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.

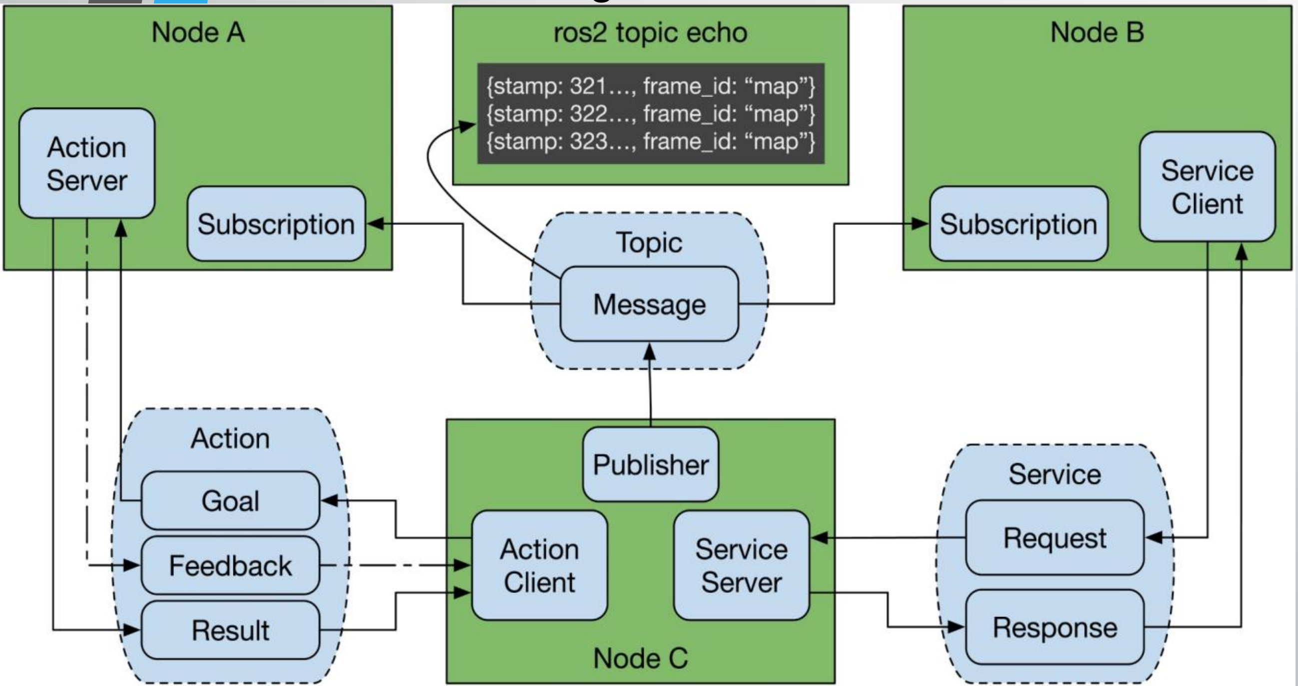
Literature survey

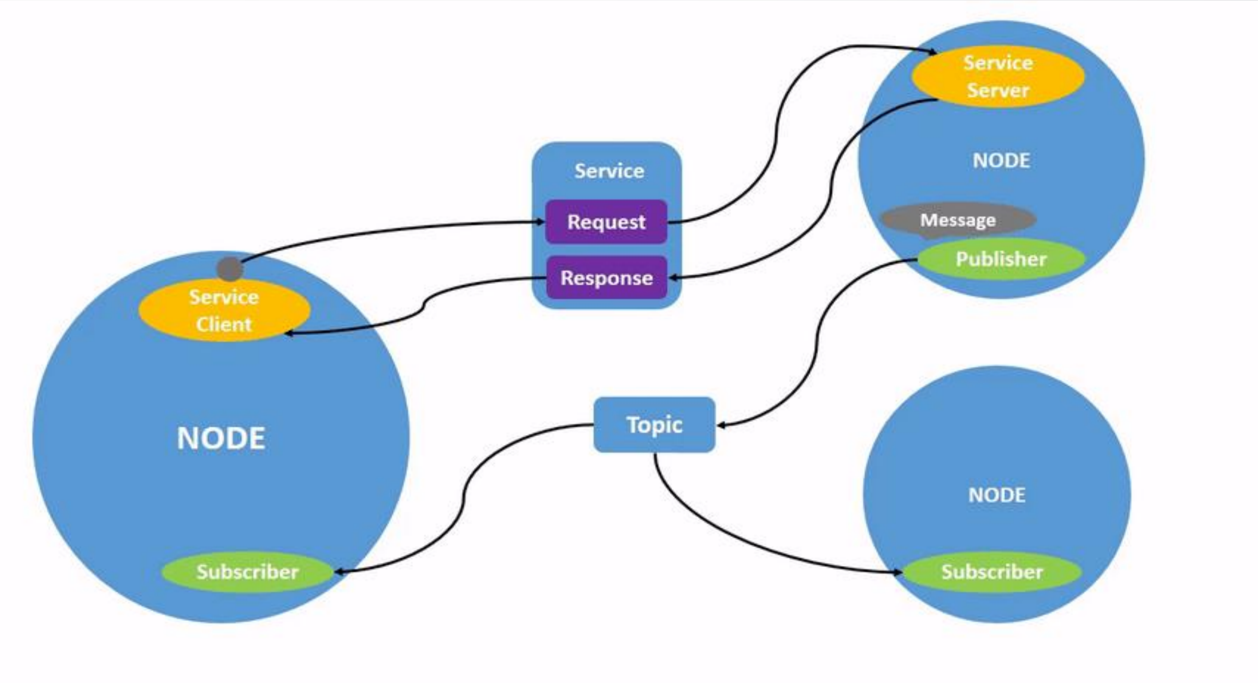
ROS2

- 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.

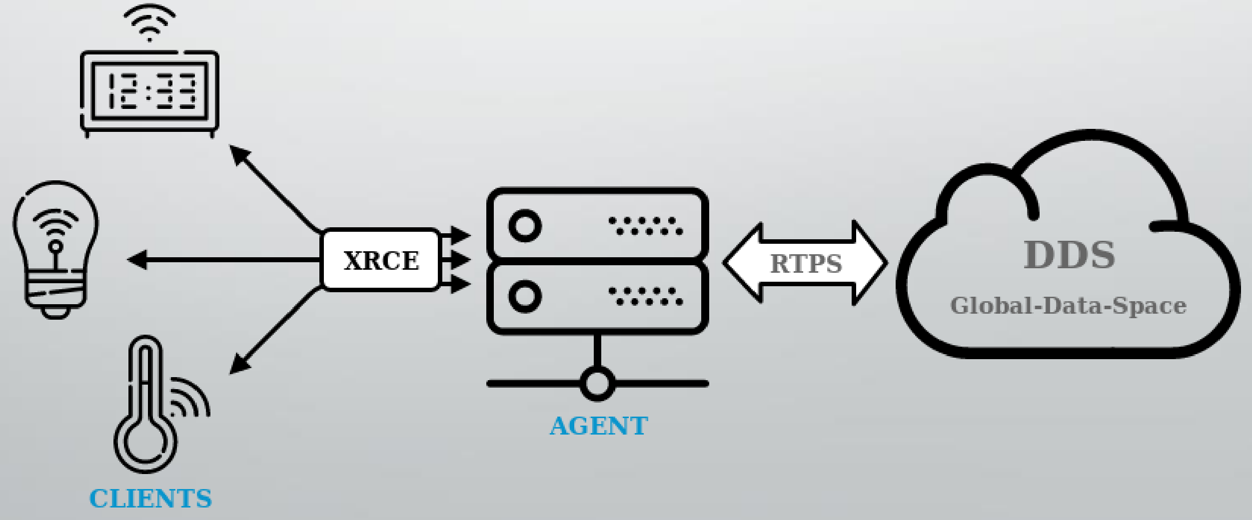
Features of Microros

- 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

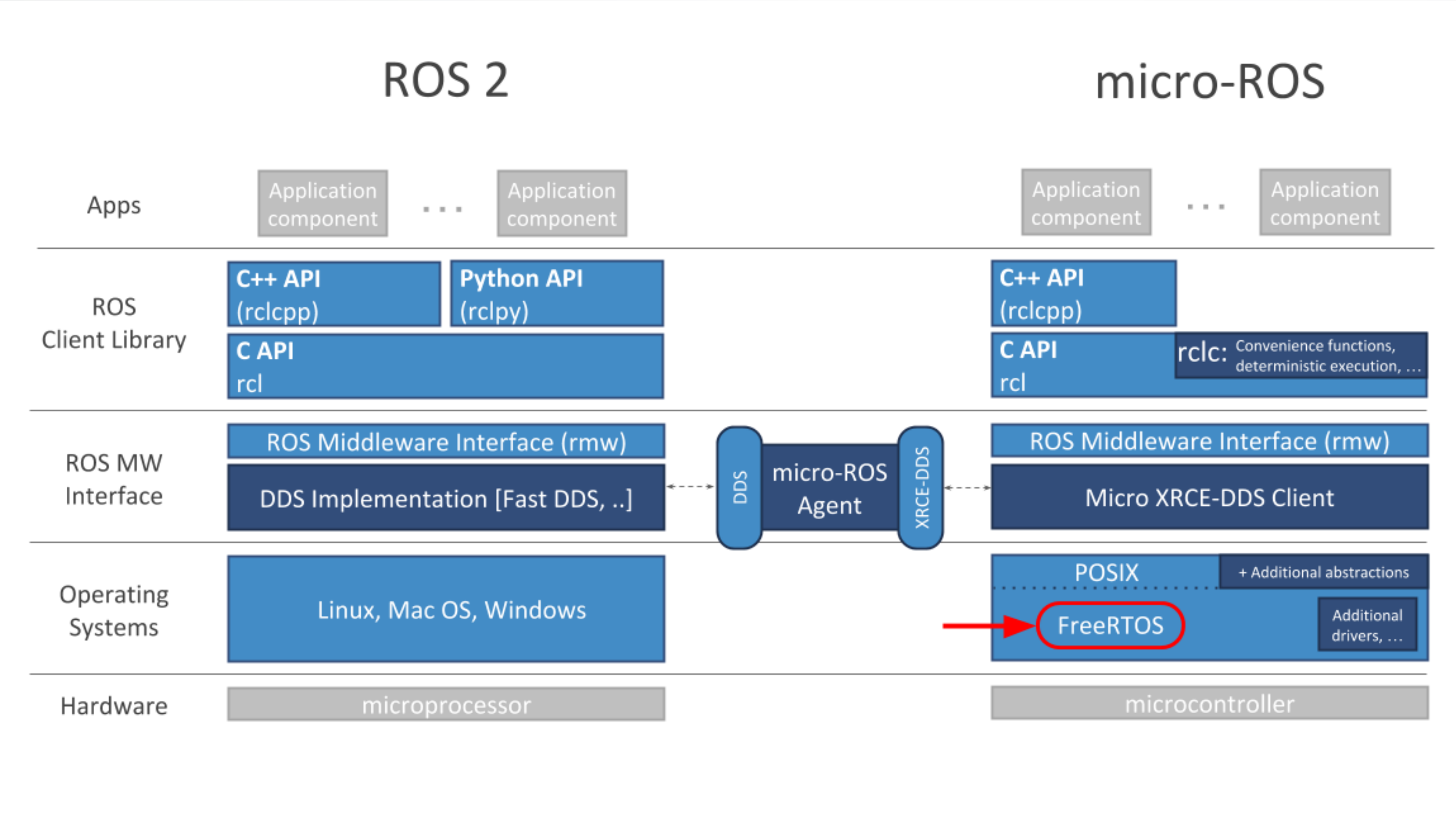
- Middleware is the software layer that sits between the operating system and the applications in a distributed system. It allows the many components of a system to communicate and share data more readily.
- There are numerous standards and products for communications middleware.
- The default middleware 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:

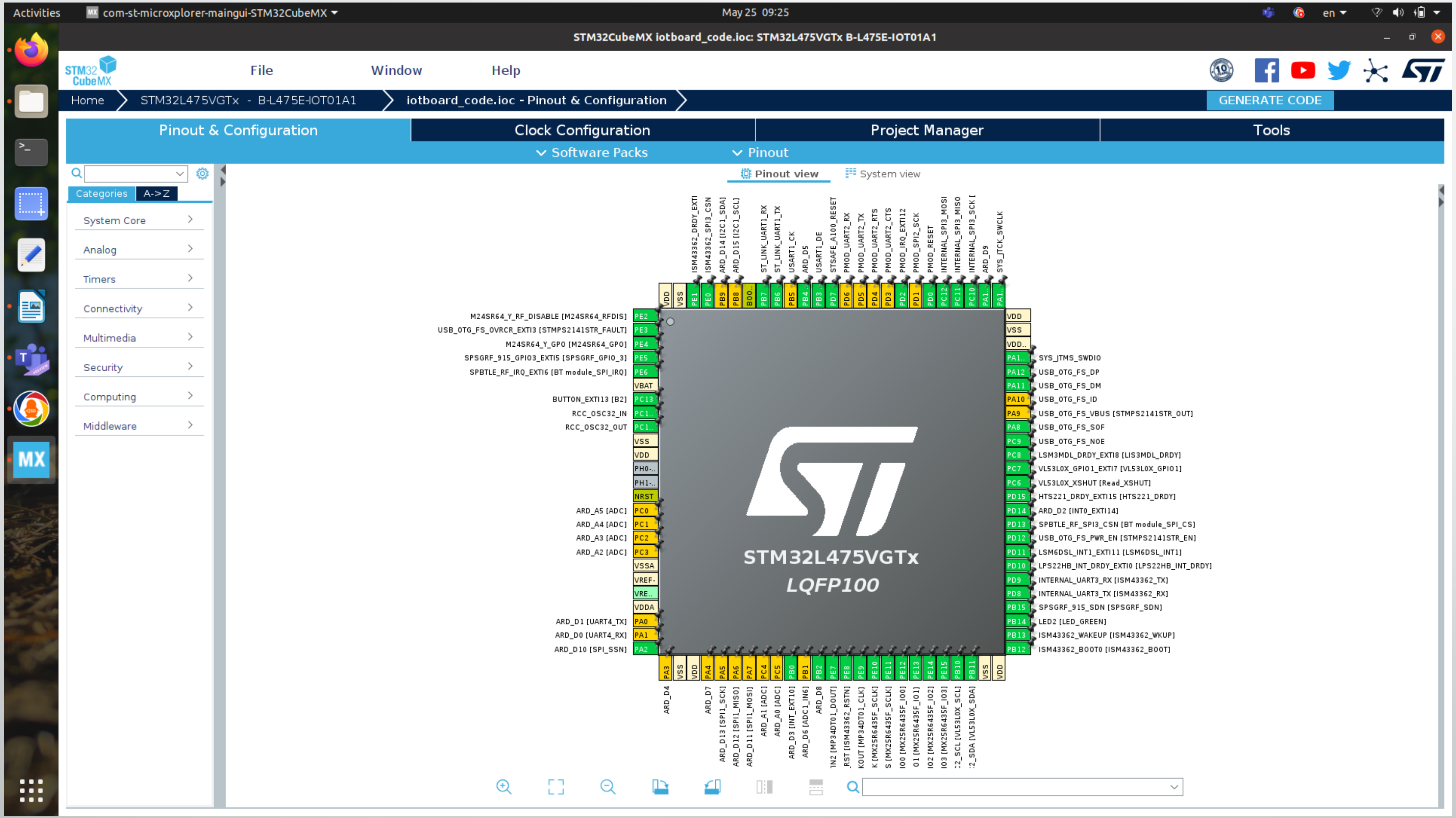
1. Checking if firmware exists
2. Flash specific firmware folder if needed
3. On chip debugger.

Implementation

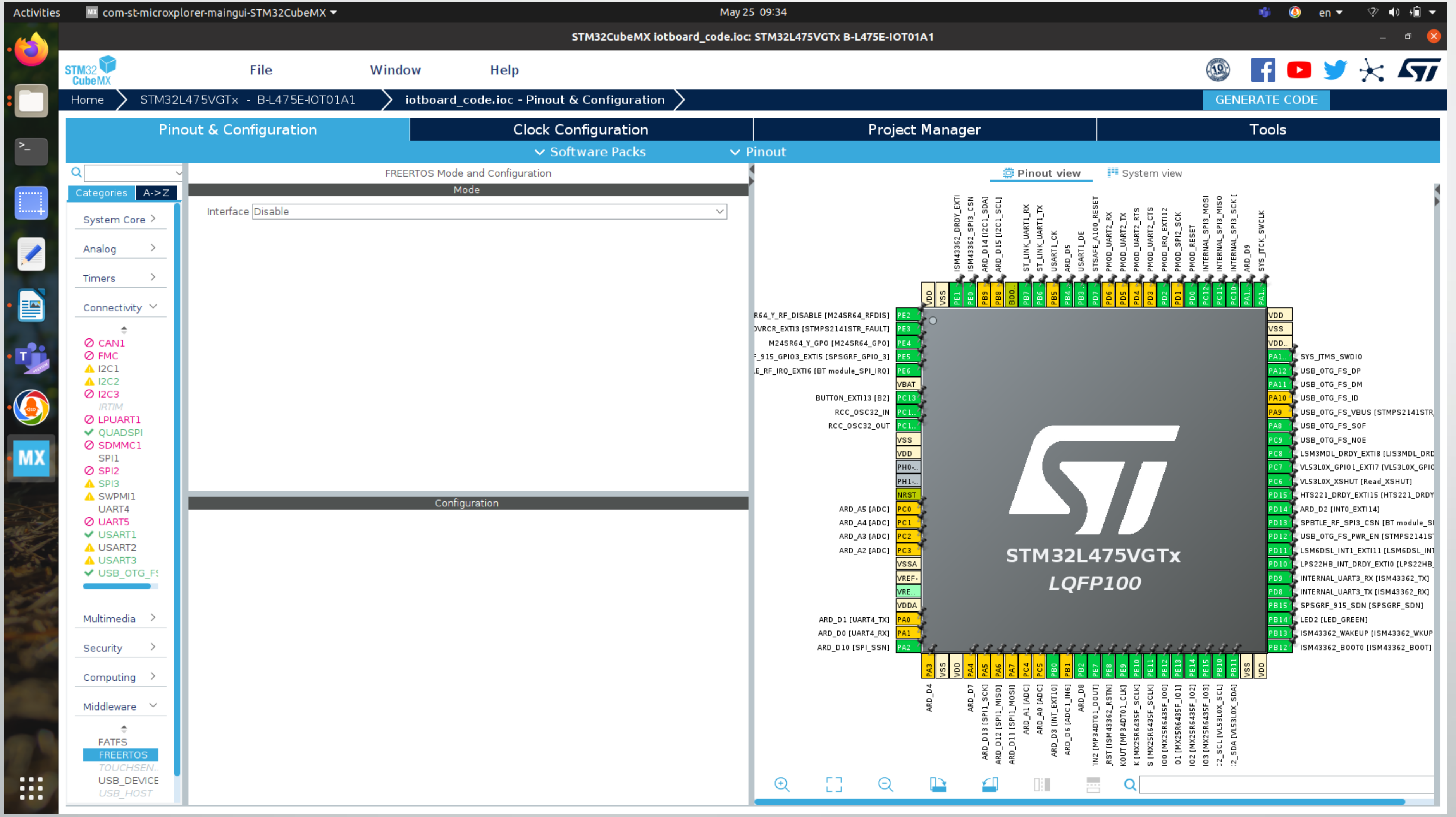
- Clone the microros build related repo and include code related to you target board.
- Implementation is based on application view point.
- Set the pin configuration using stm32cubemx tool, choose connectivity and rtos as freertos.
- Generate the code using the stm32cubemx code generator.
- Add posix 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.

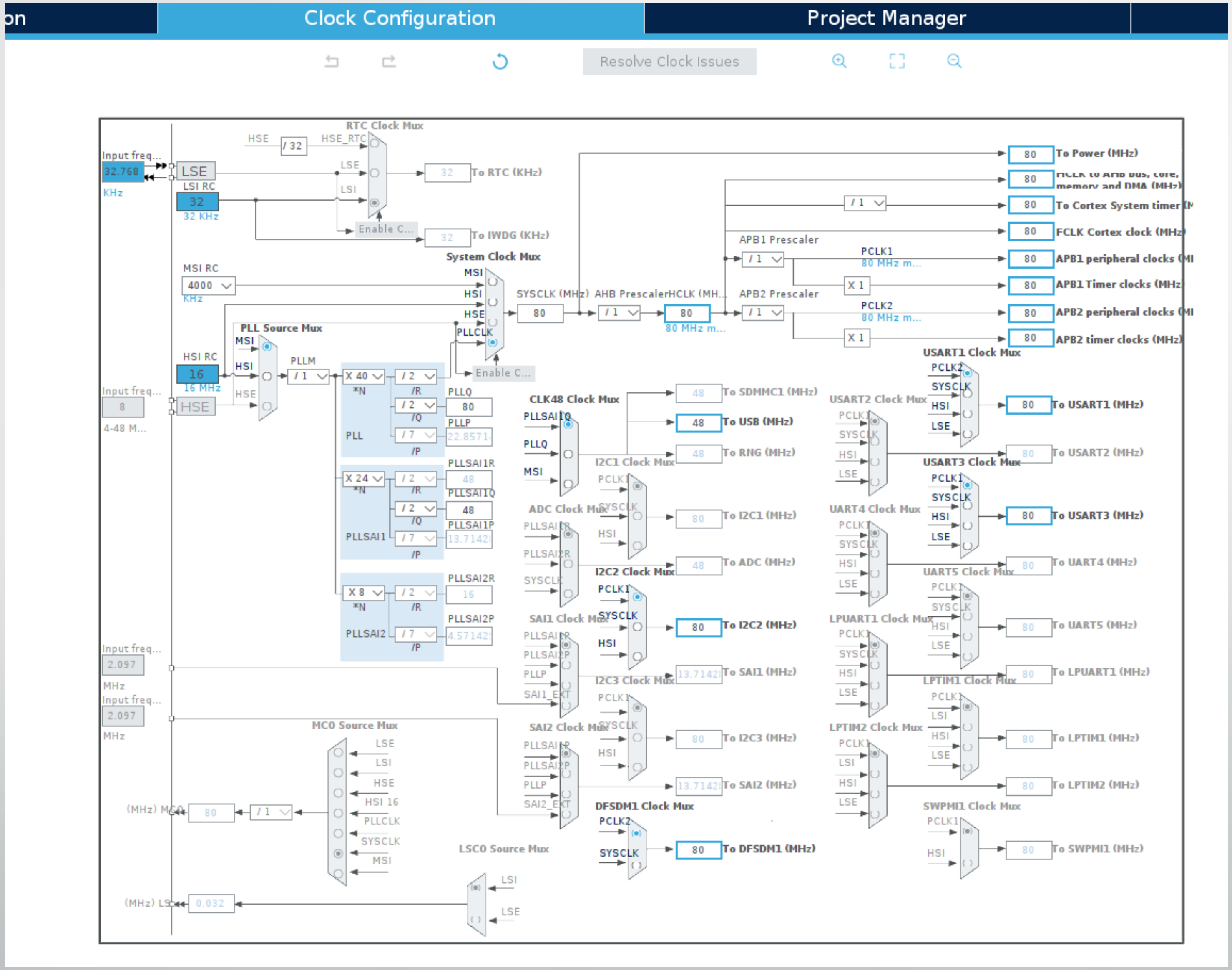
Implementation



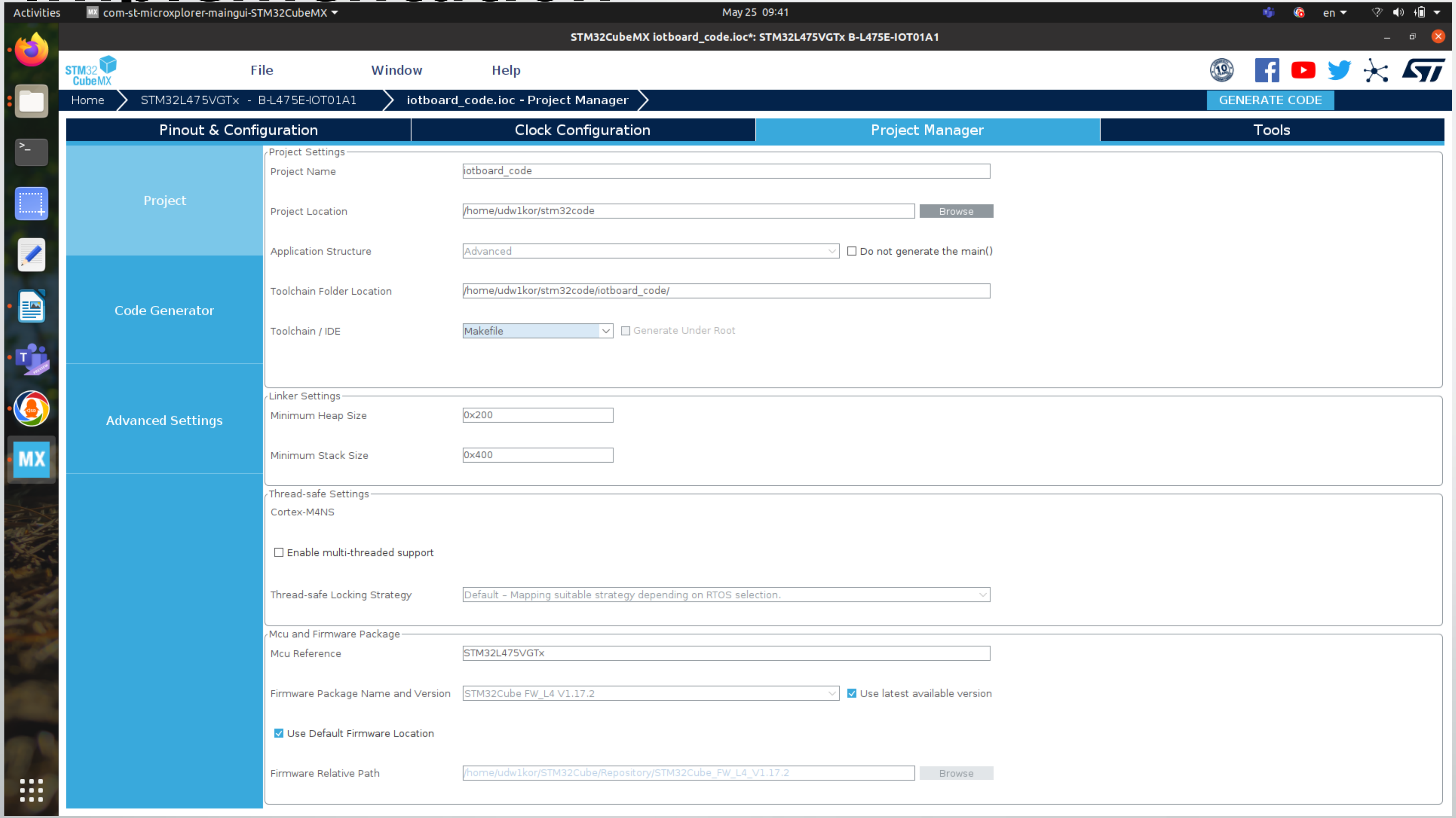
Implementation

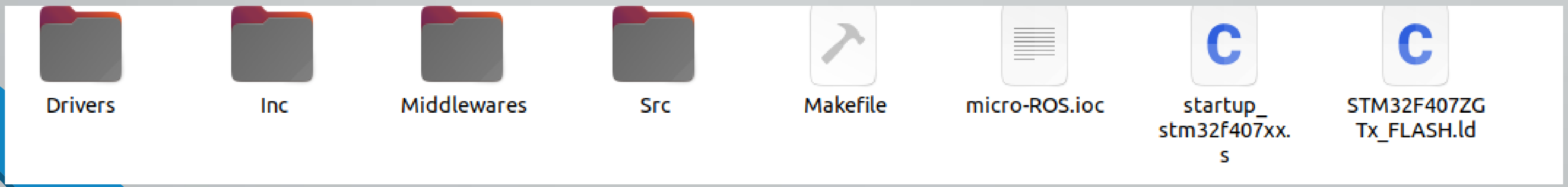


Implementation

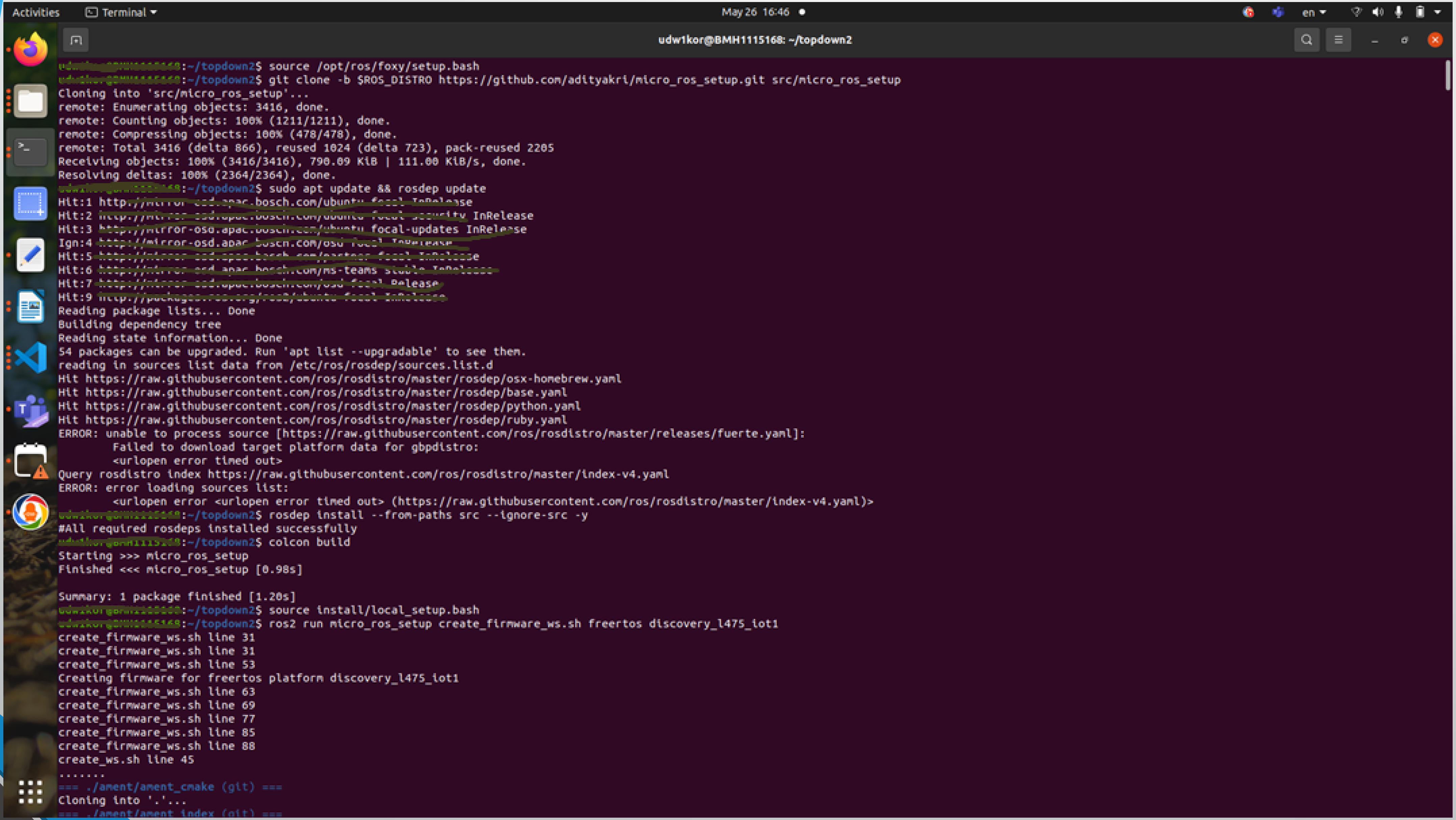


Implementation

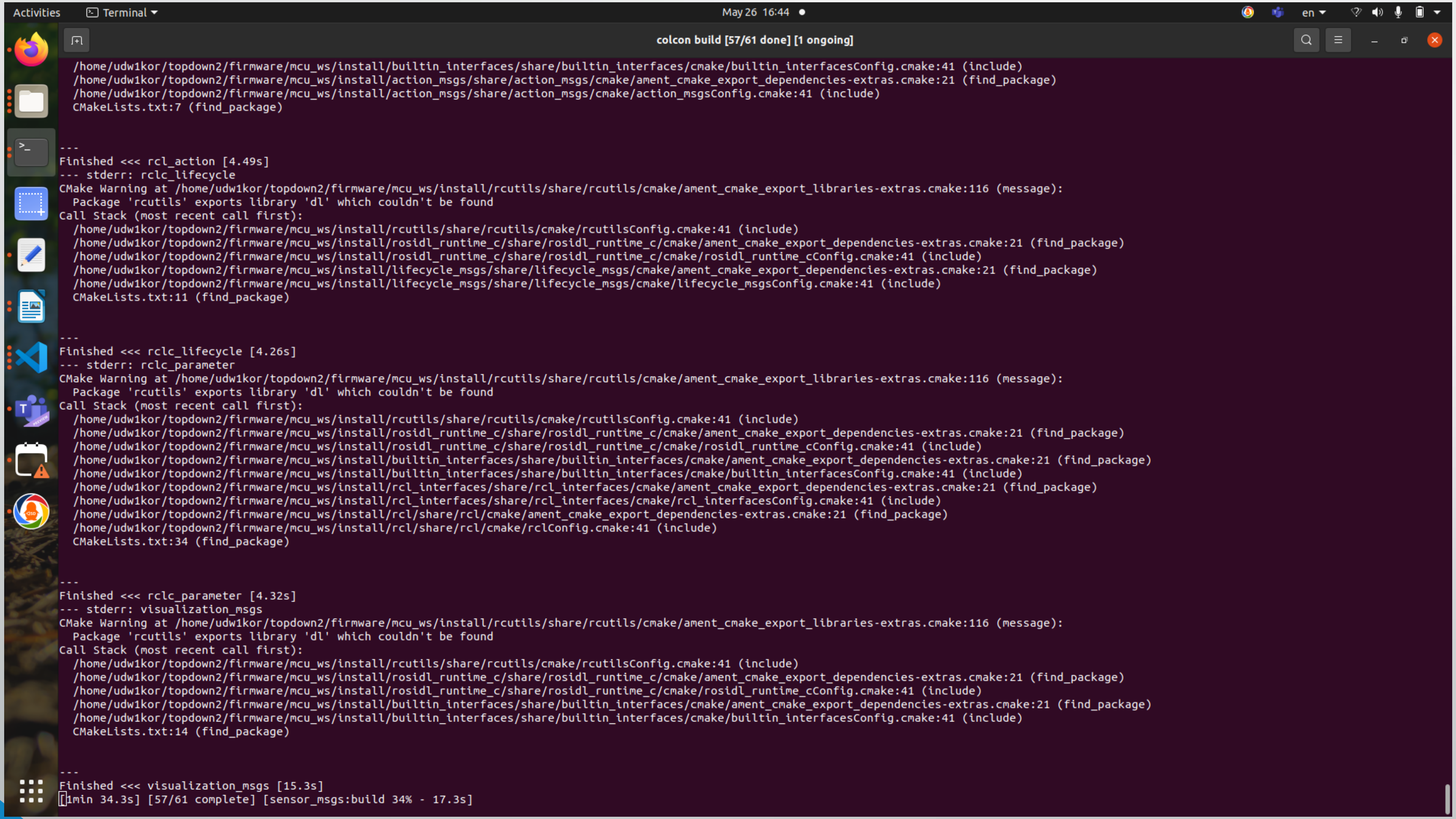




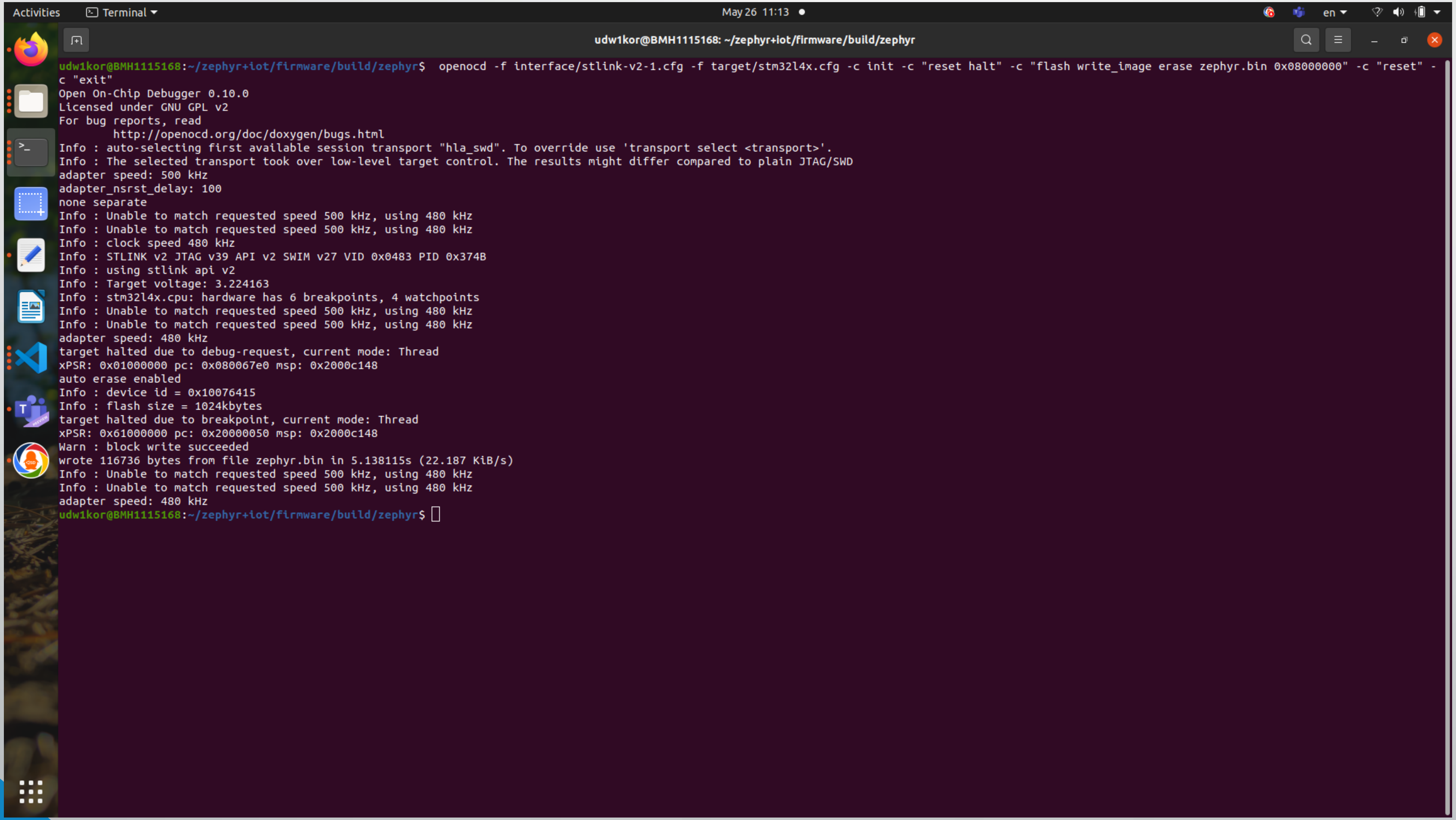
Implementation



Implementation



Implementation



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

a1

ORIGINALITY REPORT

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

PRIMARY SOURCES

1	micro-ros.github.io Internet Source	3%
2	www.bosch.com Internet Source	2%
3	micro.ros.org Internet Source	2%
4	www.coursehero.com Internet Source	2%
5	docs.zephyrproject.org Internet Source	1%
6	www.mytechlogy.com Internet Source	1%
7	www.fiware.org Internet Source	1%
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 Internet Source	1%
11	b-ok.org Internet Source	1%
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 Publication	<1%
13	github.com Internet Source	<1%
14	archive.mu.ac.in Internet Source	<1%
15	webthesis.biblio.polito.it Internet Source	<1%
16	scholars.lib.ntu.edu.tw Internet Source	<1%

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