Yet Another Latency Measuring Device

Andreas Schmid
andreas.schmid@ur.de
University of Regensburg
Regensburg, Germany

Raphael Wimmer
raphael.wimmer@ur.de
University of Regensburg
Regensburg, Germany

Figure 1: Setup for measuring end-to-end latency. A microcontroller (a) starts a timer and triggers a button press on a modified input device (b) by closing a button contact with an optocoupler. The application reacts to the button press by changing its background color from black to white. The microcontroller measures the display’s brightness with a photo resistor (c) attached to the monitor. Once a change in brightness is recognized, the timer is stopped. Results sent to the PC via USB.

ABSTRACT
End-to-end latency - the time a computer system needs from an input event until output is displayed - directly influences task difficulty and user experience. It is therefore an important topic in HCI research. Different human-computer interfaces require different ways to measure latency as it is influenced by all involved hardware and software components. However, many approaches to measuring latency rely on professional lab equipment and are therefore hard to replicate. We propose a method for accurately measuring the end-to-end latency of traditional setups with a button-equipped input device and a display. To this end, a microcontroller closes the electrical contact of a mouse button to trigger an input event, and captures the screen response via a photo-resistor. Our approach combines parts of different existing methods to measure latency and only relies on cheap and off-the-shelf components to allow for easy replication. The latency values measured by our device are very close to those measured with a high-speed smartphone camera (240 Hz). The maximum error is below 2.64 ms - lower than the camera’s temporal resolution and the screen refresh periods of high-fps computer displays. Therefore, our approach allows for repeatedly and reliably measuring end-to-end-latency.

KEYWORDS
latency, human-computer interaction

ACM Reference Format:

1 INTRODUCTION
Latency is an inherent property of human-computer interfaces. Due to processing and transmission times between input device, operating system, application, and display, each component in this chain adds to the overall delay between user input and system response. This so called end-to-end latency directly influences task difficulty [18] and should therefore be minimized to ensure optimal user experience. The influence of latency is especially important for real time applications, as it can decide about virtual life and death in video games [7, 15, 29] or influence the outcome of psychological experiments [16, 23]. Shneiderman et al. [28] recommend a system response time of 50 to 150 milliseconds for simple tasks like typing or pointing and Seow [27] classifies response times of 200 milliseconds and shorter as immediate. MacKenzie and Ware have shown that the influence of latency on pointing tasks is easily measurable at 75 milliseconds and delay times of 225 milliseconds degrade performance severely [18]. However, users can perceive significantly lower latencies such as 10 milliseconds for touch input [19, 20] or 2 milliseconds for dragging on a touch screen [19].

To build and test low latency systems it is important to have a method for measuring a system’s latency. As there are numerous different input and output modalities for human-computer interfaces, there is no general one size fits all method for measuring end-to-end latency. Therefore, measuring methods have to be tailored specifically for the system under test and with regards to further aspects such as accuracy, automation, replicability, and access system components.
In this work, we present an accurate, automatic and easily replicable method for measuring end-to-end latency of traditional desktop setups with a button-equipped input device and a visual display.

2 RELATED WORK

A straight forward approach for measuring end-to-end latency is to use a high speed camera to record input and output device and calculate the delay time by either counting frames between two events, or measuring differences in phase or distance between physical pointer and virtual cursor.

He et al. [11] measured the latency of a virtual environment by moving a pointing device over a grid and counting the number of grid cells between pointing device and a displayed cursor in a video recording. A similar approach used by Pavlovich and Stuerzlinger [22] to move a computer mouse along the side of a CRT monitor and measure the distance between mouse an cursor in a video recording. Other methods use regular motion by for example attaching an input device to a pendulum [17, 31, 33] or a turntable [32], logging the points in time when either the moving input device or the corresponding cursor reach their amplitude point an calculating the difference in phase between them. This is also often done manually by annotating video recordings.

Bérard and Blanch [3] argue that a system’s latency can vary and instead of reporting a single latency value, a latency distribution should be measured. Therefore, repeated measurements are necessary, which makes the camera-based approach even more time consuming [10]. They propose two methods for measuring latency of touch screen systems. For their camera-based high-accuracy approach, the distance between a human operator’s finger and the cursor position is measured to determine latency. For their low-overhead approach, a operator follows the circular motion of a displayed object with their finger and the distance between touch point and the theoretical object position is calculated. This method is prone to errors because of the human in the loop. Cattan et al. [6] extend Bérard and Blanche’s method by attaching a visual marker to the operator’s finger so its position in a video recording can be automatically determined with computer vision algorithms.

Because of the cumbersome process of manually analyzing videos and the limited temporal resolution of high speed cameras (1 ms for 1000 fps, 4 ms for 240 fps), fully automatic and highly accurate methods based on electrical components have been proposed.

With HammerTime!, Deber et al. [8] present a non-invasive method for measuring end-to-end latency of any system with a capacitive touch screen. A touch event is triggered by connecting a contact placed on the screen to electrical ground. A photo diode attached to the screen detects the system’s response to the touch event. The time between input event and system response is reported as a latency measurement. This process can be automated to conduct a measurement series and report latency distributions. The method has been validated using an oscilloscope and a high speed camera and it is accurate with a resolution of at least 1 ms.

Kämäräinen et al. [14] extend Deber et al.’s method by adding the capability to simulate external input devices with a microcontroller, using other sensors besides the touch screen and measuring network latency. Furthermore, they collect timestamps for various steps of the processing pipeline to report partial latencies.

Casiez et al. [5] present a different approach for measuring end-to-end latency of a touch screen device. The time of the input event is determined with a vibration sensor attached to a user’s finger. Therefore, this method can also be used for button-equipped input devices without a touch screen. Similar to other methods, a photo diode is used to detect the system’s response via a change in brightness on the display. Casiez et al. also report partial latencies gathered with software and hardware probes included in the system.

Other approaches focus on measuring the latency contributed by individual components of the processing pipeline. This allows for finding and counteracting performance bottlenecks in the system. Stadler et al. [30] present a device for measuring the response time of displays. Similar to methods discussed earlier, this approach uses a photo diode to detect brightness changes on a screen.

Casiez et al. [4] measured the latency of computer mice by positioning the mouse sensor on the computer screen which displayed a moving texture. The time between moving this texture and the reception of the input event was considered as a latency measurement. Measuring the latency of button-equipped input devices often requires inserting hardware probes into the device. Burke [2] measured the latency of computer mice by detecting the point in time when a button contact is closed and measuring the delay until a USB event is received by a logic analyzer. Wimmer et al. [1, 34] extend this approach by replacing the professional grade lab equipment with a Raspberry Pi and automatically triggering buttons so measurement series can be conducted automatically. For this method, it is required to attach wires to a button of the device under test by either soldering or using clamps.

In 2020, NVIDIA revealed several measuring devices they use for evaluating hardware during their internal evaluation process. The Latency Display Analysis Tool (LDAT) [25] senses button presses on a mouse electrically and detects changes on a display with a photo sensor. However, the LDAT has only been sent to a selection of hardware reviewers and its internal workings have not been published. Therefore, this device can not be obtained, replicated or validated by researchers or interested laypersons. NVIDIA also released the Reflex Latency Analyzer [24], which adds the capability to measure latency to new commercial monitors of different manufacturers. A computer mouse is connected to a dedicated USB port of a monitor supporting the technology. Timestamps of input events are logged an time is measured until pixels on the display are updated. Even though this method is convenient in many scenarios, it has several disadvantages. First of all, the technology does not really measure end-to-end latency as the measurement starts when an input event arrives at the system. Therefore, the input device’s internal latency is not taken into consideration. Furthermore, Reflex can only be used with monitors supporting the technology so it can not be applied to existing systems.

3 METHOD

Whereas for capacitive touch screens, input events can be easily triggered by grounding a probe on the screen [8, 14], it is not trivial to accurately measure the exact time of a physical button being pressed [13, 21, 34]. Therefore, we present a method to measure end-to-end latency of traditional desktop setups (button-equipped input device, computer, display) which combines parts of multiple
approaches from related work. By automatically triggering the buttons of modified input devices, as proposed by Wimmer et al. [1, 34], we can precisely control the moment when an input event is triggered. A photo sensor attached to a screen is used to detect a change in brightness corresponding to the system’s response. This method is considered as reliable and has been used in numerous latency measuring setups [5, 8, 9, 14, 26, 30]. Our method is similar to Schubert et al.’s response box [26], but it is compatible with all kinds of button-equipped input devices and all measurements are made on a microcontroller, so no dedicated software is needed.

As both, triggering the input event and measuring the screen’s brightness, can be done with a microcontroller, our measuring process is fully automated. This allows for repeated measurements so not only a single value but also a distribution of latencies can be measured. Furthermore, the tedious frame-by-frame analysis of slow motion videos, as needed for other methods, is not required for our approach. As all used components operate at least on a microsecond time scale, the accuracy of our method is only limited by the used microcontroller. As for example Arduino’s micros() function resolves in 4 µs steps¹, the device is more than accurate enough for measuring end-to-end latency of human-computer interfaces.

3.1 Components

For a sample implementation of our measuring device, we use an Arduino Micro² because of its small size and breadboard-friendly design. But, as only simple components are used and many current microcontroller platforms support Arduino code, the device can be easily built based on other microcontrollers.

The input device of the system under test has to be modified slightly so button presses can be triggered automatically. Just like Wimmer et al. [1, 34], wires are attached to the button’s contacts by soldering or with clamps. Those wires are connected to the transistor side of an optocoupler³ which connects the wires and therefore closes the device’s button when triggered by the microcontroller.

For the photo sensor detecting the system’s response, we use a cheap photo resistor⁴. The sensor is simply attached to the system’s monitor with electric tape (dark colors are recommended to shield from ambient light) and connected to an AC pin of the microcontroller via a voltage divider circuit.

Additionally, we connected a physical switch to the Arduino which allows for starting and stopping the measuring process manually so the device does not trigger input events when it is not supposed to. We also added two LEDs to the device which indicate the moments when the button is closed and when a change in brightness is detected. Those were only used to evaluate our device’s accuracy with a high speed camera and are therefore not required.

We soldered all components on a perfboard for higher robustness, but the components can also be connected using jumper wires on a breadboard. The complete device can be seen in Fig. 2. The total cost of this implementation was under 25€ (excluding the modified input device.)

Figure 2: Close up view of the measuring device. The button contacts of the modified input device are connected to the transistor side of an optocoupler (a). The photo resistor is connected to an ADC pin of the microcontroller via a voltage divider circuit (b). Two LEDs indicate the time when a button press is triggered (c, left) and when a change in brightness is detected (c, right). A physical switch (d) can be used to manually start and stop the measurement.

3.2 Measuring Process

The system under test runs an application which displays a black region on the spot where the photo sensor is attached. When an input event occurs, the region’s color is switched to white until the button is released. Therefore, the measuring process can be included into existing applications by adding a region with this behavior. As the display refreshes from top to bottom due to vertical synchronization (Fig. 3), the vertical position of the photo sensor on the monitor influences the measured latency significantly. This means that for a 60 Hz monitor, the difference in latency between the top and the bottom of the display is 16.67 milliseconds. Therefore, the correct position of the sensor depends on the application: placing it at the bottom right corner leads to a pessimistic measurement, whereas placing it at the top left corner leads to a best case measurement. Chapter 10.3 of the Information Display Measurement Standard [12] the center of the screen is recommended for standardized measurements of display response time.

When the measuring device is turned on, it goes through a quick calibration step by measuring the brightness of the display (black value), triggering an input event and measuring the brightness again (white value). The average of those two values is used as a brightness threshold for the measuring process. The monitor’s brightness should always be set to the maximum value as the pulse width modulated backlight can confound brightness measurements with the photo sensor.

The actual measuring process begins with the microcontroller storing the current timestamp in a variable and then closing the input device’s button by triggering the optocoupler. The microcontroller then repeatedly measures the brightness of the display by reading the ADC value of the photo diode. Once the brightness exceeds the threshold determined during calibration, a second timestamp is saved and the time difference is sent to a PC via USB. Then, the optocoupler is turned off (thus releasing the button) and

²https://store.arduino.cc/arduino-micro
³LTV 617
⁴Luna Optoelectronics PDV-P8104
the device waits until the display turns black again. An additional random delay between 100 and 1000 milliseconds (uniform distribution) ensures that the device does not accidentally sync up with the system under test. Afterwards, a new measurement is started.

4 VALIDATION

Figure 3: Step by step visualization of the measuring process. 
(1) The measurement starts. (2) The mouse button is triggered and the green LED is turned on. (3) The input event is recognized by the PC and the screen changes its color from black to white. (4) Once the photo sensor attached to the display measures a change in brightness, the blue LED turns on. This way, the results of our device can be compared to footage from a high speed camera.

To validate the accuracy of our measuring device, we conducted an evaluation in two steps. First, we measured the internal latency added by the measuring device itself by connecting the photo sensor directly to an LED in parallel to the optocoupler. This way, the LED lights up at exactly the same time the optocoupler would trigger the input device. The photo sensor then detects a change in brightness and the measuring process is stopped. In an ideal world, the time measured this way should approach zero.

All measurements were conducted on an HP EliteBook 850 G4 with a Logitech G5 mouse and a Dell U2417h monitor. The test application which reacts to a click and changes the background color was written in C using SDL2.

We conducted a measuring series of 200 individual measurements this way and measured a mean latency of 116.5 µs (min: 112, max: 124)5. As latency of human-computer interfaces is normally in a millisecond time scale, we consider this error negligible.

In a second evaluation step, we captured a series of 99 individual measurements with our device and recorded the process with a high speed camera6. Two LEDs on the device indicate the points in time when (a) the optocoupler is triggered and (b) the brightness threshold is reached. In the video, it was clearly visible that the second LED lights up exactly at the point in time the display underneath the sensor is updated (Fig. 3). By counting the video frames between the LEDs lighting up, we could approximate the actual latency with a resolution limited to 4 milliseconds due to the video’s frame rate.

5 LIMITATIONS AND CONCLUSION

The biggest limitation of our measuring method is the fact that input devices need to be opened and modified so our measuring device can automatically trigger button presses. We know that this might be a deal-breaker in some scenarios, especially for regular gamers who want to know the latency of their own system. We accept this limitation as - in our experience - electrically triggering button presses is the only replicable method to accurately and automatically trigger input events for button-equipped devices.

One limitation that most methods for measuring end-to-end latency have in common is the fact that even though they can measure latency, they can not give an explanation on partial latencies contributing to the total delay. Therefore, if the goal is to effectively improve a system’s latency, more detailed timestamps from the whole processing pipeline have to be collected [4].

Even though our measuring method only works for systems with button-equipped input devices, we do not regard this as a limitation. As stated in the introduction, latency measuring methods have to be built with the tested system in mind and there are no one size fits all solutions. Furthermore, there is plenty of work on measuring the latency of touchscreen devices [5, 8, 14].

In this work, we presented a simple measuring method for end-to-end latency of human-computer interfaces with button-equipped input devices. As the process is fully automated, no manual analysis of video footage or similar is needed, there is no error introduced by human operators and latency distributions can be measured by conducting a measurement series. We have shown that the device is accurate within a resolution of less than one millisecond, which is enough for the evaluation of human-computer interfaces. As the device is based on cheap and off-the-shelf components and the circuit is fairly simple, the device can easily be replicated for a price of under 25€. Further information on the project, as well as circuit diagrams and source code of the device are available at https://hci.ur.de/projects/end-to-end-latency.

This project is funded by the Bavarian State Ministry of Science and the Arts and coordinated by the Bavarian Research Institute for Digital Transformation (bidt).
REFERENCES


