SMB-Science Magic Box

<Author> Luc Ivarra
<Author> Marco Nicolini

<Info>

<Keywords> microcontroller, transducer, sensor, actuator, signal, physical quantity, loop, branch, sequential, process, calibration, input, output, read, write, analogue, digital, linearity, conversion, breadboard, pin, soldering, human machine interface

<Disciplines> physics, electronics, mathematics, ICT, logic, biology

<Age level of the students> 14–18

<Hardware> Arduino UNO[1] with Arduino DUE[2] and/or TI-Nspire CX CAS with TI-Innovator Hub

<Language> C++ (using Arduino IDE[3]) and/or TI-Basic

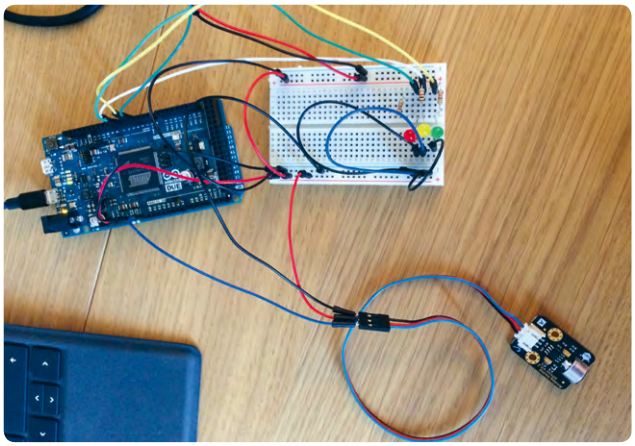
<Programming level> medium, with an audio section for advanced students

A list of the abbreviations, special terms and acronyms used in this unit is available online.[4]

<Summary>

The students will learn how to program for a self-built hardware-software environment (based on Arduino) and for a ready-to-use pocket computer (TI-Nspire CX CAS calculator with its extension, the TI-Innovator Hub). Both are used as devices for sensor data collection, conversion and transduction to easily handle, read, convert and actuate physical quantities.

Using the Arduino platform, the students will code in a framework of sensors that acquire physical quantities as input signals, and actuators that react to the acquisition and produce an output signal as a physical quantity after a microcontroller has processed the detected signal to set the proper output (see 1).



1: Arduino board

The TI-Innovator Hub is a 'ready to use' box that enables students to learn the basics of programming. It must be plugged into a TI-Nspire CX CAS calculator. It has a good I/O interface, which includes a luminosity sensor, two LEDs and an on-board buzzer, which produces a sound of a given frequency (see 2).

2: TI-Nspire and TI-Innovator Hub

<Conceptual introduction>

The unit introduces students to the coding world for physical problems, based on detecting and measuring physical quantities, processing the data, reacting and deciding to perform an action with actuators.

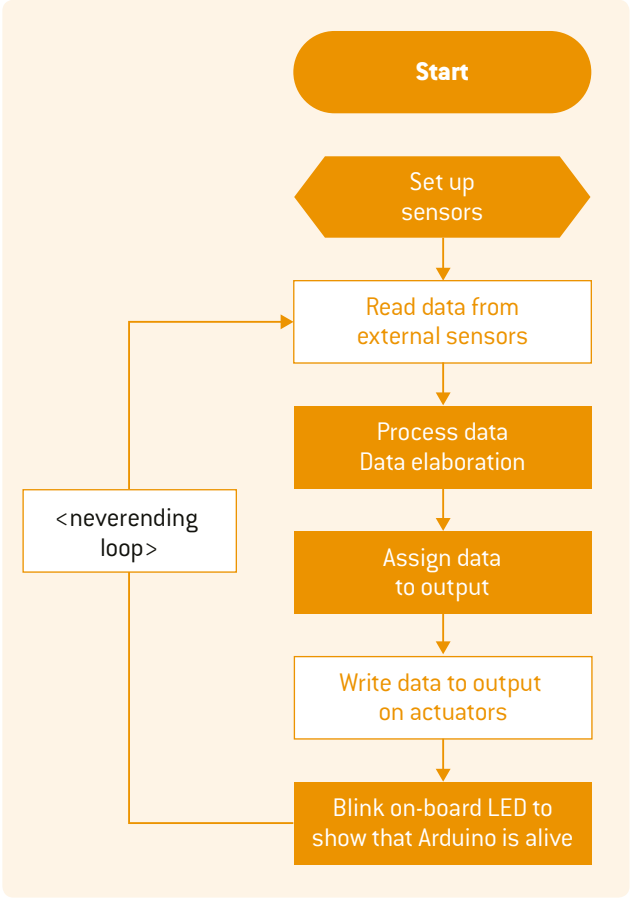
The code is normally based on an infinite loop (the machine is usually 'alive' as long as it is powered, and it must work all the time), where the three actions of measuring, processing and acting are performed in this order.

The students will learn that they can write any code with

1. sequential instructions
2. loops (while … do; repeat … until)
3. branches (if…then…else)

as stated in the Böhm-Jacopini theorem (see 'Additional information'[4]).

The second goal of the unit is to introduce the students to microcontrollers. They will learn to set up input sensors and output actuators, using digital or analogue ports, and how to write a simple program that reads input, processes data and writes output. They can choose to output either sound or light signals. These can be interpreted as 'alarms' which issue a warning based on the input read by the sensors.



3: Flow chart

The students will use the Arduino[1] integrated development environment (Arduino IDE[3]) to program in C++, with ready-to-use function libraries that make the coding process easier and faster.

A breadboard (see 'Additional information'[4]) must be available to allow the students to prototype and easily connect sensor pins to the Arduino I/Os, 5V power supply and GND pin.

The structure of any program, in metalanguage, is shown in 3.

Please note that the meta instruction *'While (TRUE)'* is a trick to tell any processor to repeat the included instructions indefinitely (as long as the microcontroller is powered).

The third main goal is to learn how to convert an acquired physical quantity into another quantity (e.g. light intensity into sound), ready for transmission to the external environment, by following these steps:

1. The physical signal (light, sound, force, energy …) is acquired by the sensor and converted into an electrical signal.
2. The electrical signal is transformed into a number available to the processor.
3. The number is processed and transformed by the processor into another number, and then used to do an action with an actuator transducer.
4. The actuators convert the number into electrical signals that are ready to be output.
5. The electrical signal is finally transformed into a physical signal (e.g. sound, light).

In 1 and 5, the signals must be converted from one form into another. In these phases, the linearity of the transformation, or the 'close to linearity' dependence, is extremely important (see 'Additional information'[4]).

See 'Additional information'[4] for a detailed explanation of the last line of the metacoding ('Blink on-board LED').

The input signal is usually called a 'stimulus' and comes from the environment where the sensors are placed to get data. The processor and the code are designed to 'react' to the stimulus with mathematical/logical operations (performed by the code instructions, processed by the microcontroller) and to output this 'response' to the environment. In our activities, the environment is the space surrounding the Arduino, which is able to 'see', 'hear' and 'feel forces' thanks to the sensors.

Working with the TI-Innovator Hub allows the students to focus more on the coding part of their work, as the microcontrollers and sensors are already set up and ready to use.

<What the students/teachers do>

We recommend that you begin by brainstorming to collect all your students' naïve ideas about sensors and automatic machine control. Collecting these ideas, gaining practical experience with the sensors and automatic machine control and then comparing the results with their previous (maybe incorrect) ideas is a good way to help the students to truly understand the process.

You could prepare a form with questions like:

↳ Do you know how a thermostat controls the temperature in a room?
↳ What is a sound warning-based parking system for? How does the driver react when the sound is emitted by the system?
↳ Do you have an induction hob in your kitchen? What does an illuminated LED mean?

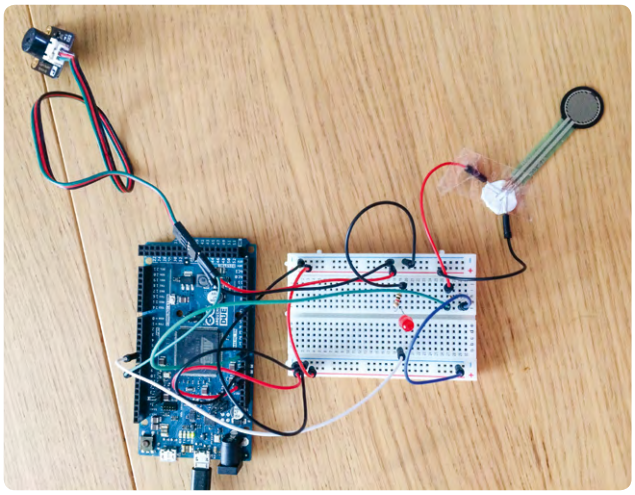
See 'PEC and list of questions'[4] for an example of a complete question list and references to the PEC (Prevision, Experience, and Correction) teaching methodology.

<Theoretical phase with Arduino[1]>

The teacher will introduce C++ programming[3] with the structure and basic instructions so that the students can write a simple loop using the instructions *analogRead, digitalRead, analogWrite, digitalWrite, if…then…else, loop, while*.

Hardware part

The teacher will present the microcontroller layout, showing the microprocessor, the analogue I/O pins (connections) and the digital input/output pins (connections).



4: Arduino, breadboard and sensors

The students will learn that any sensor/actuator usually has multiple connections:

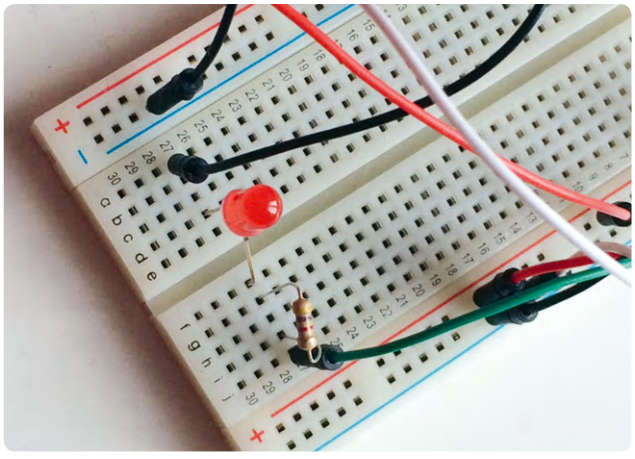
- ↳ to the 5V or 3.3V Arduino output to get the power supply
- ↳ to the GND (ground) signal so that a current can flow

and

- ↳ to another digital or analogue input pin if it is used to read (get) external data

or

- ↳ to another digital or analogue output pin if it is used to take actions that generate an output, e.g. emit a sound or light, or do anything else that signals a situation (a 'write' operation)



5: Breadboard detail

Software part

The teacher will present a simple program that reads a sensor and writes an actuator, where a clear association between the physical pins and physical address on-board the microcontroller can easily be established by students.

An example of ready-to-use instructions is available online ('Program example 1'[4]).

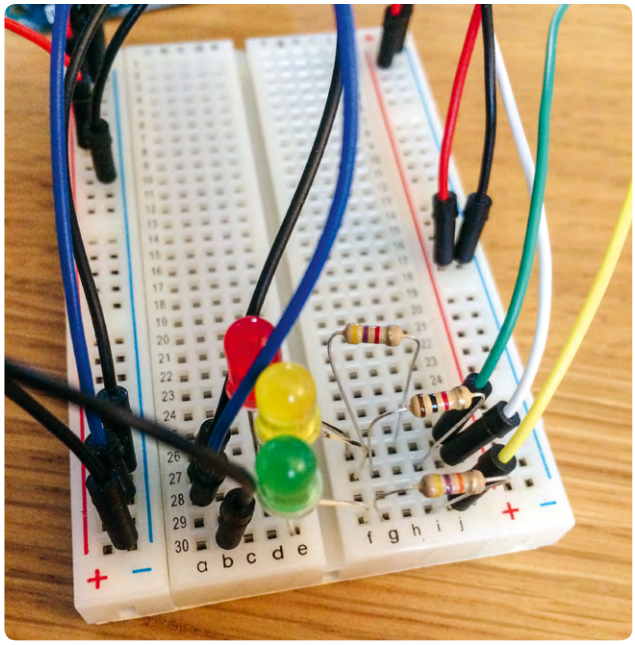
The students must keep in mind that the processor executes the code instructions one after the other and in the order in which they are written. Only the 'loop' instruction alters this principle as it tells the processor to continuously repeat the bracketed instructions as long as the microcontroller is powered.

<Practical phase with Arduino>

The students will get hands-on experience with the microcontroller, the breadboard and the sensors. The teacher should present the structure of the breadboard, showing all the available connections and how the students can take 5V, GND and I/O signals from the Arduino[1] to the breadboard. The students will be invited to copy the given coding example and to try, test and debug the code with the connected I/Os.

Hardware part

The students need the microcontroller, the sensors and short cables (10 cm) to allow easy connections between the sensor pins and the breadboard 'holes'. Sometimes you might need to solder additional cables to the sensors, but many sensors do not require this.



6: Breadboard with LEDs

The students will use the short cables to connect the 5V power supply and the GND signal to the breadboard, and the analogue/digital pins of the Arduino[1] to some 'holes' on the breadboard. This will allow the sensors to be easily lodged on the breadboard, and receive the required electrical signals (see 6).

Software part

After checking that the microcontroller-to-breadboard connections have been properly settled, with the exact correspondence between the pin logical number on the microcontroller and the sensor pin on the breadboard, the students will try to write the simple code provided ('Program example 1'[4]).

Algorithms with Arduino

We have prepared several signal conversions from one physical form to another.

Conversion of an analogue light signal into a digital light signal in an LED (modulated with a PWM feature) and a sound: the emitted sound frequency increases with the intensity of the light. Practical application: alarm clock, assistance for blind people. (See 7)

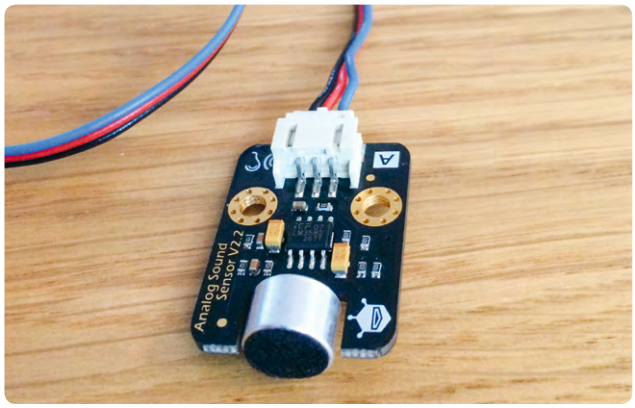
7: A light sensor

Conversion of a force detected through a force sensor into a digital light signal in an LED (modulated with a PWM feature) and a sound: the light intensity increases with the intensity of the force. Practical application: weight load alarm. (See 8)

8: A digital buzzer

Conversion of an external noise signal into a digital light signal in an LED. The higher the sound, the higher the light frequency will be. Practical application: noise pollution control.

Conversion of a distance measured with a distance sensor into a sound. Technical application: car parking sensors. (See 9)



9: A distance sensor

Conversion of a temperature signal into a sound and a light signal. Technical application: oven temperature control.

Conversion of soil moisture concentration into a light signal. Technical application: plant irrigation and watering alarms and control. (See 10)

10: A soil moisture sensor

See 'Program example 2'[4] for the code used for these applications on the Arduino board.

All these algorithms serve as signal control and warning systems which monitor a selected environment on a physical quantity and issue a warning based on the input (stimulus) read.

<Theoretical phase with the TI-Innovator Hub>

Hardware part

The teacher can show the students how easy it is to connect the sensors.

Software part

The basic programming can be done on the calculator alone; the students only need to master the TI-Basic instructions provided above. Then the hub can be connected and the students will learn how to communicate with it, i.e. to use the instructions 'read' and 'get' to acquire data and 'set' to control the outputs.

<Practical phase with the TI-Innovator Hub>

Hardware and software part

The students will start with basic examples to familiarise themselves with the hub before working on more open problems. The students will learn how to control the different outputs with small exercises, e.g. controlling the colour of the LED, making it blink, controlling the duration of the blinking and producing sounds of a given frequency. The infinite loop will again be the basic structure to continue the operations indefinitely.

Algorithms with the TI-Innovator Hub

The students will solve two open problems: computing an automatic switch which turns on the light only if the ambient light intensity is lower than a certain threshold, and computing an alarm clock, which emits a sound of increasing frequency as the ambient light increases. Further developments are possible, but extra sensors will need to be bought and plugged into the hub.

<Buying the sensors>

Information on where and how to buy the sensors is available online.[4]

<Conclusion>

At the end of these activities, we noticed that our students' understanding of coding, the general program structure as well as logic and algorithms had improved significantly.

<Cooperation activity>

A wonderful cooperation activity could be to promote self-entrepreneurship. The students could try to invent an original human machine interface (HMI) which is technically useful. This HMI should read a stimulus from an environment (atmosphere, home, human body, etc.) and react by issuing another signal that emits a warning, performs an action or signals a situation. Partner schools abroad could carry out a market survey to gauge the market demand and value that the device may have in their countries. Any school taking part in this exchange could invent a device and make market enquiries for the product assembled by the other schools. At the end of the project, the most popular tool could be produced on a small scale by a partner company, and sold. Self-entrepreneurship is highly regarded all around the world, as it offers a great opportunity to teach science, technology and finance-related subjects together.

<References>

[1] www.arduino.cc
[2] www.arduino.cc/en/Guide/ArduinoDue
[3] www.arduino.cc/en/Main/Software
[4] All additional materials are available at www.science-on-stage.de/coding-materials.

<Imprint>

<Taken form>
Coding in STEM Education
www.science-on-stage.eu/coding

<Published by>
Science on Stage Deutschland e.V.
Am Borsigturm 15
13507 Berlin, Germany

<Revision and Translation>
Translation-Probst AG

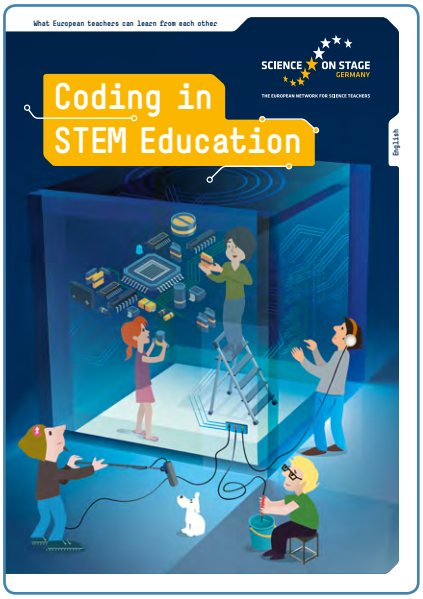
<Design>
WEBERSUPIRAN.berlin

<Illustration>
Rupert Tacke, Tricom Kommunikation und Verlag GmbH

<Please order from>
www.science-on-stage.de
info@science-on-stage.de

<ISBN PDF>
978-3-942524-58-2

First edition published in 2019



Science on Stage –
The European Network for Science Teachers

… is a network of and for science, technology, engineering and mathematics (STEM) teachers of all school levels.
… provides a European platform for the exchange of teaching ideas.
… highlights the importance of science and technology in schools and among the public.

The main supporter of Science on Stage is the Federation of German Employers' Association in the Metal and Electrical Engineering Industries (GESAMTMETALL) with its initiative think ING.

Join in – find your country on
www.science-on-stage.eu
www.facebook.com/scienceonstageeurope
www.twitter.com/ScienceOnStage

Subscribe for our newsletter
www.science-on-stage.eu/newsletter

A project by
SCIENCE ON STAGE GERMANY

Main supporter of Science on Stage Germany
think ING.
Die Initiative für Ingenieurnachwuchs

Proudly supported by
SAP