Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle

BRIC: an interactive smart object for swarm robotics research

G. Legarda Herranz, P. Rochala, P. Georgiopoulou, A. Ligot, M. Salman, and M. Birattari

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2022-015

November 2022

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:

IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
UNIVERSITÉ LIBRE DE BRUXELLES
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2022-015

The information provided is the sole responsibility of the authors and does not necessarily reflect the opinion of the members of IRIDIA. The authors take full responsibility for any copyright breaches that may result from publication of this paper in the IRIDIA – Technical Report Series. IRIDIA is not responsible for any use that might be made of data appearing in this publication.

BRIC: an interactive smart object for swarm robotics research

Guillermo Legarda Herranz, Piotr Rochala, Petroula Georgiopoulou, Antoine Ligot, Muhammad Salman and Mauro Birattari

IRIDIA, Université libre de Bruxelles, Belgium.

November 2022

1 Introduction

This document introduces the Block for Robot Interactions in Collective tasks (BRIC), a smart agent for usage in swarm robotics research and experimentation. The purpose of the BRIC is to allow robot swarms to interact with a responsive environment. In particular, the BRIC reacts to forces and changes in floor colour, and can display various colour patterns. While the mechanical design choices for the BRIC were made for its usage with swarms of e-puck robots, its design should allow any swarm of sufficiently large mobile robots to interact with it.

2 Hardware description

This section presents the hardware of the BRIC, from the electronics to the mechanical parts that compose it. The designs for the printed circuit board (PCB) and the mechanical parts described can be found on GitHub [1].

2.1 Electronics

A block diagram of the electronic components of the BRIC is shown in Figure 1. The brain of the BRIC is an Espressif ESP32-WROOM-32D microcontroller (hereinafter referred to simply as ESP32) [2]. The ESP32 has an ESP32-D0WD core and Bluetooth, Bluetooth LE and Wi-Fi modules. While Bluetooth is not employed in the current implementation of the BRIC, the Wi-Fi module provides bidirectional communication with an external client.

The ESP32 also comes with a wide range of interfaces for peripherals, including, most importantly, I2C. The I2C interface is used to connect two sensors and one actuator to the microcontroller. The first sensor is a TCS34725 RGB sensor with and infrared filter and a white LED [3]. It serves as a ground sensor for the BRIC, allowing it to detect colours or greyscale value variations. The second sensor is an ADXL345 three-axis digital accelerometer [4], which allows the BRIC to sense its own movement when it is pushed by a robot. The only actuator of the BRIC is an LP5024 24-channel I2C constant-current RGB LED driver [5]. Every set of three channels corresponds to a separate LED, thus resulting in eight LEDs that can be individually controlled. The driver is assembled on a custom-made PCB, which also integrates the microcontroller and a connector for a Panasonic NCA103450 Lithium-ion battery [6], as shown in Figure 2.

2.2 Mechanical design

The BRIC is shaped as a hollow, 11 cm-tall octagonal prism with 5 cm-wide sides, thus resulting in a 12 cm span (see Figure 3). It is divided into three horizontal sections, each serving a particular purpose. The three sections are interconnected with magnets for ease of assembly and disassembly.

[1] https://github.com/demiurge-project/smartobject-hardware
[2] https://www.espressif.com/en/products/modules
[3] https://www.adafruit.com/product/1334
[4] https://www.analog.com/en/products/adxl345.html#product-overview
[5] https://www.ti.com/product/LP5024
[6] https://industrial.panasonic.com/ww/products/pt/lithium-ion/models/NCA103450

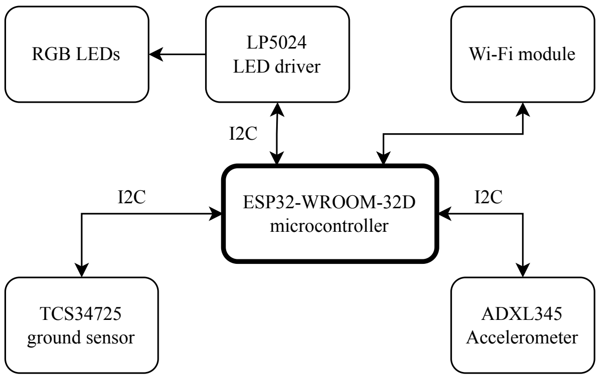
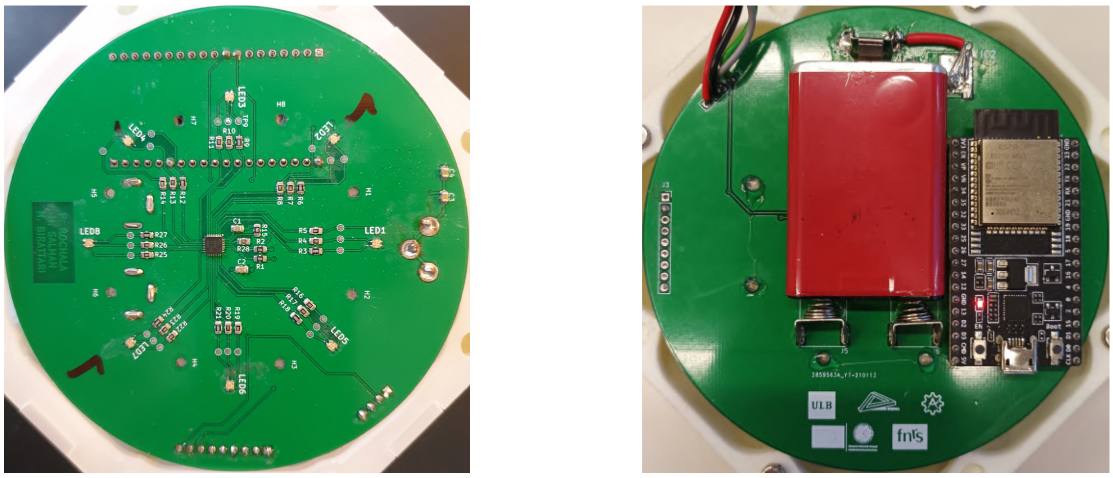


Figure 1: Block diagram of the electronic components of the BRIC.



(a) Top view. (b) Bottom view.

Figure 2: Custom-made PCB that integrates the microcontroller, the LEDs and the battery.

Both the top and bottom sections are 3.5 cm tall, with 4 mm-thick walls and 3 mm-thick ceiling and floor, respectively. The top section holds the PCB. Its interior is divided into eight separate wedges, one for each LED. To avoid saturating the camera of the e-puck, the top part is made of 3D-printed white nylon PA12, which diffuses the light from the LEDs. The bottom section holds the ground sensor and the accelerometer. To simplify the manufacturing process, it is also made of 3D-printed white nylon PA12. The ground sensor is mounted above a 1.5 cm-wide, circular hole in the centre of the bottom section. An additional conical structure made of 3D-printed PLA is glued to the floor to provide a resting surface for the accelerometer.

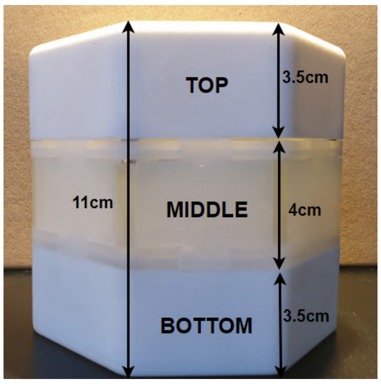
The middle section is made of laser-cut, 3 mm-wide, translucent, white acrylic parts, which are held together with hot glue. Its purpose is to raise the top section to an appropriate height. For the e-puck robot, the middle section is 4 cm tall (see Figure 4); however, the design of the individual parts makes it straightforward to increase or reduce its size for use with other platforms.

Each fully assembled BRIC, including the electronic components and the battery, weighs a total of 430 g. This ensures that they cannot me moved by a single robot and thus enforces cooperation within the swarm.

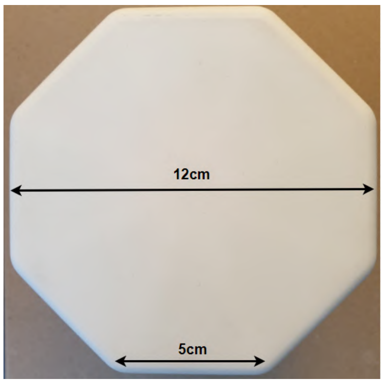
3 Software description

Two software components are currently available for the BRIC: a driver that provides an interface for the user to interact with the hardware and an ARGoS3 plug-in to use the BRIC within the ARGoS3 simulator [1]. Both are available to download from GitHub [7,8]. This section provides an overview of each component as well as the corresponding installation instructions. It is assumed that the user is working on a Linux machine; in particular, all software components were developed and tested on an Ubuntu 20.04 machine.

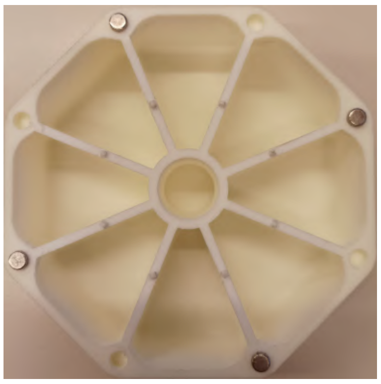
[7] https://github.com/demiurge-project/smartobject-driver.git
[8] https://github.com/demiurge-project/argos3-smartobject.git



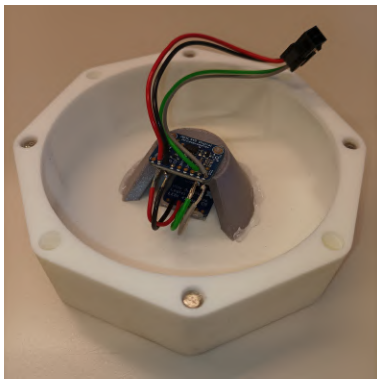
(a) Front view.



(b) Top view.



(c) Wedges of the top section.



(d) Support structure for the accelerometer.

Figure 3: Perspectives of the mechanical design of the BRIC.

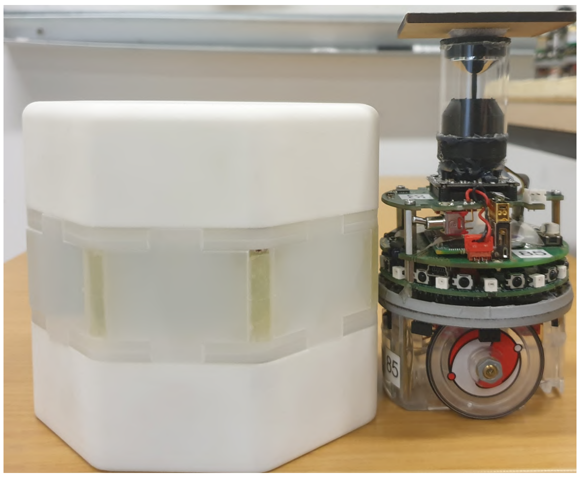


Figure 4: A BRIC next to an e-puck robot. The dimensions of the BRIC were determined for compatibility with the e-puck, but the extendable middle section makes it potentially suitable for other platforms.

3.1 Driver

The ESP32 family of chips uses FreeRTOS as an operating system with the lwIP network stack. For development purposes, Espressif provides the Espressif IoT Development Framework (ESP-IDF) to configure, build and flash firmware onto the corresponding hardware. The ESP-IDF API provides a set of C tools for writing firmware, including functions, template files and wrappers to external libraries. Consequently, the BRIC driver is written in C.

Upon booting, it sets up a local area network through the Wi-Fi module of the ESP32 microcontroller and initialises the I2C driver, the LEDs, the accelerometer and the ground sensor. A user can then set up a TCP connection, with the BRIC as a server, to interact with the device. The driver processes incoming messages as ASCII characters, where each set of characters corresponds to a configuration setting, a request for calibration of the ground sensor, an LED pattern or a particular instance of control software, for which three examples are provided (see Section 4).

To build and flash the firmware onto the ESP32 microcontroller, first set up the ESP-IDF environment [9]. Then, plug the microcontroller into a USB port and run the following commands:

```
$ get_idf
$ idf.py -p PORT flash
```

The `get_idf` command sets up all required environment variables. The `idf.py` script builds and flashes the current project, where `PORT` is the serial port to which the microcontroller is connected (e.g. `/dev/ttyUSB0`).

The driver prints various log messages through the serial port of the microcontroller, including boot data, network status and processed input. This information can be useful for debugging and development purposes. To monitor this output, run

```
$ idf.py -p PORT monitor
```

To quit the serial monitor, hit `Ctrl + ]`.

3.2 ARGoS3

The ARGoS3 simulator allows users to install plug-ins to customise their simulations. The BRIC plug-in introduces the BRIC as a robot equipped with an LED actuator, a ground sensor and an accelerometer.

To install the BRIC plug-in, first make sure ARGoS3 is already installed. Then, clone the directory from GitHub and run

```
$ cd argos-smartobject
$ mkdir build
$ cd build
$ cmake ../src
$ make
$ sudo make install
```

4 Basic use

This section provides instructions for calibration of the ground sensor, testing the LEDs and launching the provided examples of control software. For all procedures, first connect to the “esp32” network of the BRIC. The `client.py` script provided is then used at each of the following steps. It takes two arguments: an action type (`-a`), which may take any value between 1 and 5; and a message value (`-m`), which takes an eight-digit integer. The script then sets up the TCP connection with the BRIC using a fixed IP and port number and delivers the messages before closing the connection.

The meaning of the possible values of the arguments is explained in the following sections. Additionally, running the script with the `-h` or `--help` returns a summary of the same information.

4.1 Calibration

Both the ground sensor and the accelerometer should be calibrated before attempting to run any control software that uses their input.

The current implementation of the BRIC uses the ground sensor to detect changes in greyscale value. The calibration procedure thus matches white and black floor patterns to their corresponding ground sensor values.

[9] https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/linux-macos-setup.html

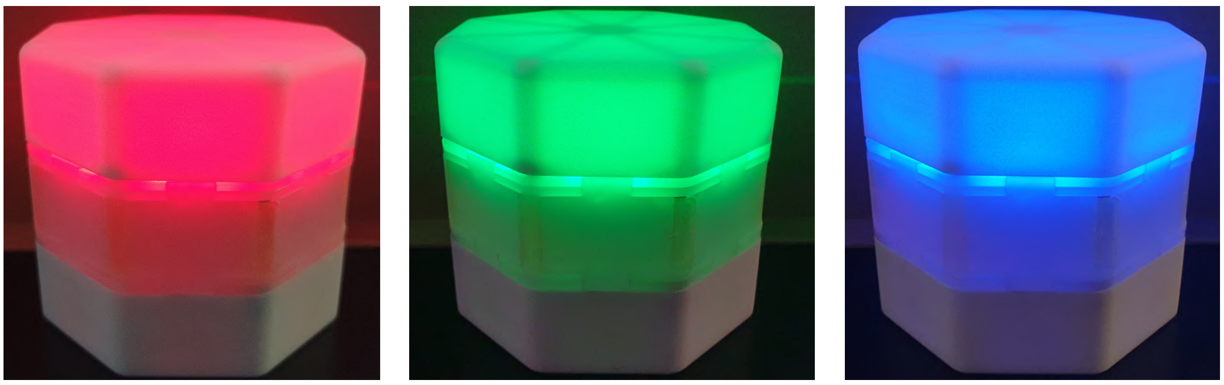


Figure 5: BRIC with all LEDs turned on with full brightness in red, green and blue.

This is done by sampling the RGB values for 10 CPU cycles (one cycle = 100 ms). The mean values are then stored in the non-volatile memory of the ESP32 and are therefore not erased when the power is cut.

For the white floor calibration, first place the BRIC on a white surface. Then, run

```
$ python3 client.py -a 5 -m 5
```

The action `5` is used for calibration and to launch the control software examples (see Section 4.3). The message `5` specifies that the calibration is performed for the white colour. Similarly, to calibrate for the black colour, place the BRIC on a black surface and run

```
$ python3 client.py -a 5 -m 6
```

The accelerometer must be calibrated whenever a software instance that uses it is run (see, for example, `mission1.c`). For this purpose, the BRIC must be placed on a flat surface and remain stationary for 50 CPU cycles. The mean accelerometer readings along the x- and y-axes are recorded and matched with a stationary state.

4.2 Test colour displays

The user can verify the correct functioning of the LEDs by lighting up three, four or all eight of them. The current version of the driver only supports pure red, green and blue values. To test the LEDs, run, for example,

```
$ python3 client.py -a 4 -m 11255
```

which will turn on all the LEDs in red at full brightness. The action `4` is used exclusively to test the LEDs. The message can then be up to eight integers long; however, only up to five are currently used. The first integer specifies the colour (`1` for red, `2` for green and `3` for blue, see Figure 5). The second integer specifies the number of LEDs that will light up (`1` for all eight, `2` for three and `3` for four). The remaining three integers determine the brightness of the LEDs and thus range between `0` (off) and `255` (full brightness).

4.3 Launch control software

Three examples are provided to showcase the usability of the BRICs in swarm robotics research. In the first example, `mission1`, the BRIC begins showing a given colour in all eight of its LEDs. It will then change its colour in a red-green-blue cycle every time it senses a positive acceleration. The goal of a robot swarm would then be to switch the colour of all the BRICs in the environment to blue. To run `mission1`, place the BRIC on a flat surface and run, for example

```
$ python3 client.py -a 5 -m 11
```

The BRIC will then automatically calibrate the accelerometer (see Section 4.1) before showing the colour red in all of its LEDs, as indicated by the second integer of the message.

For the second example, `mission2`, three of the LEDs of the BRIC show the colour blue, so long as they do not stand over a white floor, in which case they will switch to red. This particular example could be used to test how efficiently a swarm can clear a given area of obstacles that can only be detected from a certain angle. To launch `mission2`, run

```
$ python3 client.py -a 5 -m 2
```

In the third example, the BRIC simply shows the colour blue in all of its LEDs. That way, a robot could perceive it and recognise it as a movable obstacle. A swarm could then, for example, clear a path in an otherwise blocked environment. Much like for `mission2`, `mission3` is launched by running

```
$ python3 client.py -a 5 -m 3
```

5 Conclusions

The BRIC, as presented in this document, is a versatile platform to develop and test collective swarm behaviours in dynamic environments, both in real scenarios and simulated ones through the ARGoS3 plug-in. The current version of the driver provides basic functionalities for calibration of the sensors and testing the actuators, as well as examples demonstrating the potential uses of the BRIC. Future versions could include more flexible colour settings and template behaviours to construct more complex instances of control software.

Acknowledgements

The project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (DEMIURGE Project, grant agreement No 681872) and from Belgium's Wallonia-Brussels Federation through the ARC Advanced project GbO-Guaranteed by Optimization. MB acknowledges support from the Belgian *Fonds de la Recherche Scientifique* – FNRS, of which he is a Research Director.

References

[1] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni A. Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella, and Marco Dorigo. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295, 2012.