Radboud Universiteit Nijmegen

Lab Computer T3610

Test results

Author	Pascal de Water
Department	TSG
Date	12-7-2013
Version	1.2

Table of contents

1. Intoduction

This report shows the outcome of four tests on windows 7 64bit and linux ubuntu 64bit. The specifications of the system we tested on are specified in the table below.

Since Windows XP is no longer supported, an upgrade to Windows 7 is inescapable. Windows 7 in contrast to Windows xp is not a real time operating system. For this reason we create a dual boot with a linux.

To find out where the limitations are, we test both mentioned operating systems and report the results. The tests are mainly done in Delphi, PsychToolbox (an add-in for Matlab), Python and Presentation. The TSG officially supports Delphi(professional programmer in-house), Python and presentation(The TSG serves a given course for these last two). With this package we support a broad range of possibilities for our researchers. A program can be pre-programmed(Delphi), Paid software self-programming(presentation), open source software self-programming(python).

With this report comes a technical document describing the hard- and software of the recommended system, along with a systems profile.

Specifications Dell Precision Workstation T3600	
	
CPU	Intel® Xeon® Processor E5-1620 (Quad Core, 3.60GHz Turbo, 10MB)
Memory	8 GB (4 x 2 GB) 1.600 MHz non-ECC DDR3-memory
Harddisk	Western Digital 250 GB 3,5" Seriële ATA-harde schijf (7.200 rpm)
PSU	635W Chassis
Raid-Controller	PERC H310 SATA/SAS-controller
Soundcard	Realtek High Definition Audo On-Board (ALC269Q)
Video card	Nvidia GTX 760
Sound card	Integrated Intel 82579
Monitor	Benq XL2420T @ 120Hz

2. Test Results

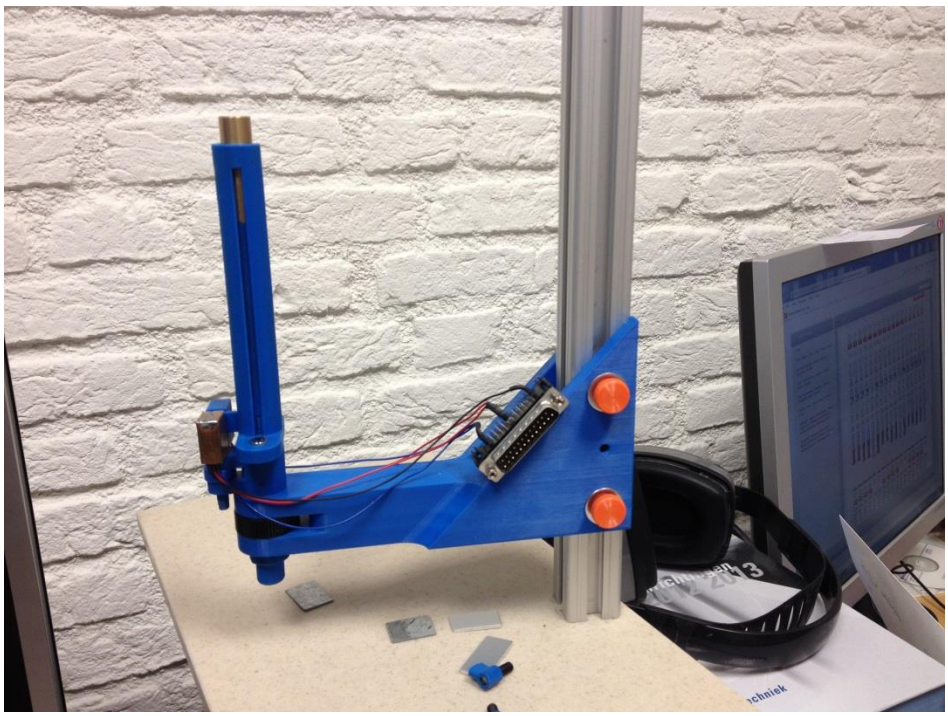
2.1 Description of the test

[1] **Button press sync screen**
button press-> marker up(send 255)->marker down(send 0)->white screen->marker up->black screen-> marker down
[2] **Black white frame drop**
white screen->marker up->black screen-> marker down->continue
[3] **Loopback parrport buttonbox**-
Use loopback connector BB, use arduino and original bb. Send code-> wait code
[4] **button press sound**
marker up->play sound(wave file)->marker down

2.2 Results

Test results						
Test	**psychopy w7@120Hz**	**psychopy linux@120Hz**	**presentation w7@120Hz**	**Delphi w7@120Hz**	**Matlab w7@120Hz**	**matlab linux@120Hz**
button press-> sync screen	< 12 ms	< 10ms	< 12 ms	<12ms?	< 16 ms	< 12ms
black white frame drop	no	no	no	no, direct 2d	frame drop 0,5%	no
loopback parrport buttonbox	2-3ms	2ms	2-3ms	0,3-0,5 ms	0,9- 2,4ms	0,3-2,3ms
button press-> sound	160-240ms	pygame 1000-1600ms pyglet 6-8ms alsa 2-3ms	70-80ms directX software	60ms	18-25ms	50-60ms
			6-8ms presentation exclusive mode			
			75-80ms presentation shared mode			

The Tijn



Test	Windows 7	linux
BITSIbox Duemilanove	1-2ms	2-4ms
BITSIbox uno	2-6ms	2-7ms
BITSIbox Leonardo	1ms	2-3ms
Default toetsenbord	75ms	16-30ms
Logitech toetsenbord G510s	2-8ms	
Logitech mouse G700 wired	2-5ms	
Logitech mouse G700 wireless	3-8ms	

2.3 Code implementation

Find the testing methods/scripts for each programming language.

2.3.1 Psychopy

Button press sync screen

```
def WaitForAllButtonpressTimeBB(DurWait):
    ButtonPressed = 0
    ButtonTime = 0
    ButtonBox.flushInput()
    DurWait = trialClock.getTime() + DurWait
    while trialClock.getTime()<DurWait:
        bytes=ButtonBox.read()
        if bytes:
            ButtonPressed = bytes
            ButtonTime = trialClock.getTime()
            Break

if  runtest == 'ButtonBoxScherm':
    while RunAmountOfTrials<TotalAmountOfTrials:
        Rectangle.setLineColor('black')
        Rectangle.draw()
        MyWin.flip()
        Rectangle.setLineColor('white')
        Rectangle.draw()
        #ButtonBox.flushInput()
        WaitForAllButtonpressTimeBB(2)
        ButtonBox.write('T')
        MyWin.flip()
        #
        ButtonBox.write('S')

        for keys in event.getKeys():
            if keys in ['escape','q']:
                QuitExperiment()
```

Black white frame drop

```
if  runtest == 'flikkerscherm':
    FreqMonintor = 1.0/120.0 + 0.001
    stampwhite=trialClock.getTime()
    stampblack=trialClock.getTime()
    while RunAmountOfTrials<TotalAmountOfTrials:
```

```
    Rectangle.setLineColor('white')
    Rectangle.draw()
    MyWin.flip()
    #logging.log(level = logging.DATA, msg = 'flip white')
    stampwhite=trialClock.getTime()
    if (stampwhite - stampblack) > FreqMonintor:
       print stampwhite - stampblack
    Rectangle.setFillColor('black')
    Rectangle.draw()
    MyWin.flip()
    #logging.log(level = logging.DATA, msg = 'flip black')
    stampblack=trialClock.getTime()
    if stampblack - stampwhite > FreqMonintor:
       print stampblack - stampwhite

    for keys in event.getKeys():
       if keys in ['escape','q',]:
          QuitExperiment()
```

Loopback parrport buttonbox

```
if  runtest == 'ButtonBoxTestLoopback':
   MaxTime = 0.0
   while RunAmountOfTrials<TotalAmountOfTrials:
      #WaitForAllButtonpressTimeBB()
      stampstart = trialClock.getTime()
      ButtonBox.write('\xFF')
      bytes = ButtonBox.read(1)
      stampstart=trialClock.getTime()-stampstart
      ButtonBox.write('\x00')
      #if len(bytes)>0:
      #    if MaxTime < stampstart:
      #       MaxTime = stampstart
      ShowText(stampstart)
      core.wait(0.1)

      for keys in event.getKeys():
         if keys in ['escape','q','space']:
            QuitExperiment()
```

button press sound

```
if  runtest == 'Sound':
   while RunAmountOfTrials<TotalAmountOfTrials:
      WaitForAllButtonpressTimeBB(2)
      ButtonBox.write('\x00')
      #Tock.play()
      SoundMp3.play()
      ButtonBox.write('\xFF')

      for keys in event.getKeys():
```

```
if keys in ['escape','q','space']:
   QuitExperiment()
```

2.3.2 Presentation

Button press sync screen

```
sub RunTrials begin

        loop iTrialCount = 1
        until iTrialCount > iRunAmountOfTrials
         begin
                loop iCountOld = response_manager.total_response_count()
                until response_manager.total_response_count() > iCountOld
                begin end;
                Oport.send_code( 255 );
                Oport.send_code( 0 );

                p_BlackWhite.set_background_color(255,255,255);
                p_BlackWhite.present();
                Oport.send_code( 255 );

                /*loop count_old = response_manager.total_response_count()
                until response_manager.total_response_count() > count_old
                begin end;

                term.print_line(clock.time());

                loop count_old = response_manager.total_response_count()
                until response_manager.total_response_count() > count_old
                begin end;

                term.print_line(clock.time());
                */
                p_BlackWhite.set_background_color(0,0,0);
                p_BlackWhite.present();
                Oport.send_code( 0 );

    end;

end;
```

Black white frame drop

```
sub RunTrials begin

        loop iTrialCount = 1
        until iTrialCount > iRunAmountOfTrials
         begin
                loop iCountOld = response_manager.total_response_count()
                until response_manager.total_response_count() > iCountOld
                begin end;
                Oport.send_code( 255 );
```

```
                Oport.send_code( 0 );

                p_BlackWhite.set_background_color(255,255,255);
                p_BlackWhite.present();
                Oport.send_code( 255 );

                /*loop count_old = response_manager.total_response_count()
                until response_manager.total_response_count() > count_old
                begin end;

                term.print_line(clock.time());

                loop count_old = response_manager.total_response_count()
                until response_manager.total_response_count() > count_old
                begin end;

                term.print_line(clock.time());
                */
                p_BlackWhite.set_background_color(0,0,0);
                p_BlackWhite.present();
                Oport.send_code( 0 );

    end;

end;
```

Loopback parrport buttonbox

```
sub RunTrials begin
Oport.send_code( 0 );
        loop iTrialCount = 1
        until iTrialCount > iRunAmountOfTrials
        begin
                iCountOld = Serial.total_count();
                ShowText("send");
                Oport.send_code( 255 );
                dTimeStamp = clock.time_double();
                iTimeStamp = clock.time();

                loop until (iCountOld < Serial.total_count()) begin
                end;
                dTimeStamp = clock.time_double() - dTimeStamp;
                iTimeStamp = clock.time() - iTimeStamp;
                Oport.send_code( 0 );

                ShowText(string(iTrialCount) +" "+string(dTimeStamp)+" "+string(iTimeStamp));

                loop iCountOld = response_manager.total_response_count()
                until response_manager.total_response_count() > iCountOld
                begin end;
                iTrialCount = iTrialCount + 1;
```

```
    end;

end;
```

button press sound

```
sub RunTrials begin

        loop int i = 1
  until i > block_size
  begin

                loop int count_old = response_manager.total_response_count()
                until response_manager.total_response_count() > count_old
                begin end;

                oport.send_code( 255 );
                GoodSound.present();
                oport.send_code( 0 );


 end;
end;
```

2.3.3 Delphi

Button press sync screen

Black white frame drop

Loopback parrport buttonbox

button press sound

2.3.4 Matlab

Button press sync screen

```
function LabComputerTest_BlackWhiteFromButton(repeats)

if nargin < 1
        repeats = 24;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
% INIT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%

% make sure everything gets cleaned up well
```

```
clean = onCleanup(@()cleanup()); % executes at cleanup of local variable clean

% add PsychToolbox Matlab path
addpath(genpath('c:\toolbox\psychtoolbox'));

% get handle to serial device
arduino = open_buttonbox();

% code snippets from MakeTextureTimingTest2()
% Open standard window:
channels = 3; %RGB
specialFlags = 4;
precision = 0;
height = 1080;
width = 1920;
winpos = []; %[1024 50 1024+width 50+height];

InitializeMatlabOpenGL(0,0,1);

screenid = max(Screen('Screens')); % handle to screen

w = Screen('OpenWindow', screenid, 0, winpos);

monitorFlipInterval = Screen('GetFlipInterval', w);

% Create black and white test pixel matrix:
wimg = zeros(height, width, channels,'uint8');
wimg = double(wimg);
bimg = ones(height, width, channels,'uint8')*255;
bimg = double(bimg);

% Preheat: Screen() may need to allocate internal buffers or create
% shaders - we don't want this one-time setup overhead to spoil the
% numbers:
black_screen_texture = Screen('MakeTexture', w, bimg, 0, specialFlags, precision);
white_screen_texture = Screen('MakeTexture', w, wimg, 0, specialFlags, precision);
% black_screen_texture = 0;
% white_screen_texture = 255;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
% Read stimulus content
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

% future extension, nothing special needed in this test

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
% Loop
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

%BlackWhiteFromButton
%button press
%marker up(send 255)
%marker down(send 0)
%white screen
%marker up(send 255)
%black screen
%marker down(send 0)

%esc = KbName('ESCAPE');
KbQueueCreate;
KbQueueStart;

WaitSecs(0.1);

% call once to load all functions into memory
draw_texture(w,black_screen_texture,0,0,monitorFlipInterval);
T0 = draw_texture(w,white_screen_texture,0,0,monitorFlipInterval);

for n = 1 : repeats
    %wait_for_keypress([],0); % wait for any key pressed, do not wait for it to be released

    IOPort('purge', arduino); % clear serial buffer
    wait_for_buttonpress();

    % send marker up
    IOPort('Write', arduino, uint8(255), 0);
    % send marker down
    IOPort('Write', arduino, uint8(0), 0);

    % black screen
    draw_texture(w,black_screen_texture,1,T0,monitorFlipInterval);
    % send marker up
    IOPort('Write', arduino, uint8(255), 0);

    % white screen
    draw_texture(w,white_screen_texture,1,T0,monitorFlipInterval);
    % send marker down
    IOPort('Write', arduino, uint8(0), 0);

disp(' ');
end


    function wait_for_buttonpress()
    % check button box incoming data
        while 1
```

```
                navailable = IOPort('BytesAvailable', arduino);
                if navailable
                        break
                end
            end
        end

        function cleanup()
            try, IOPort('close',arduino); catch, end

    %Releases queue and other resources allocated by KbQueueCreate
            KbQueueRelease;

            % release textures
            try
                Screen('Close', black_screen_texture);
                Screen('Close', white_screen_texture);
            catch
            end

            % close screen
            sca();

        end
end
```

Black white frame drop

```
function LabComputerTest_BlackWhiteScreen(repeats)

if nargin < 1
        repeats = 24;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
% INIT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%

% make sure everything gets cleaned up well
clean = onCleanup(@()cleanup()); % executes at cleanup of local variable clean

% add PsychToolbox Matlab path
addpath(genpath('c:\toolbox\psychtoolbox'));

% get handle to serial device
arduino = open_buttonbox();

% code snippets from MakeTextureTimingTest2()
```

```
% Open standard window:
channels = 3; %RGB
specialFlags = 4;
precision = 0;
height = 1080;
width = 1920;
winpos = []; %[1024 50 1024+width 50+height];

%InitializeMatlabOpenGL(0,0,1);

screenid = max(Screen('Screens')); % handle to screen

w = Screen('OpenWindow', screenid, 0, winpos);

monitorFlipInterval = Screen('GetFlipInterval', w);
%monitorFlipInterval = 1/59.8834;

% Create black and white test pixel matrix:
wimg = zeros(height, width, channels,'uint8'); wimg(:,:,1)=255;
wimg = double(wimg);
bimg = ones(height, width, channels,'uint8')*255; wimg(:,:,[1 3])=0;
bimg = double(bimg);

% Preheat: Screen() may need to allocate internal buffers or create
% shaders - we don't want this one-time setup overhead to spoil the
% numbers:
black_screen_texture = Screen('MakeTexture', w, bimg, 0, specialFlags, precision);
white_screen_texture = Screen('MakeTexture', w, wimg, 0, specialFlags, precision);
% black_screen_texture = 0;
% white_screen_texture = 255;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
% Read stimulus content
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

% future extension, nothing special needed in this test

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
% Loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

%BlackWhiteStream
%white screen
%marker up
%black screen
%marker down
```

```
%continue

% Bump priority for speed
priorityLevel=MaxPriority(w);
Priority(priorityLevel);

% call once to load all functions into memory
tic
draw_texture(w,black_screen_texture,0,0,monitorFlipInterval);
T0 = draw_texture(w,white_screen_texture,0,0,monitorFlipInterval);

tic
for n = 1 : repeats
        % white screen
%tic;
        draw_texture(w,black_screen_texture,1,T0,monitorFlipInterval);

        % send marker up
        IOPort('Write', arduino, uint8(255), 0);

        % black screen
        draw_texture(w,white_screen_texture,1,T0,monitorFlipInterval);

        % send marker down
        IOPort('Write', arduino, uint8(0), 0);
%disp(' ');
end

% set priority back to normal
Priority(0);


        function cleanup()
    try
       IOPort('close',arduino);
    catch
    end
                % release textures
                try
                        Screen('Close', black_screen_texture);
                        Screen('Close', white_screen_texture);
                catch
                end

                % close screen
                sca();

                % set priority back to normal
                Priority(0);
```

```
        end
end

function wait_for_buttonpress()
% check button box incoming data
% ....

end
```

Loopback parrport buttonbox

```
function LabComputerTest_SerialLoop(device)

if nargin < 1,     device = []; end

% add PsychToolbox Matlab path
addpath(genpath('c:\toolbox\psychtoolbox'));

% make sure everything gets cleaned up well
clean = onCleanup(@()cleanup()); % executes at cleanup of local variable clean

KbQueueCreate;
KbQueueStart;

WaitSecs(0.1);

% get handle to serial device
arduino = open_buttonbox(device);

Min=inf;
Max=0;
Som=0;
N=0;
while 1

        % send arbitrary byte
        IOPort('Write', arduino, uint8(0), 0);
        % clear buffer
        IOPort('purge', arduino);
        tic
        IOPort('Write', arduino, uint8(255), 0);

        % wait until it is received again, start polling for characters (indicating start of scan)
        wait_for_buttonpress();
        t = toc;

  if t>Max, Max=t; end
  if t<Min, Min=t; end
  Som = Som + t;
  N = N + 1;
```

```
        % send marker
        IOPort('Write', arduino, uint8(0), 0);

        fprintf('Loopback time: %f\tmin=%f\tmax=%f\taverage=%f\n',t,Min,Max,Som/N);

        % make sure IOPort/Matlab is ready to process next byte, for some weird
        % reason without this pause, bytes are being sent/received much slower.....
        WaitSecs(0.1);

end %while 1


        function wait_for_buttonpress()
        % check button box incoming data
                while 1
                        navailable = IOPort('BytesAvailable', arduino);
                        if navailable
         [newdata, when, err] = IOPort('Read',arduino,0,navailable);
%           if any(newdata>0)
             disp(newdata);
             break
%           end
                        end
                end
        end

        function cleanup()
                try, IOPort('close',arduino); catch, end

    %Releases queue and other resources allocated by KbQueueCreate
                KbQueueRelease;

        end
end
```

button press sound

```
function LabComputerTest_Sound(files)

% add library custom oal functions
addpath(fullfile(fileparts(mfilename('fullpath')),'audio'));

% add PsychToolbox Matlab path
addpath(genpath('c:\toolbox\psychtoolbox'));

% make sure everything gets cleaned up well
clean = onCleanup(@()cleanup()); % executes at cleanup of local variable clean

% get handle to serial device
%arduino = open_buttonbox();
```

```
% load file
if nargin < 1
	files = {'100HZ.WAV','1000HZ.WAV','2000HZ.WAV','bop_800.wav','2KHZright.WAV'};
else
	files = cellstr(files);
end
audio.files = files;

% get this functions folder, it should also contain the audio files
audio.folder = fileparts(mfilename('fullpath'));

% init open audio
audio = init_OAL(audio);

% start playing
for n = 1 : numel(audio.source)
	% send marker
	IOPort('Write', arduino, uint8(255), 0);
	alSourcePlay(audio.source(n));
	% send marker
	IOPort('Write', arduino, uint8(0), 0);
	pause
end


	function audio = init_OAL(audio)
		global AL

		% use N buffers (size of eeg buffering AND delay!)
		audio.N = numel(audio.files);
		% gain
		audio.Gain = 1.0;

		try
			% make sure it is closed
			CloseOpenAL; WaitSecs(0.1);
			% Initialize OpenAL subsystem, no debuglevel and using default output
device
			InitializeMatlabOpenAL(0); % puts globals AL, ALC in callers workspace!
			% clear OAL error state
			alGetError();
			% Generate required sound buffers
			audio.bufptrs = alGenBuffers(audio.N);
			% Create one sound source
			audio.source = alGenSources(audio.N); % each file its own source
			% Set emission volume to 100%, aka a gain of 1.0:
			alSourcef(audio.source, AL.GAIN,  audio.Gain);
			% fill buffer(s) with audio content
			%empty_buffers(audio,1:audio.N);
			for n = 1 : audio.N
```

```
                        file = fullfile(audio.folder, audio.files{n});
                        add_audio_buffer(audio.bufptrs(n),file,44100,1.0);
                end
                % attach buffer sequence to source
                for n = 1 : audio.N % only one audio file for now
                        buf = audio.bufptrs(n);
                        alSourceQueueBuffers(audio.source(n), length(buf), buf);
                end
                %               % number of queued buffers
                %               audio.qbufs =
alGetSourcei(audio.source,AL.BUFFERS_QUEUED); % queued buffers

        catch err
                e = get_al_error();
                if ~isempty(e), disp(e); end
                disp(err.message);
                error('Failed to initialize audio source');
        end
    end

    function cleanup()
        CloseOpenAL;
        IOPort('close',arduino);
    end
end
```