WiELD-CAVE: Wireless Ergonomic Lightweight Device for use in the CAVE

Kelvin Parian, Joshua Hegie, Andrew Kimmel
Sergiu M. Dascalu, Frederick C. Harris, Jr.

Department of Computer Science and Engineering
University of Nevada, Reno
1664 N. Virginia St.
Reno, NV 89557, USA

{kelvinp, jhegie, akimmel, dascalus, fredh}@cse.unr.edu

Abstract: *The goal of this project is to design a wireless input device for the CAVE virtual reality environment. The current solutions for this problem are not adequate, due to their high cost and wired nature. By eliminating these two problems, this team hopes to develop a more widely useable device that incorporates all of the functionality that other solutions have included, and more. For ease of use, the device will be housed inside a pair of gloves and will wirelessly communicate to the CAVE. The software driver that accompanies the device will allow users to be able to define a series of hand gestures, which will then execute a program assigned to that gesture – essentially, this will allow users to manipulate the CAVE using only their hands and this device. Hopefully, the applications of this device will eventually expand beyond the researcher demographic to the general public.*

Keywords: CAVE, Input Device, Wireless, Glove

1. Introduction

Since the advent of computing technologies, being able to interact with an electronic environment has been the center focus of a plethora of research projects. Doing things such as clicking and dragging elements of the environment (i.e. through a mouse) or inputting elements into the environment (i.e. through the keyboard), was initially sufficient due to the simplicity of the environment. However, with the increasing popularity of virtual reality environments, such simplistic interactive input devices are no longer adequate. In order to take advantage of this more "advanced" environment, a more intuitive device is needed, and it is within this context that the project takes place in.

The goal of this project is to design a wireless ergonomic lightweight device (WiELD) that will be used to interact with the CAVE [1] virtual environment. The basic functionality of the device is to allow users to wirelessly transmit "gestures" that will be recognized by a driver on the base station and subsequently translated into an "action" on the CAVE screens. The device will be contained inside of a glove in order to provide users with a familiar interface for making the "gestures" (i.e. the gesture where the user contacts the forefinger with the thumb on the glove could translate into a "grab" action on the CAVE).

Modeling the WiELD-CAVE device according to the UML notation specified in [2] helped clarify the project goals, along with making the documentation of the project simpler and more unified.

The key components of this project include: a device with numerous inputs encapsulated in a glove, wirelessly communicating with a base station using an XBee [3] wireless chip, which acts as a wireless RS232 transmitter, the device will be powered by a rechargeable lithium ion battery, all of the inputs will utilize force sensitive resistors, and the corresponding driver for the device will translate a set of contacts into the corresponding action in the CAVE.

Although there are several devices that are quite similar to the WiELD gloves, the current implementations of them are too expensive, wired to the base station, and/or have a limited number of recognizable gestures. By eliminating these problems, this team hopes to develop a more easily accessible, as well as a more usable, device, which will retain all the functionality of its "predecessors" along with additional functions.

The rest of this paper is structured as follows. Section 2 presents the requirements specification of the project. Section 3 presents the use case modeling. Section 4 presents architectural design. Section 5 presents the detailed design. Section 6 presents current status and future work. Section 7 presents our conclusion. Section 8 presents the references.

2. Requirements Specification

Following standard software engineering guidelines [4], the main functional and non-functional requirements of WiELD-CAVE for both the hardware and software sides are presented below.

2.1 Functional Requirements

The most important software and hardware functional requirements of WiELD-CAVE are:

1. The system shall output processed inputs in the form of VRPN [5] code.
2. The system shall provide users with the option of saving their configuration of the device.
3. The system shall provide users with the option of loading their previously saved configuration of the device.
4. The system shall provide users with the option of using a mouse to navigate through the system.
5. The system shall allow multiple sets of gloves to be used simultaneously.
6. The system shall allow for software synchronization of glove settings.
7. The system shall provide the status of the gloves.
8. The system shall provide a setting for secure connection
9. The system shall provide a setting for adjusting transmission power
10. The device shall communicate wirelessly with the system.
11. The device shall be powered by rechargeable batteries.
12. The device shall allow each glove to operate individually.
13. The device shall have a display that indicates the device's status.
14. The user shall be able to calibrate the device.
15. The device may gracefully power down upon a low battery status.
16. The device may have built-in motion tracking.
17. The device may have a built-in accelerometer.

2.2 Non-Functional Requirements

The most important software and hardware non-functional requirements of WiELD-CAVE are:

1. The system shall be implemented using C++.
2. The system shall have a GUI implemented using the QT windowing toolkit.
3. The system shall run on the GNU/Linux operating system.
4. The device shall be programmed in C.
5. The device shall communicate using an Xbee wireless communications chipset.
6. The device shall be implemented using a Cortex microcontroller [7].
7. The device shall be encapsulated in a pair of ergonomic gloves.

3. Use Case Modeling

The functionality of WiELD-CAVE has been defined using use cases and scenarios as defined by the formal modeling process presented in [1] (section 1 of this paper). The use case diagram, shown in Section 3.1, captures the entire functionality of WiELD-CAVE. This was done to help identify the mechanisms through which the user would interact with WiELD-CAVE. The use cases are compared to the requirements listed in Section 2 using the Requirements Traceability Matrix in Section 3.2.

3.1 Detailed Use Cases

Presented below are Use Cases for WiELD-CAVE. A use case diagram is presented in Fig 5.

UC1. sendData - is the framework for collecting and transmitting data back to the base station.

UC2. receiveData - is responsible only for receiving data sent from the WiELD glove, acknowledging that the data has been received and passing the data on to the processing function.

UC3. processData – takes the user generated inputs and turns them into something the CAVE can understand. The user initiates the data processing via the sendData and receiveData methods. The CAVE is the end destination of this processed data.

UC4. finishData - is designed to ensure that all input is received by the base station. If one of the gloves has not sent its data yet, then a request is sent to the missing glove to provide the state of its contacts.

UC5. pollDevice - is a failsafe that can be called by the receiver if a device has failed to send data. It is the responsibility of this function to respond to finish requests and handle all of the cleanup associated with sending data.

UC6. turnDeviceOn - is invoked when the device is turned off and the power button is pressed. The device then has to power up the microcontroller. Once this is done, the microcontroller should power up all of the peripherals as well as the Xbee wireless chip. At this point the Xbee chip should connect with the base station. After all of this is

complete, the device will be ready to send data to the base station.

UC7. turnDeviceOff - is used to gracefully power down one of the WiELD gloves. The user initializes this by holding the power button down for a set amount of time. The wireless chip unpairs itself from the base station and the microcontroller is powered down.

UC8. resetDevice - is initiated by the user when the reset button on one of the WiELD gloves is pressed. In the event that the hardware becomes unresponsive, this function is designed to power cycle the device. This disconnects the wireless chip from the base station and powers down the microcontroller. Once everything is powered down, everything is returned to the ON state.

UC9. coordinate - is invoked on the base station when one of the devices sends its data. The base station must read in 2 bytes from each device and then combine all of this data into a single numerical value. This is entered into the VRPN driver, so that client programs can access the status of the buttons on the gloves.

UC10. chargeBattery – is a function to allow the user to recharge the battery in the WiELD glove. This functionality is invoked by the user removing the device from the glove and placing it on the charging station. The device will then enters a low power state and begins charging. The microcontroller stays on to control the display, and once the battery has finished charging, the display turns off.

UC11. lowBattery – is invoked to gracefully power the device down, causing the Xbee wireless chip to disconnect from the base station. This is done to prevent the chip from continuing to send inaccurate data back to the base station. The device has to watch the battery state and once the battery is at or below a certain threshold, the user gets a warning. If the battery goes below a second threshold, the device then powers down, disconnecting the Xbee wireless chip from the base station and powering down the microcontroller as well as all of the peripherals. Time acts on this because leaving the device on for any amount of time causes the charge in the battery to deplete.

UC12. showValue - is a functionality designed to show programmers what value they need to catch from the shared VRPN memory space. The user can generate input on one or more WiELD device and see the value that the program will place into the VRPN memory space.

UC13. changeSetting - is designed to allow a user working at the base station to modify the firmware on the transmitters, changing things such as the transmission strength or enabling/disabling encryption.

UC14. modSetting - is the glove interface to changeSetting. This must block the device from transmitting, update the data and then re-enable the device.

UC15. calibrateDevice - is used to let the user to calibrate how sensitive the inputs are. This will change the triggering threshold for the analog to digital converter, making it higher or lower, based on user preference.

3.2 Requirements Traceability Matrix

The Requirements Traceability Matrix, detailed below (Requirements are in the left most column), shows how the use cases match up with the requirements listed in section 2.

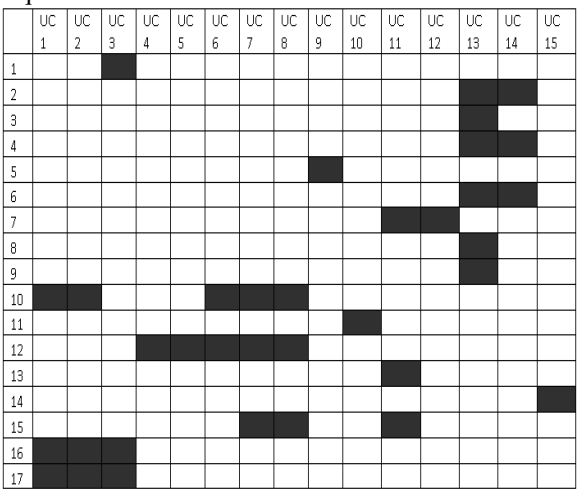

Fig 1: Requirements Traceability Matrix

4. Architectural Design

The layered architecture is one in which all data is passed through a series of hierarchical layers. A brief description of each subsystem utilized in WiELD-CAVE is as follows:

C++ Libraries: WiELD-CAVE's driver is implemented using C++.

C Libraries: WiELD-CAVE's firmware will be implemented using C.

Qt: The driver GUI for WiELD-CAVE is handled through Qt windowing toolkit libraries and Qt framework.

Help System: A series of documents along with informative error messages which will assist in the user in trying to fix any problems encountered.

Main Window: The main window displays the status of the gloves in an easy to read format.

LCD: The LCD on the gloves shows the current device status and is located directly on the glove.

Gloves: Utilizes force sensitive resistors, as well as analog-to-digital circuitry in order to translate user input to digital logic.

5. Detailed Design

The class diagram for WiELD-CAVE, presented according to the specifications laid out in [1], is included in Figure 6. This diagram lists all the classes in WiELD-CAVE as well as most of the major functions. All of the Qt variables and functions have been omitted in order to preserve room; however, this does not adversely affect the content of the diagram.

6. Current Status and Future Work

Currently, there is a working prototype of the WiELD-CAVE gloves. Both gloves have all of the sensors installed and is communicating with the wireless base station. The software driver is able to receive and interpret the data, as well as relay it to the VRPN server in the CAVE.

Future work includes building the motion tracking into the WiELD-CAVE glove using accelerometers and small gyros. Currently we are using video tracking through IR cameras and markers. The circuitry can be further minimized so that the main control unit can be built into the gloves instead of being enclosed in an armband.

6.1 Hardware Screenshots

Figure 2 shows the current implementation of the WiELD-CAVE gloves. On the fingertips are the force sensitive resistors. Figure 3 is a snapshot of the inner circuitry of the armband enclosure, behind the Luminary ARM Cortex controller board is the analog to digital conversion circuitry. Not pictured is the interface from the glove itself to the enclosure, which is currently implemented as a ribbon cable with sufficient wires to send all of the data, for all nine of the inputs, to the controller.

Fig 2: WiELD-CAVE Glove

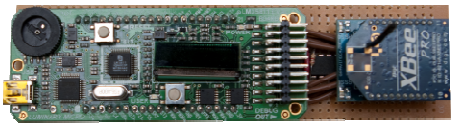

Fig 3: WiELD-CAVE Transmitter for Glove

6.2 Software Driver Screenshot

Figure 4 shows the connection status part of the driver, which updates in real time as the gloves' status changes.

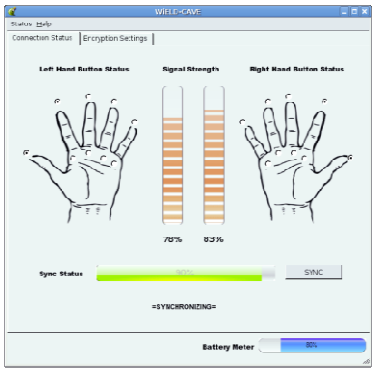

Fig 4: WiELD-CAVE Driver GUI

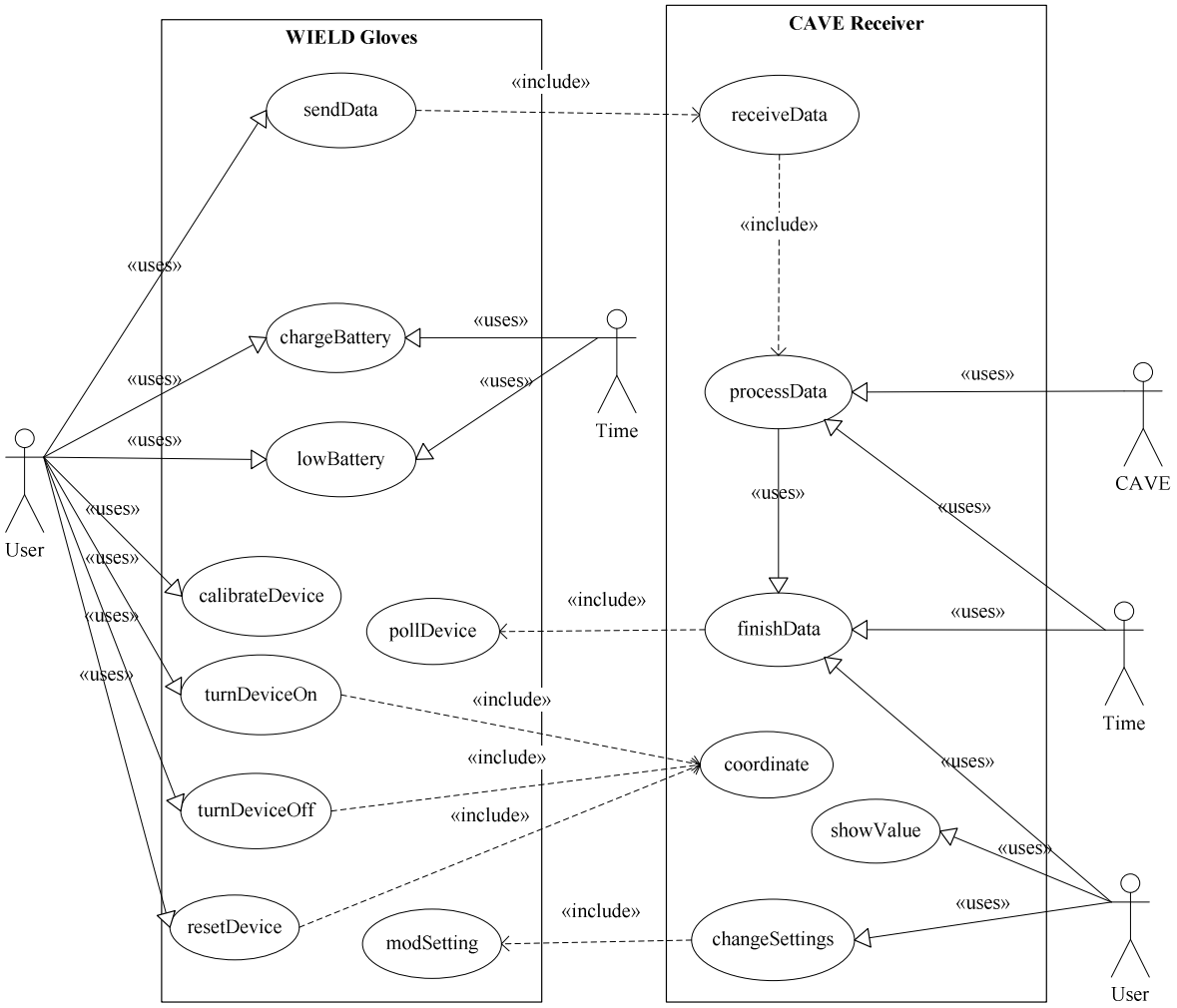

Fig 5: Use Case Diagram

7. Conclusion

The WiELD-CAVE device which has been presented in this document is a unique, cost effective solution to intuitively interacting with the CAVE environment. WiELD-CAVE provides a high degree of flexibility in what can be taken as an input, as well as allowing developers to define what the device is supposed to do. The driver is light weight, requiring very little memory, and features a GUI that is simple to understand and intuitive to use.

There are nearly endless possibilities for expansion and improvements in the device. For example, with a little bit of modification, the WiELD-CAVE device could be used in gaming to interact with in-game objects. Other input architectures, such as DirectX, could be used as a backend for the device, allowing a much larger install base to use the WiELD gloves. System cross-compatibility should also be accounted due to the initial driver being limited to Linux distributions. These improvements will allow WiELD-CAVE to be used in more computing environments outside of virtual reality and the CAVE.

8. References

[1] W. Sherman and A. Craig, *Understanding Virtual Reality: Interface, Application and design, Morgan Kaufmann*, 2003.
[2] J. Arlow and I. Neustadt, *UML 2.0 and the Unified Process: Practical Object-Oriented Analysis & Design*, Addison-Wesley, 2006.
[3] *Getting Started with ZigBee and IEEE 802.15.4*, Daintree Networks Inc., 2004-2008

[4] I. Sommerville, *Software Engineering*, Addison-Wesley, 8th Ed., 2006.
[5] R. M. Taylor II, T. C. Hudson, A. Seeger, H. Weber, J. Juliano, and A. T. Helser, *VRPN: A device-independent, network-transparent VR peripheral system*, ACM Symposium on Virtual Reality Software and Technology (2001).
[6] Dalheimer, Matthias Kalle, *Programming with Qt*, O'Reilly, 2nd Ed., 2002
[7] S. Sadasivan, *An Introduction to the ARM Cortex-M3 Processor*, ARM, 2006

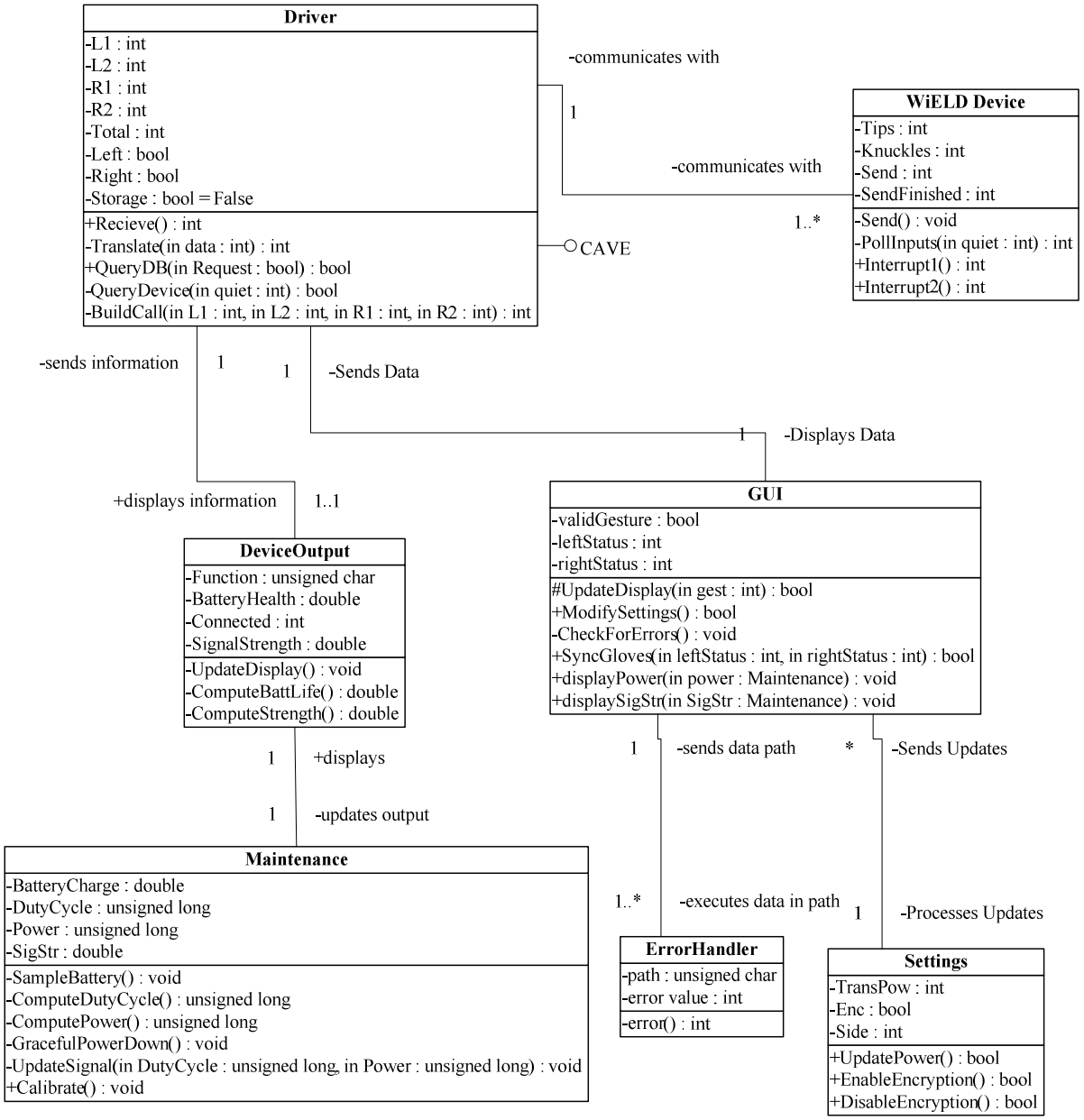

Fig. 6: WiELD-CAVE class diagram