Documentation revision: 1.0
Software version: 0.1
Date: 03/03/2022

NIC²

Neural Interface for Control of Computers

Neural Interface for Control of Computers, or NIC2, is an implementation of the Texas Instruments ADS1299 and the MyoWare SEN-13723 in Python, in order to use detected brainwaves and muscle electric values to command computers, robotics and other machines.

Table of Contents

System Architecture

Hardware

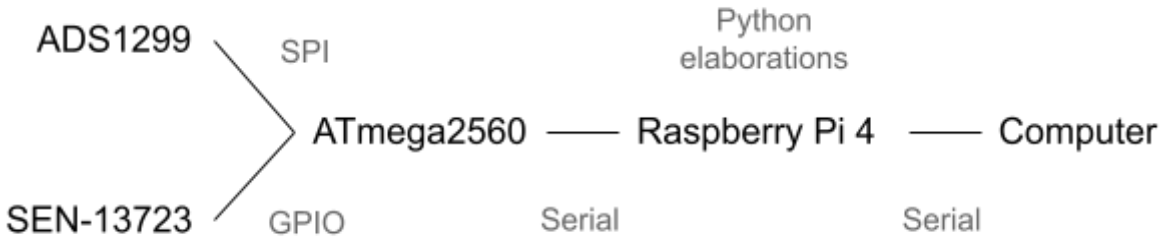


Figure 1. Hardware architecture

The system is composed of an ADS1299 (mounted on an ADS1299EEG-FE) that communicates the detected values via SPI communication to an ATmega2560 (mounted on an Arduino Mega board), that interprets the data sent by the ADS1299. The ATmega2560 also receives values from the SEN-13723 sensor muscle sensor. The ATmega2560 sends the interpreted data via USB serial communication to a Raspberry Pi 4, that performs all the necessary calculations, using a Python program, to recognize the type of brainwaves. Finally, the Raspberry can be programmed to communicate to an external computer what it has recognized, and can do this by serial or other types of communication protocol.

ADS1299

- 8 INAs and 8 24-bit high-resolution ADCs
- Data Rate: 250 SPS to 16 kSPS
- Operating tension: 4.75 - 5.25V (analog) or 1.8 - 3.6V (digital)
- Operating temperature range: –40°C to +85°C
- Input-Referred Noise: 1 μVPP (70-Hz BW)
- CMRR: –110 dB
- Power consumption: 5mW/channel
- SPI serial interface
- Integrated oscillator

The ADS1299 devices are a family of four-, six-, and eight-channel, lownoise, 24-bit, simultaneous-sampling delta-sigma (ΔΣ) analog-to-digital converters (ADCs) with a built-in programmable gain amplifier (PGA), internal reference, and an onboard oscillator. The ADS1299 incorporates all commonly-required features for extracranial electroencephalogram (EEG) and electrocardiography (ECG) applications.
Full documentation at https://tinyurl.com/ads1299.

ADS1299EEG-FE

The ADS1299EEG-FE is a development board for the ADS1299 24-bit eight-channel sampling. It yields directly the ADS1299 raw and unfiltered data via SPI communication.
Full documentation at https://tinyurl.com/ads1299eeg-fe.

SEN-13723

- Wearable Design

- Single Supply
  - +2.9V to +5.7V
  - Polarity reversal protection
- Two Output Modes
  - EMG Envelope
  - Raw EMG
- Adjustable Gain

SEN-13723 is a sensor to measure muscle activation via electric potential (practice referred to as electromyography (EMG))

ATmega2560

- Flash Memory: 256 KB
- EEPROM: 4 KB
- SRAM: 8 KB
- Maximum frequency: 16 MHz
- Operating tension: 4.5 - 5.5 V
- I/O Lines: 86
- PWM lines: 12 (16 bit)
- ADC channels: 16 (10 bit)
- Serial USARTs lines: 4
- SPI Serial Interface
- Operating temperature range: -40° C to 85° C
- 2 8-bit and 4 16-bit Timer/Counter with Separate Prescaler, Compare and Capture
- Real Time Counter with Separate Oscillator
- Byte Oriented 2-wire Serial Interface
- Analog Comparator

The ATmega2560 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega2560 achieves throughputs approaching 1 MIPS per MHz.
Full documentation at https://tinyurl.com/atmega2560.

Arduino Mega

The Arduino Mega is a board based on the ATmega2560. It has 54 digital I/O pins, 16 analog inputs, 4 UARTs, a 16 MHz oscillator, a USB connection, a power jack, an ICSP header and a reset button. It receives the raw values via SPI communication from the ADS1299EEG-FE and sends the interpreted data via serial communication.
Full documentation can be found at https://tinyurl.com/arduinomega2560doc.

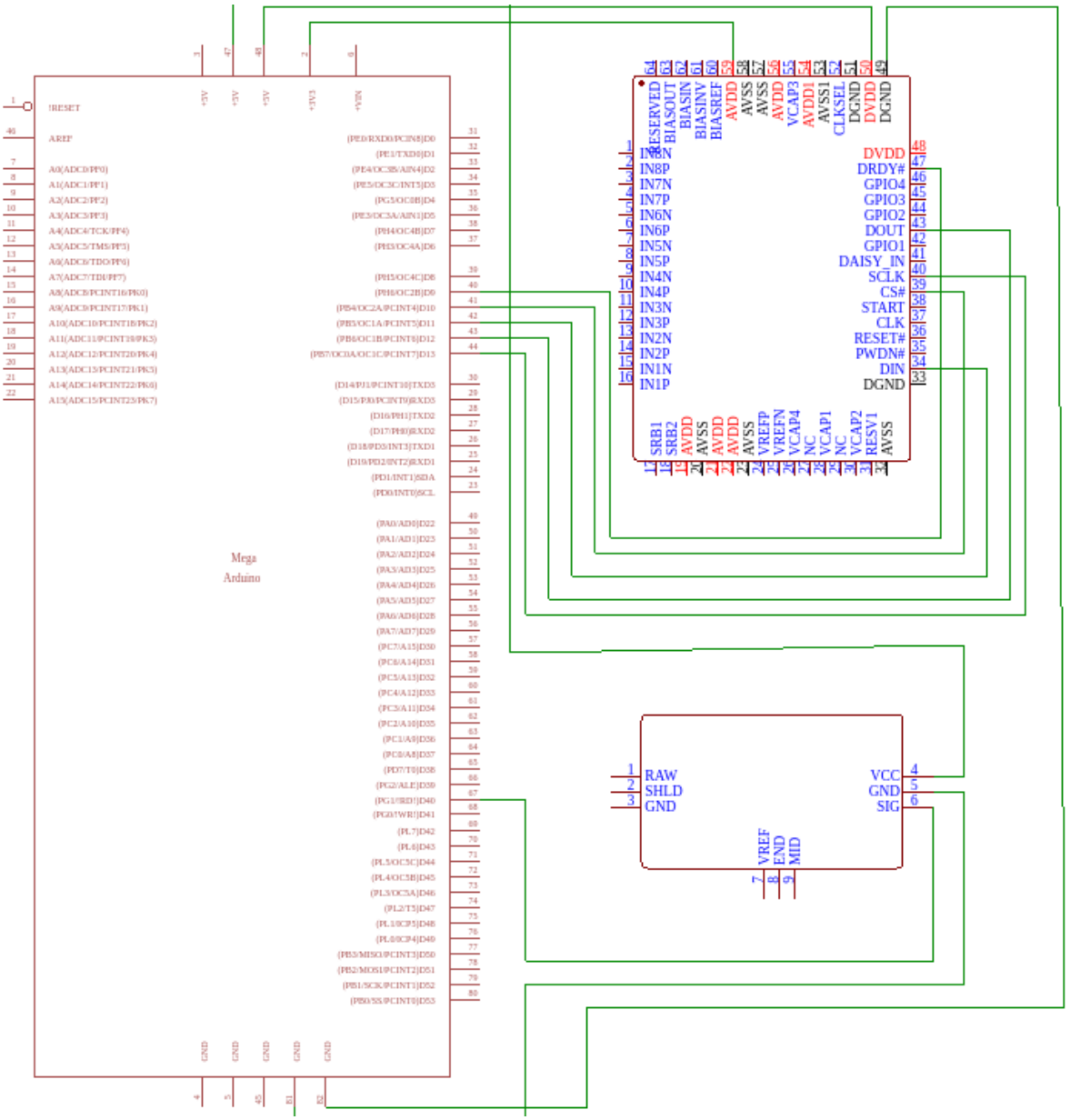
Raspberry Pi 4 (Model B)

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB LPDDR4-3200 SDRAM
- Operating tensions: 5V DC via USB-C or GPIO, minimum of 3A
- Operating temperature: 0 – 50 degrees C ambient
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet (PoE enabled)
- 2 USB 3.0 ports; 2 USB 2.0 ports; 2 micro-HDMI ports (4k60 supported).
- Raspberry Pi standard 40 pin GPIO header

- 2-lane MIPI DSI display port; 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port

Raspberry Pi 4 Model B is a single board computer, based on an ARM64 CPU, 2GB of RAM and the Linux kernel. It offers high processor speed, multimedia performance, memory, and connectivity.

Schematics



Software

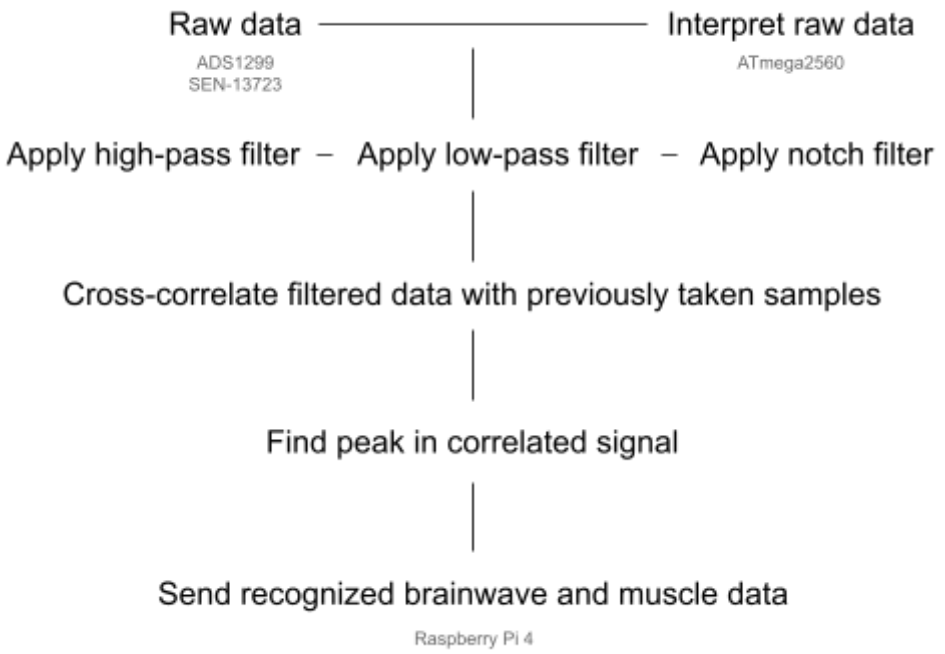


Figure 2. Software architecture

The raw data sent by the ADS1299 are interpreted by the ATmega2560 by reading them using an Arduino ADS1299 library and grouping them in batches of 8 bytes of data, that are then sent via serial communication to the Raspberry Pi 4. The ATmega2560 also detects the data sent by the SEN-13723, as raw numbers, since they are already interpreted by the module. The Raspberry then filters the brainwaves signal to remove noises, compares it with previously taken samples and finds an eventual peak in the correlated signal. If a valid peak is found, the Raspberry will send via serial or any other type of communication the detected movement.

Interpreting raw data

The interpretation of the raw data happens by detecting when the DRDY pin of the ADS1299 is LOW. Then, all the values sent in this period of time are treated as signed floats, grouped in batches of 8 bytes and sent via USB serial communication.
Plotting the raw data will give something like this:

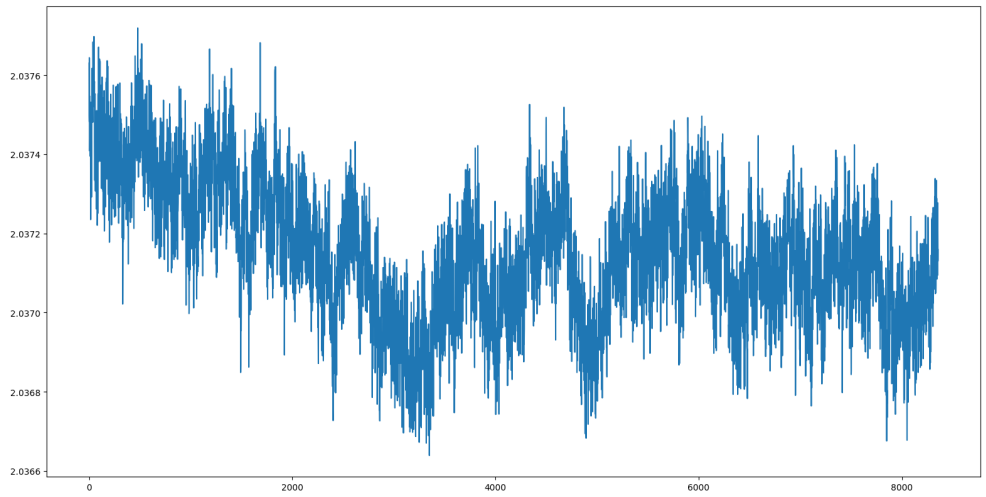


Figure 3. Plot of raw data

All of the data obtaining process and the values interpretation process is done thanks to an Arduino library for the ADS1299. More information on the library is available here https://github.com/conorrussomanno/ADS1299.

Processing interpreted data

The data processing is fully done in Python, using the Python libraries Numpy and Scipy (and Matplotlib for data visualization).
The first thing that is done, after the exclusion of garbage values, is getting enough values to apply the necessary filters, so the program will acquire 2000 samples (that are equivalent to 8 seconds, with 250 samples per second). Then, all the necessary filters are applied: firstly, a high-pass filter with a frequency of 0.35 Hz, then a low-pass filter with a frequency of 15 Hz, and finally a notch filter to the 50 Hz frequency.
Plotting the filtered data will give something like this:

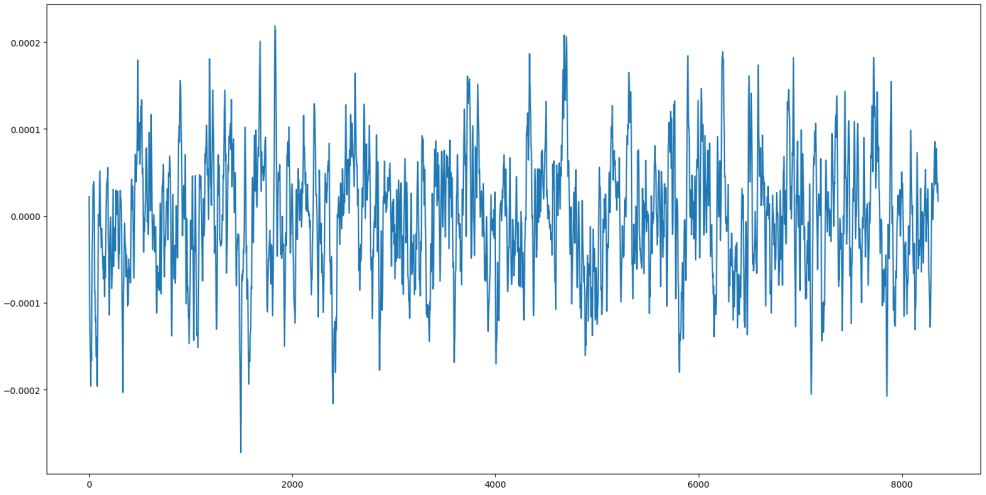


Figure 4. Plot of filtered data

Now, cross-correlation with fast Fourirer transformations is applied to the filtered values, with documented filtered samples taken previously. This will return another signal that presents peaks where the original signal is more similar to the sample, a cross-correlated signal.

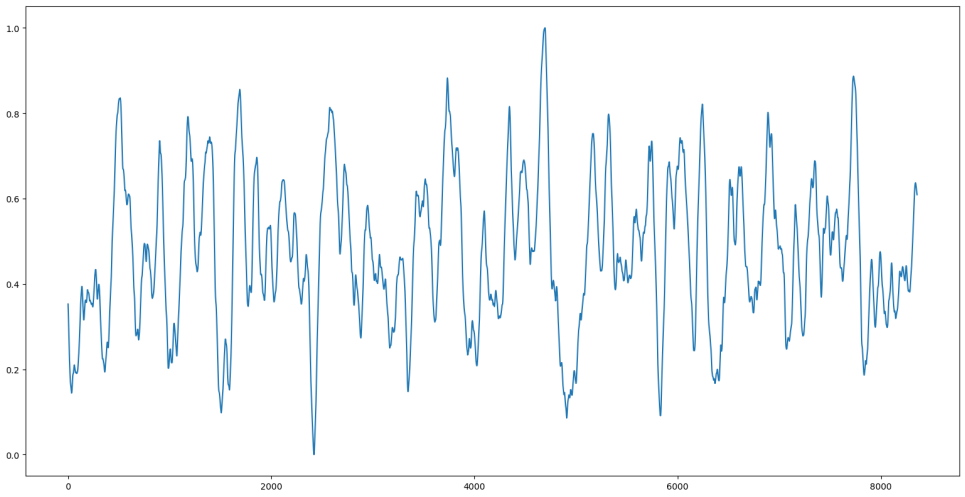


Figure 5. Cross-correlated and normalized signal

The correlated signal is then normalized, and from this signal are searched all the peaks that have a prominence of at least 0.65 and a maximum width of 140 samples (values found after a lot of trial and error).
If a peak with these characteristics is found, then the sample is marked as similar, and returned in an array with all the similar samples.
All of the code that does the computations is contained in the brainlib Python module.

Brainlib

The brainlib Python module contains all the necessary functions to do the computation to recognize the brainwaves signals.
The code is fully open source and fully documented, in this GitHub repository: https://github.com/GianlucaTarantino/nic2.

filter

Filters the signal passed as parameter in data and returns the filtered signal.
Applied filters are high pass, low pass and notch filters.

Arguments:
data -- the input signal to filter. Must be a 1 dimensional array of values. Also numpy array are accepted
sample_rate -- rate in Hz of the signal (default 250.0)
cutoff_low -- the cutoff of the low pass filter. The smaller, the more aggressive is the filter (default 5.0)
cutoff_high -- the cutoff of the high pass filter. The smaller, the more aggressive is the filter (default 0.4)
frequency_notch -- the frequency in Hz to cut from the signal with the notch filter (default frequency_notch 50.0)

Returns:
1 dimensional numpy array with the filtered values of the signal

recognize
Function used to cross-correlate two signals and to find the peak characteristics of the correlated signal. It takes multiple samples, and returns a list of indexes of recognized signals.
It recognizes a signal based on the prominence of the peak (that has to be a minimum of 0.65) and the width of the peak (that has to be a maximum of 140).

Arguments:
data -- the main signal, in which to find the signal contained in a sample. Must be 1 dimensional array of numbers
samples -- the array with all the samples, of which will be returned the index of the most similar sample
Returns:
list with indexes of recognized peaks

System features

NIC2 is a Brain-Computer Interface that can detect and recognize brainwaves and send commands based on the detected signal.

- Driven by Raspberry Pi 4
  - Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
  - 2GB LPDDR4-3200 SDRAM
- Data acquisition by TI ADS1299
  - 250 SPS to 16 kSPS
  - Operating temperature range: –40°C to +85°C
  - Input-Referred Noise: 1 μVPP (70-Hz BW)
  - CMRR: –110 dB
  - Power consumption: 5mW/channel
- 250 Samples per second
- Operating tension: 5-12 V
- Operating current: 3A
- Multiple outputs
  - 2.4-5.0 GHz Wi-Fi
  - Gigabit Ethernet
  - Bluetooth (5.0 and Low Energy)
  - USB Serial
  - PIN Serial
  - I2C
- Easy to use and configure

System use

To use the system, a few steps are necessary:

Setup the ATmega2560 and the ADS1299

Once wired the ATmega2560 and the ADS1299 as required in the aforementioned Arduino library, and once loaded the Arduino sketch that includes the library, it will simply be needed to connect the serial port of the ATmega2560 to the Raspberry Pi 4 serial port, with wires or a USB cable.

Raspberry Pi 4 program configuration

Once connected to the serial port, you will need to download the Python program to recognize the received data. It can be done by simply cloning the repository from Github:

```
git clone https://github.com/GianlucaTarantino/nic2.git
```

Then, it will be necessary to install all the Python modules necessary for the correct working of the program:

```
cd nic2
```

```
pip install -r requirements.txt
```

Once installed, the program can be launched by running

```
python src/main.py
```

To do all of this, you have to install on the Raspberry a Linux-based operating system. Other operating systems aren't supported.

Troubleshooting

- It may be needed to adjust the serial port or the baud rate on the Raspberry Pi 4 program to connect properly to the ATmega2560. Detect the right port and substitute it.
- It is possible to change samples to use, for a more precise detection. They are placed in "data/interim_samples/" directory as csv files.

Using the band

Once you start detecting brainwaves, you will have to put on the band with the electrodes. It has to be placed with the middle electrode at the center of the forehead, and the red electrode at the right of the forehead.

Test

The tests of the project were conducted using a Dobot Magician, a robotic arm with multiple joints and rotation axis.

Dobot Magician

- Number of Axis: 4
- Payload: 500g

- Max. Reach: 320mm
- Position Repeatability (Control): 0.2 mm
- Communication: USB / WIFI / Bluetooth
- Power Supply: 100 V - 240 V , 50/60 HZ
- Power In: 12 V / 6.5A DC
- Consumption: 60W Max
- Working Temperature: -10°C - 60°C

Dobot Magician is a multifunctional desktop robotic arm. Installed with different end-tools, Dobot Magician can have multiple functions such as 3D printing, laser engraving, writing and drawing.
More information is available at https://www.dobot.cc/dobot-magician/specification.html.

Test procedure

The system was tested by connecting via serial USB the Raspberry Pi 4 to the Dobot Magician. While the right arm was raised, the robotic arm was commanded to rotate right, and while the left arm was raised, the robotic arm was commanded to rotate left.
It was done by using the Pydobot library. This demonstrates the high scalability and programmability of the project. More information about the Pydobot library is available at https://github.com/luismesas/pydobot.

Code

All the code is available in the “src/main.py” file and in the project repository on Github, at https://github.com/GianlucaTarantino/nic2.

EV3

- Linux based operating system
- Controller: 300 MHz ARM9
- Flash Memory: 16 MB
- RAM: 64 MB
- Screen Resolution: 178x128 / Black & White
- USB 2.0 Communication to Host PC up to 480 Mbit/sec
- USB 1.1 Host Communication up to 12 Mbit/sec
- Micro SD Card

Lego Mindstorms EV3 programmable brick is the central processing unit within the new Lego Mindstorms platform. The programmable brick consists of various electronics to enable its functionalities.

Test procedure

The system was tested by connecting via serial Bluetooth the Raspberry Pi 4 to the EV3 brick, which moves 4 motors connected to wheels, in order to create a car. The car was moved as follows:

- Move right or left arm to turn right or left;
- Close left hand to move. Open the hand to stop moving;
- Speed is determined by the strength of the right hand closing.

This car was then used to navigate in a labyrinth, simulating also the rehabilitation of a person with motor problems.
The data needed to be converted before sending them to the EV3 brick, with a standard defined by the brick. This was done by using the code found here: http://www.geekdroppings.com/2018/01/21/raspberry-pi-and-the-lego-ev3-connected-by-bluetooth/.

Code

All the code is available in the “src/main.py” file and in the project repository on Github, at https://github.com/GianlucaTarantino/nic2.