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

Joshua M. Hegie, Andrew S. Kimmel, Kelvin H. Parian
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
Tel:(775)784-6974 Fax:(775)784-1877

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

Abstract: The goal of this project was 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 aimed 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 is housed inside a pair of gloves and is capable of wirelessly communicating with the CAVE. The software driver that accompanies the device allows users to define a series of hand gestures, which are then mapped to either a button press in software – essentially, this allows users to manipulate the CAVE using only their hands and this device. Hopefully, the applications of this device will eventually expand beyond the researchers' interests 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 was to design a wireless ergonomic lightweight device (WiELD) that will be used to interact with the CAVE [1] virtual environment [2]. 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 is 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 [3] 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 [4] wireless chip, which acts as a wireless RS232 transmitter, the device is powered by a rechargeable lithium ion battery, all of the inputs utilize force sensitive resistors, and the corresponding driver for the device translates 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 has developed a more easily accessible, as well as a more usable, device, which retains 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.

2. Requirements Specification

Following standard software engineering guidelines [5, 6], 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 [7] 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 to the user.
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 [9].
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 [3] (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, see Fig. 2.

3.1 Detailed Use Cases

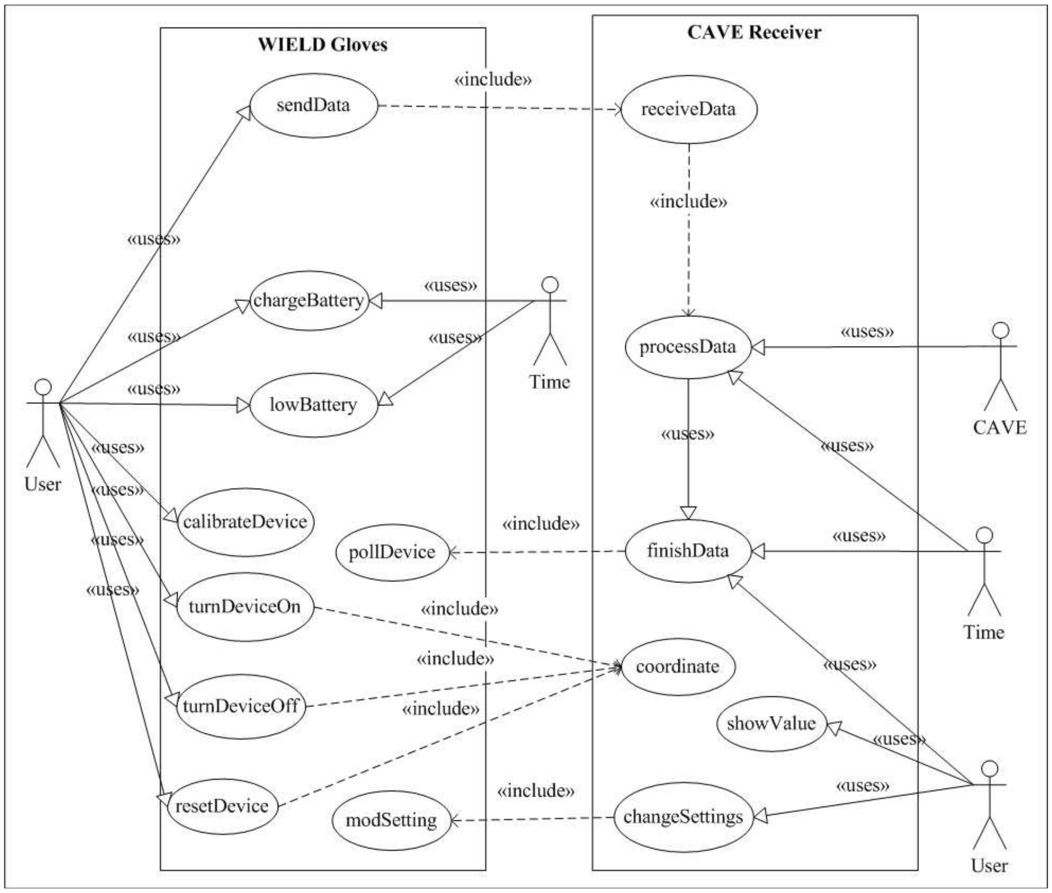


Fig. 1: Use case diagram

Presented below are the Use Cases for WiELD-CAVE. A use case diagram, detailing how the use cases for both the hardware and the software interact, is presented in Fig 1.

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, via the send and receive functionality of the systems, initiates processing. The CAVE is the endpoint for this branch of the system.

UC4. finishData - is designed to ensure all input is received by the base station. Based on which gloves are connected to the base station. If one of the gloves has not sent its data, a request is sent to request that it 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 the entire 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 will power up all of the peripherals and the Xbee wireless chip. Now the Xbee chip should pair with the base station. After all of this, 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 disconnects 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. This function is designed to power cycle the device, in the event that the hardware becomes unresponsive. 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 it 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 programs running on the cave can access the state of the buttons.

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 enter a low power state and begin 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, so that the Xbee wireless chip disconnects from the base station, in order to prevent it from interfering with a new glove for that hand being introduced. The device has to watch the battery state once the battery is at or below a threshold. The user gets a warning once the battery is below a second threshold and the device powers down by disconnecting the Xbee wireless chip from the base station and powering down the microcontroller and 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 which contacts are activated (have sufficient pressure applied) and the value that the program will place into the VRPN memory space.

UC13. changeSetting - is designed to allow a user at the base station to modify the firmware on the transmitters to change things such as the transmission strength or enable/disable 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 (Fig. 2), detailed below (Requirements are in the left most column), shows how the use cases match up with the requirements listed in Section 2.

	UC 1	UC 2	UC 3	UC 4	UC 5	UC 6	UC 7	UC 8	UC 9	UC 10	UC 11	UC 12	UC 13	UC 14	UC 15
1		■													
2													■	■	
3													■		
4													■	■	
5									■						
6													■	■	
7											■	■			
8													■		
9													■		
10	■	■				■	■	■							
11										■					
12				■	■	■	■	■							
13											■				
14															■
15							■	■			■				
16	■	■	■												
17	■	■	■												

Fig. 2: 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 is 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 (Liquid Crystal Display) on the gloves shows the current device status and is located directly on the glove.

Gloves: Utilizes force sensitive resistors to generate signals that are passed to the transmitter.

Transmitter: The core of the hardware. This contains the analog-to-digital circuitry required to translate user input to digital logic, and interface with the wireless transmitter.

5. Detailed Design

The class diagram for WiELD-CAVE, presented according to the specifications laid out in [3], is included in Fig. 3. 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. The major classes for this system are WiELD Device, Driver and Maintenance.

WiELD Device is the class that encompasses the hardware element of the project, shown in Fig. 5. The primary responsibilities of this class are to notify the base station of changes in the button status. This is done through the private functions Interrupt1 and Interrupt2. These interrupt routines are triggered by the analog to digital converter, which is built into the ARM Cortex microcontroller, when it reads a voltage that has passed a threshold that is defined in code. Once this happens, the system checks the state of the transmit buffers to see if there are new buttons, using fast bitwise logic. In the event that there has been a change the send function is used to notify the driver of the change, otherwise nothing is sent to conserve power.

The Driver class is what is run on the VRPN server. This class is responsible for receiving and decoding the information that the WiELD device provides to it. The Receive function is automatically invoked by the USB XBee receiver. Once all data has been passed from the hardware to the driver, the Translation function is invoked. This function uses bitwise logic, bitwise AND and bitwise rotates, to determine which of the data sets it is currently inspecting. Once this information is available it is possible to place the received data into the correct storage variable. The BuildCall function is used to put all four of the storage variables into a single coherent value, so that it can be passed to VRPN. The nature of battery powered devices is, unfortunately, unstable. To prevent data loss there is a heartbeat function, QueryDevice, which is built into the driver. This makes it possible for the driver to correct for instances where the device loses connection, either from wireless interference or because the battery has run out. In the event that a heartbeat signal is not replied to the driver can zero out all of the information associated with the non-responsive device, effectively setting all buttons to the state where they are not pressed, and continue operating. Once the device comes back, it will be able to seamlessly begin communication with the base station.

The Maintenance class is distributed across both the hardware, and the software driver. This class allows the vital information, such as the signal strength or remaining charge of the battery, to be passed between the gloves and the driver. This allows for functionality that powers the gloves down automatically, and with a clean disconnect from the driver, when the remaining charge on the battery drops below a defined threshold. This also allows for the driver, and the LCD display on the glove, to show how strong the signal strength for the wireless transmitters are, allowing the user to easily get out of a location that prevents the gloves from communicating with the base station.

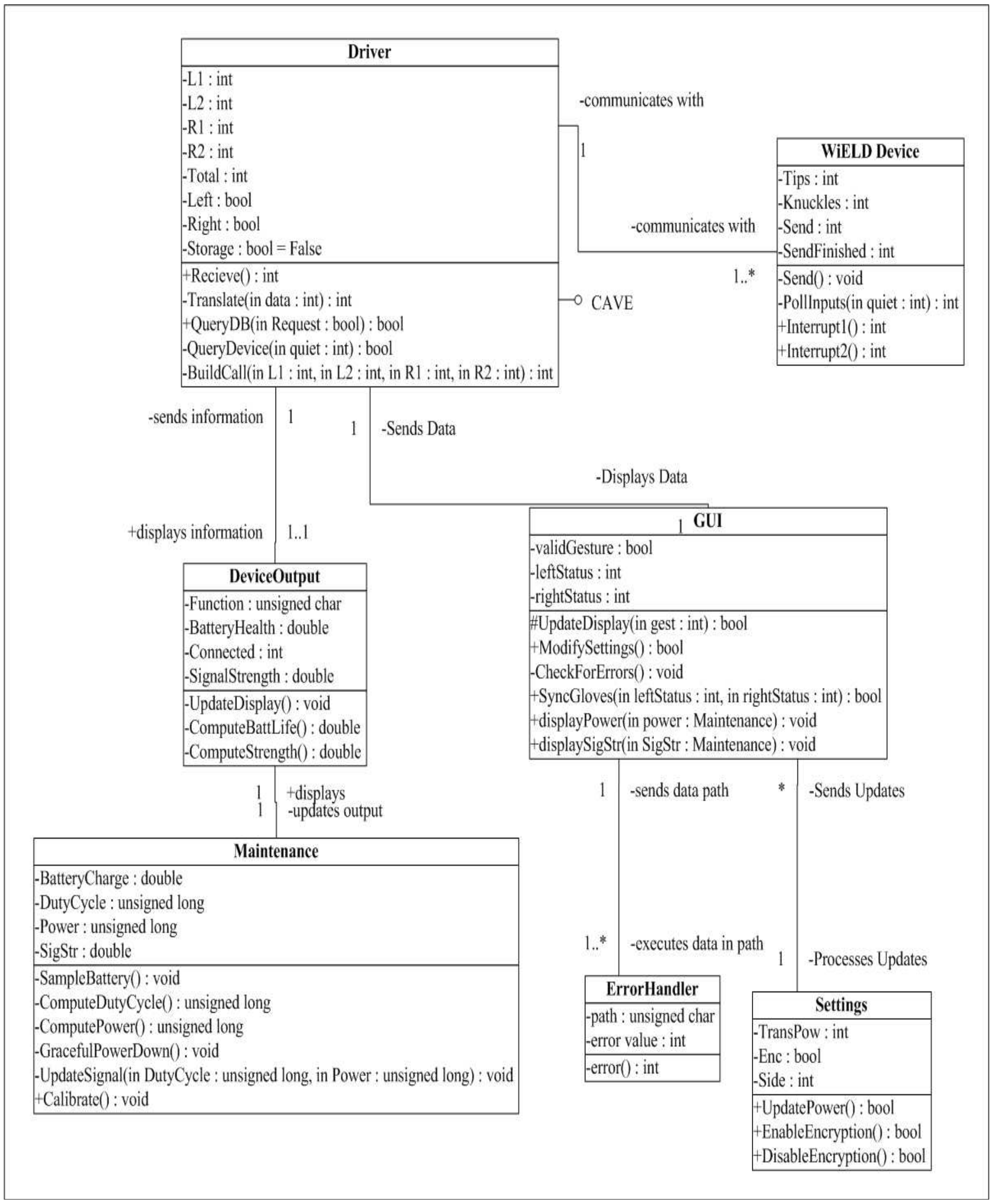


Fig. 3: Class Diagram

6. Current Status and Future Work

Currently, there is a working prototype of the WiELD-CAVE gloves. This prototype includes two gloves and two transmitters, the base station and all of the associated software. The software driver is able to receive and interpret the data, as well as relay it to a VRPN server.

The work done on this project can easily be expanded. Future work includes building motion tracking into the WiELD-CAVE gloves, using accelerometers and small gyros. Currently hand motions are being tracked using video tracking through IR cameras and reflective markers. While this does what it needs to, it is expensive, and can be cumbersome. The circuitry in the transmitter can be greatly minimized, allowing the main control unit can be built into the gloves instead of being enclosed in an armband and externally connected.

6.1 Hardware Screenshots

Fig. 4 shows the current implementation of the WiELD-CAVE gloves. On the fingertips are the force sensitive resistors. Fig. 5 is a snapshot of the inner circuitry of the armband enclosure, behind the Luminary ARM Cortex controller board is the circuitry that allows for the internal analog-to-digital conversion to work. Since the voltage that is permitted through the force sensitive resistors, even when the pressure is at the maximum allowed, is so small that it normally would not be detected. To compensate for this a series of amplifier circuits have been put behind the microcontroller. Not pictured is the interface from the glove itself to the enclosure, which is currently implemented as a ribbon cable [10] with a sufficient number of wires to send all of the data, for all nine of the inputs, to the controller.

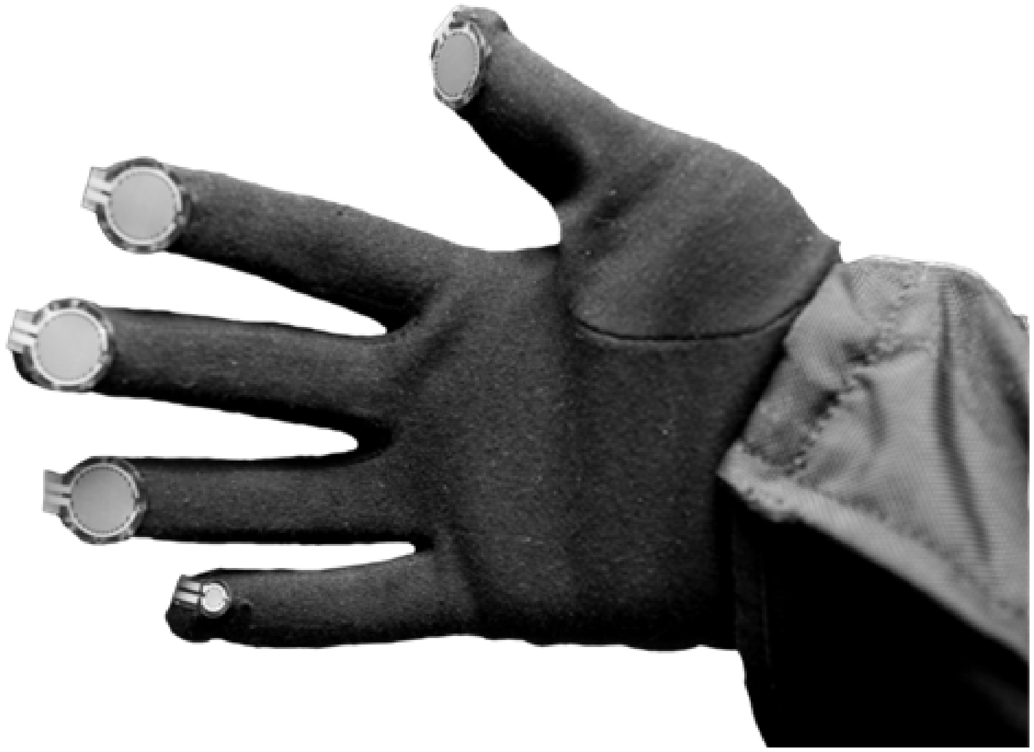


Fig. 4: WiELD-CAVE Glove

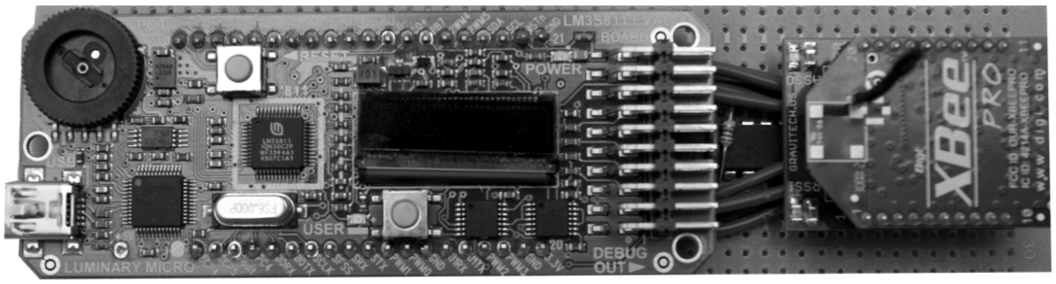
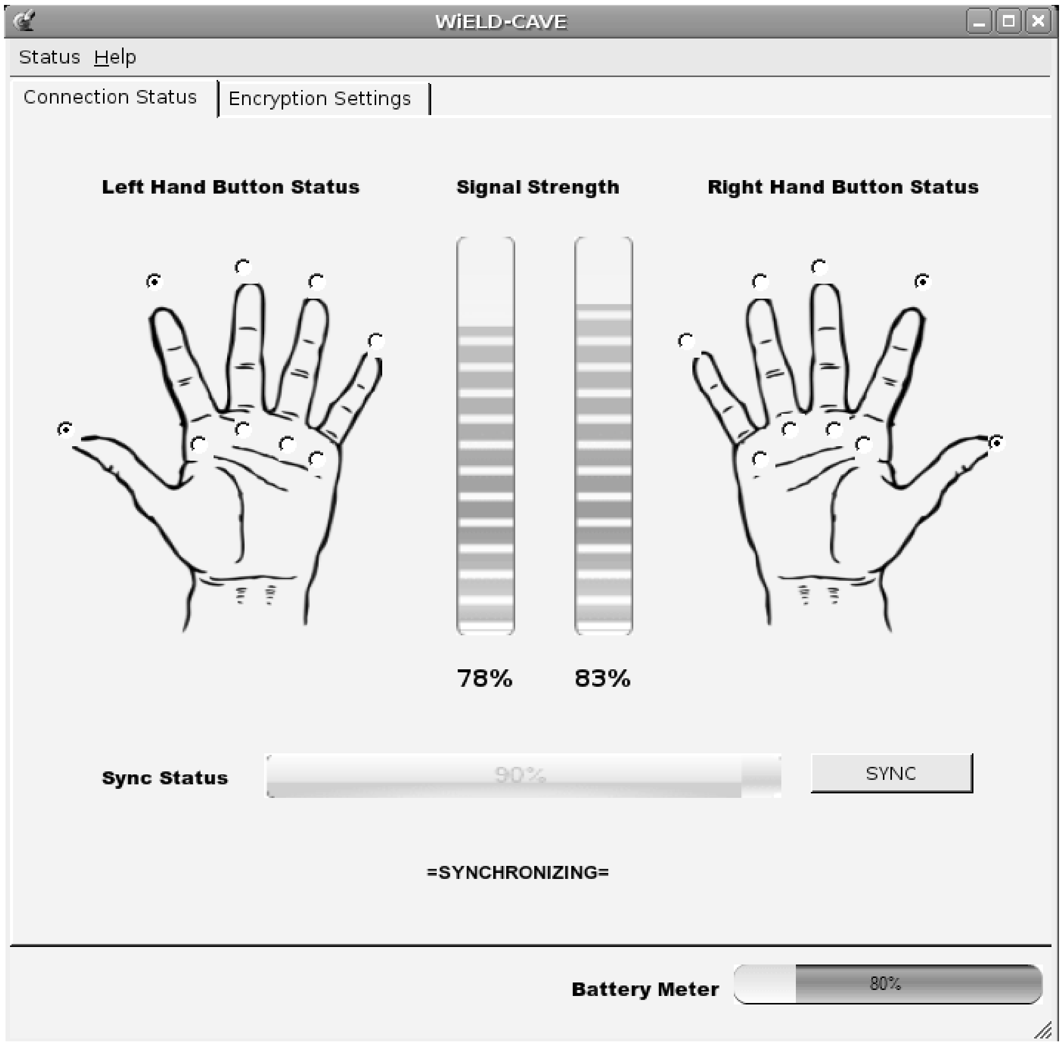


Fig. 5: WiELD-CAVE Transmitter for Glove



78%

83%

Sync Status 90%

=SYNCHRONIZING=

Battery Meter 80%

Fig. 6: WiELD-CAVE Driver GUI

6.2 Software Driver Screenshot

Fig. 6 depicts the user interface [11, 12] of the system and shows the connection status part of the driver, which updates in real time as the gloves' status changes. The hands on the left and right hand sides of the interface allow the user to see which buttons are currently pressed. This can be useful if a button does not seem to be responding, the user can launch this application and test any button they like and see if it causes an update on the display. In the center is a graphical representation of the signal strength. This allows users to determine if there needs to be a settings change between low and high powered transmission. Along the bottom of the main section of the window is a bar that displays the synchronization status, which is the terminology used to describe the signal strength and a software check of the coherence of the various settings on each glove, such as encryption. There is a second tab, which is not shown here, which allows the user to enable or disable the hardware encryption mechanism for the XBee transmitters. This functionality allows the user to encrypt the transmissions [13] between the gloves and the base station, making it easier to have multiple sets of gloves on the same wireless channel or to prevent other stations from reading the button presses that are being generated.

7. Conclusion

The WiELD-CAVE device which has been presented in this document is a unique and 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 does with the inputs. 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.

References

[1] W. Sherman and A. Craig, *Understanding Virtual Reality: Interface, Application and Design, Morgan Kaufmann*, 2003.

[2] R.S. Kalawsky, *Science of Virtual Reality and Virtual Environments*, Addison-Wesley, 2004.

[3] J. Arlow and I. Neustadt, *UML 2.0 and the Unified Process: Practical Object-Oriented Analysis & Design*, Addison-Wesley, 2006.

[4] *Getting Started with ZigBee and IEEE 802.15.4*, Daintree Networks Inc., 2004-2008.

[5] I. Sommerville, *Software Engineering*, Addison-Wesley, 8th Ed., 2006.

[6] R. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Ed., Prentice-Hall, 2005.

[7] 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).

[8] M.K. Dalheimer, *Programming with Qt*, O'Reilly, 2nd Ed., 2002.

[9] S. Sadasivan, *An Introduction to the ARM Cortex-M3 Processor*, ARM, 2006.

[10] J. Hegie, K. Parian, A. Kimmel, *WiELD-CAVE*, Senior Project in Computer Science and Engineering, University of Nevada, Reno, 2009. Accessed May 2009 at http://www.cse.unr.edu/cs426/09/team4/

[11] B. Shneiderman and C. Plaisant, *Designing the User Interface*, 5th Ed., Addison-Wesley, 2009.

[12] J. Spolsky, *User Interface Design for Programmers*, Apress, 2001.

[13] A. Silberschatz, P.B. Galvin, and G. Gagne, *Operating System Concepts*, 7th Ed., Wiley & Sons, 2004.