LagBox – Measuring the Latency of USB-Connected Input Devices

Florian Bockes
University of Regensburg
Franz-Mayer-Straße 1
93053 Regensburg, Germany
florian.bockes@ur.de

Raphael Wimmer
University of Regensburg
Franz-Mayer-Straße 1
93053 Regensburg, Germany
raphael.wimmer@ur.de

Andreas Schmid
University of Regensburg
Franz-Mayer-Straße 1
93053 Regensburg, Germany
andreas1.schmid@stud.uni-regensburg.de

Abstract
High latency in an interactive system limits its usability. In order to reduce end-to-end latency of such systems, it is necessary to analyze and optimize the latency of individual contributors, such as input devices, applications, or displays. We present a simple tool for measuring the latency of USB-connected input devices with sub-millisecond accuracy. The tool, based on a Raspberry Pi 2 microcomputer, repeatedly toggles a button of a game controller, mouse, or keyboard via an opto-coupler soldered to the button and measures the time until the input event arrives. This helps researchers, developers and users to identify and characterize sources of input lag. An initial comparison of multiple input devices shows differences not only in average latency but also in its variance.

Author Keywords
Latency; input devices; interaction; measurements; hardware; open-source;

ACM Classification Keywords
H.5.2 User Interfaces: Input devices and strategies

Introduction and Motivation
The amount of latency or lag in human-computer interfaces affects how effectively, efficiently, and satisfactorily users interact with a computer system. As shown by Ng et al. [11], users are able to detect the effects of
In the following, we present an overview of previous approaches to measuring the latency of input devices, describe the implementation of LagBox, show initial measurements, and discuss limitations and future work.

**Related Work**

Traditionally, researchers and gamers have been employing two different approaches for measuring the latency of input devices. Many previously presented approaches measure end-to-end latency, i.e. the time difference between an input event (e.g., the user pressing a button) and an output event (e.g., a change of screen contents). This is often done using a video camera. Karesoja and Brewster deliver one good example for this approach [6]. By varying only one part of the processing pipeline – such as the input device that is being used - one may indirectly determine latency differences caused by these changes.

Teather et al. measured the input latency of eight computer mice, a keyboard, and a response box connected to a computer via the serial port, PS/2 or USB [13]. An oscilloscope was used to measure the delay between a (simulated) button press on the device and a signal being emitted to the computer. Casiez et al. - who also give a good overview of related work - present a non-destructive approach for measuring absolute end-to-end latency for different computer mice [1]. Furthermore, they developed a low cost method to measure and characterize the end-to-end latency of a touch system (tap latency) or an input device equipped with a physical button [2]. In the method used by Casiez et al., the time of touch is detected by a vibration sensor, whereas the time of response is detected by a photodiode affixed to the display. Researchers at Google measure touch screen latency with a similar approach using a laser sensor to detect touches [7]. A commonly...
used approach used by hobbyists is to measure the difference in latency of two computer mice by smashing both together so that their mouse buttons are clicked at the same time. Software running on a PC captures the timestamps of the two button events and prints out the difference [14].

Deber et al. [4] present Hammer Time, a tool for measuring the latency of capacitive touch screens. Their approach allows for rapidly generating touch events and measuring system response using a light sensor. While the tool offers high precision due to repeated measurements, it is only capable of measuring end-to-end latency, too.

In summary, common approaches for measuring latency are: 1) measuring end-to-end latency by visual means, 2) building custom circuits to capture touch input and system response, and 3) simultaneously pressing buttons on two input devices in order to measure the difference in their latencies. The first two approaches only allow capturing end-to-end latency, thereby hiding the partial latencies of individual components. The latter approach is inherently imprecise and does not scale.

Implementation

With LagBox we contribute an approach that allows for conducting rapid, repeated measurements of the latency of USB-connected input devices. In order to simplify device design and make the device affordable for a wide audience, we use a Raspberry Pi 2 (abbreviated RPi2 in the following) as the core of LagBox. As button mechanics make it inherently hard to define a precise point in time for the input, we define the start time as the point of time when electrical connection is made by the button. Therefore, all USB devices under test have to be prepared by soldering two wires to one of the button pads on the device. These wires are then connected via a 3.5mm audio jack to the phototransistor side of an LTV817 optocoupler (Figure 2). The optocoupler is connected to one GPIO pin and ground on the RPi2. A resistor protects the optocoupler’s LED against overcurrent (Figure 2). When the GPIO pin is activated, the optocoupler electrically connects both wires, thereby simulating a button press on the tested device. A low-latency C application running on the RPi2 repeatedly triggers the optocoupler and measures the time it takes for an USB packet to arrive from the device. The inherent latency of our measurement setup is in the low microseconds range and therefore has little effect on the measured device latencies which are in the milliseconds range.

Limitations

Currently, the system is limited to reading input events from the Linux kernel’s device interface instead of directly detecting USB traffic. As USB devices are polled by the host computer in intervals of 1, 2, 4, 8, etc. milliseconds – depending on the configuration they report – polling rate has a major effect on actual latency. First measurements where we enforced a higher polling rate, indicate that some devices do offer significantly lower latency in this case. A further limitation of our current approach is that it currently requires physically modifying an input device. While the wires can be desoldered again from the device, our approach is not suitable for testing devices e.g., in a store. Extending our system so that it mechanically presses buttons would allow non-invasive measurements.

First results and discussion

To validate our prototype, several experiments were conducted during the development process where we
tested a small number of input devices. The following devices were tested: three gamepads (Logitech Wingman and two different no-name gamepads using the same DragonRise controller IC), two keyboards (Logitech G15 and Gembird Mini USB Keyboard), and three mice (Logitech G5, G300, and RX250). For each device, we collected 5000 samples. Delays between individual measurements were randomized between 0.1 and 10 ms with an even distribution in order to avoid accidental synchronization between measurement interval and USB polling interval. Great differences exist between devices regarding both latency and consistency (Table 1). Moreover, plotting the kernel density estimates for the latency distributions reveals further peculiarities of certain devices (Figure 3). For example, the Logitech G15 keyboard has a bimodal distribution of latencies.

<table>
<thead>
<tr>
<th>Device</th>
<th>Type</th>
<th>Polling rate</th>
<th>Median ± SD (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logitech G5</td>
<td>Mouse</td>
<td>1000 Hz</td>
<td>13.3 ± 2.8</td>
</tr>
<tr>
<td>Logitech RX250</td>
<td>Mouse</td>
<td>125 Hz</td>
<td>2.2 ± 0.3</td>
</tr>
<tr>
<td>Logitech G300</td>
<td>Mouse</td>
<td>1000 Hz</td>
<td>29.0 ± 2.5</td>
</tr>
<tr>
<td>Gembird Mini</td>
<td>Keyboard</td>
<td>125 Hz</td>
<td>25.8 ± 4.9</td>
</tr>
<tr>
<td>Logitech G15</td>
<td>Keyboard</td>
<td>1000 Hz</td>
<td>3.9 ± 0.7</td>
</tr>
<tr>
<td>Logitech Wingman</td>
<td>Gamepad</td>
<td>125 Hz</td>
<td>5.6 ± 2.3</td>
</tr>
<tr>
<td>DragonRise (green)</td>
<td>Gamepad</td>
<td>125 Hz</td>
<td>17.3 ± 4.5</td>
</tr>
<tr>
<td>DragonRise (black)</td>
<td>Gamepad</td>
<td>125 Hz</td>
<td>17.5 ± 4.3</td>
</tr>
</tbody>
</table>

Table 1 – Results of initial latency measurements.

Figure 3 – Latency distribution for various USB-connected mice, gamepads and keyboards measured by our implementation.
Conclusion and outlook
In summary, we have presented a versatile open-source system which currently supports an automated/destructive mode for measuring latency of USB devices, such as mice, keyboards, and game controllers. It can be augmented with non-destructive measurement modes. For example, using an accelerometer similar to Google WALT [7] supplementary to or instead of a force-sensitive resistor might offer greater flexibility. In the future we want to add the approach demonstrated by Casiez et al. [2] and furthermore measure exact timing of “button-pressed” events with a piezoelectric sensor. A mechanical actuator for pressing buttons would allow for automated standardized testing. The preliminary results indicate that devices differ not only in average latency, but that also great differences in the latency distributions exist. These might warrant further investigation.

Acknowledgements
Simon Fürnstein, Oliver Pieper, and Mark Engerißer implemented earlier versions of the prototype and conducted initial latency measurements for their bachelor’s theses.
This project is funded by the Bavarian State Ministry of Education, Science and the Arts in the framework of the Centre Digitisation.Bavaria (ZD.B)

References


