# LatencyMeasure

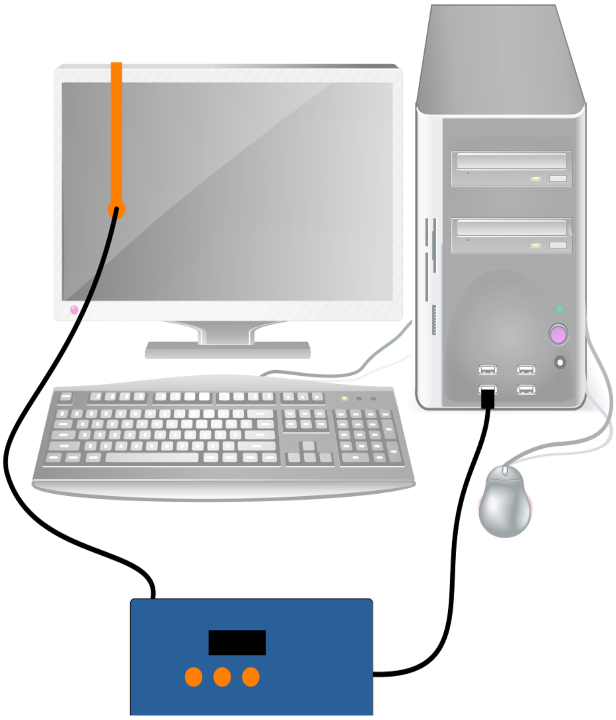
A tool to measure latency
from keyboard or mouse input
to a visual response on the screen.

27. November 2020

# Content

# 1. Introduction

The LatencyMeasure device is used to measure the latency from a human input device (HID) to the response on the screen. This overall latency is sometimes called lag[1], input lag or finger-to-eye-latency.



The measured overall latency is the sum of all signal processing, bus transfers, refresh rates, response times, etc.

The complete project is published on GitHub under

https://github.com/gromeck/LatencyMeasure

The distribution contains the circuit, a PCB layout, a device enclosure, sensor mount, the devices software as well as test clients for Linux.

# 2. Device hardware

The device is based on an Arduino[2] compatible micro controller board. This board has to be capable to emulate keyboard and mouse via USB (32u4- oder SAMD-micro-based controllers).

| Item | Specification |
|---|---|
| Controller Board | SparcFun Pro Micro |
| CPU | ATMEL[3] ATmega32U4 |
| CPU frequency | 8MHz |
| Measurement resolution | 1ms |
| Number of digital inputs used | 3 |
| Number of digital outputs used | 3 |
| Number of analog inputs used | 2 |
| USB connector | yes, for keyboard and mouse HID emulation |
| Power | via USB connector |
| Sensor connector | via cinch connector |

1 https://en.wikipedia.org/wiki/Lag
2 https://en.wikipedia.org/wiki/Arduino
3 https://en.wikipedia.org/wiki/Atmel

# 3. Build the Software

If not already done, clone the repository

```
git clone git@github.com:gromeck/LatencyMeasure.git
```

## 3.1. Software of the micro controller

### 3.1.1. Arduino IDE support for the device

To install the Arduino IDE support for the used micro controller follow the instrutions found here:

```
https://github.com/sparkfun/Arduino_Boards
```

If this extension is installed correctly, the Arduino IDE should offer the board type „`SparcFun Pro Micro 3.3V/8MHz`“. Select it!

After connecting the micro controller to your host, it should appear as „`/dev/ttyAMC<number>`“ or „`/dev/ttyUSB<number>`“. Select it!

### 3.1.2. Adding necessary libraries

Ensure that the following libraries are installed in the Arduino IDE environment:

- Adafruit SSD1306

All other necessary software components are already contained in the Arduino IDE environment.

### 3.1.3. The sketch

Start the Arduino IDE and open LatencyMeasure.ino sketch from the directory `./LatencyMeasure/LatencyMeasure/`.

If the former steps were successfully done, the code can by compiled and linked. After finally uploading the sketch into the controller with the Arduino IDE, the controller should reset and startup with its splash screen and prompting in the main menu.

## 3.2. Test clients

LatencyMeasure comes with a set of different test clients which can be used in different setups. All test clients run under Linux.

### 3.2.1. Ncurses-based test client

This clients can be used in text console environments, or, of course, in the text console of a graphical shell like Xterm or others.

This implementation can be found in `./LatencyMeasure/TestClient/ncurses`.

To build the test client, do

```
./build.sh
```

To build the test client statically, do

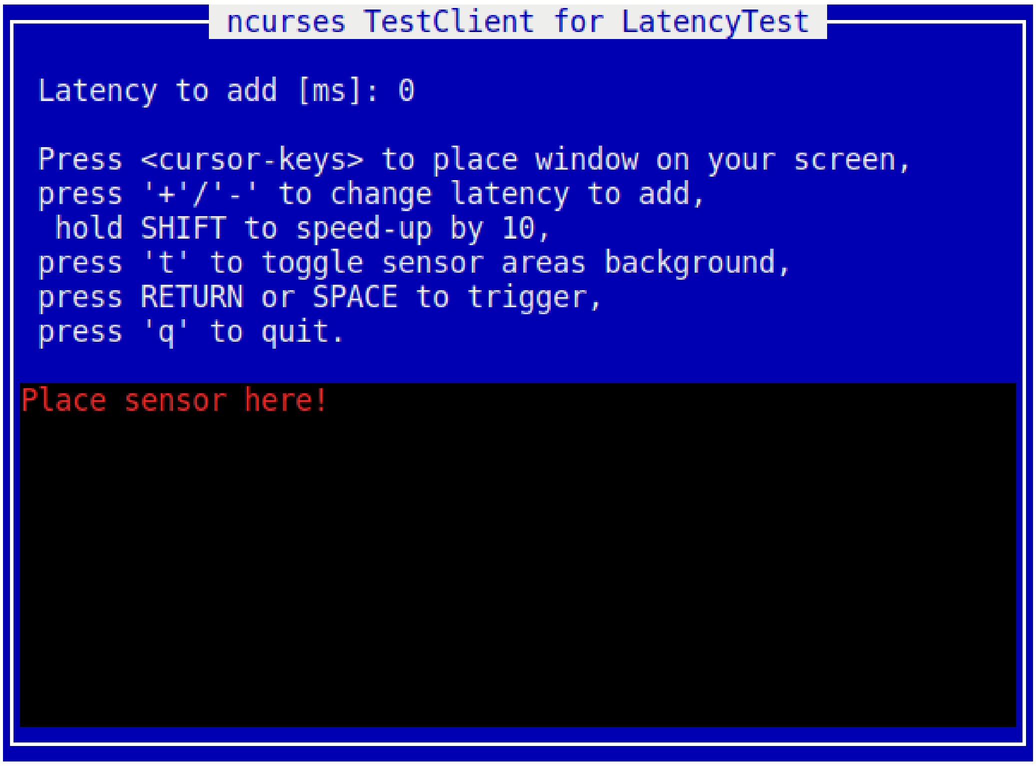
```
./build.sh --enable-static
```

If missing libraries or necessary tools are reported, fix them and rerun.

Finally, the client can be started with

```
./LatencyTest
```

The test client should appear like this:



### 3.2.2. FLTK-based test client

This client can be used in graphical environments like X11.

This implementation can be found in `./LatencyMeasure/TestClient/FLTK`.

To build the test client, do

```
./build.sh
```

To build the test client statically, do

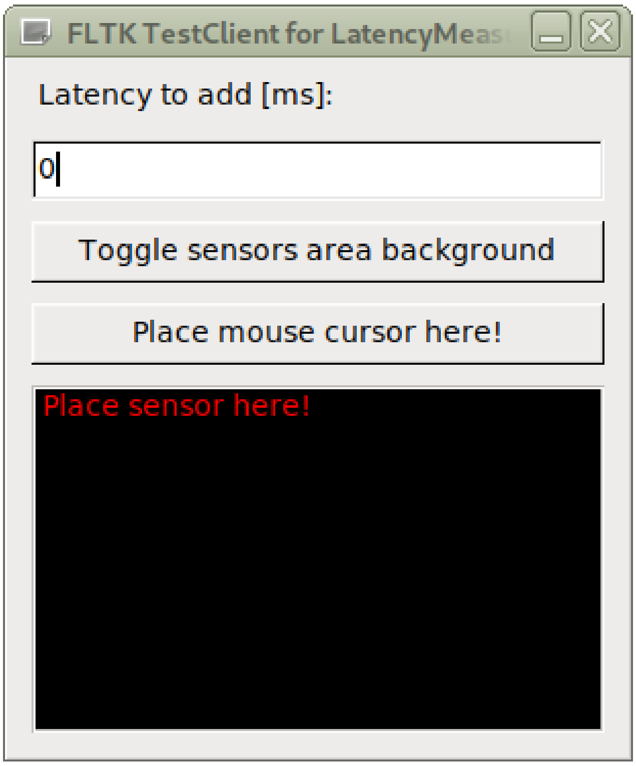
```
./build.sh --enable-static
```

If missing libraries or necessary tools are reported, fix them and rerun.

Finally, the client can be started with

```
./LatencyTest
```

The test client should appear like this:



### 3.2.3. GTK+3.0-based test client

This client can be used in graphical environments like X11 or Wayland.

This implementation can be found in `./LatencyMeasure/TestClient/GTK+3`.

To build the test client, do

```
./build.sh
```

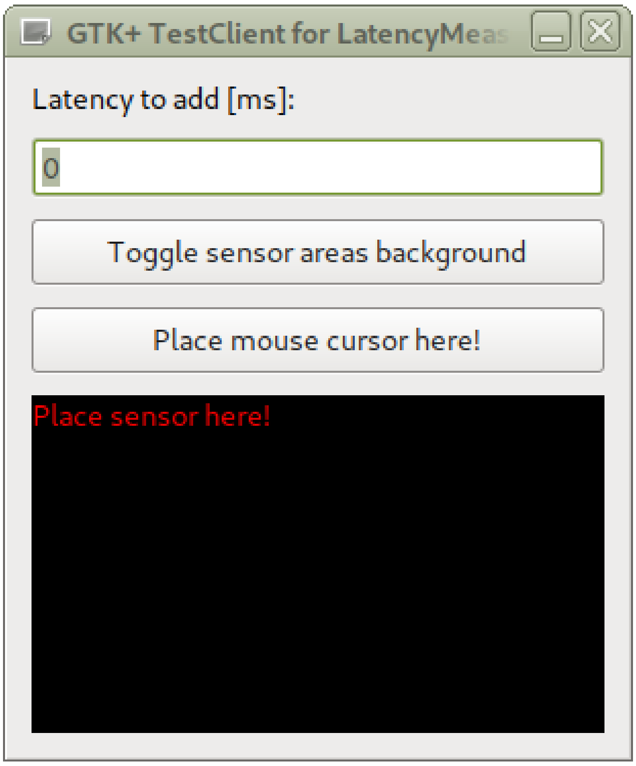
Static linking is not supported.

If missing libraries or necessary tools are reported, fix them and rerun.

Finally, the client can be started with

```
./LatencyTest
```

The test client should appear like this:



### 3.2.4. Qt-based test client

This client can be used in graphical environments like X11 or Wayland.

This implementation can be found in `./LatencyMeasure/TestClient/Qt4`.

To build the test client, do

```
./build.sh
```

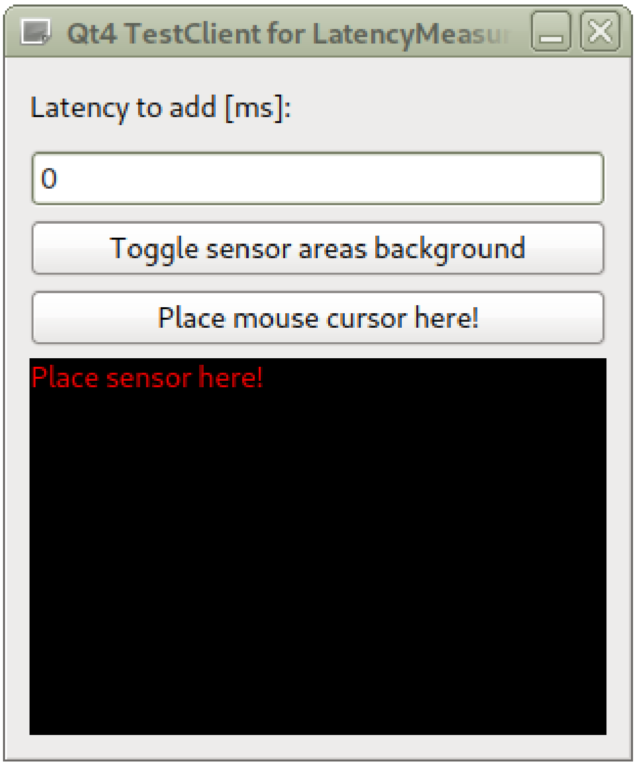
Static linking is not supported.

If missing libraries or necessary tools are reported, fix them and rerun.

Finally, the client can be started with

```
./LatencyTest
```

The test client should appear like this:



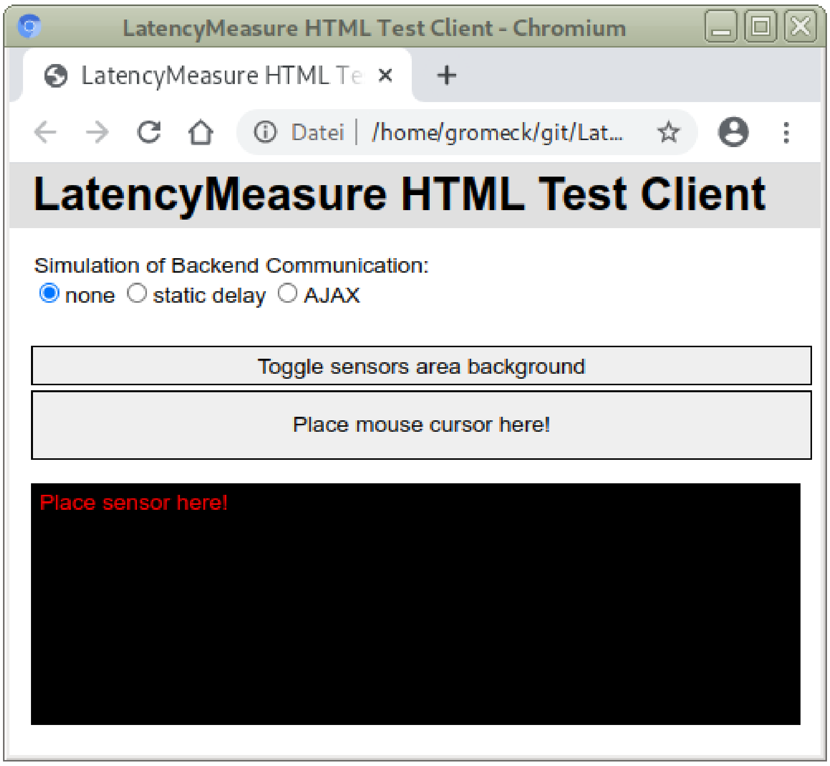
### 3.2.5. Web-based test client

This client is implemented in a single HTML file and makes use of JavaScript.

This implentation can be found in `./LatencyMeasure/TestClient/HTML`.

To run it, simply point a browser to `./index.html` and follow the instructions.

The test client should appear like this:



# 4. Use the device

The following section describes the use of the device from calibration to the measuring procedure.

## 4.1. Connectors

The device has two connectors:

- USB
  The USB port is connected to the system under test. The device will emulate a keyboard respectively a mouse.
- Sensor
  This cinch connector is used to plug in the sensor.

## 4.2. Buttons

The device has four buttons:

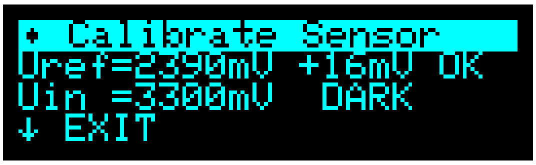
- RST is the reset button which simply restarts the device
  This is helpful when a series of measurements should be stopped.
- MENU is the button to switch between different choices which are shown in to bottom line of the display.
- OK is the button to select the selected option or menu item.
- SCREENSHOT is the button to dump the current display content over the USB link (see section 4.6).
  This button is not available from the outside of the enclosure.

## 4.3. Calibration

A test client is used for the calibration procedure. The test clients always use a change from dark (black) to bright (white) on the screen whenever the key or mouse event occurs. To ensure a correct detection of dark respectively bright on the screen, the devices photo transistor has to be calibrated.

1. Connect the devices USB to power it up.
2. Select “Calibrate Sensor” from the menu and press OK.


3. Start one of the test clients.
4. Place the sensor over the sensor area of the test client.
5. Press OK once more on the device. The device will show the reference voltage ($U_{ref}$) and the voltage of the photo transistor ($U_{in}$) .




6. Use the Button “Toggle sensors area background” in the test client some times. The device will compute the optimal value for $U_{ref}$ and display the delta to the current $U_{ref}$.
7. On the printed circuit you will find the trimmer R2 to adjust $U_{ref}$.
8. Adjust the reference voltage $U_{ref}$ , so that the delta is minimal and “OK” is displayed.

Whenever the sensor area is switched from dark to bright or vice versa, the current detected value should be shown as DARK or BRIGHT on the devices display very spontaneously.

The following voltages should be an orientation:

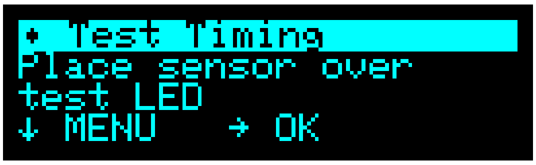
- $U_{in,DARK}$ = 3.300mV for dark background (which is the absolute maximum; adjust the monitors brightness to reach this level)
- $U_{in,BRIGHT}$ = ~300 - ~500mV for bright background (considerably below 1.000mV; adjust the monitors brightness to reach this level)
- $U_{ref,optimal}$ = ~2.700mV

## 4.4. Run the built-in test

The LatencyMeasure device has a built-in test procedure to verify the detection and check the devices response time. The response time should always be <2ms.

1. Connect the devices USB to power it up.
2. Select “Test Timing” from the menu and press OK.


3. Place the sensor over the test LED.
4. Select once more OK to start the verification.




5. The LED lights up with doubling the time between each times.
6. The device measures the time between the flashes.
7. After the test series is done, the overall result is shown.

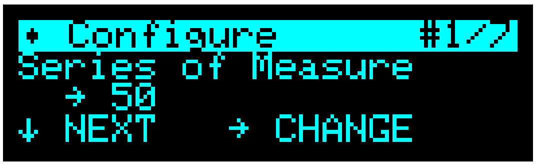

A single failing measurement is still fine, repeat the procedure in that case.

## 4.5. Configure measurement

### 4.5.1. Series of Measure

The device is capable to run series of measurements without user action.

Select values of 1 (single), 10, 25 or 50 measurements.




### 4.5.2. Measure Pause

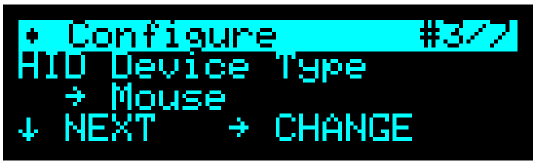
The device will perform a pause between each measurement when doing series.

Select values of 500ms, 1s, 5s or 10s pause.


### 4.5.3. HID Device Type

The device can act as a keyboard as well as a mouse.

Select values of Mouse or Keyboard.




### 4.5.4. HID Mouse Button

If the device operates as a mouse, the emulated mouse button can be configured.

Select values of left, middle or right.


### 4.5.5. HID Keyboard Key

If the device operates as a keyboard, the emulated key can be configured.

Select values of <SPACE>, <RETURN> or <F10>.

### 4.5.6. Timer start at …

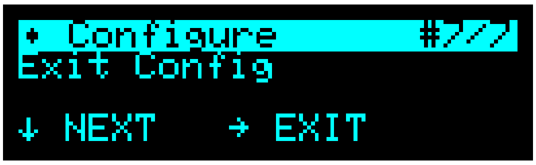
Whenever a key or mouse button event is emulated, the devices first transmits the key/button down event, and after a pause of 50ms the key/button up event.

Some environments react upon the first event (e.g. the text console), some do react on the second one (mostly graphical environments except the GUI requests the key/button down event).

This configuration controls whether timing starts upon the first or second one.


### 4.5.7. Exit config

Select this entry to return to the main menu.



## 4.6. Screnshots

As a special or hidden feature, the device can dump the current content of the OLED display via the serial connection as a PBM[4] bitmap (P1). This format is ASCII-based and uses one bit per pixel.

The screenshot is triggered by the hardware switch 3 on the PCB.

The software distribution contains the script

```
./LatencyMeasure/Ressources/Screenshots/getPBMviaSerial.sh
```

to receive a screenshot from the serial link, and directly convert it to a colorized PNG.

4 https://de.wikipedia.org/wiki/Portable_Anymap

## 4.7. Measure latency

Use one of the test clients to measure the latency in the requested environment.

Ensure that the device was calibrated.

1. Connect the devices USB to power it up.
2. Select "Measurement" from the menu and press OK.


3. Start the test client and setup the sensor to be over the sensor area on the screen.


4. Depending on the configuration, a single measurement or a series of measurements is performed.
5. When the measurement is done, the result is displayed on the device.


The displayed values are

| | |
|---|---|
| `ok` | The number of successfully performed measurements. |
| `total` | The total number of performed measurements. |
| `min` | The smallest value of measured latency in milliseconds. |
| `avg` | The average value of measured latency in milliseconds. |
| `max` | The highest value of measured latency in milliseconds. |

# 5. Interpret measured latency values

## 5.1. Latency from input to output

The following table shows the different steps in the overall chain between input and output.

| Step | Explanation | Typical Latency |
|---|---|---|
| USB | The input is processed from the HID device via a USB hub into the USB host controller, passed over the system bus, queued and processed by the kernels IO-system. | <1ms |
| GUI | The GUI toolkit receives the input from the IO-System, does the event-to-widget correlation, and passes the input as an event to the application. | <10ms |
| Application | The input is processed by the application, special processing is performed, like DB or file operation, computations, or what ever this event should do.<br>The application also triggers the output. | <1ms<br>for quick operations or asynchronous processing |
| GUI | GUI toolkit processes the instruction from the application to initiate a change on the GUI (e.g. drawing the changed widget). | <1ms |
| Network layer | This layer is missing in a local setup.<br>In an environment where the display is forwarded via VNC, RDP, X11, or any other technology, this might produce high latency, as multiple network messages are passed in the processing of events.<br>This layer not always at this point in the stack; this simply depends on the technology. | n * RTT[5] |
| Graphical sub-system | The graphical sub-system (like X11 or Wayland) processes the instruction on base of drawing primitives. These are processed (e.g. clipping, Z-axis, …) and the graphical operations are passed to the graphic card. | <10ms |
| Graphic card driver | This component does the translation from the graphical sub-system into operations that work on the graphic cards video memory. | <1ms |
| Graphic card | The video memory is transferred via the display connection (e.g. VGA, HDMI, DisplayPort, …) to the displaying device. | <1ms |
| Monitor signal processing | The monitor processes the stream of incoming data and pushes this to the LED panel. This doesn't produce a constant latency, as this is done in a certain frequency. | 16.6ms @ 60Hz<br>10ms @ 100Hz<br>6.94 @ 144Hz |
| Monitor display | Finally the response time[6] of the monitor depends on the used technology. This is normally measured in gray-to-gray | >1ms for TN[7] panels<br>>4ms for IPS[8] panels |

5 https://en.wikipedia.org/wiki/Round-trip_delay
6 https://en.wikipedia.org/wiki/Response_time_(technology)#Display_technologies
7 https://en.wikipedia.org/wiki/Thin-film-transistor_liquid-crystal_display#Twisted_nematic_(TN)
8 https://en.wikipedia.org/wiki/IPS_panel

## 5.2. Typical Latency values

The following tables shows the typical overall latencies with the different test clients in typical environments. All measured values are based on a series of 100 single measures.

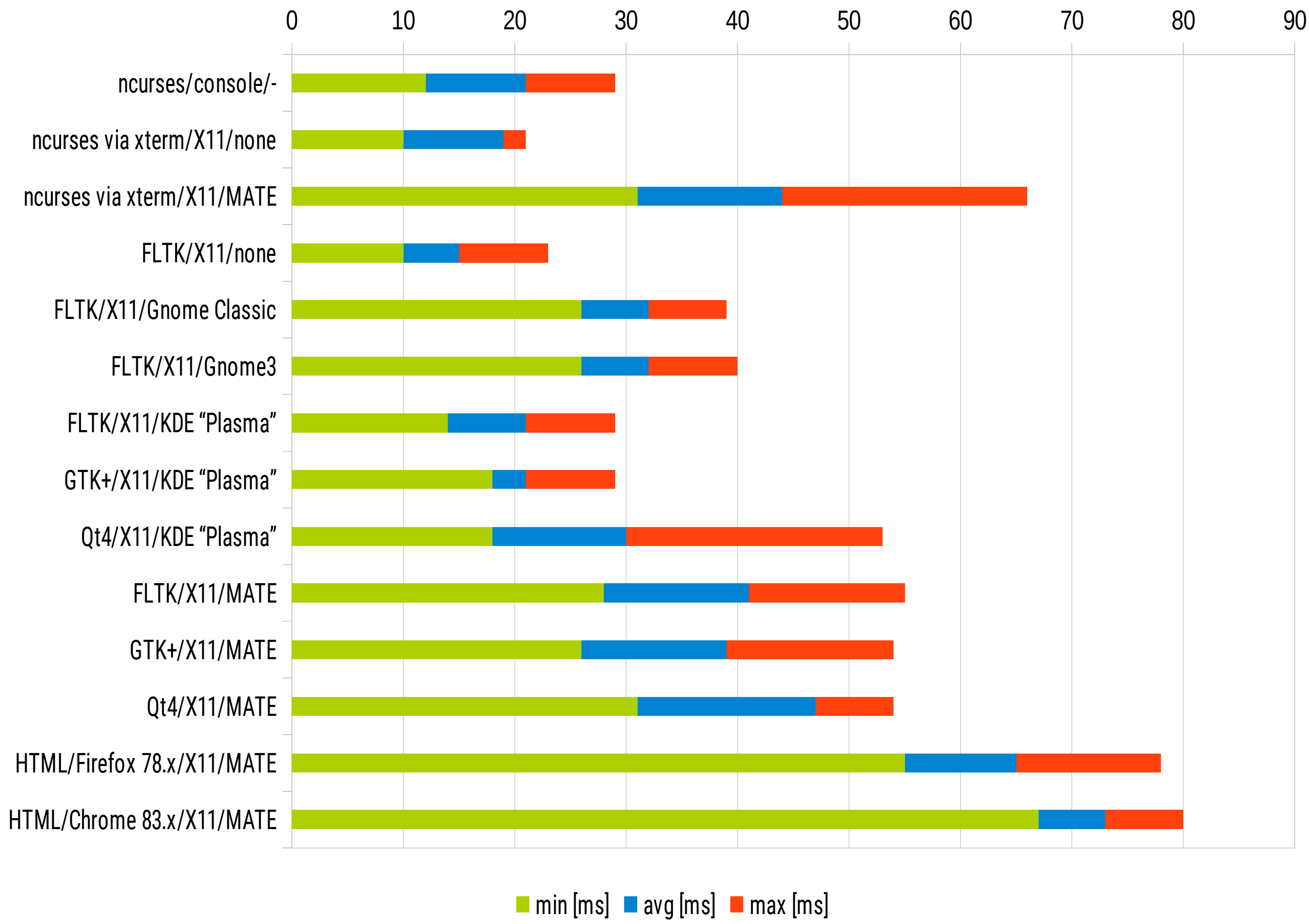
The monitor was always the same (1920x1200 @ 60Hz) and the client was an Intel® Core™ i3-4160 CPU @ 3.60GHz.

As a result of these tests it can be stated that:

- plain X11 is quite as fast as the text console on the frame buffer and ncurses is more or less the fastest HMI
- the used toolkit for the client might have in impact: FLTK and GTK+ are almost equal, where Qt4 brings in a latency of +7 to +9ms
- the desktop environment (esp. the Window Manager) might have an impact compared to plain X11: KDE is quite quick with just +6ms, where Gnome adds +17ms and MATE about +25ms
- web technology introduce a lot more latency as there are a lot more software-layers involved.

| Test Client | Environment | Window Manager | Test Client/Environment/Window Manager | min [ms] | avg [ms] | max [ms] |
|---|---|---|---|---|---|---|
| ncurses | console | - | ncurses/console/- | 12 | 21 | 29 |
| ncurses via xterm | X11 | none | ncurses via xterm/X11/none | 10 | 19 | 21 |
| ncurses via xterm | X11 | MATE | ncurses via xterm/X11/MATE | 31 | 44 | 66 |
| FLTK | X11 | none | FLTK/X11/none | 10 | 15 | 23 |
| FLTK | X11 | Gnome Classic | FLTK/X11/Gnome Classic | 26 | 32 | 39 |
| FLTK | X11 | Gnome3 | FLTK/X11/Gnome3 | 26 | 32 | 40 |
| FLTK | X11 | KDE “Plasma” | FLTK/X11/KDE “Plasma” | 14 | 21 | 29 |
| GTK+ | X11 | KDE “Plasma” | GTK+/X11/KDE “Plasma” | 18 | 21 | 29 |
| Qt4 | X11 | KDE “Plasma” | Qt4/X11/KDE “Plasma” | 18 | 30 | 53 |
| FLTK | X11 | MATE | FLTK/X11/MATE | 28 | 41 | 55 |
| GTK+ | X11 | MATE | GTK+/X11/MATE | 26 | 39 | 54 |
| Qt4 | X11 | MATE | Qt4/X11/MATE | 31 | 47 | 54 |
| HTML/Firefox 78.x | X11 | MATE | HTML/Firefox 78.x/X11/MATE | 55 | 65 | 78 |
| HTML/Chrome 83.x | X11 | MATE | HTML/Chrome 83.x/X11/MATE | 67 | 73 | 80 |

The following graphic gives a better overview of these numbers.




## 5.3. Notes

Whenever you change the monitor, re-calibrate, as different monitors have different brightness.

In a local display setup you will measure latency values with an empirical variance at least of the refresh rate of the monitor.

Whenever a system is under load, the minimum values will almost not change, but the variance will let the maximum value grow. As a result the average value will also increase.

If you have to compare values between different setups – especially when you want to measure network impact – try to use the same monitor. Also check the specification of the other used hardware components (e.g. USB speed, network speed, screen resolution, display standard and its speed, monitor refresh rate and response times).