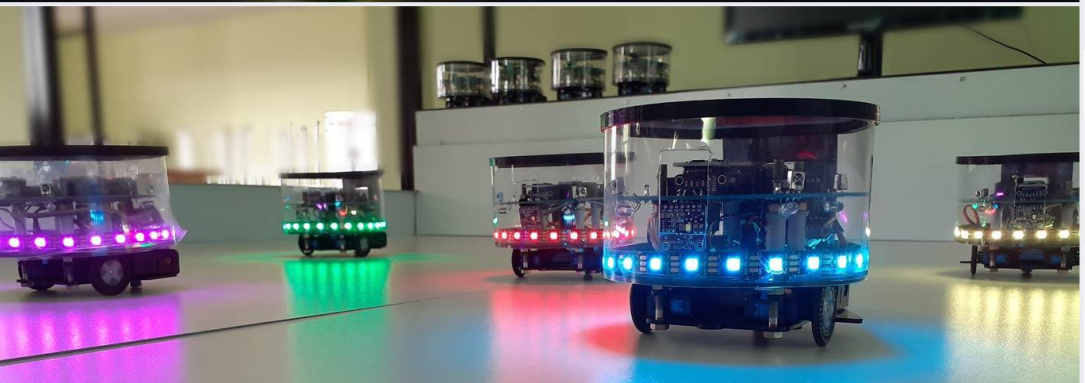
Programming & Compiler Toolchain for Multi-Agent Systems

Group 07
E/17/398 Hashini Wijerathne
E/17/159 Kavinaya Yogendran
E/17/352 Isara Tillekeratne

Supervisors
Dr Isuru Nawinne
Dr Mahanama Wickramasinghe
Prof Roshan Ragel

Introduction

Background and Problem Statement



Swarm Robotics

Inspired by the collective behaviour observed in natural swarms

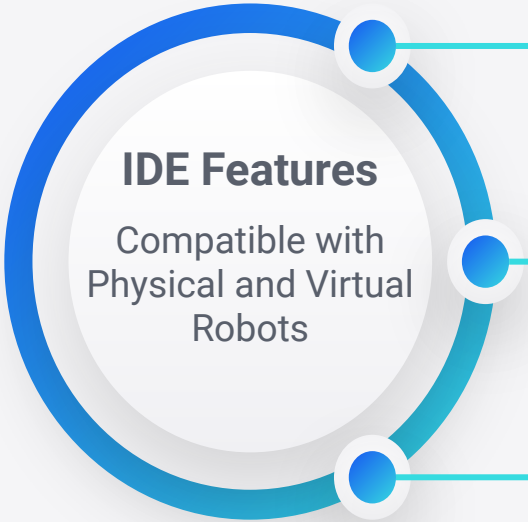
- Autonomous
- Homogeneous
- Locality
- Decentralised

Problems

- Complexity in programming swarm robots
- No block-based programming support
- Limited to software-level simulations rather than comprehensive development libraries
- Limited to a few pre-programmed behaviours
- Lack of support for both physical and virtual robots

02

Methodology



High-level algorithm composition

Block-based visual programming

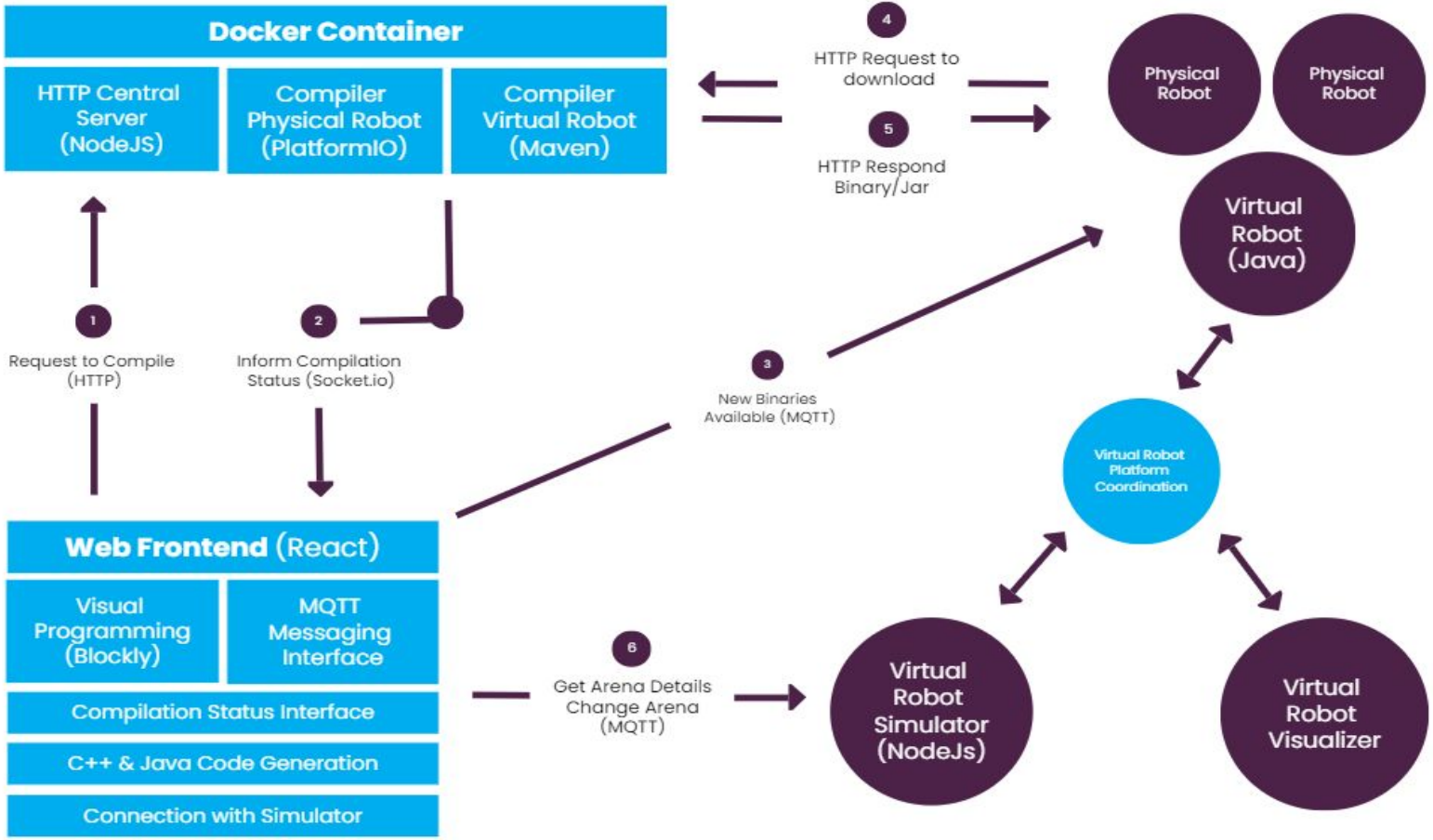
Converts the graphical-level algorithm to executables

Code generation and compilation

OTA code upload and execution

Reprogram multiple robots without physical access

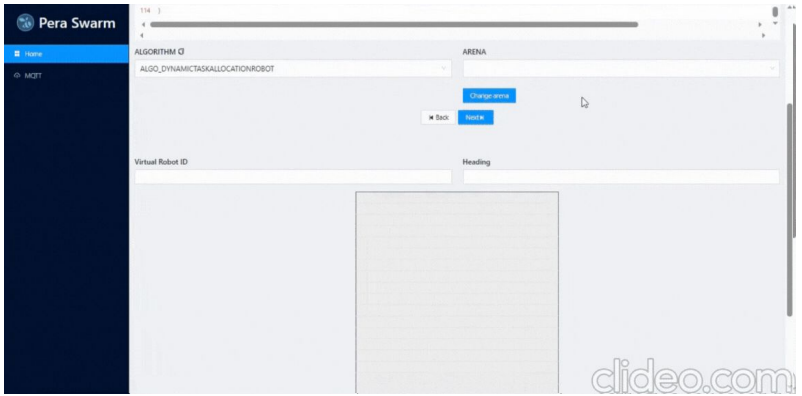
Solution Architecture

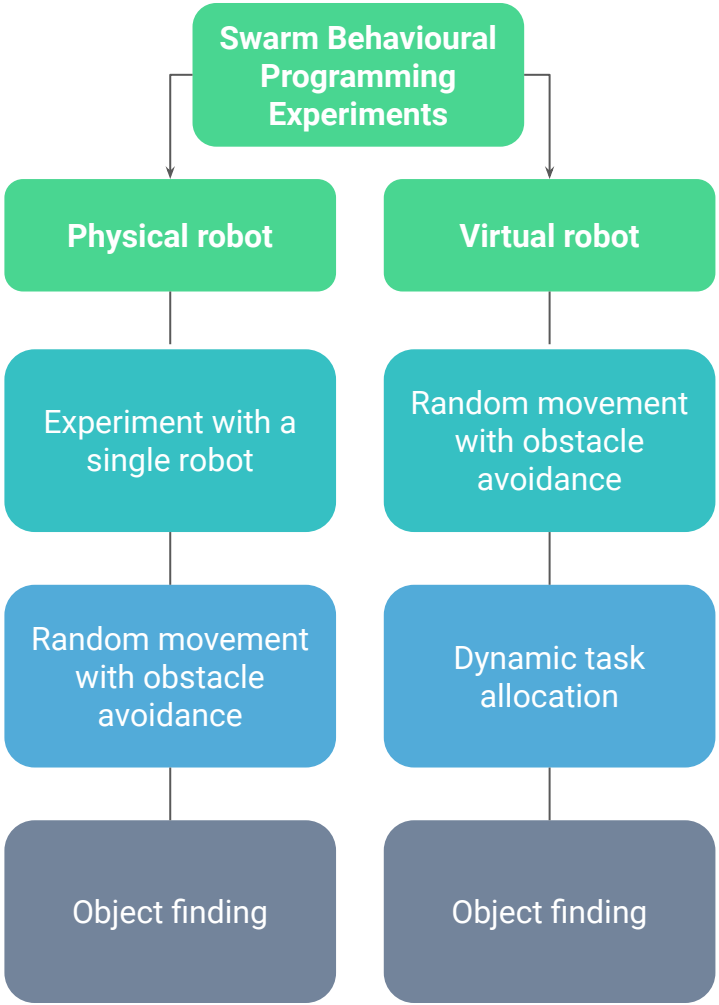


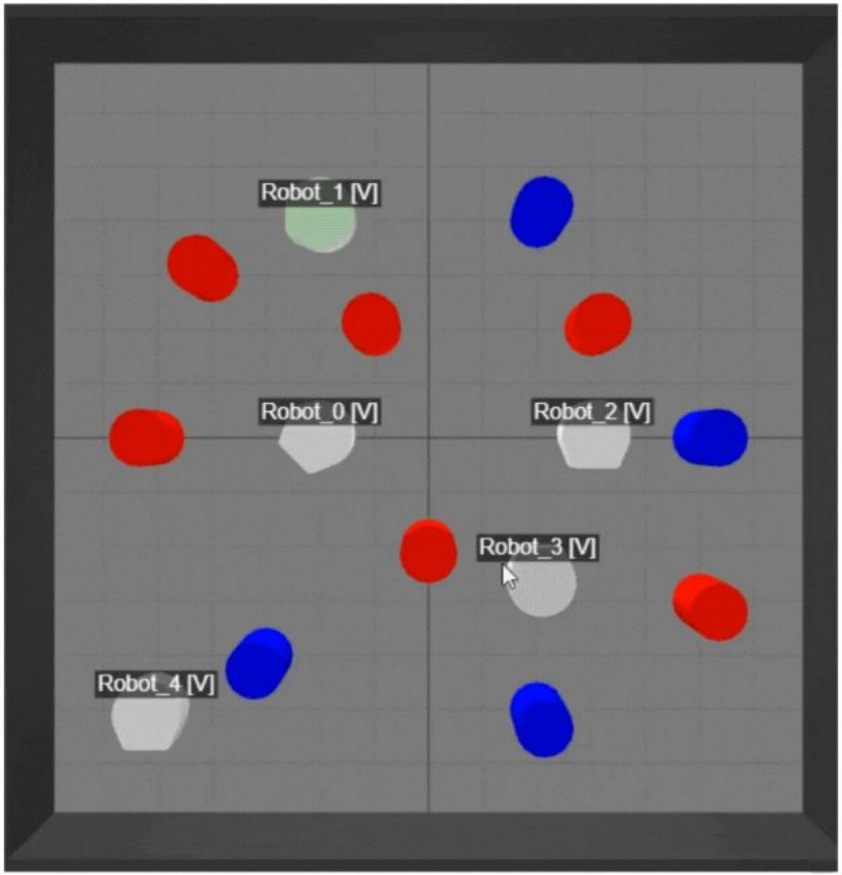
Progress & Effectiveness

Features of the Software Tool

- **Programming interface**
  - Behavioural
    IO
    General
    - Blocks related to specific behaviours
      - Choose between inbuilt behaviours or create new ones
- **Code generation & compilation**
  - C++
    (Pera Swarm
    Physical robots)
    - Java
      (Pera Swarm
      Virtual robot
      platform)
- **Other improvements**
  - Robot arena representation in the frontend
    - Specifying #robots, coordinates, directions deterministic/ auto
      - Automatic MQTT messaging
- **Demonstration**




Swarm Behavioural Programming Experiments
Physical robot
Virtual robot
Experiment with a single robot
Random movement with obstacle avoidance
Random movement with obstacle avoidance
Dynamic task allocation
Object finding
Object finding


Robot_1 [V]
Robot_0 [V]
Robot_2 [V]
Robot_3 [V]
Robot_4 [V]

Object Finding Behaviour

Object Detection and Inter-robot Communication

- Compare between detected color and object color
- The communication of object positions among robots through the LED signaling

Virtual Robot

Virtual Proximity Sensor

- Positioned at different angles
- Can detect colors and distances from defined angles

Move Towards Object

- If distance<distance threshold ⇒ change the color and stop the robot
- Else ⇒ turn the robot towards the object move it in that direction

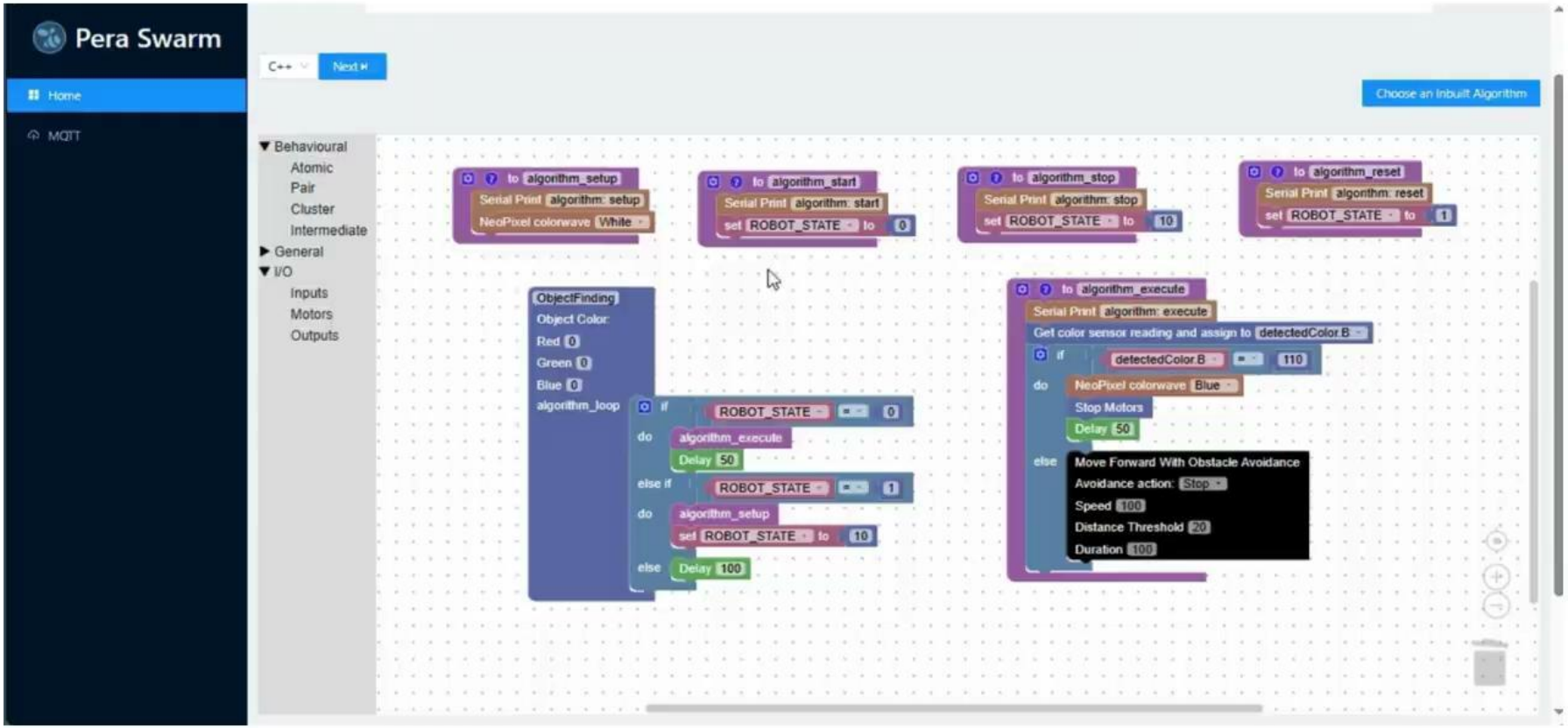
Physical Robot

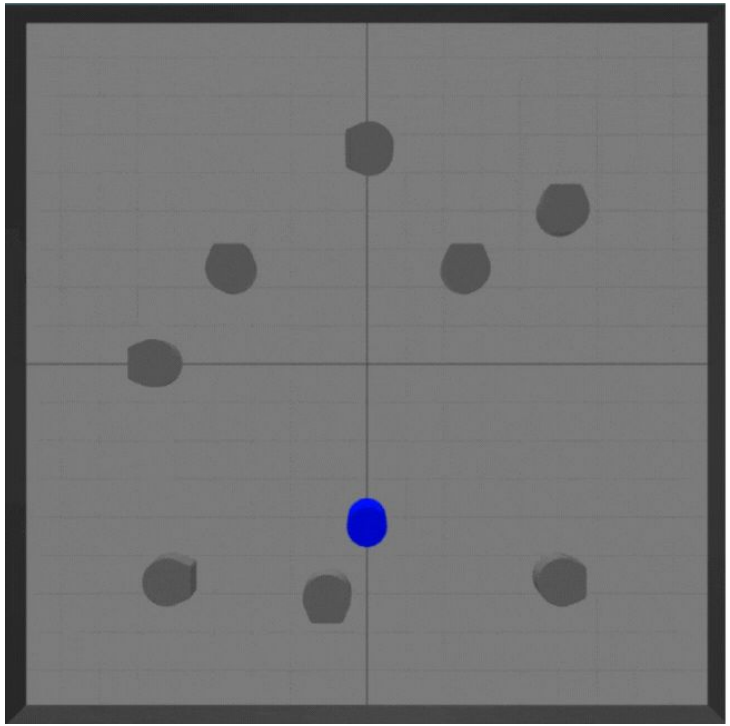
Color Sensor & Distance Sensor

- Positioned at the front
- Can only detect things in front of it

Move Forward with Obstacle avoidance

- If distance<distance threshold ⇒ change the color and stop the robot
- Else ⇒ Rotates to identify a direction without obstacles while checking for the expected color and Moves in that direction


Pera Swarm
Home
MQTT
C++
Next
Choose an Inbuilt Algorithm
Behavioural
Atomic
Pair
Cluster
Intermediate
General
I/O
Inputs
Motors
Outputs
to algorithm_setup
Serial Print algorithm: setup
NeoPixel colorwave White
to algorithm_start
Serial Print algorithm: start
set ROBOT_STATE to 0
to algorithm_stop
Serial Print algorithm: stop
set ROBOT_STATE to 10
to algorithm_reset
Serial Print algorithm: reset
set ROBOT_STATE to 1
ObjectFinding
Object Color:
Red 0
Green 0
Blue 0
algorithm_loop
if ROBOT_STATE = 0
do algorithm_execute
Delay 50
else if ROBOT_STATE = 1
do algorithm_setup
set ROBOT_STATE to 10
else Delay 100
to algorithm_execute
Serial Print algorithm: execute
Get color sensor reading and assign to detectedColor.B
if detectedColor.B = 110
do NeoPixel colorwave Blue
Stop Motors
Delay 50
else Move Forward With Obstacle Avoidance
Avoidance action: Stop
Speed 100
Distance Threshold 20
Duration 100



04

Findings

Dynamic Task Allocation Behaviour

Testing Environment 1

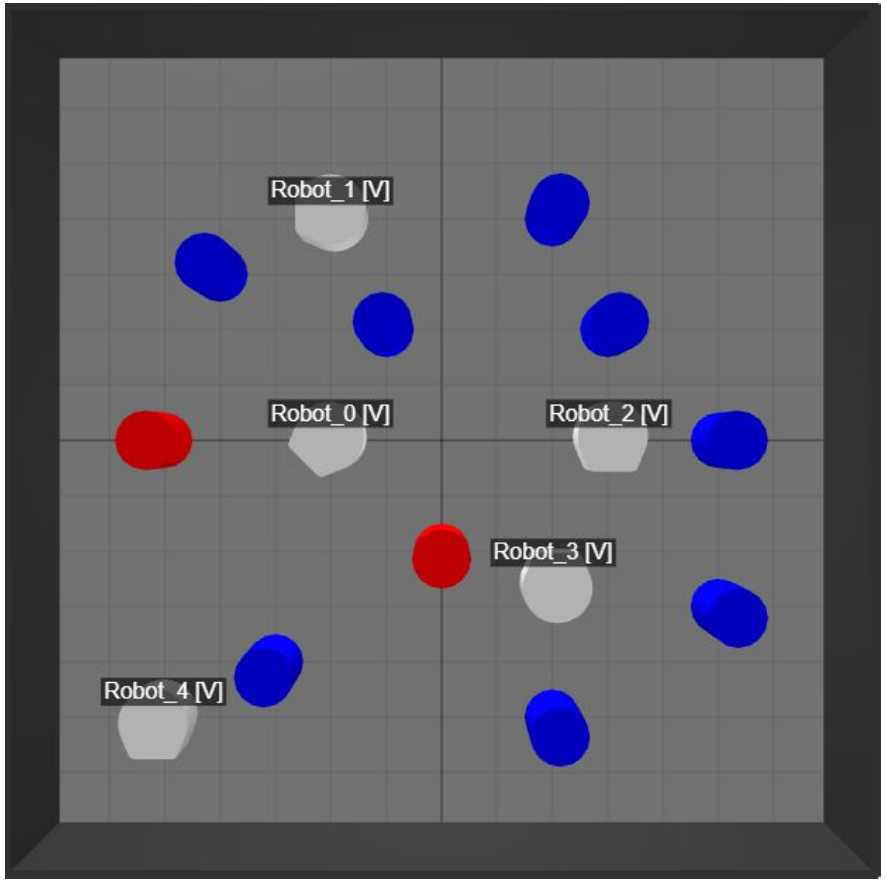
Number of Robots = 5 (Initially Red)

Number of Objects = 10
(Denoting Red & Blue tasks)

Task Distribution:

Scenario 1: 20% Red, 80% Blue
Scenario 2: 60% Red, 40% Blue

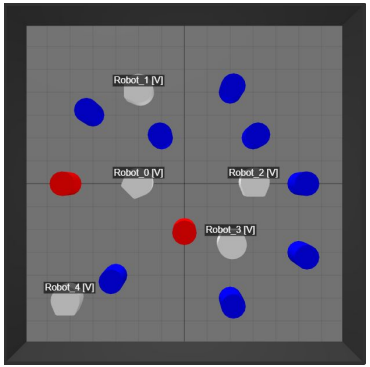
10 times per each scenario



Test 1: Time Vs Task Distribution

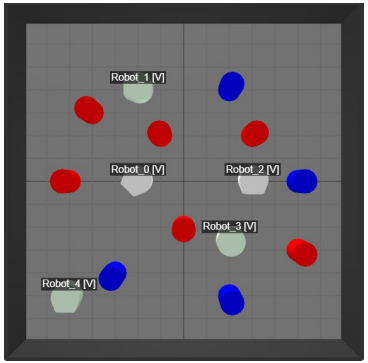
20% Red

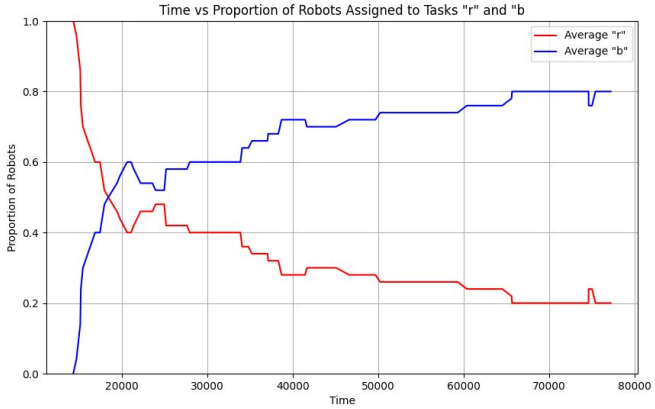
80% Blue

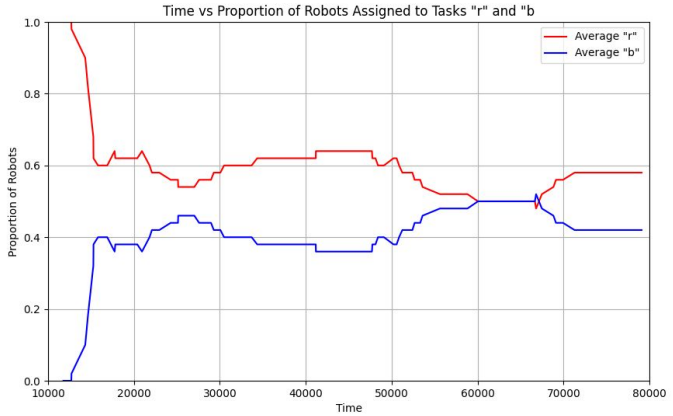


60% Red

40% Blue







Testing Environment 2

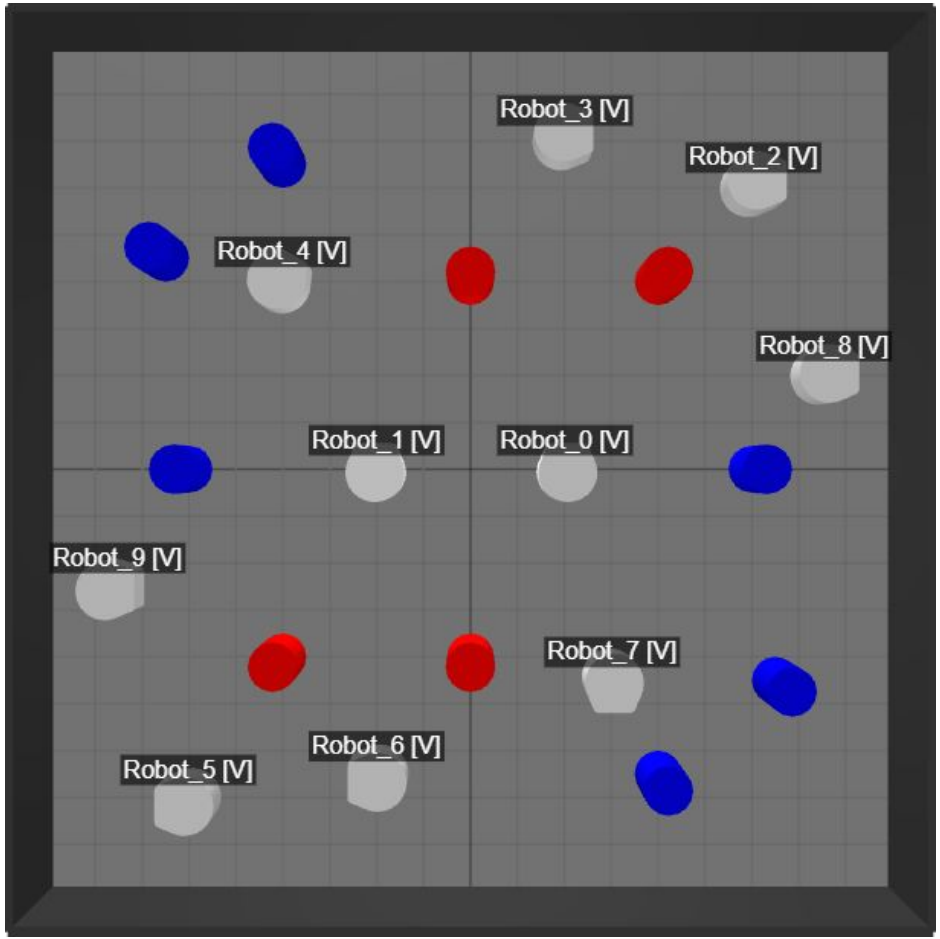
Number of Robots = 10 (Initially Red)

Number of Objects = 10
(Denoting Red & Blue tasks)

Task Distribution: 40% Red, 60% Blue

Robots and Objects are placed randomly on the arena

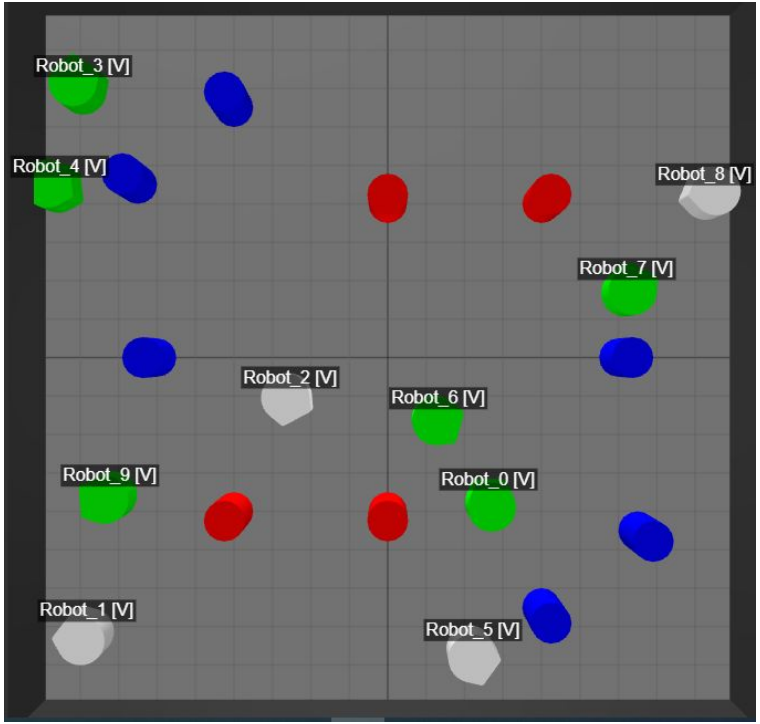
Red Assigned Robot - White
Blue Assigned Robot - Green

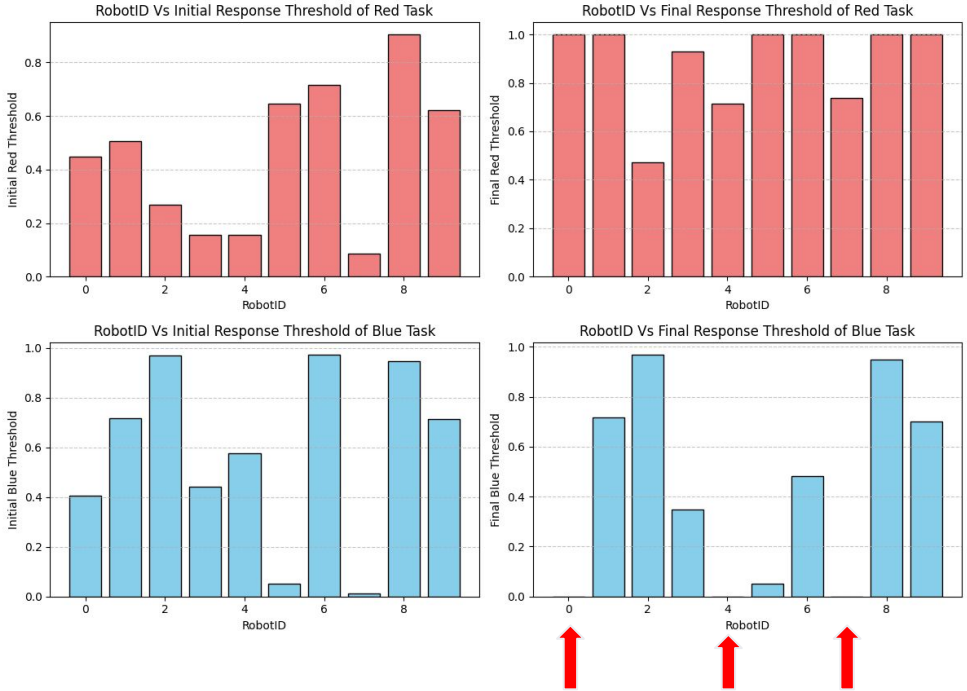


Test 2: Robot Id Vs Initial/Final Threshold Values for Task Red/Blue

Task Distribution: 40% Red, 60% Blue

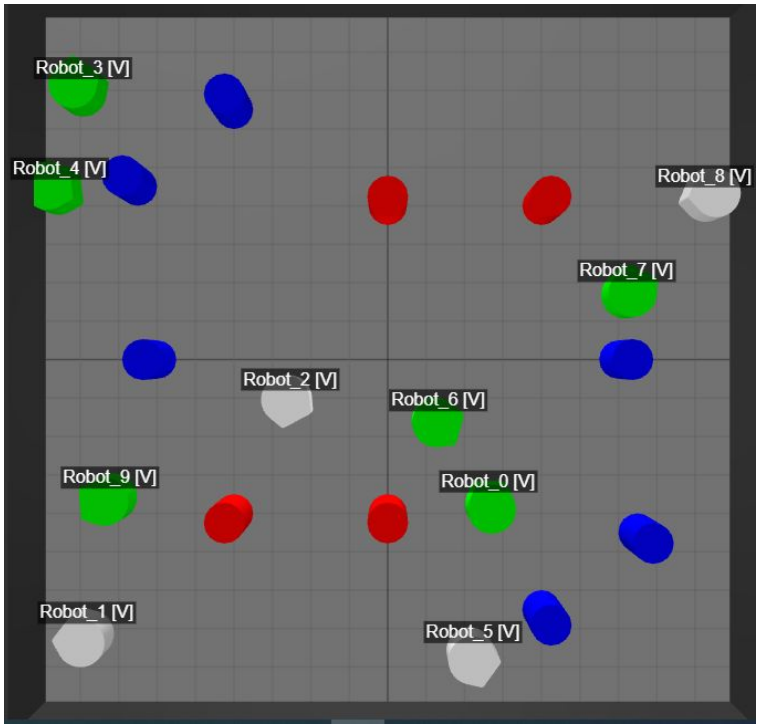
Observation: Some robots are strongly specialized (0, 4, 7) while some are weakly specialised

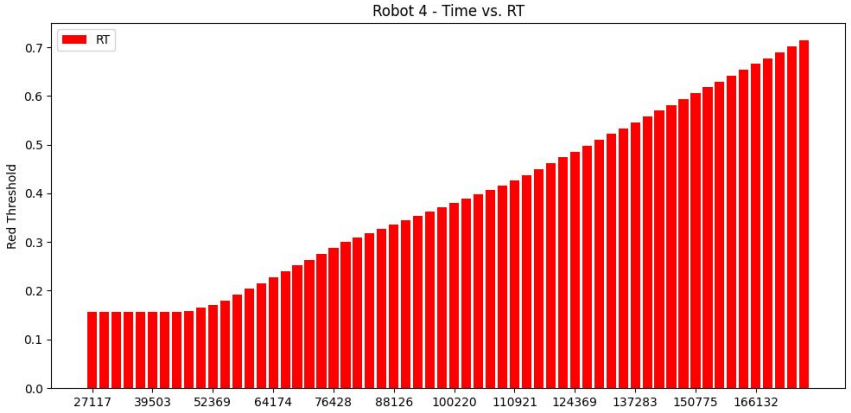


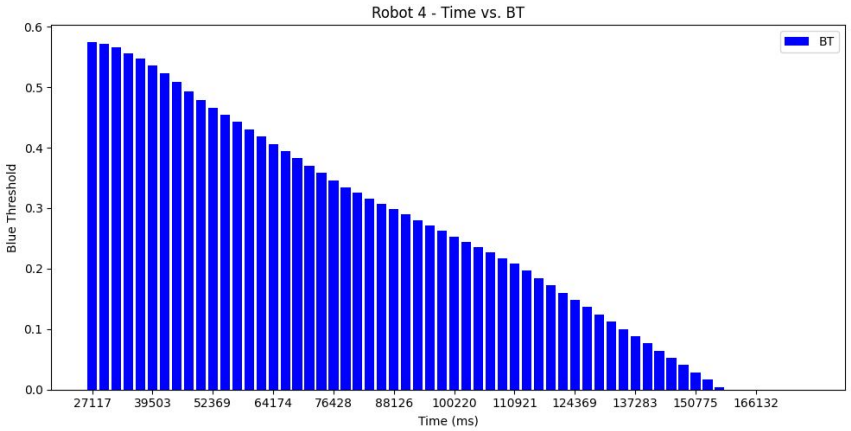


Test 3: Time Vs Threshold Values for Task Red/Blue

Observation: Robot 4 is strongly specialised into task Blue







Qualitative Performance Comparison

Criteria	Buzz Programming Language	MATLAB Simulink Tool
Usability of the graphical block based interface	Higher learning curve due to the Domain Specific Language barrier	A general purpose tool with a graphical code editor
Ability to programme new behaviours	Need C programming knowledge to change low level behaviours	Comes with pre built behavioural units. Doesn't support low level changes
Compatibility with virtual robots	Can be used with separate simulator platforms manually. Eg: ARGoS	Has an inbuilt 2D simulator
Compatibility with physical robots	Buzz VM resolves hardware dependency issues	Doesn't support swarm physical robot systems

Project Deliverables & their Impact

- Fully completed tool for programming both physical and virtual swarm robots
- Comprehensive set of swarm behaviours implemented as blocks
- Re-programmability for new behaviours
- Beginner friendly
- Beneficial to the educational/research sectors

THANK YOU!