# iWiimote

**Adam El Sheikh**
Virginia, United States
age2bh@virginia.edu

**Logan Ramsey**
Virginia, United States
lr8fm@virginia.edu

**Quinn Ciccoretti**
Virginia, United States
qjc5fb@virginia.edu

**Zachary Bilmen**
New York, United States
zac@virginia.edu

## ABSTRACT
Mobile phones offer a wide variety of sensors that user interfaces (UIs) can leverage. Additionally, mobile phones have low barrier to entry since many users already own one. The iWiimote is a functional game controller, implemented on the web, that uses gyroscopic and touch sensors to emulate input on the computer it is connected to. Though it has some limitations, the iWiimote is fun and easy to use with a variety of games.

### Author Keywords
Networking; sockets; HCI; Nintendo Wii; Apple; Android; 9-DOF

### CCS Concepts
•**Human-centered computing** → **Web-based interaction;**

## INTRODUCTION
Orientation based peripherals for computers are not mainstream and should be examined as a potential interactive tool for digital experiences. In our case, we used the gyroscope on iOS and Android devices to control mouse input and keystrokes over a local network connection. Our flexible system allows a mobile phone to control one's computer in a variety of scenarios, such as playing a game like Minecraft or browsing the web. To implement this control scheme we sent commands with websockets to a Python script that performed input emulation. We evaluated the overall paradigm initially using a Wizard of Oz study and then a simple user test receiving qualitative comments on the experience.

## DESIGN
The initial design plan was to map device movements to actions within a game. Drawing inspiration from the Nintendo Wii remote, we opted for a design that splits the controller interface into two parts: the pointer and the gamepad. The pointer interface is accessible to the user when his or her smartphone is in portrait orientation.



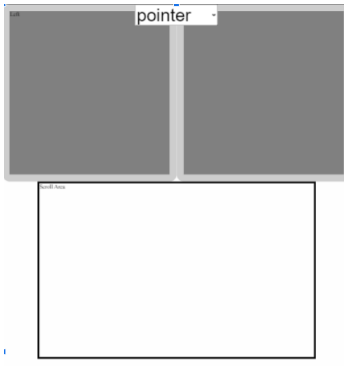


**Figure 1. Pointer**

Pointing the phone when in pointer mode moves the computer's cursor across the screen. On the pointer screen, there are two buttons displayed that the user can click. One button simulates a left mouse click when pressed and the other simulates a right mouse click when pressed. An area on the screen below the mouse buttons is designated for scrolling.

Switching the smartphone to landscape orientation makes the gamepad interface available to the user. On the phone screen, the user will see a gamepad template that shows a directional pad and four face buttons.


**Figure 2. Gamepad**

The elements on this interface map to the functions on a normal gamepad. For instance, tapping the right button on the

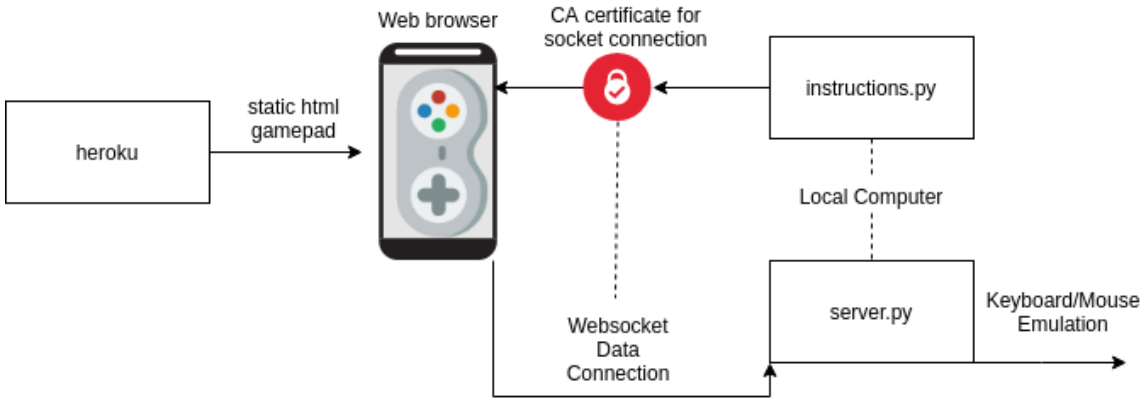


**Figure 3. Data Flow**

directional pad corresponds to a right key press. The elements can also be remapped by changing the input they produce on the server.

## IMPLEMENTATION

### Iteration 1

Early in the design process we used the cross-platform data collection app Phyphox [6]. Phyphox starts an HTTP server on your phone, which a computer connects to with a Python script. The first iteration of the Python script used the input emulation library *pyautogui* [7]. *pyautogui*, however, caused significant, unnecessary latency, which was removed when we switched to the lower level *keyboard* [2] and *mouse* [3] libraries. The implementation of these libraries requires super-user permissions on Linux, and displayed some OS-based inconsistencies before we resolved which methods worked well cross platform.

The first iteration of iWiimote was driven by this python script using HTTP to query for device orientation. By tilting the phone, you could move your computer mouse or emulate arrow key presses. And though it worked, there were some issues. First, the Phyphox app did not allow us to provide any input from the touchscreen, so our game controller lacked buttons. Second, the HTTP GET requests, even when performed asynchronously, were rather slow.

### Iteration 2

To combat the latency inherent to HTTP requests, we decided to switch to WebSockets [5]. Fortunately Mozilla's Web APIs allow mobile phone browsers access to the same data that Phyphox collects. What's more, with HTML and CSS we had a pretty robust way of designing touch input — something iteration 1 lacked entirely.

The webpage is statically served from heroku, but contains javascript that connects to a websocket server on the computer over the local network. The site also contains javascript that enables permission for Mozilla's DeviceMotion and DeviceOrientation API [4]. Users must give the website permission before it can poll sensor data.

All processing of sensor data was performed on the mobile device. The mobile device then determined the gesture or interaction, and sent a command to the server, which parsed the simple command and performed the appropriate emulation.

Here is an example set of commands:

```
{
"key":  "press left"
}
{
"GyrX":  25.6797,
"GyrZ":  0.3390
}
```

The first command tells the server to use the keyboard library to emulate a left keyboard press and hold. The second command provides the server with gyroscope data, which is processed into a mouse cursor movement. The command-based server allows for multiple mobile phones to connect, enabling multiplayer for split-screen style games.

To access sensor data with Mozilla's Web API's for device orientation, the site must use HTTPS. In order for a secure connection to be facilitated over the local network, we generated self signed Certificate Authority (CA) certificates specific to our computer's local IP address, and then explicitly trusted these certificates on our mobile phones. On Android, this meant accepting the "Proceed to unsafe site" dialog. In the case of iOS however, it meant that we needed to install the CA as a profile in System Settings, then explicitly trust the certificate, which is rather too involved for a casual user to follow easily. Additionally, Apple appears to frequently update the requirements for their certificates, which might eventually obsolete our method [1].

To clarify this process, we added an additional server that hosts an instruction webpage and the CA certificate for easy download to the device.

Despite its complexity, the socket approach allowed us to sample and transmit data with noticeably lower latency, resulting in a very responsive controller, especially for the pointer, which used the gyroscope data to move the mouse. A command-based server also allows for multiple mobile phones to connect, enabling multiplayer for split-screen style games. A video demo is available at https://youtu.be/4x07fuPlfCU

## EVALUATION

### Wizard of Oz

Our initial feedback on this interface was obtained from "Wizard of Oz Testing." Participants were given a smartphone and told it can perform gesture controls to play a web based video game. The controls were very different from the final product. They were as follows:

1. Tilting left to move the character left
2. Tilting right to move the character right
3. Shaking the device to jump

Despite the sample size of 2, the feedback was insightful and consistent in certain aspects. The participants reported that the interface was unique and had potential, but they were critical of certain controls. Shaking to jump was uncomfortable, and using analog motion for directional input was less desirable than traditional controller buttons. Based on this feedback, we determined touch input would improve interaction with the iWiimote, and make it better resemble the Wii remote.

### Final Testing

We set up a room of skribbl.io, an online pictionary game, for four people (the authors of this paper) to evaluate whether our pointer could compete with traditional mouse input. We found that any sudden movements caused the drawing to lose its scale, indicating that the sensitivity may have been too high for the application of drawing. Additionally, it's difficult to rest the phone pointer without putting it down.

As another usability test, we each used Google's autodraw.com to see how fast we could draw basic items. Once Google was able to recognize the object within its top 5 suggestions, the test was concluded. Two of the team members drew with the mouse first and two of the team members drew with the iWiimote pointer first. We recorded cumulative times to draw a teddy bear, a computer mouse, a smiley face, and an apple (as recognized by the website). This is shown in the table below for each team member.

| Team Member | Z. | A. | Q. | L. |
|---|---|---|---|---|
| Mouse | 0:40 | 0:33 | 2:28 | 0:48 |
| Pointer | 0:39 | 1:01 | 2:40 | 1:15 |

Since our sample size was 4, it is difficult to draw conclusions from the raw data. Three out of the four team members performed the drawing tasks quicker with the computer mouse than with the iWiimote. Qualitatively, however, the pointer felt better. One team member concluded that although the iWiimote was "wobbly", the mouse interface on the iWiimote felt better than his low-quality laptop trackpad. Since we've been using mouse pointers for years, it is likely that we could perform drawing with the mouse pointer quicker.

## DISCUSSION

In the end, a socket-based approach to controlling your computer with a mobile phone proved quite robust. Switching between gamepad and pointer gives the user control over input. In terms of capabilities, one could use the controller for everything besides generalized keyboard input (such as writing text in a text box). Despite the robustness of this system, there are still minor flaws in the quality of life. Although we found the pointer interface usable, we believe a traditional mouse is better interface for most ordinary tasks. On the plus side, the controller seems to enable convenient browsing of the web if a mouse is not in one's immediate vicinity, for example, you are lying on your bed and your computer is on your desk. With further development, we believe that the controller could be a plausible alternative to the mouse.

The largest drawback to our current product is the complicated setup. We set up the Certificate Authority so that it is generated anew every time a set up is done, so no two individuals will have the same CA, preventing a potential security issue.

The potential of the iWiimote is largely dependent on the programs you use it with. In a game like skribbl.io, the sensitivity is too high to draw effectively but is perfect to pan the camera in a game like Minecraft. Catering towards drawing would turn the iWiimote into a stylus and detract from the effectiveness of other games/simulations. The iWiimote is not one size fits all. The next immediate steps would be to add calibration (or auto calibration preferably) to set up the iWiimote for specific applications.

## REFLECTION

After drastic changes due to Covid-19, this project started as a lightsaber game played on the internet. The idea seemed to specific, so we decided to make a generalized controller with defined motions, such as shaking the device. With user testing and the addition of touch input, we narrowed down the supported gestures.

Along the way we worked with software none of us expected to encounter in an Interactive Technologies class, such as networking and encrypted sockets. Learning new things, like sending encrypted messages to iOS devices, and building a generalized paradigm for message passing, we made the project cross-disciplinary within Computer Science. Since the data we transmitted was very high resolution and the latency was very low, we did not need to lean to heavily on signal processing.

In the end, we produced a user interface with great potential. You can try it out for yourself at https://github.com/QuinnCiccoretti/iWiimote. We had fun using the controller, and we hope you do too.

## REFERENCES


[1] Apple. 2019. *Requirements for Trusted Certificates in iOS 13 and macOS 10.15*. https://support.apple.com/en-us/HT210176

[2] boppreh. 2020a. keyboard. https://github.com/boppreh/keyboard. (2020).

[3] boppreh. 2020b. mouse. https://github.com/boppreh/mouse. (2020).

[4] Mozilla Contributors. 2019. *DeviceMotionEvent*. https://developer.mozilla.org/en-US/docs/Web/API/DeviceMotionEvent/

[5] Ian Hickson. 2012. *The WebSocket API*. https://www.w3.org/TR/websockets/

[6] S Staacks, S Hütz, H Heinke, and C Stampfer. 2018. Advanced tools for smartphone-based experiments: phyphox. *Physics Education* 53, 4 (may 2018), 045009. DOI:http://dx.doi.org/10.1088/1361-6552/aac05e

[7] Al Sweigart. 2019. *Welcome to PyAutoGUI's documentation*. https://pyautogui.readthedocs.io/en/latest/