**Michael Schultes**

**Project 4 – Report**

# Device Drivers

## Connecting to the ChompApp Device

Using the *'libusb'* library, a session is created, and a list of devices is populated using the *'libusb_get_device_list()'* function. From this list the function *'libusb_open_device_with_vid_pid()'* is used to open the ChompApp device using the known *'vendorID'* and *'productID'* for the device. The list is then freed and the devices in it are unreferenced using *'libusb_free_device_list()'*.

## Read ChompApp data

First, a buffer is created to store the ChompApp data e.g.*'unsigned char* data = (char*)malloc(1);'*. The device is claimed (because you must claim the interface before you do any operation on the device) using *'libusb_claim_interface()'* and a bulk transfer is initiated with *'libusb_bulk_transfer()'*. The bulk transfer reads in data from the device, in this case a single byte of data, and stores it into the buffer for consumption by the joystick driver.

## Creating the Virtual Joystick Device

Using the *'uinput'* library, a new virtual device is opened e.g. *'int fd = open("/dev/uinput", O_WRONLY | O_NONBLOCK);'*. For Linux to recognize this device as a joystick, a button and two absolute axes are created using *'ioctl()'*. Events are also created for the button and axes e.g. '*ioctl(fd, UI_SET_EVBIT, EV_KEY);'* and *'ioctl(fd, UI_SET_EVBIT, EV_ABS);'* respectively. Once the settings for the device are written, it is created e.g. *'ioctl(fd, UI_DEV_CREATE);'*. The device now appears as *'/dev/input/jsX'* in Linux.

## Routing ChompApp data into the Linux Input System

Finally, the data from the bulk transfer is parsed into *emit* statements for the button and axes. *Emits* are a simple struct containing a *timestamp*, the *event type* (button or stick press), *identifier* (name of the button or axis), and *value*. For button presses, a *value* of 1 indicates the button was pressed, 0 indicates it was released. Axis movements *values* are signed shorts, ranging from -32768 to 32767. After all updates have been made, a *sync report* is emitted which routes the state of the joystick to the Linux Input System.

## Testing

The driver was tested using *'jstest'* which was run in a separate window from the ChompApp device and driver. ChompApp was run in the first terminal, a second terminal was connected to the USB subsystem e.g. *'sudo usbip -a 127.0.0.1 1-1'* and the device was started, and then *'jstest'* was run in a third terminal e.g. *'jstest /dev/input/js0'*. As the joystick state changed in the ChompApp device, the output for the '*jstest'* changed accordingly. The driver has no way of exiting and assumes the ChompApp device is always running.

**Code and other information were referenced from the following links on 12/4/2018:**


1. https://blog.marekkraus.sk/c/linuxs-uinput-usage-tutorial-virtual-gamepad/
2. https://www.dreamincode.net/forums/topic/148707-introduction-to-using-libusb-10/
3. https://stackoverflow.com/questions/16032982/getting-live-info-from-dev-input/