ROS-Industrial Developer's Meeting - June 2023

TURBOCHARGE IDE FOR ROS2 DEVELOPMENT

IDE configuration to improve ROS2 development productivity

Managed by Advanced Remanufacturing and Technology Centre ARTC

Overview

- Why IDE?
- VSCode
  - Semantic auto-completion
  - Debugger
- Vim
  - Semantic auto-completion
- Final Word

Why IDE?

Why IDE?

VSCode

VIM

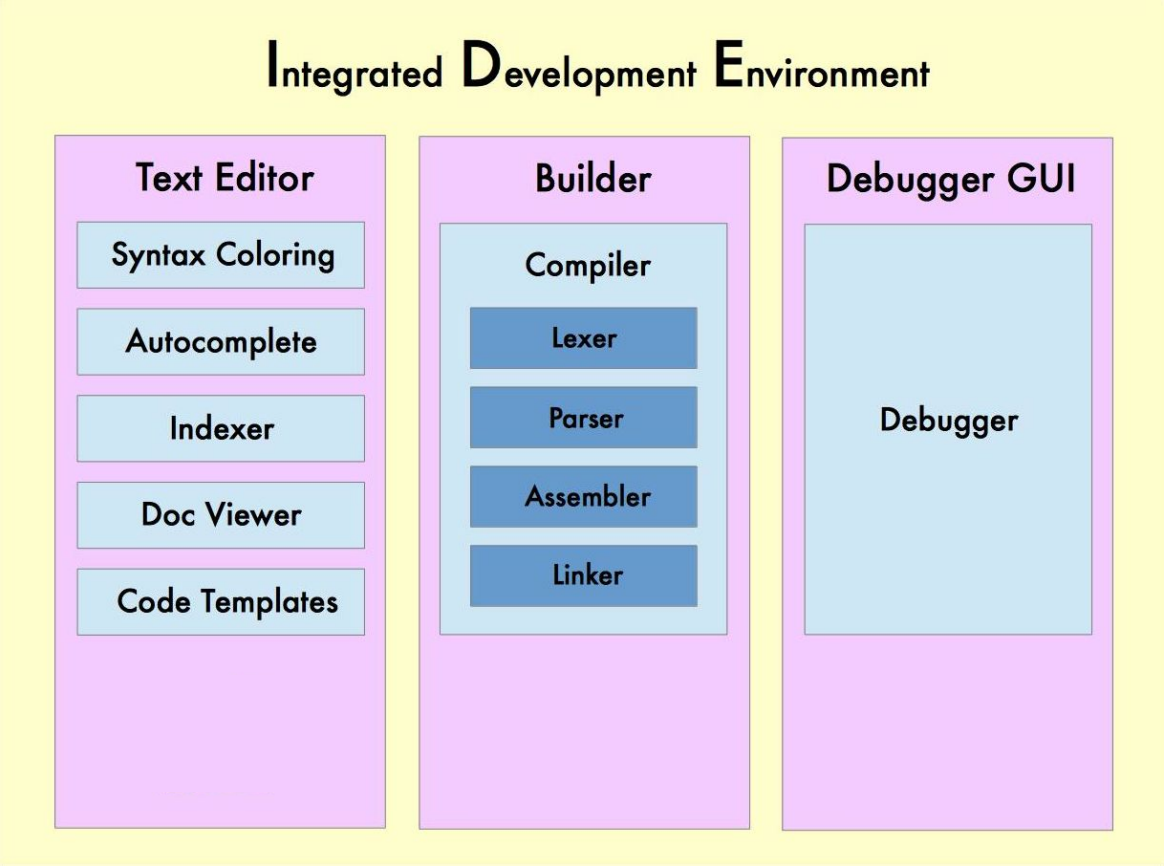
Final Word

What is an IDE?

An integrated development environment (IDE) is a software application that helps programmers develop software code **efficiently**.

An IDE typically consists of at least

- A source-code editor,
- Build automation tools, and
- Integration with a debugger.



ROS development pain

"colcon build" spitting out millions of the compilation errors

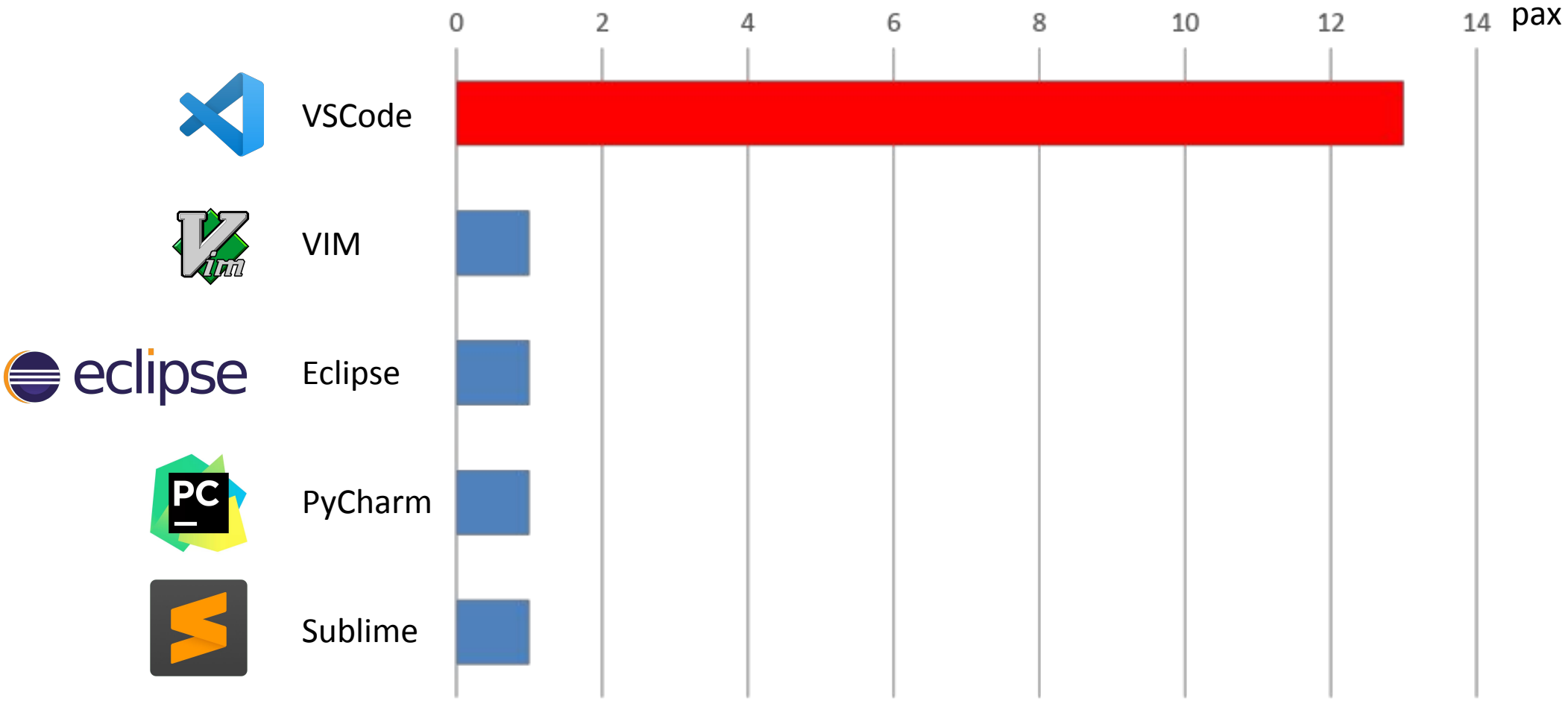
Checking back and forth in the headers and API documentation.

VSCode highlighting every part of your code for warning although there is nothing wrong

Writing hundreds of RCLCPP_INFO statement just to find a single Segfault

Different IDEs / Text Editor for ROS?

IDE / Text Editor usage in ROS-I AP



Qt Creator (Honorable mention)
https://github.com/ros-industrial/ros_qtc_plugin.git

VSCode

Why IDE?

VSCode

VIM

Final Word

VSCode – Why semantic completion?

- No need to check API documentation for usage
- No need to run colcon build to check syntax
- Easier to on-board new developers

VSCode – Autocompletion

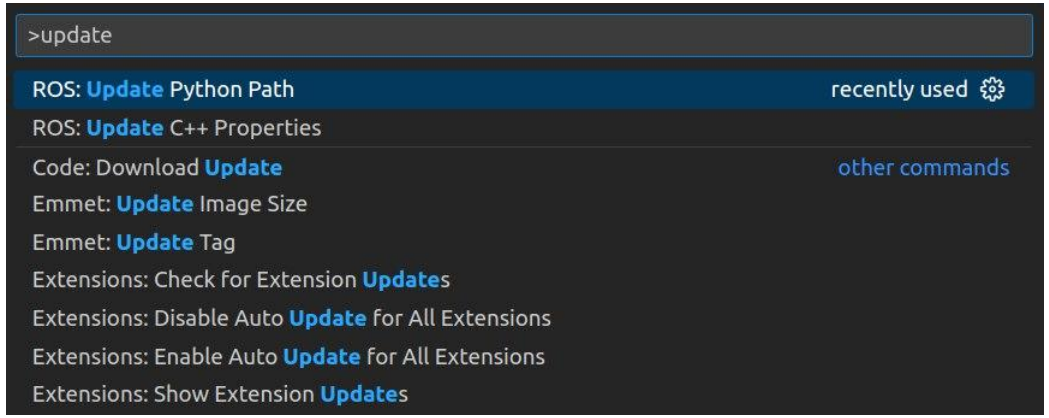
Extension needed

- C/C++ (ms-vscode.cpptools)
- CMake (twxs.cmake)
- Python (ms-python.python)
- ROS Extension (ms-iot.vscode-ros)

VSCode – Autocompletion

Method 1: ROS Extension

- `Ctrl+Shift+P` to open Command Palette



- Select “ROS: Update C++ Properties” to generate

  `c_cpp_properties.json`

VSCode – Autocompletion – Method 2 [1/2]

Method 2: Compile Commands

- Generate `compile_commands.json`
- Update `c_cpp_properties.json` to use the generate compile commands

VSCode – Autocompletion – Method 2 [2/2]

Method 2: Compile Commands

- Generate `compile_commands.json`

```
colcon build --cmake-args \
  -DCMAKE_EXPORT_COMPILE_COMMANDS=ON
```

- Update `c_cpp_properties.json` to use the generate compile commands

```
{
    "configurations": [
        {
            "name": "Linux",
            "includePath": [
                "${workspaceFolder}/**"
            ],
            "defines": [],
            "compilerPath": "/usr/bin/gcc",
            "cStandard": "c11",
            "cppStandard": "c++14",
            "intelliSenseMode": "linux-gcc-x64",
            "compileCommands": "${workspaceFolder}/build/compile_commands.json"
        }
    ],
    "version": 4
}
```

https://gist.github.com/Briancbn/650235313cac72d7f4e5d78e41e4db79

Debugging in Visual Studio Code

- Benefits of using a debugger
- User-friendly debugging in Visual Studio Code
- Set up with ROS 2 development

Why use a Debugger?

A debugger can provide more information about exceptions and crashes:

```
$ ros2 run
```

```
[INFO] [1686134186.929851235] [minimal_publisher]: Calculating 1/0..
[ros2run]: Floating point exception
```

Why use a Debugger?

$ ros2 run --prefix 'gdb -ex run' followed by **backtrace**

```
[INFO] [1686134603.423013258] [minimal_publisher]: Calculating 1/0..

Thread 1 "publisher_membe" received signal SIGFPE, Arithmetic exception.
0x00005555555629c8 in MinimalPublisher::complicated_function (this=0x555555627a50, n=1) at /home/r
g/workspacefloatingpoint/src/examples/rclcpp/topics/minimal_publisher/src/member_function.cpp:56
56          return n/0;
(gdb) backtrace
#0  0x00005555555629c8 in MinimalPublisher::complicated_function (this=0x555555627a50, n=1)
    at /home/rosi/devmtg/workspacefloatingpoint/src/examples/rclcpp/topics/minimal_publisher/src/m
ction.cpp:56
#1  0x000055555556286a in MinimalPublisher::timer_callback (this=0x555555627a50)
    at /home/rosi/devmtg/workspacefloatingpoint/src/examples/rclcpp/topics/minimal_publisher/src/m
ction.cpp:49
#2  0x0000555555588669 in std::__invoke_impl<void, void (MinimalPublisher::*&)(), MinimalPublisher
```

Stack trace and line numbers!

Why use a Debugger?

A debugger allows us to pause the execution of our program and step through it line by line.

It lets us observe:

- The program's flow, branches and loops
- Variables and their values

Debugging with gdb

```
(gdb) file install/examples_rclcpp_minimal_publisher/lib/.../publisher
Reading symbols from ...
(gdb) l 48
46        message.data = "Print me if I'm an even number " +
   std::to_string(count_++);
47
48        if (complicated_function(count_))
49        {
(gdb) b 48
Breakpoint 1 at 0xe5fb: file … src/main.cpp, line 48.
(gdb) r
Starting program: /home/rosi/...
Thread 1 "publisher" hit Breakpoint 1, MinimalPublisher::timer_callback …
   cpp:48
48        if (complicated_function(count_))
(gdb) p count_
$1 = 1
```

Demo: Debugging in Visual Studio Code

Demo: Debugging in Visual Studio Code

Configure with just existing launch files without having to specify included parameters

launch:

Launches all the executables in your launch file in VS Code's integrated terminals.

attach:

Attach to a process that is already running. Useful when you are launching multiple applications and only want to debug one.

Setup and Useful Guides

gdb: apt install gdb

VS Code Extensions: 'C/C++' and 'ROS' by Microsoft

ROS/ROS 2 extension

https://github.com/ms-iot/vscode-ros

Using and configuring debugging in VS Code

https://code.visualstudio.com/docs/editor/debugging

https://code.visualstudio.com/docs/cpp/cpp-debug

Code modified from

GitHub - ros2/examples: Example packages for ROS 2

Final tips and reminders

- If using the ROS extension, launch VS Code from your workspace (and not a subdirectory)

- Build with Debug flag on

  $ colcon build –cmake-args –DCMAKE_BUILD_TYPE=Debug

For reference: Screenshot

RUN AND... ROS: Laun

launch.json | main.cpp U | publisher.launch.xml U

src > examples > rcl ... er > src > main.cpp > MinimalPublisher > is_e

VARIABLES

Locals

remainder: 1

this: 0x5555556274f0

n: 5

Registers

BREAKPOINTS

All C++ Exceptions

main.cpp src/examples/rclcp... 48

WATCH

count_: 5

```
41
42
43  void print_even_numbers()
44  {
45    auto message = std_msgs::msg::String();
46    message.data = "I'm an even number: " + std::to_string(count_++);
47
48    if (is_even(count_))
49    {
50      RCLCPP_INFO(this->get_logger(), "Publishing: '%s'", message.data.c_str());
51      publisher_->publish(message);
52    }
53  }
54
55  bool is_even(int n)
56  {
57    int remainder = n % 2;
58    if (remainder == 0)
59    {
60      return true;
61    }
62    else
```

PROBLEMS OUTPUT DEBUG CONSOLE GITLENS TERMINAL cppdbg: publisher

```
[INFO] [1686288522.025656185] [Talker]: Publishing: 'I'm an even number: 1'
[INFO] [1686288525.625476837] [Talker]: Publishing: 'I'm an even number: 3'
```

For reference: Launch Configurations

```
{
   "name": "ROS: Launch", // Name of this configuration
   "type": "ros", // Use the ROS extension
   "request": "launch", // Launch
   // Path to launch file
   "target": "${workspaceFolder}/install/examples_demo/share/examples_demo/publisher.launch.xml"
},
{
   "name": "ROS: Attach",
   "type": "ros",
   "request": "attach", // Attach to an existing process
   "processId":"${action:pick}"
},
```

For reference: Launch Configurations

```
{
   "name": "cppdbg: Attach",
   "type": "cppdbg", // Use the C/C++ extension
   "request": "attach",
   "processId":"${command:pickProcess}",
   "program":"${workspaceFolder}/build/examples_demo/publisher" // Path to executable
},
{
   "name": "Python bindings",
   "type": "cppdbg",
   "request": "attach",
   "program": "/usr/bin/python3",
   "processId": "${command:pickProcess}",
   "MIMode": "gdb",
   "setupCommands": [
      {
         "description": "Enable pretty-printing for gdb",
         "text": "-enable-pretty-printing",
      }
   ]
}
```

VIM

Why IDE?

VSCode

VIM

Final Word

VIM – Why VIM?

- Server application
- There is always vi on every system
- Really don’t need a mouse

VIM – Warnings ahead

This is not for you if

- Don’t know how to exit VIM
- Only know how to exit VIM
- Don’t want to spend copious amount of time customizing.

VIM – Autocompletion

Steps needed

- Install YouCompleteMe
- Generate Compile Commands
- Download .ycm_extra_conf.py

Check out my detail guide https://github.com/Briancbn/ros_vim_autocomplete

Final words

Why IDE?

VSCode

VIM

Final Word

Final Words

工欲善其事，必先利其器

If a workman wishes to do a good job, he must first sharpen his tools.

Happy Coding!

Q&A

Thank You!