Creating an artwork with a modular robot composed of 1,824 modules

Benoît Piranda[1], Frédéric Lassabe[1], Rémy Tribhout[2], Grégory Lasserre[3], and Julien Bourgeois[1],[2]

Abstract The modular robots community has developed many different systems throughout the years, however, very few have been tested with a large number of modules. In this article, we present an artistic creation made of 1,824 *Blinky Blocks*. It is the first time a modular robot composed of such a number of modules has been assembled and programmed and this artwork has been certified by a Guinness World Record. We detail the hardware used for setting up this record as well as the software, from the firmware to the application as both aspects are very important to scale up in number of modules.

1 Introduction

Since the first attempt to create a robot composed of modules attached to each others [3], the domain is still looking for real applications. Although, the structure of modular robots presents theoretical advantages like versatility and robustness [12], these come at a price of a greater complexity. Having a modular robot composed of few modules cannot really show its advantages, that is why scaling up in numbers is important. Building and programming a modular robot composed of thousands of modules involves hardware and software problems to be solved. Our modular robots is composed of *Blinky Blocks* [6], originally developed at Carnegie Mellon University within the Claytronics project [4] and now redesigned at UBFC, FEMTO-ST Institute, CNRS within the Programmable Matter Project [1]. The idea of *Blinky Blocks* is to develop simple and affordable modules by using as much as possible standard components and removing troublesome elements like the battery. We also keep scalability as a driving force for the design, relying only on neighbor-to-neighbor communications and prohibiting the use of a bus. The firmware and the applications are designed to deal with faults as increasing the number of modules

Université Marie et Louis Pasteur, CNRS, institut FEMTO-ST, France e-mail: fname.lname@univ-fcomte.fr · Phigi, e-mail: fname@phigi.io · Scenocosme, e-mail: scenocosme@gmail.com

multiplies the causes of errors. In this article, we present an artwork created by the couple of artists, Scenocosme, and composed of 1,824 *Blinky Blocks*. To the best of our knowledge, it is the first time a modular robots composed of such a number of modules has been assembled and programmed which has been certified by a Guinness World Record. Finally, the work between researchers of the FEMTO-ST Institute and the artists of Scenocosme has given birth to many different artworks that have been displayed in various museums and exhibitions, attracting the interest of tens of thousands of visitors which were able to interact with the different artworks. These exhibits working all day long for months show that, despite its complexity, a modular robot is a robust technology.

2 The Programmable Matter Project

The vision behind programmable matter is to have a material able to completely change its physical appearance through the collective power of a vast number of mm-scale robots that can stick together and move around each other to form an overall material. The mm-scale robot should be able to compute, communicate, move, and adhere with other robots using just two basic mechanisms: a processing unit and electrostatic electrodes [5]. This objective has been defined by the Claytronics project [4], and continued by the Programmable Matter Project [1]. Within these two projects, different kinds of hardware have been built but moving requires complex hardware that is why a much simpler modular structure has been designed: the *Blinky Blocks*.

3 The *Blinky Blocks*

3.1 Hardware presentation

This section presents the *Blinky Block*'s hardware architecture. First, we describe the overall capabilities of a *Blinky Block*, as well as its shape and size. Second, we describe the memory architecture. Third, we present the communications interfaces and their mapping on *Blinky Blocks*' physical configuration.

The *Blinky Blocks* system is a modular distributed execution environment composed of centimeter-size blocks that are attached to each other using magnets (See Fig. 1). Each block is a 4 cm cube, embedding a processing unit (STM32F091CB micro-controller), neighbor-to-neighbor communications, sensors and actuators to implement a wide variety of distributed tasks. Each block embeds two sensors: a microphone and an inertial measurement unit (IMU) which can detect sounds, movements and by extension tapping, for example. Each block has also a speaker to play sounds and two RGB LEDs to glow in different colors which is an important user

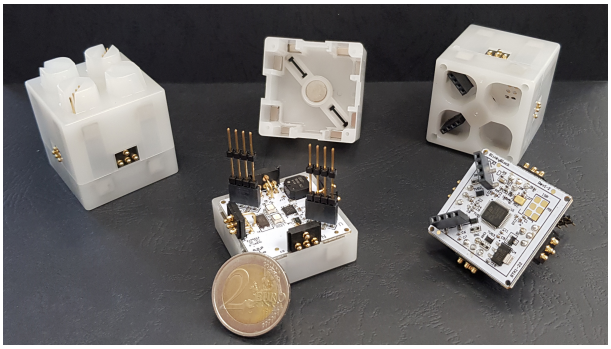

Fig. 1 The *Blinky Blocks* system V2: 4 cm cubes with a processing unit, neighbor-to-neighbor communications, sensors and actuators.

interaction possibility. *Blinky Blocks* can be moved by hand to change the topology of the ensemble but they cannot move by themselves. Power and communications pass through the pogo pin connectors on the sides and through pin connectors on top and bottom.

Concerning Memory architecture, *Blinky Blocks* have two main memories: a 128 KB flash memory, separated into 3 areas in our implementation, and a 32 KB SRAM memory used during programs execution (See Fig. 2). The three areas in the flash memory are the following:

- first 30 KB are reserved for our custom bootloader,
- next 2 KB are reserved to emulate a NVRAM, used to store up to 16 32-bits wide configuration values. 32 slots in this area allow to provide some wear-leveling algorithm in order to avoid exhausting read/write cycles too fast when changing *Blinky Blocks* persistent parameters.
- last 96 KB are dedicated to applications, that are implemented by *Blinky Blocks* users and uploaded through a dedicated protocol provided by the bootloader.

Flash memory ranges from address `0x8000000` to address `0x801ffff`.

Communication in *Blinky Blocks* deserves a description since its architecture is strongly linked to the physical layout of a *Blinky Blocks* ensemble. Each *Blinky Block* has 6 serial communication interfaces, one on each side. Therefore, *Blinky Blocks* can only communicate with their connected neighbors, thus presenting a network mapped on their physical layout. *Blinky Blocks* are able to identify their interfaces so they know where a message comes from (top, bottom, and so on) as well as to direct their outgoing messages to a particular neighbor.

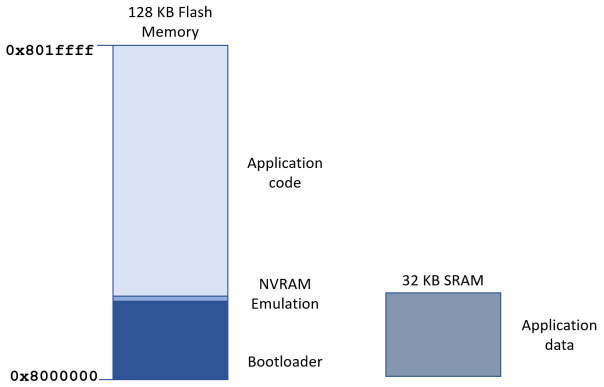

Fig. 2 Memory organization of the *Blinky Blocks* with the three areas in the flash and the SRAM.

3.2 Bootloader presentation

In this section, we present the *Blinky Blocks*' software architecture. First, we briefly describe the network stack we designed and implemented. Then, we deal with bootloader deployment and update, followed by a description of user applications upload through the bootloader. Finally, we provide a quick manual of the applications development API.

Communications are handled by a lightweight protocol designed for *Blinky Blocks*. First, a special character is broadcast periodically to maintain interfaces status (i.e. identifying which interface has a neighbor, and which one has none). Second, two characters are used to delimit regular frames.

Since *Blinky Blocks* can only communicate with their direct neighbors, propagating messages through a whole ensemble requires multi-hop messaging. In the current case, we rely on the construction of a spanning tree over the ensemble, with its root connected to a computer. Then, messages are routed top-down (top being the computer) for commands, and bottom-to-top for ensemble acknowledgments. Commands may range from lighting the LEDs in a given color to an application deployment over all *Blinky Blocks*.

Communication protocol is inspired from Open Systems Interconnection (OSI) network model, with layers, and their corresponding protocols are defined in the table of the Fig. 3.

The physical layer is composed of 6 Universal Asynchronous Receiver Transmitter (UART). Each UART connector has its own identifier so that a *Blinky Block* is able to associate a received frame to one of its sides, and to send frames to only one neighbor. UART operations are managed by the chip. Outgoing frames are read in a circular buffer provided by the upper layer, while incoming frames are stored to a circular buffer provided to upper layers.

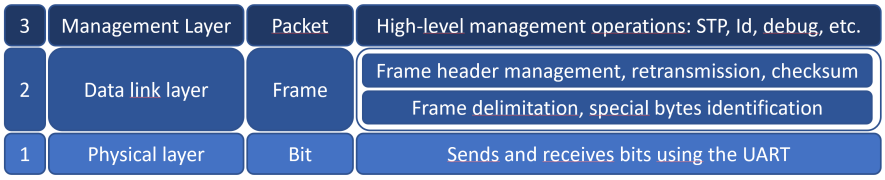

Fig. 3 *Blinky Blocks* network layers.

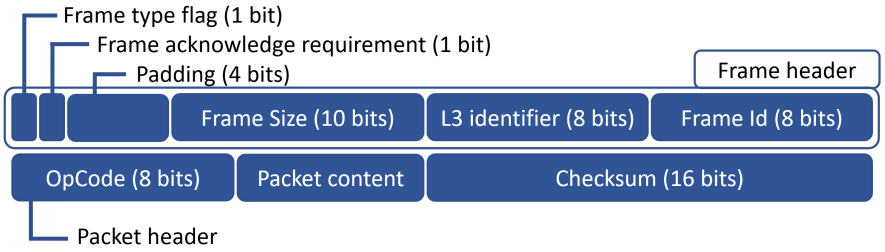

Fig. 4 Packet structure with the frame part and the packet part.

The data link layer is composed of two sublayers to manage frames that are composed of: a frame header, followed by the packet header, the packet content and the checksum[1]. as it can be seen on Fig. 4. Detailed fields are the following:

- Frame type flag (one bit): states whether the frame is a data frame (1) or an acknowledgment frame (0).
- Frame acknowledge requirement (one bit): states whether the frame must be acknowledged (1) or no (0).
- Frame size (10 bits): the complete frame size, in bytes, including headers and checksum.
- Layer 3 type identifier (8 bits): states the type of encapsulated layer 3 packet (STP, etc.).
- Frame id (8 bits) an identifier for the frame. It may be:
  - its own internal id when sending the frame (to match a returning acknowledgment frame).
  - a remote frame id in case the frame is an acknowledgment.

Acknowledgment relies on the following process: when sending a frame, the acknowledgment requirement bit is set to 1. Upon reception, the destination block checks the frame delimiter, its size, and its checksum. If all are valid, it sends back to the source a frame whose id is equal to the source frame id, whose frame type and acknowledgment requirements bits are both 0 and whose content is the acknowledged frame's checksum, followed by the acknowledgment frame's own checksum. When the source block does not receive an acknowledgment for 250 ms, it will retry

[1] We apply a 16 bits BSD checksum: https://en.wikipedia.org/wiki/BSD_checksum

sending the frame. There are three retries before dropping the frame and notifying the upper layer.

Layer 3 is a set of dedicated protocols. All protocols have varying header's length, with a similar first byte as an operation code (OpCode). Operation codes are dependent on the layer 3 implementation. According to the operation code and the layer 3 type, other header fields may follow, for instance, STP (Spanning Tree Protocol) has a parent request composed only of its operation code, while its parent accept packet is composed of an operation code followed by the size of the sub-tree. Layer 3 implementations are the following:

- STP: builds an overlay network without cycle to ease forward and backward transmissions between the tree root and the other *Blinky Blocks*.
- BBPP (*Blinky Block* Batch Programming Protocol): loads an application into a set of *Blinky Blocks* from one unique entry point (i.e. connection to a computer) by a flooding of the application code through the spanning tree.
- Device command: sends commands to *Blinky Blocks*, such as getting the configuration of the connected modules, updating LEDs color, emitting a sound, rebooting, going into the application and writing configuration parameters.
- Soft id: gives a software identification (32 bits) to all *Blinky Blocks*. Each id is unique within the set.
- Debug messages: used to send messages from any *Blinky Block* inside the spanning tree towards the root node (i.e. computer) allowing a user to get feedback from any *Blinky Block*.
- Standard messages: dedicated to user defined messages in applications deployed in *Blinky Blocks*. Each application defines its own operation codes and protocol.

Deploying the bootloader may be achieved with two procedures: the first one relies on the Serial Wire Debug (SWD) interface accessible through a special interface. This type of deployment can only be applied to one *Blinky Block* at a time. Second one relies on BBPP and a particular application.

With the SWD way, a *Blinky Block* must be plugged on a support with pins accessing the SWD port on the *Blinky Block* bottom. It accesses directly the flash memory to write into it. Since a *Blinky Block* automatically runs the bootloader at startup, it is the safest way to program the bootloader into a *Blinky Block*. However, for large deployments such as thousands of *Blinky Blocks*, the task becomes long and cumbersome, with possible manipulation errors. This is why we designed a special application to upgrade a bootloader into an ensemble of *Blinky Blocks* with an already operational bootloader.

This process relies on an application that has the bootloader ability to deploy applications with BBPP, but is located in application space, starting at the address `0x8008000`. After using the old bootloader to deploy this application, we run the application. Thus, the address range of the bootloader is not accessed anymore and can be erased/written on. We then use this application as a bootloader to send the updated version of the bootloader to all *Blinky Blocks*.

Application deployment relies on the BBPP implementation within the bootloader. The user writes the application code in an IDE provided by ST Microelec-

tronics using C language, compiles it to the ARM Cortex M0 architecture and gets a binary .hex file. Using this .hex file, it can start a deployment over a set of connected *Blinky Blocks*.

Application deployment is typically done in the order of tens of seconds, with the limiting factor being the time to erase the target flash memory space multiplied by the depth of the spanning tree. For a compact rectangle composed of 450 *Blinky Blocks*, the depth of the spanning tree is equal to 42, and it therefore requires less than one minute for the whole process.

Programming a *Blinky Blocks* ensemble requires a connection from a computer to one *Blinky Block* from the ensemble. The computer is the root node of the tree, it triggers a spanning tree construction over the ensemble. When the tree is complete, the programming process starts. After receiving the program parameters (address, and size), a *Blinky Block* first tests if it is a valid parameters set. If so, it erases the target flash pages, and propagates the parameters to all its children. When reaching leaves, a confirmation message is sent to the parent, and it goes back to the root node. Then, the program is decomposed into small chunks (128 bytes each), that go from a parent node to its children. *Blinky Blocks* can propagate chunks to their subtrees while requesting next chunks from their parent. This way, programming an application on a *Blinky Blocks* ensemble is parallelized and very fast while depending mainly on the depth of the spanning tree.

Programming an application is performed by a custom program at default microcontroller address. We consider this program as a bootloader, with extended capabilities such as:

- Constructing a spanning tree on a set of *Blinky Blocks*,
- Assigning unique identifiers to a set of *Blinky Blocks*,
- Programming the application into a set of *Blinky Blocks*,
- Running the application by restarting at its address.

With these features, the usual process is to first write and compile an application on a computer, second is to connect to a *Blinky Block* ensemble, third to make a spanning tree, fourth to send the application, and last, to run the application.

3.3 Blinky Block applications

Blinky Block applications are stored in a specific C file of the project. In a way similar to Arduino, the developer must define at least three functions in this file: `void BBinit()` called once when the block starts, `void BBloop()` called indefinitely while the *Blinky Block* is powered and `process_standard_packet(...)` that must associate a processing with the reception of a message. Some functions are dedicated to the interactions with sensors and actuators. The main tools are:

- `setColor(color)` changes the color of the *Blinky Block*, color may be an integer or a color name to access to 20 different predefined colors.

- `sendMessage(port,data[],size,has_ack);` sends a message to the neighbor connected to the chosen port (*port*). The message is an array of *size* bytes stored in *data*. A message is typically composed of the type of the message as an operation code on the first byte and the embedded data in the next bytes.
- `getId()` returns the software defined identifier of the *Blinky Block*.
- `is_connected(port)` returns 1 if the local port identified by *port* is connected to another *Blinky Block*.

And we propose a set of similar functions to manage all the other capabilities of the *Blinky Block*.

For example, the following code details a starting function that changes the color of *Blinky Block* #1 to red and the color of the others to green. The *Blinky Block* #1 begins the program by sending a message labeled *DISTANCE_MSG* to all its connected neighbors.

```
#define DISTANCE_MSG 1
#define ACK_MSG 2

void BBinit() {
  if (getId() == 1) {
    setColor(RED);
    // send message to all neighbors
    uint8_t data[2]={DISTANCE_MSG, 1};
    for (uint8_t p=0; p<NB_SERIAL_PORT; ++p) {
      if (is_connected(p)) {
        sendMessage(p,data,2,1);
      }
    }
  } else setColor(GREEN);
}
```

To receive this message, a function that will associate a reaction to the reception of a message has to be defined. The following code shows that, when a *Blinky Block* receives a new message, it executes a function depending on the the label of the message (stored in the first byte of its data).

```
uint8_t process_standard_packet(L3_packet *packet) {
  switch (packet->packet_content[0]) {
    case DISTANCE_MSG: return distanceUpdate(packet); break;
    case ACK_MSG: return ackFunc(); break;
  }
  return 0;
}
```

Configurations and codes are usually evaluated in *VisibleSim* simulator [10],[11] before being compiled for the *Blinky Block* hardware (as shown in Fig. 5). *VisibleSim* is a behavioral simulator that simulates all the events that occur in all the connected robots. All these events are timestamped and we simulate exactly the diffusion time of each robot which gives a good simulation of the embedded code running on a set of *Blinky Blocks* (we show in [9] that results are identical in simulation and real *Blinky Blocks* after thousands of messages). *VisibleSim* mimics the lighting in color of the *Blinky Blocks*, producing realistic visual simulations.

4 Technical Setup

This section describes the technical setup of our 1824 *Blinky Blocks* artwork and its power supply. The technical setup consists in a 2D setup of 1,824 *Blinky Blocks* grouped in 9 meta-modules of 200 *Blinky Blocks* each, and 6 vertical links between meta-modules, composed of 4 *Blinky Blocks* each (See Fig.5). It is powered through 150 entry points plugged to 5 V power supplies.

Except from *Blinky Blocks* connected to links between meta-modules, each *Blinky Block* has only two neighbors. The *Blinky Blocks* connected to inter-meta-modules links have three neighbors. From the network's point of view, each meta-module can be modeled as a loop.

5 Artwork

This section describes the artistic point of view and prior work that led to the current large setup. The artistic approach aims at hijacking the *Blinky Blocks* to create sculptural and interactive works with organic behaviors. To do this, artistic work and technical research have been conducted on the *Blinky Blocks* by the couple of artists Scenocosme.

Our body is a continuous sensory interface with the world. The skin plays its role also being a protective and porous border: it perceives and emits heat at the same time for example. As described by David Le Breton in [8], it is alive in that it breathes, exchanges with the environment, emits odors, and translates states of mind through its texture, its heat, and its color. Between the outside and the inside, it establishes the passage of the stimulations and of the senses. Instance of separation, it encloses the individuality but it is simultaneously a place of exchange with the world, as it transmits heat, light, enjoyment or pain.

"*Rhizome 001*" (See Fig.7) is a sculptural and interactive work that behaves like a living organism. Like a root, this creation takes shape and unfolds in space to better perceive and feel its environment. This hybrid electronic ecosystem is composed of more than 120 robotic cellular structures, independent but attached to each

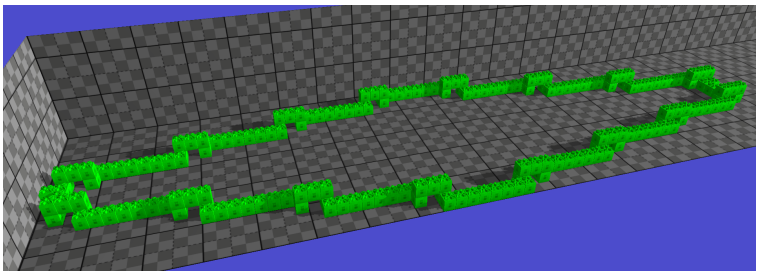

Fig. 5 One meta-module, *VisibleSim* rendering.

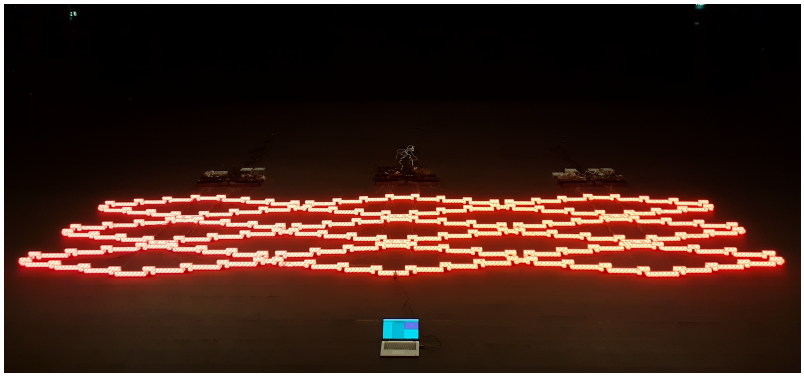

Fig. 6 *Arborescence 003*, a sculptural and interactive work composed of 1,824 *Blinky Blocks*, establishing a Guinness World Record in modular robotics.

other. Each group of cells is grafted to a transparent membrane that allows it to feel the caresses and breath of the spectators. Each robotic structure also perceives the sounds and reacts with different feedbacks according to the intonations of the voices more or less low to high and the duration of the vocalizations. Each of the 120 electronic cells is retroactive. They emit different sound behavioral scenarios, rhythms and light intensities in response to the stimuli of the audience. The work deploys on its surface, 120 micro-controllers, 120 independent micro-speakers, 120 light sources, etc. Being sensitive, the *Blinky Blocks* also influence each other, between neighboring cells, like cooperative living organisms. ”Rhizome” is inspired by the modes of communication and arrangement of plants, corals, termites, fireflies, micro-organisms, etc.

”*Arborescence 003*” is a sculptural and interactive work composed of 1,824 *Blinky Blocks* all latched together. The bricks that compose this work are also like voxels endowed with intelligence in a real physical space. The whole behaves like a living organism. Each electronic cell perceives the sounds and reacts according to the intonations of the voices, from low to high pitched, and the duration of the

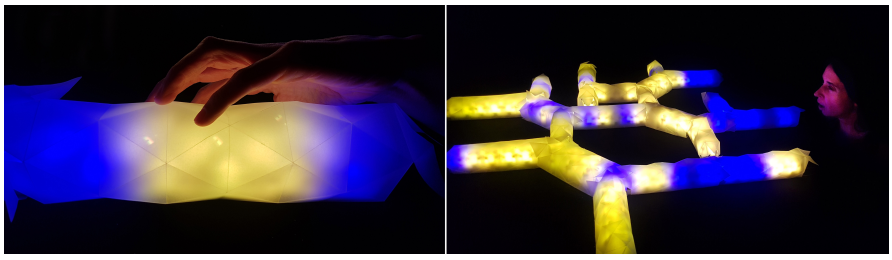

Fig. 7 Two views on *Rhizome 001*, a sculptural and interactive work that behaves like a living organism.

vocalizations. Each of the 1,824 electronic cells is reactive. The work thus deploys in its volume: 1,824 micro-controllers, 1,824 independent micro-speakers, 1,824 light sources, 1,824 $\times 6 = 10{,}944$ communication links. They emit different behavioral sound scenarios, rhythms and light colors in response to stimuli. A sound stimulus influences the robotic bases between them, between neighboring cells, like cooperative living organisms. Visible communications in the form of luminous and sonorous chases are then deployed throughout the structure. The information propagates while causing other emergences of sounds and colors under certain conditions. When two messages of the same energy of color meet within a cell, they annihilate. The phenomena of self-generation generated here is inspired by cellular automata and the algorithmic principles of the "Game of Life" imagined by the mathematician John Horton Conway.

The light and sound feedback of the work evolve and change according to the duration and intensity of the audiences' sound interactions. The sound flows produced by the electronic devices also play a role in their mode of communication. The data transmitted is therefore both digital via the connectors that link them together but also analog via the audio transmitters / receivers that make them up individually. The result is complex and unpredictable interactions linked largely to the physicality of the works, to the nature of their support, the plastic chest cover which composes them, or the membranes which surround them. The software code takes into account the sound variations and the data exchanges between the cells to generate the sound and light feedback.

6 Conclusion

In this article, we presented the *Blinky Block* system which is composed of 4 cm cubes able to sense and act, and of a software environment designed to manage large number of modules. The different versions of artworks created by Scenocosme has been displayed in many exhibitions and museums, with a nice feedback from the visitors that were pleased to interact with the artworks. Some creations have run without any problem during months which show the robustness of the whole system which have been a driving force in the hardware and software design. Furthermore, "Arborescence 003" has been used to set up a new Guinness World Record with 1,824 blocks working all together.

The *Blinky Block* robot is currently undergoing major development, and we are using it as the basis for an autonomous robot to drive new on-board peripherals. Latest works [2] have led to the creation of the *XBlock*, which includes an ESP8266 board inside the *Blinky Blocks*. It provides wireless communication capabilities (Wi-Fi) enabling messages to be broadcast between remote *XBlocks* or a master computer. Another connected development [7] involved connecting a screen to the top of the *XBlock*, which is directly controlled by it, to add display and debugging capabilities.

Acknowledgment

This work was partially supported by the ANR (ANR-21-CE33-0009), the EIPHI Graduate School (ANR-17-EURE-0002) and the Mobilitech Project. The record holders would like to thank Simon Hauser and Samuel Gomes for agreeing to act as witnesses on behalf of the Guinness World Record and MA Scène Nationale for supporting our work.

References

1. Bourgeois, J., Piranda, B., Naz, A., Boillot, N., Mabed, H., Dhoutaut, D., Knychala Tucci, T., Lakhlef, H.: Programmable matter as a cyber-physical conjugation. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2016). IEEE (Oct 2016)
2. Dhoutaut, D., Piranda, B., Bourgeois, J.: Multi-networks communications in large set of modular robots. In: Proceedings of the 1st International Conference on Smart Medical, IoT & Artificial Intelligence - ICSMAI'24. Saida, Morocco (2024)
3. Fukuda, T., Nakagawa, S.: Dynamically reconfigurable robotic system. In: Proceedings. 1988 IEEE International Conference on Robotics and Automation. pp. 1581–1586. IEEE (1988)
4. Goldstein, S.C., Mowry, T.C.: Claytronics: An Instance of Programmable Matter. In: Wild and Crazy Ideas Session of ASPLOS. Boston, MA (Oct 2004)
5. Karagozler, M.E., Campbell, J.D., Fedder, G.K., Goldstein, S.C., Weller, M.P., Yoon, B.W.: Electrostatic Latching for Inter-module Adhesion, Power Transfer, and Communication in Modular Robots. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS '07) (Oct 2007)
6. Kirby, B.T., Ashley-Rollman, M., Goldstein, S.C.: Blinky blocks: a physical ensemble programming platform. In: CHI '11 Extd Abstracts on Human Factors in Computing Systems. pp. 1111–1116. ACM, New York, NY, USA (2011)
7. Lassabe, F., Dhoutaut, D., Piranda, B., Kouchnarenko, O., Bourgeois, J.: Building a First Prototype of a Multi-scale Modular Distributed Display . In: IEEE Intl Conference on Internet of Things (iThings). pp. 276–281. IEEE Computer Society, Los Alamitos, CA, USA (Aug 2024)
8. Le Breton, D.: La saveur du monde: une anthropologie des sens. Traversées, Métailié, Paris (2006)
9. Piranda, B., Chodkiewicz, P., Holobut, P., Bordas, S., Bourgeois, J., Lengiewicz, J.: Distributed Prediction of Unsafe Reconfiguration Scenarios of Modular Robotic Programmable Matter. IEEE Transactions on Robotics **37**(6), 2226–2233 (Dec 2021)
10. Thalamy, P., Piranda, B., Naz, A., Bourgeois, J.: Behavioral Simulations of Lattice Modular Robots with VisibleSim. In: 15th International Symposium on Distributed Autonomous Robotic Systems (DARS 2021). Kyoto, Japan (2021)
11. Thalamy, P., Piranda, B., Naz, A., Bourgeois, J.: Visiblesim: A behavioral simulation framework for lattice modular robots. Robotics and Autonomous Systems **147**, 103913 (2022)
12. Yim, M., Shen, W.M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.: Modular Self-Reconfigurable Robot Systems [Grand Challenges of Robotics]. Robotics Automation Magazine, IEEE **14**(1), 43 –52 (Mar 2007). https://doi.org/10.1109/MRA.2007.339623