Terrick Adolphe

# Survivor Buddy Web Interface: Human-Robot Interaction with Survivor Buddy Robot

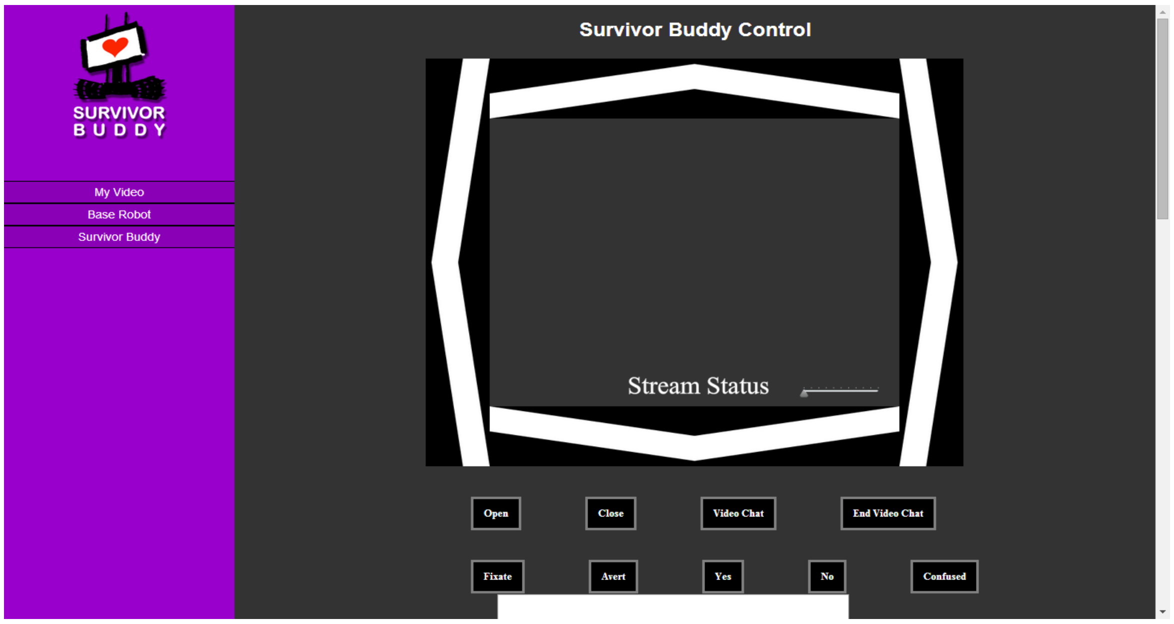
Abstract – The purpose of this project is to create a simplified interface that medical professionals can use in order to operate Survivor Buddy. Presently, the existing web application has proven to be too complicated for medical responders and the existing interface for responders did not utilize gaze acts for comfortable conversations. The changes being made will make it easier for medical professionals to effectively utilize all of the interface's features. If successful, most medical responders will be capable of operating Survivor Buddy while being reminded of assessment protocols and that in turn will potentially save lives. Started in 2007, the Survivor Buddy's project has been evolving for seven years. The midterm check will be a functioning web application in its beginning stage. The final exam is a fully functioning web application ready to be used on the field with protocol reminders, video and audio conferencing capabilities, and a text and picture sharing feature.

## 1. INTRODUCTION

Survivor Buddy is a rescue robot that is deployed in situations where it may be too dangerous for a first responder. When deployed, Survivor Buddy finds victims of disasters and connects them to a medical profession through the web interface. The Survivor Buddy Web Interface is a web application accessible through web-enabled devices that acts as a tool for communication between the user and Survivor Buddy. Users operating the web interface are able to see a live stream video from Survivor Buddy's camera, communicating with the disaster victims through video/audio chats and picture chat. Users operating the interface can also move the Survivor Buddy with manual movements, engage in text-to-speech, and enable automated head-gaze generation. The new version of the web interface incorporates the already existing programs of manual movements, text-to-speech, and automated head gaze generation into a web application. The new version has added video/audio feeds and picture display so that there are multiple forms of communication between Survivor Buddy and the operator computer.

## 2. RELATED WORK

The previous version of the interface already supported automated head gaze generation, text-to speech functionality, manual movements, and "Protocol Coach". Automated head gaze generation was dependent on the text that the user inputted. Depending on the amount of words, the amount of punctuation, and the amount of uses of backspace, the Survivor Buddy robot would mimic the head movements of humans in an effort to give Survivor Buddy more human-like attributes. The text-to-speech function allows the user to input text and have it spoken through the Survivor buddy. On the interface, there were several buttons in which the user can manually move Survivor Buddy's head mount or have the head mount mimic common head gestures such as fixation and aversion. Finally, Protocol Coach is made for the purpose of giving medical responders script overview and context questions when interacting and assessing the victim.



## 3. APPROACH

The approach builds on implementing real-time communication to the previously made web interface to enable communication between victims of disasters and medical responders through Survivor Buddy. There was a need to implement various forms of communication between the victim and the medical responder. The primary forms of communication include video and audio feeds, real-time image display, text chat, and displaying Survivor Buddy's screen on the web interface.

## 4. IMPLEMENTATION

To implement browser-to-browser communicative applications to the web interface, Node.js and several Node.js frameworks were used, playing a vital role in the reformatting of the interface. Node.js is a runtime environment used for running web server applications and transporting that data to remote computers in real-time. Node.js had to be used in order to run the interface for the purpose of connecting the interface to a server that can be monitored and adjusted if needed. The idea of Node.js is using non-blocking, event-driven input/output to remain lightweight and efficient in the face of data-intensive real-time applications that run across distributed devices. Node.js particularly shines in building fast, scalable network applications that are capable of handling large numbers of simultaneous connections with high throughput, which equates to high scalability. Therefore, Node.js was the best option for handling the server-side component of the web interface. Express.js, one of Node.js frameworks, played a vital role for the request and responses on the web interface. Express.js allows for the handling of requests in web applications and the redirection of views to actually render them. Requests for web applications first start at the route that defines the URL schema. It captures the matching request and passes the control to a template. The template then constructs the HTML for the response and sends it to the browser. The route handler doesn't necessarily need to always pass the control to a template as it can optionally send the response to a request directly, which is primarily how the web interface handles request. WebSockets was needed to handle information sending. WebSockets are protocol providing full-duplex communications channels over a single TCP connection in Node.js. It allows servers to send information to the client. The web interface heavily relies of WebSockets to operate the bi-directional connection simultaneously in real-time. In addition to its main functionality, The WebSockets API includes cross origin communication, cross platform compatibility (for the purpose of operating the web interface from most web-enabled devices such as mobile devices), and has low weight envelop when passing massages to and from the server and the client. Websockets are a powerful tool for adding real-time functionality to the web interface and is very useful for the communication purposes. Lastly, Socket.IO, another Node.js framework, allows for real-time communication channel between a web browser and a server (in this case, a server running Node.js and Express). By default, Socket.IO will use the WebSockets protocol if it is supported by the browser. Older browsers such as IE9 do not support WebSockets. In that case, Socket.IO will fall back to other technologies, such as using Flash sockets or an Ajax technique called long-polling. Socket.io is used to enable real-time communication between Survivor buddy and a remote operator computer. It also provides a failsafe incase the operator computer does not have the latest supported version needed to run the interface.

## 5. CONCLUSIONS

The new web interface adds several new features to the table. Multiple forms of communication were added to allow the medical responder that is operating the web interface more leeway when communicating with disaster victims through Survivor Buddy. The new features include video/audio feeds and real-time image display. As of now, the interface is reliant on Chrome Remote Desktop to see and operate the Survivor Buddy's screen and desktop.