An Arduino Family Controller and its Interactions via an Intelligent Interface

M. Papoutsidakis
Dept. of Automation Eng.
Piraeus University of Applied Sciences,
Athens, Greece

A. Chatzopoulos
Dept. of Automation Eng.
Piraeus University of Applied Sciences,
Athens, Greece

C. Drosos
Dept. of Automation Eng.
Piraeus University of Applied Sciences
Athens, Greece

K. Kalovrextis
Dept. of Computer Science,
University of Thessaly,
Lamia, Greece

ABSTRACT
Arduino is an open source computer hardware and software company, project community and its users, who designs and manufactures microcontroller kits for the construction of digital devices and interactive objects that can be explored and controlled objects in the physical world.

The project is based on microcontroller board designs, manufactured by various vendors, using various microcontrollers. Tables have serial communication interfaces, including USB on some models, for personal computer loader programs. To program microcontrollers, Project Arduino provides a development-based integrated development environment (IDE) that includes support for programming languages C and C ++.

Keywords
Arduino, programming, app inventor, smart, home, sensors

1. INTRODUCTION
The first Arduino was introduced in 2005, aiming to offer a cheap and easy way for beginners and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such products designed for beginners include simple robots, thermostats, and motion detectors.

Arduino boards are commercially available in pre-assembled form, or as "your own" kits. The hardware design specifications are openly available, allowing the Arduino boards to be manufactured by everyone. The Adafruit industries are estimated to have produced more than 300,000 Arduinos commercially in mid-2011, and in 2013 700,000 official boards were in the hands of users.

Arduino programs can be written in any programming language, with a compiler generating binary machine code. Atmel provides a development environment for its microcontrollers, AVR Studio and the newest Atmel Studio.

The Arduino project provides the complete Arduino development environment (IDE), which is a cross-platform application written in Java. It comes from the IDE for the programming language processing project and for the cabling project. It is designed to introduce programming to artists and beginners who are not familiar with software development. It includes a code editor with features such as syntax highlighting, bracket matching, and auto recess, and provides a simple one-click mechanism for drawing and loading programs on an Arduino board. A program written with the IDE for Arduino is called "Sketch".

IDE Arduino supports programming languages C and C ++ using specific code organization rules. IDE Arduino provides a software library called "Wiring" from the wiring diagram, which provides many common input and output procedures. A typical Arduino C / C ++ sketch consists of two functions that are compiled and linked to a main () program in an executable circular execution program:

setup (): a function that runs once at the start of the program and can prepare the settings.

loop (): a repeat operation until the board goes out.

After compiling and connecting to the GNU toolkit, also included in the IDE distribution, IDE Arduino uses the avrdude program to convert executable code into a hexadecimal encoded text file placed on the Arduino board from a loader program to firmware of the board.

2. METHODOLOGY
As Arduino's webpage describes, Arduino is an "open source" electronic platform based on flexible and easy-to-use hardware and software designed for anyone with little programming experience, elementary electronic knowledge and interested in creating interactive objects or environments .

In essence, it is an electronic circuit based on ATmega microcontroller from Atmel, and all of its designs, as well as the software it needs for its operation, are distributed freely and free of charge so that it can be manufactured by everyone (from where the curious - for hardware - "open source"). Once built, it can behave like a tiny computer, since the user can connect multiple I / O modules on it and program the microcontroller to receive data from the input units, process them and send appropriate commands to the output units. Indeed, one could claim - and it would be a fairly successful parallel - that the Arduino functionally resembles the NXT Brick of the Lego Mindstorms NXT. Robotics is one of many applications in which Arduino excels.

Arduino, of course, is neither the single nor the best possible way to create any interactive electronic device. But its main advantage is the huge community that supports it and has created, maintained and expanded a similar sized online knowledge base. Thus, although an experienced electronist may prefer a different platform or components depending on the application he has in mind, Arduino, with extensive documentation, manages to win all those whose knowledge of electronics is limited to what little they have learned at school.

Just because it is mainly aimed at electronics novices and because, despite the very detailed instructions that exist, not all have the knowledge and the means to build an electronic board, ready-made, pre-fabricated Arduino boards can be purchased for about € 25. With a little extra money, most suppliers have the Arduino Starter Kit, which, in addition to Arduino itself, contains several other components and tools we may need for the first applications (such as the necessary

USB cable to connect to the computer , rasters, cables, LEDs, switches, potentiometers, resistors, diodes, transistors, etc.).

1. What does the Arduino board consist of?

Microcontrollers

Arduino is based on ATmega328, an 8-bit RISC microcontroller, which clocks at 16MHz. The ATmega328 has built-in memory of three types:

• 2Kb of SRAM memory, which is the useful memory that programs can use to store variables, tables, etc. during runtime. As with a computer, this memory loses its data when the Arduino power supply stops or resets.

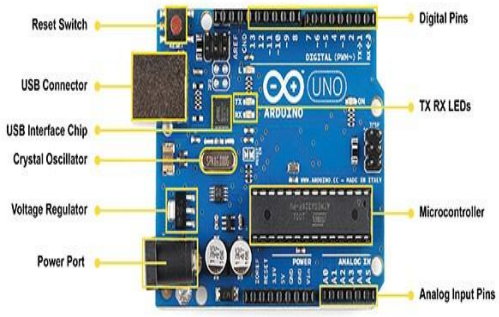
• 1Kb of EEPROM that can be used to "raw" write / read data (without datatype) per byte of the programs during the runtime. Unlike SRAM, the EEPROM does not lose its contents with power loss or reset, so it is the analog of the hard drive.

• 32Kb Flash memory, of which 2Kb are used by the Arduino firmware already installed by the manufacturer. This firmware, which is called bootloader in Arduino terminology, is necessary to install its own programs on the microcontroller via the USB port, without the need for an external hardware programmer. The remaining 30Kb of Flash memory is used to store exactly these programs once they are compiled into the computer. Flash memory, like the EEPROM, does not lose its contents with power loss or reset. Also, while Flash is not normally intended for runtime usage through programs, due to the small total memory available to them (2Kb SRAM + 1Kb EEPROM), a library is designed to allow the use of as much space as possible (30Kb less program size in compiled form).

Inputs - Outputs

First of all, Arduino has a serial interface. The ATmega microcontroller supports serial communication, which Arduino advances through a Serial-over-USB controller to connect to the computer via USB. This connection is used to transfer the programs designed by the computer to Arduino, but also for Arduino's two-way communication with the computer through the program while it is running.

In addition, at the top of Arduino there are 14 female pins, numbered from 0 to 13, which can act as digital inputs and outputs. They operate at 5V and each can supply or receive up to 40mA.



As a digital output, one of these pins can be programmed in HIGH or LOW mode, so Arduino will know whether or not power should be fed to that pin. In this way we can for example turn on and off an LED that we have connected to that pin. If we set one of these pin as a digital input through the program again, we can read its status (HIGH or LOW) with the appropriate command depending on whether or not the external device connected to this pin drives current to pin (this way, for example, we can "read" the state of a switch).

At the bottom of the Arduino, marked ANALOG IN, we will find another series of 6 pins, numbered from 0 to 5. Each of these functions as an analogue input using the ADC (Analog to Digital Converter) that is embedded to the microcontroller. For example, we can supply one of these with a voltage that we can range with a potentiometer from 0V to a Vref reference voltage which, if we do not make a change, is preset at 5V. Then, through the program, we can read the pin value as an integer 10-bit resolution, from 0 (when the pin voltage is 0V) to 1023 (when the pin voltage is 5V). The reference voltage can be set by a command at 1.1V or at any desired voltage (between 2 and 5V) by externally feeding this pin to the AREF pin on the opposite side of the board. So if we feed the AREF pin with 3.3V and then try to read some analog input pin to which we apply a voltage of 1.65V, Arduino will return the value 512.

Finally, each of these 6 pin, with proper command through the program, can be converted into a digital input / output pin such as the 14 on the opposite side and described previously. In this case the pins are renamed from 0 ~ 5 to 14 ~ 19 respectively.

Power and Led

The Arduino can be powered from the computer via the USB connection or from an external power supply via a 2.1mm socket (positive pole to the center) and located on the lower-left corner of the Arduino.

To avoid problems, the external power supply should be from 7 to 12V and may come from a common commercial transformer, from batteries or any other DC source. Next to the analog input pin, there is another 6-pin array with the POWER label.

On the Arduino board there is a micro-switch and 4 tiny LED surface mounts. The operation of the switch (labeled RESET) and the one LED marked POWER is rather obvious.

The two LEDs with the TX and RX markings are used as a function indicator of the serial interface as they light up when Arduino sends or receives data via USB. These LEDs are controlled by the Serial-over-USB controller and therefore do not work when serial communication is exclusively via the digital pin 0 and 1.

Finally, there is the LED with the L mark. The basic test of Arduino operation is to assign an LED to flash (we will see this again when we make the first application). Its builders thought of incorporating an LED on the board, which connected to the digital pin 13. Thus, even if one has not connected anything to the physical pin 13, assigning it the HIGH value through the program, it will light up this built-in LED.

2. Sensors

Sensors are sophisticated devices that are often used to detect and respond to electrical or visual signals. A sensor converts the physical parameter (for example: temperature, blood pressure, humidity, speed, etc.) into a signal that can be measured electrically. Let's explain the example of temperature. Mercury in the glass thermometer dilates and shrinks the liquid to convert the measured temperature that can be read by an atom to the calibrated glass tube.

There are some features to consider when choosing a sensor. These features are: accuracy, the environmental situation -

usually has temperature / humidity limits, sensor limit range, calibration - Essential to most of the measuring devices, as the way in which something is being read changes over time, resolution - Smaller increment is detected by the sensor, cost, and repeatability - Reading, which varies, is repeatedly measured in the same environment.

The sensors are classified according to main input quantity (measurable), Switching Authority (Using Physical and Chemical Effects), hardware and Technology, property and application

The switching authority is the key criteria for an effective approach. Typically, the material and technology criteria have been selected by the engineering development team.

3. SOFTWARE INTERFACE

App Inventor for Android is a visual programming interface originally provided by Google, and is now available and supported by MIT. Google App Inventor was first released to the public at the end of 2010, and after one year on December 31, 2011, Google finished the program's support and made it open source. App Inventor was re-released to the public, now called MIT App Inventor as a beta service (in March 2012), from MIT Center for Mobile Learning after the project was taken over by Google.

App Inventor is an on-line programming environment. In particular, for application development with App Inventor, it is necessary to use a web browser and a connected Android phone or a phone emulator.

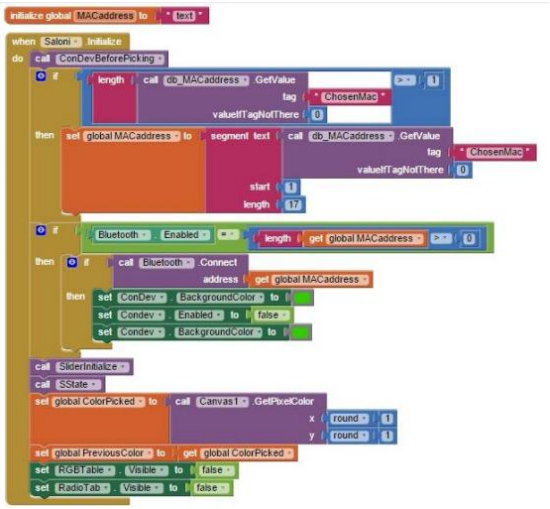
The App Inventor environment allows anyone who is familiar with computer programming to create applications for an Android operating system. It uses a graphical user interaction environment similar to Scratch and StarLogo TNG, which allows users to drag and drop tiles, creating applications that can run on Android, the operating system of today's many mobile devices.

The generated application is displayed step-by-step on the connected phone or on the phone emulator, as parts are added to it so that the application can be controlled as it is built. Upon completion of the app, it is possible to package the application so it can be installed on Android devices.

1. Programming of smart home in Arduino

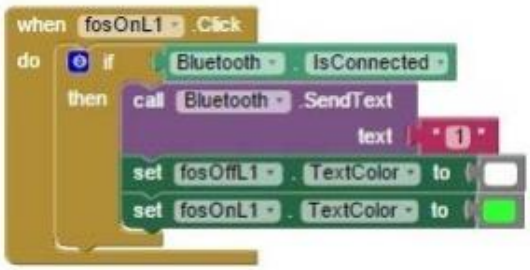
The logic is the following, sensors get data they send to Arduino, which in turn uploads them over Ethernet Shield to an HTTP Server. The Android app then reads this data from HTTP Server, and displays it.

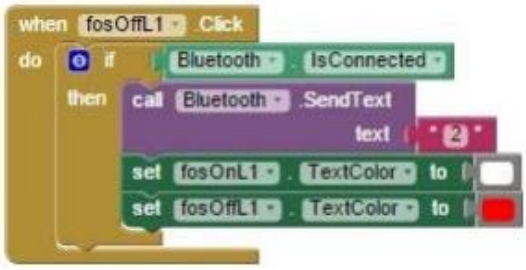
For the construction of the Smart Home application, the AppInventor platform was used, which in a simple and fast way allows us to design with the application interface as well as the code running from behind in a block diagram. In the following section the most basic pieces of the code are presented.



When entering "Lounge" the application searches the desired MAC of the WiFi in the database. It then checks if the mobile WiFi is connected to Arduino and greenens the corresponding interface icon, but initializes the entire interface of the application. When initiating the lounge, the slider buttons and buttons are displayed and hides the interface of the RGB board and HiFi, and if selected by the user, the unwanted texts and buttons will be hidden and hidden.

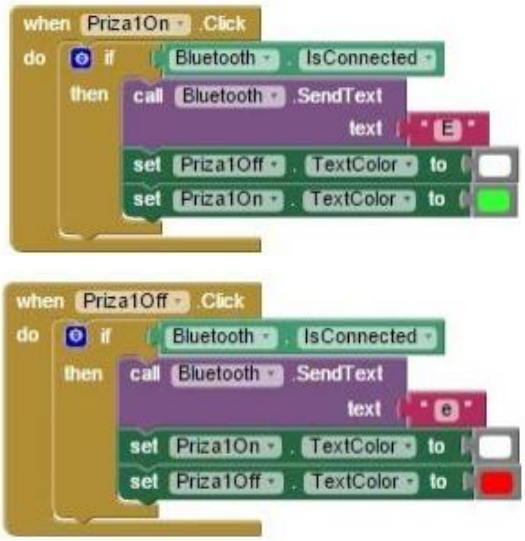
To enable the light, the following procedures are performed: Once the button is pressed the application checks if the WiFi is connected to the board if YES then sends the text "1" to the Arduino board and finally changes the color of the text to the interface instantly in gray and immediately afterwards in green. In order to turn off the light, the following procedures are performed: Once the button is pressed the application checks if the WiFi is connected to the board if YES then sends the text "2" to the Arduino board and finally changes the color of the text to the interface instantly in gray and immediately afterwards in red.





To enable a socket, the following procedures are performed: Once the on button is pressed the application checks if the WiFi is connected to the board if YES then sends the text "E" to the Arduino board and finally changes the color of the text to the interface instantly in gray and immediately afterwards in green. In order to disable a socket, the following procedures

are performed: Once the button is pressed the application checks if the WiFi is connected to the board if YES then sends the text "e" to the Arduino board and finally changes the color of the text to the interface instantly in gray and immediately afterwards in red.



The board we relied on is Arduino, for which we also needed its Ethernet Shield.

If Arduino is to be placed away from the home router (and we do not want to pull an ethernet cable) then we use an A / P which we set in Client Mode.

On every device (or socket / switch) that we want to be able to control from Android, we'll need to place a Relay, where we've used it. Of course, make sure that the relay current resistance you choose will exceed the needs of the device.

Other things we used: breadboard, jumper wires, LM35 temperature sensor, photocells, other sensors, resistors, insulators, multimeters, screwdrivers, etc.

In the simplest form of the Project, where we simply control 4 Android devices, we will cost 23.37 (Arduino) + 35.60 (Ethernet Shield) + 37.72 (4x9.43) (relays) + 10 (other cables, insulating tape, etc.) = 106.69 euro.

The whole process is divided into individual parts, on the one hand, to make it easier to set up the final platform and on the other hand to make it easier to manage. So the first step is to take the necessary hardware actions to connect the controller to the heat sensor and then to write the necessary code for heat recording in the space where the sensor is installed. The LM35 is a temperature sensor that has 3 pins, one V (in), one V (out), and one Ground.

4. ACKNOWLEDGMENTS

All authors would like to express their gratitude to the Post-Graduate Program of Studies “Automation of Productions and services” of PUAS, for the financial support to undertake this research project.

5. REFERENCES

[1] Brock, J. D., Bruce, R. F., & Reiser, S. L. (2009). Using Arduino for introductory programming courses. Journal of Computing Sciences in Colleges, 25(2), 129-130.

[2] Kato, Y. (2010). Splish: a visual programming environment for Arduino to accelerate physical computing experiences. In Creating Connecting and Collaborating through Computing (C5), 2010 Eighth International Conference on (pp. 3-10). IEEE.

[3] Evans, B. (2011). Beginning Arduino Programming. Apress.

[4] Noble, J. (2009). Programming Interactivity: A Designer's Guide to Processing, Arduino, and Openframeworks. " O'Reilly Media, Inc.".

[5] Buechley, L., & Eisenberg, M. (2008). The LilyPad Arduino: Toward wearable engineering for everyone. Pervasive Computing, IEEE, 7(2), 12-15.

[6] Monk, S. (2012). Programming Arduino. United States of America: McGraw-Hill Companies.

[7] Booth, T., & Stumpf, S. (2013). End-user experiences of visual and textual programming environments for Arduino. In End-User Development (pp. 25-39). Springer Berlin Heidelberg.

[8] Buechley, L., Eisenberg, M., Catchen, J., & Crockett, A. (2008). The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 423-432). ACM.

[9] Wolber, D. (2011). App inventor and real-world motivation. In Proceedings of the 42nd ACM technical symposium on Computer science education (pp. 601-606). ACM.

[10] Wolber, D., Abelson, H., Spertus, E., & Looney, L. (2011). App Inventor. " O'Reilly Media, Inc.".

[11] Gray, J., Abelson, H., Wolber, D., & Friend, M. (2012, March). Teaching CS principles with app inventor. In Proceedings of the 50th Annual Southeast Regional Conference (pp. 405-406). ACM.

[12] Morelli, R., De Lanerolle, T., Lake, P., Limardo, N., Tamotsu, E., & Uche, C. (2011). Can android app inventor bring computational thinking to k-12. In Proc. 42nd ACM technical symposium on Computer science education (SIGCSE'11) (pp. 1-6).

[13] Hsu, Y. C., Rice, K., & Dawley, L. (2012). Empowering educators with Google's Android App Inventor: An online workshop in mobile app design. British Journal of Educational Technology, 43(1).