Contents

[Introduction to GNU-Octave 2](#_Toc191305260)

[1.1 Technical details. 2](#_Toc191305261)

[1.2 The Octave language 2](#_Toc191305262)

[1. Implementation 2](#_Toc191305263)

[2.1 Loading the sketch into the Arduino 2](#_Toc191305264)

[2.2 Connecting the Arduino to GNU Octave 2](#_Toc191305265)

[2.3 Reading data in GNU Octave 2](#_Toc191305266)

[2.4 Processing data 2](#_Toc191305267)

[2. Code 3](#_Toc191305268)

# Introduction to GNU-Octave

Octave or GNU Octave is a program and programming language for performing numerical calculations. As the name suggests, Octave is part of the GNU project. It is considered the free equivalent of MATLAB. Among several features they share, it can be noted that both offer an interpreter, allowing you to execute commands in interactive mode. Note that Octave is not a computational algebra system, as Maxima is, but is oriented to numerical analysis.

The project was created around 1988, but with a different purpose: to be used in a course on chemical reactor design. Later, in 1992, it was decided to extend it, and its development began by John W. Eaton.1 The first alpha version was released on January 4, 1993. A year later, on February 17, 1994, version 1.0 appeared.

The name comes from Octave Levenspiel, a professor of one of the authors and known for his good approximations, by means of elementary calculations, to numerical problems in chemical engineering.

## 1.1 Technical details.

* Octave is written in C++ using the STL library.
* It has an interpreter of its own language (with syntax almost identical to Matlab), and allows interactive or batch execution.
* Its language can be extended with functions and procedures, by means of dynamic modules.
* It uses other GNU programs to offer the user the possibility of creating graphics to later print or save them (Grace).
* Within the language it also behaves like a shell. This allows you to list directory contents, for example.
* In addition to running on Unix platforms, it also runs on Windows.
* You can upload files with Matlab functions (recognizable by the .m extension).

## 1.2 The Octave language

* The syntax is almost identical to that used in MATLAB.
* It is an interpreted language.
* It does not allow arguments to be passed by reference. They are always passed for value.
* It does not allow pointers.
* Scripts can be generated.
* It supports much of the functions of the standard C library.
* It can be extended to support UNIX system calls.
* The language is designed to work with matrices, and provides a lot of functionality to work with them.
* Supports structures similar to C structs.
* It has an integrated development environment and others have been developed to teach programming, such as ToolboX.

# Implementation

## 2.1 Loading the sketch into the Arduino

* First, make sure that the Arduino is sending data through the serial port. For example, for this example in which more than one variable is owned, the data enters through Arduino Pin 2 and is sent through the serial port.
* Make sure your Arduino has the correct program loaded:

*display\_flowrate\_v4\_GR\_301\_CALIB\_OK.ino*

## 2.2 Connecting the Arduino to GNU Octave

* Connect the Arduino to your computer using a USB cable
* Identifies the serial port it is connected to on the Arduino. On Windows, it would be something like COM3, COM4, COM5, or COM7. On linux or macOS it should be /dev/ttyUSB0 or /dev/ttyACM0

## 2.3 Reading data in GNU Octave

* In Octave you must start installing the instrument-control package to communicate with the serial port. If it is not already installed you can install it by running:

pkg install –forge instrument-control

* The package is then loaded into the script

pkg load instrument-control

* Next, you must open the serial port and read the data, specifying the number of bytes to be read.

## 2.4 Processing data

* If the Arduino is sent numbers (as is our case), the read data can be converted to numerical values.
* These values are then stored to perform mathematical operations depending on whether it is required or not.
* These values are shown in two graphs of *(values vs t)*
* They can also be saved in spreadsheets such as Excel for later handling.

# Code

% GNU-OCTAVE visualization for the flow project

% By: Rojas Rondon

%

clc, clear;

% Load instrument control PKG

pkg load instrument-control;

%config serial port

port = 'COM7'; % Change this for ('COM5','COM7')

bd = 9600; % baud rate from Arduino

% Open serial port

s = serial(port, bd);

% Start vector values

display\_time = [];

value1 = [];

value2 = [];

% Start read config (in secounds)

duration = 100; % number of durations

start\_time = time(); % Get start time

% Create a fig

figure;

% Principal loop for read and display

while time() - start\_time < duration

% Read a line completly

line = '';

while isempty(line) || line(end) != char(10) % 10 es el cÃ³digo ASCII para '\n'

line = [line, char(srl\_read(s, 1))];

end

% Quit first character

line = strtrim(line); % clean lines

% Divide the line in 2 values

values = strsplit(line, ',');

% Check

if numel(values) == 2

% Convert values in numbers

flowrate = str2double(values{1});

vol = str2double(values{2});

% Save data

new\_time = time() - start\_time;

display\_time = [display\_time, new\_time];

value1 = [value1, flowrate];

value2 = [value2, vol];

% Display values

disp(['Time: ', num2str(new\_time), ' s, FlowRate: ', num2str(flowrate), ', Vol: ', num2str(vol)]);

% Start graph in real time

clf; % Clean graphs

% Count

subplot(2, 1, 1); % 2 rows, 1 column, 1st graph

plot(display\_time, value1, 'r', 'LineWidth', 1.5);

xlabel('Time (s)');

ylabel('FlowRate (L/min)');

title('FlowRate vs Time');

grid on;

% 2nd graph

subplot(2, 1, 2); % 2 rows, 1 column, 2nd graph

plot(display\_time, value2, 'b', 'LineWidth', 1.5);

xlabel('Time (s)');

ylabel('Vol (L)');

title('Vol vs Time');

grid on;

% Update the fig

drawnow;

else

disp('Error: Something is wrong, two values â€‹â€‹were not received');

end

pause(0.1); % break

end

% Close serial port

fclose(s);

The Fig 1. 1 shown below is the result of successfully running this program.

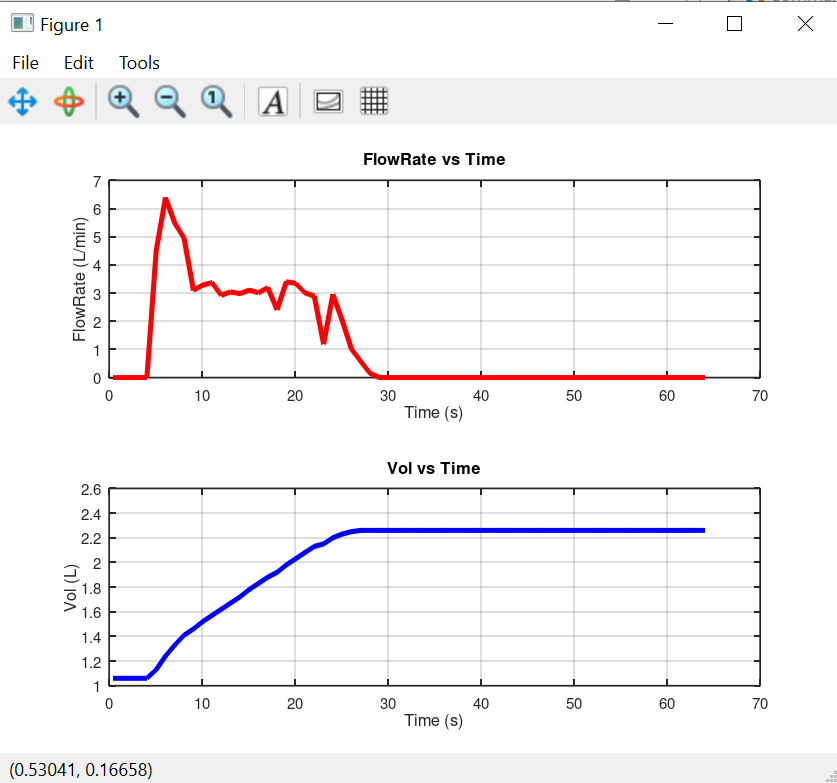


Fig 1. Data display.