

School of Electronics and Communication Engineering

internship Project Work 2021-22

on

MICROROS SUPPORT ON DEVELOPMENT BOARDS

SL.NO	NAME	USN
1.	Aditya krishna vamsy Mudragada	01FE18BEC006

Under the guidance of:
Prof. Prabha C N
College Guide
KLE Technological University

Ravikumar Nanjaiah
Industry Guide
Bosch global software technologies

Overview

- Introduction
- Literature Survey
- Problem statement
- Objectives
- Prerequisites
- Implementation
- Demonstration of results
- Conclusion and Future Scope
- References

Introduction

- Micro-ROS is a robotic framework designed for embedded and deep-embedded robot components that have limited processing resources.
- Micro-ROS allows traditional robots to communicate with IoT sensors and devices, resulting in truly dispersed robotic systems based on a common foundation
- Support for this standard is crucial on several development boards that simulate the automotive environment, where ECUs with various RTOS and hardware are common.

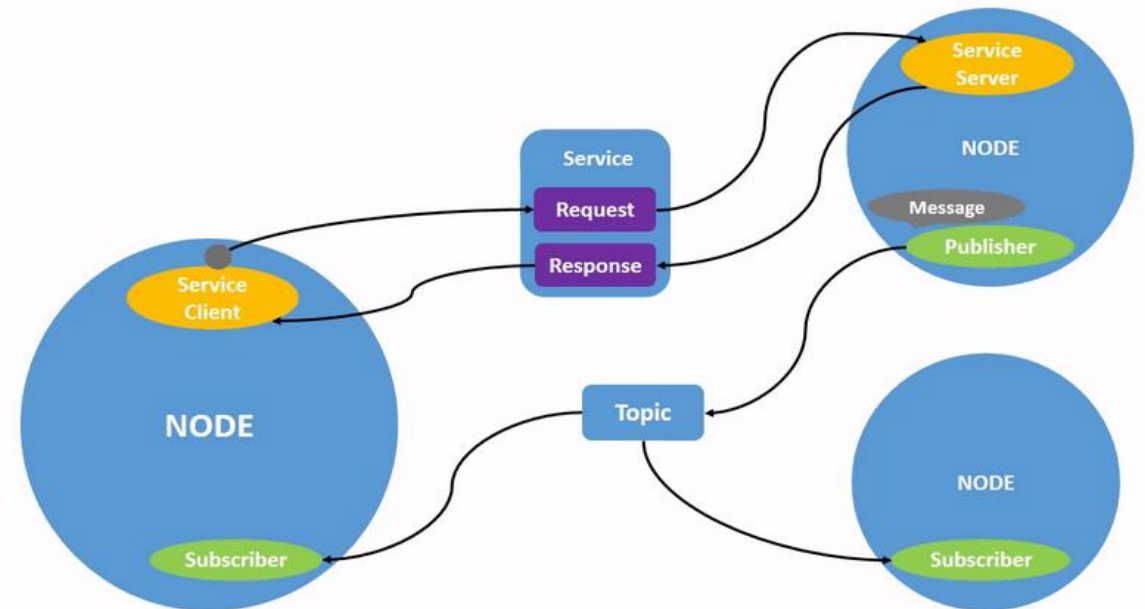
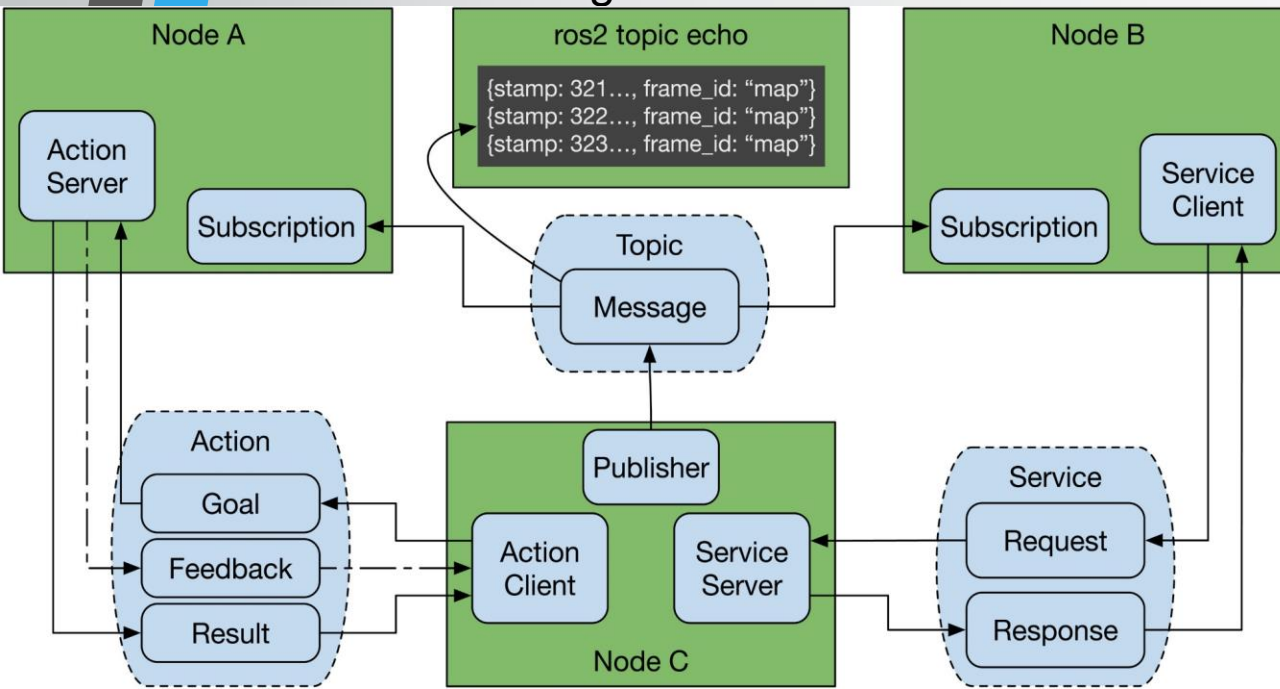
Literature survey

ROS2

- The Robot Operating System (ROS) is a set of software libraries and tools for building robot applications and is an open-source robotics middleware suite.

Essential concepts and features of ROS2

- Nodes
- Communication patterns
- Messages
- Rosbag



Literature survey

Microros

- Microros is a robotic framework the is made for extremely resource constrained devices such as embedded devices. Micro-ROS is compatible with Robot Operating System (ROS 2.0), the de facto standard for robot application development.
- micro-ROS **puts ROS 2 onto microcontrollers.**
- Microros has a middleware interface which is written in C language.

Features of Microros

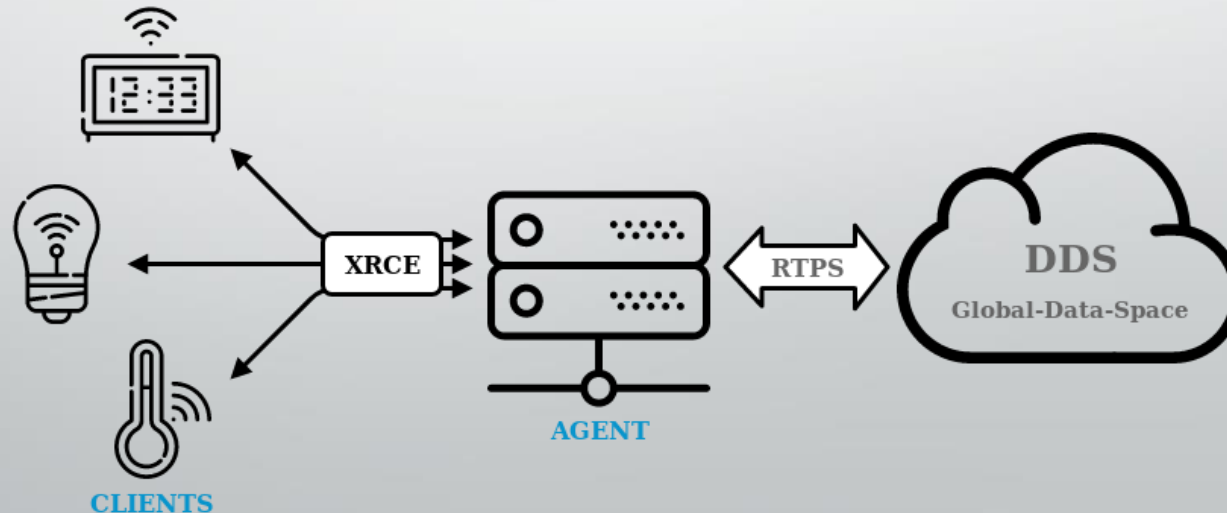
- Microcontroller-optimized client API supporting all major ROS concepts
- Seamless integration with ROS 2
- Extremely resource-constrained but flexible middleware
- Multi-RTOS support with generic build system



Literature survey

Middleware

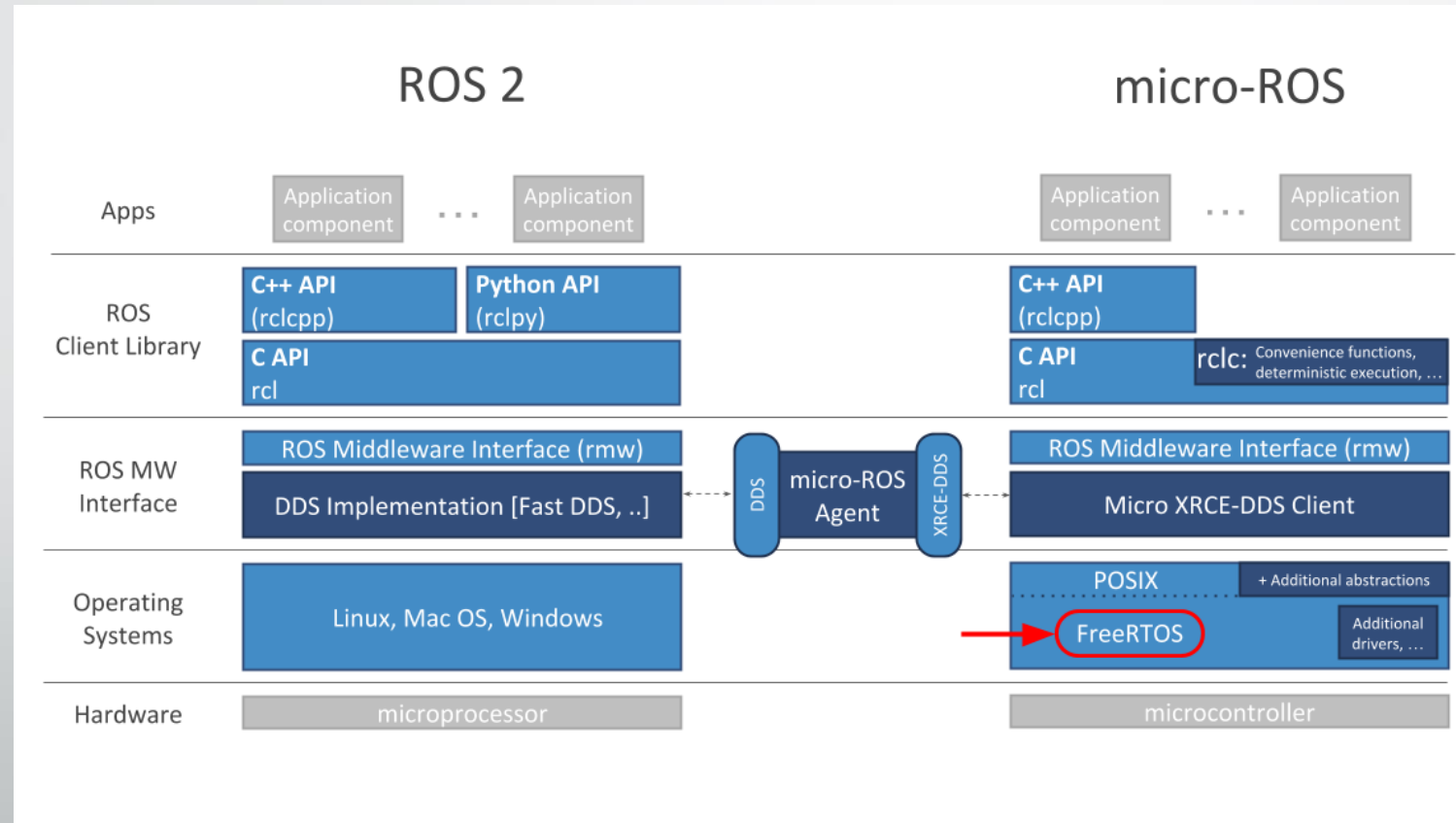
- Middleware is the software layer that sits between the operating system and the applications in a distributed system. It allows the many components of a system to communicate and share data more readily.
- There are numerous standards and products for communications middleware.
- The default middleware implementation for micro-ROS' rmw layer is Micro XRCE-DDS.
- Micro XRCE-DDS is an open-source wire protocol that implements the OMG DDS for eXtremely Resource Constrained Environment standard (DDS-XRCE). The aim of the DDS-XRCE protocol is to provide access to the DDS Global-Data-Space from resource-constrained devices.
- Micro-XRCE-DDS couples with the standard DDS middleware used in ROS 2 by the micro-ROS agent.



Literature survey

Architecture of ROS and Micro-ROS

- The microros has the same architecture as ROS2 with distributed real-time system architecture .
- The microros applications access the ros2 features using the ROS client library.



Problem statement

Document the software build steps of microROS and to Identify and document the steps specific to integrating RTOSes on development boards on microROS. Demonstrate the know-how gathered by integrating microROS on specific development board

Objectives

- Understand and compare the embedded build steps with microros build steps.
- Integrate Microros with RTOSes on a target board from application perspective.

Prerequisites

Build steps of embedded system

- The build process is to compile the c code to machine understandable code for the target system
- An Embedded system would also use tools such as a Compiler, Linker, Locater and Debugger to perform the entire build process.
- The different tools used for the build process in embedded systems:
 1. Toolchain
 2. Target system
 3. Cross compilers
 4. Linker script
 5. Flashing to the target hardware
- The need for such a procedure is because the host pc we are working on may be using windows which has a separate architecture like intel etc but the embedded system has a different architecture so, the host system cannot directly run the program on the target architecture.

Prerequisites

Build steps of microros

We start with setting up a workspace with a ready-to-use micro-ros build system. build system oversees downloading the necessary cross-compilation tools and building the apps for the required platforms.

Workflow of the build procedure:

Create step: This step is responsible for downloading the relevant code repositories and cross-compilation toolchains for the hardware platform in question

Configure step: The user can choose which app will be cross-compiled by the toolchain at this stage and transport.

Build step: This is where cross-compilation and platform-specific binaries are generated.

Flash step: :The binaries generated in the preceding step are flashed onto the hardware platform memory to allow the micro-ROS application to run.

Prerequisites

Microros on freertos

Create step:

- 1.Validation: retrieving RTOS , check if the firmware exists , Checking folder
- 2.Setting common environment
- 3.Check if generic build
- 4.Creating development directory and muc directory
- 5.Building development workspace
- 6.install dependencies for specific platform.
7. Installing toolchain
8. Import repos related to the board

Analysis of freertos board repos: Because different microcontroller families have distinct architectures, the approach taken by freertos varies.

Configure the firmware:

1. Checking if firmware exists and configure script exist
2. Parsing micro-ROS arguments
3. Configure specific firmware folder if needed
4. Set the DDS and Transport.

Prerequisites

Microros on freertos

Building the firmware:

1. Parse cli arguments
2. Checking if firmware exists
3. Clean paths
4. Building the specific firmware folder
5. Initialize the environment
6. Retrieve the app to be built
7. Choose configuration based on transport and host.
8. Build the app using west command line tool.

Flashing the firmware:

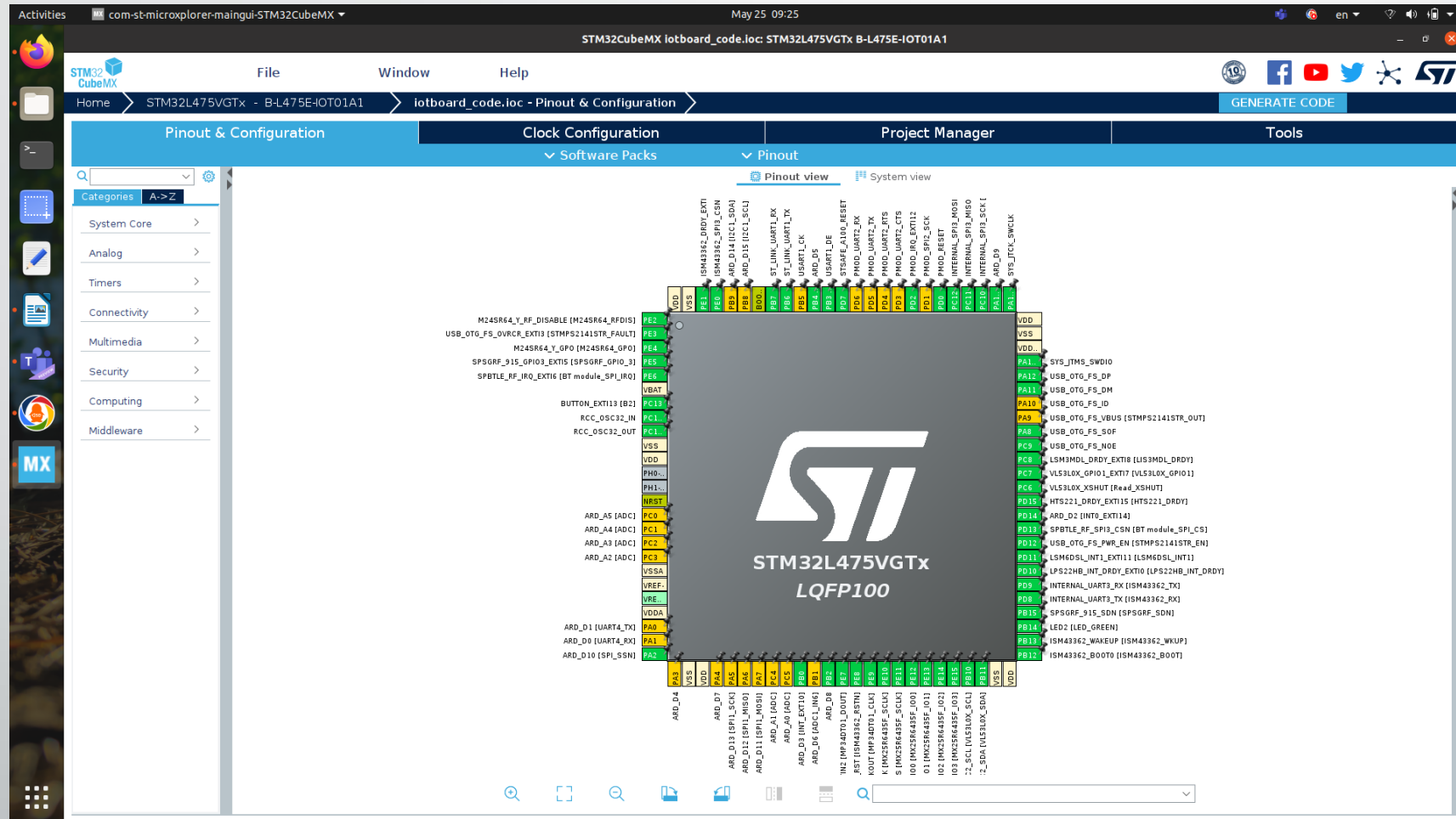
1. Checking if firmware exists
2. Flash specific firmware folder if needed
3. On chip debugger.

Implementation

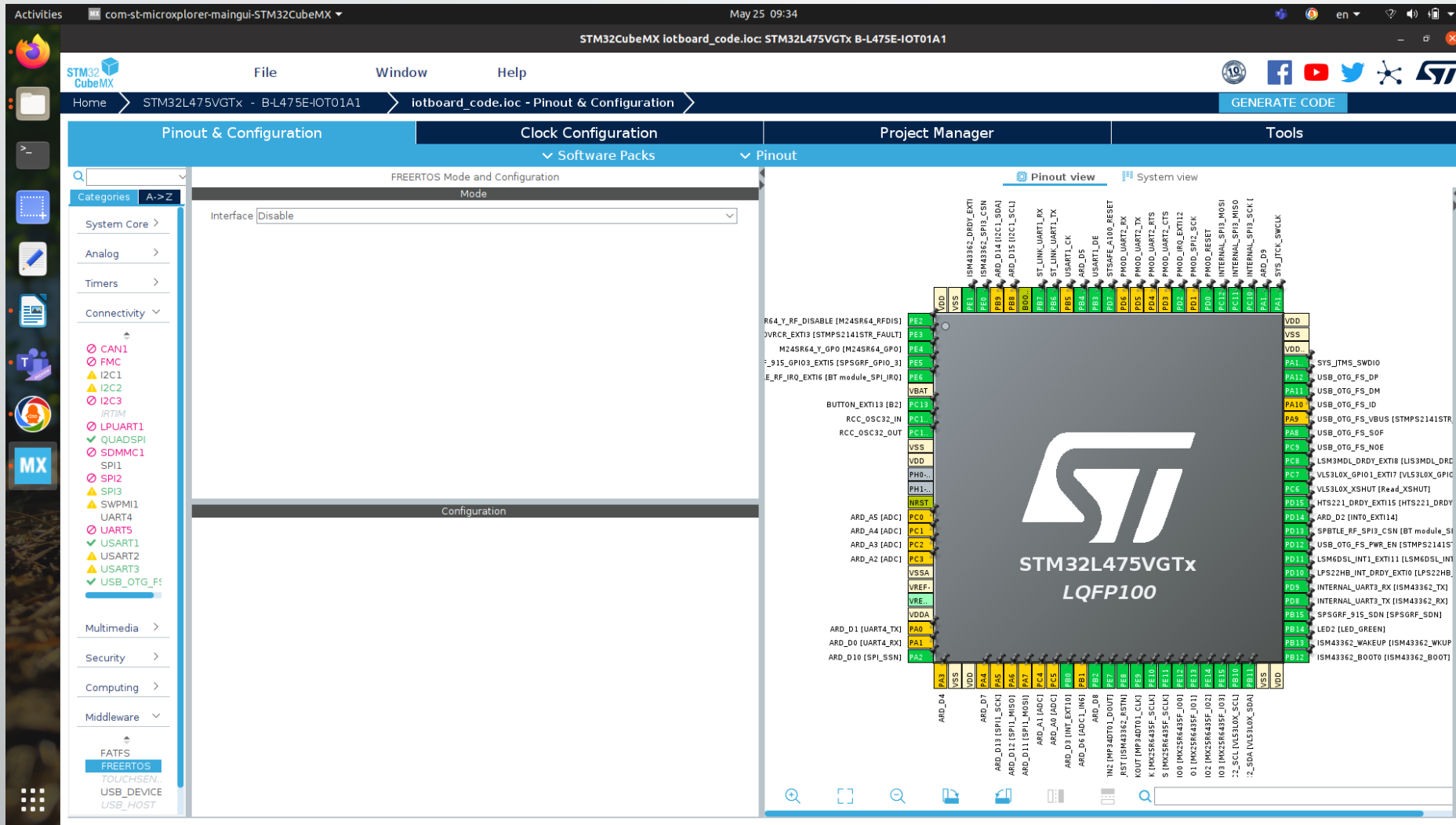
- Clone the microros build related repo and include code related to you target board.
- Implementation is based on application view point.
- Set the pin configuration using stm32cubemx tool, choose connectivity and rtos as freertos.
- Generate the code using the stm32cubemx code generator.
- Add posix api , toolchain Cmake file which are needed by microros.
- There are changes that needs to be made ranging from source file to connectivity related.
- Upload the after making changes and connect it board related code at microros build repo.
- Run the Create , configure , build , flash steps.

There are a lot of sub steps involved while making these changes the points are mentioned to understand the changes need to be made in microros build flow.

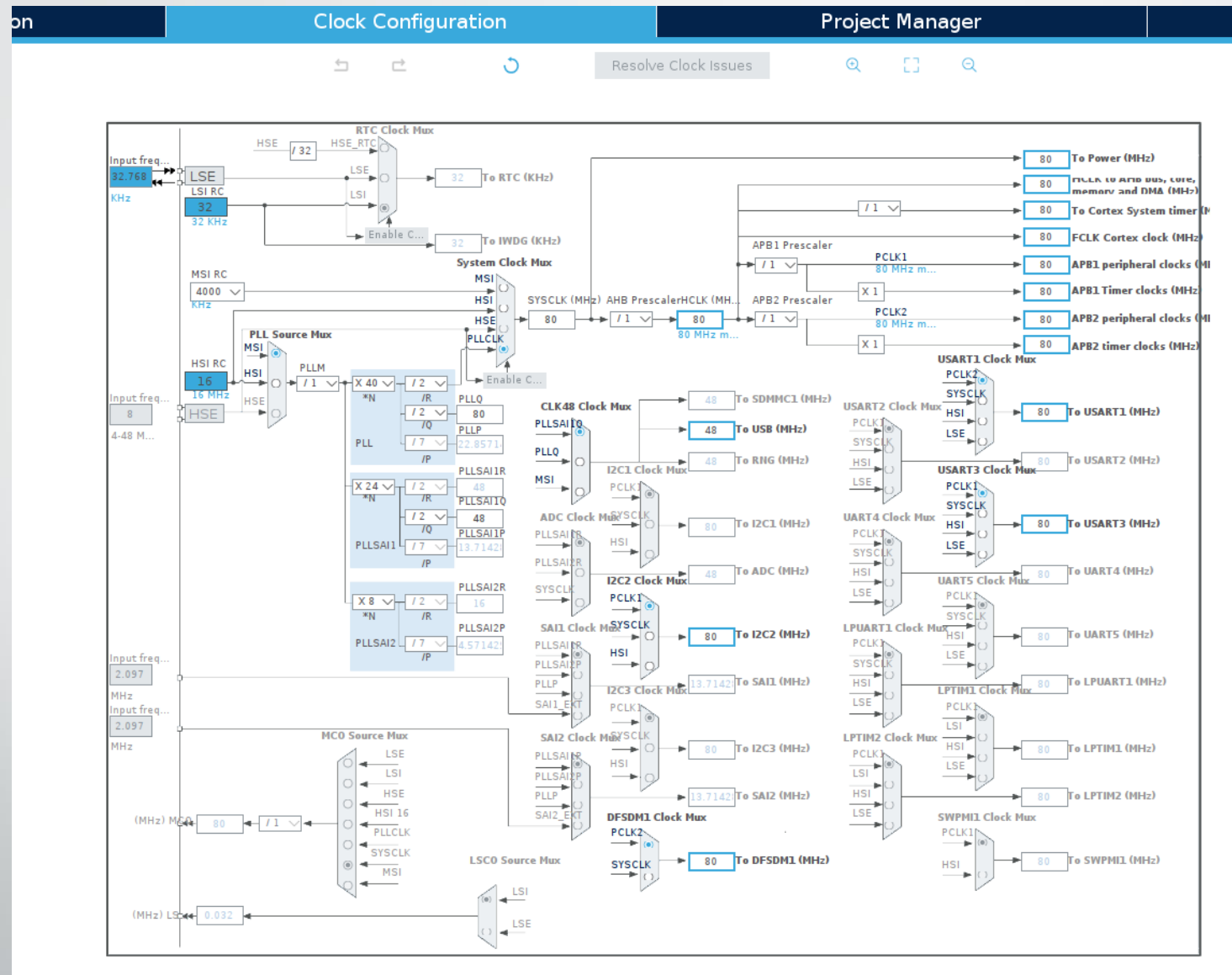
Implementation



Implementation



Implementation



Implementation

Activities | com-st-microexplorer-maingui-STM32CubeMX | May 25 09:41 | STM32CubeMX | en |

STM32CubeMX | File | Window | Help | 19 | f | y | t | s | ST |

Home | STM32L475VGTX - B-L475E-IOT01A1 | **iotboard_code.ioc - Project Manager** | GENERATE CODE

Pinout & Configuration	Clock Configuration	Project Manager	Tools
Project	<p>Project Settings</p> <p>Project Name: <input type="text" value="iotboard_code"/></p> <p>Project Location: <input type="text" value="/home/judwlkor/stm32code"/> <input type="button" value="Browse"/></p> <p>Application Structure: <input type="text" value="Advanced"/> <input type="checkbox"/> Do not generate the main()</p> <p>Toolchain Folder Location: <input type="text" value="/home/judwlkor/stm32code/iotboard_code/"/></p> <p>Toolchain / IDE: <input type="text" value="Makefile"/> <input type="checkbox"/> Generate Under Root</p>		
Code Generator			
Advanced Settings	<p>Linker Settings</p> <p>Minimum Heap Size: <input type="text" value="0x200"/></p> <p>Minimum Stack Size: <input type="text" value="0x400"/></p> <p>Thread-safe Settings</p> <p>Cortex-M4NS</p> <p><input type="checkbox"/> Enable multi-threaded support</p> <p>Thread-safe Locking Strategy: <input type="text" value="Default - Mapping suitable strategy depending on RTOS selection."/></p> <p>Mcu and Firmware Package</p> <p>Mcu Reference: <input type="text" value="STM32L475VGTX"/></p> <p>Firmware Package Name and Version: <input type="text" value="STM32Cube_FW_L4_V1.17.2"/> <input checked="" type="checkbox"/> Use latest available version</p> <p><input checked="" type="checkbox"/> Use Default Firmware Location</p> <p>Firmware Relative Path: <input type="text" value="/home/judwlkor/STM32Cube/Repository/STM32Cube_FW_L4_V1.17.2"/> <input type="button" value="Browse"/></p>		

Drivers

Inc

Middlewares

Src

Makefile

micro-ROS.ioc

startup_stm32f407xx.s

STM32F407ZG Tx_FLASH.ld

Implementation

```
Activities Terminal May 26 16:46 udw1kor@BMH1115168: ~/topdown2

udw1kor@BMH1115168:~/topdown2$ source /opt/ros/foxy/setup.bash
udw1kor@BMH1115168:~/topdown2$ git clone -b $ROS_DISTRO https://github.com/adityakri/micro_ros_setup.git src/micro_ros_setup
Cloning into 'src/micro_ros_setup'...
remote: Enumerating objects: 3416, done.
remote: Counting objects: 100% (1211/1211), done.
remote: Compressing objects: 100% (478/478), done.
remote: Total 3416 (delta 866), reused 1024 (delta 723), pack-reused 2205
Receiving objects: 100% (3416/3416), 790.09 KiB | 111.00 KiB/s, done.
Resolving deltas: 100% (2364/2364), done.
udw1kor@BMH1115168:~/topdown2$ sudo apt update && rosdep update
Hit:1 http://mirror-osa.apac.bosch.com/ubuntu focal InRelease
Hit:2 http://mirror-osa.apac.bosch.com/ubuntu focal-security InRelease
Hit:3 http://mirror-osa.apac.bosch.com/ubuntu focal-updates InRelease
Ign:4 http://mirror-osa.apac.bosch.com/osu focal InRelease
Hit:5 http://mirror-osa.apac.bosch.com/partner focal InRelease
Hit:6 http://mirror-osa.apac.bosch.com/ms-teams stable InRelease
Hit:7 http://mirror-osa.apac.bosch.com/osu focal InRelease
Hit:8 http://mirror-osa.apac.bosch.com/osu focal InRelease
Hit:9 http://packages.ros.org/ros/ubuntu focal InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
54 packages can be upgraded. Run 'apt list --upgradable' to see them.
reading in sources list data from /etc/ros/rosdep/sources.list.d
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/python.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/ruby.yaml
ERROR: unable to process source [https://raw.githubusercontent.com/ros/rosdistro/master/releases/foxy.yaml]:
Failed to download target platform data for gbpdistro:
<urlopen error timed out>
Query rosdistro index https://raw.githubusercontent.com/ros/rosdistro/master/index-v4.yaml
ERROR: error loading sources list:
<urlopen error <urlopen error timed out> (https://raw.githubusercontent.com/ros/rosdistro/master/index-v4.yaml)>
udw1kor@BMH1115168:~/topdown2$ rosdep install --from-paths src --ignore-src -y
#All required rosdeps installed successfully
udw1kor@BMH1115168:~/topdown2$ colcon build
Starting >>> micro_ros_setup
Finished <<< micro_ros_setup [0.98s]

Summary: 1 package finished [1.20s]
udw1kor@BMH1115168:~/topdown2$ source install/local_setup.bash
udw1kor@BMH1115168:~/topdown2$ ros2 run micro_ros_setup create_firmware_ws.sh freertos discovery_l475_iot1
create_firmware_ws.sh line 31
create_firmware_ws.sh line 31
create_firmware_ws.sh line 53
Creating firmware for freertos platform discovery_l475_iot1
create_firmware_ws.sh line 63
create_firmware_ws.sh line 69
create_firmware_ws.sh line 77
create_firmware_ws.sh line 85
create_firmware_ws.sh line 88
create_ws.sh line 45
.....
=== ./ament/ament_cmake (git) ===
Cloning into '.'...
=== ./ament/ament_index (git) ===
```

Implementation

```
Activities Terminal May 26 16:44 colcon build [57/61 done] [1 ongoing]

/home/udw1kor/topdown2/firmware/mcu_ws/install/builtin_interfaces/share/builtin_interfaces/cmake/builtin_interfacesConfig.cmake:41 (include)
/home/udw1kor/topdown2/firmware/mcu_ws/install/action_msgs/share/action_msgs/cmake/ament_cmake_export_dependencies-extras.cmake:21 (find_package)
/home/udw1kor/topdown2/firmware/mcu_ws/install/action_msgs/share/action_msgs/cmake/action_msgsConfig.cmake:41 (include)
CMakeLists.txt:7 (find_package)

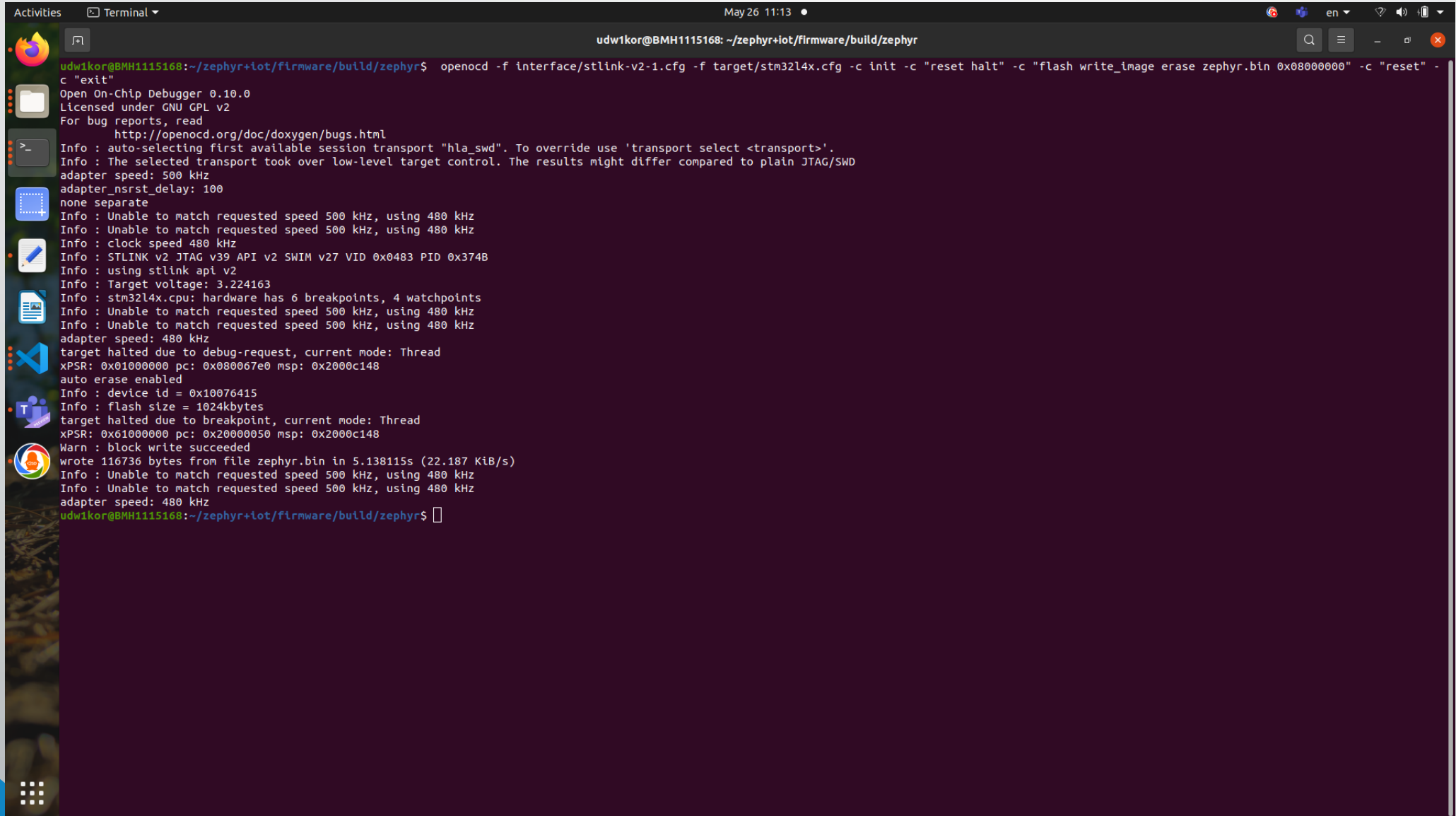
---
Finished <<< rcl_action [4.49s]
--- stderr: rcl_lifecycle
CMake Warning at /home/udw1kor/topdown2/firmware/mcu_ws/install/rcutils/share/rcutils/cmake/ament_cmake_export_dependencies-extras.cmake:116 (message):
  Package 'rcutils' exports library 'dl' which couldn't be found
Call Stack (most recent call first):
/home/udw1kor/topdown2/firmware/mcu_ws/install/rcutils/share/rcutils/cmake/rcutilsConfig.cmake:41 (include)
/home/udw1kor/topdown2/firmware/mcu_ws/install/rosidl_runtime_c/share/rosidl_runtime_c/cmake/ament_cmake_export_dependencies-extras.cmake:21 (find_package)
/home/udw1kor/topdown2/firmware/mcu_ws/install/rosidl_runtime_c/share/rosidl_runtime_c/cmake/rosidl_runtime_cConfig.cmake:41 (include)
/home/udw1kor/topdown2/firmware/mcu_ws/install/lifecycle_msgs/share/lifecycle_msgs/cmake/ament_cmake_export_dependencies-extras.cmake:21 (find_package)
/home/udw1kor/topdown2/firmware/mcu_ws/install/lifecycle_msgs/share/lifecycle_msgs/cmake/lifecycle_msgsConfig.cmake:41 (include)
CMakeLists.txt:11 (find_package)

---
Finished <<< rcl_lifecycle [4.26s]
--- stderr: rcl_parameter
CMake Warning at /home/udw1kor/topdown2/firmware/mcu_ws/install/rcutils/share/rcutils/cmake/ament_cmake_export_dependencies-extras.cmake:116 (message):
  Package 'rcutils' exports library 'dl' which couldn't be found
Call Stack (most recent call first):
/home/udw1kor/topdown2/firmware/mcu_ws/install/rcutils/share/rcutils/cmake/rcutilsConfig.cmake:41 (include)
/home/udw1kor/topdown2/firmware/mcu_ws/install/rosidl_runtime_c/share/rosidl_runtime_c/cmake/ament_cmake_export_dependencies-extras.cmake:21 (find_package)
/home/udw1kor/topdown2/firmware/mcu_ws/install/rosidl_runtime_c/share/rosidl_runtime_c/cmake/rosidl_runtime_cConfig.cmake:41 (include)
/home/udw1kor/topdown2/firmware/mcu_ws/install/builtin_interfaces/share/builtin_interfaces/cmake/ament_cmake_export_dependencies-extras.cmake:21 (find_package)
/home/udw1kor/topdown2/firmware/mcu_ws/install/builtin_interfaces/share/builtin_interfaces/cmake/builtin_interfacesConfig.cmake:41 (include)
/home/udw1kor/topdown2/firmware/mcu_ws/install/rcl_interfaces/share/rcl_interfaces/cmake/ament_cmake_export_dependencies-extras.cmake:21 (find_package)
/home/udw1kor/topdown2/firmware/mcu_ws/install/rcl_interfaces/share/rcl_interfaces/cmake/rcl_interfacesConfig.cmake:41 (include)
/home/udw1kor/topdown2/firmware/mcu_ws/install/rcl/share/rcl/cmake/ament_cmake_export_dependencies-extras.cmake:21 (find_package)
/home/udw1kor/topdown2/firmware/mcu_ws/install/rcl/share/rcl/cmake/rclConfig.cmake:41 (include)
CMakeLists.txt:34 (find_package)

---
Finished <<< rcl_parameter [4.32s]
--- stderr: visualization_msgs
CMake Warning at /home/udw1kor/topdown2/firmware/mcu_ws/install/rcutils/share/rcutils/cmake/ament_cmake_export_dependencies-extras.cmake:116 (message):
  Package 'rcutils' exports library 'dl' which couldn't be found
Call Stack (most recent call first):
/home/udw1kor/topdown2/firmware/mcu_ws/install/rcutils/share/rcutils/cmake/rcutilsConfig.cmake:41 (include)
/home/udw1kor/topdown2/firmware/mcu_ws/install/rosidl_runtime_c/share/rosidl_runtime_c/cmake/ament_cmake_export_dependencies-extras.cmake:21 (find_package)
/home/udw1kor/topdown2/firmware/mcu_ws/install/rosidl_runtime_c/share/rosidl_runtime_c/cmake/rosidl_runtime_cConfig.cmake:41 (include)
/home/udw1kor/topdown2/firmware/mcu_ws/install/builtin_interfaces/share/builtin_interfaces/cmake/ament_cmake_export_dependencies-extras.cmake:21 (find_package)
/home/udw1kor/topdown2/firmware/mcu_ws/install/builtin_interfaces/share/builtin_interfaces/cmake/builtin_interfacesConfig.cmake:41 (include)
CMakeLists.txt:14 (find_package)

---
Finished <<< visualization_msgs [15.3s]
[1min 34.3s] [57/61 complete] [sensor_msgs:build 34% - 17.3s]
```

Implementation



The image shows a terminal window titled "Terminal" with a dark background. The user is logged in as "udw1kor" on a machine named "BMH1115168". The current directory is "~/zephyr+iot/firmware/build/zephyr". The user has executed the command: `openocd -f interface/stlink-v2-1.cfg -f target/stm32l4x.cfg -c init -c "reset halt" -c "flash write_image erase zephyr.bin 0x08000000" -c "reset" -c "exit"`. The output shows the OpenOCD version (0.10.0) and license (GNU GPL v2). It then displays information about the selected transport (hla_swd) and the target (stm32l4x). The target is reset and the firmware is flashed. The output indicates that the flash write was successful and the target is now in the "halt" state. The user then enters the prompt `udw1kor@BMH1115168:~/zephyr+iot/firmware/build/zephyr$`.

```
udw1kor@BMH1115168:~/zephyr+iot/firmware/build/zephyr$ openocd -f interface/stlink-v2-1.cfg -f target/stm32l4x.cfg -c init -c "reset halt" -c "flash write_image erase zephyr.bin 0x08000000" -c "reset" -c "exit"
Open On-Chip Debugger 0.10.0
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
Info : auto-selecting first available session transport "hla_swd". To override use 'transport select <transport>'.
Info : The selected transport took over low-level target control. The results might differ compared to plain JTAG/SWD
adapter speed: 500 kHz
adapter_nsrst_delay: 100
none separate
Info : Unable to match requested speed 500 kHz, using 480 kHz
Info : Unable to match requested speed 500 kHz, using 480 kHz
Info : clock speed 480 kHz
Info : STLINK v2 JTAG v39 API v2 SWIM v27 VID 0x0483 PID 0x374B
Info : using stlink api v2
Info : Target voltage: 3.224163
Info : stm32l4x.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : Unable to match requested speed 500 kHz, using 480 kHz
Info : Unable to match requested speed 500 kHz, using 480 kHz
adapter speed: 480 kHz
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x080067e0 msp: 0x2000c148
auto erase enabled
Info : device id = 0x10076415
Info : flash size = 1024kbytes
target halted due to breakpoint, current mode: Thread
xPSR: 0x61000000 pc: 0x20000050 msp: 0x2000c148
Warn : block write succeeded
wrote 116736 bytes from file zephyr.bin in 5.138115s (22.187 KiB/s)
Info : Unable to match requested speed 500 kHz, using 480 kHz
Info : Unable to match requested speed 500 kHz, using 480 kHz
adapter speed: 480 kHz
udw1kor@BMH1115168:~/zephyr+iot/firmware/build/zephyr$
```

Conclusion and future scope

The level of complexity in software development required in robotics projects can be considerably decreased with the help of ROS.

ROS has an integrated framework and toolsets for robotics development, which speeds up software creation and aids in redistribution there by leading to new and faster innovations.

By attempting to incorporate microros onto a target board and keeping the technique as general as feasible, this is a first step in understanding the feasibility of using microros in automotive domain.

Plagiarism Report snap shot

al

ORIGINALITY REPORT

19%	18%	4%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	micro-ros.github.io	3%
	Internet Source	
2	www.bosch.com	2%
	Internet Source	
3	micro.ros.org	2%
	Internet Source	
4	www.coursehero.com	2%
	Internet Source	
5	docs.zephyrproject.org	1%
	Internet Source	
6	www.mytechlogy.com	1%
	Internet Source	
7	www.fiware.org	1%
	Internet Source	
8	jessicasiboro.blogspot.com	1%

platform for NAO robot in cognitive interaction applications", 2014 IEEE International Symposium on Robotics and Manufacturing Automation (ROMA), 2014.

Publication

10	developer.arm.com	1%
	Internet Source	
11	b-ok.org	1%
	Internet Source	
12	Than D. Le, An T. Le, Duy T. Nguyen. "Model-based Q-learning for humanoid robots", 2017 18th International Conference on Advanced Robotics (ICAR), 2017	<1%
	Publication	
13	github.com	<1%
	Internet Source	
14	archive.mu.ac.in	<1%
	Internet Source	
15	webthesis.biblio.polito.it	<1%
	Internet Source	
16	scholars.lib.ntu.edu.tw	<1%
	Internet Source	

Plagiarism Report snap shot

19	Supriya Katwe, Nalini Iyer. "Map-Based Particle Filter for Localization: Autonomous Vehicle", 2020 International Conference on Computational Performance Evaluation (ComPE), 2020 Publication	<1 %
20	www.asee.org Internet Source	<1 %
21	www.mail-archive.com Internet Source	<1 %
22	Daneil Steen Losvik, Adrian Rutle. "A Domain-Specific Language for the Development of Heterogeneous Multi-robot Systems", 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), 2019 Publication	<1 %
23	dokumen.pub Internet Source	<1 %
24	"Recent Findings in Intelligent Computing Techniques", Springer Science and Business Media LLC, 2018	<1 %



THANK YOU