# Toward a 2D Modular and Self-Reconfigurable Robot for Conveying Microparts

Sebastian Möbes, Benoît Piranda, Guillaume J. Laurent, Julien Bourgeois, Cédric Clévy, Nadine Le Fort-Piat

***Abstract*— This paper describes the design, prototyping and control of a 2D modular and self-reconfigurable robot for conveying microparts. The elementary block is designed to have a package dimension under 1cm$^3$ and will include the actuators, the electronics and the micro-controller. Electro-permanent (EP) magnets are used for both the linkage and the traveling system to avoid any power consumption during the linkage. Some prototype blocks have been realized and show a well working of the motion and a sufficient holding force. The paper presents also an algorithm, common to all blocks units, allowing to reconfigure a set blocks from a spatial configuration to another one. This algorithm is implemented in a simulator software showing in real-time the reconfiguration of the robot.**

## I. Introduction

Conveyors are usually designed as monolithic entities solving one problem at a time. Furthermore, if monolithic design fits the need of fixed types of environments and/or objects, it lacks reactivity to environment changes and failures that occurs at small scales [1]. Our idea is to build a modular conveyor composed of hundreds of similar blocks that can detect objects, move and communicate with each other to form a flexible conveying path. This modular conveyor can be used to transport small objects but also to recognize and to sort the objects. A further possibility is that the conveyor will be able to automatically replace blocks that have failed with working ones which adds a self-healing characteristic. Building this conveyor is the focus of the Smart Blocks project.

In order to move very small objects as the green cylinder shown in figure 1, each block will embed arrayed-MEMS actuators on its upper face. Miniaturization of the block is therefore one of the key aspects of the project. The final system edge length is foreseen to be of 10 mm, including everything from supplying the MEMS array and connecting the blocks but also to move them. The focus of this article is the design of the linkage and motion system, which should be fast, precise and energy-saving while requiring the smallest possible space. Another goal is to achieve a high level control of the spatial configuration of the conveyor to be able to form any shapes. This article describes an original solution which consists of a linear electromagnetic motor that enables blocks to glide on each other and presents a shape transformation algorithm of the conveying path.

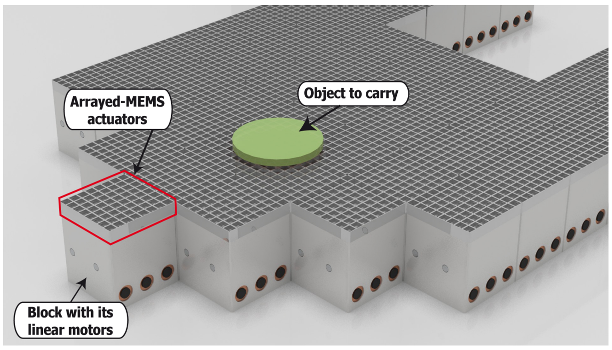


Fig. 1. Example of using reconfigurable blocks for conveying objects.

## II. Related works

Several projects have influenced our work, but the first thing to mention is the Smart Surface Project[1] which is our direct predecessor. It has been developed different hardware and software systems to produce an air cushion in order to move small, flat objects. One of these, a tilted-air-jet surface, reached a size of 9mm×9mm and gave therefore the target system dimensions for the Smart Blocks Project [2]. Additionally, there were algorithms developed in order to control multiple sensor-actuator units with their own processor in a decentralized way [3]. Other interesting ideas comes from the modular robot field. The projects like M-TRAN [4], Superbot [5], Roombots [6] and Molecube [7] have already shown motions of autonomous parts and self-assembly albeit in bigger dimensions. The miniaturization of their mechanical connection systems is complex and does not seem to be the right solution for a smaller system. Another connection system has been realized by Neubert et al. [8] using a Fields Metal solder, melting at 60°C. This principle uses electrical heating of the contact area where the solder is deposited, to melt it and to let it solidify again. The reached connection is very strong and can also be used for power or data connections. The problem is that there is no

This work was partially supported by the Smart Blocks ANR (French National Research Agency) project (ANR-2011-BS03-005).

S. Möbes, G. J. Laurent, C. Clévy and N. Le Fort-Piat are with the Automatic Control and Micro-Mechatronic Systems Department, FEMTO-ST Institute, Université de Franche-Comté, ENSMM, CNRS, Besançon, France, S. Möbes is also student at the Ilmenau University of Technology, Ilmenau, Germany. `guillaume.laurent@ens2m.fr`

B. Piranda is with Université de Franche-Comté, Laboratoire LASELDI, équipe Outils et Usage Numériques (OUN), Montbéliard, France, `benoit.piranda@univ-fcomte.fr`

J. Bourgeois is with Computer Science Departement , FEMTO-ST Institute, Université de Franche-Comté, CNRS, Montbéliard, France, `julien.bourgeois@univ-fcomte.fr`

[1]`http://www.smartsurface.cnrs.fr`

attraction force to move the modules. In terms of connection and attraction, the most inspiring work comes form Robot Pebbles [9] using small cubes (12mm×12mm), capable of forming two dimensional shapes using electro-permanent (EP) magnets. The advantages of EP magnets is that they are able to keep their polarity after a short energizing. But because it is planed as a kind of stochastic self-reconfigurable matter with motion from outside self-motion has not been studied.

In this paper, we propose to use EP magnets to design a linear motor able to move 1cm$^3$ cubes. Using this motor, the blocks can glide one each others and can be stopped in every state keeping a strong connection without any power consumption. This means that after the system has formed its optimal configuration by linear motion of the blocks in respect of each other, it can perform its conveyor function using no other resources for the linkage.

## III. Hardware

### A. Electro-permanent (EP) magnets

The basic part of the motor unit is the EP magnet. It consists of a coil with an permanent magnet core, like AlNiCo. This material, made from aluminum, nickel and cobalt, is an alloy with a specific material characteristics. It has a remanence like neodymium magnets around 1.2T, but a relatively weak coercive field strength around 50 to 100kA m$^{-1}$ in contrast to neodymium magnets with over 1000kA m$^{-1}$. A qualitative diagram of this characteristics can be seen in [9]. The result is a big magnetic force (depending on the remanence), but also the ability of a very easy magnetization and demagnetization (depending on the coercive field strength). Wrapping a coil around the AlNiCo, a magnetic field can be generated to switch the magnet polarity in a very short time. The result is a bistable system, able to have an attraction or a repulsion mode to a permanent magnet pole.

### B. Linear Motor Unit

To achieve the highest possible force, the air gap in the magnetic circuit has to be as small as possible. Therefore U-formed core designs produce a much higher attraction force but the installation space is bigger. We decided to save space and use three parallel EP magnets in front of two permanents magnets (cf. Fig. 2).

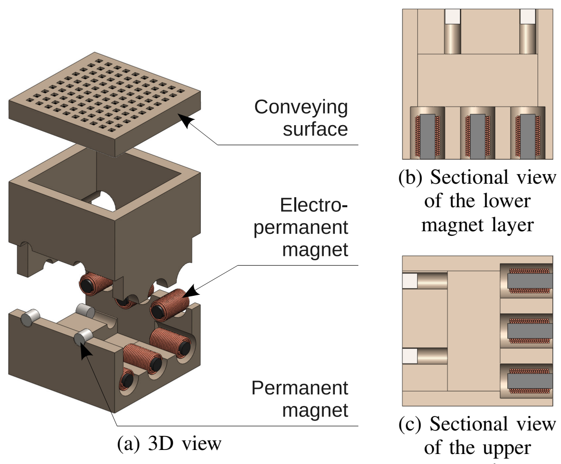
To build the EP magnets, the AlNiCo core which is 1mm in diameter and 3mm in length has been wrapped by 60 turns of 0.1mm enameled copper wire.

The resulting coils have a size of nearly 2mm in diameter. The counterpart are two cylindrical, 1mm by 1mm neodymium magnets. Every block has two active layers, on for each direction of the plan. Each layer contains three EP magnets and two permanent magnets. To be able to move the magnets in a chosen direction, the distance between the two neodymium magnets has to be 50% bigger than the distance between two EP magnets on the opposite side.

In regards to a standard linear motor, we are using just two states in every EP magnet, so we do not use an off-state as well as some intermediate states with another remanence in the AlNiCo as the chosen maximum. Most electric motors are using a sinusoidal control to reach a smooth motion, while we are using a kind of block commutation to get a bistable effect. The state transition and its effect on the movement can be seen in Fig. 3. Because every block has two permanent magnets on each of its passive sides, six state changes enable to move the three EP magnets of an active block side 10mm far.

### C. Electronics and Power supply

To change the polarity of the EP magnets, high current pulses have to be done in both directions. Because the



(a) 3D view (b) Sectional view of the lower magnet layer (c) Sectional view of the upper magnet layer

Fig. 2. Design of the block with the configuration of the EP and the neodymium magnets. The block is a 10mm cube and consists of two parts, an upper and a lower part, to make the final assembly easier.

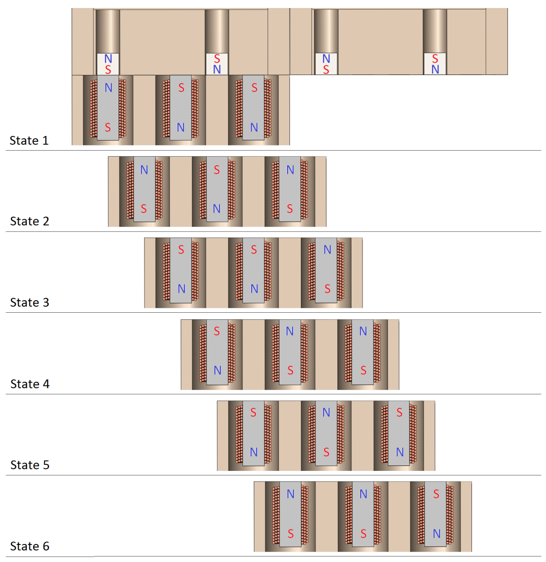


Fig. 3. State diagram of the linear motor showing the 6 different states of the EP magnets and their caused position in respect to the passive sides of two blocks. The magnetic poles are marked with S and N for South and North.

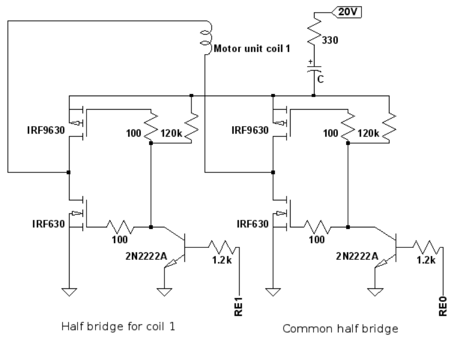


Fig. 4. Schematics of the electronic for driving one coil.

resistance of the coil is very low, a 15V difference causes more than 15A current in the coils with is sufficient to magnetize the AlNiCo core. Therefore a 100µF capacitor is used to buffer and a resistor to uncouple the system from the power supply.

To supply the EP magnets with current in both directions and to reduce the components to a minimum in order to save space, a shared H bridge structure using MOSFETs has been designed. As it can be seen in Fig. 4, every EP magnet has its own half bridge, allowing the choice of the direction of the current. Additionally there is a fourth half bridge, able to decide which chosen current direction should be powered. Every half bridge consists of two MOSFETs, a P-channel for the upper side and a N-channel for the lower side. If both transistors have their Gate on the supply voltage level, the lower side is opened and the upper side closed, connecting this end of the EP magnet to the ground. By giving a 5V signal to the 2N2222A bipolar transistor it opens and pulls down the MOSFET gates to ground, which causes an opening of the upper side and a closing of the lower side transistor, connecting this end of the EP magnet to the supply voltage.

The blocks will be powered externally via the ground surface. The housing of the other electronic parts in the block is possible. Tantalum capacitors with 100µF and 20V can be found with the dimensions 7.3mm×4.3mm×2.9mm. The eight half bridges for two motor units in each block will consist of eight Fairchild Semiconductor FDME1043CZT with a size of 1.6mm×1.6mm×0.55mm including each the necessary N- and P-channel MOSFET. Bipolar transistor and resistor can also be reduced to very small size. However, we build the first test electronic on a breadboard with standard components.

### D. Processor

To drive the bridges, we used a Microchip dsPIC30F4011. We needed four output pins to give the signal to the half bridges and one analog-to-digital converter (ADC) input to measure the capacitor voltage. In the final solution, we will need 8 I/O pins for running two motor units. The data connection between the blocks is planned but not implemented yet. Furthermore, an algorithm for sensing the block position with the EP magnets is possible, using the different inductance in different positions. Seven additional ADC ports will be necessary for this future self-sensing purpose. A corresponding IC from Microchip would be the 28 pin PIC16F723A in QFN design with a size of 4mm×4mm. The magnets and the capacitor will use most of the space of the block but the microcontroller could also be integrated in the available space.

### E. Motion control

To produce the motion described in Fig. 3, we develop a program to generate the appropriate sequence of pulses. Fig. 5 shows the timing diagram of I/O pins of the microcontroller.

For example, if the polarization of the motor unit is NSN (S stands for south and N for north). The next state must be SSN. So, the first two half bridge bipolar transistors will get a 5V signal, opening the upper side transistor. The third one will stay with no signal and an opened low side transistor. Because the fourth half bridge has got no signal and connects therefore all EP magnets on one end to ground, the current will flow from the two first upper sides over the corresponding magnets mainly to the fourth low side and to ground. Hence the first two magnets have been powered but not the last one. For this, the fourth half bridge needs to be switched, whereby only the third EP magnet will be powered. The result is that two shots has to been done successively if the switching of all magnets is necessary.

As we can see on Fig. 5, the capacitor needs some time to recharge after a shot. This time depends on the voltage supply, the resistor in front of the capacitor and the duration of the last shot. The capacitor voltage will therefore be read over a one to ten voltage divider with an ADC channel, so that the next shot can be started as soon as the sufficient voltage is reached (i.e. 15V).

## IV. Experimental results

### A. Construction

All the manufacturing has been done in our lab, from the fabrication of the block housing over the EP magnets assembly to the soldering of the components.

The block housing, seen in Fig. 2a, has been made by rapid prototyping with a fused deposition modeling machine. The final assembly has been done by placing and gluing all magnets in the lower part of the block, before gluing the upper part on it. The separate parts and the final assembly can be seen in Fig. 6.

### B. Measurement of the attraction force

Before we build the first motor unit, we measured the attraction force for different magnetizing voltages between one EP magnet and one permanent magnet. After a proper demagnetizing, we changed the polarity of all magnets using two shots. To measure the force, we used weights, hanging them carefully on a special build anchor with a hook, until the connection broke. We made each experiment five times to guaranty a good result.

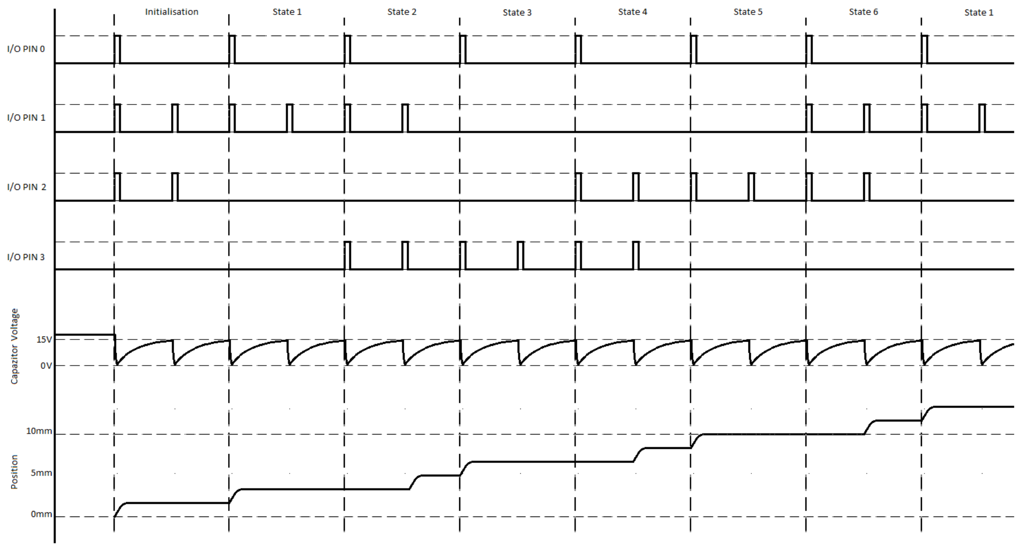


Fig. 5. Timing diagram showing the signals coming from the IC. The I/O PIN are labeled from 0 to 3. Additionally, the voltage of the capacitor and the displacement of the block are shown.

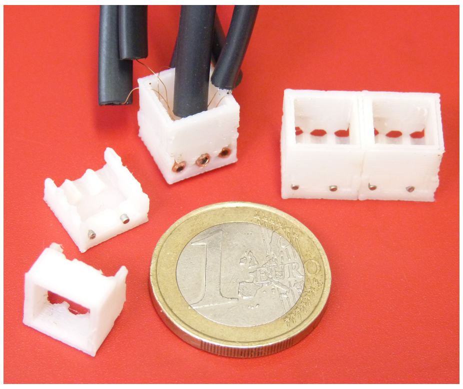


Fig. 6. Rapid-prototyped parts of the block (on the left), one assembled with EP magnets (on the middle) and two with permanent magnets (on the right).

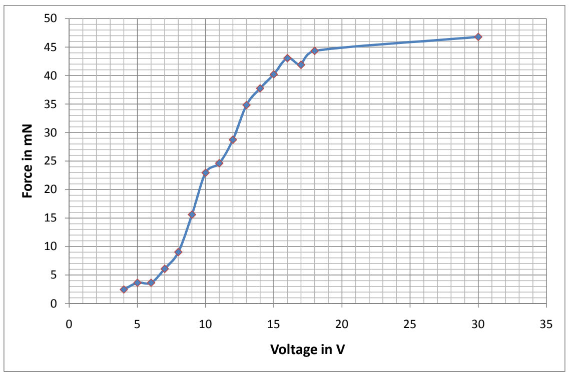


Fig. 7. Magnetic force between the three EP magnets of the active side and the two neodymium magnets of the passive side according to the magnetizing voltage (using two shots).

The result can be seen in Fig. 7, showing until a voltage of 6V that there is not enough field strength to start a proper magnetization. From 7V up to 16V is a linear region of the AlNiCo and over 17V the maximal force seems to be reached. This result shows that a voltage over 15V is not really able to increase the strength but to reduce the speed of the system. Therefore to charge to 18V instead of 15V takes 66% longer but increasing the attraction force just by 10%.

The next test was to check how a shorter or longer pulse influences the attraction force. Therefore the magnets were demagnetized as before and then magnetized with a shorter or longer 15V shot.

We found out that it makes no sense to increase the shot time over 25µs, because while the coil is powered, the voltage of the capacitor is shrinking, causing a gradually lower current in the coil. Actually after 25µs the current seems to be too low to polarize the magnet any further.

### C. Motion and energy consumption

Fig. 8 shows a image sequence of the motion of the blocks. The block is jumping from one state to another. The motion can be further appreciated in the video clip accompanying this paper. The mean speed is 14mm/s.

After a shot, the drop of voltage of the capacitor does not exceed 8V. Knowing the capacitance value (100µF), we can deduce that the magnets consume less than 10mJ for one shot. As we need twelve shots to move the block for one to another, a 10mm move consumes about 120mJ.

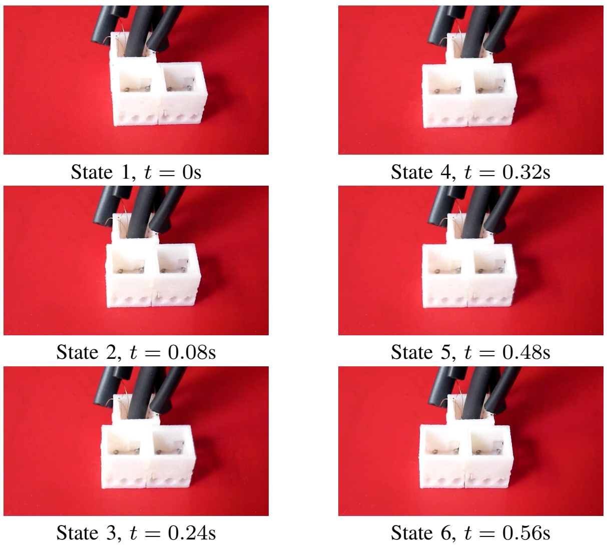


Fig. 8. Snapshots of the blocks during the motion.

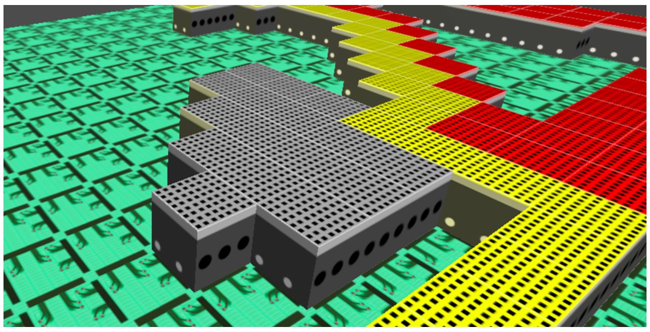


Fig. 9. Captured Image from the simulator software showing blocks with conveyor on the top. The floor is cover by tracks that guide the blocks during their displacement.

## V. Blocks reconfiguration algorithm

### A. *Operating principle of the blocks*

In this paper we present an algorithm common to all blocks units allowing to reconfigure a set of blocks from a 4-connected organization to another one. This algorithm is implemented in a simulator software showing in real-time the displacements of blocks (see Fig. 9).

The algorithm is based on exchanges of messages between neighboring blocks, where each message may contain transmitted data. In order to write this algorithm for blocks reconfiguration, blocks are considered as entities with their own memory, able to execute a set of instructions to manage this memory. Each block is connected with 1 to 4 neighbors and communicates using messages. The algorithm presented requires only data stored within each block, which can run independently in autonomous units of computation.

One of the blocks is chosen to propagate the actions of the reconfiguration algorithm to the other blocks. We call this block *Master*, it is connected to an external module that manages the reconfiguration.

The desired configuration is stored in a 4-connected map,

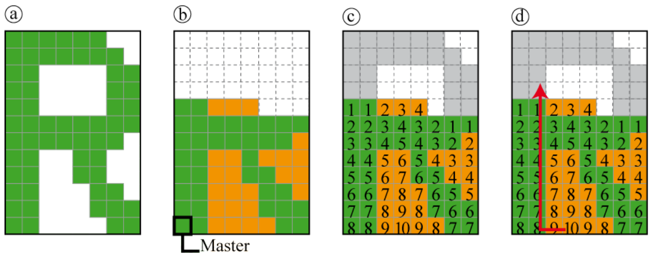


Fig. 10. Some steps of the reconfiguration algorithm : a) a goal map; b) an initial configuration of blocks; c) the distance from each block to the closest empty space; d) the path from one block to a free cell.

where the number of cells to fill corresponds exactly to the number of blocks that receive the card. Fig. 10.a shows an example of a map where the cells to fill are drawn in green.

### B. *Steps of the algorithm*

The algorithm for blocks reconfiguration is divided into four major stages:

A preliminary step, executed only once for the whole of the reconfiguration consists in transmitting the map describing the desired configuration to each block. In order to save memory, it is possible not to transmit the entire map to every block. A solution is to decompose the map into several parts and distribute these sub-maps over the sets of blocks. In this first paper, we will transmit the entire map to each block, this optimization will be the subject for future enhancements. The propagation algorithm of the map to all blocks is detailed in paragraph V-C. Fig. 10.b shows the result of transmission of a map to a set of blocks, blocks can obtain boolean value from the map for the cell they cover (green for true and orange otherwise).

After this preliminary step, we will repeat three successive stages for evolving the current configuration to the desired solution: calculating the distance between blocks and empty areas, defining the direction of movement, and the motion of the blocks.

*1) Calculating distance between blocks and empty areas:* The first phase is to search for each block the shortest distance (in following 4-connected blocks) which separates it from a free position in the card as shown. For example, the value placed over each block in Fig. 10.c is the distance from the block to the closest empty cell.

First we initialize all distances stored in the blocks to the infinity value, then we search which blocks are in contact with empty places. Since each block has a copy of the map in its own memory, it can check if in one of its sides : there is no neighbor block and the corresponding position in the map shows an empty place. In this case it receives a distance value equal to 1.

Every block must have a distance corresponding to the minimum distance from their neighbors plus 1. Then, to determine the distance value for each block, when a neighbor updates its distance, it will propagate this information to its neighbors as a message containing its distance. When a block

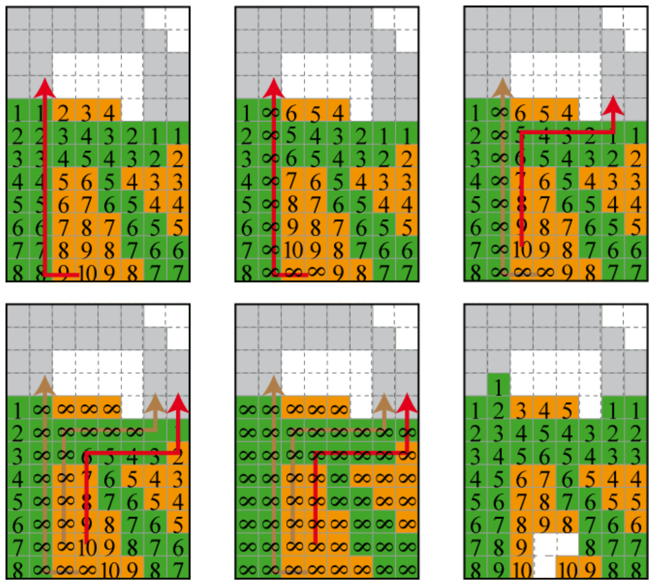


Fig. 11. Algorithm for determining the direction of simultaneous displacements of many blocks.

receives a message from a neighbor, it compares its value to the one received in the message, and corrects its value if it is greater than that received plus 1.

*2) Defining the direction of movement:* When all the distances are memorized in the blocks, we search the block $B_{max}$ that admits the greatest distance and placed on a cell of the map that must be emptied. This block is easy to identify because it is surrounded by neighbors whose distance is less than its.

From this block $B_{max}$, we have to find a path to a block close to an empty cell by following the decreasing values of distance as shown in Fig. 10.d.

To do this, we seek the neighbor of $B_{max}$ that admits the smallest distance. To choose the direction where distances are equal values we follow the order: West, North, East, and South (WNES is here a convention). The relative position of this neighbor defines the direction of movement for the block, this direction is stored in the 'direction' variable.

The two previous steps can be repeated as there is blocks that have not been used for a movement (they are marked with $\infty$ on Fig. 11). In order to avoid deadlocks, we set the distance stored in these moving blocks to the infinity value. Thus, each block can only participate to one reconfiguration at a time but many movements of blocks are possible simultaneously.

*3) The motion of the blocks:* This step of the algorithm consists in starting the physical displacement of the blocks and waiting for them to reach their destination before performing the next sequence. This movement is made by successive small motion taking into account that when two successive neighbors don't have the same direction of displacement, the first block must wait for the end of the displacement of the second one before beginning to move.

## *C. Details on the use of messages.*

We recall that the same program is charged on each block. The program behaves like a finite state machine where the evolutions of states are triggered by receiving messages.

The first step of the program is to send the map to each block. The block '*Master*' receives the first message from an external module.

The algorithm detailed below allows to start a treatment on all blocks by diffusing the order of starting, step by step with the guarantee that it will be applied only once per block. At the end of treatment, the initial block receives an acknowledgment stating that the treatment has been performed on all blocks.

For each type of information conveyed to the blocks we define a dedicated message. For example, the message *MAP_MESSAGE* used to send data on the card contains the following parameters:

- the pointer to the transmitter (which is necessarily a neighbor of the receiver),
- the size of the card,
- and the bit array indicating whether each cell must be filled or not.
- The transmission time is added to the message in order to simulate the transfer delay.

When a block receives a message type *MAP_MESSAGE*, two actions are possible:

1) if there is already a map stored in the block, then it returns an *ACQ_MAP_MESSAGE* message to the sender;
2) else, it copies the card into its local memory and sends a new message *MAP_MESSAGE* containing the card data to each of its neighbors except to the one who sent him. Moreover it memorizes the sender of the message (in a variable *sender*) and the number of neighbors to whom he sent this message (in the variable *waitedAnswers*).

When a block receives a message *ACQ_MAP_MESSAGE*, it decrements the variable *waitedAnswers* of 1. If *waitedAnswers* is null, which indicates that all his neighbors have responded, then the block sends an acknowledgment to the '*sender*' block in sending a message *ACQ_MAP_MESSAGE*.

Similar kinds of messages are defined to manage the other steps of the algorithm (calculation of distances and start of motion), each of these treatments being based on traversals of all blocks.

## *D. The real-time simulator*

We developed a simulator to visualize in real-time states of blocks during the course of the reconfiguration algorithm. This software, developed in C++, allows to observe the asynchronous executions of the code on the different blocks by managing its own global clock. It calculates the simultaneous evolution of the state of each of the blocks, taking into account the delay of transfer of messages. Thus, the emission of a message and its reception can not be achieved simultaneously.

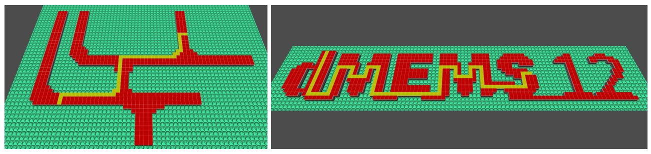


Fig. 12. Two screen captures of the final configuration of blocks for two examples : a conveyor and the text dMEMS_12. The yellow blocks are those that have just reached their final position.

| Shape Goal | Conveyor | dMEMS_12 |
|---|---|---|
| Horizontal line | 121 | 66 |
| 2 horizontal lines | 136 | 64 |
| Diagonal line | 108 | 65 |
| Sinusoidal line | 71 | 71 |
| 2 boxes | 132 | 177 |

TABLE I
NUMBER OF STEPS TO REACH THE GOAL FROM DIFFERENT INITIAL SHAPES.

After each displacement, the positions of the blocks is displayed in real time with OpenGL, many textures are used to show the states of blocks. Some screen captures are presented in Fig. 12 and some animations of reconfiguration can be visualized in a short video accompanying this paper.

### *E. Comparison of speeds of reconfigurations*

The speed of reconfiguration of the blocks strongly depends on the initial configuration of blocks relatively to the desired final map. Even more than the average distance traveled by each block, the complexity of the 4-connected path separating blocks is an important parameter of the reconfiguration complexity.

We performed two types of reconfigurations, a first one defining a card with a fairly simple pattern representing a conveyor (composed by 546 blocks), and a more complex one formed by the text : 'dMEMS_12' (706 blocks) as shown in Fig. 12).

For each of these situations, several initial configurations were tested :

- horizontal line : blocks are initially placed on a set of horizontal lines crossing the map in the direction of its length passing through its center,
- two horizontal lines : blocks are placed along two horizontal boxes crossing the map in the direction of its length, one on the top and one in the bottom,
- diagonal line: large line that crosses the map from the upper left to lower right corner,
- sinusoidal line : a curve line crosses the map,
- two boxes : one box on the left and one on the right of the map.

For each configuration, the numbers of steps to achieve the goal are shown in Table I. This value represents only the number of repetitions of the algorithm required for reconfiguration without taking into account the duration of displacement of the blocks.

One can notice that the speed of convergence of the algorithm varies significantly depending on the initial configuration. The algorithm seems to be more effective with the horizontal lines for the second case because it places blocks in a configuration close to their final distribution.

## VI. CONCLUSION

In this paper, we have first presented an efficient 2D motion and an holding system integrated in 1cm$^3$. Several blocks have been realized integrating electro-permanent magnets in order to save energy. The experiments show that a block is able to hold another one with a force of 45mN and move two blocks at the speed of 14mm/s. The blocks will be powered externally via the ground surface but the housing of the other electronic parts in the block is possible.

In the last part, we proposed an algorithm allowing to reconfigure a set of blocks according to a target map. The blocks use the same program and are able to organize themselves by exchanging asynchronous messages. This first reconfiguration algorithm could be optimized to allow more simultaneous movements of blocks.

In the future, this work will be used as a basis to realize the above-mentioned Smart Blocks project. We have therefore given suggestions, hints and solutions to problems that will be faced in later work, when the actuators will be integrated to form a modular and reconfigurable contactless conveyor.

## REFERENCES

[1] N. Chaillet and S. Régnier, Eds., *Microrobotics for Micromanipulation*. John Wiley and Sons, 2010.

[2] R. Zeggari, R. Yahiaoui, J. Malapert, and J.-F. Manceau, "Design and fabrication of a new two-dimensional pneumatic micro-conveyor," *Sensors & Actuators: A.Physical*, vol. 164, pp. 125–130, 2010.

[3] K. Boutoustous, G. J. Laurent, E. Dedu, L. Matignon, J. Bourgeois, and N. L. Fort-Piat, "Distributed control architecture for smart surfaces," in *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, 2010, pp. 2018–2024.

[4] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata, "Distributed self-reconfiguration of M-TRAN III modular robotic system," *Int. Journal of Robotics Research*, vol. 27, no. 3-4, pp. 373–?386, 2008.

[5] B. Salemi, M. Moll, and W.-M. Shen, "Superbot: A deployable, multi-functional, and modular self-reconfigurable robotic system," in *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 3636–3641.

[6] A. Spröwitz, S. Pouya, S. Bonardi, J. van den Kieboom, R. Möckel, A. Billard, P. Dillenbourg, and A. Ijspeert, "Roombots: Reconfigurable robots for adaptive furniture," *IEEE Computational Intelligence Magazine, special issue on "Evolutionary and developmental approaches to robotics"*, vol. 5, no. 3, pp. 20–32, 2010.

[7] V. Zykov, E. Mytilinaios, M. Desnoyer, and H. Lipson, "Evolved and designed self-reproducing modular robotics," *IEEE Transactions on robotics*, vol. 23, no. 2, pp. 308–319, 2007.

[8] J. Neubert, A. P. Cantwell, S. Constantin, M. Kalontarov, D. Erickson, and H. Lipson, "A robotic module for stochastic fluidic assembly of 3d self-reconfiguring structures," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 2479–2484.

[9] K. Gilpin, A. Knaian, and D. Rus, "Robot pebbles: One centimeter modules for programmable matter through self-disassembly," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 2485–2492.