A Review on Existing Swarm intelligence Programming Frameworks

S.M Ekanayake
Computer Engineering Department
Faculty of Engineering
University of Peradeniya
Peradeniya, Sri Lanka
e16094@eng.pdn.ac.lk

H.M.K Madhushanka
Computer Engineering Department
Faculty of Engineering
University of Peradeniya
Peradeniya, Sri Lanka
e16221@eng.pdn.ac.lk

A.L.H.E Perera
Computer Engineering Department
Faculty of Engineering
University of Peradeniya
Peradeniya, Sri Lanka
e16275@eng.pdn.ac.lk

Prof. Roshan Ragel
Computer Engineering Department
Faculty of Engineering
University of Peradeniya
Peradeniya, Sri Lanka
roshanr@eng.pdn.ac.lk

Dr. Isuru Nawinne
Computer Engineering Department
Faculty of Engineering
University of Peradeniya
Peradeniya, Sri Lanka
isurunawinne@eng.pdn.ac.lk

Dr. Sithumini Ekanayake
Computer Engineering Department
Faculty of Engineering
University of Peradeniya
Peradeniya, Sri Lanka
sithuminie@eng.pdn.ac.lk

Dr. Mahanama Wickramasinghe
Computer Engineering Department
Faculty of Engineering
University of Peradeniya
Peradeniya, Sri Lanka
mahanamaw.pdn.ac.lk

***Abstract*—This review address the main requirements that benefits the researchers. Those main requirements are having a compatible framework, tools that helps to implement the logic of the swarm operations, and implementing swarm behaviors. Further here discussed a code-less approach : a different approach called Swarm UI implementation. Also another less heterogeneity hardware platform support called EmsBot but withstand other swarm related issues such as real time operability, less resource and memory consumption. Finally a better approach is called EmSBotScript which supports heterogeneity among other fixes for challenges mentioned above. Also a crucial aspect is there when considering the swarm behavior modeling there are approaches taken in simple algorithms like if and elses, mathematical models and advanced algorithms based on graphs. Here the identified characteristics each of the pros and cons and the research gap is mentioned in the due review on each article.**

***Index Terms*—search-surrounding swarm algorithm, target-surrounding swarm algorithm, collective behavior, swarm robotics, swarm intelligence, Kilobot, multi-robot autonomous navigation and exploration, decentralized behaviors, heterogeneous robotic teams, over-the-air update,swarm systems, control framework, swarm programming**

I. Introduction

Swarm robotics is popular and gaining attention due to its practical applications. Here the idea is to perform complex collective behaviors using a large number of simple individual robots. But programming such a distributed robot system is quite complicated when increasing the number of the robots. On the other hand in swarm robotic systems, communication and coordination of the robots behavior is not done using a centralized or hierarchical control communication system. This makes programming swarm robots more difficult than other multi-robot systems.

This literature review focuses at strategies used to develop a framework for programming swarm robots in a novel approach based on different techniques. The main obstacle faced during this swarm project is to unify the heterogeneous swarm robots in to a single unit for ease of operations. For that the underline architectures of each robots must not depend on the software framework following research shows the approaches took for implementing swarm software framework for ease the problem in the heterogeneity of the swarm robotic system as well as address other related problems in the field of swarm robotics such as maintaining concurrency, less consumption of memory, resource utilization, manage scalability of the swarm robotic system. These are covered under the following sections.

II. Swarm Robotics

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 [1] 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 identified in the literature are described in this section.

This section features a selection of the most significant experimental Swarm Robotics efforts. The various experimental findings are organized into groups based on the tasks or behaviors that the swarms performed. Certain behaviors, such as aggregation and collective movement [2], are extremely simple and serve as a foundation for more difficult tasks. They are given in order of increasing complexity.

1) Aggregation
   - On simulated S-Bot robots, Trianni et al [3] experimented with an evolutionary algorithm. The microphones and proximity sensors are the sensory inputs. The motors and speakers are the actuators. Although their work is more on evolutionary algorithms than on aggregation, Bahçeci and ahin [4] also apply evolutionary algorithms and simulated S-bot robots to produce scalable outcomes.
2) Dispersion
   - Various researchers have researched dispersion using both real robots and simulation. Robots are repelled by objects and other robots in Howard et al [5] proposed field algorithm for the deployment of robots. The above mentioned approach is distributed and doesn't need centralized localization, therefore it may be scaled. However, the work is only conducted through simulation.
3) Pattern Formation
   - Martinson and Payton [6] describe a method that generates square lattices using local control laws acting on orthogonal axes and a common reference orientation for the robots.
   - A method for placing robots in various forms and patterns determined by implicit functions is displayed by Chaimowicz et al [7]. Robots position themselves within the required contour using a distributed strategy based on local knowledge. Both simulated tests and actual robot testing are conducted on algorithms.
4) Collective Movement
   - A distributed method for collective movement based on lattices is proposed by Lee and Nak [8]. The Lyapunov theorem is used to demonstrate that it converges. [9] proposes a decentralized algorithm based on lattice forms for the collective movement. In a specific scenario of research, the algorithm's stability is demonstrated. The plane is divided into Voronoi regions to implement obstacle avoidance.
5) Task Allocation
   - A distributed and scalable approach for labor division in robot swarms is presented by Jones and Matari [10] Each robot autonomously conducts a division of labor by keeping a history of the tasks carried out by other robots based on observation. It can then adjust its own behavior to fit this division.
6) Source Search
   - In [11], the odor localization problem is addressed, and robots search for the source of the odor using a distributed method. Both real robots and simulations are used in experiments.
7) Collective Transport of Objects
   - The collective transport of prey by ants, in which individuals wait for other mates if the object being transported is too large, serves as inspiration for Kube and Bonabeau [12]. In their studies, which were carried out with actual robots, six robots were able to work together to push an object in a purely distributed manner.
8) Collective Mapping
   - [13] describes a collection of algorithms for the exploration and mapping of vast indoor regions employing a significant number of robots. Up to 80 robots were used in their investigations, which take up a $600m^2$space.
   - In order to comprehend this area of multi-robot research better, a brief introduction to swarm robotics has been provided. The issue [14] has been introduced in the initial parts, which also highlight its key attributes and place the area in relation to more broad multi-robotic systems. The primary tasks, experimental findings, and platforms used in swarm robotics have then been summarized.

III. Architecture of Swarm Robotics System Software Infrastructure

Swarm robotics is a contemporary branch of robotics that focuses on fully autonomous robots that lack a centralized control system or shared body of knowledge. One of the most frequent issues is the process of developing such systems; at the present, there are no clear guidelines for designing and modelling such systems, and there are no industry standards for tools or software in the field of swarm robotics. From a software engineering perspective, swarm robotic systems are researched in this work. The goal of the study is to show how a software engineering strategy can reduce the issue of tight coupling between hardware platforms and swarm systems by providing instructions and tools for producing high-quality swarm robotic applications quickly.

The primary objective is to recommend appropriate methodologies and tools for developing, deploying, debugging, and continuously supporting SRS (Swarm Robotic System) software.

It should provide design and high abstraction layer modelling methods, per the Analysis and Design discipline. Implementation discipline states that: standard libraries of swarm algorithms, machine learning, and genetic algorithms should

be provided; a framework for designing and developing standard swarm robotics problems (foraging, coverage, flocking, etc.); and a set of abstractions for various communication approaches should be provided (e.g. wireless and stigmergy). It should offer logging and debugging tools, as well as profiling and performance testing, following the Test discipline. It should also offer visualization and simulation capabilities. It should include tools to develop, deploy, and describe packages, per the deployment discipline.

There are no SR standards in place at this time. The de facto standard for the creation of software for general-purpose robots is ROS, for instance, in the area of robot programming.

The fundamental concept is to move a temporary deliberator to the server component for sequencer model training. Robots that have been equipped with the trained model can operate independently and provide as much data as feasible for the model's continued improvement. Therefore, it is suggested that the controller and trained sequencer be mounted on the robot.

The deliberator should be a server software system that accepts information from the robot sequencers about the environment and enhances the robots' current actions. It is suggested that the development be divided into two phases: 2) The initial models are deployed and executed directly on robots, and the server continues to improve the existing models while allowing for the debugging of a system that has already been deployed. 1) The server simulates robots on the basis of a real stand, receiving real data about the environment and forming the initial behavior model for robots.

The notion of swarm robotics, its primary issues, and existing methods for SRS design and development have all been studied in this article. The SRS software infrastructure requirements and the software engineering methodology had been laid forth. It had been suggested to use a hybrid architecture and implement some SRS software infrastructure components.

IV. Swarm Behavior Modeling

A. Swarm Behavior : A behavior-based strategy for single and multi-robot autonomous exploration

[15] This research addresses a crucial challenge in the swarm area in Swarm robots. That is autonomous navigation. This article introduces a way of navigation which is based on behavior along with an efficient data structure to avoid previously visited areas by storing them. Based on the related work the approach was frontier exploration which has no need to structure the area of the robots are experimented. Also the provided solution avoided the complexities such as rubusting localization and mapping. Also address the issues with implementations at that time. Like there are exploration algorithms more machine learning based and there will be a bidding process those complexities are avoided and less complex implementations are defined in the article. Since this approach was a simplified approach in exploration. This solution may have issues with minimizing the uncertainties like navigating more than one robot for the same unknown location. Also this solution is using an external vision for localization. Therefore some different but more accurate localization techniques are missed. For example : Without external equipment the robot swarm can localize an area using round robin which is a decentralized peer to peer communication and without centralized server. But this approach will be efficient but make it more complex when identifying the workload for the implementation. Considering the methodology of their approach their behavior model introduces four behaviors along with a resultant emergent behavior those are,

1) Avoiding obstacles : avoid the obstacles and similar robots
2) Avoid Past : for gathering the newest location
3) Disperse : for executing a conditional disperse action
4) Explore : Wandering unknown locations according to FSA (Finite State Automata)

Those above implementations are avoiding complex implementations by using if else ladders only. But it would be more appropriate if all the behaviors can be linked using a FSA (Finite State Automata). This may increase the complexity of the problem that is gonna solve but it will be an expressive implementation. Finally the approach taken in the article is able to successfully decrease the computational complexity using hash tables by saving known locations. In the articles the robots are homogenous, therefore this work can more extend to having virtual robots as well as for having different architecture robots. This will create a challenge in maintaining two code bases for robots. Because hardware robots have a C++ related low level implementation and virtual robots have Java based code base. In the Pera Swarm project perspective, the exploration serves more efficiently and as well as less complex. But configuring the robots, all the robots in the mixed reality should be taken into consideration when creating codebase/es.

B. Swarm Behavior : UAV Swarms Behavior Modeling Using Tracking Bigraphical Reactive Systems [16]

This article suggests an approach based on bigraphs along with tracking to model different tasks. Determining all possible behaviors is the key function in this approach with some arbitaries. Further the main goal is to find an algorithm that determines all the possible scenarios should be there on the swarm. Other than the functional requirements there are non functional implementations such as making scalable, automated and solving provided tasks. Unlike other research articles this article models the properties and features of a swarm of systems which can be highly effective in creating a programmable model for implementing swarm behaviors. Another key feature is this implementation's coupling status. This model is able to introduce new algorithms to the system without affecting the current progress of the remaining parts of the model. This is a best practice when it comes to testing the behavioral model in some scenarios. Further in the modeling of the swarm system the implementations could be able to gain following features to the system. Firstly able to separate the swarm behavior from the low level implementation details of the hardware (in this case UAV drones). Secondly able to

isolate and abstract each of the drones, swarm of drones. The ability to represent the whole automation structure and others using a mathematical model (using a bigraphical pattern) helps to determine the capability to automate the swarm behaviors. Further the state behavior of this implementation adds another feature to the system which is to deploy different swarm behaviors. But since this approach is only tested for UAV drones this approach has the limitations that are only limited for UAV drones but with proper implementation this has the capability to upgrade this research compatible with other hardware multi robot systems. Further it is stated that this approach is limited for specialized tasks only. This implies having issues with the reusability for other tasks. Also it is stated that implementing a behavioral policy for this approach causes abnormalities and that each agent in the system can be beyond the control of the designer. Further those controlling or avoiding those anomalies are not present in the defined article. Finally the localization is not well defined in the article therefore there are some misconfusions when it comes to the localization of each drone happening peer to peer or in a centralized manner.

C. Decision making in heterogeneous robotics systems : ROS and Buzz: consensus-based behaviors for heterogeneous teams [17]

This research article mainly addresses the challenging parts of a heterogeneous multi robot system. Which are the decision making and controlling of a heterogeneous robotic system. Further this explains why decentralized peer to peer communications make the embedded intelligence increase which can result in a lack of memory to handle such operations. Further this explains well with the buzz language how the standard ROS systems will be able to operate successfully. But in the Pera Swarm approach having a complexity in the domain of the robot architecture that is the difference in the hardware architecture even though this addressed by this solution the next challenge is having virtual robots integrating into the system as a result ROS system should be handled virtual robots also they should be supported with the buzz language. The combination of ROS systems with Buzz language is able to create consensus behaviors in robot swarms. And ROSBuzz will be a framework for programming such swarm intelligence events. This implementation was successful for heterogeneous multi robot systems because this framework is driven in a virtual machine based environment. This enables the feature that supports heterogeneous support.

D. Formations : Chain Formation in a Swarm of Robots

When navigating in an unknown environment it is crucial for having a form of robots that will reduce the missing undiscovered locations and localization of each robot can be identified using a via the formation. This research emphasizes the approach of forming chains of robots to explore the environment more efficiently. This research shows the systematic approaches to form chains as well as the speed of chain formation. Also this approach is capable and eases the explorations and localization of robots as those robots have such formations. Then considering the formation the form of robots are analyzed and following formations are considered. One is forming all the robots as a chain in one direction. Another is the branching of those robots, along the other branches. However Using these formations it is identified that chain formation along one direction does not completely cover the area. But it also has the capability to cover long distances. Therefore a proper formation of robots must be considered for swarm operations otherwise it will be very difficult to automate each robot separately. This approach will cause many problems like a lot of collisions, and re exploring an discovered location. Robots are limited to one of the small regions of the environment. Further this experiment is based around a nest. All the robots are creating a chain formation around the nest and exploring from the nest. But this experiment is not designed to perform in an environment with obstacles. Therefore the impact of the obstacles were not considered. Further all the robots are forming a chain. The most suitable approach is decentralized communication but because of the close pattern followed by decentralized communication and localization results in the robots being limited into a single unit of area. Therefore the total area covered by the robots at a particular time will be lower.

E. Target Searching Algorithms in physical and virtual environment : Comparison of the behavior of swarm robots with their computer simulations applying target-searching algorithms

This is an experiment on a virtual and real environment for searching and surrounding a target. This experiment is done emphasizing the both real and virtual simulations shows how these searching algorithms work in real scenarios. Further for experiments the swarm robots used in this work are the Kilobots. Also this experiment did not use heterogeneous multi robot systems because of the more scope regarding the searching algorithms and surrounding the object tasks. Further according to the search algorithm the number of search patterns will be increased when the target is away from the robots. This will delay the robot system. Also the number of computations performed by the system will be increased. Also the message passing is done via a centralized server this is a useful and quick approach when it comes to developing. Also simple is the development of such algorithms. And the memory allocation for the project will be much lower.

F. Behavioural modeling based programming framework [18]

They provide a higher-level framework for swarm robot programming in this research. The definition and implementation of the behaviours in this framework are done using a bottom-up, behaviour-based design approach.

Communication

A virtual pheromone-based communication technique is used for the communication. It is made to work with IR

communication systems that have the ability to gauge the strength and direction of incoming signals. According to Fig. 1, there are three main measures related to the communication system.

Heading: Angle from the north direction to the robot head measured counterclockwise. Bearing: The angle between a robot's heading and the distance line, measured counterclockwise. Distance: The distance between the present robot's center and the place where the incoming signal came from.

Programming Model

Behaviour programming and pheromone processing are the two main divisions of the programming model. Programming behaviour is possible with or without the usage of states, depending on the user's preferences. The complete programming model is depicted in Fig. 2 by the interactions between the core aspects, pheromones, behaviours, and states. Only after receiving a stimulus pheromone message does a robot modify its state. In some cases, a robot may change specific behaviours but not changing states in response to a pheromone communication. The robot remains in that condition when receiving pheromone messages and engages in the behaviours that are permitted in that state. New pheromone types, states, and behaviours can be defined by the user.

Built-in Behaviours

Based on the degree of robot engagement, these behaviours are divided into four tiers. These are the categories:

Preliminary Behaviours: These preliminary behaviours don't require any communication with other robots. They are utilised by a single robot for a variety of movements. The first tier of the overall design hierarchy is represented by this. Couple behaviours: These are actions taken by only two robots, the active robot and the neighbour robot. Neighbour Cluster behaviours: An asset with more than two robots are referred to as a cluster. The third level of the hierarchy is this. Global behaviours: The bottom level behaviours already included in the library can be used to generate these behaviours. The behaviour hierarchy's top layer is represented by this. From this research article, we can take the above behaviour architecture directly to our project development since they used bottom-up architecture in their design.

G. Colias: An Autonomous Micro Robot for Swarm Robotic Applications.

[19] In Robotic Swarm, we focus on controlling a large number of individual robots to solve complex problems. Swarm robotics and related research are done mainly by using simulation software because of the hardware complexity of the robots and the cost of the robot platforms. In this article, they provide low-cost, open-platform autonomous micro-robots for robotic swarm applications. By defining well-defined interaction rules for each individual robot, decentralized control of the swarm can be achieved. These rules will be executed continuously in an infinite loop in the individual robot. The interaction between simple robots that lead to the collective behavior of a swarm has an indirect impact on the behavior of each individual robot. Thus, the homogeneity of the robot platform is an important issue when executing robotic swarm scenarios.

It is quite difficult to simulate such a vast number of robots, and the results we got differ from what we would see in actual robot trials. Therefore, a robot platform must have some criteria such as low-cost design, long-term autonomy, long-range communication, bearing, distance and obstacle detection, neighboring robot detection, fast motion, and open source design. In this article, they developed the Colias platform to meet the above-mentioned requirements.

Using various basic high-level functions, Colias provides simple programming and user-friendly implementation. Swarm behaviors use sensors and communications in order to make decisions. Decision making is done in two forms such as calling a function and hardware modules. Colias uses GNU compiler collection (GCC) for the compiling process. For the swarm behavior colias implemented state-of-the-art swarm algorithm (BEECLUST) with different population sizes. Below diagram shows the algorithm which robots are followed.

When a robot detects an obstacle it rotates itself and executes the obstacle avoidance routine. When a robot detects another robot it stops and measures the illuminance of the ambient light and uses that measured illuminance to calculate waiting duration for the robot. When the waiting time is over the robot takes a random degree turn and moves forward.

Using this open-hardware platform they did experiments using hardware components which is much better than using simulation softwares. This has been shown that the build robot is capable of serving as an autonomous platform.

When we create the framework for modeling and programming swarm robots, we can get the idea of modeling swarm behaviors using high-level basic functions and combining them to achieve more complex swarm behaviors. Also we can follow the work they have done when developing colias and develop high-level swarm behaviors like clustering, using proposed collision detection and avoidance methods in this research paper.

H. Cooperative Pollution Source Localization and Cleanup with a Bio-inspired Swarm Robot Aggregation

Exploring dangerous and severe environments with robots has the potential to greatly increase human safety. As an example robot systems can be used to locate the source of a chemical spill and clean the polluted region. In this paper it provides a method to chemical leak detection and cleanup scenario using robot swarm. When we consider an extreme environment, most robots fail to give reliable results for those environments with chemical and radiation contamination. Since most of the multi-robot systems use wireless

communication methods to transfer data between themselves, radiation sources can hamper wireless connections which will make the robots ineffective. Using short range communication techniques we can overcome this problem. In robot swarm systems it uses inter-robot connection protocol which we can use in this case.

Since a group of cooperating robots operates and has an ability to complete a task at higher speed and it gives more reliable decisions when compared to a single robot doing the same task, researchers tend to use multi-robot systems and divide the task between large numbers of simple robots and achieve the task cooperatively.

In order to create a protocol by which several robots may work together and finish a given task, robot interaction rules and techniques must be defined. This will lead to high level data transfer among robots and centralized decision making centers. Many swarm behaviors inspired by social animals like bees, ants etc. Such behavior we can identify is aggregation. This can be defined as gathering individuals around an area. Aggregation can be identified in two types. They are cue-based aggregation and self aggregation. In extreme environments cue based aggregation is more suitable since physical queues are crucial in those situations. In this paper they use the BEECLUST aggregation method as a basis to implement the exploration system. Detecting and cleaning the source of chemical leakage is the main goal of the robot in this scenario. So robots have to find and reach the source and start cleaning the area with the presence of other robots. State diagram for the task can be defined as below.

When robot-robot collision happens, the robot waits a random time based on the cue intensity that the senses. After a timeout the robot turns at a random angle degree between 90 to 180 degrees and continues the initial task. In this paper exploration scenario is implemented using mona robots and simulation models are created using Webots software.

In this proposed method swarm robots are used to chemical leakage localization and cleanup. It uses BEECLUST algorithm and pheromone following behavior. Even Though we could use the advantage of short range communications, when leakage vanishes robot cooperation decreases and this might be a disadvantage.

We can use the idea proposed in this article to model the swarm behavior like find-and-rescue.

V. Recent implementations on Swarm Programming Frameworks

A. An Actor-based Programming Framework for Swarm Robotic Systems [20]

To describe the high-level virtualization for robot platforms, where they develop the concept of the actor. Each created Actor maintains a data structure, is bound to various plug-in groups and serves as the fundamental building block for the control of a group of behaviors. They suggest a system for managing all Actors together. A swarm robotic system's robot platforms are efficiently structured. Actor-level synchronization is one of the cooperative task primitives that evolved automatically. Task creators can focus on intricate swarm robotic task coordination tactics, instead of the tedious intricacies of operating individual robots by using a domain-specific language (DSL) to create Actor-based tasks. The suggested framework is developed in C++, and both simulations and field testing have been used to validate it both quantitatively and qualitatively.

The suggested framework makes it simple for users to manipulate one to many autonomous vehicles and offers a reliable self-organization method for swarm robotic systems.

Actor Control Block (ACB) is the name of the data structure that the Actor scheduler keeps for each Actor in this software architecture. Basic Actor operations include start, stop, pause, activate, and switch, and basic Actor attributes include name, identification (ID), status, priority, software resources, hardware resources, permission, task, relationship with other Actors in the swarm, and statistical information at runtime. These are all included in the ACB.

The "Actor" configuration is mostly used to configure the actor information, such as the actor's name, priority, and necessary plug-ins. Task creators can add the configuration of required Actors in robot swarms and select the appropriate plug-ins for them. Distinct Actors will be given different abilities by loading various plugins.

The introduced "Actor" based control unit helps to decouple the high-level task programming with specific robot platforms. Robots are able to autonomously behave, handle unusual situations, and network disturbances with the provided framework.

From this work they have done, we can take their 'Actor' based approach to our project as well. By using that, we can use all the atomic swarm behaviors as the building blocks of complex swarm behaviors. Since we are going to develop a visual programming language, the language they have used C++ is not appropriate for our work.

TABLE I
State of the art framework

Framework	**Orchestration**	**Type**	**Testing environment**
Buzz	Decentralized	Heterogeneous	Simulation
ROS	Centralized	Heterogeneous	Real world
Karma	Centralized	MAVs	Simulation
Voltron	Decentralized, Centralized	UAVs	Simulation, Real world
COMETS	Decentralized	UAVs	Real world
Meld	Decentralized	Heterogeneous	Simulation
PaROS	Centralized	UAVs	Simulation, Real world

B. Buzz: An Extensible Programming Language for Self-Organizing Heterogeneous Robot Swarms [21]

In this study, they state that a DSL is an essential tool for the deployment of actual robot swarms.

A DSL for robot swarms could serve as a platform that (i) filters the low-level information about networking and space in an effective, resource-conscious manner; (ii) provides a coherent system abstraction; and (iii) serves as a standard

platform for benchmarking and code reuse. They introduce Buzz, a unique DSL for robot swarms.

There are some essential requirements that a successful swarm robotics programming language must satisfy. As was previously said, the language must first empower the programmer to operate at an appropriate level of abstraction. Second, the language must support compositional thinking by offering predictable primitives that may be logically coupled to create more intricate algorithms and constructions. Third, the language must demonstrate that it is sufficiently general to (i)express the most widely used swarm coordination algorithms, such as task distribution, flocking, and collective decision making; and (ii)support heterogeneous swarms. Fourth, the language's run-time platform must guarantee appropriate levels of resilience and scalability.

This paper's primary contribution is the design and implementation of Buzz, a programming language that satisfies the aforementioned objectives.

Buzz allows the robots to be divided into numerous teams. Each squad in Buzz is referred to as a swarm. A unique identifier that is recognized and acknowledged by all the robots in the newly established swarm is required by the programmer in order to build a swarm. The result is a swarm-type class-like organization.

The method $exec()$ can be used to assign tasks to a swarm after it has been generated.

```
// Join the swarm if the robot
  identifier (id) is even
// 'id' is an internal symbol that
  refers to the numeric id of the robot
  executing the script
s.select(id%2 == 0)
// Join the swarm unconditionally
s.join()
// Leave the swarm if the robot id is
  greater than 5
s.unselect(id > 5)
// Leave the swarm unconditionally
s.leave()
// Check whether a robot belongs to s
if(s.in())...
// Assigning a task to a swarm
s.exec(function()...)
```

Buzz scripts are compiled using two tools: buzzc, an assembler/linker, and buzzasm, a compiler.

There are more complex functionalities mentioned in their article like neighbour operations, Virtual stigmergy, Collective decision making, Separating into multiple swarms, etc.

They introduced Buzz, a cutting-edge programming language created for massive, diverse robot swarms. One of the contributions of their work is a mixed paradigm for robot swarm implementation, which enables the developer to specify fine-grained, bottom-up logic as well as reason in a top-down, swarm-oriented manner; another is the definition of a compositional and predictable approach to swarm behaviour development, and a third is the implementation of a general language capable of expressing the most typical swarm behaviours.

As the inspiration for our research, we can use Buzz programming language as a high-level language, but we need to extend our project as a visual programming language.

C. GSDF: A Generic Development Framework for Swarm Robotics [22]

"Generic development framework for swarm robotics" also called as GSDF is,

1) Component based development scheme
2) Decentralized runtime and swarm programming interfaces
3) Completely compatible with ROS (ability to use ROS ecosystem)
4) Flexible and extensible framework

In GSDF it has decentralized lightweight runtime which maintains all swarm-related data and process messages. Other than that it provides multiple programming interfaces and a swarm library which will be easier to define swarm behaviors using component based application development approach.

GSDP framework consists of several layers such as,

1) Application Layer + Library
2) Interfaces related to swarm behaviors
3) Runtime
4) Abstract communication Layer

In this framework runtime is responsible for maintaining the entire swarm, swarm related data, individual swarm robots and neighbor robots. This runtime is lightweight since individual robot's resources are limited. When discussing programming interfaces and libraries in GSDP it is delighted by buzz programming language. GSDF provides an efficient near-range information sharing method called BlackBoard. It also contains floking, particle swarm optimization like commonly adopted algorithms for typical swarm behaviors. Extension of the library is also possible.

Since this GSDP framework is fully compatible with ROS, it also uses component based development which is recommended by ROS 2.0.

This framework provides an abstract and unified communication layer, which encapsulates lower-level details of communication mechanisms. In the GSDF communication model it has basic three functions like init(), broadcast(), receive(). Its supports both ROS message protocols like TCPROS, UDPROS and OpenSlice DDS.

In GSDF runtime each robot has an independent process called "daemon node". To identify robots uniquely it has an integer Id. Daemon node is responsible for publishing different kinds of messages in different frequencies and process received messages accordingly. Output messages are put into a specific output queue, which are scheduled using the "Weighted Round Robin" strategy while input messages are

entered in the input queue in "First In First Out" manner and scheduled sequentially. Also daemon nodes responsible for maintaining all swarm related data such as individuals and neighbors. For the programming framework it is enlightened by the buzz framework. GSDF has similar programming interfaces like buzz, such as Neighbors, Swarm, Virtual Stigmergy. They are implemented using C++ language and it uses object-oriented programming for that.

D. Introducing requirements must be on a robotic software framework by abstracting low level functions.

This is research based on swarms of UAVs. This research is able to address the challenges in programming swarms. First challenge is to make drones work in parallel and be able to implement the program to be scale independent despite the number of available drones. Further while execution on events drones failures may happen. Therefore they must be handled automatically. Also this describes the current state of orchestrating swarms by programming each drone. This research paper focuses more on the PaROS (Programing Swarm) framework based on the programming primitive called abstract swarm (a set of instructions that can be utilized for simplifying programming drones of swarms).

Proposed frameworks for programming individuals or swarms of drones are following.

The goal of this introduction of such a framework is to avoid low level programming. It was avoided using programming primitives by abstracting the implementation details of the swarm robots. Abstract Swarm defines a set of operations to simplify the programmability. A detailed representation of the operations can be shown as below.

Following are the brief descriptions of the derived operations. Path Planning for determining the flight plan for a single drone. pathPlanning() is invoked to determine flight plans depending on the existing swarm drones. In the PaROS framework there are two algorithms for PIA (Path Planning Algorithm) and the nearest neighbor algorithm. Also the framework is capable of overriding the developed pathPlanning() for customizations. Under Points of Interest and Area Declaration, programmers should be able to assign swarms in the defined points and set priorities to the locations. So each swarm is able to cover each of the areas according to the priority level. Drone Enumeration is there for identifying each individual drone and enabling communication between each other drone. Another operation is task partitioning, there are many activities needed to perform all the swarms. For that each of the individual drones should be able to receive the tasks which should be properly divided within available drones. Therefore there should be a defined operation for separating tasks within the swarm. Also there should be operations related to events. Event handling operations, when a specific event occurs operations should be implemented describing how the drone should react. Finally operations like Failure Handler must be present because if one drone fails to perform a task, Another should be able to perform the same task for that failure detection and correction should be there.

Experiments done with the drones tested the effectiveness of the nearest neighbor algorithm and execution times were lower. Further using PaROS execute a complete coverage path planning algorithm. Finally shown some of the similar research that is able to smooth the turns as well as reduce the number of turns during an experiment. Programming with PaROS shows the concept of abstract swarms. The programming complexity is determined by the SLOCCount tool. For that the experiment was aerial imagery (an experiment, drones go to an assigned location and take an image and return to the deployed area.). Here the communication between the drone and the server is hidden. As a result, it is able to lower the programming complexity. Therefore, according to this research PaROS framework is able to simplify the programmability of swarms of drones as well as individual drones. Furthermore this framework is able to eliminate the low level programming by introducing swarm abstraction like programming primitives.

E. CrazySwarm : a python based approach for programming swarm robots. [22]

In around 2015, very few (research) labs were able to operate more than 10 robots due to high single-robot cost, high engineering effort and limited reliability. Since this swarm robotics topic has been more popular lately, many research projects are happening around the world now. Crazyswarm is one of them. It is a platform that allows you to fly quadcopters in tight, synchronized formations.

When talking about its system architecture, it has tight integration with motion capture systems (Vicon, OptiTrack, Qualisys) and a custom, robust object tracker (a single marker per CF is sufficient). In order to make it fast enough to serve so many, they have optimized quite a bit for latency. They introduced One-way data flow (broadcast communication) and also implemented a native (C++) backend optimized for low latency. It relies on on-board autonomy (state estimation, planning, control) and works with the official firmware. From the user's point of view, the user doesn't need to know about these technical details. They can script into Crazyswarm using python. It has an API which is optimized for swarms. Also, they have introduced some tools for swarms (simulation, visualization, battery check, reboot etc.)

Crazyswarm was initially built for motion capture systems, but now it supports a wide range of localization systems. Practically when they implement the system, they use a centralized approach. They implemented one big C++ application which allows very low latency and few data copies in between. However, Crazyswarm also uses an anti-ROS pattern under the hood. And also they optimized low-level communication using broadcasts and compressed radio packets.

Typically commercial motion capture systems are designed for tracking humans (skeleton, face) or a few, large rigid bodies. Crazyswarm developed a library called 'libobjecttracking' for custom frame-by-frame tracking. Crazyswarm supports two different modes for rigid body tracking.

1) Tracking the Pose (Position + Orientation) of their robot: uses Iterative Closest Point (ICP) algorithm.
2) Pure position tracking: uses Optimal Assignment algorithm.

In both cases, the Dynamics filter (max a, v, ω) was used and also initial position guess was required.

When comparing the first approach, the Tracking the Pose method needs more markers on the robot and is less robust. But as an advantage, the full pose can be tracked using this method. In the second approach, Pure position tracking requires XY movement to recover yaw, but only needs one marker on top of the robot and is also more robust.

When talking about Crazyswarm tools, it has a python script called 'Crazyfile Chooser' which allows users to get a visualization of the initial positions of robots, enable/disable crazyflies which users want to operate on, battery check and reboot.

In terms of simulation, it uses the same script as used for physical robots and users only need to add —-sim flag. It has no ROS dependency and has Linux and Mac support.

F. Different approaches of a Swarm UI interface codeless approach: a direct manipulation in the environment causes behavioral changes in a swarm.

This research is based on creating a swarm user interface using direct physical manipulation. The difference is other frameworks using robotic programming frameworks. But in this research rather than using those frameworks the aim is to use the direct manipulation of objects. This approach is based on current UI programming practices. That is creating elements, abstract attributes, specify behaviors and propagate changes. The main challenge is the limitation of this area for highly skilled programmers who have good knowledge in robotics programming. The reason is robot programming focuses more on the low level programming in sensors and actuators.

Direct manipulation refers here to moving any object in the environment that results in a trigger to act as a swarm of robots to do a certain task. Therefore there is no involvement in the complex programming. Only there is the swarm UI programming using the above steps described. Firstly, creating elements by drawing and construction. Second is abstract the attributes through demonstrations. Thirdly, specify the expected behavior. And finally propagate changes in the behavior and how to react to those changes.

This research proves the usability of the concept of using less programmability. This way is able to avoid programming because this experiment defined model will react as the environment is changed. Also since because of the high level implementation the study participants most of them found it difficult to predict program behaviors. Also this states that increasing the number of robots causes errors that may result in errors that cannot be considered negligible. This causes the unique design of the environment for this research purpose.

G. A modular framework that supports limited architectures of hardware called EmSBot

Robotics systems complexity is higher because of the advances in sensors, actuators and computer technologies. From low level drivers to high level functions named planning and navigation have to be dealt with even though robotic systems are equipped with heterogeneous embedded processors. The challenge required to overcome is the coordination and communication when it comes to a swarm of robots. Because of this difficulty in developing advanced complex robotics applications from scratch a number of robotic software frameworks are developed. The paradigm shared between all the frameworks are the same. All of them are component-based. This originated from CBSE (Component Based Software Engineering). Using these principles enables robotic applications to become loosely coupled. Using Robotic Software Framework introduces better software quality, code re-usability and collaborative development. Using Robotic Software Frameworks have the following requirements and challenges.

The major requirement is supporting multi hardware platforms and operating systems. Further the consideration in the resource constraints of those embedded systems should be considered. However the framework must be able to abstract the above requirement. Next challenge is to develop the independence between communication networks and transport protocols. The framework must be flexible enough to switch between different network vendors. Another challenge is real time support because most of the swarms operate in real time therefore the communication must support real time operations. The framework must withstand such coordination the real time communication with the network layer. Also fault tolerance support is also required the targeted framework must have the simplification of the reconfiguration support. Finally the simplicity and ease of use is another requirement when it comes to software frameworks.

EmSbot Framework is introduced in this to support the problem of communication transparency. The following architecture describes how the framework operates in one micro controller-based robot.

Embedded modular component based software frameworks such as EmSbot suitable for making resource contained swarms of drone controls. Further OS abstraction extends to work with other operating systems. Using a port based mechanism is the method of exchanging messages between agents. Because of this only approach the agents are able to handle separately. Uniform message passing capabilities provides this software framework the capability to operate real time. Further this framework supports self fault handling as well as other agent binded fault handlings which supports the ease of fault handling. EmSbot is fully implemented using C and very suitable for micro controller-based robotic systems.

H. EmSBotScript : A smaller virtual machine based software framework for addressing most challenging aspects in programming swarms robots.

To deal with the diverse and miniature hardware components present in the swarm of robots, for handling such heterogeneity developers adopt a VM (virtual machine) based approach. Major challenge is that VM based systems are not capable enough to handle resource constraints. The introduction of EmSBotScript, a small VM based software framework is to address the above problem. EmSBotScript is able to support CPU independence, low memory footprint, concurrency and synchronization. This was shown using programming models, a script language and proposing VM architecture.

EmsBotScript is able to support cross platform support, this solves the heterogeneity problem. Also simplifying the codelet functions is able to support concurrency. For reducing memory usage implementing event - based synchronizations are helpful and use hardware virtual paging mechanisms. Furthermore stack paging is proposed.

VI. Visual Programming Approaches

A. A Survey on Visual Programming Languages in Internet of Things [23]

Visual programming plays a key role when considering user interaction in today's software industry. Visual Programming Language (VPL) lets user creates a program by manipulating elements graphically. Many VPLs are uses graphical shapes or blocks to represent entities while enabling the addition of relationships between them. VPLs are used in many fields such as education, multimedia, video games, simulation automation, data warehousing etc. Examples of visual programs can be identified as Scratch, Pure Data, Unreal Engine, VisSim, CiMPLE, etc. Even though VPLs are shredded in various domains, the field of IoT is still far behind compared to other sectors. IoT mainly focused on the interconnection between several heterogeneous objects or "things". It refers to a variety of heterogeneous hardware platforms with micro controllers for the creation of different applications. These platforms more often have a dedicated Integrated Development Environment (IDE) and separate programming language which raises difficulty when writing a program. VPLs are developed to ease programming hardware platforms with drag and drop components or blocks. This doesn't require much knowledge of the programming language.

There are a few VPL open source platforms for IoT development such as Node-Red, NETLab Toolkit, Ardublock, Scratch for Android, Modkit, MiniBloq, and NooDL. Some of the VPLs mentioned above are desktop environments and some of them are web-based VPL environments. Also, there are a few proprietary VPL platforms such as DGLux5, AT and T Flow Designer, Reactive Blocks, and GraspIO.

There are a few challenges associated with the integration of VPLs with the Internet of Things.

1) Extensibility
   We know VPLs are scoped to a very limited set of operations. Also, some edge cases are extremely difficult or impossible to implement in VPL.
2) Slow code generation
   In IoT, it can engage with a large number of devices. It is necessary to detect and solve the problems. Due to the fact that VPL results in slow code generation, it is really hard to optimize by a developer.
3) Integration
   To leverage the ease of programming Integrated Development Environments and Visual Programming Languages should be designed together.
4) Standard Model
   Different service professionals address modeling a problem in a different manner. Hence there is a strong need for a standard model structure for the VPLs.
5) Abstraction
   VPLs are made to better show and interact with the current abstractions. It allows developers to control something logical and challenging.
6) User Interface
   VPLs in IoT brad to categories for different situations such as for non-programmers, program learning environments, and advanced data flow aggregators.

In conclusion, VPLs help to visualize the program logic. Also helps users to develop the programs without worrying about textual programming. On the other hand, it has less burden over handling syntax errors. But this will take a larger amount of time spent just to write a basic and small application or a program.

For the swarm intelligence framework, we can get the idea of VPL and simplify the overall swarm behaviors through a visual interface rather than going for low-level complexities.

B. Smart Block: A Visual Programming Environment for SmartThings

This article proposes a visual block language called "Smart Block for IoT" which mainly focuses on SmartThings. [24] When designing the visual block language they based it on IoTa calculus. It is a core calculus for IoT automation that generalizes Event-Condition-Action (ECA) rules. Visual programming language makes programming easier by manipulating programming elements graphically, so users don't have to specify the program textually. Even though the fact that users are not good at programming, visual block language enables the ability to write a program and build SmartApps by dragging and dropping blocks in the environment. When implementing the visual environment, they used google blockly. Blockly developer tool helps to design custom blocks and implement the code generator, which translates blocks into textual codes. SmartApp interacts with the devices through events and in the meantime, it subscribes to events and takes action by invoking event handlers. After building the blocks in the environment properly, the compilation process takes place. It is a two-pass approach. First, it generates an Abstract Syntax Tree (AST) for blocks then the code generator takes care of the actual code generation as below.

There are several visual languages for programming IoT applications like Node-RED, Scratch for Arduino, BlocklyDuino, etc. On the other hand, Smart Block is a block-based visual environment for SmartThings.

This approach enables users to program SmartThings visually, which is more practical when the user is not very good at writing programs textually. And this makes the program to be syntactically correct, but the downside of this approach is it needs to go through some processing to generate the actual code for the hardware, which takes more time.

When building the Swarm Intelligence Framework, we can use the idea of a visual tool for modeling the swarm behaviors. We can represent Swarm behaviors using blocks and switch from one behavior to another by Event-Condition-Action rules.

C. Flowboard: A Visual Flow-Based Programming Environment for Embedded Coding [25]

In many fields such as science, technology, engineering, and mathematics, embedded programming become an important part. Learning embedded programming requires both hardware electronic-related knowledge and coding knowledge and skills. This article tries to understand if the visual flow-based approach can help to develop embedded programs rather than the usual imperative programming paradigms. In the traditional embedded programming approach user needs to type their program as a sequence of statements. This will not only require the programming languages but also leads to syntax errors. Also, block-based programming environments like scratch provide a graphical editor so the user can assemble the program using visual blocks. In a flow-based programming approach, programs are not sequences of commands. It uses a network of processing nodes through which the data flows and is manipulated. In FBP, parallel processing in one program is pretty straight forward unlikely in imperative programming. Flow-based approaches are used in many commercial IDEs in many domains. As an example, we can show LabView and Max/MSP IDEs. Many FBP-based embedded environments like Microflo and XOD have two missing aspects. They are liveness and Seamlessness. In flow-board it addresses this liveness and seamlessness which is more challenging to address using imperative programming environments. Flowboard [26] contains an Arduino Uno board and another micro-controller with 18 electronic switches. Users can create programs using the visual, flow-based editor on the iPad. They can drag nodes into the visual editor and link those nodes together by virtual wires between them.

In this, it tries to use a flow-based approach for programming embedded devices, which is more effective for the end user. The user doesn't need to write the program manually in a textual representation. This helps users to develop embedded programs even though they don't have skills in programming. Also swarming strategies [27] is also need abstract from the VPLs.

Also abstracting required forming pattern in swarm operations the formation would be a good strategy when finding objects there fore form patterns [28] and algorithms play a major role.

When developing the swarm intelligence framework, we can get the idea of a flow-based approach since we can easily model the swarm behaviors and transition from one swarm behavior to another together with a help of a state machine. [29] [30]

VII. Conclusion

According to above particulars we observed in order to have a software framework should be loosely coupled between different communication nodes.In the GSDF framework, the objective is similar to ours which is to build a swarm programming framework to easily build programmes for swarm robots. We can study and analyze how ROS integrated into this framework and how they use ROS features and also we can follow similar kinds of communication layer abstraction and all. But what was missing here is how we can support different microcontroller architectures when we are programming. Such as if we wrote a program for the Atmel family microcontroller, how we make it compatible with another kind of microcontroller such as ESP family or PIC microcontroller family. A requirement of the abstraction by hiding low level functionalities and adjustments in the software framework. Different approach 'Direct manipulation' for code less approach implemented to reduce the complexity. EmSBot is able to withstand a small number of hardware platforms. Also it is able to handle resource constraints and real time operability issues. And finally EmSBotScript which is a virtual machine based approach that solves the major problem of heterogeneity and various other memory related issues with fault tolerance. After investigating the above approaches, a virtual machine based approach such as EmSBotScript is the more convenient way suitable considering the software framework for the swarm robotic project.

References

[1] Navarro, I naki, and Fernando Mat ıa. "An introduction to swarm robotics." International Scholarly Research Notices 2013 (2013).

[2] E. S ahin, "Swarm robotics: from sources of inspiration to domains of application," in Swarm Robotics Workshop: State-of-the-Art Survey, E S ahin and W. Spears, Eds., Lecture Notes in Com- puter Science, no. 3342, pp. 10–20, Berlin, Ger- many, 2005.

[3] V. Trianni, R. Groß, T. H. Labella, E. S ahin, and M. Dorigo, "Evolving aggregation behaviors in a swarm of robots," in Proceedings of the 7th Eu- ropean Conference on Artificial Life (ECAL '03), W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, Eds., vol. 2801 of Lecture Notes in Artificial Intelligence, pp. 865–874, Springer, Heidelberg, Germany, 2003.

[4] E. Bah ceci and E. S ahin, "Evolving aggre- gation behaviors for swarm robotic systems: a sys- tematic case study," in Proceedings of the IEEE Swarm Intelligence Symposium, pp. 333–340, June 2005.

[5] A. Howard, J. Maja Mataric, and S. Gau- rav Sukhatme, "Mobile sensor network deploy- ment using potential elds: a distributed, scalable solution to the area coverage problem," in Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02), Fukuoka, Japan, June 2002

[6] E. Martinson and D. Payton, "Lattice for- mation in mobile autonomous sensor arrays," in Proceedings of the Swarm Robotics Workshop: State-of-the-art Survey , E. S ahin and W. Spears, Eds., no. 3342, pp. 98–111, Berlin, Germany, 2005.

[7] L. Chaimowicz, N. Michael, and V. Kumar, “Controlling swarms of robots using interpolated implicit functions,” in Proceedings of the IEEE In- ternational Conference on Robotics and Automa- tion (ICRA ’05) , pp. 2487–2492, April 2005.

[8] G. Lee and Y. C. Nak, “Self-configurable mobile robot swarms with hole repair capabil- ity,” in Proceedings of the IEEE/RSJ Interna- tional Conference on Intelligent Robots and Sys- tems (IROS ’08), pp. 1403–1408, September 2008.

[9] M. Lindh e and K. H. Johansson, “A forma- tion control algorithm using voronoi regions,” in CTS-HYCON Workshop on Nonlinear and Hybrid Control, 2005.

[10] C. Jones and M. J. Matari c, “Adaptive division of labor in large-scale minimalist multi- robot systems,” in Proceedings of the IEEE/RSJ Inter- national Conference on Intelligent Robots and Systems, pp. 1969–1974, Las Vegas, Nev, USA, October 2003.

[11] A. T. Hayes, A. Martinoli, and R. M. Goodman, “Swarm robotic odor localization: off- line optimization and validation with real robots,” Robotica, vol. 21, no. 4, pp. 427–441, 2003

[12] C. R. Kube and E. Bonabeau, “Coopera- tive transport by ants and robots,” Robotics and Autonomous Systems , vol. 30, no. 1, pp. 85–101, 2000.

[13] A. Howard, L. E. Parker, and G. S. Sukhatme, “e SDR experience: experiments with a large-scale heterogeneous mobile robot team,” in Proceedings of the th International Symposium on Experimental Robotics, pp. 121–130, Singapore, June 2004.

[14] M. A. Efremov and I. I. Kholod, ”Architec- ture of Swarm Robotics System Software Infras- tructure,” 2020 9th Mediterranean Con- ference on Embedded Computing (MECO), 2020, pp. 1-4, doi: 10.1109/MECO49872.2020.9134247.

[15] Cepeda, J. S., Chaimowicz, L., Soto, R., Gordillo, J. L., Alanís-Reyes, E. A., & Carrillo-Arce, L. C. (2012). A behavior-based strategy for single and multi-robot autonomous exploration. Sensors, 12(9), 12772-12797.

[16] Cybulski, P., & Zieliński, Z. (2021). UAV swarms behavior modeling using tracking bigraphical reactive systems. Sensors, 21(2), 622.

[17] St-Onge, D., Varadharajan, V. S., Li, G., Svogor, I., & Beltrame, G. (2017). ROS and Buzz: consensus-based behaviors for heterogeneous teams. arXiv preprint arXiv:1710.08843.

[18] Dassanayaka, M., Bandara, T., Adikari, N., Nawinne, I., & Ragel, R. (2020, July). A Programming Framework for Robot Swarms. In 2020 Moratuwa Engineering Research Conference (MERCon) (pp. 578-583). IEEE.

[19] Yi, Wei, et al. ”An actor-based program- ming framework for swarm robotic systems.” 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020.

[20] Pinciroli, Carlo, and Giovanni Beltrame. ”Buzz: An extensible programming language for heterogeneous swarm robotics.” 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016.

[21] Chang, X., Cai, Z., Wang, Y., Yi, X., & Xiao, N. (2017, August). GSDF: A generic development framework for swarm robotics. In International Conference on Intelligent Robotics and Applications (pp. 659-670). Springer, Cham.

[22] Preiss, J. A., Honig, W., Sukhatme, G. S., Ayanian, N. (2017, May). Crazyswarm: A large nano-quadcopter swarm. In 2017 IEEE International Conference on Robotics and Automation (ICRA) (pp. 3299-3304). IEEE.

[23] Ray, Partha Pratim. ”A survey on visual programming languages in internet of things.” Sci- entific Programming 2017 (2017).

[24] N. Bak, B. Chang and K. Choi, ”Smart Block: A Visual Programming Environment for SmartThings,” 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMP- SAC), 2018, pp. 32-37, doi: 10.1109/COMP- SAC.2018.10199.

[25] Anke Brocker, Simon Voelker, Tony Zelun Zhang, Mathis M uller, and Jan Borchers. 2019. Flowboard: A Visual Flow-Based Programming Environment for Embedded Coding. In Ex- tended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems (CHI EA ’19). Association for Computing Machin- ery, New York, NY, USA, Paper INT019, 1–4. https://doi.org/10.1145/3290607.3313247
—-change from here —-

[26] Hunt, E. R., Jones, S., Hauert, S. (2019). Testing the limits of pheromone stigmergy in high-density robot swarms. Royal Society open science, 6(11), 190225.

[27] Zhong, V. J., Dornberger, R., & Hanne, T. (2018). Comparison of the behavior of swarm robots with their computer simulations applying target-searching algorithms. International Journal of Mechanical Engineering and Robotics Research, 7(5).

[28] Gambardella, M. D. L. M., Martinoli, M. B. A., & Stützle, R. P. T. (2006, September). Ant colony optimization and swarm intelligence. In 5th international workshop, Springer.

[29] Arvin, F., Murray, J., Zhang, C., & Yue, S. (2014). Colias: An autonomous micro robot for swarm robotic applications. International Journal of Advanced Robotic Systems, 11(7), 113.

[30] Amjadi, A. S., Raoufi, M., Turgut, A. E., Broughton, G., Krajník, T., & Arvin, F. (2019). Cooperative pollution source localization and cleanup with a bio-inspired swarm robot aggregation. arXiv preprint arXiv:1907.09585.