Raspberry Pi, Python, Open Source Software, GPIO, RC circuit, MCP3008
Andrea Mandanici, Dipartimento di Scienze Matematiche e Informatiche Scienze Fisiche e Scienze della Terra, Universitá degli Studi di Messina, Messina, Italy

Studying Physics, getting to know Python: RC circuit, simple experiments, coding and data analysis with Raspberry Pi

Andrea Mandanici[1], Salvatore Alessandro Sara'[1], Giacomo Fiumara[1], and Giuseppe Mandaglio[1]

[1]University of Messina

November 3, 2020

Abstract

Raspberry Pi (RPi) is a well known single-board computer natively equipped with a Linux-based operating system, Raspbian, and a powerful programming language, Python. In this work, we propose an integrated project on physics and computer science exploiting RPi and Python: a set of lab activities, coding, and discussion related to the study of charging and discharging phases of a capacitor in an RC circuit. In our simple experiments, entirely computer-controlled, the RPi and Python scripts are used to: (i) apply a known constant voltage to the circuit at a desired time; (ii) measure the voltage on selected circuit elements as a function of time; (iii) evaluate and analyze experimental data. This approach is based on inexpensive hardware and open source software. It allows a hands-on experience with electric circuits and with dedicated examples of Python coding. The codes involve Python modules such as Numpy, Scipy, and Matplotlib that prove to be easy to use and efficient for our goals, supporting the choice of Python language for further study or research tasks.

Introduction

Eager learners who undertake the study of physics with the help of computing can achieve a better understanding of the problems, improve the presentation of the results, explore different cases, simulate systems, test the limits of the models (and have fun). This can be done with open source software (Mandanici, 2018) and, in particular, introductory topics in physics can be afforded with basic skills of Python coding (Malthe-Sørenssen, 2015; Pine, 2019). On the other hand, the search for smart suitable codes for the study of physics can become an opportunity to learn Python, even better if this is done with low-cost hardware. With these ideas in mind, RPi represents a cheap, portable, and effective tool for educational experiments within integrated projects involving physics, mathematics, and computer science (Mandanici & Mandaglio, 2020; Singh & Hedgeland, 2015; Robinson, 2018). Here we use it to study the transients in RC circuits, an introductory topic to time-dependent electric signals (Galeriu et al., 2015; Pereira, 2016; Sanjaya et al., 2018). A conventional laboratory setup would include a DC power supply or a signal generator, an oscilloscope, and a personal computer connected to the oscilloscope for data acquisition and evaluation. Moreover, dedicated software would be needed to transfer the data from the oscilloscope to the computer and perhaps also for data analysis. An alternative solution that we propose is to use the following hardware: the RPi and an Analog to Digital Converter available as an integrated circuit at low cost. The software needed consists of Python scripts based on open source libraries: (a) a module to control the Input/Output of signals via suitable pins of the RPi board; (b) selected modules for data evaluation and analysis. The circuits are powered by the RPi. We describe an example of how to set up the experiment, acquire the data, and analyze them, providing the Python codes for the different tasks.

Hardware and Software

The experiments are performed using a Raspberry Pi 3 Model B board running the Raspbian operating system. We exploit the set of General Purpose Input Output pins (GPIO) of the RPi (Fig. 1)

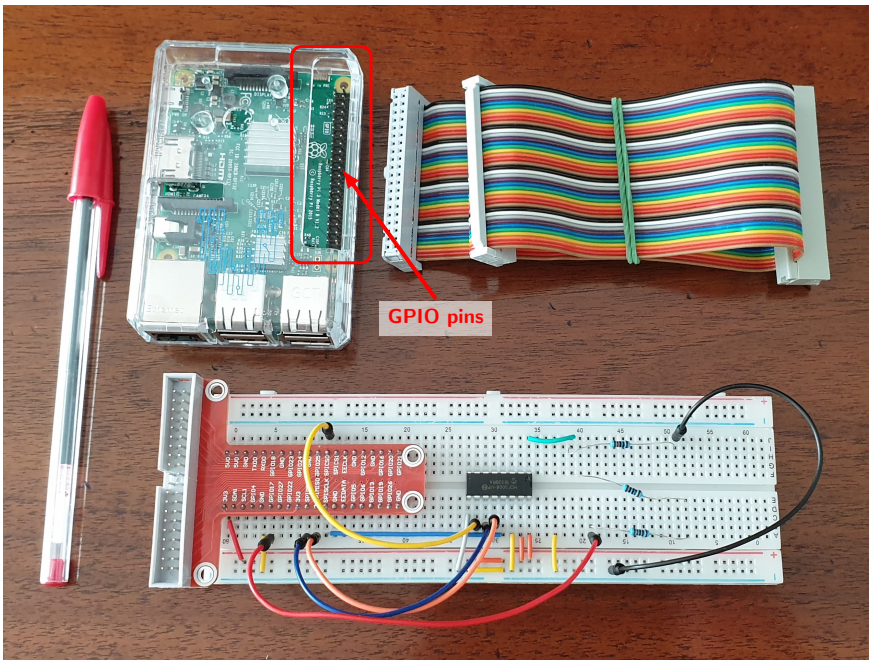

Figure 1: *Equipment.* Raspberry Pi, 40-pin cable, extension board, solderless breadboard, jumpers.

The RPi can be controlled to provide digital output through some pins of the GPIO that can be designated as output pins, giving high signal (3.3 V) or low signal (0 V). The RPi can also be set via software to receive digital signals at pins designated for input. The codes can be edited, debugged, and executed using the Thonny Integrated Development Environment, available with the Raspbian operating system. For our experiments in which it is required to read the values of analog inputs, the RPi is connected to an Analog to Digital Converter, the integrated circuit MCP3008. A 40-pin cable and an extension board are used to allow easier connections. The circuits studied are assembled on a solderless breadboard, a quite convenient arrangement for a hands-on tutorial. All the codes and wiring diagrams are provided as Supplementary Materials in a GitHub Repository along with a parts list.

Experiments and results

Experiment Zero: turn a LED on and off

In this study, the circuits are powered by the RPi through the GPIO. The first basic experiment is realized connecting a Light Emitting Diode to the GPIO. The LED can be switched on and off via software using a few lines of Python code based on the **gpiozero** library, as shown for instance in the listing **gpio**$_s$*witching.py.The***time***library is also used, in order to deliberately introduce pauses of assigned duration between selected ste*

Read analog signals: measure a constant voltage

In order to read the values of the voltage on a given component of a circuit, we connect the MCP3008 to the RPi through the GPIO pins. One of the pins supplies a $V_{DC} = 3.3\,\mathrm{V}$ constant voltage and it can be used as a power supply for the MCP3008. The Analog to Digital Converter MCP3008 has 8 analog input channels $0-7$. It samples the signal with 10-bit resolution. The value attribute of the gpiozero.MCP3008 Python class provides a decimal number, in the range between 0.0 and 1.0, that is the ratio between the measured voltage and the reference voltage, V_{REF}, which in the present case is equal to V_{DC}. A simple circuit used to test the setup is a voltage divider realized with three $1\,\mathrm{k\Omega}$ resistors connected in series (Fig. 1).

The script measure_voltage.py allows the student to read and print the voltage on one of the resistors of the series when the circuit is powered on and off, respectively. With a similar experiment we observe that, if the signal is kept on for a period of $10\,\mathrm{s}$, occasional fluctuations of the measured voltage occur. They correspond to one Least Significant Bit only, i.e., to the voltage resolution. The uncertainty on the voltage measurements can be assumed equal to the voltage resolution. This is limited by the number n of bits used for sampling, as $V_{RES} = V_{REF}/(2^n - 1)$.

RC circuit

The transient response of the RC circuit is studied by applying a $3.3\,\mathrm{V}$ voltage between the two ends of a series made by a $10\,\mathrm{k\Omega}$ resistor and a $100\,\mu\mathrm{F}$ electrolytic capacitor. The expected time constant of the RC circuit is $\tau = RC = 1\,\mathrm{s}$. We need to measure the voltage between the two ends of the capacitor, but we also need to acquire the time to which the measurement refers to. This is done using the perf_counter() function of the time library.

The script measure_RCtransients.py is used for running an experiment on the transient response of the RC circuit. The voltage applied to the RC series is set to high level, $V_{DC} = 3.3\,\mathrm{V}$, via software at the beginning of the experiment. Immediately after, the potential difference on the capacitor starts to be measured as a function of time. The time duration between two subsequent voltage measurements is set by the parameter sleep_time, which in this example is $100\,\mathrm{ms}$. The applied voltage is kept to the same level for $20\,\mathrm{s}$, then, again via software, it is switched to $0\,\mathrm{V}$ and measurements are performed during the next $20\,\mathrm{s}$. The results are written to a text file. The graphical representation of the measured voltage values as a function of time as shown in Fig. 2 can be obtained with the script plot_voltage.py based on the Matplotlib library.

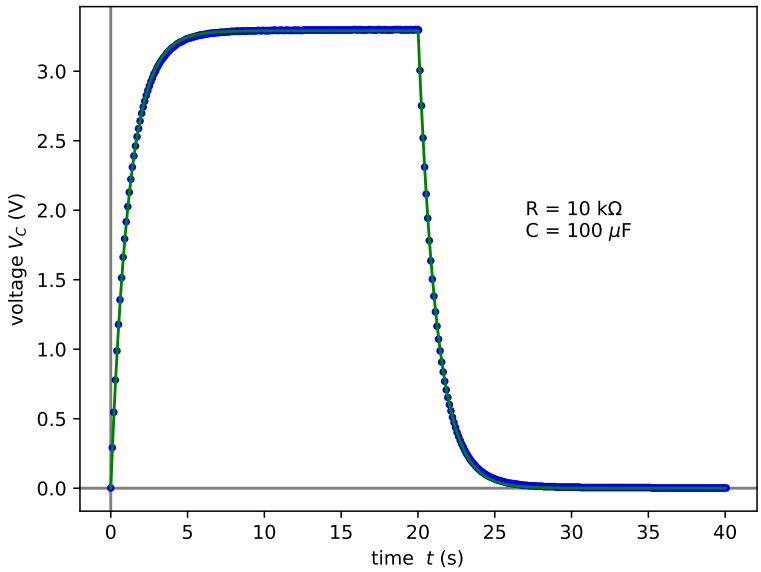

Figure 2: *Voltage on the capacitor* in an RC circuit as a function of time while charging and discharging the capacitor. The voltage values are measured using a Raspberry Pi 3 and the Analog to Digital Converter MCP3008.

Analysis

Modeling the experimental data

The measured values of the voltage V_C on the capacitor as a function of time for the RC circuit while charging the capacitor can be modeled by the equation

$$V_C(t) = V_0\{1 - \exp[-(t - t_0)/\tau]\} \tag{1}$$

whereas, during the discharging process,

$$V_C(t) = V_0 \exp[-(t - t_1)/\tau] \tag{2}$$

We analyze the two phases separately. Concerning the capacitor charging process, the values of the parameters V_0, τ, and t_0 can be obtained as those providing the best fit of Eq. 1 to the experimental data. In the Python script for the analysis, we use the NumPy package to deal with numerical arrays. For the non-linear curve fitting, we use instructions based on the optimize package of the SciPy library, referring to the examples available in (Pine, 2019). The best-fitting curve resulting from the analysis is shown along with the experimental data in Fig.2. The example code curveFit_charging.py producing also a plot of the residuals is provided as Supplementary Material. The values of the parameters are $V_0 = 3.2915 \pm 0.0008\ V$, $t_0 = 0.0000 \pm 0.0023\ s$, $\tau = 1.1687 \pm 0.0033\ s$. The values of τ obtained from the curve fitting procedure, both for the charging and discharging processes, are compatible with the expected value of the time constant of the RC circuit. In fact, the nominal value of the capacitance is $C = 100\,\mu\text{F} \pm 20\,\%$, while the nominal value of the resistance is $R = 10\,\text{k}\Omega \pm 1\,\%$, leading to an estimated value of $1.00 \pm 0.21\ s$ for the time constant τ.

A closer inspection of the results of the fitting can be performed plotting the residuals, and plotting the data on a semilogarithmic scale. These approaches (see Supplementary Materials) reveal that the behavior of the experimental data deviates from the model curve, which physically only corresponds to an approximation (Chabay & Sherwood, 2015). The deviation from the ideal behavior could also explain the difference of about 3 % between the values of the RC time constant corresponding to charging and discharging, respectively.

Insights

Acquisition rate

The delay between two subsequent voltage measurements is set through the sleep parameter of the time library. If we want to check what is the real rate at which the voltage measurements are acquired, we read the experimental values of time and we compute the length of the time between measurements $\Delta t_i = t_{i+1} - t_i$. Applying statistic functions to the set of Δt_i's we obtain the average difference between measurements and the standard deviation (rate_analysis.py). Moreover, the distribution of the time differences can be easily shown as a histogram. The maximum acquisition rate can be increased by nearly a factor of 3 using the spidev Python module.

Conclusions

A setup based on a Raspberry Pi and on the Analog to Digital Converter MCP3008 can be suitable to study with Python codes the behavior of an RC series circuit during the transient phases observed while charging and discharging the capacitor. The steps of the experiment and of the analysis are easily understandable and accessible to undergraduates. Basic skills in programming are sufficient to deal with the different tasks of the proposed laboratory activity. The hardware needed to build the experimental setup can be easily obtained at low cost. The software to perform the measurements and the data analysis can be developed using open source libraries among which NumPy, SciPy, and Matplotlib. The project offers opportunities for improving the knowledge of the physical problem, for achieving a lab experience, and for obtaining a practical demonstration of the potentialities of the Python language. Further experiments on electric circuits could be designed starting from the example proposed.

Acknowledgments

The authors would like to recognize our encouraging interactions with the Editor-in-Chief Lorena Barba and with the Department Editor Dirk Colbry. They would also like to thank Dirk Colbry for the constructive revision of the manuscript and for his suggestions (Colbry, 2020).

References

Studying a physics problem with the help of open source software. (2018). *European Journal of Physics*, *39*(5), 055805. https://doi.org/10.1088/1361-6404/aad16a

Elementary Mechanics Using Python. (2015). Springer International Publishing. https://doi.org/10.1007/978-3-319-19596-4

***Introduction to Python for Science and Engineering*. (2019). CRC Press.** https://doi.org/10.1201/9780429506413

Experiments and data analysis on one-dimensional motion with Raspberry Pi and Python. (2020). *Physics Education*, *55*(3), 033006. https://doi.org/10.1088/1361-6552/ab73d2

Special relativity in the school laboratory: a simple apparatus for cosmic-ray muon detection. (2015). *Physics Education*, *50*(3), 317323. https://doi.org/10.1088/0031-9120/50/3/317

Solar wind monitor: a school geophysics project. (2018). *Physics Education*, *53*(3), 035017. https://doi.org/10.1088/1361-6552/aaacb2

An Arduino Investigation of the RC Circuit. (2015). *The Physics Teacher*, *53*(5), 285288. https://doi.org/10.1119/1.4917435

Measuring theRCtime constant with Arduino. (2016). *Physics Education*, *51*(6), 065007. https://doi.org/10.1088/0031-9120/51/6/065007

Numerical Method and Laboratory Experiment of RC Circuit using Raspberry Pi Microprocessor and Python Interface. (2018). *Journal of Physics: Conference Series*, *1090*, 012015. https://doi.org/10.1088/1742-6596/1090/1/012015

***Matter & Interactions*. (2015). Wiley.**

***Review: Studying Physics, getting to know Python: RC circuit, simple experiments and coding with Raspberry Pi*. (2020).** https://doi.org/10.22541/au.159309337.74459966