Xbox One Controller Driver for Ubuntu 16.04 LTS with Advanced Rumble Support

Noah Steinberg and Jeremy Kielbiski

COMP3000 – Operating Systems

Dr. Michel Barbeau

April 5th, 2017

1 Introduction

1.1 Context

The Xbox One controller works on USB (universal serial bus) by sending and receiving data packets to the Linux kernel. The controller sends button presses and both trigger and thumbstick status outputs to the kernel which handles transferring the necessary data to any applications that support the controller. The kernel sends output to the controller in the form of acknowledgement packets, which answer initial connection-received requests that the controller sends upon plug-in, and rumble packets, which are responsible for different vibration profiles.

Communication with USB devices relies on URBs (USB Request Blocks) that send and receive data to and from specified endpoints on that particular device. The kernel sends output URBs to the controller to request force feedback and acknowledge connection. The controller sends output interrupt URBs which the kernel interprets using a driver.

1.2 Problem

The rumble feature for USB Xbox One controllers is implemented in both Windows 10 but has limited functionality in Ubuntu 16.04 LTS (henceforth referred to as Ubuntu). The problem currently is that Ubuntu has basic support for Xbox One controllers with a built-in kernel level driver named xpad but it has limited rumble functionality, supporting only a weak and strong entire-controller vibration. Xpad is a kernelspace driver that facilitates communication from many Xbox controller variations, which makes it less streamlined than a specific driver intended to only support Xbox One controllers. The xpad driver works on many Xbox controller variations but our skx supports only Xbox One controllers, but provides more functionality to them. Our project was to enhance the existing rumble functionality for specifically Xbox One controllers via USB connection on Ubuntu by developing a driver that maintains controller functionality but supports a greater number of rumble types while occupying less disk space.

Advanced rumble functionality is important because haptic feedback is a critical part of game design as it enhances immersion (Cha et al., 2011). Therefore, having several types of rumble that surpass basic entire-controller vibration allows for specialized feedback for certain in-game events. For example, in a first-person shooter, when the trigger is pulled to fire the player’s gun, having a basic rumble that affects the entire controller would not provide the engagement as a customized rumble where the trigger vibrates more than the rest of the controller. In addition, racing games often implement vibration based on g-forces attained through cornering, braking, or acceleration. Having support for several specialized rumble types, discussed in-depth later on, enhances the user’s experience as it increases the level of realism in the game.

This problem is important to individuals who play video games on Ubuntu as currently the rumble functionality of the Xbox One controller does not meet the same level as players who use Windows or Mac with the same controller because of a lack of advanced rumble types in xpad, the Ubuntu default driver responsible for handling many Xbox controller devices. Our motivation for improving the currently implemented driver is to increase the immersion of players using Xbox One controllers through a USB connection on Ubuntu. Force feedback is a Linux kernel feature which allows force effects to be sent to supported devices. We created a driver, skx, which maintains controller functionality but supports more types of force feedback than xpad.

A userspace driver titled xboxdrv does have several supported rumble types, but it is not at kernelspace and therefore cannot access and manipulate system resources directly, and as a result it must request through system calls all necessary operations. Userspace drivers, by this fact, are therefore slightly slower than kernelspace drivers.

1.3 Result

We implemented a Linux kernel driver that supports all types of Xbox One controllers and has more advanced force feedback support than the default Linux driver xpad. The xpad driver supports only the force feedback type FF_RUMBLE with strong and weak modes. Our driver skx maintains controller functionality and supports FF_RUMBLE strong and weak, FF_CONSTANT, FF_SPRING, and FF_DAMPER. Our skx driver was also more lightweight than xpad, as will be discussed in Section 3.

1.4 Outline

The rest of this report is structured as follows. Background information is presented in Section 2. Our result is discussed in Section 3. Section 4 is the analysis and evaluation of our work. Section 5 concludes with opportunities for future work. Finally, Section 6 discusses team member contributions.

2 Background Information

Our driver was based off the default Xbox controller driver for Linux, xpad (2), a userspace driver, xboxdrv (1), as well as several other sources. The textbook Linux Device Drivers Third Edition (3) provided invaluable insight into the details of how drivers operate. The Linux library *ff-memless* had to be mined (4) to find information on how memoryless devices like the Xbox One controller handle force feedback.

Firstly, basic information on defining, installing, and removing kernel modules was required to proceed with our project. We had to learn how to define and install our own driver and remove the default driver, xpad. In addition, because the driver runs completely in the kernel, several unfamiliar problems appeared, such as learning how to debug inside the kernel. To achieve this, we use the *dev_dbg* to print output to the kernel via KERN_DEBUG. To print out the debugging results to the console, we used the shell command *dmesg --level=DEBUG --follow*.

After we had set up the basic modules, the USB_device and USB_driver data structures and their usage was critical for our driver, as was the proper input and output of interrupt URBs. Furthermore, knowledge of direct memory addressing (henceforth known as DMA) was required to receive input and generate output for the controller. Knowledge of Linux's input reporting system including key and absolute position information from the controller was vital for attaining full functionality for controller input and customized rumble output. In addition, kernel methods and structures from *ff-memless* had to be studied to implement force feedback in our driver. *ff-memless* stands for force-feedback, memoryless. This structure allows the kernel to control the order and timing of force feedback events, and the driver simple must input a single function to handle these events. Furthermore, we must report to the kernel the types of force feedback our driver can handle. The alternative would involve the kernel sending all force feedback packets to the driver, which would handle the scheduling, uploading and timing of events through its own *ff-device* structure.

Intimate knowledge of how the Xbox One Controller communicates was vital to understand what input packets represent each possible controller event, and what output packets trigger certain actions. Through a combination of researching xpad, the Xbox One Controller

Protocol (6), and our own investigation into the behaviour of the Xbox One controller, we managed to map out the purpose of all required packets for our driver. Specifically, our own investigation taught us the packets required for trigger based rumbling, the maximum rumble strength and duration values, and the current positions of the control sticks and triggers.

In regard to the increased functionality of rumble, we have been able to create dynamic rumbles based of off the controller’s trigger positions and stick positions, as well as including rumble inside of the triggers themselves. However, these rumbles are based off the stick and trigger positions at the time the packet is received by our driver. To improve this, we attempted to use a *delayed_work_queue* to delay a set of rumble packets to be sent in sequence, each of them checking the trigger or stick position at their time of sending. The result of this would be one force feedback event from the kernel would allow our controller to send many force feedback packets in sequence that are updated based on the controller’s status. However, this feature was not completed in time, and it remains a future project.

Lastly, a basic understanding of Bluetooth communication allowed us to understand the limitations of our project being able to allow Bluetooth functionality for the Xbox One Controller with Ubuntu. Because the Xbox One controller communicated through a host controller interface, the control of it’s communication lies on the Bluetooth card driver, and the capacity of the OS to send packets over Bluetooth, which is above and beyond the goals for this project.

3 Results

We developed a device driver that runs entirely in kernel space and uses direct memory addressing for reading and writing data. Our *Makefile* compiles the kernel module and puts it in the correct directory. A module is implemented as a container for the driver and is installed using the command *insmod skx.ko* in Terminal. Furthermore, the user of the module muse be sure to remove the default driver using *rmmod xpad* before attempting to use the skx driver, otherwise the Linux kernel will default to xpad.

The default Linux driver responsible for handling Xbox controllers (xpad) has limited rumble functionality, and only supports FF_RUMBLE with two modes, strong and weak, which activate the heavy and weak motors in the controller body respectively. These rumbles are hard-coded for a specified length of time and are not adjusted for different situations. This is sufficient for basic rumble effects, but does not account for the spinning weights in the trigger which can be activated individually for a specialized effect. Most games on Ubuntu cannot use the advanced rumbles we implemented using the force feedback library because there is a lack of standardization in Linux for special force feedback types, possibly because of a lack of proper documentation of rumble packet information. However, with the documentation in our project, there is a possibility for standardization for the Xbox One controller with Ubuntu.

Specialized rumble functionality can be demonstrated using the *fftest* utility, invoked by running *fftest /dev/input/event#* (number of device) on Terminal. We used a module named *jstest-gtk* that shows controller button and stick statuses in a graphical user interface to check that the driver is working correctly and every packet sent from the controller is caught.

In conclusion, our driver expands upon xpad by implementing advanced force feedback capabilities including: controller body and trigger rumble, strong and weak modes for both sets of weights, support for variable vibration length, repetition, timing, and strength. In addition to the previously discussed advantages of skx over xpad as the preferred Xbox One controller driver for Linux, skx occupies substantially less space than xpad. Our driver consumed 68.103%

less memory than xpad (skx = 16.908kB, xpad = 53.009kB). This both increases performance and leaves more storage for other operations.

4 Evaluation

Our driver maintained all xpad driver Xbox One controller functionality while introducing support for several previously unsupported force feedback types (see Appendix). We consider our driver a success as it accomplished the goal of improving rumble support and had a smaller memory footprint than the current driver Linux driver xpad.

Bluetooth is handled through HCI (Host-Controller Interface) and is relies on standard communication, not requiring a device specific driver. In order to implement identical functionality using Bluetooth, we must deal with the Bluetooth card which is entirely different than dealing with a direct driver. In the future, we hope to implement rumble functionality of the Xbox One controller similar to our skx driver with a Bluetooth controller-to-Ubuntu connection.

During our work on the skx driver, we gained important knowledge about kernel debugging techniques and functionality. We learned that features like spinlocks must be carefully used as failing to release a data lock causes the kernel to hang, requiring a restart. For example, *spin_lock( )* does not pose such dangers, but *spin_lock_irqsave( )* blocks all interrupts on a specific processor and should only be used when necessary, which it was for our driver. It was required for skx because we did not want any other process to interrupt packet input and output.

5 Conclusion

We succeeded in creating a Linux kernel driver for Xbox One controllers that had more advanced force feedback than the default driver xpad. The skx driver not only met the functionality requirements of xpad but surpassed it by implementing several previously unsupported rumble types.

5.1 Summary

We achieved our project goals and recognize that some of the extensions envisioned were unobtainable. We created a Ubuntu USB kernel driver for the Xbox One Controller that used direct memory accessing and URBs to read and write data. Furthermore, we extended driver functionality to include several customized force feedback effects that are unavailable on other Xbox One kernel drivers. We discovered that writing a Bluetooth driver was not a feasible goal, as Bluetooth control is handled solely by the Bluetooth card, not by any specific drivers. Bluetooth works natively on systems running Ubuntu and as a result, writing a Bluetooth driver was excluded from our project framework.

5.2 Relevance

Our skx driver exists directly in kernelspace as a module. Using the knowledge gained from the course about device drivers, direct memory access, and modules, we were able to implement a working Linux device driver. Combining our knowledge about the Linux kernel with existing information from several references, we created a functional driver for the Xbox One controller that supports greater force feedback than xpad. Through our own research, we studied the necessary resources and brute-forced packet data to implement specialized Xbox One controller rumble functionality. We obtained knowledge from class, external resources, and through our own packet data-mining. In conclusion, we developed a feature-complete Xbox One controller driver that had greater force feedback capabilities than the default driver xpad with a smaller memory footprint.

5.3 Future Work

Our driver can be improved by implementing *delayed_work_queues*, which would allow for dynamic rumble effect changes dependent on trigger depression in real time. The use of workqueues in this way would facilitate the rendering of advanced force feedback in the form of specialized event-sensitive vibration. For example, a spring effect where controller rumble increases linearly with the percentage of trigger-press would require a constant packet dialogue between the controller and kernel to report trigger values and send adjusted rumble packets without any delay.

6 Contributions of Team Members

Noah Steinberg: Performed research and testing into the exact effects of Xbox One controller packets. Researched through reading Linux Device Drivers the different requirements for a USB device. Assisting programming and debugging the skx driver to meet the same functionality of xpad. Researched and implement Linux kernel force feedback standards. Researched and attempted implementation of workqueues. Final editing of the report for technical consistency.

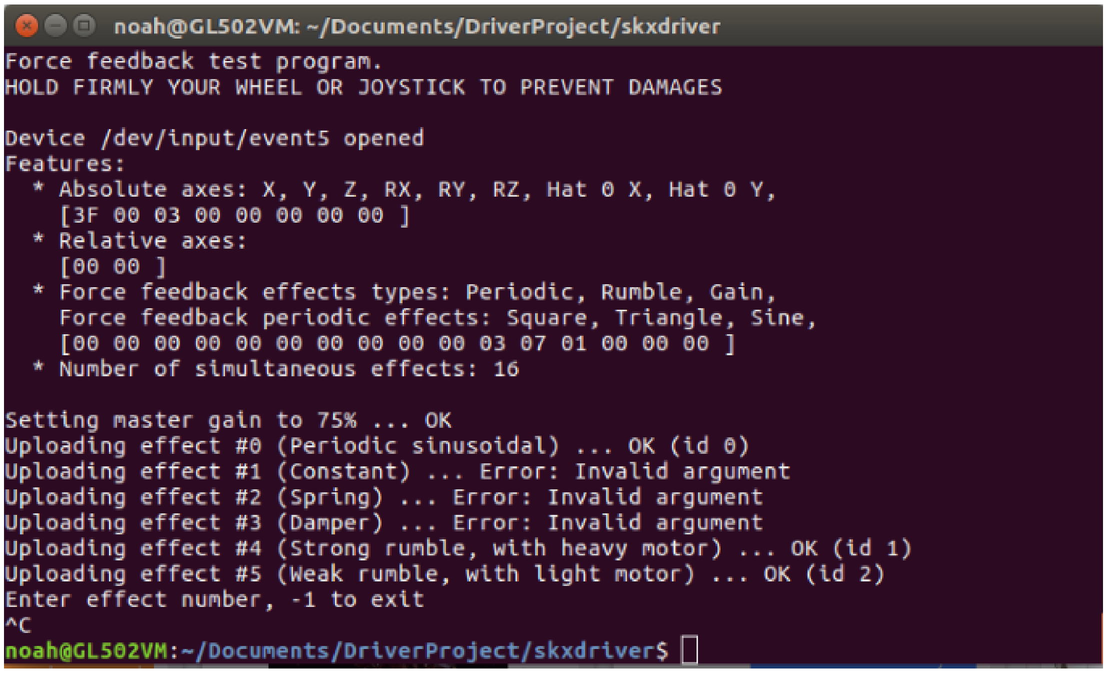
Jeremy Kielbiski: Main programmer for making the skx driver meet the functionality requirements of xpad. Developed rough list of rumble packet outputs. Looked into the viability of using Bluetooth as the controller-Ubuntu connection method instead of USB. Researched and implemented Linux kernel force feedback standards. Wrote the first draft of the report.

References

1. Ruhnke, I. (2015). *Xbox/Xbox360 USB gamepad driver for userspace.* Github. Retrieved from: https://github.com/xboxdrv/xboxdrv
2. Buchbinder, A., Cerquetti, D., Friedemann, M., Fritz, C., Hawkes, I., Kratochvil, J., Lehner, F., Schwartz, O., Toth, S. (2017). *Xpad.* Github. Retrieved from: https://github.com/paroj/xpad
3. Corbet, J., Rubini, A., Kroah-Hartman, G. (2005). *Linux device drivers, third edition.* Sebastopol, CA: O'Reilly Media, Inc.
4. *Linux cross reference.* (n.d.). Retrieved from: http://lxr.free-electrons.com/
5. Cha, J., Eid, M., El Saddik, A., Orozco, M. (2011). *Haptics technologies: bringing touch to multimedia*. Berlin, NY: Springer.
6. Pöyry, P. (2017). *Xbox One controller protocol.* Github. Retrieved from: https://github.com/quantus/xbox-one-controller-protocol

Appendix

Screenshot of *fftest* using the xpad driver, showing that only FF_RUMBLE with strong and weak modes is the only supported force feedback type.

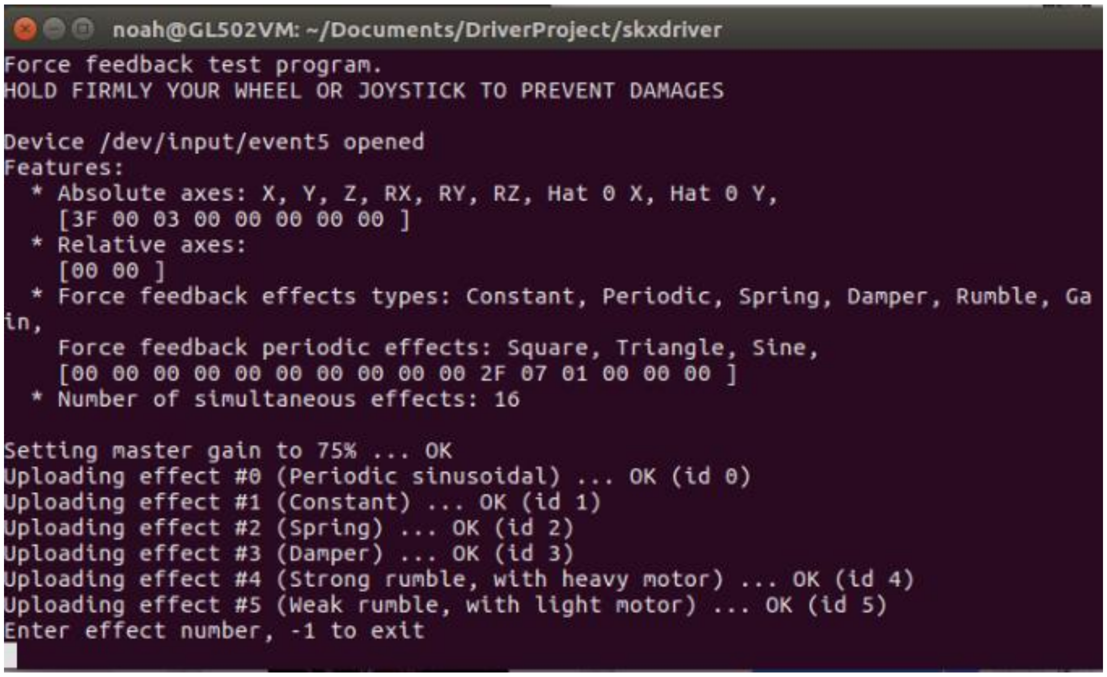


```
noah@GL502VM: ~/Documents/DriverProject/skxdriver
Force feedback test program.
HOLD FIRMLY YOUR WHEEL OR JOYSTICK TO PREVENT DAMAGES

Device /dev/input/event5 opened
Features:
  * Absolute axes: X, Y, Z, RX, RY, RZ, Hat 0 X, Hat 0 Y,
    [3F 00 03 00 00 00 00 00 ]
  * Relative axes:
    [00 00 ]
  * Force feedback effects types: Periodic, Rumble, Gain,
    Force feedback periodic effects: Square, Triangle, Sine,
    [00 00 00 00 00 00 00 00 00 00 03 07 01 00 00 00 ]
  * Number of simultaneous effects: 16

Setting master gain to 75% ... OK
Uploading effect #0 (Periodic sinusoidal) ... OK (id 0)
Uploading effect #1 (Constant) ... Error: Invalid argument
Uploading effect #2 (Spring) ... Error: Invalid argument
Uploading effect #3 (Damper) ... Error: Invalid argument
Uploading effect #4 (Strong rumble, with heavy motor) ... OK (id 1)
Uploading effect #5 (Weak rumble, with light motor) ... OK (id 2)
Enter effect number, -1 to exit
^C
noah@GL502VM:~/Documents/DriverProject/skxdriver$
```

Screenshot of *fftest* using the skx driver, showing that all force feedback types are supported.



```
noah@GL502VM: ~/Documents/DriverProject/skxdriver
Force feedback test program.
HOLD FIRMLY YOUR WHEEL OR JOYSTICK TO PREVENT DAMAGES

Device /dev/input/event5 opened
Features:
  * Absolute axes: X, Y, Z, RX, RY, RZ, Hat 0 X, Hat 0 Y,
    [3F 00 03 00 00 00 00 00 ]
  * Relative axes:
    [00 00 ]
  * Force feedback effects types: Constant, Periodic, Spring, Damper, Rumble, Ga
in,
    Force feedback periodic effects: Square, Triangle, Sine,
    [00 00 00 00 00 00 00 00 00 00 2F 07 01 00 00 00 ]
  * Number of simultaneous effects: 16

Setting master gain to 75% ... OK
Uploading effect #0 (Periodic sinusoidal) ... OK (id 0)
Uploading effect #1 (Constant) ... OK (id 1)
Uploading effect #2 (Spring) ... OK (id 2)
Uploading effect #3 (Damper) ... OK (id 3)
Uploading effect #4 (Strong rumble, with heavy motor) ... OK (id 4)
Uploading effect #5 (Weak rumble, with light motor) ... OK (id 5)
Enter effect number, -1 to exit
```