A Method For Feedback Delay Measurement Using a Low-cost Arduino Microcontroller
Lesson learned: Delay Influenced by Video Bitrate and Game-Level

Justus Beyer
Quality & Usability Lab
TU Berlin, Germany

Richard Varbelow
Quality & Usability Lab
TU Berlin, Germany

Jan-Niklas Antons
Quality & Usability Lab
TU Berlin, Germany

Steffen Zander
Quality & Usability Lab
TU Berlin, Germany

Abstract—Studying cloud gaming QoE, we found the video compression bitrate and the played game level to significantly influence the system’s feedback delay. We present a novel low-cost approach to accurately measure the latency between a user command and the system response using the popular Arduino microcontroller platform and conclude with lessons learned for future cloud gaming lab studies.

Index Terms—cloud gaming; feedback delay; QoE

I. MOTIVATION AND INTRODUCTION

In early 2015 we performed a study regarding Gaming Quality of Experience (QoE) using a cloud gaming setup where the audiovisual output of the game is generated on a server, compressed, and streamed over a network to the client and the client’s input commands are sent to the server vice versa. Using different video compression bitrates in the open source cloud gaming platform GamingAnywhere [1] we created varying levels of video quality. While delay is a known factor of Gaming QoE, the study focused on video quality and no variations of delay were intended. We connected the client and server using a Gigabit Ethernet switched network with a network-level delay of 2ms. Over the course of the study we became suspicious, that our system’s feedback delay (i.e., the time duration between user input to system response) varied slightly between the conditions. To measure the feedback delay of our setup, we first employed a method adapted from Kaaresoja et al. [2] who proposed using a high-speed camera to capture both the input device/method and the device output. Due to the known constant frame-rate of the camera, the timespan between input and output can be measured by counting frames of the video. We performed this method using an iPhone 6 camera recording at 240fps but found this method difficult to use and imprecise for multiple reasons: 1) Due to the required effort of manually counting video frames, the possible number of samples was limited. 2) The method’s accuracy is constrained by the limited speed of the camera. 3) It is often difficult to decide objectively in which frame an event (e.g., a mouse click, change of content on a TFT screen, lighting of an LED) occurred, as these often unfold gradually over multiple video frames. In the literature, purely network traffic-based approaches have been proposed [3]. However, these were inappropriate for our use case since we wanted to measure the user-perceivable feedback delay including input, decoding and play-out delays. We therefore investigated more objective (in terms of real starting and end time of feedback, button press to visual reaction) and reliable methods and found a promising approach in using popular Arduino microcontrollers. In the following Section 2 we present our method and describe the test paradigm. Section 3 details results from the cloud gaming setup measurements and briefly illustrates another use case of the method. Section 4 outlines conclusions and future work.

II. MEASUREMENT SETUP

The core part of the measurement setup is an Arduino Micro microcontroller board which is able to imitate a USB keyboard or mouse. This board has programmable input and output pins and an internal clock with a temporal resolution of 4μs 1 which we use to perform accurate delay measurements. In all our tests the general structure of these measurements was equal: We used an external electrical trigger to start a measurement (provided e.g., by a button for manually triggered measurements, or a secondary Arduino microcontroller for fully automated tests). This signal would be used to generate a user input and record a precise timestamp using the microcontroller’s internal clock. The controller would then repeatedly sample the voltage on one of its electrical inputs and calculate the mean of the last n sampled values (n being a configurable window size) and the values’


Fig. 1. Measured feedback delay for two different video compression bitrates in two game levels (200 measurements per condition).
mean squared error. A threshold would then be applied to this error to recognize changes in the signal. In the event of a detected change, another timestamp would be recorded and the time between the two stamps be calculated and output using a serial data connection and on a connected screen.

To emulate a real user’s input for different input modalities, matching approaches have to be chosen: Mouse and keyboard inputs can be directly generated using the Arduino board’s ability to send USB HID (Human Interface Device) events to the connected USB host. This method is applicable for a wide range of USB-capable devices, such as Macs/PCs, iPads, certain Android devices, and game consoles. In other cases an electrical signal generated by the microcontroller can be used as trigger: Simulated touches on capacitive touchscreens can be generated by connecting a conductive object (e.g., a coin, aluminum foil, or anti-static foam) on the screen to the device’s ground potential (e.g., through a relay operated by the trigger signal). Other actuators might be speakers for acoustic input, or servomotors for mechanical pressure (e.g., to operate a button). To monitor a system’s response we used a photoresistor taped to the surface of the screen. Although we just monitored visual feedback, other types of responses can equally be monitored using appropriate sensors.

Measurements using our setup were confirmed using the described high-speed camera approach.

III. USE CASES AND RESULTS

A. Cloud Gaming Delay Measurement

To measure the delay between user input and system reaction in the cloud gaming setup described above, we configured the Arduino to generate a USB mouse click triggering a gunshot on the cloud gaming client. To sense the game’s response, we taped a photoresistor to the spot of the computer’s screen where the gun’s muzzle flash would appear. Contrary to our assumptions, we found the delay to not only vary significantly with different video bitrates (HQ: 10Mbps, LQ: 1Mbps), but also with different game levels in the game Cube 2: Sauerbraten (see Figure 1, error bars: 95% CI). The observed feedback delays spanned a considerable and likely QoE-relevant range from 73ms (LQ, Lost) to 105ms (HQ, Level9).

B. Smartphone Feedback Delay Measurement

To illustrate another use case of the test setup we measured feedback delays of popular Android and iOS devices. With these, we simulated touches using a piece of conductive material (e.g., coin, aluminum foil, or anti static foam) being connected to ground potential. For the measurement we wrote native apps that would turn the screen white during touch events. This was detected by a photoresistor (see Figure 2). The observed feedback delays varied surprisingly strongly between devices and ranged from 45ms (iPad mini 2) to 118ms (HTC One M8) (see Figure 3).

IV. CONCLUSION AND FUTURE WORK

Using the proposed setup, feedback delay can be measured more efficiently and reliably than using a standard high-speed camera. As a lesson learned we found that even unsuspecting video compression parameters and the selection of game scenes may influence feedback delay in cloud gaming. As a consequence, cloud gaming QoE studies might require adding compensational delays in selected conditions to achieve constant overall latency. To support these studies, we release both the source code and schematics of our test setup.

REFERENCES