Swarm Intelligence Programming Framework

- Thesis Report -

S.M Ekanayake
H.M.K Madhushanka
A.L.H.E Perera

Department of Computer Engineering
University of Peradeniya

Final Year Project (courses CO421 & CO425) report submitted as a requirement of the degree of
B.Sc.Eng. in Computer Engineering

February 2023

Supervisors: Dr. Roshan Ragel (University of Peradeniya), Dr. Isuru Nawinne (University of Peradeniya) , Dr. Mahanama Wickremasinghe (University of Peradeniya), Dr. Sewwandi Nanayakkare (University of Peradeniya)

This research is dedicated to all the researchers in the field of swarm robotics who tirelessly work to develop innovative solutions for complex problems. It is also dedicated to the Pera Swarm Project of the University of Peradeniya and its team members who have contributed to the advancement of swarm robotics research and development.
I would also like to express my heartfelt appreciation to my supervisors who have provided invaluable support and guidance throughout the implementation of this research. Their expertise, encouragement, and feedback have been instrumental in shaping this research and bringing it to fruition.
Finally, I would like to dedicate this research to all the individuals who are passionate about swarm robotics and are committed to pushing the boundaries of what is possible. May this research serve as a source of inspiration and motivation for future endeavors in this exciting field.

Declaration

I/We hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my/our own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgments.

S.M Ekanayake
H.M.K Madhushanka
A.L.H.E Perera
February 2023

Acknowledgements

The past research groups who served as our supervisors have our sincere gratitude for their unwavering support and important advice. Their knowledge and perceptions have greatly influenced our research and helped us to accomplish our objectives.

We also want to express our gratitude to the Pera Swarm Project team's earlier developers and researchers, whose innovative work opened the ground for our investigation. Their commitment and diligence have served as an example to me, and their efforts have helped the area of swarm robots advance.

In particular, we would like to thank former members for their contributions. Our study has a solid foundation because to their tremendous efforts in creating the Pera Swarm Project, and we are incredibly appreciative of what they have done.

Last but not least, we would want to thank everyone of my family, friends, and coworkers for their support throughout this journey. We could not have accomplished this without your help, and your words of support and encouragement mean the world to me.

Once more, we would like to express our gratitude to the Pera Swarm Project team's former engineers and researchers as well as our supervisors. My ability to accomplish our study objectives has been greatly aided by their direction, assistance, and contributions.

Abstract

This is where you write your abstract ...

The abstract is the condensed form of your report. Abstract is a very important part of a report. The abstract is often the last item that you write, but the first thing people read when they want to have a quick overview. We suggest you leave writing the abstract to the end, because you will have a clearer picture of all your findings and conclusions.

The abstract is usually about one paragraph. It should have around 200 words. The abstract should cover all the essential elements of the report, namely the background, aim or purpose of research, method used, findings/results and conclusion. Each of the part can consist about 1 to 2 sentences. The abstract should not contain any information not included in the report. The abstract usually does not include any referencing.

Here are some steps you may follow to write a good abstract.

- First re-read your report for an overview. While reading each section condense the information in each down to 1-2 sentences.
- Next read these sentences again to ensure that they cover the major points in your report.
- Ensure you have written something for each of the essential element ; background, aim or purpose of research, method used, findings/results and conclusion.
- Check the word length and further reduce your words if necessary by cutting out unnecessary words or rewriting some of the sentences into a single, more brief sentence.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

ABC	Artificial Bee Colony Optimization
ACO	Ant Colony Optimization
CLI	Command Line Interface
HTTP	Hyper Text Transfer Protocol
MQTT	MQ Telemetry Transport
PIO	Platform IO
PSO	Particle Swarm Optimization

Chapter 1

Introduction

1.1 Background

Swarm robotics is a branch of robotics that examines how big hordes of relatively simple robots behave when they cooperate to achieve a common objective. The behavior of social insects like ants, bees, and termites, which display complex collective behavior patterns despite the limited cognitive skills of individual individuals, served as inspiration for the idea. The development of algorithms and control systems that allow swarms of robots to coordinate their actions decentralized, without relying on a central coordinator, is the main goal of swarm robotics. This enables them to display emergent behaviors, in which the swarm's collective behavior is superior to the sum of its component parts. Also Swarm robotics is the study of how to use local rules to control huge groups of robots. All the basic behaviors and swarm robotic-related experiments are stated here. For a very long time, the collective behaviors [**?**] of social insects were thought to be weird and mysterious. Researchers have shown in recent years that individuals can exhibit such complicated behaviors without the assistance of advanced information or representations. The concept of locality is the basis for this interaction, and neither individual is aware of the bigger picture. The many experimental platforms, such as robotic platforms and simulators, employed in the most relevant swarm-robotic studies developed in order to study the swarm behaviors.

1.2 The problem

One of the main challenges of programming individual robots separately in swarm robotics is the complexity that arises from the interactions between the robots. Each robot is programmed to follow a set of rules or behaviors, which can be influenced by the behavior of other robots in the swarm. As a result, it can be difficult to predict how the robots will behave collectively, and how they will react to changes in the environment or in the behavior of other robots. Another challenge is the need for communication and coordination between the robots. In order to achieve a collective goal, the robots must be able to communicate with each other and coordinate their actions. This can be particularly difficult in large and complex swarms, where individual robots may have limited sensory capabilities or processing power.

To program robots in a swarm, one typically uses a decentralized approach, where each robot has its own set of rules or behaviors that guide its actions. These rules may be simple, such as avoiding obstacles or following a gradient, or more complex, such as cooperating with other robots to perform a task. The interactions between the robots may be modeled using mathematical models, such as game theory or swarm intelligence, to predict their behavior.

One of the challenges of programming robots in a swarm is ensuring that they are able to communicate and coordinate their actions effectively. This requires the use of communication protocols that allow the robots to share information about their state and the environment. The communication protocols may be based on wireless communication, such as Bluetooth or Wi-Fi, or on physical communication, such as infrared or sound signals.

Another challenge is ensuring that the robots are able to adapt to changes in the environment or in the behavior of other robots. This may require the use of learning algorithms, such as reinforcement learning or evolutionary algorithms, that allow the robots to adjust their behavior over time.

Overall, programming robots in a swarm requires a deep understanding of the interactions between the robots and the environment, as well as the ability to model and predict their behavior. It also requires the use of sophisticated algorithms and communication protocols to ensure that the robots are able to work together effectively. Despite the challenges, swarm robotics has the potential to revolutionize many fields, such as agriculture, manufacturing, and search and rescue, by providing scalable, robust, and flexible solutions to complex problems.

1.3 The proposed solution

As a solution for the problem a programming framework can be proposed. Programming framework is a collection of tools, libraries, and APIs that provide a set of abstractions and common patterns for developing software. In the context of swarm robotics, a programming framework may provide a set of high-level abstractions for modeling the interactions between the robots, as well as tools for simulating and testing the behavior of the swarm.

A programming framework can provide several benefits in swarm robotics. First, it can reduce the complexity of developing software for swarm robotics by providing pre-built modules and abstractions that can be reused across different applications. This can help to speed up the development process and reduce the likelihood of errors.

Second, a programming framework can provide a consistent and standardized approach to developing swarm robotics software. This can make it easier for different developers to collaborate and share code, as well as making it easier to maintain and evolve the software over time.

However, while a programming framework can provide some benefits in swarm robotics, it is unlikely to eliminate the need for complex programming and individual attention to each robot. This is because swarm robotics involves designing and implementing decentralized systems, where each robot has its own set of rules and behaviors that interact with those of the other robots. These interactions can be highly complex, and it may be difficult to capture all of the nuances of the system in a single programming framework.

In addition, different applications of swarm robotics may have different requirements, and a one-size-fits-all programming framework may not be appropriate for all applications. Therefore, while a programming framework can provide some benefits in swarm robotics, it is still important to have skilled and knowledgeable developers who can implement and customize the software to meet the specific needs of the application.

A visual programmer can be a useful tool in a swarm robotics framework as it can simplify the process of programming and testing the behavior of the robots in the swarm. A visual programmer is a software tool that allows developers to create software using a graphical interface rather than traditional text-based coding.

In swarm robotics, a visual programmer can allow developers to easily design and test the behavior of individual robots, as well as the interactions between the robots. This is because a visual programmer typically provides a set of drag-and-drop or point-and-click tools that allow developers to create and connect different behaviors or modules, without the need for manual coding.

Visual programming can also make it easier to understand the behavior of the swarm as a whole. This is because a visual programming framework can provide a graphical representation of the interactions between the robots, which can be easier to understand than a complex set of code.

Another advantage of using a visual programming framework in swarm robotics is that it can facilitate collaboration among different team members with different skill sets. A visual programming framework can provide a user-friendly interface that makes it easier for non-programmers to contribute to the development process. This can enable more people to participate in the development of swarm robotics applications, leading to more diverse and innovative solutions.

Overall, a visual programming framework can be a valuable tool in swarm robotics as it can simplify the process of programming and testing the behavior of the robots, make it easier to understand the behavior of the swarm as a whole, and facilitate collaboration among different team members.

As for the project deliverables and milestones, following are proposed and developed so far. Reducing the individual configuration times considering each of the robots separately develop completely a remote cross compiler hosted on supporting linux containers with a frontend and in the backend. Developed a color changing algorithm. Code generation implementation for the above algorithms implemented, and uploading the program into robots.

Chapter 2

Related work

A group of tools and methods called swarm intelligence programming frameworks are used to create and improve swarm intelligence algorithms. The implementation, testing, and comparison of various swarm intelligence techniques, such as particle swarm optimization (PSO), ant colony optimization (ACO), and artificial bee colony optimization, are made possible by these frameworks (ABC).

The creation of swarm intelligence programming frameworks has completely changed the field of swarm intelligence by enabling scientists to create and test sophisticated algorithms more quickly and effectively. Researchers can study and examine the behavior of swarm algorithms using these frameworks' various features, which include visualization tools, simulation environments, and optimization algorithms.

A number of swarm intelligence programming frameworks have been created in recent years, each with its own distinct characteristics and capabilities. Many applications, such as optimization issues, image processing, robotics, and data mining, have made use of these frameworks.

As a result of the popularity of swarm intelligence programming frameworks, standardized benchmarking platforms have been created to assess the effectiveness of swarm intelligence algorithms. These benchmarks, which offer a common platform for assessing and contrasting the performance of various swarm intelligence algorithms, have grown to be an indispensable tool for academics.

Swarm intelligence programming frameworks, in general, have been essential in developing the subject of swarm intelligence by giving researchers and developers the resources and tools they need to create and improve sophisticated swarm intelligence algorithms.

Among those of the available programming frameworks following were well identified well in the industrial usage and related robotics researches.

2.1 SwarmOps

SwarmOps[1] is a swarm intelligence optimization toolbox created for MATLAB that offers academics and developers a variety of optimization algorithms. The framework has been applied in a variety of domains, including robotics, where it has shown to be a crucial tool for applications based on swarm intelligence.

The framework uses swarm intelligence methods to optimize several robotics-related problems, including particle swarm optimization (PSO), differential evolution (DE), and artificial bee colony (ABC). SwarmOps offers an easy-to-use interface for putting these algorithms into practice as well as visualization tools for studying how swarm algorithms behave.

Many robotics applications, including as cooperative control, task distribution, and path planning, have exploited SwarmOps. SwarmOps can be used for path planning to optimize a robot's path through a changing environment. SwarmOps can be used in task distribution to distribute jobs to a swarm of robots based on their resources and skills. SwarmOps can be used in cooperative control to optimize a swarm of robots' behavior in order to accomplish a shared objective.

The framework has also been used to optimize the coordination of a swarm of robots in multi-robot systems. SwarmOps has proven to be a useful tool for creating swarm intelligence-based solutions that are efficient and successful for a variety of robotics applications.

Overall, SwarmOps provides a valuable tool for researchers and developers working in the field of swarm robotics, allowing them to optimize complex problems using a range of swarm intelligence algorithms. With its user-friendly interface and visualization tools, SwarmOps has become an essential tool for developing efficient and effective swarm intelligence-based solutions for robotics applications.

2.2 Swarm-Bench

A benchmarking toolkit for swarm robotics called SWARM-BENCH[2] was created to offer a common platform for assessing swarm intelligence algorithms. A simulator and a selection of test scenarios are included in the toolkit for assessing swarm algorithms.

The SWARM-BENCH architecture enables researchers to compare the effectiveness of various swarm algorithms in lab settings on a single platform. This enables researchers to recognize the advantages and disadvantages of various swarm algorithms and create more effective and efficient algorithms.

Many robotics applications, including swarm navigation, swarm coordination, and swarm pattern generation, have employed SWARM-BENCH. It has been used to gauge how well PSO, ACO, and ABC perform among other swarm intelligence algorithms.

2.3 PySwarm

A Python programming framework called PySwarm[3] is used to create and test swarm intelligence systems. Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Differential Evolution (DE) are just a few of the optimization algorithms it offers, along with visualization tools for studying swarm algorithm behavior. PySwarm is a well-liked option for swarm robotics researchers and developers because it is open-source and simple to install and integrate into current Python projects.

For the creation and testing of swarm intelligence algorithms in robotics, PySwarm offers a variety of tools and resources. The framework offers a variety of optimization techniques for testing and comparing various swarm intelligence algorithms and allows programmers to design their own swarm intelligence algorithms using Python code. Moreover, PySwarm offers visualization tools for examining swarm algorithm activity, which are helpful for comprehending how swarm algorithms function and locating potential areas for optimization.

Robotics applications like as path planning, navigation, and swarm coordination have all made use of PySwarm. PySwarm has been used, for instance, to create a swarm of robots that can self-organize and coordinate their movement in challenging conditions. Moreover, PySwarm has been used to optimize the performance of robotic swarms, boosting their efficiency and efficacy in executing complicated tasks. Overall, PySwarm is an effective platform for swarm robotics researchers and developers, offering a variety of tools and resources for creating and refining sophisticated swarm intelligence systems.

2.4 NetLogo

A well-liked programming framework for creating agent-based models and multi-agent systems is NetLogo[4]. It provides a user-friendly interface for building and simulating complicated systems, making it a popular choice for researchers working in the field of robotics. It is the perfect environment for creating and testing swarm robotics systems since NetLogo supports a variety of swarm intelligence techniques, including ant colony optimization (ACO) and particle swarm optimization (PSO).

Support for multi-agent systems is one of NetLogo's standout characteristics. In a single setting, this enables researchers to model and simulate the behavior of several entities, such as robots or autonomous cars. A variety of tools, such as visualization tools for following these agents' behaviour, are available in NetLogo for studying their behavior.

Swarm robotics, in which a group of robots cooperate to complete an objective, is one robotics application where NetLogo has been employed. Swarm robot behavior has been modelled and simulated using NetLogo, allowing researchers to examine and study the behavior of these robots in various contexts. Moreover, it has been applied to create and evaluate control algorithms for autonomous robots.

Chapter 3

Methodology

3.1 Conceptual design

A remote cross compiler PIO CLI[5] is a tool that allows developers to compile code for different platforms and architectures, without the need for physical access to the target devices. By deploying this tool in a Docker container, developers can easily manage and scale their cross-compiling environment, while also ensuring consistency across different development environments. In this design, the PIO CLI tool will be integrated with a central server using MQTT[6], enabling developers to easily deploy compiled binaries to a swarm of robots.

The central server will act as the primary communication hub between the swarm of robots and the developer's cross-compiling environment. The MQTT protocol will be used to establish a lightweight messaging system, allowing robots to receive signals from the central server indicating when new binary files are available for download. Once a robot receives the signal, it will initiate an HTTP download request to the central server, which will transfer the compiled binary to the robot for execution.

To ensure secure communication between the central server and the swarm of robots, authentication and authorization mechanisms will be implemented. This will prevent unauthorized access to the central server and ensure that only authorized robots can download and execute the compiled binaries. Additionally, the central server will maintain a log of all communication activities, enabling developers to monitor and troubleshoot any issues that may arise.

Overall, the deployment of a remote cross compiler PIO CLI in a Docker container and integration with a central server using MQTT offers a scalable and efficient solution for deploying compiled binaries to a swarm of robots. This design enables developers to easily manage and scale their cross-compiling environment, while also ensuring secure and reliable communication with the swarm of robots.

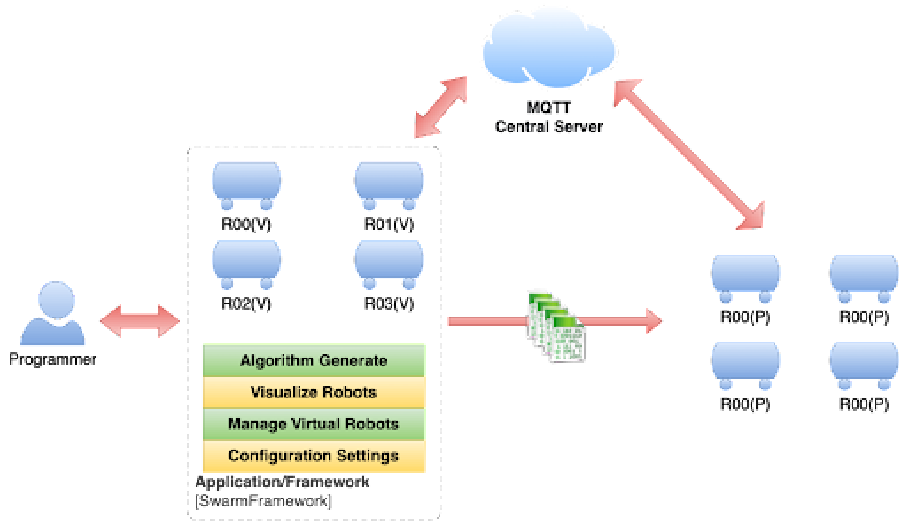


Fig. 3.1 Conceptual Design

3.2 Methodological approach

To create a central server that provides binaries when HTTP[7] requests are received, we started by setting up an MQTT broker to handle communication between the central server and the swarm of robots. We used the Mosquitto MQTT broker, which provides a lightweight and scalable solution for handling MQTT communication. We then set up an HTTP server using Node.js and Express, which will be used to serve binary files to the swarm of robots when HTTP requests are received.

Next, we created a script that uses the PlatformIO (PIO) CLI tool to compile the source code into a binary file suitable for the target device. We ensured that the appropriate firmware was selected based on the target device and architecture. We also implemented a version control system to ensure that the latest version of the compiled binary is always available on the central server.

To deploy the system into a Docker[8] container, we first created a Dockerfile that specified the base image and installed the required dependencies, including the MQTT broker, Node.js, and the PIO CLI tool. We then built the Docker image using the docker build command and ran the container using the docker run command. We also configured the container to mount a volume to the central server's file system, ensuring that the latest version of the compiled binary is always available to serve HTTP requests.

Finally, we developed a React frontend web application that enables developers to interact with the central server using a user-friendly interface. The application allows developers to upload their source code and compile it using the PIO CLI tool, select the appropriate firmware for the target device, and deploy the compiled binary to the central server. The application also enables developers to monitor the status of the central server, view logs of HTTP requests and MQTT communication, and manage the version control of the compiled binaries.

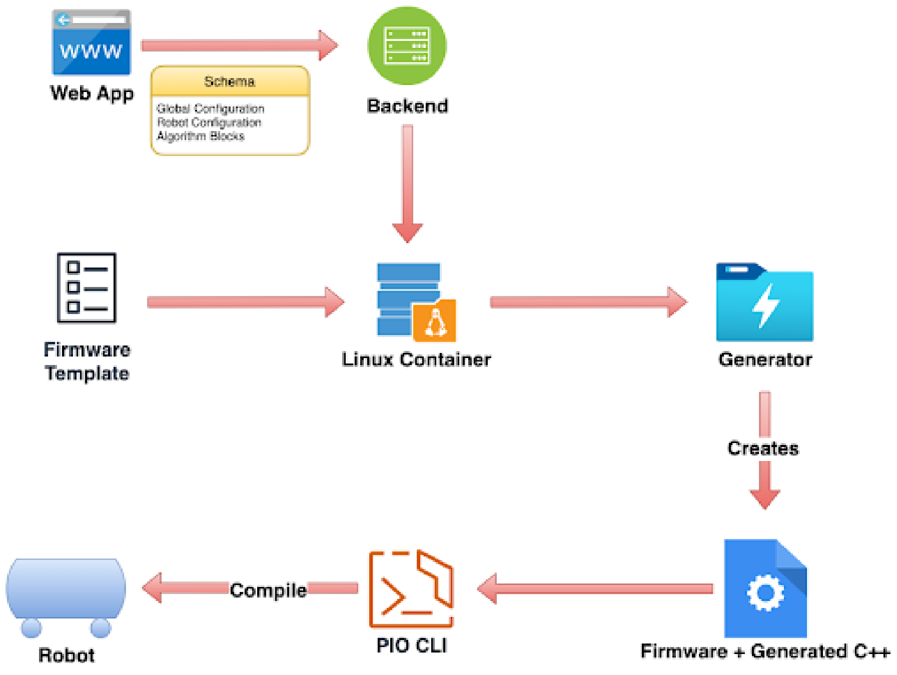


Fig. 3.2 Software Architectural Diagram

Overall, this methodology provided a scalable and efficient solution for managing and deploying compiled binaries to a swarm of robots. The integration of the PIO CLI tool, MQTT broker, and HTTP server into a Docker container enabled developers to easily manage and scale their cross-compiling environment, while the React frontend web application provided a user-friendly interface for managing the central server and interacting with the swarm of robots.

3.2.1 Development of the Cross Compiler

To develop a remote cross compiler using the PIO CLI tool, we first defined a set of configuration schemas for the different target platforms and architectures that the cross compiler will support. These configuration schemas include parameters such as the device type, target board, and firmware version, as well as any additional configuration options specific to the target device. These configuration schemas are used to generate the appropriate firmware binaries for the target device.

We then integrated the PIO CLI tool into a web application that is deployed inside a Docker container. The web application provides a user-friendly interface for developers to interact with the cross compiler and generate firmware binaries for their target devices. The application allows developers to upload their source code, select the appropriate configuration schema for the target device, and compile the source code into a firmware binary using the PIO CLI tool.

The web application also enables developers to manage the version control of the compiled binaries, ensuring that the latest version is always available for deployment to the swarm of robots. The compiled binaries are stored in a centralized repository, which can be accessed by the swarm of robots using HTTP requests.

Overall, this methodology provides a scalable and efficient solution for generating firmware binaries for swarm robots using the PIO CLI tool. By integrating the PIO CLI tool into a web application and deploying it inside a Docker container, we provide a user-friendly interface for developers to interact with the cross compiler and generate firmware binaries for their target devices. This approach also enables developers to easily manage the version control of the compiled binaries, ensuring that the latest version is always available for deployment to the swarm of robots.

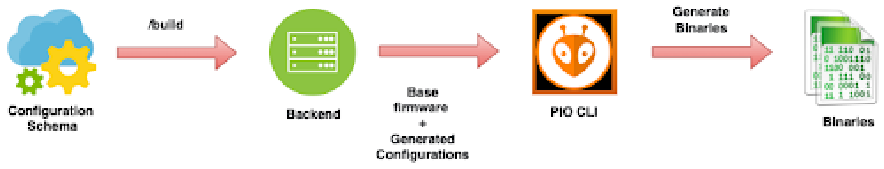


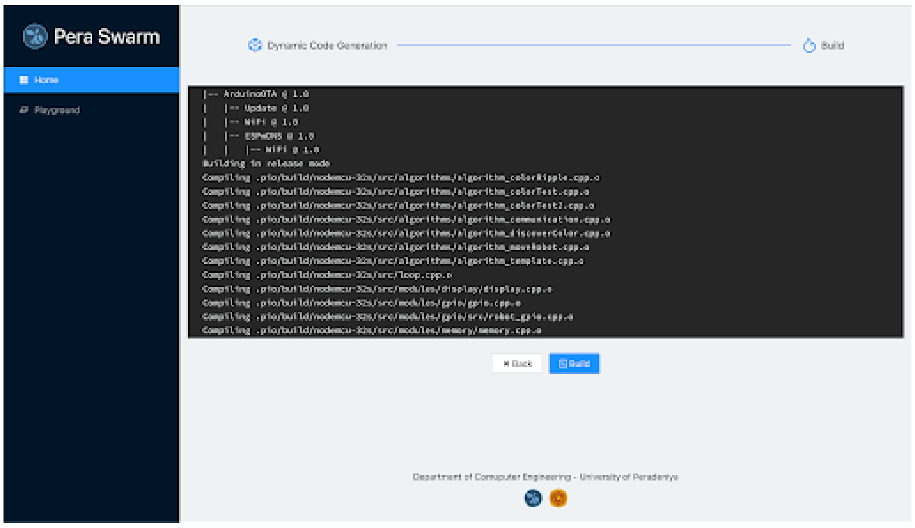
Fig. 3.3 Remote cross compiler design workflow

3.2.2 Development of Visual Programmer tool

There are numerous crucial phases involved in creating a visual programming tool inside of a web application that integrates Google Blockly[9] to create C++ codes and further improve swarm algorithms. Identifying the characteristics and functions necessary to create the swarm algorithms is the first step in defining the needs and objectives of the tool. The development team can start designing and developing the web application once the requirements have been established, utilizing an appropriate framework and programming language.

The next step is to integrate Google Blockly, an open-source visual programming editor, into the web application. This allows users to visually design and manipulate code blocks to generate C++ code for their swarm algorithms. The tool must also be designed to support remote cross-compilation using the PlatformIO (PIO) CLI, allowing the tool to build binaries for different platforms. To achieve this, the development team must design the tool to interact with the PIO CLI, passing the generated C++ code to it for remote compilation.

Once the web application is developed and integrated with Google Blockly and the PIO CLI, the final step is to test and refine the tool to ensure it meets the desired requirements and goals. This involves testing the tool's functionality, usability, and performance, and making necessary adjustments based on user feedback.


Pera Swarm
Home
Playground
Dynamic Code Generation
Build
Back
Build
Department of Computer Engineering - University of Peradeniya

```
|-- ArduinoOTA @ 1.0
|   |-- Update @ 1.0
|   |-- WiFi @ 1.0
|   |-- ESPmDNS @ 1.0
|   |   |-- WiFi @ 1.0
Building in release mode
Compiling .pio/build/nodemcu-32s/src/algorithms/algorithm_colorRipple.cpp.o
Compiling .pio/build/nodemcu-32s/src/algorithms/algorithm_colorTest.cpp.o
Compiling .pio/build/nodemcu-32s/src/algorithms/algorithm_colorTest2.cpp.o
Compiling .pio/build/nodemcu-32s/src/algorithms/algorithm_communication.cpp.o
Compiling .pio/build/nodemcu-32s/src/algorithms/algorithm_discoverColor.cpp.o
Compiling .pio/build/nodemcu-32s/src/algorithms/algorithm_moveRobot.cpp.o
Compiling .pio/build/nodemcu-32s/src/algorithms/algorithm_template.cpp.o
Compiling .pio/build/nodemcu-32s/src/loop.cpp.o
Compiling .pio/build/nodemcu-32s/src/modules/display/display.cpp.o
Compiling .pio/build/nodemcu-32s/src/modules/gpio/gpio.cpp.o
Compiling .pio/build/nodemcu-32s/src/modules/gpio/src/robot_gpio.cpp.o
Compiling .pio/build/nodemcu-32s/src/modules/memory/memory.cpp.o
```

Fig. 3.4 Visual Programming tool

3.2.3 Development of Over the Air uploading feature

The system for downloading binaries to swarm robots over the air through wifi modules using the PlatformIO (PIO) CLI and a central server involved several key components. First, each swarm robot was equipped with a wifi module to enable wireless communication. Next, the central server was set up to communicate with the swarm robots using MQTT messaging protocol.

When the swarm robots were ready to download new binaries, central server would send a signal to the swarm robots using MQTT to initiate the download process. The central server would then use the PIO CLI to compile and build the binaries for the specific robot based on its hardware and platform specifications.

Once the binaries were built, they were packaged and sent back to the swarm robot via MQTT messaging. The robot's wifi module would then receive the binaries over the air and save them to its memory. The robot would then be ready to execute the new code, which could include updated algorithms or new functionalities.

Throughout the process, the central server would monitor the progress of the download and installation, ensuring that each robot received the correct binary and that the process completed successfully. This allowed for seamless and efficient updates to the swarm robots without the need for physical connections or manual updates.

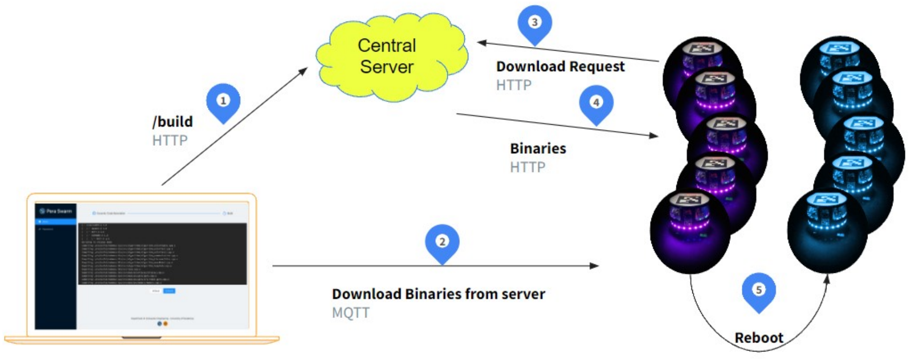


Fig. 3.5 Over the Air upload feature diagram

Chapter 4

Experimental Setup and Implementation

To set up an experimental scenario using Docker, a web application with Blockly for visual programming, and the PlatformIO (PIO) CLI to create and upload binaries to swarm robots over the air using MQTT signals, several key steps were involved.

First, the Docker container was set up to host the web application and integrated with Google Blockly for visual programming. This allowed users to design and manipulate code blocks to generate C++ code for their swarm algorithms. The container was also configured with the PIO CLI to allow for remote compilation and building of binaries for the swarm robots.

Next, a scenario was created for uploading the color changing algorithm to two robots over the air. The robots were equipped with wifi modules to enable wireless communication. The web application was designed to communicate with the robots over HTTP to initiate the upload process. Once the upload was initiated, the web application used the PIO CLI to compile and build the binaries for the specific robots based on their hardware and platform specifications.

Once the binaries were built, they were packaged and sent back to the robots over the air using MQTT messaging. The robots' wifi modules received the binaries and saved them to their memory. The robots then executed the new code, changing their color based on the uploaded algorithm.

Throughout the process, the web application monitored the progress of the upload and installation, ensuring that each robot received the correct binary and that the process completed successfully. The web application also sent MQTT signals to the robots to notify them of the successful upload and installation of the new code.

4.1 Pitfalls

Despite its advantages, the implementation of downloading binaries over the air through WiFi modules using PIO CLI with a central server and MQTT communication also has some disadvantages that need to be considered.

One of the main disadvantages is the issue of dependencies when building the existing firmware of robots. Developers must ensure that all the necessary libraries and dependencies are included in the firmware to avoid issues when executing the algorithm on the robot platform. This can be time-consuming and may require additional testing and troubleshooting to ensure the firmware is functioning correctly.

Another disadvantage is that Google Blockly visual programming tools do not support C++ by default. This means that developers must either use a different visual programming tool or develop the algorithm using a traditional programming language. This can be a significant barrier to entry for those who are not familiar with traditional programming languages and may limit the usability of the system.

The need to upload specific firmware to each robot by OTA over the air can be time-consuming and can potentially lead to issues with connectivity and compatibility. Each robot must be equipped with a WiFi module and firmware that allows it to communicate with the central server and receive OTA updates, which can be a significant cost for large-scale projects.

Overall, while the implementation of downloading binaries over the air through WiFi modules using PIO CLI with a central server and MQTT communication offers several advantages, it is important to consider the potential disadvantages and limitations of the system before implementation. By carefully considering these factors and addressing them as needed, developers can create a powerful and effective system for swarm robotics applications.

Chapter 5

Results and Analysis

5.1 Results

In a swarm robotics system, the ability for robots to download and execute algorithms over the air can be a valuable feature, as it allows for the swarm to adapt to changing conditions and tasks. In this scenario, we explore how a swarm of robots is able to download a color changing algorithm after receiving an MQTT signal, using the implementation of downloading binaries over the air through WiFi modules and the PIO CLI with a central server.

The first step in implementing this system is to create the color changing algorithm using a suitable programming language, such as C++. The algorithm must be designed to receive input from the robots' sensors and change the color of their LEDs accordingly. Once the algorithm is designed and tested, it must be compiled into a binary file using the PIO CLI, which provides support for cross-compiling and uploading binaries to the robots' platforms.

Next, a central server must be set up to communicate with the robots over WiFi using the MQTT protocol. The server must be configured to receive messages from the robots, indicating their current state and any other relevant information. When the server receives a message indicating that a new color changing algorithm is required, it must send an MQTT signal to the robots, triggering them to download the binary file from the server.

To enable the robots to download the binary file, each robot must be equipped with a WiFi module and firmware that allows it to communicate with the central server. When a robot receives an MQTT signal from the server indicating that a new algorithm is available, it must use the PIO CLI to download the binary file and update its firmware.

This process can be achieved through the use of OTA (over the air) updates, which allow the robots to download the new firmware without the need for physical access.

Once the new firmware is downloaded and installed, the robots will begin executing the new color changing algorithm, adapting to the new conditions and tasks. The robots can continue to communicate with the central server over WiFi, sending updates on their current state and receiving any new algorithms as required.

In conclusion, by using the implementation of downloading binaries over the air through WiFi modules and the PIO CLI with a central server, swarm robots can adapt to changing conditions and tasks by downloading and executing new algorithms. This process involves creating and compiling the new algorithm into a binary file, setting up a central server to communicate with the robots over WiFi using MQTT, and equipping each robot with a WiFi module and firmware that allows it to download and install new firmware over the air. Through this process, the swarm of robots can continue to adapt and perform their tasks effectively.

5.2 Analysis

Swarm robots that can download binaries to the platform over the air through WiFi modules can be implemented using various tools such as pySwarm, SwarmOps, Swarm Bench, and PIO CLI. However, each tool has its own unique features and functionalities that make them suitable for different applications.

pySwarm is a Python-based swarm intelligence optimization library that allows users to design and implement swarm algorithms using Python code. It provides a variety of swarm algorithms and optimization techniques, making it ideal for research and experimentation. However, it does not have built-in support for OTA updates or WiFi communication.

SwarmOps, on the other hand, is a C++ library that provides a range of optimization algorithms and is suitable for large-scale optimization problems. It does have support for OTA updates, but the WiFi communication is not built-in and requires additional modules.

Swarm Bench is a software framework that provides a web-based interface for designing and testing swarm algorithms. It has built-in support for OTA updates and WiFi communication, making it ideal for swarm robotics applications. However, it may be limited in terms of customization and flexibility.

PIO CLI is a command-line interface that allows developers to build, upload, and manage embedded projects using different platforms and frameworks. It has built-in support for OTA updates and WiFi communication, making it suitable for swarm robotics applications. Additionally, with a central server, communications can be established using MQTT protocol, allowing the swarm robots to receive signals and coordinate their actions. PIO CLI is highly customizable and can be used with various platforms and frameworks, making it ideal for large-scale and complex projects.

Chapter 6

Conclusions and Future Works

In conclusion, the implementation of downloading binaries over the air through WiFi modules using PIO CLI with a central server and MQTT communication provides a powerful tool for swarm robotics applications. While there are several other tools available such as pySwarm, SwarmOps, and Swarm Bench, each with their own unique features and functionalities, the implementation using PIO CLI and MQTT communication offers several advantages.

By using PIO CLI, developers can easily build, upload, and manage embedded projects with cross-platform support. The central server and MQTT communication provide a reliable and efficient means of communicating with the swarm robots over WiFi, allowing for remote updates and coordination. Additionally, the ability to download and install new firmware over the air using OTA updates provides a powerful means of adapting the swarm to changing conditions and tasks.

Overall, the implementation of downloading binaries over the air through WiFi modules using PIO CLI with a central server and MQTT communication offers a flexible, customizable, and powerful tool for swarm robotics applications. The ability to update and adapt the swarm robots remotely using OTA updates and MQTT communication allows for efficient coordination and adaptation to changing conditions, making it an ideal solution for large-scale and complex swarm robotics projects.

References

[1] A. Vikhar and A. P. Engelbrecht, "Swarmops: A swarm intelligence optimization toolbox for matlab," *Journal of Open Research Software*, vol. 5, no. 1, 2017.

[2] P. H. V. Lima, E. Ferrante, A. E. Turgut, and M. Dorigo, "Swarm-bench: A benchmarking toolkit for swarm robotics," *Robotics and Autonomous Systems*, vol. 97, pp. 10–20, 2017.

[3] M. A. Elsayed, E. A. Aziz, and A. A. Ewees, "Pyswarm: a python framework for swarm intelligence," *Procedia Computer Science*, vol. 170, pp. 817–824, 2020.

[4] U. Wilensky, "Netlogo," 1999.

[5] PlatformIO, "PlatformIO Core (CLI)." https://docs.platformio.org/en/latest/core/index.html, 2021. Accessed: September 17, 2021.

[6] OASIS Open, "Mqtt version 3.1.1." http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html, 2014. Accessed: March 1, 2023.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1." RFC 2616, June 1999. [Online; accessed 1 March 2023].

[8] "Docker." https://www.docker.com/. Accessed: March 1, 2023.

[9] Google, "Blockly." https://developers.google.com/blockly, 2023.