Media Engineering and Technology Faculty
German University in Cairo

Vitual Arduino

Bachelor Thesis

Author: Ebtessam Zoheir
Supervisors: Georg Jung

Submission Date: XX July, 20XX

Media Engineering and Technology Faculty
German University in Cairo

Vitual Arduino

Bachelor Thesis

Author: Ebtessam Zoheir
Supervisors: Georg Jung

Submission Date: XX July, 20XX

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

---

Ebtessam Zoheir
XX July, 20XX

Acknowledgments

I would like to express my greatest gratitude to the people who have helped supported me throughout my project. I am grateful to my supervisor Georg Jung for his continuous support for the project, from initial advice in the early stages of conceptual inception through ongoing advice encouragement to this day.

A special thank of mine goes to my project partner Ahmed Sabbah who helped bring this idea to life.

I would also like to thank the Arduino developers community as they were extremely helpful and supportive.

Abstract

Virtual Arduino is a desktop application that simulates the Arduino board. It is an educational tool aims to provide hands-on experience for embedded systems programming and hardware beginners. This includes writing, verifying and uploading Arduino code. It should also include building hardware circuits to connect with the Arduino. The interactive simulator will help the user visualize the output of the code and the hardware failure scenarios make it more like real-life. These concepts give Virtual Arduino an edge over other simulators out there.

Contents

Chapter 1

Introduction

The use of microcontroller evaluation boards is becoming more popular year by year. The post by Oskay, W. on Why are Atmel AVRs so popular? on April 22nd 2010 gives insight on the importance of Arduino and similar AVR based boards. The fact that the Arduino is easily programmed in C, has built-in Analog Digital Converter (ADC), Pulse Width Modulation (PWM) and has good cross-platform support makes it a very efficient way to learn and practice embedded system programming.

Given the importance of AVR microcontroller boards, acknowledging the fact that not all students have access to such evaluation boards we have decided that a simulator which could virtually replace the Arduino experience would be useful. Already existing simulators do not provide close to real life experience and hence cannot replace the actual Arudino. Differences between the Virtual Arduino project and already existing ones will be discussed later in the paper.

Our aim is to make a real time hardware simulator that is as close as possible to the real life experience. The simulator will later achieve this by covering all aspects of an embedded system including hardware components, hardware interfacing, hardware failure, board setting, code compiling and uploading and finally visualizing the whole process. Another advantage is that it will give the user a wider variety of hardware components to work with that may not be available or may be too expensive. The simulator library will be very flexible and easily scalable to include future components and newer versions of the boards.

Chapter 2

Background

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software, retrieved from Arduino.cc. The Arduino board is designed around the 8-bit Atmel AVR microcontroller. Its software is composed of a standard C-based programming language compiler and a boot loader that executes on the microcontroller.

2.1 Program Specs

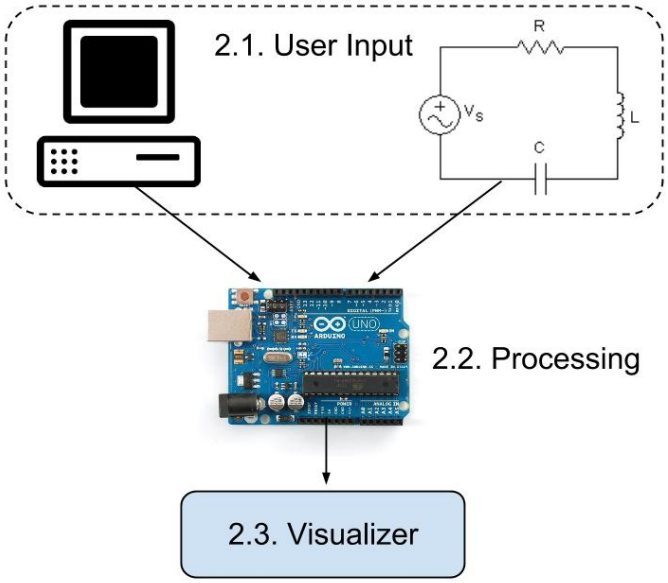

Figure 2.1: Virtual Arduino Architecture

The following section will discuss the architectural hierarchy of the full program. The section will also partition the program into modules so it can be more easily conquered.

2.1.1 User Input

The architecture is divided into three main parts, first part in the user input which can be in form of text i.e Arduino code or in the form of circuitry.

Computer Module

Real-life Arduino system can be divided into two parts, Integrated development environment (IDE) and hardware. The IDE or Computer Module is where the user can write, compile and upload code onto the Arduino. This shall be implemented by using the already existing opensource Arduino IDE and redirecting its output to a virtual serial port so that this binary code can be later used for the simulation. This binary code will be transimitted down from user input to the processing module. Details about the computer mode internal workings shall be dicussed in sections 2.2 amd 2.3.

Circuitry Module

The second part of the Arduino experience is external hardware. One of the main advantages of the Arduino board is that it is easily interfaceable with most hardware components as it has built-in ADC and PWM. This module is where the user gets to experiment with hardware and connect with the Arduino board. The circuitry workspace should include an easily scalable hardware library and a circuit builder. The hardware components are divided into 3 types which are input, connectors and output. Input components are mainly sensors, connector components are wires and resistors and output components are LEDs and motors. We plan on implementing this part by using and already existing opensource simulator and upgrading its GUI and its hardware components to fit the concept of the Virtual Arduino. This will be done by adding some hardware failures scenarios and giving the simulator a more realistic GUI. The circuit output or simplification if I may say so will also be sent to the next level of the architecture for further processing and linking with Computer Module output.

2.1.2 Processing Module

This module is the junction between the upper and lower level of the architecture. It takes the binary code from the Computer Module and translates it into actions in the Virtual Arduino processor. After evaluating the different internal components of the processor the Virtual Arduino output pins shall be assessed and this will reflect on the different output hardware components. The status of the different hardware components - whether high, low or PWM signal- shall be sent to the next level of the architecture which is the Visualizer.

2.1.3 Visualizer

This is the part where the user sees the output of his program. After the code is verified and is clear of any syntactic errors the user uploads the code onto his/her Virtual Arduino and output components start to show changes. The user will also be able to alter the environment by changing the input to the sensor and see the immediate change in the output components.

2.2 Bootloader

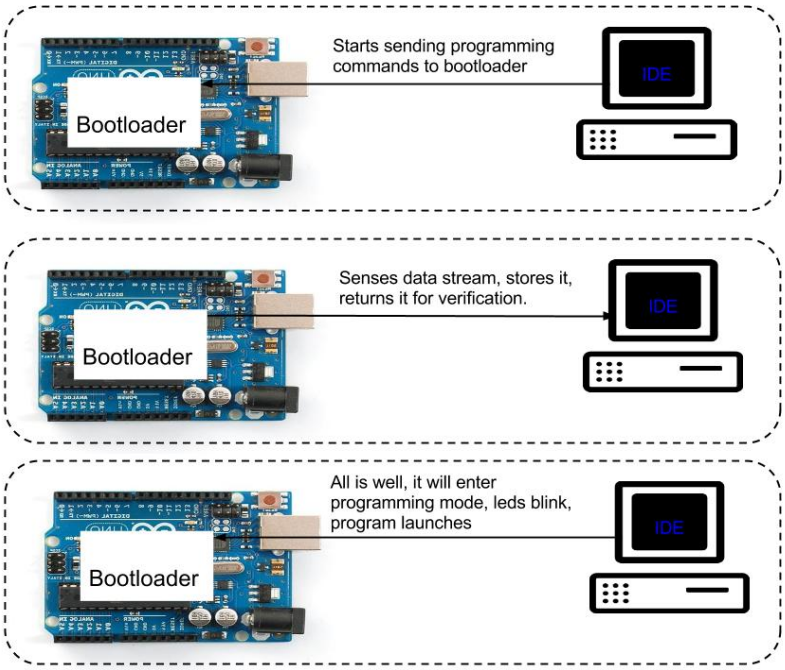

Figure 2.2: Bootloader-Programmer Handshake

Stackoverflow [2] user *angelatlarge* commented on April 3, 2013 (Upload Arduino code on virtual serial port through Arduino IDE):

> "The process of uploading code is not a uni-directional process. There is a program on the Arduino called a bootloader [*later discussed in section 2.2, the author*] which is responsible for communicating with the programmer ("programmer": a program that programs the Arduino, assume it is the Arduino IDE for now). The Arduino CPUs cannot be programmed across serial lines. Rather these chips are programmed either via the in system programming (ISP) or via the JTAG protocol.

Retrieved from Stackoverflow. Angelatlarge added (2013, April 3).Upload Arduino code on virtual serial port through Arduino IDE

> "The bootloader is a program that runs on the Arduino CPU, the program runs at startup and looks for programming commands over the serial port. If it discovers that a programmer is trying to communicate programming information, it will read the compiled Arduino binary coming over the serial link, store it in flash memory, send it back over the serial link for verification, and if everything is successful, exit and launch the stored sketch. If no programming information appears on the serial port, that is, no programmer is trying to write a new sketch, then the bootloader simply quits and launches the program already stored in flash.

Retrieved from http://stackoverflow.com/questions/15785087/upload-arduino-code-on-virtual-serial-port-through-arduino-ide/15792961?noredirect=1comment22546153$_1$5792961.

Each Arduino board uses a different bootloader. In our program we will be simulating the ArduinoUno which uses the optiboot bootloader (publicly available on GitHub [1]). Our plan is to mimic the responses sent from the bootloader to the IDE in order to get the code the is sent from the IDE to the flash on the Arduino board. The bootloader communicates with the IDE using the STK500 protocol. The STK500 is a communication protocol of 8-bit AVR. The handshake in Figure 2.2 will also be discussed in further details in the next chapter.

2.3 Communication

In order to imitate the bootloader we had to sniff the serial port to understand what is being transmitted between the IDE and the board. First, the Programmer in the IDE tries to sync with a device so it sends an GETSYNC command ([30] [20]). If a device is connected, the bootloader replies with INSYC, OK ([14] [10]). See Figure 2.3. After establishing the connection, the Programmer asks the bootloader about some of the board's specs such as the hardware, firmware, system clock and reference voltage. After setting the parameters, the IDE sends an ENTERPROGMODE command which tells the bootloader to program the chip. See Figure 2.4. The code is sent from the IDE to the board and is written to the flash memory. But before programming the chip, this code has to be verified. The bootloader then sends back all what is on the flash for verification. After it is verified, the chip is programmed and the IDE tells the bootloader that it is done and sends a LEAVEPROGMODE command. See Figure 2.5.

2.4 Java Libraries

To receive the Programmer's commands and respond to them like the bootloader, we had to communicate with our virtual serial port. There are several communication libraries in Java that offer such features. Sun had a serial communication API called JavaComm but they withdrew it's support for windows in 2005. This led to the development of the free open-source RxTx library. We chose to use the Java Simple Serial Connector (JSSC) library which is based on the RxTx library.

```
Is anyone home? Is anyone home? Is anyone home? (STK_GET_SYNC, CRC_EOP)

avrdude: Send: 0 [30]   [20]

avrdude: Send: 0 [30]   [20]

avrdude: Send: 0 [30]   [20]

Next two lines says yes (this is the boot-loaders do nothing response and says you are talking to it)

avrdude: Recv: . [14] (STK_INSYNC, STK_OK)

avrdude: Recv: . [10]
```

Figure 2.3: SYNC with device

```
Write data (0x64, STK_PROG_PAGE)

avrdude: Send: d [64] . [00] . [80] F [46] . [0c] . [94] a [61] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00] .
[0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e]
. [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] . [9a] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] .
[94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00] . [0c] . [94] ~ [7e] . [00]
. [0c] . [94] ~ [7e] . [00] . [00] . [00] . [00] . [00] $ [24] . [00] ' [27] . [00] * [2a] . [00] . [00] . [00] . [00] . [00] % [25] . [00] ( [28] . [00] + [2b] . [00] . [00] . [00] .
[00] . [00]   [20]

avrdude: Recv: . [14]

avrdude: Recv: . [10]
```

Figure 2.4: First Chunk of Code

```
######avrdude: Send: U [55] . [00] . [02]   [20]

avrdude: Recv: . [14]

avrdude: Recv: . [10]

avrdude: Send: t [74] . [00] . [02] F [46]   [20]

avrdude: Recv: . [14]

avrdude: Recv: . [ff] . [cf]

avrdude: Recv: . [10]

# | 100% 0.38s


avrdude: verifying ...

avrdude: 1026 bytes of flash verified


Quit (0x51 STK_LEAVE_PROGMODE)

avrdude: Send: Q [51]   [20]

avrdude: Recv: . [14]

avrdude: Recv: . [10]
```

Figure 2.5: Verifies, End connection

Chapter 3

Conclusion

Conclusion

Chapter 4

Future Work

Text

Appendix

Appendix A

Lists

List of Figures

Bibliography

[1] GitHub cloud repository server, public projects. http://github.com.

[2] Stack-overflow help forum web site. http://stackoverflow.com.