

04-4 Graphical Front End via node.js

How to add a pretty face via a web browser

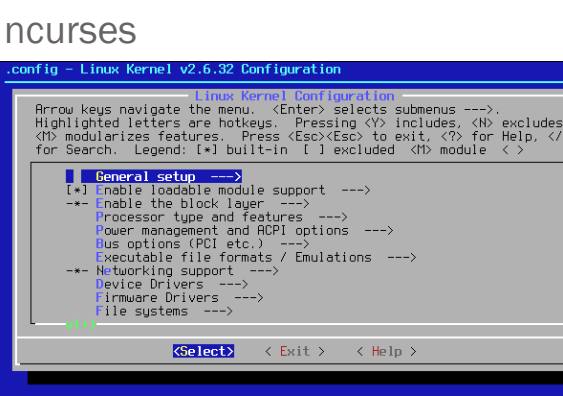
How to add a pretty face via a web browser

Ways to Add Graphics

- ncurses (<http://www.gnu.org/software/ncurses/>)
- X Window System (<http://www.x.org/wiki/>)
- Qt (<http://qt.digia.com/>) both X-based and embedded
- Web server
 - node.js (<http://nodejs.org/>)

- ncurses (<http://www.gnu.org/software/ncurses/>)
- X Window System (<http://www.x.org/wiki/>)
- Qt (<http://qt.digia.com/>) both X-based and embedded
- Web server
 - node.js (<http://nodejs.org/>)

ncurses



```
.config - Linux Kernel v2.6.32 Configuration

Linux Kernel Configuration

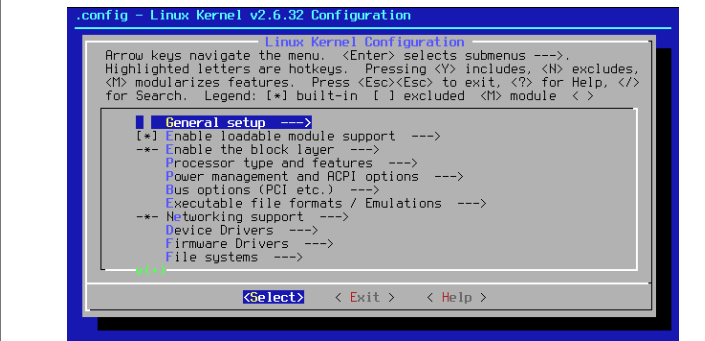
Arrow keys navigate the menu.  <Enter> selects submenus --->.
Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
<M> modularizes features.  Press <Esc><Esc> to exit, <P> for Help, </>
for Search.  Legend: [*] built-in [ ] excluded <M> module <>

[*] General setup --->
-- Enable loadable module support --->
-- Enable the block layer --->
Processor type and features --->
Power management and ACPI options --->
Bus options (PCI etc.) --->
Executable file formats / Emulations --->
-- Networking support --->
Device Drivers --->
Firmware Drivers --->
File systems --->

v(+)

<Select>  < Exit >  < Help >
```

<http://en.wikipedia.org/wiki/File:Linux-menuconfig.png>

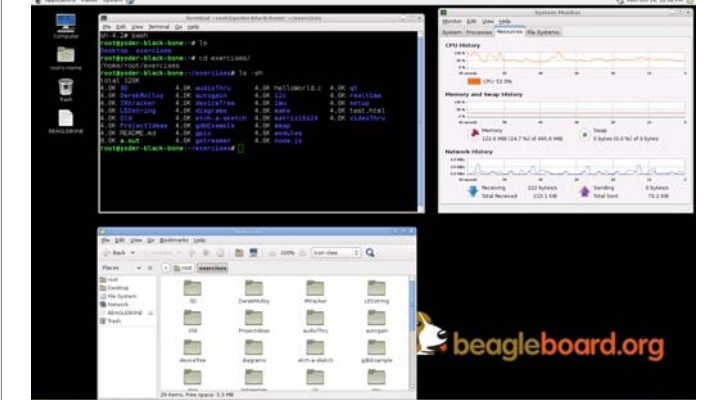


<http://en.wikipedia.org/wiki/File:Linux-menuconfig.png>

X Windows

The collage consists of four screenshots from a BeagleBoard Linux system:

- Top Left:** A terminal window showing a root user login and the execution of `ls -l`, listing files in the root directory.
- Top Right:** The 'System Monitor' application showing real-time usage graphs for CPU, Memory, and Network.
- Bottom Left:** The 'Nautilus' file manager showing the contents of the root directory, including folders like `bin`, `dev`, `etc`, and `usr`.
- Bottom Right:** The BeagleBoard.org logo.

[illegible]

```
$ opkg install qt4-demos
$ qtdemo
```

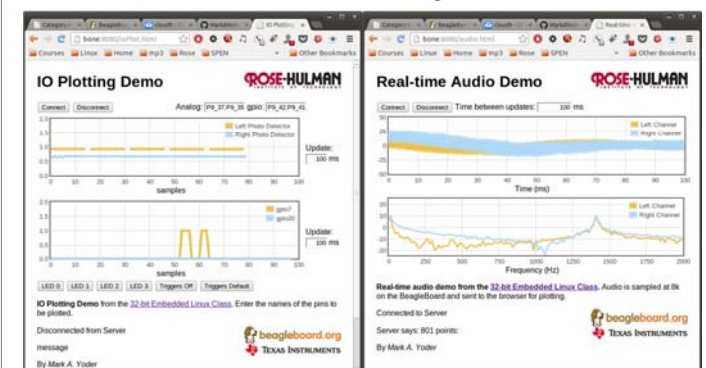
http://elinux.org/ECE497_Notes_on_Qt

```
$ opkg install qt4-demos
```

Via the Web via node.js

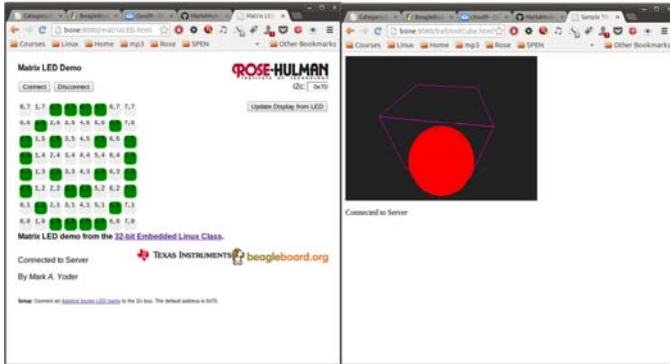
The image shows two side-by-side browser windows displaying web-based audio processing demos. Both windows have a navigation bar with links like 'Courses', 'Linux', 'Home', 'mp3', 'Power', and 'SPDN'. The left window is titled 'IO Plotting Demo' and shows a graph of 'Left Photo Detector' and 'Right Photo Detector' over 100 samples. The right window is titled 'Real-time Audio Demo' and shows two graphs: 'Time between updates' and 'Frequency (Hz)', both showing 'Left Channel' and 'Right Channel' data. Both demos are powered by 'ROSE-HULMAN INSTITUTE OF TECHNOLOGY' and 'beagleboard.org'.

beagle\$ cd ~/exercises/realtime
beagle\$ node boneServer.js



```
beagle$ cd ~/exercises/realtime
beagle$ node boneServer.js
```

Via the Web via node.js



```
beagle$ cd ~/exercises/realtime
beagle$ node boneServer.js
```

node.js

- Platform built on [Chrome's JavaScript runtime](https://v8.dev/docs) for easily building fast, scalable network applications.
- Uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.
- Programmed in JavaScript on both server and client.

<http://nodejs.org/>

node.js example: Webserver

- This simple web server written in Node responds with "Hello World" for every request.
- ```
var http = require('http');
http.createServer(function (req, res) {
 res.writeHead(200, {'Content-Type': 'text/plain'});
 res.end('Hello World\n');
}).listen(1337);
console.log('Server running on port 1337');
```
- To run the server, put the code into a file example.js and execute it with the node program:  
**\$ node example.js**  
Server running on port 1337

## Things to know

- JavaScript
  - socket.io
  - jQuery
  - DOM
- html
- CSS
- Where to you start?

## Javascript – C-like

```
#include <stdio.h>
main() {
 int i;
 for(i=0; i<5; i++) {
 printf("i=%d\n", i);
 }
}

var i;
for(var i=0; i<5; i++) {
 console.log("i=%d", i);
}
```

## JavaScript in 10 minutes

- By Spencer Tipping
- <https://github.com/spencertipping/js-in-ten-minutes>
- 27 pages
- OR
- <https://sites.google.com/site/solopuotutoriales/javascript-in-ten-minutes>
- 9 pages
- Here are the highlights...

## JS - Types

- **Strings** – e.g. 'foo', "foo" (single vs. double quotation – no difference)
- **Numbers** – e.g. 5, 3e+10 (all numbers behave as floats)
- **Booleans** – true and false
- **Arrays** – e.g. [1, 2, "foo", [3, 4]]
- **Objects** – e.g. {foo: 'bar', bif: [1, 2]}, which are really just hashtables
- **Functions** – e.g. var example=function(x) {return x + 1}

## JS - Functions

- Functions are first-class lexical closures
- ```
var f = function () { // f is toplevel, so global
    var x = 5;        // x is local to f
    y = 6;            // y is global
};
```
- Watch out
- ```
var f = function () { // f is toplevel, so global
 y = 6; // y is global
 x = 42;
 Do stuff...
 var x = 5; // x is local to f
};
```

## JS - Semicolon

- Javascript doesn't require a semicolon at the end of each line, but you should anyway.

```
var x = f
(y = x) (5)
```

- Is treated as:

```
var x = f(y = x) (5)
```

- You probably meant

```
var x = f;
(y = x) (5);
```

## JS - Equality

- Never use == or !=
  - Always use === or !==
  - All these are **true**:
- ```
null == undefined
null == 0
false == ''
'' == 0
true == 1
true == '1'
```

JavaScript: The Good Parts



- Intended for programmers who, by happenstance or curiosity, are venturing into JavaScript.
- Also intended for programmers who have been working with JavaScript at a novice level and are now ready for a more sophisticated relationship with the language.
- Most programming languages contain **good parts and bad parts**. I discovered that I could be a better programmer by using only the good parts and avoiding the bad parts.
- JavaScript is a language with more than its share of bad parts.
- 172 pages

Things to know

- JavaScript
 - socket.io
 - jQuery
 - DOM
- html
- CSS

socket.io

- <http://socket.io/>
- **Socket.IO** aims to make realtime apps possible in every browser and mobile device, blurring the differences between the different transport mechanisms.
- It's care-free realtime 100% in JavaScript.

socket.io

```
• Server
var io = require('socket.io').listen(80);
io.sockets.on('connection', function (socket) {
  socket.emit('news', { hello: 'world' });
  socket.on('my other event', function (data) {
    console.log(data);
  });
});
• Client - Browser
<script>
var socket = io.connect('http://localhost');
socket.on('news', function (data) {
  console.log(data);
  socket.emit('my other event', { my: 'data' });
});
</script>
```

socket.io

- See: **Getting Started With node.js and socket.io (v0.7+) – Part 2**
- <http://codehenge.net/blog/2011/12/getting-started-with-node-js-and-socket-io-v0-7-part-2/>
- My code is based on this

Things to know

- JavaScript
 - socket.io
 - DOM
 - jQuery
- html
- CSS

DOM

- **Essentials of the DOM and JavaScript in 10 Minutes**
- <http://www.youtube.com/watch?v=URF2sVQWuxU>
- **10 minute YouTube video**
- However we'll use jQuery, it's much more compact



Things to know

- JavaScript
 - socket.io
 - DOM
 - jQuery
- html
- CSS

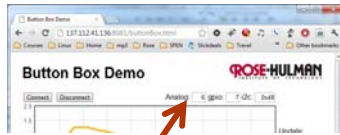
jQuery

- <http://jquery.org/>
- jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.
- jQuery is designed to change the way that you write JavaScript
- Looks like `$()` in html

Analog: `<input id="ainNum" type="text" value="" style="text-align: right; width:2em">`

- In JavaScript

```
$("#ainNum").val(ainNum).change(function () {  
    ainNum = $(this).val();  
});
```



Things to know

- JavaScript
 - socket.io
 - DOM
 - jQuery
- html
- CSS
- FLOT
- <http://www.flotcharts.org/>
- Flot is a pure JavaScript plotting library for jQuery, with a focus on simple usage, attractive looks and interactive features.

To Do

- Look at `~/exercises/realtime` and see what you can figure out.

