Robot Based Computing System: an Educational Experience *

Diletta Cacciagrano, Rosario Culmone, Leonardo Mostarda,
Alfredo Navarra and Emanuele Scala

Abstract Robot based computing systems have been widely investigated in the last years. One of the main issues is to solve global tasks by means of local and simple computations. Robots might be cooperative or competitive, still the algorithm designer has to detect a way to accomplish the desired task. In this paper, we propose a platform made up of small and self-propelled robots with very limited capabilities in terms of computing resources, storage and sensing. In particular, we consider cheap robots moving within a confined area. The area is suitably coloured so as to be able for a robot endowed with a light sensor to reasonably detect its position. Moreover, robots can communicate with each other by exchanging short messages. Based only on those weak capabilities, we show how it is possible to realise interesting basic tasks. Apart for the relevance in educational contexts, our platform also represents an interesting case study for the main question posed in the literature about the minimal settings under which interesting tasks can be distributively solved.

1 Introduction

Robotic systems have been widely studied in the last decades. Robotics finds applications in many field of research and it involves both computer scientists and engineers. In fact, facing robotic issues concerns the design, the construction, the operation, and the use of robots. In particular, two main field of research have been investigated so far: *swarm robotics* (e.g., see [2, 20]) and *modular robotics* (e.g., see [3]).

Diletta Cacciagrano, Rosario Culmone, Leonardo Mostarda, Emanuele Scala
Computer Science Division, University of Camerino, Italy
e-mail: {name.surname}@unicam.it

Alfredo Navarra
Mathematics and Computer Science Department, University of Perugia, Italy
e-mail: alfredo.navarra@unipg.it

* Work supported by the Italian National Group for Scientific Computation GNCS-INdAM.

The former is mainly about robotic systems in which interconnected entities can somehow recover from failures or rearranging themselves in order to accomplish the required task (eg, see [23]). Their main objective is to achieve systems that are more versatile, affordable, and robust than their standard counterparts, at the cost of a probable reduced efficacy for specific tasks, see, e.g. [1, 22].

The latter mainly differ from modular robotics as individual robots in the system do not need to be connected with each other all the time, but they are usually fully autonomy mobile units (see, e.g., Kilobot [19]). The interaction among robots specified by means of robots' capabilities should lead to a desired collective behaviour. The approach has been mainly investigated in the field of artificial swarm intelligence, but it finds applications also in biological studies of insects, ants and other fields in nature, where swarm behaviour occurs. The research concerning swarm robotics is mainly focused on theoretical aspects, by considering robot systems in the abstract, where capabilities of the robots as well as the complexity of the environment are reduced to their minimum. Basic models widely investigated in this context are the Amoebot model [12, 13], and the more recent Silbot [8, 9], Moblot [4], and Pairbot [15] models. One of the main issue faced when dealing with such models is that they help, in general, in rigorously analyse the designed algorithms, hence providing new theoretical insights that subsequently also extend the practical aspect of the studied systems.

One of the models well investigated in swarm robotics is certainly $\mathcal{OBLOT}$ (see, e.g., [5, 6, 7, 10]). Such a model can be considered as a sort of framework within which many different settings can be manipulated, each implied by specific choices among a range of possibilities, with respect to fundamental components like time synchronisation as well as other important elements, such as memory, orientation, mobility and communication. Settings are often maintained at their minimum.

From a practical view point, the technology required to implement algorithms designed within $\mathcal{OBLOT}$ does not rely on special sensors nor actuators. Hence, cheap hardware might be used and experimented. An example of real robots working this way can be found in [11, 18, 21]. In general, robots' capabilities are maintained as weak as possible so as to understand what is the limit for the feasibility of the problems. Moreover, the less assumptions are made, the more a resolution algorithm is robust with respect to possible disruptions.

In this paper, we propose a new practical swarm robotic system. Our study started from educational purposes but we believe it can find many interesting applications, suggesting non-trivial solutions in case of reduced capabilities allowed to the robots.

2 System model and communication protocol

We envision and implement a robotic system where a set of identical robots is equipped with the following minimum components: one sensor, one actuator and one communication subsystem.

All robots operate in a finite environment, acquiring data from it through the sensor (position) and interacting with it through the actuator (motion). In what follows we frequently refer to robots as *ANTs*. An optional *monitor* (laptop) module is used to start, stop and monitoring the robots; it is connected to the same communication subsystem of ANT_s, but it does not interact with the environment nor participate to the robots' behaviour. The robot model is depicted in Fig. 1.

Other important constraints for robots regard the limited computational and memorisation capabilities. Those lead to keep main attention on efficient algorithms that cannot count on complex sensors and actuators.

Further, this choice is aimed at experimenting with real-time systems for resource constrained devices. To this ending, it is important to accurately define the memory settings for the data structures in each single task, as well as the composition of tasks at a higher level.

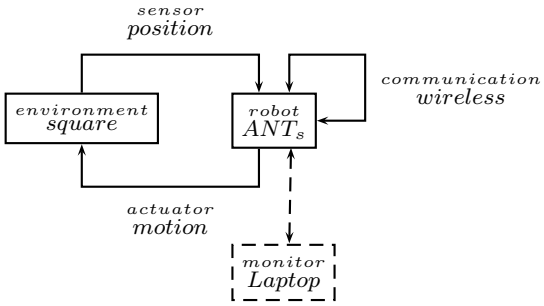


Fig. 1: System model.

The communication subsystem we consider implements a mutual-exclusive ring protocol where a set of z robots, denoted with $R = \{r_0, \ldots, r_z\}$, provide to each other their position in the environment. What a robot sends on its turn t, with $t \in [0, z]$, is a broadcast message $b_t = r_t||r_{t+1}||currPos$ where r_t is the identifier of itself, r_{t+1} is the identifier of the robot that will broadcast the message b_{t+1} on the next turn and $currPos$ is the current position of r_t on the environment. An additional payload can be provided with b_t, consisting

of information for verifying the correct reception of the message and a bit to control the start and stop of all robots by the monitor.

Each robot is configured to be in two states, the listening state S_L and the broadcasting state S_B. When it is not his turn to communicate, a robot is always in S_L, parsing the message received form the only robot r_t of the ring enabled to broadcast. Once received, a robot in S_L obtains from the message b_t the current position sent by r_t, storing it locally, and checks if r_{t+1} matches its identifier. If so, the robot switches to state S_B. At each turn t, the r_t robot in S_B assembles and broadcasts the message b_t by deriving the identifier of the next ring member and the current position. In order to detect faulty ring members, two local timeouts are set in each robot, a global timeout T_g and a response timeout T_r. The first is of the order of seconds, the time required for a complete ring round trip. It is useful both to start the protocol, ensuring that the first robot started enters in S_B, and to prevent the protocol from ending. The second is used in the S_B state so that the broadcasting robot r_t verifies that the next member of the ring is active. If not, the robot r_t will continue this acknowledgement i times with the successors of r_{t+1} until a robot r_{t+1+i} in the ring responds with a broadcast message.[2] All the robots $r_{t+1}...r_{t+i}$ are declared inactive for the turn t and will be allowed to re-enter in the successive turns. Our ring protocol is implemented by a single infinite loop algorithm installed in all robots, see Algorithm 1. We call this program *RACom* together with two subroutines *listen* and *broadcast*.

2.1 Implementation details

Our implementation of the model is made up of small ANTs, a platform on which they operate and an optional monitoring computer. Each ANT (see Fig. 2a) consists of:

- One load-bearing aluminium structure on which it is positioned a Arduino UNO (ATmega328P at 16 MHz, 2 KB SRAM, 32 KB);
- Two stepper motor SY35ST26-0284A with 200 step and Adafruit Motor Shield v2.3;
- One RGB light sensor TCS34725;
- One serial wireless module HC-12 433 MHz SI4438;
- Two 9 volts rechargeable battery 900 mAh.

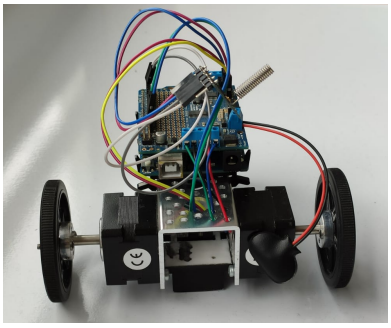
The platform on which the ANTs move is a two meters square. The surface is made of coloured photographic paper (see Fig. 2b). Colours are distributed so as a robot can deduce its position by sensing the corresponding colour. The system can be monitored using a notebook with a specific ANT (Arduino and communication module only) connected via USB. The software has been designed in C with FreeRTOS [14]. Four subroutines have been realised:

[2] Sums in the subscript of the robots' identifiers must be considered modulo z.

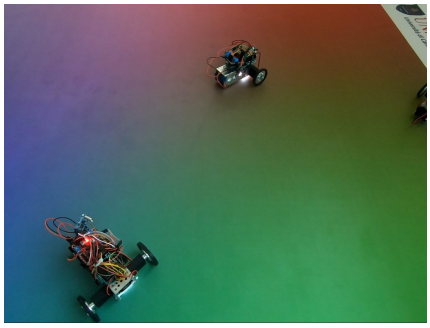
Algorithm 1 RACom

```
1: while true do
2:     if init = false then
3:         Let z be the number of robots
4:         Let myID be the robot's identifier
5:         Start global timeout T_g
6:         Set state = S_L
7:         Set init = true
8:     end if
9:     if T_g not expired then
10:        listen()
11:    end if
12:    if T_g expired then
13:        broadcast()
14:    end if
15: end while
16:
17: function BROADCAST :
18:    Set i = 1, t = myID, state = S_B
19:    while state = S_B do
20:        Set t = (t + i) mod z
21:        Set currPos
22:        Set b_myID = myID||r_t||currPos
23:        Broadcast message b_myID
24:        Start response timeout T_r
25:        while T_r not expired do
26:            listen()
27:        end while
28:        i = i + 1
29:    end while
30: end function
31:
32: function LISTEN :
33:    Read b_t from serial
34:    if b_t not empty then
35:        Parse r_{t+1} from b_t
36:        if r_{t+1} = myID then
37:            broadcast()
38:        end if
39:        if r_{t+1} ≠ myID then
40:            Set state = S_L
41:        end if
42:        Restart T_g
43:    end if
44: end function
```

- *Communication routine.* It deals with the communications among ANTs. The driver of the HC-12 module performs a timed reading of the buffer and transfers the data to the Communication routine which interprets them and sends them to the other ANTs involved in the communication;



(a) One of our own built ANTs.



(b) The coloured surface among which ANTS move.

Fig. 2: Our implementation of the ANTs system.

- *Position routine.* It deals with the identification of the positioning of the ANT. The TCS34725 module drive takes timed readings of the RGB values of the surface under the ANT. The values are transferred to the *Position routine* which interprets them by two functions:

  $map(r, g, b) \rightarrow sites$: Given a tuple of red, green, blue values, *map* produces the set *sites* composed by all the coordinates of the plane that are consistent with these values. Note that although the plane has been generated in such a way that the uniqueness of colours for each point is guaranteed, the low sensitivity of the sensors or interference can produce incorrect reading values;

  $trace(traces, sites) \rightarrow (x, y)$: Given two sets of points, *traces* and *sites*, the trace function identifies the point where the ANT is located, selecting among the sites those that are consistent with the path taken.

- *Motion routine.* It deals with the motion of the ANT. The motion is made by sending commands to *Adafruit Motor Shield*. A control has been created that allows to issue commands for the advancement in a straight line, the curves with a given arc and control of the accelerations in order to have a fluid motion without jerks. This is also useful for preventing the wheels from losing grip. This feature makes it possible to monitor, by means of a feedback system, the real position of the ANT with respect to that reached after a movement.
- *Brain routine.* It deals the autonomous control of all the system. This routine allows the coordination of the system by interacting with all the other routines and realising its objectives. A mini control language has been created that allows to load the objectives with a laptop connected to the ANTs network. Once the relative objectives have been loaded on the ANTS, the function of the laptop is for supervision only (position and states of the ANTs) or it can be disconnected from the network.

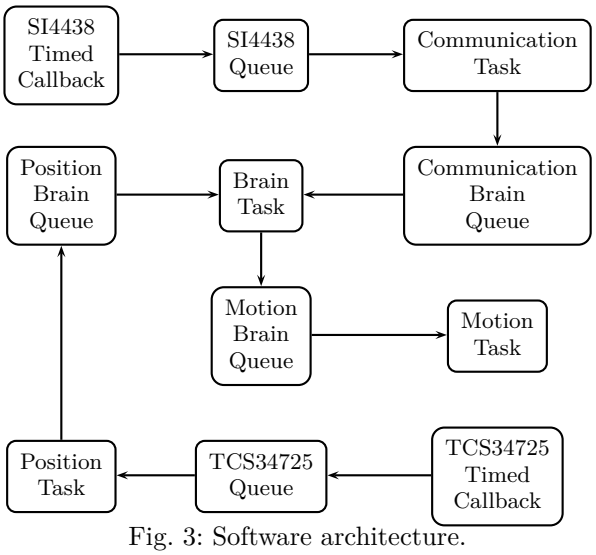


Fig. 3: Software architecture.

Timed callbacks have been created in order to manage the hardware modules TCS34725 (colour sensor) and SI4438 (wireless communication). The routines and callbacks are connected via queues for correct interactions. The general architecture is shown in Fig. 3.

2.2 Patrolling

An example of task for which we applied the ANTs system previously described concerns the *Patrolling*. The requirement is to continuously monitor/visit the area of interest (our coloured square surface) without incurring in colliding ANTs. Moreover, ANTs are subject to disruptions, that may interrupt their functioning. Hence, we aim to design a fault-tolerant distributed algorithm to solve Patrolling. Due to the reduced capabilities of the ANTs, a simple solution in terms of running complexity is also required.

In our system each ANT broadcasts its position according to the described mutual-exclusive ring protocol. In so doing, all ANTs have a map with the coordinates of all ANTs continuously updated. Referring to this map and its current position, an ANT calculates two movements, “going straight” or “dodging”. The idea is that, based on two consecutive positions of each other ANT, an ANT can understand whether along its (straight) way a collision may occur, in which case it decides to slightly deviate in order to avoid the positions occupied by the other ANTs. Deviations are also required if the border of the area to monitor is reached. If an ANT breaks, clearly it cannot communicate its position during its turn. Hence, all other ANTs can deduce

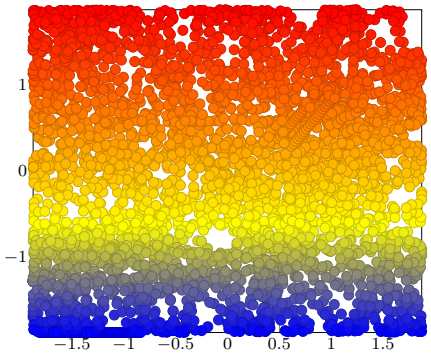


Fig. 4: ANTs patrol coverage.

that the ANT is broken by exploiting the associated timeouts not met by the broken ANT. In so doing, the failure will not alter the correct functioning of the ANTs. An ANT_i can avoid the border or any other ANT_j, with $i \neq j$, along its way, by calculating the dodging movement. ANT_i can in fact deduce the direction (in terms of angle α_{ij} wrt the coordinate system given by the area of interest) and hence the vector between the two coordinates (x_i, y_i) and (x_j, y_j). If α_{ij} is large enough and the obstacle is close enough (with respect to a predefined safety measures), ANT_i will turn according to the sign of the angle, otherwise it continues to go straightforward. A similar approach is used in [16]. Our solution differs in the use of less expensive hardware and that the calculation is done on the basis of the positions of the ANTs.

We build a simulator for our solution in ARGoS [17]. The goal is to trace the coverage of the area of interest where ANTs perform the patrolling. The environment is configured to be a square of $(-2, 2)$ length per side in the ARGoS unit. In there, 10 ANTs are randomly placed with uniform distribution at the start of the simulation. We run 80 seconds simulation and we detect 5 coordinates per second from each ANT, collecting 4000 coordinates in total. The speed of the ANTs is set in a reasonable way to avoid errors in the simulation. In Fig. 4 we show the coverage obtained, it is possible to notice a coverage of more than the 90% of the area.

3 Conclusions and Future Work

We have proposed a new platform for the implementation of modular robotics solutions where robots' capabilities have been reduced to their minimum. Although we started from educational intents, we believe the composed system has some potentials that can be exploited for designing non-trivial solutions in environments where robots cannot count on sophisticated hardware. We

have reported our experience by means of a basic task, namely the Patrolling. It is worth noting that in the proposed solution each ANT moves in a deterministic way. Instead, we could modify this behaviour by considering that each ANT calculates future positions in a random way such that, by knowing the future locations of the other ANTs, each ANT can decide in advance to travel a safe path. We conjecture that such an algorithm might be a good solution to the Patrolling task and it might be even faster in terms of number of iterations. We plan to verify the conjecture by making comparisons with our current system via the simulator, possibly defining also different movement policies.

Another direction of research that deserves investigation is about the possible tasks that can be approached within the simple proposed system. For instance, an interesting task where a competitive rather than collaborative behaviour must be designed has been faced while investigating the so-called *Puss in the corner* puzzle. Six ANTs are placed at the corner of a regular hexagon and one is placed in the centre. From this initial configuration, a configuration must be reached where each ANT does not occupy its initial position. In this case, the choice of each ANT competes with all the other ANTs to achieve its goal.

References

1. Ahmadzadeh, H., Masehian, E., Asadpour, M.: Modular robotic systems: Characteristics and applications. J. Intell. Robotic Syst. **81**(3-4), 317–357 (2016). DOI 10.1007/s10846-015-0237-8
2. Amir, M., Bruckstein, A.M.: Minimizing travel in the uniform dispersal problem for robotic sensors. In: Proc. 18th Int.'l Conf. on Autonomous Agents and MultiAgent Systems (AAMAS), pp. 113–121. International Foundation for Autonomous Agents and Multiagent Systems (2019)
3. Christensen, A.L.: *Self-Reconfigurable Robots - An Introduction.* Artif. Life **18**(2), 237–240 (2012). DOI 10.1162/artl_r_00061
4. Cicerone, S., Di Fonso, A., Di Stefano, G., Navarra, A.: MOBLOT: molecular oblivious robots. In: F. Dignum, A. Lomuscio, U. Endriss, A. Nowé (eds.) AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021, pp. 350–358. ACM (2021)
5. Cicerone, S., Di Stefano, G., Navarra, A.: Asynchronous arbitrary pattern formation: the effects of a rigorous approach. Distributed Computing **32**(2), 91–132 (2019)
6. Cicerone, S., Di Stefano, G., Navarra, A.: Solving the pattern formation by mobile robots with chirality. IEEE Access **9**, 88,177–88,204 (2021). DOI 10.1109/ACCESS.2021.3089081
7. Cicerone, S., Di Stefano, G., Navarra, A.: A structured methodology for designing distributed algorithms for mobile entities. Information Sciences **574**, 111–132 (2021). DOI 10.1016/j.ins.2021.05.043
8. D'Angelo, G., D'Emidio, M., Das, S., Navarra, A., Prencipe, G.: Asynchronous silent programmable matter achieves leader election and compaction. IEEE Access **8**, 207,619–207,634 (2020)
9. D'Angelo, G., D'Emidio, M., Das, S., Navarra, A., Prencipe, G.: Leader election and compaction for asynchronous silent programmable matter. In: Proc. 19th Int.'l

Conf. on Autonomous Agents and Multiagent Systems (AAMAS), pp. 276–284. International Foundation for Autonomous Agents and Multiagent Systems (2020)
10. D'Angelo, G., Di Stefano, G., Navarra, A.: Gathering on rings under the look-compute-move model. Distributed Computing **27**(4), 255–285 (2014)
11. Das, S., Focardi, R., Luccio, F.L., Markou, E., Squarcina, M.: Gathering of robots in a ring with mobile faults. Theor. Comput. Sci. **764**, 42–60 (2019). DOI 10.1016/j.tcs.2018.05.002
12. Daymude, J.J., Hinnenthal, K., Richa, A.W., Scheideler, C.: Computing by programmable particles. In: P. Flocchini, G. Prencipe, N. Santoro (eds.) Distributed Computing by Mobile Entities, Current Research in Moving and Computing, *Lecture Notes in Computer Science*, vol. 11340, pp. 615–681. Springer (2019). DOI 10.1007/978-3-030-11072-7_22
13. Derakhshandeh, Z., Gmyr, R., Strothmann, T., Bazzi, R.A., Richa, A.W., Scheideler, C.: Leader election and shape formation with self-organizing programmable matter. In: A. Phillips, P. Yin (eds.) DNA Computing and Molecular Programming - 21st International Conference, DNA 21, Boston and Cambridge, MA, USA, August 17-21, 2015. Proceedings, *Lecture Notes in Computer Science*, vol. 9211, pp. 117–132. Springer (2015). DOI 10.1007/978-3-319-21999-8_8
14. Fei, G., Long, P., Luc, P., Martin, T.: Open source freertos as a case study in real-time operating system evolution. Journal of Systems and Software **118**, 19–35 (2016)
15. Kim, Y., Katayama, Y., Wada, K.: Pairbot: A novel model for autonomous mobile robot systems consisting of paired robots (2020)
16. Piardi, L., Lima, J., Oliveira, A.S.: Multi-mobile robot and avoidance obstacle to spatial mapping in indoor environment. In: 11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications, SIMULTECH 2021, pp. 21–29 (2021)
17. Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., et al.: Argos: a modular, parallel, multi-engine simulator for multi-robot systems. Swarm intelligence **6**(4), 271–295 (2012)
18. Pásztor, A.: Gathering simulation of real robot swarm. Tehnicki vjesnik **21**(5), 1073–1080 (2014)
19. Rubenstein, M., Ahler, C., Nagpal, R.: Kilobot: A low cost scalable robot system for collective behaviors. In: IEEE Int.'l Conf. on Robotics and Automation (ICRA), pp. 3293–3298. IEEE (2012). DOI 10.1109/ICRA.2012.6224638
20. Sahin, E.: Swarm robotics: From sources of inspiration to domains of application. In: E. Sahin, W.M. Spears (eds.) Swarm Robotics, SAB 2004 Int.'l Workshop, Santa Monica, CA, USA, July 17, 2004, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 3342, pp. 10–20. Springer (2004). DOI 10.1007/978-3-540-30552-1_2
21. Salman, M., Garzón-Ramos, D., Hasselmann, K., Birattari, M.: Phormica: Photochromic pheromone release and detection system for stigmergic coordination in robot swarms. Frontiers Robotics AI **7**, 591,402 (2020). DOI 10.3389/frobt.2020.591402
22. Tucci, T., Piranda, B., Bourgeois, J.: A distributed self-assembly planning algorithm for modular robots. In: E. André, S. Koenig, M. Dastani, G. Sukthankar (eds.) Proc. of the 17th Int.'l Conf. on Autonomous Agents and MultiAgent Systems (AAMAS), pp. 550–558. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM (2018)
23. Yim, M., Shen, W., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.S.: Modular self-reconfigurable robot systems [grand challenges of robotics]. IEEE Robotics Automation Magazine **14**(1), 43–52 (2007)