Software Development for a Pediatric Gait Trainer: From LabVIEW VI to Arduino Sketch

Supachai Vorapojpisut

***Abstract*—This paper presents the rapid prototyping of a software system for a low-cost pediatric gait trainer. Major requirements are to control the speed of a DC motor according to gait sequence and to display the progress of gait training. Due to their advantages, LabVIEW and Arduino platforms have been selected the implementation on computer and embedded board, respectively. At first, the firmware of LIFA toolkit was programmed into an Arduino board to operate as an I/O board. A LabVIEW VI has been developed to study PWM patterns suitable for leg-pulling sequences. Then, an Arduino sketch has been customized to achieve the requirement of standalone operations by mapping their corresponding design patterns. The prototyped system was completed and demonstrated at the i-CREATe 2012 event.**

***Index Terms*—Pediatric gait trainer, design patterns, LabVIEW, arduino.**

I. Introduction

An effect of neurological and musculoskeletal disorders in children is walking (gait) abnormalities [1]. Without proper treating/training, such abnormalities cause pains and discomfort, have thus prevented children to walk in daily activities resulting in the weakness of supporting muscles. Gait training is an approach to let children learn how to walk safely and properly. Even pediatric gait training is usually done by rehabilitation specialists; further trainings at home with assistive devices under parental supervision can improve strength, balance, endurance, and coordination. However a major shortcoming of pediatric gait trainers in the market is the gap between automatic gait training systems (i.e., Lokomat) which are very expensive and manual-assisted gait trainers which require time and patience for each course (see Fig. 1).

Fig. 1. Pediatric gait trainers: automatic type and manual-assisted type.

Manuscript received June 15, 2014; revised August 25, 2014. This work was partially supported by the research fund of Faculty of Engineering, Thammasat University.

S. Vorapojpisut is with the Department of Electrical and Computer Engineering, Thammasat University, Pathumthani 12121 Thailand (e-mail: vsupacha@engr.tu.ac.th).

To fill such gap, researchers in the Medical Engineering Program, Thammasat University has developed a prototype of low-cost pediatric gait trainer for cerebral palsy children [2]. As the extension to manual-assisted pediatric gait trainers, the design relies on the reverse spindle-crank mechanism coupled to a geared DC motor. The gait training is realized by generating a PWM signal to drive the motor in 180-degree patterns. In addition to gait movement, other functional requirements are to display/record training data and to time stance/swing periods (see Fig. 2).

Fig. 2. The prototype of pediatric gait trainer.

Software features of the prototyped pediatric gait trainer can be classified into three groups including I/O interface, data management and GUI. Due to size, weight, and power supply constraints, the software system have to be realized on the integration of an embedded board and a computer. Arduino platform has been selected for the embedded board due to its advantages on simplified I/O programming and cost. LabVIEW platform has been selected for the development of GUI part due to its advantages on data acquisition, signal processing features, and user interface development. The combination of LabVIEW/Arduino platforms was recognized by several biomedical research projects such as heart rate variability [3], biosignal acquisition and processing [4], and medical device prototyping [5].

II. Software Platforms

A LabVIEW

LabVIEW platform [6] is a development environment based on the dataflow programming paradigm using a visual programming language. LabVIEW programs are called virtual instruments (VIs) consisting of three components: a front panel, a block diagram, and a connector pane. Front panel represents a user interface part, while a block diagram represents graphical source code that perform operations on data. Each object placed on the front panel will appear on the

DOI: 10.7763/IJET.2015.V7.846

back panel as terminals. The graphical approach allows developers to build VIs by dragging and dropping nodes representing inputs, outputs, structures, and functions into working area. Then nodes are connected using wires to transfer data (see Fig. 3).

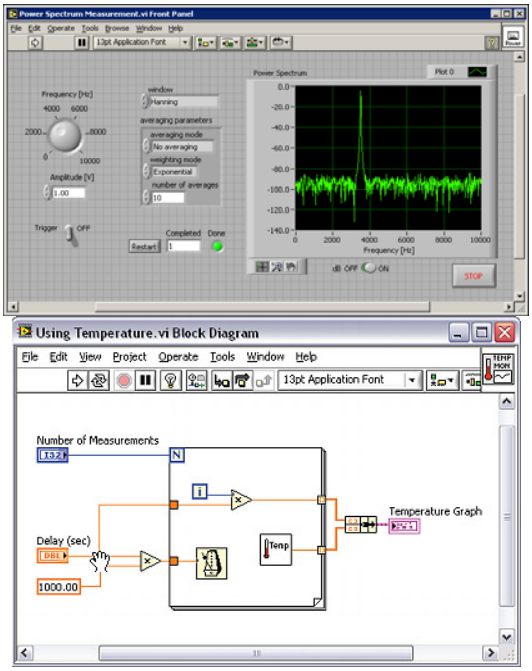

Fig. 3. LabVIEW front panel and block diagram.

LabVIEW development addresses measurement and data analysis requirements with the concept of *Acquire-Analyze -Present*. The acquisition part is to collect data from hardware and user interface. The analysis part is to perform mathematical and logical operations on the collected data. Then the presentation part is to display the analyzed data in meaningful ways.

B Arduino

Arduino [7] is an open-source electronics prototyping platform designed to provide inexpensive devices that interact with their environment using sensors and actuators. Arduino hardware is a single-board computer based on an 8-bit Atmel AVR, or a 32-bit Atmel ARM microcontroller. The standardized I/O interface consists of 6 analog input pins, and 14 digital pins which also support serial and PWM features as illustrated in Fig. 4.

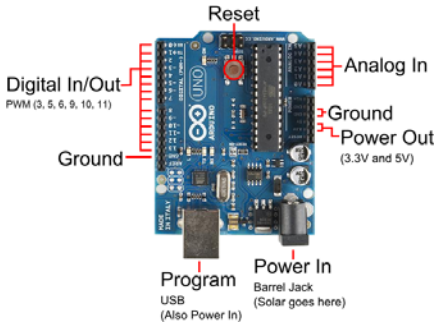

Fig. 4. Arduino Uno board with pin arrangement.

Using an integrated development environment (IDE) on computers, Arduino programs are written using C or C++. The code or sketch is built using cross toolchain and transferred to an Arduino board via its USB interface. The main advantage of the Arduino software platform is its "*Wiring*" library that abstracts the hardware access. Therefore I/O programming can be done without the knowledge of microcontroller.

Arduino platform has been extended in many forms regarding to its open-source license. The existence of many clone boards makes the price to be as low as $9. While the porting of Arduino APIs to other microcontrollers makes the code to be portable across different processor architecture. Such flexibility gives a benefit for using Arduino boards in cost-constrained applications.

C LabVIEW Interface For Arduino Toolkit

LabVIEW Interface For Arduino toolkit or LIFA [8] is a LabVIEW library to interface with Arduino boards. Using a sketch provided with the LIFA toolkit, an Arduino board can be used as a data acquisition (DAQ) hardware. Digital I/Os and voltage measurement with analog pins can be managed from within LabVIEW VI without any C/C++ code. However the pre-programmed Arduino board must be connected to the computer with LabVIEW through a USB link (see Fig. 5).

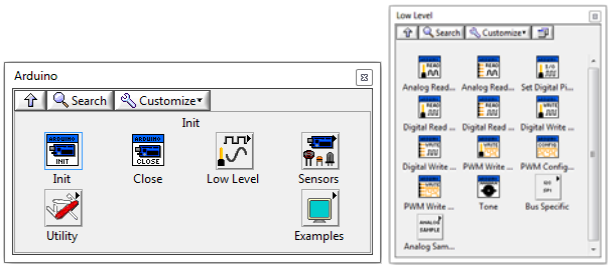

Fig. 5. Nodes available in the LIFA toolkit.

III. Design Patterns

Design patterns are general reusable solutions to commonly occurring problems within a given context in software design. As description or template, applying design patterns in software development provides simplicity and also reliability.

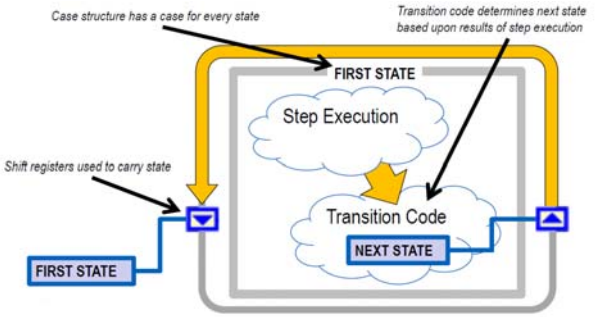

Fig. 6. Design pattern: state machine.

A LabVIEW Design Patterns

Since LabVIEW uses a graphical dataflow programming approach, design patterns are templates of interconnection among *structures*(while/case/for) and *nodes* as pre-existing solutions to common problems. There are five design patterns [9] suggested by National Instruments:

- Functional global variable,
- State machine,
- Event-driven user interface,
- Producer/consumer,
- Queued state machine.

Two design patterns considered in our development are *state machine* for PWM generation and *event-driven user interface* for interactive parameter management.

The state *machine* pattern uses a case structure within a while loop as shown in Fig. 6. Each case represents the system state with shift registers to pass the value of next states.

Fig.7 shows the *event-driven user interface* pattern based on a nested event structure within a while loop. The event structure will be blocked until receiving registered events or timeout.

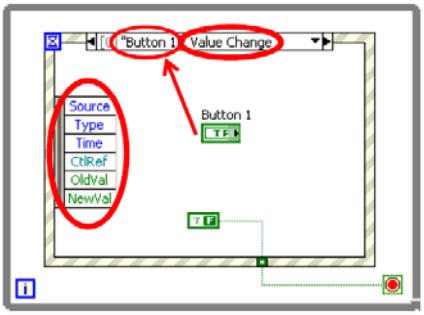

Fig. 7. Design pattern: event-driven user interface.

B Arduino Design Patterns

Design patterns for Arduino are similar to those [10] available to C/C++ development. Internally, the *adapter* pattern has been applied extensively to realize the hardware abstraction of I/O interfaces. Therefore Arduino API is almost unified across hardware platforms. The *state* pattern is widely used in applications that implement a command handler for serial protocol. For a simple command handler, the C-based pseudo-code as listed in Fig. 8 shows the structure of a hierarchical switch statement that processes received commands.

```
int state = FIRST_STATE;
void commandHandler() {
  while ( Serial.available() && c != '\n' ) {
    c = Serial.read();
    cmdbuf[i++] = c;
  }
  if ( c == '\n' ) {
    event = parseCommand(cmdbuf);
    switch (state) {
      case FIRST_STATE:
        state = firstStateHandler(event);
        break;
      ...
    }
  }
}
int firstStateHandler(int event) {
  int nextState = STATE0;
  switch (event) {
    case EVENT0:
      transitionCode();
      nextState = SECOND_STATE;
      break;
    ...
  }
  return nextState;
}
```

Fig. 8. State machine implementation for command handler.

The state pattern in Fig. 8 is analogous to the *state machine* pattern in Fig. 6. Therefore the code porting of a switch structure in a VI into a family of handler functions can be performed by mapping events and states into switch statements in C code.

IV. Software Development of Gait Trainer

The main concept is to partition software development process into three phases (Development-Implementation-Deployment) according to corresponding users. To shorten delivery time, the development process follows the evolutionary prototyping approach in which software artifacts of each phase are reused as the starting point of next phase.

A Software Requirements

Software requirements are formulated based on the main user of software system in each phase. In the Development phase, the main user is the research team who is responsible for the investigation of suitable profile parameters. Therefore the main requirement is to develop an interactive UI part for adjusting and generating PWM patterns.

In the Implementation phase, the main user becomes a selected group of rehabilitation specialists who will feedback their experience and suggestion on the prototype gait trainer. The main requirement of this phase is to make the prototype embedded board running stand-alone, while PC software is used for monitoring purpose via wireless communication.

Finally the main user of the Deployment phase will be staffed at the rehabilitation center and children families. Two required features for each user group are data collection from multiple gait trainers and on-board UI respectively.

B Software Design

By assisting each walk with two strings attached with each knee, the major concern of firmware development is the safety of children from improper string pulling force that is generated by constant and fixed PWM signals. To overcome this concern, we have proposed the generation of motor driving signals based on a four-modes PWM profile as depicted in Fig. 9.

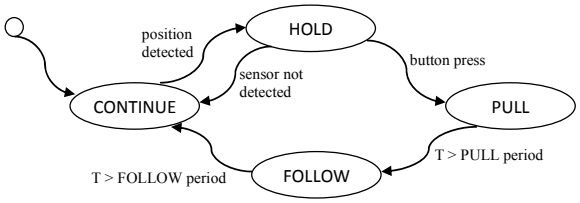

Fig. 9. State machine of the proposed PWM profile.

PULL mode: duty cycle is increased to the maximum value (*PULL PWM*) within a given period (*PULL period*). The objective of this mode is to apply enough pulling force to raise the child's foot off the ground, hence assists the propulsion in stance phase.

FOLLOW mode: duty cycle is decreased from the maximum value to a value (*FOLLOW PWM*) within a specified period (*FOLLOW period*). This mode matches the

toe-off in swing phase in which leg motion is less resistance to pull strings.

CONTINUE mode: duty cycle is maintained at a value (*CONTINUE PWM*) until marked crank position is detected, i.e. the presence of heel strike in the swing phase. Since both strings are slack, the objective of this mode is to complete the swing phase as fast as possible.

HOLD mode: duty cycle is maintained at the lowest level (*HOLD PWM*) enough to hold the tension within the string. Therefore, self-walk situations can be detected by checking the status of crank position sensor. To assist a walk, a button is pressed to advance into the PULL mode.

Due to the difference in weight and height of child patients, six parameters can be customized for each patient, namely PULL PWM, PULL period, FOLLOW PWM, FOLLOW period, CONTINUE PWM, and HOLD PWM.

V. Rapid Prototyping of Software System

At first, the software system was developed using the LIFA toolkit consisting of a LabVIEW library and a source code of Arduino firmware. The usage of LIFA toolkit relies on the LabVIEW programming model to communicate a series of commands synchronously with the Arduino firmware. In other words, Arduino board is just an I/O board directly managed by VIs.

Based on the requirement of the Development phase, we have developed a VI consisting of a UI and a control loops to generate PWM signal via an Arduino Mega 2560 board interactively. Since no work is needed on the Arduino side, the first revision of working software was finished within one week with some limitations: on-screen button and wired communication via USB cable as shown in Fig. 10. Fig. 11 shows the overview of two loops that handles UI and controls the speed of DC motor, respectively. Then the working software has been experimented with some children at a rehabilitation center to find an appropriate set of parameters.

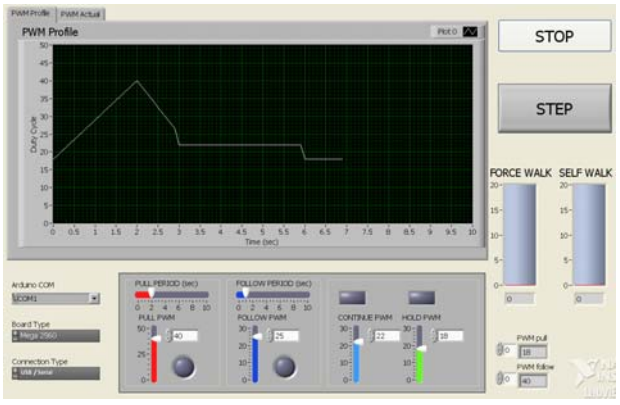

Fig. 10. User interface in the Development phase.

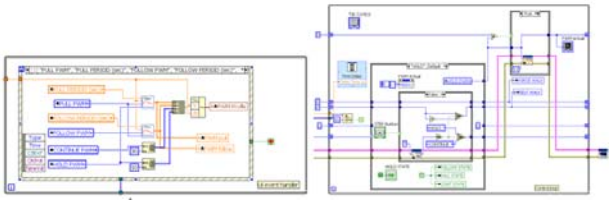

Fig. 11. Two major loops of the VI: UI handler and control loop.

In the second phase, the operation of LabVIEW control loop has been ported into a customized Arduino firmware such that the prototype gait trainer can be used independently of the PC. Two matched XBee modules were selected to realize wireless communication between the PC and the Arduino board. Since the control loop in the original VI is implemented based on the *state machine* pattern, the code porting has been performed by mapping the logic depicted in Fig. 9 into corresponding event and state enumerations.

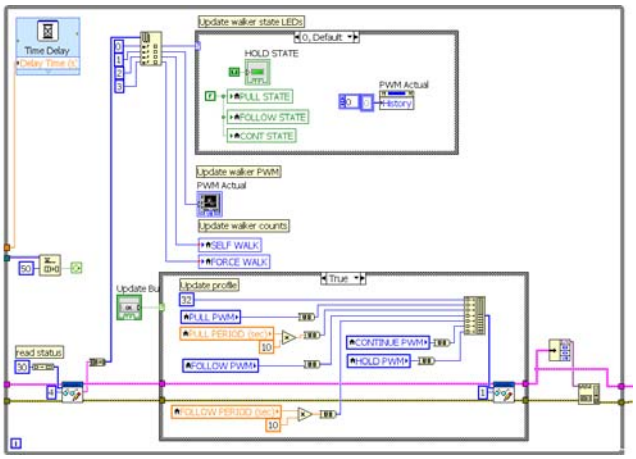

Fig. 12. New monitoring loop of the Implementation phase.

Based on the original LIFA firmware, three additional commands (read profile, update profile, and read status) have been implemented to support the UI-based parameter configuration, while the VI control loop was replaced by a monitoring loop as illustrated by Fig. 12.

Following the concept of evolutionary prototyping, almost of a UI portion of the PC is not modified except for the replacement of "STEP" button with the "UPDATE" button. The software transition was also finished within one week with respect to the reuse of UI portion and the command loop in LIFA firmware. At present, we are working on the extension of the VI to support the management of multiple gait trainers.

VI. Conclusion

This paper concludes our experience on the development of LabVIEW VIs and an Arduino sketch for a prototyped pediatric gait trainer. The main objective is to develop a software system to control the operation of the gait trainer based on two major requirements; multi-mode speed control and UI for clinical tests. Based on the *state machine* pattern, LabVIEW VIs has been developed to realize UI and control features. Then the control loop portion has been ported to a customized Arduino sketch via the mapping into the *state* pattern. By matching design patterns, the development of both LabVIEW VIs and Arduino sketch was rapidly prototyped within a very short period.

References

[1] C. Nieuwenhuijsen, M. Donkervoort, W. Nieuwstraten *et al*, "Experienced problems of young adults with cerebral palsy: Targets for rehabilitation care," *Archives of Physical Medicine and Rehabilitation*, vol. 90, pp. 1891-1897, 2009.

[2] M. Jirojananukun and B. Rungroungdouyboon, "Design assistive stepping for posterior walker in cerebral palsy children," presented at

the 22nd Conf. of The Mechanical Engineering Network in Thailand, ChiangMai, Thailand, 2012.
[3] M. Schiavenato, C. Oliu, E. Bello, J. Bohorquez, and N. Claure, "Development of a system for the assessment of heart rate variability in the NICU," in *Proc. the 29th Biomedical Engineering Conference*, Miami, 2013, pp. 25-26.
[4] H. P. da Silva, A. Lourenço, A, Fred, and R. Martins, "BIT: Biosignal igniter toolkit," *Computer Methods and Programs in Biomedicine*, vol. 115, pp. 20-32, June 2014.
[5] C. Loncaric, Y. Tang, C. Ho, M. A. Parameswaran, and H. Yu, "A USB-based electrochemical biosensor prototype for point-of-care diagnosis," *Sensors and Actuators B: Chemical*, vol. 161, pp. 908-913, Jan. 2012.
[6] J. Travis and J. Kring, *LabVIEW for Everyone: Graphical Programming Made Easy and Fun*, Prentice Hall, 2006.
[7] M. Banzi, *Getting Started with Arduino*, 2nd ed., Maker Media, 2011.
[8] A. Jamaluddin, L. Sihombing, A. Supriyanto, A. Purwanto, and M. Nizam, "Design real time battery monitoring system using labview interface for arduino (LIFA)," in *Proc. the Joint Int.Conf. on Rural Information & Communication Technology and Electric-Vehicle Technology*, Bandung, Indonesia, 2013, pp. 1-4.
[9] National Instruments. (2012). Design Patterns in LabVIEW. [Online]. Available: http://www.ni.com/white-paper/7605/en/.
[10] B .P. Douglass, *Design Patterns for Embedded Systems in C: An Embedded Software Engineering Toolkit*, Newnes, 2010.

Supachai Vorapojpisut was graduated from Tokyo Institute of Technology, Japan with the degree of D.Eng (control engineering) in 2000. He works as an assistant professor at the Department of Electrical and Computer Engineering, Thammasat University, Thailand.

His research interests include wireless sensor networks, embedded software development, and computer applications in measurement and control.