GAINER

A reconfigurable I/O module and software libraries for education

Shigeru Kobayashi, Takanori Endo, Katsuhiko Harada, Shosei Oishi
IAMAS
3-95, Ryoke-cho, Ogaki City
Gifu, Japan
{mayfair, endo, harada03, shosei05}@iamas.ac.jp

ABSTRACT

How to teach creating musical interface or installations to students who don't have backgrounds in electronic engineering is a longtime issue. This paper describes how it is taught at IAMAS (an institute dedicated to media arts) with the use of the newly developed environment, 'GAINER.' The GAINER environment consists of a reconfigurable I/O module and software libraries for common programming environments.

Keywords

learning, reconfigurable, rapid prototyping, sensor interface

1. INTRODUCTION

In 2004, we held a one-month workshop about physical computing[1] (mainly focused on creating musical interfaces and installations) using commercially available I/O boards, a few readymade sensors and actuators (one set lent to each student). The participants were six students who wanted to acquire skills to create interfaces or installations. Most of them had had some soldering experience (in junior high) and some programming experience (Max/MSP and Java), but little knowledge of electronics. During the workshop, we found the following difficulties:

- The participants feel a big gap between using readymade sensors or actuators and using bare sensors or actuators.
- The participants tend to break I/O ports by accident (i.e. electric overload). For most commercial products, repair parts are provided. But, it's difficult to purchase repair parts from foreign countries, so I/O modules tend to be left partly broken.
- Since configuration (e.g. number of analog inputs, PWM outputs) is fixed, the participants often find difficulty connecting sensors or actuators (e.g. the participant wants to connect a full color LED that requires three PWM outputs, but the module has only two PWM outputs). It is possible to solve this issue using proper external components (e.g. a multiplexor), but not so easy for a participant who does not have enough knowledge about electronics.
- To handle small signals from sensors (e.g. accelerometer), the participants have to add an external amplifier. It's difficult to design a proper amplifier circuit for a participant who has no experience.
- The price of the I/O board was reasonable as a commercial product, but it was a little bit expensive for students. Most of them became interested in the I/O board, but non purchased one.

As a result, the quality of the final projects (a prototype of an interface or an installation) was not satisfactory: Only one of the six students could create a prototype of an installation, the others remained at experimental level.

Based on the experience of the workshop, we started development of an environment (GAINER) for both education and actual interfaces and installations. Around that time, we decided that PSoC microcontrollers from Cypress Semiconductor would be a key component in our new I/O board[2]. The PSoC microcontroller is a mixed signal array that has configurable analog and digital blocks[3]. This flexibility allows the user to make their own configuration within the limitations of the hardware, and can change from one configuration to another on the fly. And the analog blocks have programmable gain amplifiers. This microcontroller is the key component of our I/O board.

2. CONCEPTS

Key concepts of the GAINER are as follows:

- The user starts with bare components and a solderless breadboard. This combination is used in real applications.
- The user builds their own I/O module from components by soldering to acquire basic techniques of electronics work and keep the cost of the I/O board as low as possible.
- The user can replace a broken microcontroller themselves.
- The user can choose from various configurations to suit their needs.
- The user can easily utilize a programmable gain amplifier to amplify small signals from sensors.
- The user can easily handle an I/O module with both graphical (i.e. Max/MSP[4]) and code-based (i.e. Processing[5]) programming environment .

NIME06, June 4-8, IRCAM, Paris, France

- All I/O ports are directly connected to the I/O pins of a microcontroller. The user can enhance capabilities of I/O ports with the use of 'bridge' modules if needed.
- Open source hardware and software: Advanced users can modify existing hardware to create a new one for their project.

3. RELATED WORKS

As related works, Wiring[6], Arduino[7] and The CREATE USB Interface[8] have been proposed. GAINER different from these projects in terms of concept and implementation, but basically the same with regard to orientation. Verplank et al reported about a course on controllers at Stanford University[10], and D'Arcangelo reported about a cource on musical controllers at New York University[11]. Additionally, Lehrman et al. has proposed that creating digital musical instruments is effective in bridging the perennial gap between the arts and the sciences[9].

Wiring is an open project initiated by Barragán et al. that is a programming environment and I/O board. Arduino is a sister project to Wiring. The programming environments build on Processing, and an I/O board equips an AVR microcontroller from Atmel. Both I/O boards can be used to develop stand-alone interactive objects, or can be connected to software on a PC (e.g. Processing, Max/MSP).

The CREATE USB Interface is an I/O board project by Overholt. The I/O board equips a PIC microcontroller with built-in USB function from Microchip Technology. The user can purchase an I/O board at low cost, or purchase components and build themselves.

For Wiring, the user can purchase an I/O board at low cost, and for Arduino and the CREATE USB Interface, the user can purchase an I/O board at low cost, or purchase components and build themselves.

4. THE GAINER ENVIRONMENT

The GAINER environment consists of the following parts: A GAINER I/O module, 'bridge' modules (if needed), software libraries for programming environment on a PC (i.e. Max/MSP and/or Processing). Figure 1 shows the relationships between these components.

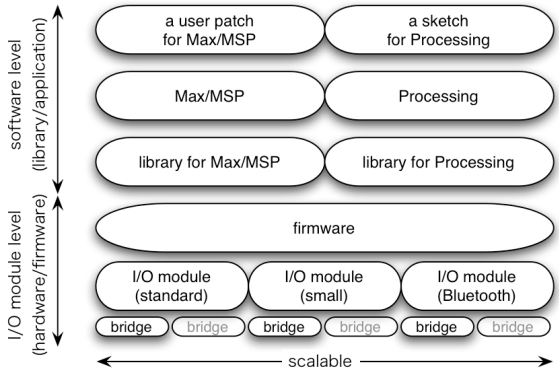

Figure 1: The structure of the GAINER environment. A user can combine a proper I/O module and a proper software platform as needed.

As mentioned in the key concepts, the user can choose from various configurations. Table 1 shows all configurations of GAINER. The first six configurations are versatile. For example, if the user chooses C4, they can connect eight potentiometers as input devices and eight LEDs with brightness control as output devices. On the other hand, the last two configurations are for specific purposes. For example, if the user chooses C8, the person can utilize four capacitive sensing switches by just connecting four electrodes and four resistors, and furthermore, the person can use an additional four digital inputs and eight digital outputs.

Table 1: All possible configurations of the GAINER I/O module. 'ain' stands for analog inputs, 'din' for digital inputs, 'aout' for PWM pseudo analog outputs and 'dout' for digital outputs.

config	ain	din	aout	dout	caption
C1	4	4	4	4	default configuration
C2	8	0	4	4	
C3	4	4	8	0	
C4	8	0	8	0	
C5	0	16	0	0	
C6	0	0	0	16	
C7	0	8	0	8	capacitive sensing
C8	0	0	8	8	matrix LED control

4.1 GAINER I/O module

4.1.1 Hardware: I/O module

Figure 2 shows the actual GAINER I/O module. The key components are a PSoC mixed-signal microcontroller (Cypress CY8C29466) and a USB-to-UART bridge (FTDI FT232RL). Except for these key components, the rest are standard and common components (i.e. a USB connector, LEDs, capacitors, resistors and so on). From a PC side, an I/O module appears as a serial port (38400bps, 1 stop bit, non-parity, no flow control).

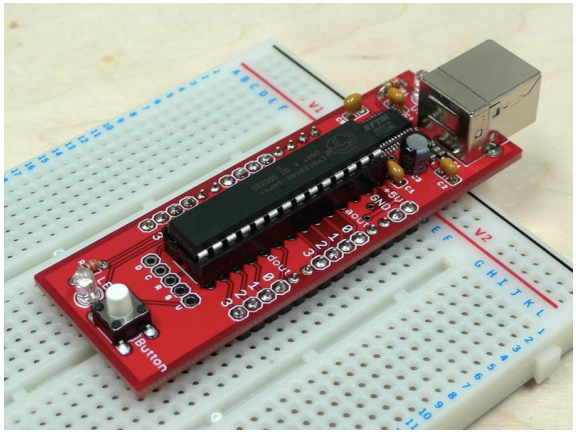

Figure 2: The GAINER I/O module. The module is placed on a breadboard with some components, and connected to a PC via a USB cable.

4.1.2 Hardware: 'bridge' modules

Figure 3 shows an actual 'bridge' I/O module to be combined with an I/O module. The right module is the 'powered

output bridge' for expanding the capability of the output ports (utilizing field emission transistors). For one or two ports, the user can substantialize the same function through use of discrete components on a breadboard. But it becomes difficult as the number grows since the space of a breadboard is limited. The user can create complicated (i.e. realistic) circuts on a breadboard through the use of 'bridge' modules.

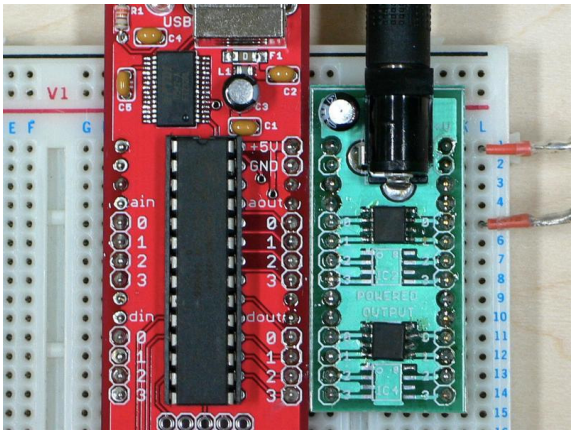

Figure 3: An examples of 'bridge' module: On the right is the 'powered output bridge.'

4.1.3 *Firmware*

While the hardware connection between the microcontrollers and the I/O ports is fixed, the configuration within the microcontroller is reconfigurable by the firmware. Figure 4 shows the internal configurations of 1 and 8. We designed specific hardware configuration and firmware for each. As shown, these internal configurations are totally different except for a UART module for communication. From the standpoint of the user, choosing or changing a configuration is a matter of simply selecting a 'patcher' (in the case of Max/MSP) or supplying an argument (in the case of Processing). Analog inputs are 8bit resolution, about 300sps speed (available in C1, C2, C3 and C4). Digital inputs are pulled-down internally. Analog outputs are pseudo analog (PWM) outputs. Both analog and digital outputs are set to 'strong' drive mode.

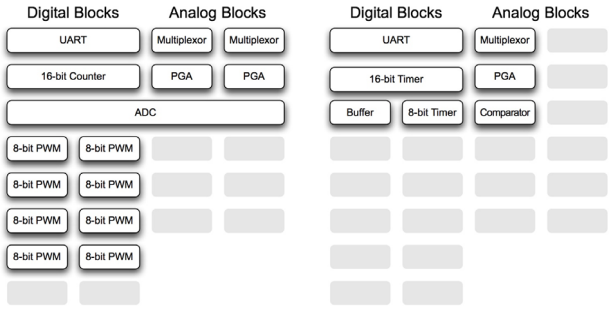

Figure 4: Examples of configurations: The left one is configuration 1 and the right one is configuration 8. The user can change from one configuration to another on the fly.

4.2 Software libraries

Figure 5 and 6 shows an example of a software library for Max/MSP. Currently, the software libraries for Max/MSP are provided as a 'patcher' (e.g. gainer.io.c1.pat) and a help patch (e.g. gainer.io.c1.help) for each configuration. As a start point, the user can use a help patch, or create their patch from scratch.

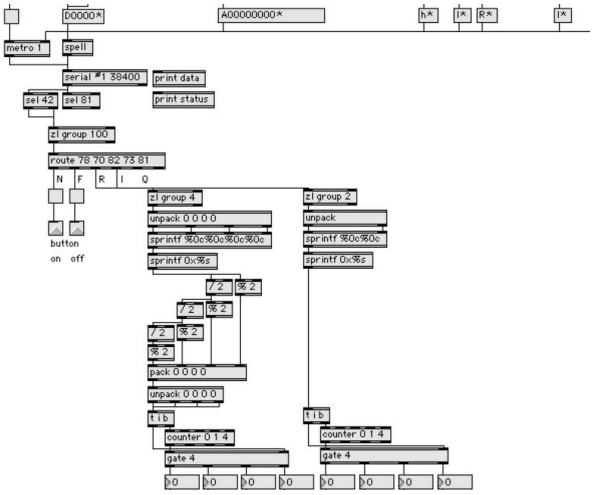

Figure 5: An example of a software library for Max/MSP. The library encapsulates low messages between a PC and an I/O board, representing them as higher level messages.

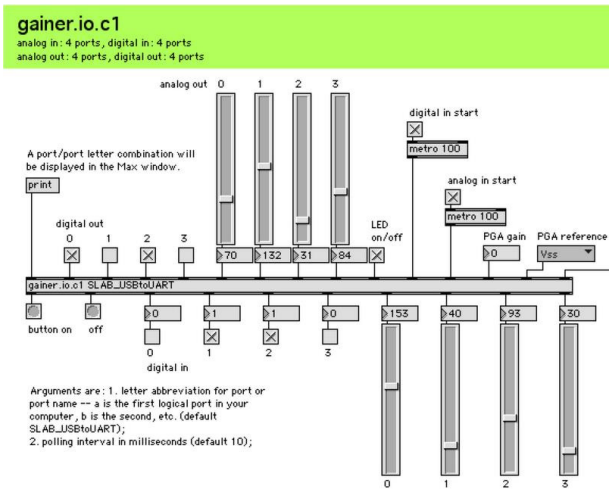

Figure 6: An example of a help patch. All connections in a configuration are displayed in a help patch. The user can start creating their own patch from a help patch.

Figure 7 shows an example of a 'sketch' in Processing environment, and figure 8 shows an example of a reference document of a class for Processing. All messaging from/to a GAINER I/O module are implemented as methods of the 'Gainer' class. A user can easily communicate with a GAINER I/O module via an instance of the Gainer class (e.g. gainer.setHigh(0)).

Figure 7: An example of a 'sketch.' Since all I/O module functions are represented as methods of the 'Gainer' class, a user doesn't have to deal low messages from/to an I/O module.

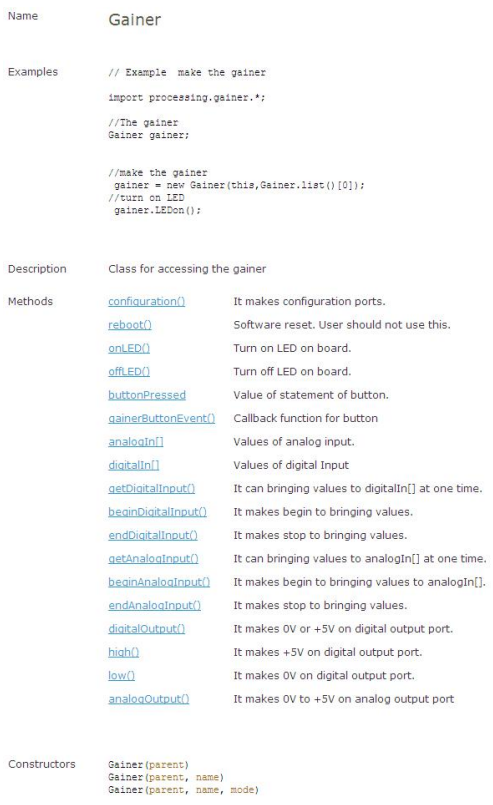

Figure 8: An example of a reference document. A user can easily open reference documents to see detailed descriptions.

5. CURRICULUM

We held a workshop on physical computing from November 8th to December 5th at IAMAS in 2005. IAMAS consists of a specialized training college (International Academy of Media Arts and Sciences) and a graduate school (Institute of Advanced Media Arts and Sciences). The participants numbered ten (eight from the specialized training college and two from the graduate school). All of them were interested in creating interfaces and/or installations. All of them had a certain degree of experience of programming (e.g. Max/MSP and/or Java), but none of them had knowledge about electronics. We had three hours per day, two days (on Mondays and Thursdays) per week.

The schedule of the workshop was as follows:

- week 1, day 1: Basic electronics
  - How to use a breadboard?
  - Ohm's law
  - Turn on a LED with a resistor
  - Ordering components of your I/O module
- week 1, day 2: Building your I/O module
  - How to do soldering?
  - Building your I/O module
  - Testing the I/O module
- week 2, day 1: How to handle outputs?
  - How to use software libraries for Max/MSP?
  - How to connect a LED?
  - How to connect a SSR?
  - How to connect a R/C servo motor?
- week 2, day 2: How to handle inputs?
  - How to connect a switch?
  - How to connect a potentiometer?
  - How to connect a CdS?
  - How to connect an accelerometer?
- week 3, day 1: How to process data in Max/MSP?
  - How to do scaling in Max/MSP?
  - How to do data processing in Max/MSP?
  - How do you map incoming data to outputs?
- week 3, day 2: What is Processing?
  - Basic introduction about Processing
  - How to use software libraries for Processing?
  - How to handle inputs?
  - How to handle outputs?
  - How to process data in Processing?
- week 4, day 1: Plan presentation
  - Show a plan for the final presentation
  - Discussion with lecturers
  - Order components and materials
- week 4, day 2: Building
  - Build a prototype
- week 5, day 1: Final presentation
  - Present one's prototype
  - Discussion
  - Closing

Figure 9 shows a scene of the second day of the first week of the workshop.

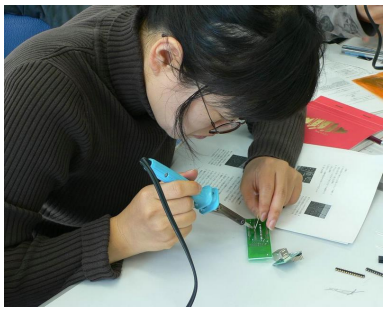

Figure 9: A student is soldering to assemble her own I/O module. Most participants had some soldering experience in junior high.

6. RESULTS

6.1 Winter 2004 student projects

At the final presentation, all participants presented as described in Table 2. Although the preparation period was short (about one week), most students presented a working prototype as shown in Figure 10. Additionally, a few students actually used prototypes in their musical performances or installations in the weeks following after the final presentation. With regard to configuration, eight students used the default configuration (C1), and two students used another configuration (C4) as needed.

Table 2: The breakdown of the final presentation

type	number
musical interface	6
installation	3
performance	1

Figure 10: A student is presenting his prototype to all participants.

Figures 11, 12 and 13 show examples of the final projects in musical interface category. The first one is a musical interface consisting of RGB color sensors and a full color LED. The RGB color sensors on the bottom face recognizes a color, then PC side software turns the color information into sounds. An I/O module is used to handle the sensors and the LED. The second one is a musical interface consisting of pressure sensors and a CCD camera-based computer vision. A performer plays the instrument with their hands, and changes parameters by pressing on the sensors. An I/O module is used to handle pressure sensors. The third one is a musical interface consisting of CdS sensors and LEDs under a half-mirror. A performer plays the instrument with a bulb light. When a sensor detects the light, a corresponding LED is turned on and a sound is played for a short while. An I/O module is used to handle sensors and LEDs.

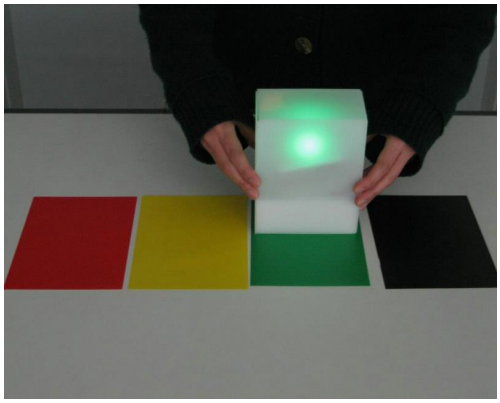

Figure 11: Input is color information from a RGB color sensor. Output is a light from a RGB color LED and corresponding sounds from a PC.

Figure 12: Input is color information from a CCD camera and pressure information from pressure sensors. Output is visual feedback on a projected screen and corresponding sounds from a PC.

Figure 13: Input is brightness information from CdS sensors. Output is lights from white LEDs and corresponding sounds from a PC.

6.2 The results of questionnaires

We asked students to answer questionnaires several times throughout the workshop. First of all, the result of "Was the workshop of interest to you?" was 4.44 (5 was "very well" whereas 1 "not well.", SD = 0.96). According to this result, the workshop was of interest to most of the students.

Secondly, the result of "Did you easily assemble your I/O module?" was 3.33 (SD = 0.80). According to this result, the degree of difficulty was thought to be reasonable.

And the result of "Did you become interested in the GAINER after assembling by yourself?" was 3.78 (SD = 1.03). Except for two students, the score was 4 or 5. The two students who scored 2 to this question gave supplemental answers to the question as follows: "Since no interesting real applications are presented currently, I'm not very interested." This was one of the points of reflection about the curriculum of the workshop.

Q1 in Table 3 shows understanding of electronics before and after the workshop. According to the result, the understanding of electronics seems to have been deepened through the workshop.

Q2 in Table 3 shows understanding of programming before and after the workshop. According to the result, there was no change about the understanding of programming. We hoped that the understanding of programming would increase as it did in electronics, but it didn't. We think that there was not enough time for most of the students to understand both electronics and programming simultaneously.

Q3 in Table 3 shows interests in creating works with electronics. According to the result, the score is mostly same before and after the workshop. Before the workshop, the students were highly motivated. But during the workshop, they experienced many difficulties in creating their final projects. In spite of the difficulties, they remained highly motivated throughout the workshop. We think that the workshop was meaningful for the students.

Table 3: An excerpt of questions: Q1 was "How well do you understand electronics?" Q2 was "How well do you understand programming?" Q3 was "Do you want to create works with electronics?" The numbers shown in parentheses are SD values.

	before	after
Q1	1.67 (0.47)	2.78 (0.91)
Q2	2.72 (0.85)	2.72 (0.85)
Q3	4.44 (0.68)	4.33 (0.81)

7. CONCLUSIONS AND FUTURE WORK

According to the results of the workshop, we achieved some positive results with utilizing the GAINER environment. During the workshop, we couldn't explore possibilities of the reconfiguration in deep. According to the questionnaire after the workshop, 66% of the students wanted to attend an advanced workshop. We have no detailed plans for an advanced workshop, but we would like to explore possibilities of the reconfiguration in the workshop. In regard to scalability, we would like to provide a wider range of I/O modules and 'bridge' modules as follows:

- Documentation in multiple languages (currently in single languages only).
- Smaller I/O modules to be embedded into a small device (to expand scalability).
- Stand-alone capability (both firmware side and development tool side).
- Wireless connection capability (e.g. Bluetooth, partially tested).
- More software libraries (e.g. PureData, Adobe Flash and so on).

In regard to the effectiveness of the GAINER environment and the workshop, it's hard to do control experiments. So we want to keep holding the workshop for a few years to examine its effectiveness and to improve the environment.

8. ACKNOWLEDGMENTS

This research has been a part of the 'Programmable Device Project' (PDP)[12] at IAMAS. The authors wish to acknowledge of the assistance of Masayuki Akamatsu for suggestions as a media artist, Takahiro Kobayashi for technical advice, Kazuki Saita for reviewing the documentations and all members of the PDP and workshop attendees for cooperative suggestions. The source code of the firmware, the hardware and software libraries are available at the following location: http://gainer.sourceforge.net/

9. REFERENCES

[1] O'Sullivan, D. and Igoe T. Physical Computing. Muska LipmanPremier-Trade, 2004.

[2] Seguine, D. Just add sensor - integrating analog and digital signal conditioning in a programmable system on chip. Sensors, 2002. *In Proceedings of IEEE* Volume 1, 12-14, p.665–668.

[3] Ashby R. Designers Guide to the Cypress PSoC. Newnes, 2005.

[4] Cycling '74. Max/MSP. Available at: http://www.cycling74.com/

[5] Reas, C. and Fry, B. Processing: a learning environment for creating interactive Web graphics. *In Proceedings of the SIGGRAPH 2003.*

[6] Wiring. Available at: http://wiring.org.co/

[7] Arduino. Available at: http://arduino.berlios.de/

[8] Overholt, D. The CREATE USB Interface - where art meets electronics. Available at: http://www.create.ucsb.edu/~dano/CUI/

[9] Lehrman, P. and Ryan, T. Bridging the Gap Between Art and Science Education Through Teaching Electronic Musical Instrument Design. *In Proceedings of New Interfaces for Musical Expression* (NIME05) (Vancouver, Canada, May 26–28). p.136–139

[10] Verplank, B. A Course on Controllers. *In Proceedings of New Interfaces for Musical Expression* (NIME01) (Seattle, USA, April 1–2)

[11] D'Arcangelo G. Creating a Context for Musical Innovation: A NIME Curriculum. *In Proceedings of New Interfaces for Musical Expression* (NIME02) (Dublin, Ireland, May 24–26)

[12] Programmable Device Project. http://www.iamas.ac.jp/project/pdp/