Arduino as a tool for physics experiments

Giovanni Organtini
Sapienza Università di Roma & INFN Sez. di Roma

Abstract. Arduino is a widely used open-source platform composed of both hardware and software tools that can be very useful in a physics laboratory. Its low price, the large availability of sensors and transducers, its ease of use and its open nature makes it a perfect tool to involve students in active and cooperative learning. In this paper we show a few examples of what can be done using an Arduino and some sensor, and propose a template for documenting activities.

1. Introduction
New trends are emerging in education and, in particular, in physics education. Besides inquiry based learning (IBL), cooperative learning (Slavin 2011) and project based learning (Krajcik & Blumenfeld 2006) are being recognised as effective learning strategies. Moreover, the integration of physics with other disciplines, in the past, was very limited to integration with mathematics. Nowadays, the ability of integrating different disciplines in learning strategies is considered very important. In particular, the need for the development of some ability in coding and in so-called tinkering, is widely accepted, and integration with various forms of art is becoming to be advisable. Already in 1988 Italo Calvino wrote that “it is true that software cannot exercise its powers of lightness except through the weight of hardware. But it is the software that gives the orders” (Calvino 1988): a very clear tribute to the fact that the ability to organise and manipulate data is much more important than the tools used to do that.

Teaching physics should not be limited to the transfer of physics laws to pupils: the reason for which we should teach physics to our students in schools is that physics is a very successful example of scientific methodology. We should then teach them how to apply methodology in physics, in order to develop their abilities in applying similar strategies in everyday life, rather than expect from them to be able to solve problems that, in most cases, they will never face.

With this respect, it is somewhat bizarre that pupils are practically trained in many disciplines, but not in scientific ones. Only exceptional students learn how to do physics: they usually learn the physics produced by others, not the physics made by themselves, nor how to do that. Learning physics this way is like learning literature without doing a composition, or learning art without doing a drawing, or even learning music without having tried to play an instrument.

A very common reason for this is that doing physics, at least until recently, needs a laboratory equipped with many expensive and complicated instruments. Where available, instruments are often limited in number and cannot be used at the same time by each student. Most of the time a teacher or a technician uses the available equipment to show some effect to the pupils.

With the advent of Arduino (www.arduino.cc), it is possible to fully equip a laboratory with enough instruments to allow each student to perform a lot of experiments and measurements with limited investment. Moreover, learning the Arduino basics requires the development of some coding ability and the open nature of the product promotes cooperative learning. Project based learning is a natural consequence of the need to design and build the experiment with the provided material.
In this paper we illustrate how Arduino can be a useful tool for a physics teacher, assuming that the reader is completely unfamiliar with it. In the following sections we describe the Arduino platform and the most important programming tools for using it as a tool for doing physics, together with examples of experiments.

2. The Arduino platform
Arduino is an open source platform composed of electronic boards, sensors and expansion boards, as well as a software development environment, initially developed in Italy by a team lead by Massimo Banzi. Thanks to its low cost and the free availability of its hardware design and software, the Arduino platform rapidly became an international standard and is now widely adopted all over the world for various kinds of projects: from fast prototyping to Internet of Things (IoT) projects.

The main boards exist in various flavors and form factors. The simplest and most widely used one is the Arduino UNO, based on the ATmega328P chip running with a clock at 16 MHz. All the boards share the same programming environment and may differ for form factor, memory size, number and type of ports and speed. For the purpose of a physics laboratory the Arduino UNO board is perfectly suitable. The board can be bought for as low as about 20 euros.

![Arduino UNO Board](image)

**Fig. 1.** The Arduino UNO board. On top of the figure there is a row with 14 digital pins. Power and analog pins are on the opposite side.

The Arduino UNO board (Fig. 1) has 14 digital I/O pins. Digital pins may have two states: LOW and HIGH, corresponding to a voltage of 0V and 5V, respectively. When used as output ports, digital pins can provide a current up to 20 mA.

Six out of 14 pins can be used as Pulse Width Modulation (PWM) outputs. They can be assigned a value between 0 and 255 and, correspondingly, the pin stays HIGH for a percentage of time between 0 and 100% with a 2 ms base.

It also has six analog inputs, each with ten bits resolution, and with a dynamic range from 0 V to 5 V. Power ports provide a 5 V and 3.3 V source together with the ground reference (GND) and provides currents up to 50 mA.
Fig. 2. The Arduino IDE window. The program is written inside the main, white window. The program is transferred to the board using the button with an arrow on the top left side of the window.

A USB connection allows a user to program the chip using the Integrated Development Environment (IDE) software, freely available from the Arduino website. The user writes his/her own program on a PC, compiles it and then transfers the program to the Arduino memory, that starts immediately executing it every time it is powered on. Arduino boards can be powered either using the USB cable connecting it to the PC for programming it, or via an external source providing from 7 to 12 V (a 9 V battery is perfectly suitable).

3. Arduino programming
The board is fully programmable in C++. The programming environment is such that a deep knowledge of Object Oriented Programming is not needed. Contrary to a computer, a microprocessor does not run an operating system: it just executes the only task loaded in its memory. In order to write the program, the user can use the freely available IDE on a PC. When launched it appears as in Fig. 2.

The program is entered in the main, white window as in most common text editors. Once ready, the syntax of the program can be checked using the checkmark button and, if correct, can be transferred to the board acting on the button with an arrow seen on the top left part of $\vec{F} \cdot \Delta t = \Delta \vec{V} \cdot \Delta t$. Once transferred, the program starts soon and restarts from scratch each time the board is powered on. Programming an Arduino consists of writing a sequence of operations that the microprocessor has to perform.
once, at the beginning of the run, collected under the function \texttt{setup()} (see Fig. 2), and a sequence of operations that the microprocessor executes repeatedly, until it is powered off, within the function \texttt{loop()}.

![Fig. 3. The voltage across a capacitor in an RC circuit as measured by a Arduino UNO board connected to an RC circuit](image)

Programmers may add functions to the program, even if they are not strictly needed. They can be regarded just as a way to keep the code well organised and simple enough. It must be noted that, in order to perform most of the experiments described below, no particular programming abilities are required and that, in most cases, the needed abilities can be gained in one to three days of trial and error sessions even in the absence of formal training sessions.

We describe the main software functions for using Arduino together with the relevant examples in the sections below.

**Using analog ports**

A very basic usage of analog ports consists in performing simple electrical measurements. Analog ports can be electrically connected to any voltage source from 0 V to 5 V. The Arduino Analog to Digital Converter (ADC) converts the input voltage into a ten bits number from 0 to 1023. Hence, connecting any part of a circuit to an analog input pin, with the ground in common with the Arduino GND pin, allows one to perform electrical measurements with a resolution of 5 V/1024 \(\approx\) 5 mV.

A great advantage with respect to the usage of a common multimeter is that the measurements can be taken repeatedly at short intervals and/or based on a given pattern. For example, a RC circuit can be easily realised using commonly available capacitors and resistors without the need for choosing large values for both C and R to let the student measure the voltage across the capacitor during the charge and the discharge phase.

Thanks to the 16 MHz clock, measurements can be repeated many tens of times every microsecond, allowing very detailed and accurate measurements of the voltage \(V_C\) across the capacitor versus time, as shown in Fig. 3.
Fig. 4. An Arduino board connected to an RC circuit using jumper wires.
The circuit is realised with the help of a breadboard

Building the circuit, in this case, is extremely simple. It is enough to connect a terminal of the capacitor to the GND pin, the resistor in series to the capacitor and the free lead of the resistor to the 5 V pin of the Arduino, while keeping the other terminal of the capacitor connected to one of the analog pins of the Arduino board. Connections can be easily done using a breadboard and jumper wires, as seen in Fig. 4.

Programming an Arduino to perform this kind of measurement is straightforward. The setup() function is as simple as

```c
void setup() {
    Serial.begin(9600);
}
```

while the loop() function reads as

```c
void loop() {
    Serial.println(analogRead(A2)*5./1023);
    delay(150);
}
```

The Serial.begin(9600) statement (statements are separated by a semicolon) lets Arduino connect to the PC via the USB port at a speed of 9600 bauds (a unit of measure of the transmission speed). The loop() function consists of a statement to convert the voltage at the analog pin A2 in a ten digit number (analogRead(A2)) and to convert it to Volts using the conversion factor 5/1023 as explained above. The result of the operation is sent to the PC via the USB connection using the Serial.println() statement. The delay(150) statement causes the microprocessor to wait for 150 ms before executing again the loop.

This extremely simple program performs a measurement of the voltage between the GND and the A2 pin every 150 ms and prints the values read on the “serial monitor”: a tool available in the Arduino
IDE consisting in a window to which each `Serial.println()` statement causes the appearance of the number inside the parenthesis. Each time the statement is invoked a new line is written.

The Arduino IDE also provides a “serial plotter”: a window to which a graph appears based on the numbers sent to the PC via the statement above.

**Using digital ports**

Digital ports can be used in a variety of ways. For example, with ultrasonic sensors, one can perform interesting measurements in the field of mechanics. Position (in one dimension) versus time can be measured using available modules made of two piezoelectric devices (like, e.g. the HC-SR04): one used as a transmitter and the other as a receiver.

The module has four pins: two of them (GND and VCC) are used to power it and must just be connected to the corresponding GND and 5 V Arduino pins. The other two pins are called “trigger” (TRG) and “echo” (ECHO). Sending a square wave of at least 10 μs width to the TRG pin causes the module to produce a train of ultrasound waves from the transmitter. If the waves encounter and obstacle, they are reflected back and can then be detected by the receiver, that raise the ECHO pin from LOW to HIGH. Measuring the time t elapsed from the trigger to the echo, one can obtain the distance d of the obstacle knowing the speed of sound c in air, as d = ct/2.

A set of repeated measurement of distances and times can be obtained with the following code (in the example the TRG pin is connected to Arduino digital pin 2, while the ECHO pin is connected to the Arduino digital pin 3; row numbers are added for reference in the text):

```cpp
1. void loop() {
2.   int x = 0;
3.   digitalWrite(2, HIGH);
4.   delayMicroseconds(10);
5.   digitalWrite(2, LOW);
6.   x = C*pulseIn(3, HIGH)/2.;
7.   t = micros();
8.   Serial.print(t);
9.   Serial.print(" ");
10.  Serial.println(x);
11. }
```

The `digitalWrite()` function allows a programmer to set the state of the given pin to one of the possible states of the pin (LOW or HIGH). The `delayMicroseconds()` function put the microprocessor in pause for the number of microseconds specified in parenthesis. Rows from 3 to 5 produce a square wave as requested by the ultrasonic module: digital pin 2 is initially in the LOW state; it is then put at the HIGH state and back to LOW after 10 μs.

The `pulseIn()` function takes two arguments: a pin and a state. It waits until the given pin (the ECHO pin in this case) changes into the specified state (in the example from LOW to HIGH) and returns the time elapsed in μs. The integer variable x is then assigned with the distance measured as outlined above using the constant C defined elsewhere as the speed of sound m/μs. The t variable, instead, defined elsewhere, is set to the current time using `micros()`, a function returning the elapsed time since the beginning of the program in microseconds. Lines 8 to 10 print the values of t and x on the serial monitor.

This is just an example of the enormous number of possible measurements that can be done using the digital pins. Indeed, digital pins can be used to represent data in binary form and can then be used for much more complex measurements. A variety of dedicated modules exists engineered such that the values measured by onboard sensors are digitised on the module itself and transmitted to Arduino using the I2C protocol via digital pins.
4. External transducers
The open nature of the Arduino platform stimulated the production of many, cheap expansion boards, called “shields”, that just plug onto the Arduino board and provide specialised functions. Interface shields and transducers of various types can be bought for as low as 1-2 euros and usually do not exceed 10 euros each, depending on the complexity of the systems.

Sensors for temperature, humidity, pressure, light, sound, acceleration, magnetic field, currents, etc. exist, allowing the realisation of a lot of physics experiments, ranging from calorimetry to electromagnetism, from thermodynamics to the physics of waves. The only limit is the ability in designing an experiment that, in turn, can be developed with time and experience gained on the simplest ones.

Using the Arduino platform to perform the measurements has another advantage with respect to specifically designed equipment: the latter appears to students more like “black boxes”. The student does not really understand how they work and just get results from them. Using Arduino, students can understand much more deeply the principles of operation of the system. Moreover, using Arduino allows to appreciate many aspects of the measurement process that are usually neglected using dedicated, “closed” apparatus. As an example, consider the measurement of the distance of an object using an ultrasonic sensor: the speed of sound, e.g., depends on temperature, pressure and humidity, at least. The size of the obstacle matters, as well as the presence of other sources of reflections around the experimental area. A proper, detailed analysis of these sources of systematic error can lead to interesting, instructive discussions.

5. Resources
There are plenty of tutorials on Internet about how to operate Arduino. Even if they are very instructive, they are not specifically designed for building scientific experiments. Only recently the pedagogical content of the Arduino platform is being recognised and we expect the number of dedicated resources to increase in the future.

We already started to provide some dedicated resources: we are going to organise regular sessions of a school of physics with Arduino, to take place at least twice per year, and we made available an e-book about the scientific use of Arduino on our website (Organtini 2017). During the school of physics with Arduino we ask participants to design and implement physics experiments using such a technology. A brief documentation of each experiment is then made available through our website (http://www.phys.uniroma1.it/fisica/Arduino-Smartphone-Esperimenti). The documentation is still only available in Italian, but we plan to translate it in English, too.

We look forward for collaborators willing to help in the realisation of detailed guidelines about specific experiments that can be done with Arduino and to create an international network of experts to organise regular international workshops on this subject.

References
Calvino I. (1988) *Six Memos for the Next Millennium*. Cambridge, USA: Harvard University Press.
Krajcik J.S. & Blumenfeld P.C (2006) Project Based Learning. In *The Cambridge Handbook of the Learning Sciences* (Sawyer R.K., ed.), New York, USA: Cambridge University Press.
Organtini G. (2017) *Scientific Arduino Programming*. Available for download at http://www.roma1.infn.it/people/organtini/publications/scientificArduino.pdf.
Slavin R. (2011) Instruction Based on Cooperative Learning. In *Handbook of research on learning and instruction* (Mayer R.E. & Alexander P.A., eds.), New York, USA: Routledge.