Digital Signage Player

SLQ Wiki Fabrication Lab 2026/04/24 08:26

Digital Signage Player

This is where we are documenting our long and evolving search for the perfect fabrication lab signage. The signage will/has been used for basic info, but ideally can be adapted for use with specific equipment, quickly edited in-house and easily updated.

A brief explanation of each of the solutions we have trial or research is given, then detailed instructions are provided where appropriate. Resources for the signage can also found here.

Prototype One - The Pi-looper

IMPLEMENTED - in occasional use for installations

The pi-looper prototype was explored in 2015-2016 as a solution for signature programming by Daniel. It has two versions, one that uses video on the SD card, a second that plays back of a USB. Other options include output of sound over HDMI or minijack.

The Signage was used in the fabrication Lab in 2015 before the refurbishment.

Credit goes to Steven Hickson for this work, This is the latest info.

http://stevenhickson.blogspot.com.au/2015/04/rpi-videolooper-not-booting-blinking.html

Source is here.

https://github.com/StevenHickson/RPiVideoLooper

Check out the pi-maker documentation to make your own pi looper.

PROS:

- Existing solution
- Documented
- Works
- Minimal hardware spec (pi2)

CONS:

- Only video files.
- Limited Customisation

Verdict

Useful for stand-alone player only. Not updated easily or web-based.

Prototype Two - The Chromecast

STALLED - with ICTS networking

This was seen as possibly the cheapest way to get a remote screen working. It has stalled due to network requirements which SLQ ICT is working on. It seems the chromecast requires multicast and TCP/UDP set-up outside the scope of usual SLQ wifi networks. This uses a webpage as source which is opened in a browser (chrome) and then 'cast' to the chromecast.

Pros

- Cheap solution
- reliable, available hardware
- will play video
- can be used for presentations

Cons

- Networking set-up is not enterprise friendly (requires multicast and dns)
- Each screen requires a browser on another device (or possibly tabs?)

Prototype Three - The Pi Kiosk + Reveal.js Signage Server

implimented - then discarded due to need for extra node service needed on server

Based on a running a simple web slide show as the signage, with a pi running a web browser in kiosk (full screen) mode.

PROS:

- Existing solution
- Documented
- Works
- Minimal hardware spec (pi2)
- Web-based

CONS:

- Complexity of client/server

Raspberry Pi Kiosk

This is a pi running in kiosk mode which boots to full screen browser (using lxde autostart) and goes to web address using a web browser call kweb

Info found here

https://www.raspberrypi.org/forums/viewtopic.php?f=66&t=40860

Manuals here

http://steinerdatenbank.de/software/kweb_manual.pdf

The browser has comprehensive keyboard shortcuts, designed to control the pi entirely through a browser.

Signage Server

Many options exist for web signage, from simple stand-alone players to networked client-server models. As we want a simple proof-of-concept, I've gone with a simple web-based slide presenter called reveal.js. There is an online editor for making slides here Reveal.js can run locally, over a network on a server, or multicast over web sockets. For this prototype it run an the programming server.

Reveal.js

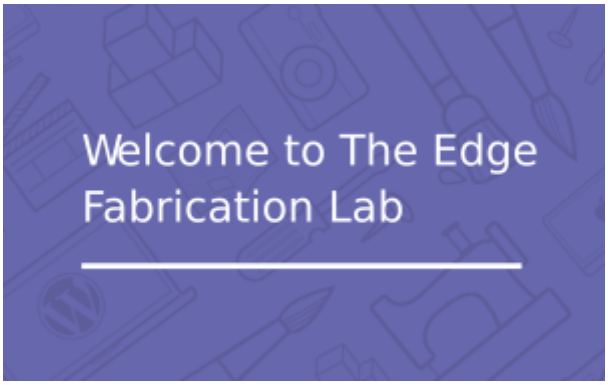
This was the first attempt. For our purposes I've hacked the index.html, modifying the css to fomat inline, a couple of pictures as background, added adding the supervisor's name (mick) and turned on autoslide in the js options.

This is the changed index.html, renamed to welcome.html

```
 <!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0, maximum-scale=1.0, user-scalable=no">

        <title>welcome to The Fab Lab</title>

        <link rel="stylesheet" href="css/reveal.css">
        <link rel="stylesheet" href="css/theme/black.css">

        <!-- Theme used for syntax highlighting of code -->
        <link rel="stylesheet" href="lib/css/zenburn.css">

        <!-- Printing and PDF exports -->
        <script>
            var link = document.createElement( 'link' );
            link.rel = 'stylesheet';
            link.type = 'text/css';
            link.href = window.location.search.match( /print-pdf/gi ) ?
```

```
'css/print/pdf.css' : 'css/print/paper.css';
            document.getElementsByTagName( 'head' )[0].appendChild( link );
        </script>
    </head>
    <body>
        <div class="reveal">
            <div class="slides">
                <section data-background-image =
"fabrication_lab_signage_v1-01.png">

        </section>
                <section data-background-image = "3pointcheck_one_page.png">
        <h1 style="color:black;align-content: center;margin-top: 580px;">
          Mick


          </h1>

        </section>
            </div>
        </div>

        <script src="lib/js/head.min.js"></script>
        <script src="js/reveal.js"></script>

        <script>
            // More info https://github.com/hakimel/reveal.js#configuration
            Reveal.initialize({
                history: true,

                // More info
https://github.com/hakimel/reveal.js#dependencies
                dependencies: [
                    { src: 'plugin/markdown/marked.js' },
                    { src: 'plugin/markdown/markdown.js' },
                    { src: 'plugin/notes/notes.js', async: true },
                    { src: 'plugin/highlight/highlight.js', async: true,
callback: function() { hljs.initHighlightingOnLoad(); } }
                ]
            });
        </script>
    </body>
</html>
```

These are the background images:



and the settings changed in reveal.js are;

```
// Loop the presentation
loop: true,

     // Number of milliseconds between automatically proceeding to the
// next slide, disabled when set to 0, this value can be overwritten
// by using a data-autoslide attribute on your slides
autoSlide: 7000,

// Stop auto-sliding after user input
autoSlideStoppable: true,
```

This is the working version revealjsfablab which can be copied to a server or opened locally.

~~You can see it running here:~~ http://192.168.36.78/reveal/welcome.html#/

Prototype Four - The Pi Kiosk + Dokuwiki Reveal.js Signage Server

implimented - but broken - power pulled and wiped SD card

Reveal.js is great - but ideally we'd like to hang it off our wiki - Once again a dokuwiki plug-in to the rescue. Therevealjs plugin for dokuwiki lets us use standard wiki pages and markup to create our

slides. Here us the welcome signage. For more examples check out the fabrication lab Fabrication Lab Signage page.

PI

Raspian Lite install built with pibakery, a gui based piSD authoring utility

Minimal customisation;

- /LXDE-pi/autostart launches start.sh
- start.sh has 180 sec delay then launches chromium, kiosk mode, pointed to signage http address
- sudo crontab -e has shutdown at 830pm

Teensy

Arduino

sketch
that reads the teensy buttons and sends out up/down/left/right/F5/esc

```
/* Buttons to USB Keyboard Example

   You must select Keyboard from the "Tools > USB Type" menu

   This example code is in the public domain.
*/

#include <Bounce.h>

// Create Bounce objects for each button.  The Bounce object
// automatically deals with contact chatter or "bounce", and
// it makes detecting changes very simple.
Bounce button0 = Bounce(0, 10);
Bounce button1 = Bounce(1, 10);  // 10 = 10 ms debounce time
Bounce button2 = Bounce(2, 10);  // which is appropriate for
Bounce button3 = Bounce(3, 10);  // most mechanical pushbuttons
Bounce button4 = Bounce(4, 10);
Bounce button5 = Bounce(5, 10);  // if a button is too "sensitive"
Bounce button6 = Bounce(6, 10);  // to rapid touch, you can
Bounce button7 = Bounce(7, 10);  // increase this time.
Bounce button8 = Bounce(8, 10);
Bounce button9 = Bounce(9, 10);

void setup() {
  // Configure the pins for input mode with pullup resistors.
  // The pushbuttons connect from each pin to ground.  When
```

```
  // the button is pressed, the pin reads LOW because the button
  // shorts it to ground.  When released, the pin reads HIGH
  // because the pullup resistor connects to +5 volts inside
  // the chip.  LOW for "on", and HIGH for "off" may seem
  // backwards, but using the on-chip pullup resistors is very
  // convenient.  The scheme is called "active low", and it's
  // very commonly used in electronics... so much that the chip
  // has built-in pullup resistors!
  pinMode(0, INPUT_PULLUP);
  pinMode(1, INPUT_PULLUP);
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(4, INPUT_PULLUP);
  pinMode(5, INPUT_PULLUP);
  pinMode(6, INPUT_PULLUP);  // Teensy++ LED, may need 1k resistor pullup
  pinMode(7, INPUT_PULLUP);
  pinMode(8, INPUT_PULLUP);
  pinMode(9, INPUT_PULLUP);
}

void loop() {
  // Update all the buttons.  There should not be any long
  // delays in loop(), so this runs repetitively at a rate
  // faster than the buttons could be pressed and released.
  button0.update();
  button1.update();
  button2.update();
  button3.update();
  button4.update();
  button5.update();
  button6.update();
  button7.update();
  button8.update();
  button9.update();

  // Check each button for "falling" edge.
  // Type a message on the Keyboard when each button presses
  // Update the Joystick buttons only upon changes.
  // falling = high (not pressed - voltage from pullup resistor)
  //           to low (pressed - button connects pin to ground)
  if (button0.fallingEdge()) {
    Keyboard.press(KEY_LEFT);
  }
  if (button1.fallingEdge()) {
    Keyboard.press(KEY_RIGHT);
  }
  if (button2.fallingEdge()) {
    Keyboard.press(KEY_UP);
```

```
}
if (button3.fallingEdge()) {
  Keyboard.press(KEY_DOWN);
}
if (button4.fallingEdge()) {
  Keyboard.press(KEY_F5);
}
if (button5.fallingEdge()) {
  Keyboard.press(KEY_ESC);
}
if (button6.fallingEdge()) {
  Keyboard.println("B6 press");
}
if (button7.fallingEdge()) {
  Keyboard.println("B7 press");
}
if (button8.fallingEdge()) {
  Keyboard.println("B8 press");
}
if (button9.fallingEdge()) {
  Keyboard.println("B9 press");
}

// Check each button for "rising" edge
// Type a message on the Keyboard when each button releases.
// For many types of projects, you only care when the button
// is pressed and the release isn't needed.
// rising = low (pressed - button connects pin to ground)
//          to high (not pressed - voltage from pullup resistor)
if (button0.risingEdge()) {
  Keyboard.release(KEY_LEFT);
}
if (button1.risingEdge()) {
  Keyboard.release(KEY_RIGHT);
}
if (button2.risingEdge()) {
  Keyboard.release(KEY_UP);
}
if (button3.risingEdge()) {
  Keyboard.release(KEY_DOWN);
}
if (button4.risingEdge()) {
  Keyboard.release(KEY_F5);
}
if (button5.risingEdge()) {
  Keyboard.println("B5 release");
}
if (button6.risingEdge()) {
```

```
    Keyboard.println("B6 release");
  }
  if (button7.risingEdge()) {
    Keyboard.println("B7 release");
  }
  if (button8.risingEdge()) {
    Keyboard.println("B8 release");
  }
  if (button9.risingEdge()) {
    Keyboard.println("B9 release");
  }
}
```

Prototype Five - The picade + Dokuwiki + Reveals

Build on the suceess of the last version - its time make it all self contained and build in the essential features.

PROS:

- Builds Existing solution
- Minimal hardware spec (pi + picade hat)
- Web-based

CONS:

- Requires custom Pi Image
- Requires Picade Hat (for shutdown switch and extra buttons)

Install Retro Pi

Use official instructions

https://github.com/RetroPie/RetroPie-Setup/wiki/First-Installation

Install Picade hat

Once again, the official installer

https://github.com/pimoroni/picade-hat

Install Extras

- xdtools to create a refresh loop (make the web browser refresh by pressing ctrl+f5 with a

script)
- unclutter to hide pointer

Configure Retro Pi

These instructions are based on this raspberry pi forum thread, summarised here.

Run Retro-pi Setup to boot to terminal - this is included in RetroPi and accessible on boot.

mod .bashrc to start start on TTY1 (this means the pi starts with the first virtual terminal - which we will then use to launch our desktop, otherwise for some reason it starts the desktop on a different terminal - meaning we can't see it...)

```
# Startx if [[ -z $DISPLAY ]] && [[ $(tty) = /dev/tty1 ]]; then exec startx;
fi
```

Edit the default user (called pi) autostart, which sets up our desktop, makes the mouse disappear after 5secs with unclutter, and calls a couple of scripts which we will write next.

`nano /home/pi/.config/lxsession/LXDE-pi/autostart`

```
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
@point-rpi
@xset s off
@xset s noblank
@xset -dpms
@unclutter -idle 5 -root
@/home/pi/start_URLrefresh.sh
@/home/pi/start_chromium.sh
```

Create Start-up Scripts

nano /home/pi/start_chromium.sh

```
# Run browser after boot to desktop
/bin/sleep 3
# run as pi user
sudo -u pi
# clean chromium on disk prefs to disable messages
sed -i 's/"exited_cleanly":false/"exited_cleanly":true/'
~/.config/chromium/'Local State'
sed -i 's/"exited_cleanly":false/"exited_cleanly":true/;
```

```
s/"exit_type":"[^"]\+"/"exit_type":"Normal"/'
~/.config/chromium/Default/Preferences
# launch  chromium
chromium-browser --kiosk --no-default-browser-check --no-first-run --
disable-infobars --disable-session-crashed-bubble
# End of script
```

nano /home/pi/start_URLrefresh.sh

```
# Start a goofy command loop to refresh the browser every 560 seconds usubg
xdt$
/bin/sleep 6
/usr/bin/lxterminal --command watch -n 560 xdotool key ctrl+F5 &
# End of goofy script
```

Configure Chromium

All we need to to is set our desired slide as the homepage, this way when Chromium launches, it goes straight to the slide we want. For example, to set the player to display the welcome signage for the Fabrication Lab - the link below will work.

http://wiki.edgeqld.org.au/doku.php?do=export_revealjs&id=facilities:fablab:signage:welcomepi-signage&theme=welcome#/

Just set the link as the homepage in chromium.

Its important to use an **external** link and not a wiki link.

Future Changes

Things that could be built into future versions.

What if Wifi goes Down?

And it always does... check out this

http://weworkweplay.com/play/rebooting-the-raspberry-pi-when-it-loses-wireless-connection-wifi/External Link

Chron Shutdown timer

Digital Signage Resources

laser_cutter_monitor_signs_preflight_v2.ai

3pointcheck.pdf

3d_printer_monitor_signs.pdf

fabrication_lab_signage_v1.ai

3pointcheck.jpg

fabrication_lab_signage_v1-01.svg

fabrication_lab_signage_v1-02.svg

fabrication_lab_signage_v1-03.svg

fabrication_lab_signage_v1.svg