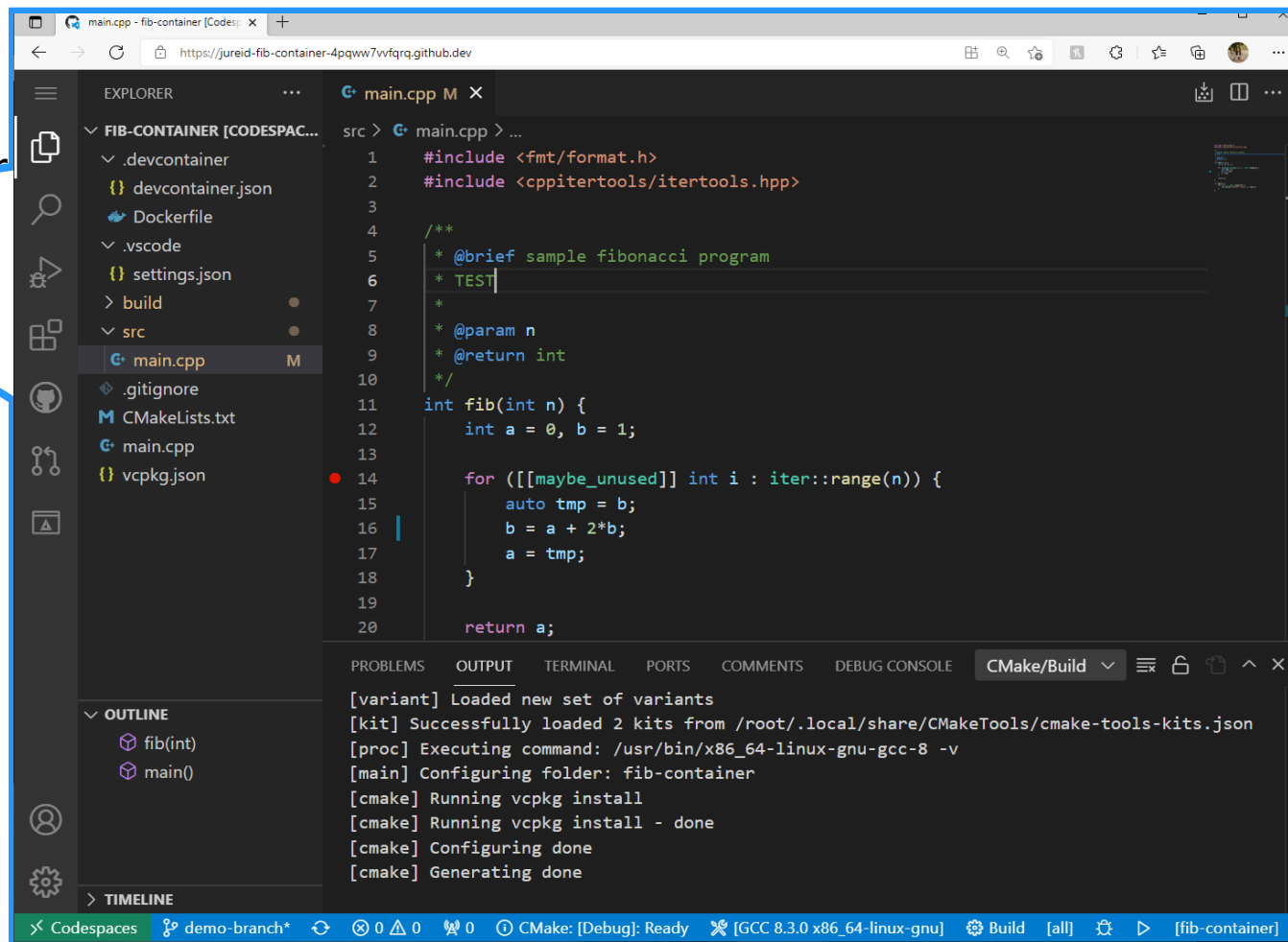
What's new?

1. GitHub Codespaces (coding from your browser!)

Visual Studio Code

What's new?

1. GitHub Codespaces (coding fr

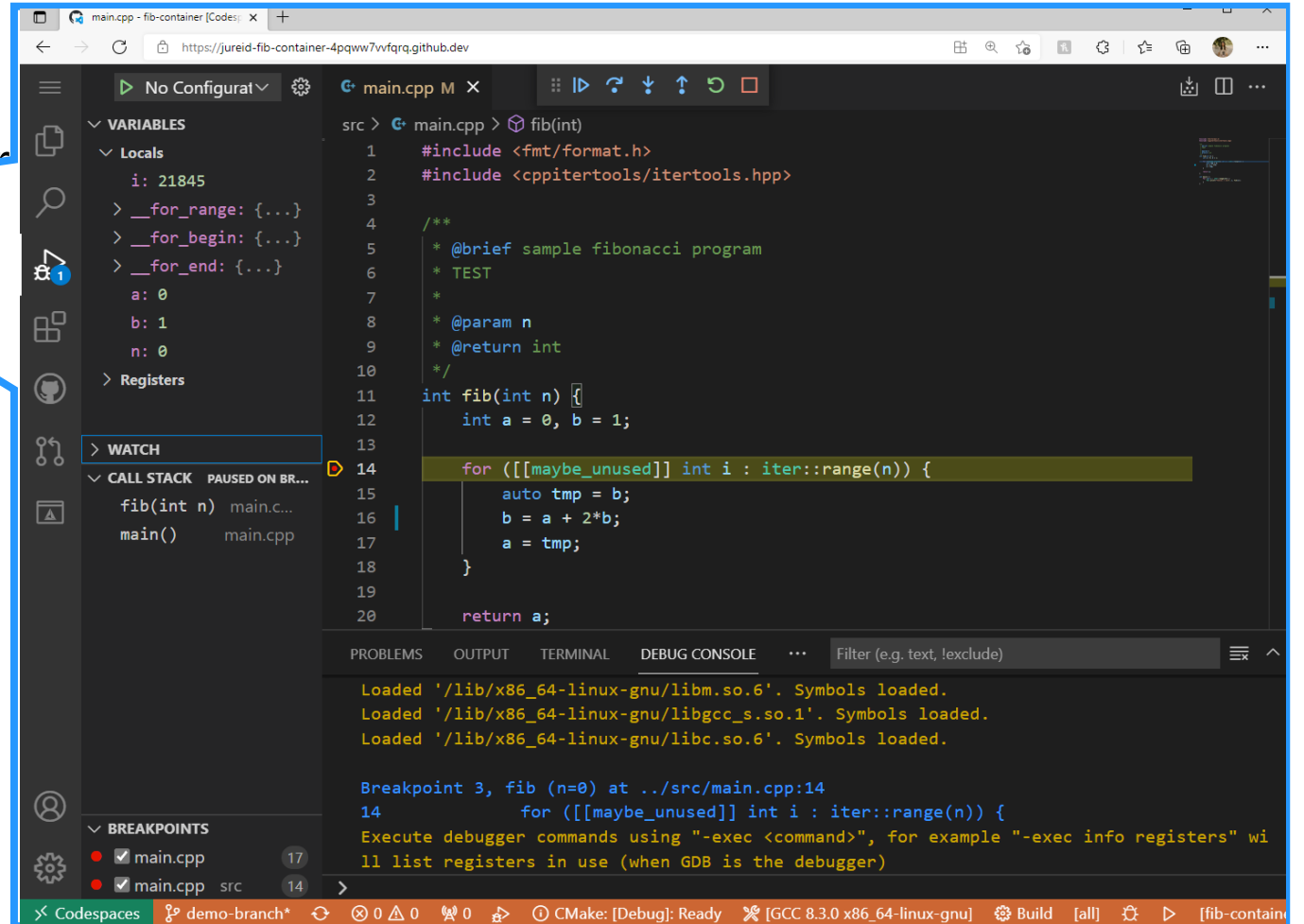




Visual Studio Code

What's new?

1. GitHub Codespaces (coding fr





Visual Studio Code

What's new?

1. GitHub Codespaces (coding from your browser!)



Visual Studio Code

What's new?

1. GitHub Codespaces (coding from your browser!)
2. CMake Presets support



What's new?

1. GitHub Codespaces (coding from your browser!)
2. CMake Presets support
3. ARM and ARM64 support (Raspberry Pi, Surface Pro X, Apple Silicon)



Visual Studio Code

What's new?

1. GitHub Codespaces (coding from your browser!)
2. CMake Presets support
3. ARM and ARM64 support (Raspberry Pi, Surface Pro X, Apple Silicon)
4. CUDA IntelliSense and GPU debugging



Visual Studio Code

What's new?

1. GitHub Codespaces (coding from your browser!)
2. CMake Presets support
3. ARM and ARM64 support (Raspberry Pi, Surface Pro X, Apple Silicon)
4. CUDA IntelliSense and GPU debugging
5. Disassembly View while debugging **Preview!**



Visual Studio Code

What's

1. Git
2. CM
3. AR
4. CU
5. Disassembly View while debugging **Preview!**

microsoft / **vscode-cpptools** Public

Unwatch 209 Star 4.4k Fork

<> Code Issues 925 Pull requests 5 Discussions Actions Projects 3 Wiki Security Insights

[Feature Request] Implement Assembly View/Disassembly View for Debugger #206

Closed ghost opened this issue on Sep 7, 2016 · 51 comments

ghost commented on Sep 7, 2016

Being able to view assembly and step through assembly instructions would be useful. Especially when debug information isn't available for a certain library. Right now it just displays a message saying unknown source, when it goes into one of these functions. Being able to view values of registers in the GUI some way. Not entirely sure how that would work with VSCode, but being able to the registers all the time and able to modify them such other assembly debuggers.

416 36 25 3

Assignees: yuehuang010

Labels: debugger, Feature Request, fixed (release pending)



Visual Studio Code

What's new?

1. GitHub Codespaces (coding from your browser!)
2. CMake Presets support
3. ARM and ARM64 support (Raspberry Pi, Surface Pro X, Apple Silicon)
4. CUDA IntelliSense and GPU debugging
5. Disassembly View while debugging **Preview!**



Visual Studio Code

What's new?

1. GitHub Codespaces (coding from your browser!)
2. CMake Presets support
3. ARM and ARM64 support (Raspberry Pi, Surface Pro X, Apple Silicon)
4. CUDA IntelliSense and GPU debugging
5. Disassembly View while debugging **Preview!**
6. The Makefile Tools extension **Preview!**



Visual Studio Code

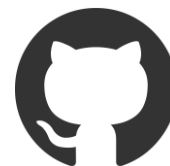
+



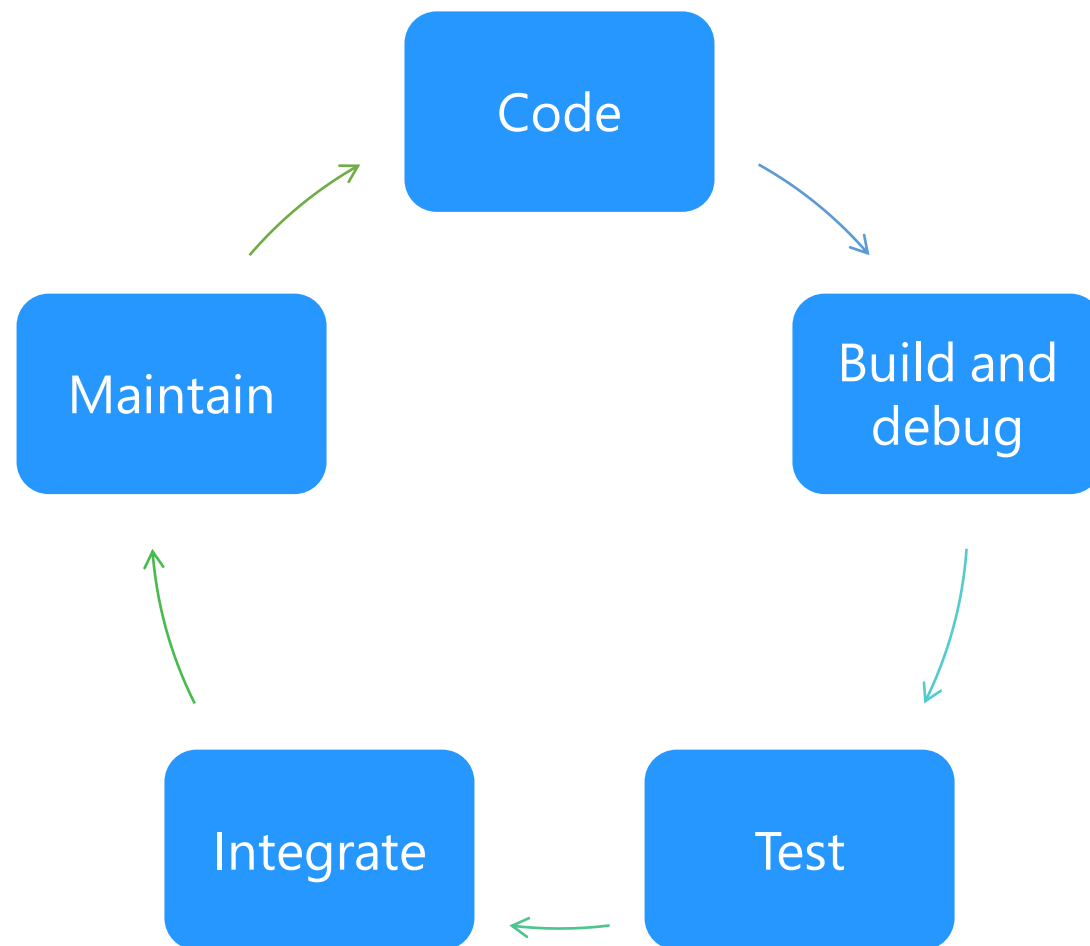
GitHub



Visual Studio Code +



GitHub

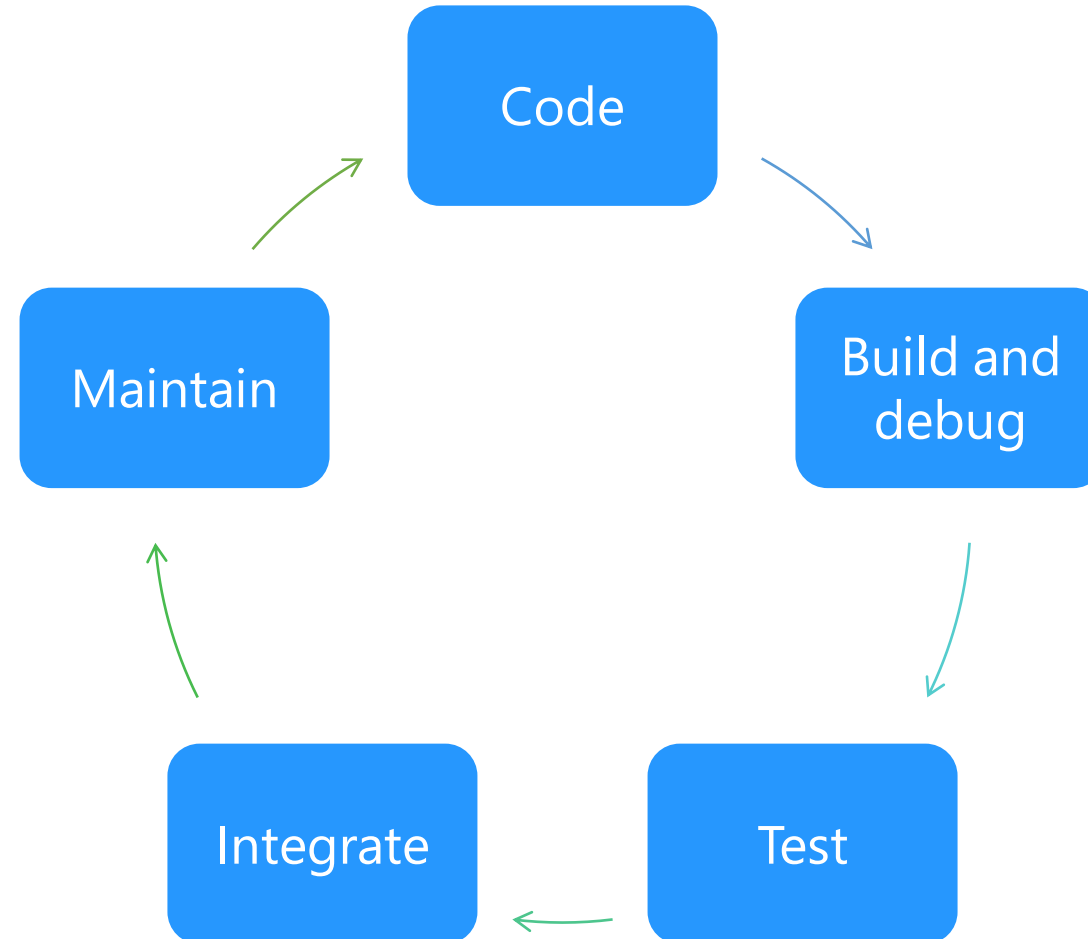
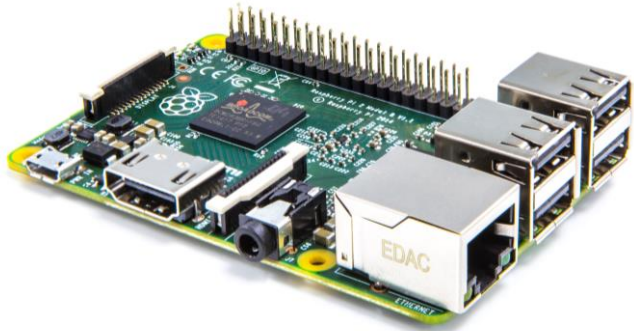




Visual Studio Code +



GitHub

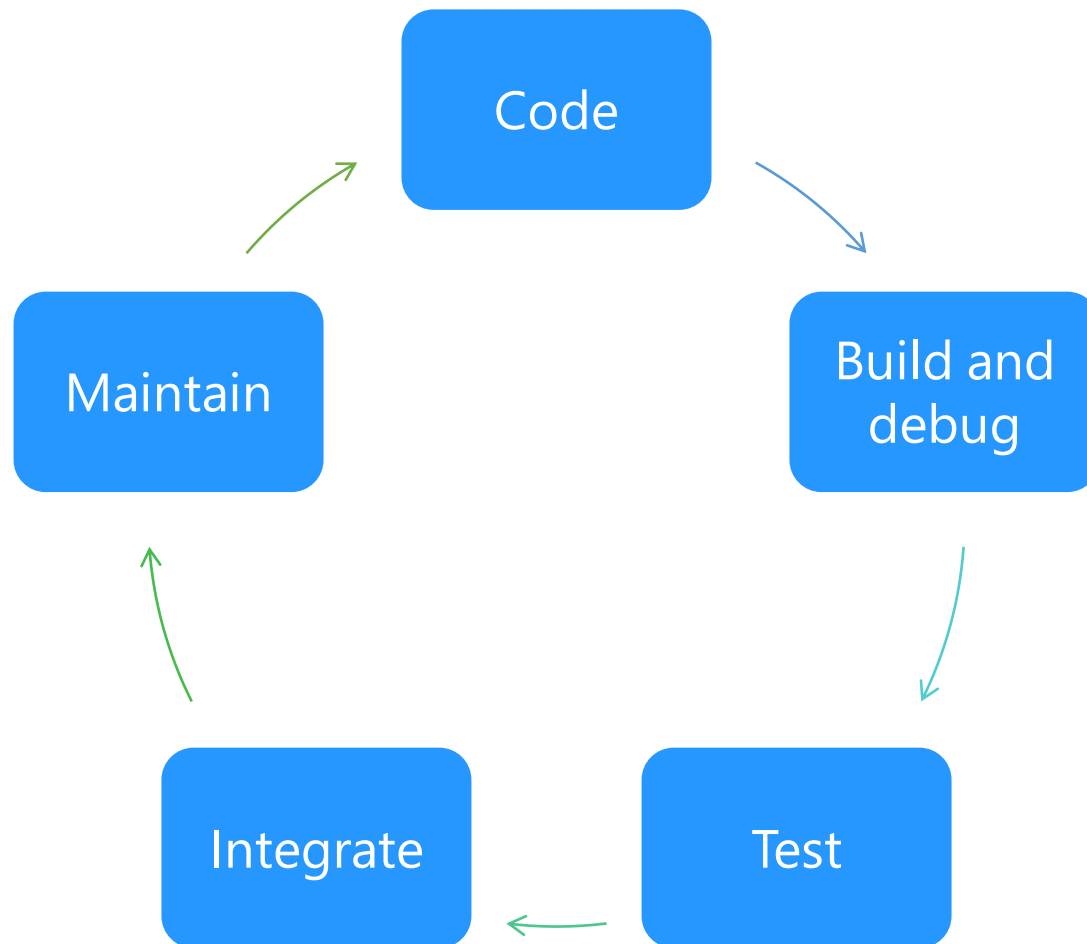
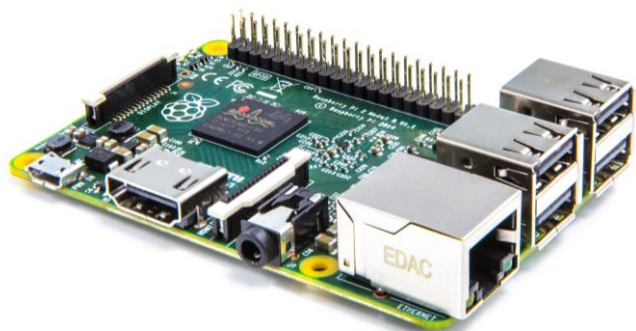




Visual Studio Code +



GitHub



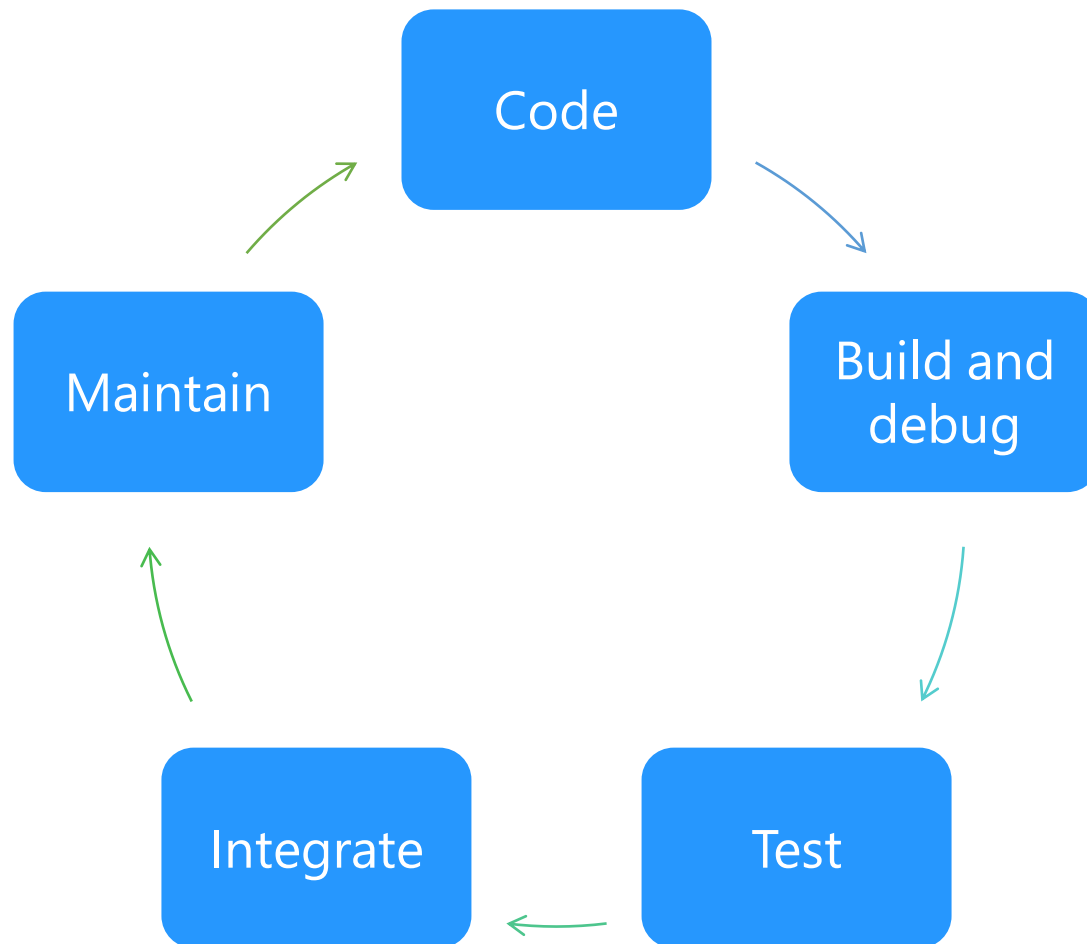
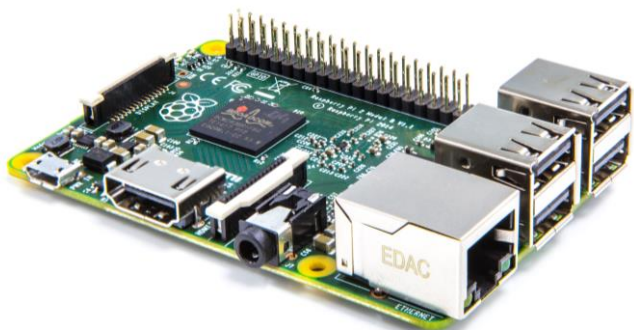


Visual Studio Code

+



GitHub

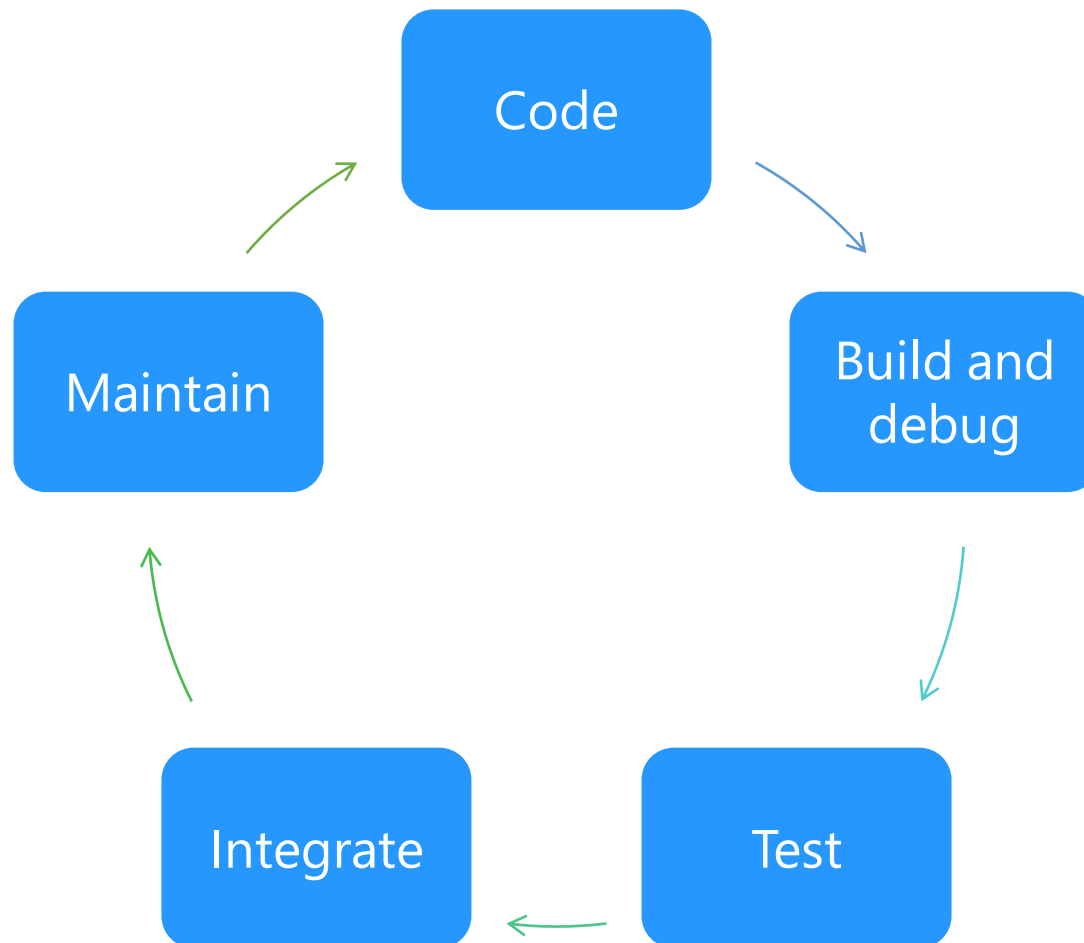
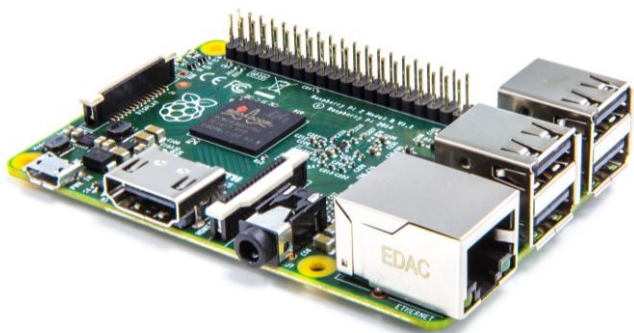




Visual Studio Code +



GitHub



Continuous Integration (CI)

What is Continuous Integration (CI)?

- “The process of automating the build and testing of code every time a team member commits changes to version control” [[What is Continuous Integration? - Azure DevOps | Microsoft Docs](#)]

Continuous Integration (CI)

What is Continuous Integration (CI)?

- “The process of automating the build and testing of code every time a team member commits changes to version control” [[What is Continuous Integration? - Azure DevOps | Microsoft Docs](#)]

What are the benefits?

- Reduces risk – detect and locate errors more quickly
- Ensures quality – shipped product is always tested
- Efficient – spend more time coding, less time manually testing and waiting



GitHub Actions for Continuous Integration (CI)

GitHub repository view for `jureid / blink` (Private). The **Actions** tab is selected, showing 42 workflow runs. The interface includes buttons for **Select workflow** and **New workflow**, and a search bar for **Filter workflow runs**.

Showing runs from all workflows

42 workflow runs

Event	Status	Branch	Actor	Workflow	Run ID	Time	Duration
● Update cmake.yml				delayTime-updates	CI #42: Commit c4df2e8 pushed by jureid	11 minutes ago	Queued
✓ added test job that is dependent on build job				delayTime-updates	CI #41: Commit aff6a20 pushed by jureid	21 hours ago	46s
✗ separated build and test into different steps				delayTime-updates	CI #40: Commit 77f12a8 pushed by jureid	21 hours ago	10m 2s
✓ added comment for numBlinks				delayTime-updates	CI #39: Commit 59ff8d6 pushed by jureid	2 days ago	26s

- **What:** GitHub Actions offers workflows that build and test your code



GitHub Actions for Continuous Integration (CI)

GitHub repository view for `jureid / blink` (Private). The `Actions` tab is selected, showing workflow runs.

Buttons: `Select workflow`, `New workflow`

Showing runs from all workflows

Filter workflow runs

42 workflow runs

Event	Status	Branch	Actor
● Update cmake.yml	delayTime-updates		11 minutes ago ...
CI #42: Commit c4df2e8 pushed by jureid			
✓ added test job that is dependent on build job	delayTime-updates		21 hours ago ...
CI #41: Commit aff6a20 pushed by jureid			
✗ separated build and test into different steps	delayTime-updates		21 hours ago ...
CI #40: Commit 77f12a8 pushed by jureid			
✓ added comment for numBlinks	delayTime-updates		2 days ago ...
CI #39: Commit 59ff8d6 pushed by jureid			

- **What:** GitHub Actions offers workflows that build and test your code
- **When:** Workflows run when a specified GitHub event occurs



GitHub Actions for Continuous Integration (CI)

Showing runs from all workflows

Filter workflow runs

42 workflow runs

Event	Status	Branch	Actor
Update cmake.yml	delayTime-updates		11 minutes ago ... Queued
added test job that is dependent on build job	delayTime-updates		21 hours ago ... 46s
separated build and test into different steps	delayTime-updates		21 hours ago ... 10m 2s
added comment for numBlinks	delayTime-updates		2 days ago ... 26s

- **What:** GitHub Actions offers workflows that build and test your code
- **When:** Workflows run when a specified GitHub event occurs
- **Where:** Workflows run on GitHub-hosted virtual machines or self-hosted machines



GitHub Actions for Continuous Integration (CI)

Showing runs from all workflows

42 workflow runs

Event	Status	Branch	Actor
Update cmake.yml	delayTime-updates	11 minutes ago	Queued
added test job that is dependent on build job	delayTime-updates	21 hours ago	46s
separated build and test into different steps	delayTime-updates	21 hours ago	10m 2s
added comment for numBlinks	delayTime-updates	2 days ago	26s

- **What:** GitHub Actions offers workflows that build and test your code
- **When:** Workflows run when a specified GitHub event occurs
- **Where:** Workflows run on GitHub-hosted virtual machines or self-hosted machines
- **How:** Workflows are defined in .yaml files in your project's .github/workflows folder



GitHub Actions for Continuous Integration (CI)

jureid / blink Private

Unwatch 1 Star 0 Fork 0

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

Choose a workflow template

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow template to get started.

Skip this and [set up a workflow yourself](#) →

Workflows made for your repository Suggested

C/C++ with Make

By GitHub Actions

Build and test a C/C++ project using Make.

[Set up this workflow](#)

```
./configure
make
make check
```

actions/starter-workflows C ●

CMake based projects

By GitHub Actions

Build and test a CMake based project.

[Set up this workflow](#)

```
cmake -B ${github.workspace}/build -
DCMAKE_BUILD_TYPE=${env.BUILD_TYPE}
cmake --build ${github.workspace}/build --config
${env.BUILD_TYPE}
ctest -C ${env.BUILD_TYPE}
```

actions/starter-workflows C ●

EXPLORER

OPEN EDITORS

.github/workflows

cmake.yml

.vscode

out

tests

.gitignore

blink.c

CMakeLists.txt

CMakePresets.json

README.md

cmake.yml

```
.github > workflows > ! cmake.yml
1  # Build and test a CMake Blink project on a self-hosted
2  name: CI
3
4  # Run CI on every push
5  on: push
6
7  jobs:
8    build:
9      # The CMake configure and build commands are platform
10     # well on Windows or Mac. You can convert this to
11     # cross-platform coverage.
12     # See: https://docs.github.com/en/free-pro-team@latest
13     runs-on: self-hosted
14
15
16     # Checks-out your repository under $GITHUB_WORKSPACE
17     steps:
18       - uses: actions/checkout@v2
19
20     # Configure and build on Raspberry Pi with Ninja and
21     - name: 'Configure and build on Raspberry Pi'
22       run: |
23         cmake --preset=raspi-debug
24         cmake --build --preset verbose-build-raspi
25
26     # Run CTest on Raspberry Pi
27     - name: 'Run CTest on Raspberry Pi'
28       shell: 'bash'
29       run: 'ctest --preset core-test-raspi'
30
```



GitHub Actions for Continuous Integration (CI)

Microsoft C++ Code Analysis with GitHub Actions

NEW!



Daniel

October 26th, 2021

We previously talked about [GitHub Code Scanning capabilities](#) which enabled developers to incorporate security checks into their CI/CD environment and developer workflow. [CodeQL](#) is the default analysis engine behind Code Scanning. Today we are introducing support for MSVC Code Analysis which will provide a great companion to CodeQL for C++ GitHub repos with Windows workflows.

⚠ Check warning on line 26 in CppConDemo/CppConDemo.cpp

🔍 Code scanning

Dereferencing NULL pointer 'pContext'. ⚠ Warning

Dereferencing NULL pointer 'pContext'.

[Show more details](#)

Show paths

Dismiss ▾

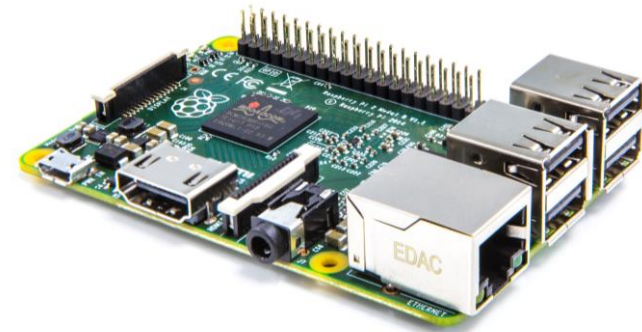
Static Analysis and Program
Safety in C++: Making it Real

Sunny Chatterjee

View Recording ↗

Demo #1

Achieving efficient CI workflows for Raspberry Pi development with GitHub Actions & GitHub Codespaces (VS Code in the browser!)






Demo #1 set up

Extensions for VS Code:

- github.vscode-pull-request-github




GitHub Pull Requests and Issues Preview

GitHub | 📦 3,554,510 installs | ★★★★★ (81) | Free

Pull Request and Issue Provider for GitHub

[Install](#) [Trouble Installing?](#)

- github.codespaces




GitHub Codespaces

GitHub | 📦 634,616 installs | ★★★★★ (6) | Free

Your instant dev environment

[Install](#) [Trouble Installing?](#)

- ms-vscode.cpptools



C/C++

Microsoft | 📦 24,527,317 installs | ★★★★★ (451) | Free

C/C++ IntelliSense, debugging, and code browsing.

[Install](#) [Trouble Installing?](#)



Demo #1 key takeaways

1. **GitHub.dev** and **GitHub Codespaces** enable you to **write and commit code directly from your web browser** on any device



Demo #1 key takeaways

1. **GitHub.dev** and **GitHub Codespaces** enable you to **write and commit code directly from your web browser** on any device
 - ✓ No need to install an editor or clone the repo!



Demo #1 key takeaways

1. **GitHub.dev** and **GitHub Codespaces** enable you to **write and commit code directly from your web browser** on any device
 - ✓ No need to install an editor or clone the repo!
2. **GitHub Actions** makes it easy to **build and test your code in CI/CD workflows**



Demo #1 key takeaways

1. **GitHub.dev** and **GitHub Codespaces** enable you to **write and commit code directly from your web browser** on any device
 - ✓ No need to install an editor or clone the repo!
2. **GitHub Actions** makes it easy to **build and test your code in CI/CD workflows**
 - ✓ GitHub-hosted runners for automatic machine upgrades and zero maintenance
 - ✓ Self-hosted runners for more control over hardware and OS

CMakePresets.json

1. Released by Kitware in **CMake 3.19**
 - ✓ 3.21 or higher required for CMakePresets.json v3



CMakePresets.json

1. Released by Kitware in **CMake 3.19**
 - ✓ 3.21 or higher required for CMakePresets.json v3
2. Allows users to specify common **configure, build, and test options** and **share** them **with others**



CMakePresets.json

1. Released by Kitware in **CMake 3.19**
 - ✓ 3.21 or higher required for CMakePresets.json v3
2. Allows users to specify common **configure, build, and test options** and **share** them **with others**
3. Lives at the root of the project, **intended to be checked in to source control**
 - ✓ CMakeUserPresets.json intended for developers to save their own local builds



Example configurePreset

```
{  
    "name": "raspi-debug",  
    "displayName": "Raspberry Pi Debug",  
    "description": "Sets debug build type",  
    "inherits": "base",  
    "cacheVariables": {  
        "CMAKE_BUILD_TYPE": "Debug"  
    }  
}
```



Example configurePreset

```
{  
    "name": "raspi-debug",  
    "displayName": "Raspberry Pi Debug",  
    "description": "Sets debug build type",  
    "inherits": "base",  
    "cacheVariables": {  
        "CMAKE_BUILD_TYPE": "Debug"  
    }  
}
```



Example configurePreset

```
{  
    "name": "raspi-debug",  
    "displayName": "Raspberry Pi Debug",  
    "description": "Sets debug build type",  
    "inherits": "base",  
    "cacheVariables": {  
        "CMAKE_BUILD_TYPE": "Debug"  
    }  
}
```



Example base configurePreset

```
{  
  "name": "base",  
  "description": "For more information: http://aka.ms/cmakepresetsvs",  
  "hidden": true,  
  "generator": "Ninja",  
  "binaryDir": "${sourceDir}/out/build/${presetName}",  
  "installDir": "${sourceDir}/out/install/${presetName}",  
  "cacheVariables": {  
    "CMAKE_C_COMPILER": "gcc",  
    "CMAKE_CXX_COMPILER": "g++"  
  },  
  "environment": {  
    "VCPKG_FEATURE_FLAGS": "manifests,versions,binarycaching,registries"  
  },  
  "condition": {  
    "type": "equals",  
    "lhs": "${hostSystemName}",  
    "rhs": "Linux"  
  }  
}
```



Example buildPreset

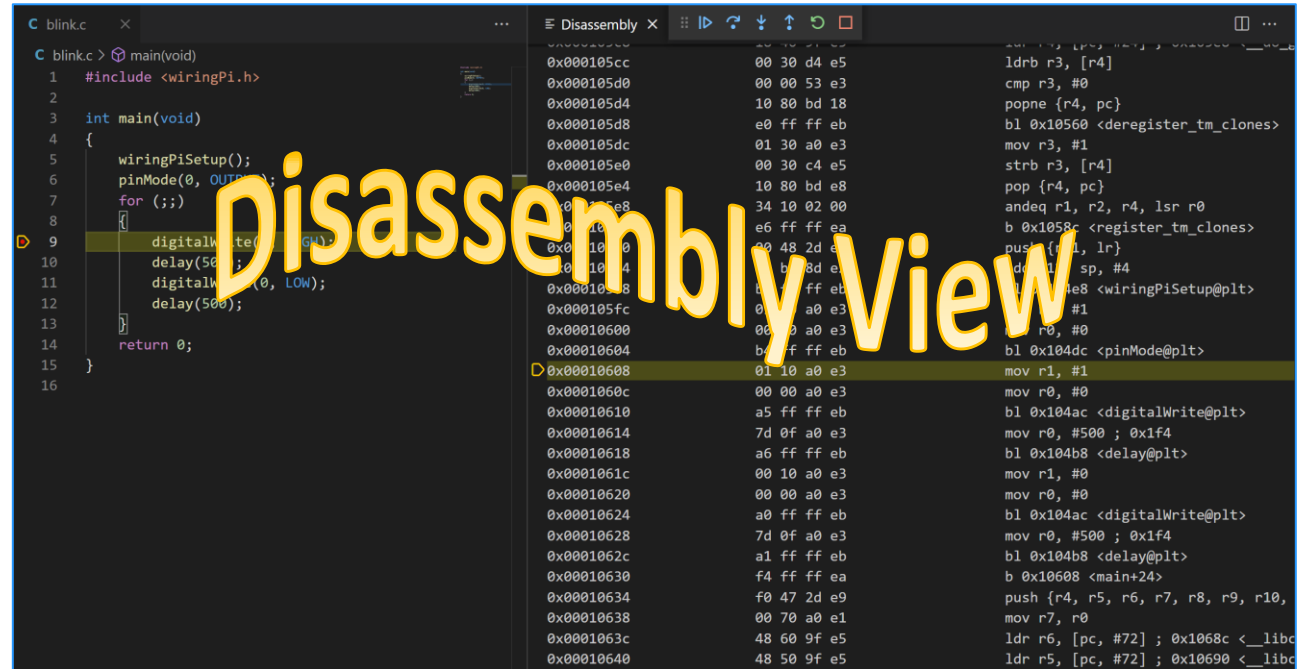
```
{  
    "name": "verbose-build-raspi",  
    "displayName": "Verbose Build",  
    "description": "Passes -v to Ninja",  
    "configurePreset": "raspi-debug",  
    "nativeToolOptions": [ "-v" ]  
}
```


Demo #2

Build and debug on a Raspberry Pi in VS Code Desktop with CMakePresets.json

Demo #2

Build and debug on a Raspberry Pi in VS Code Desktop with CMakePresets.json



The screenshot displays the Visual Studio Code interface with two panels. The left panel shows a C source file named `blink.c` with the following code:

```
1 #include <wiringPi.h>
2
3 int main(void)
4 {
5     wiringPiSetup();
6     pinMode(0, OUTPUT);
7     for (;;)
8     {
9         digitalWrite(0, HIGH);
10        delay(500);
11        digitalWrite(0, LOW);
12        delay(500);
13    }
14    return 0;
15 }
```

The right panel shows the disassembly of the code, with the following instructions visible:

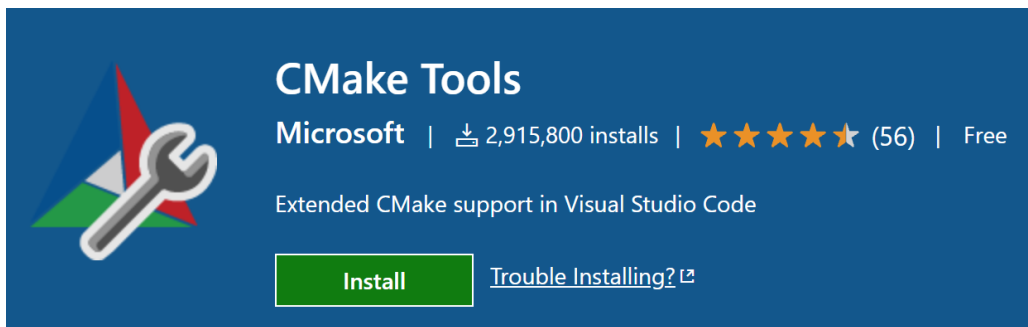
```
0x000105cc 00 30 d4 e5 ldrb r3, [r4]
0x000105d0 00 00 53 e3 cmp r3, #0
0x000105d4 10 80 bd 18 popne {r4, pc}
0x000105d8 e0 ff ff eb bl 0x10560 <derefer_tm_clones>
0x000105dc 01 30 a0 e3 mov r3, #1
0x000105e0 00 30 c4 e5 strb r3, [r4]
0x000105e4 10 80 bd e8 pop {r4, pc}
0x000105e8 34 10 02 00 andeq r1, r2, r4, lsr r0
0x000105ec e6 ff ff ea b 0x1058c <register_tm_clones>
0x000105f0 00 48 2d e1 push {r1, lr}
0x000105f4 00 00 00 00 dcl sp, #4
0x000105f8 00 00 00 00 ldr r0, [0x105f8 <wiringPiSetup@plt>]
0x000105fc 00 00 00 00 mov r0, #1
0x00010600 00 00 00 00 mov r0, #0
0x00010604 b0 ff ff eb bl 0x104dc <pinMode@plt>
0x00010608 01 10 a0 e3 mov r1, #1
0x0001060c 00 00 a0 e3 mov r0, #0
0x00010610 a5 ff ff eb bl 0x104ac <digitalWrite@plt>
0x00010614 7d 0f a0 e3 mov r0, #500 ; 0x1f4
0x00010618 a6 ff ff eb bl 0x104b8 <delay@plt>
0x0001061c 00 10 a0 e3 mov r1, #0
0x00010620 00 00 a0 e3 mov r0, #0
0x00010624 a0 ff ff eb bl 0x104ac <digitalWrite@plt>
0x00010628 7d 0f a0 e3 mov r0, #500 ; 0x1f4
0x0001062c a1 ff ff eb bl 0x104b8 <delay@plt>
0x00010630 f4 ff ff ea b 0x10608 <main+24>
0x00010634 f0 47 2d e9 push {r4, r5, r6, r7, r8, r9, r10,
0x00010638 00 70 a0 e1 mov r7, r0
0x0001063c 48 60 9f e5 ldr r6, [pc, #72] ; 0x1068c <__libc
0x00010640 48 50 9f e5 ldr r5, [pc, #72] ; 0x10690 <__libc
```



Demo #2 set up

Extensions for VS Code:

- ms-vscode.cmake-tools

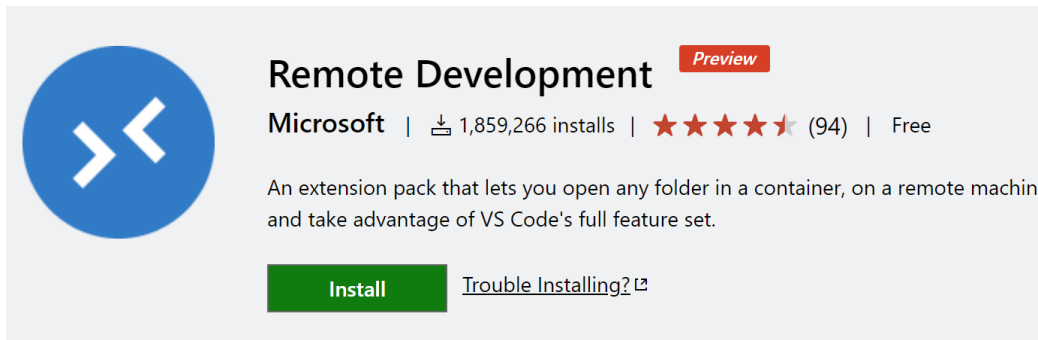


CMake Tools
Microsoft | 📄 2,915,800 installs | ★★★★★ (56) | Free

Extended CMake support in Visual Studio Code

[Install](#) [Trouble Installing?](#)

- ms-vscode-remote.vscode-remote-extensionpack

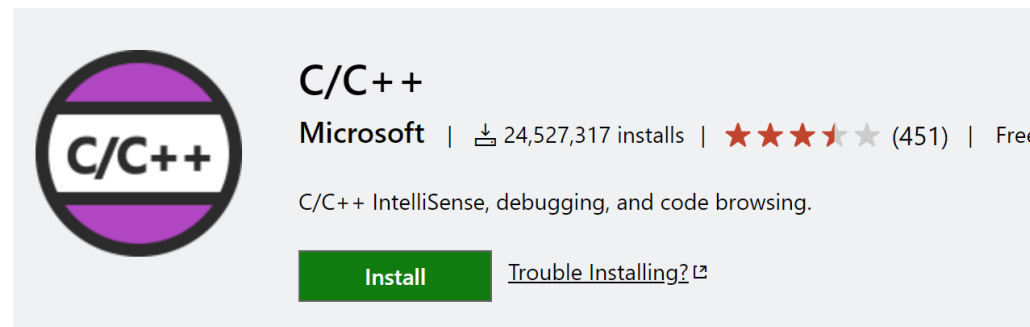


Remote Development Preview
Microsoft | 📄 1,859,266 installs | ★★★★★ (94) | Free

An extension pack that lets you open any folder in a container, on a remote machine and take advantage of VS Code's full feature set.

[Install](#) [Trouble Installing?](#)

- ms-vscode.cpptools



C/C++
Microsoft | 📄 24,527,317 installs | ★★★★★ (451) | Free

C/C++ IntelliSense, debugging, and code browsing.

[Install](#) [Trouble Installing?](#)



Demo #2 key takeaways

1. **VS Code** provides full C++ [IntelliSense and debug support for Raspberry Pi](#)
 1. Disassembly View to debug assembly language



Demo #2 key takeaways

1. **VS Code** provides full C++ [IntelliSense and debug support for Raspberry Pi](#)
 1. Disassembly View to debug assembly language
2. Develop [on and for](#) Raspberry Pi with VS Code by either:
 1. Installing VS Code on the Raspberry Pi
 2. Installing VS Code on a laptop and using the **Remote-SSH extension**



Demo #2 key takeaways

1. **VS Code** provides full C++ [IntelliSense and debug support for Raspberry Pi](#)
 1. Disassembly View to debug assembly language
2. Develop [on and for](#) Raspberry Pi with VS Code by either:
 1. Installing VS Code on the Raspberry Pi
 2. Installing VS Code on a laptop and using the **Remote-SSH extension**
3. **CMakePresets.json** enables [consistent builds](#) from the command line, in CI/CD pipelines, from Visual Studio, and VS Code



Visual Studio Code

+



GitHub

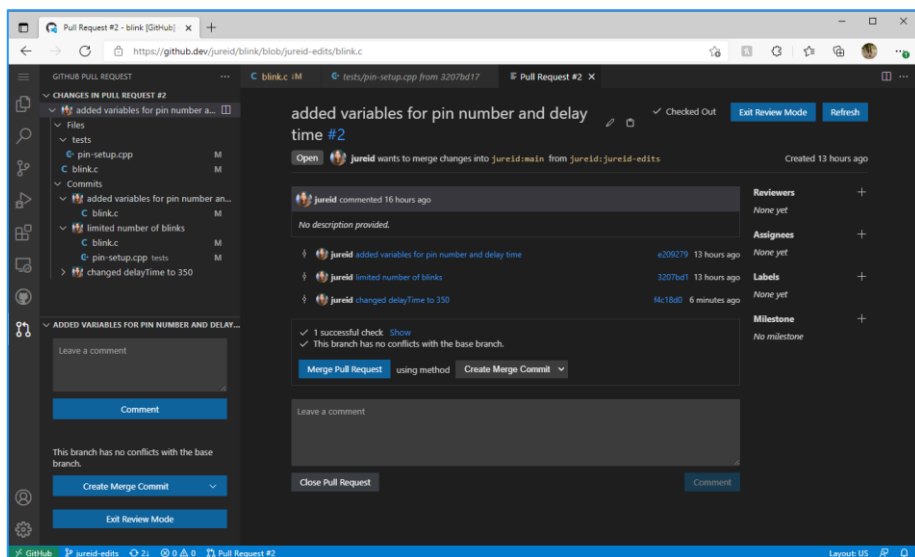


Visual Studio Code

+



GitHub



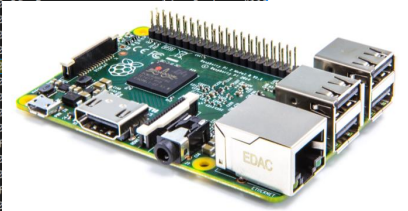
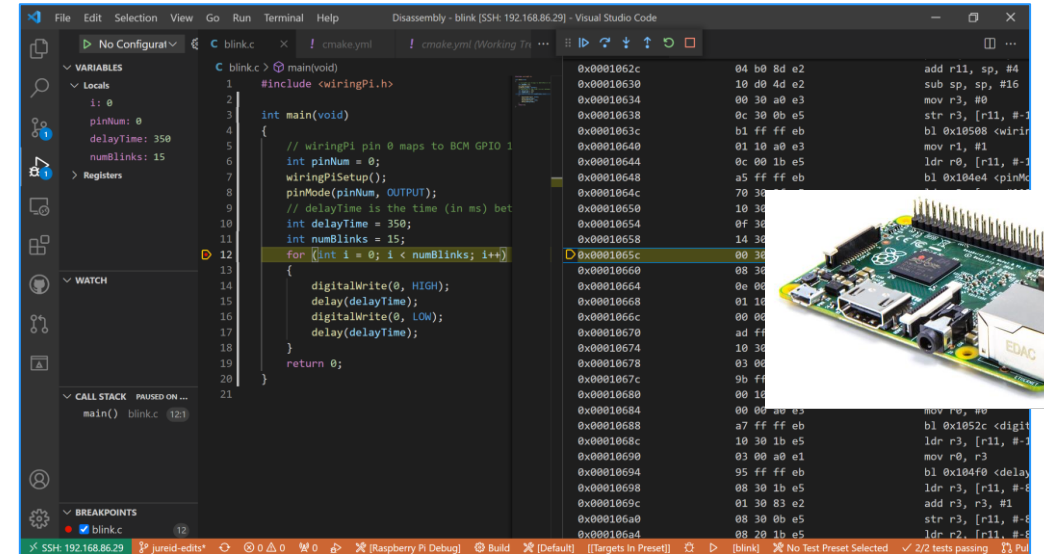
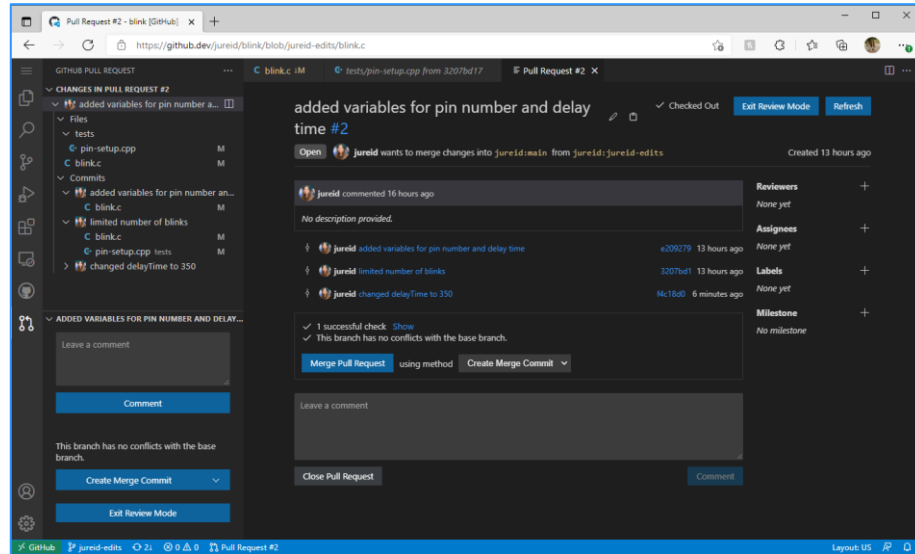


Visual Studio Code

+

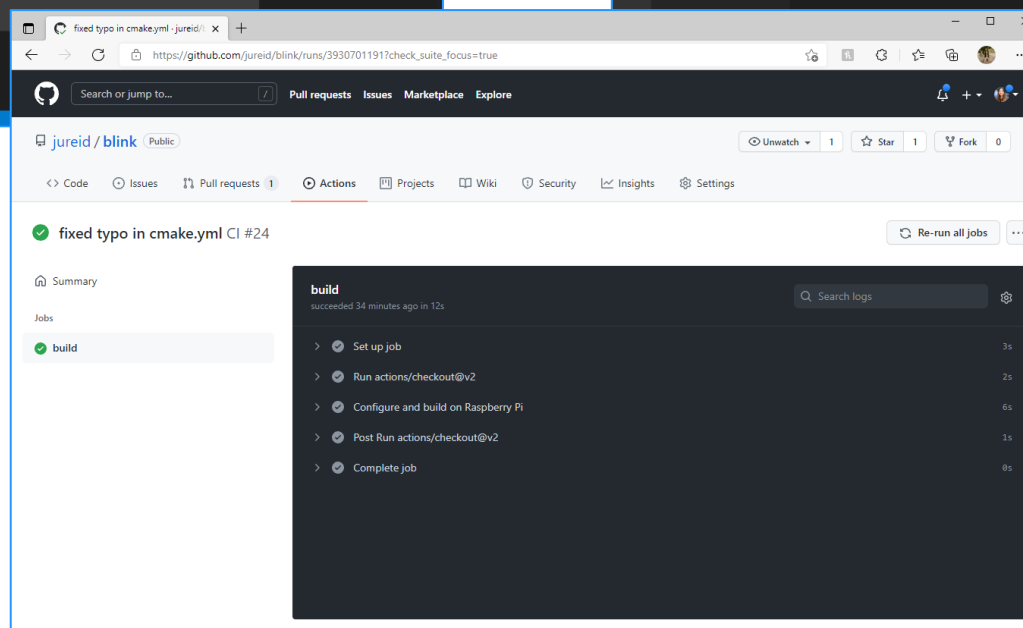
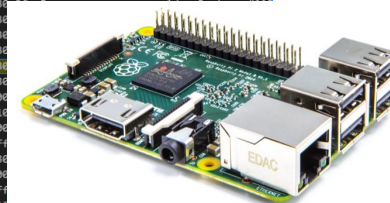
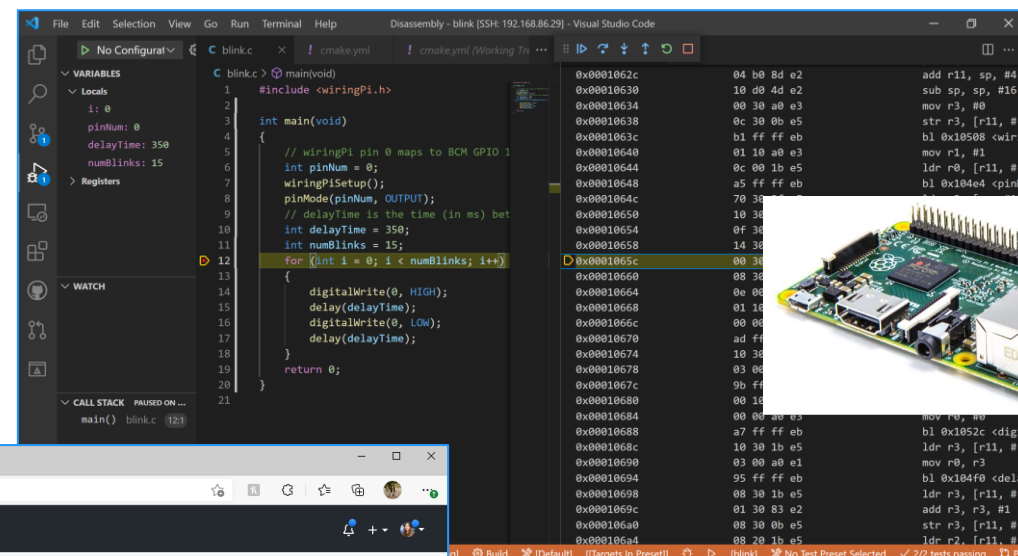


GitHub





GitHub

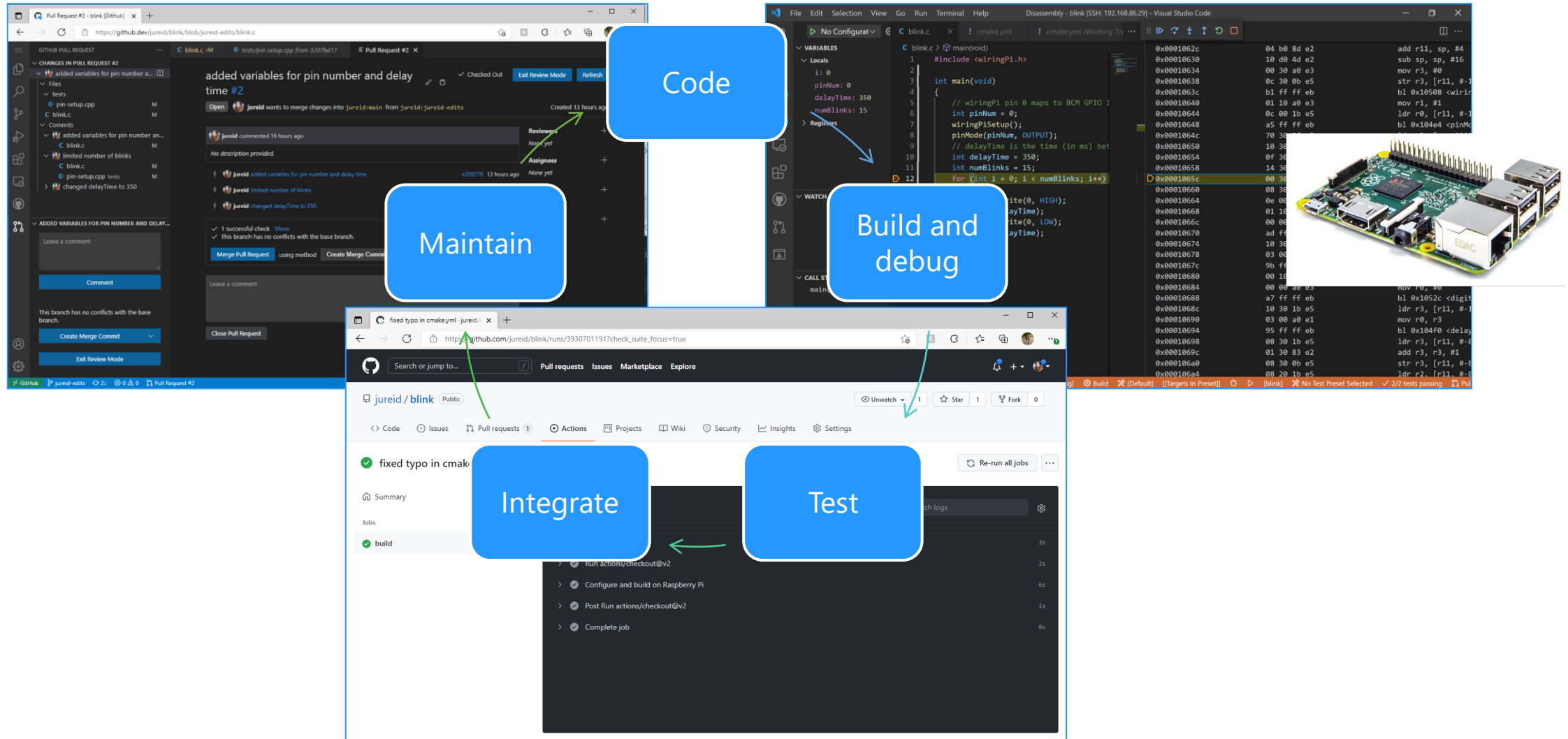




Visual Studio Code + GitHub



GitHub





Demo #3

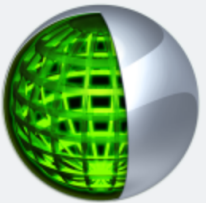
Build and debug CUDA C/C++ programs with Nsight Visual Studio Code Edition



Demo #3 set up

Extensions for VS Code:

- nvidia.nsight-vscode-edition




Nsight Visual Studio Code Edition
NVIDIA | 📄 10,186 installs | ★★★★★ (5) | Free

CUDA development and debugging support for VS Code

[Install](#) [Trouble Installing?](#)

- ms-vscode.cpptools




C/C++
Microsoft | 📄 24,527,317 installs | ★★★★★ (451) | Free

C/C++ IntelliSense, debugging, and code browsing.

[Install](#) [Trouble Installing?](#)

- ms-vscode-remote.vscode-remote-extensionpack



Remote Development Preview
Microsoft | 📄 1,859,266 installs | ★★★★★ (94) | Free

An extension pack that lets you open any folder in a container, on a remote machine and take advantage of VS Code's full feature set.

[Install](#) [Trouble Installing?](#)



Demo #3 key takeaways

1. **The C++ extension** provides [IntelliSense](#) for [CUDA C/C++](#) programs
2. **Nsight Visual Studio Code Edition** provides [build and debug](#) support for [CUDA C/C++](#) programs, including [GPU debugging](#)



Demo #3 key takeaways

1. **The C++ extension** provides **IntelliSense** for **CUDA C/C++** programs
2. **Nsight Visual Studio Code Edition** provides **build and debug** support for **CUDA C/C++** programs, including **GPU debugging**
3. Target a machine with a **CUDA-capable GPU** with the **Remote-SSH extension**



Visual Studio Code

What else?



What else?

1. The Makefile Tools extension [Preview!](#)



Visual Studio Code

What else?

1. The Makefile Tools extension **Preview!**
2. Clang-tidy integration **Coming soon!**



What else?

1. The Makefile Tools extension **Preview!**
2. Clang-tidy integration **Coming soon!**
3. Create definition from declaration (and vice-a-versa) **Coming soon!**

Helpful resources

- **C++ *extension pack*:** [C/C++ Extension Pack - Visual Studio Marketplace](#)
 - C++ extension (ms-vscode.cpptools)
 - CMake Tools (ms-vscode.cmake-tools)
 - Remote Development extension pack (ms-vscode-remote.vscode-remote-extensionpack)
 - And more!
- **C++ *Team blog*:** [C++ Team Blog \(microsoft.com\)](#)
- ***Getting Started with C++ in VS Code*:** [Introductory Videos for C++ in Visual Studio Code](#)
- ***CUDA support in VS Code*:**
 - [Nsight Visual Studio Code Edition Homepage](#)
 - [Nsight Visual Studio Code Edition Spotlight Video](#)
 - [It's Alive: CUDA in Visual Studio Code! – GTC 2021 Presentation](#)
- ***CMake Presets*:** [Cross-Platform Pitfalls and How to Avoid Them - Erika Sweet - \[ACCU 2021 \] - YouTube](#)

Enjoy the rest of the conference!

Join #visual_studio channel on CppCon Discord
<https://aka.ms/cppcon/discord>

- Meet the Microsoft C++ team
- Ask any questions
- Discuss the latest announcements

Take our survey
<https://aka.ms/cppcon>

Our Sessions

Monday 25th

- **Implementing C++ Modules: Lessons Learned, Lessons Abandoned** – Cameron DaCamara & Gabriel Dos Reis

Tuesday 26th

- **Documentation in The Era of Concepts and Ranges** – Sy Brand & Christopher Di Bella (Google)
- **Static Analysis and Program Safety in C++: Making it Real** – Sunny Chatterjee
- **In-memory and Persistent Representations of C++** – Gabriel Dos Reis (online 27th)
- **Extending and Simplifying C++: Thoughts on pattern Matching using ``is`` and ``as``** – Herb Sutter

Wednesday 27th

- **What's New in Visual Studio: 64-bit IDE, C++20, WSL 2, and more** – Sy Brand & Marian Luparu

Thursday 28th

- **C++20's `<chrono>` Calendars and Time Zones in MSVC** – Miya Natsuhara
- **An Editor Can Do That? Debugging Assembly Language and GPU Kernels in Visual Studio Code** – Julia Reid
- **Why does `std::format` do that?** – Charlie Barto
- **Finding bugs using path-sensitive static analysis** – Gabor Horvath (online 29th)

Happy Coding!

Thank you

Twitter: @jureid22