

---

# Running PHP on Embedded Devices

PHP Dutch Conference 2009  
12.06.2009

Michael Wittke

Leibniz University of Hannover  
Institute of Systems Engineering  
Department of Computer Science



## Michael Wittke:

### ☐ PhD student from Hannover:

- Research focuses on [self-configuration](#) for Mobile Vision Networks:
- Networks with high mobility
- Needs for a steady re-calibration of the network

### ☐ I am addicted to PHP:

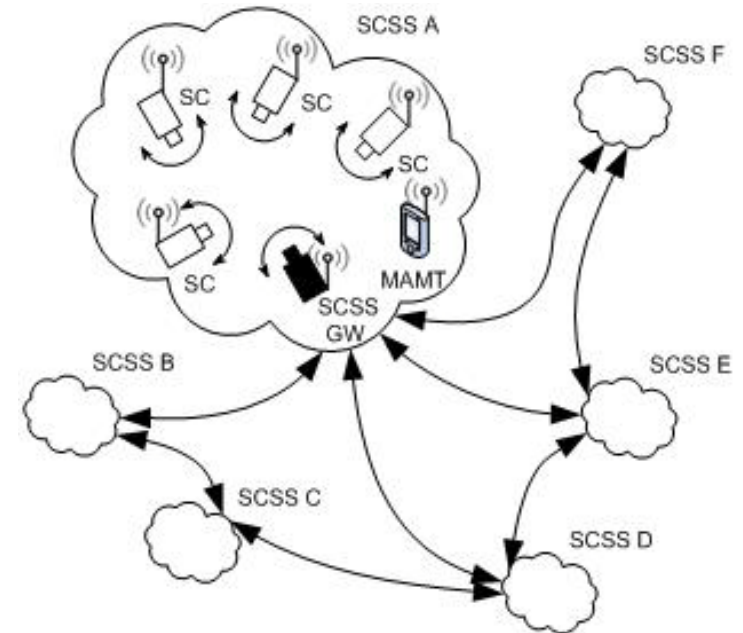
- Programming websites in PHP since 1998
- Programmed my website ([foodplaner.de](#)) in PHP with more than 2,000 LOCs
- [foodplaner.de](#) is a platform for planning the nutrition (e.g. food diary, calorie table with more than 30,000 foods etc.)
- ...and I use PHP at University for programming embedded devices, since life is too short for C!

# Distributed Smart Cameras at a glance

## What is it?

### □ Each Smart Camera Node:

- Includes a **PTZ camera or webcam**
- **Local processing resources** (CPU, memory, etc.)
- **Communication interface** to exchange information with other Smart Camera Nodes (e.g. wireless ad-hoc or wired network)



### □ Networked Smart Camera Nodes

- Cameras can **cooperatively solve surveillance tasks**
- Achieve goals that cannot be achieved with a single camera, e.g. **wide-area object tracking, multi-view observation**

# System Architecture for Distributed Smart Cameras

❑ Advances in computer vision make way for intelligent cameras:

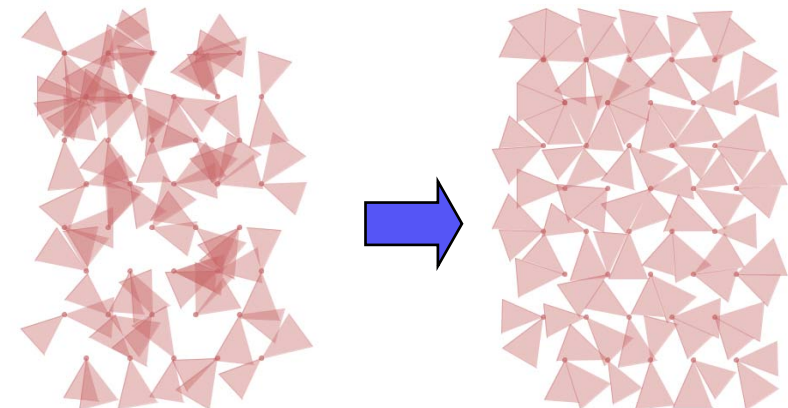
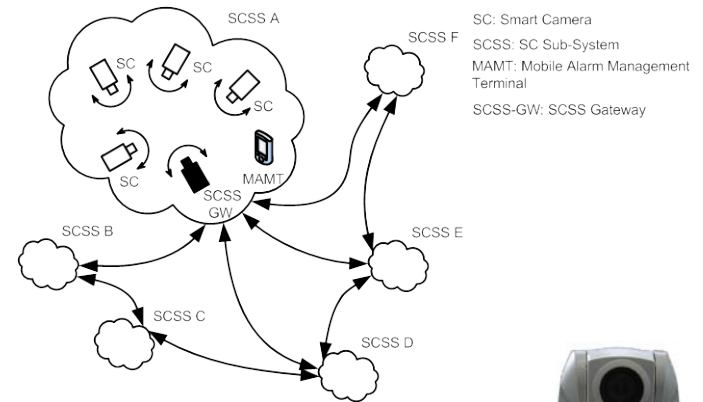
- Smart Cameras cooperate and detect events of interest autonomously
- Alarm Management: Smart Cameras communicate with mobile devices

❑ Research focuses on e.g. [system architecture](#):

- Distributed algorithms, protocols

One example:

- Robust Online Camera Alignment System (ROCAS)



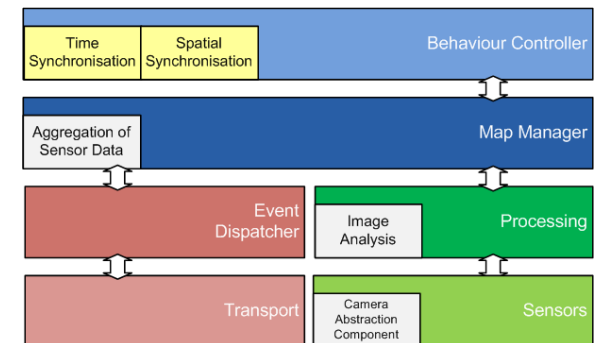
❑ Each mobile phone with built-in CMOS sensor can be seen as Smart Camera itself:

- Due to mass production their price decreases steadily
- But: Processing capacity, networking abilities and connectivity increases
- Makes way for:  
Mobile Surveillance Applications



❑ Research focuses on **self-configuration** for Mobile Vision Networks:

- Dynamic configuration space of mobile nodes
- Time synchronization (*when?* and *how long?*)
- Spatial calibration (*where?*)
- Network Synchronization



## **Platform-independent Filing System**

*How to set up PHP on embedded devices*

NSLU2 (Network Attached Storage)

Setting up Debian ARM

Nokia N810

Compiling natively

Virtualization / Static Cross-Compiling

Scratchbox / Dynamic Cross-Compiling

## **The SmartCam Lab**

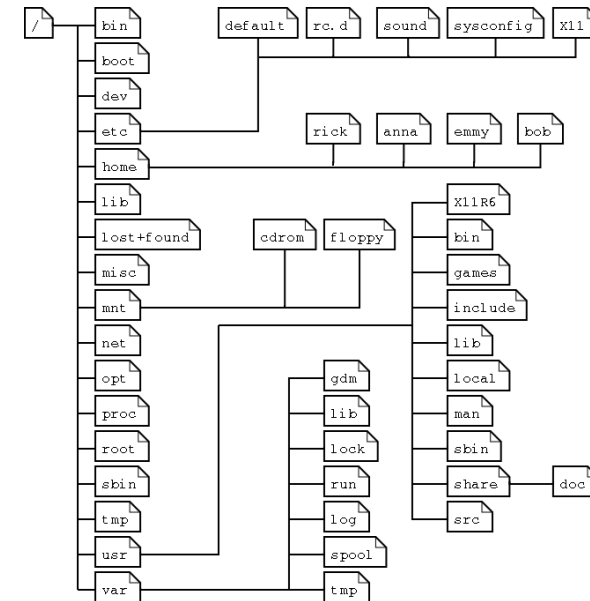
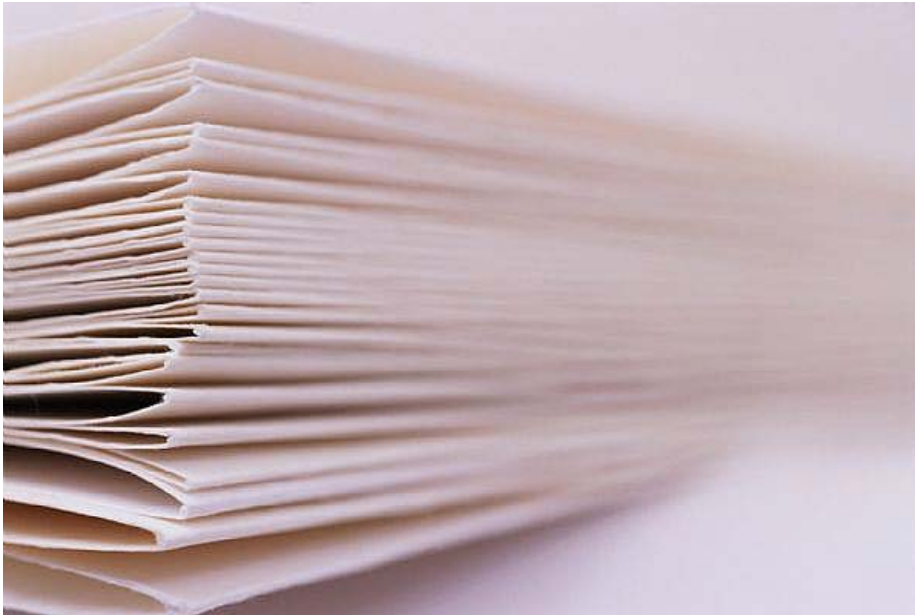
*The Power of PHP*

*e.g.* Live Video Streaming (10 LOCs)

---

# Platform-independent Filing System

# Filing System vs. File System



## (Digital) Filing System

- Temporary and permanent storage (e.g. of references, commands, configurations, settings etc.)

## File System

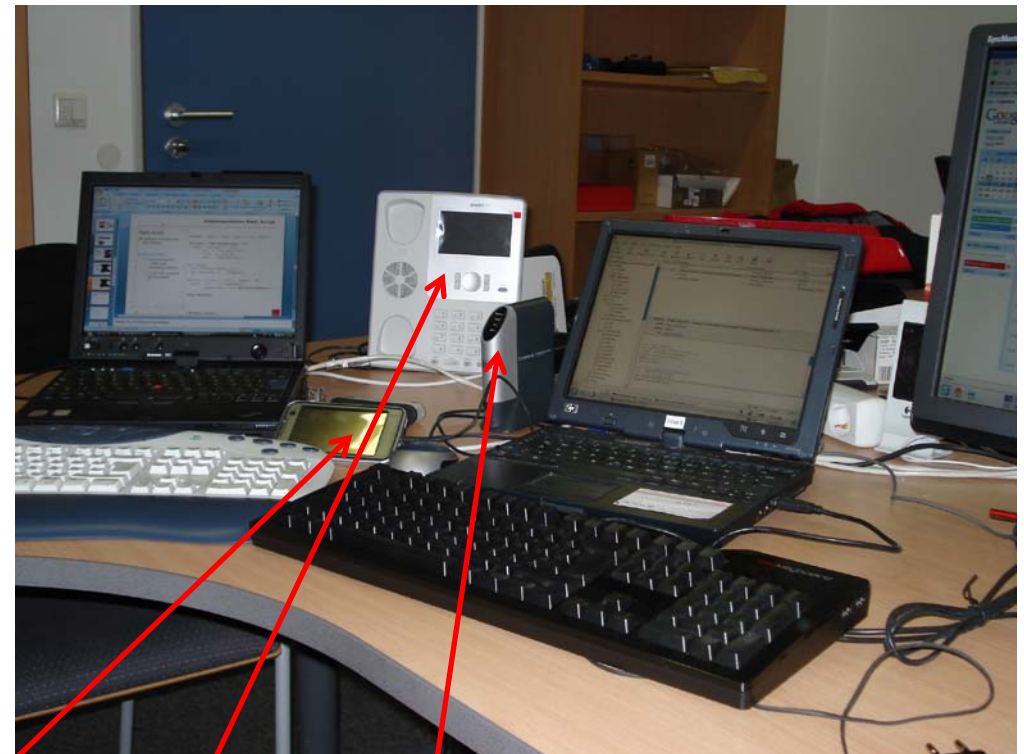
- Method for storing and organizing computer files and the data they contain

# My Working Place

A wild mixture of (embedded) devices:

❑ Five devices dominate my everyday working life for building embedded smart cameras:

- ThinkPad [laptop](#) (running **Ubuntu 7.10**)
- HP [laptop](#) (running **Ubuntu 9.04**)
- Nokia N 810 [wireless Internet tablet](#) (running **MAEMO**)
- NSLU2 [network attached storage](#) (running **Debian ARM**)
- SNOM [IP telephone](#) (running **Debian ARM**)



Wireless Internet Tablet  
Nokia N810

IP telephone

NSLU2

Problem:

❑ **Linux shell is a **command line** tool**

- Approx. 30 commands are used each day
- Not all commands are used frequently
- Commands are used throughout all devices
- Not all devices allow for copying instructions (e.g. in case of web access from any browser window)

❑ **...but I suffer from a **short term memory** ;)**

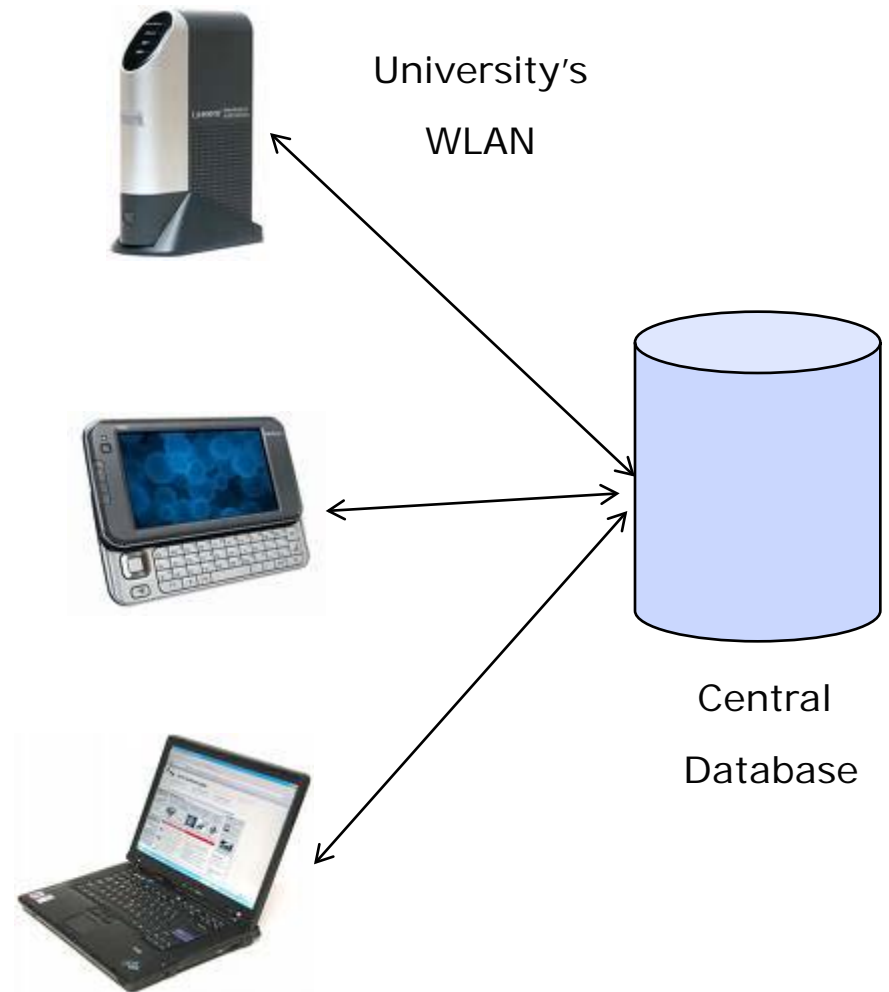
- Bash history is hard to synchronize via multiple devices
- List lengths may vary on distinct devices
- History is cycle buffer

## Idea: Cross-device filing system

### □ Storing data such as

- Syntax of Linux commands (e.g., `find <path> -i name <keyword>, ...`)
- References to websites (e.g. "how to set up Debian ARM?" etc.)
- Configurations (e.g. setting in `/etc/network/interfaces,` `/etc/wpa_supplicant.conf` etc.)

### □ Reading stored by each device



## □ Storing interface

- Syntax: **h <command> <description>**  
e.g. h 'echo hello' 'print messages on command line'
- Running on each device
- Easy access via command line (shell script)

## □ Reading interface

- Syntax: h -<option> <keyword>  
**h -d <keyword>**  
e.g. h -d 'print messages'  
Output: echo hello

**h -r <keyword>**  
e.g. h -r 'echo'  
Output: echo hello

# Implementing a SOAP-Server in PHP

---

## Communication using SOAP\*

- ❑ Simple Object Access Protocol
- ❑ Lightweight XML-based protocol for exchanging information between distributed applications
- ❑ Uniform interface for the clients
  
- ❑ Setup in PHP:
  - Easy to install (i.e. SOAP extension)
  - 4 LOCs for setting up the server

```
function add($command, $desc, $os)
{
    //business logic for adding
}

function search($s)
{
    //business logic for searching
}

$server = new SoapServer(NULL,
    array('uri' =>
        "http://localhost/"));
$server->addFunction('add');
$server->addFunction('search');
$server->handle();
```

\*<http://de.wikipedia.org/wiki/SOAP>

# Implementing a SOAP-Client in PHP

## Client-side Communication

□ Uniform interface for clients

□ Setup in PHP:

- Easy to install  
(--with-xml  
compiling option)
- 3 LOCs for setting  
up the client

```
$command = $argv[1];$desc = $argv[2]; $os = $argv[3];
```

```
$client = new SoapClient( NULL,  
    array( "location" => $url,  
          "uri" => "urn:xmethodsTS",  
          "style" => SOAP_RPC,  
          "use" => SOAP_ENCODED) );
```

```
$p = array(  
    new SoapParam($desc, 'str'),  
    new SoapParam('c', 'flag'));
```

```
$result = $client->__call( "search", $p,  
    array( "uri"=>"urn:xmethodsTS",  
          "soapaction"=>"urn:xmethodsTS#search" ) );
```

```
echo $result;
```

# Wrapping PHP into a Shell Script

## Bash script

- ❑ Bash wrapper script for calling PHP
- ❑ Bash script has to be adapted to the operating system (e.g. Nokia N810)
- ❑ Added to /usr/bin for easy access



```
#!/bin/sh // for Nokia N810
#!/bin/bash // for NSLU2, Ubuntu

OS=`uname -a`
echo "OS: $OS"
echo "command: $1"
echo "description: $2"

php5 /usr/bin/soapclient.php
"$1" "$2" "$OS"
```

# Example Embedded Devices and Setting up PHP

---



# Hardware platform #1 for running PHP

---

## NSLU2 (Network Attached Storage)

- ARM processor, 250Mhz
- DDR RAM 64MB
- External hard disk
- Network connection
- USB** high speed for PC connectivity



## Installation of Debian ARM

- Change IP settings  
(e.g. DHCP server)
- Download [SlugOS firmware](#)
- Flash device  
`sudo upslug2 -i di-slu2.bin`
- Grab Debian GNU/Linux lenny (5.0)  
from [slug-firmware.net](#) and follow  
this installation guide:  
[http://www.cyrius.com/debian/  
nslu2/install.html](http://www.cyrius.com/debian/nslu2/install.html)
- Installation will take [roughly](#)  
[4 hours](#)
- `apt-get php5-cli` 😊  
`apt-get php5-xml` 😊 😊

# Hardware platform #2 for running PHP

---

## Nokia N 810

- ❑ High-resolution display (800 x 480 pixels) with up to 65,000 colors
- ❑ TI OMAP 2420 ARM processor, 400Mhz
- ❑ DDR RAM 128MB, Flash 256MB
- ❑ WLAN standard: IEEE 802.11b/g
- ❑ USB high speed for PC connectivity
- ❑ Internet Tablet OS:  
maemo Linux based OS2008



## Installing PHP

❑ apt-get install php5-cli

→ 😊

❑ apt-cache search xml | grep php

→ ☹

Welcome to the world of  
cross-compiling!

## What is cross-compiling?

- ❑ Compiling **natively** is too slow
- ❑ Use some **fast** processor (**HOST**) to compile software for some slow processor (**TARGET**) that uses a different architecture:
  - HOST cannot run the software natively which it compiles
  - software is compiled for another processor
- ❑ **But:** Build environments need to run programs during the compiling process (**CRASH!**)

## Native Compiling on ARM hardware

### ❑ Dynamic (file size: 7 KByte)

- `gcc ./hello.c -o hello`
- `file ./hello`

Output:

```
ELF 32-bit.. dynamically  
linked (uses shared libs)
```

- `ldd ./hello`

Output:

```
libc.so.6 => /lib/libc.so.6 ..
```

### ❑ Executing `hello` on Nokia N810

- `-sh: ./hello: not found`

```
#include <stdio.h>  
  
main()  
{  
    printf("Hello world!");  
}
```

## Native Compiling on ARM hardware

### ❑ Static (file size: 500 KByte)

- `gcc --static ./hello.c -o hello`
- `file ./hello`

Output:

ELF 32-bit.. **statically** linked

- `ldd ./hello`

Output:

not a dynamic executable

### ❑ Executing `hello` on Nokia N810

- Hello world 😊

```
#include <stdio.h>

main()
{
    printf("Hello world!");
}
```

## Static Compiling of PHP on NSLU2

### □ Static (`--static`)

- Compilation time on NSLU2
  - configure: 5 minutes
  - make: roughly 3 hours
- File size: 15 MByte  
(bad for memory performance on embedded devices)

## What is Virtualization?

### □ Cross-platform virtualization

- allows software compiled for a specific CPU and operating system to run unmodified on computers with different CPUs and/or operating systems

### □ Platform virtual machines

[http://en.wikipedia.org/wiki/Comparison\\_of\\_platform\\_virtual\\_machines](http://en.wikipedia.org/wiki/Comparison_of_platform_virtual_machines)

- X86: Bochs, VMware
- MIPS: GXemul, Imperas, OKL4, OVPsim
- ARM: GXemul, Imperas, QEMU

## Virtualizer QEMU

<http://www.nongnu.org/qemu>

- ❑ QEMU is a generic and open source machine emulator and virtualizer
- ❑ Shows a good performance by using dynamic translation
- ❑ Driver called KQEMU (QEMU accelerator) achieves great speed up in case of x86 architectures

## Virtualized Debian-ARM in QEMU

[http://www.aurel32.net/info/debian\\_arm\\_qemu.php](http://www.aurel32.net/info/debian_arm_qemu.php)

### Install QEMU

```
sudo apt-get install qemu
```

### Create image

```
qemu-img create hd0.img 5G
```

### Download vmlinuz and initrd from debian.org

### Start QEMU

```
qemu-system-arm  
-M versatilepb  
-kernel vmlinuz-2.6.26-2-versatile  
-initrd initrd.gz  
-hda hd0.img  
-append "root=/dev/ram"
```

## Static Compiling PHP in QEMU

- ❑ Static (`--static`)
  - Compilation time in QEMU
    - configure: 5 minutes
    - make: roughly 1.5 hours
  - File size: 15 MByte as well
- ❑ Dynamic compiling is not possible due to distinct libc-versions

## What is the Scratchbox?

- ❑ Comes from "Linux from scratch" + "chroot jail" (sandbox).
- ❑ Virtualization is limited to a minimum
  - Great speedup
  - E.g. virtualizing build-environment
  - E.g. /proc is not emulated
- ❑ Cross-compilation toolkit designed to make embedded Linux application development easier
- ❑ Supports ARM and x86 targets (PowerPC, MIPS and CRIS targets are experimental)
- ❑ Provides glibc and uClibc as C-library choices

## Scratchbox for Maemo

[www.maemo.org](http://www.maemo.org)

- maemo = Nokia N810 operating system (Debian-based)
- Scratchbox for maemo = software package to implement development sandboxes (for isolation)
- Contains easy-to-use tools to assist cross-compilation
- Supports multiple developers using the same development system
- Supports multiple configurations for each developer
- Supports running non-native binaries on the host system via instruction set

## Setting up Maemo Scratchbox

[http://maemo.org/maemo\\_release\\_documentation/maemo4.1.x/node4.html](http://maemo.org/maemo_release_documentation/maemo4.1.x/node4.html)

### Installing Scratchbox

Download installer script from  
<http://repository.maemo.org/>

### Installing Maemo SDK

Download installer script from  
<http://repository.maemo.org/>

### Give executable permissions to both scripts

### Start sbox and login:

```
sudo /scratchbox/sbin/sbox_ctl start  
/opt/scratchbox/login
```

## Dynamic compiling of PHP

- ❑ Dynamic compiling is possible due to the sane environment
- ❑ Dynamic
  - Compilation time in QEMU
    - configure: 2 minutes
    - make: 5 minutes 😊
  - File size: 9 MByte 😊

---

# The SmartCam Lab

## The SmartCam Lab

- ❑ Lab for students to improve their abilities in programming embedded devices
- ❑ Time: 3 days à 4 hours
- ❑ Goal:
  - Home protection by [video surveillance](#)
  - Use [off-the-shelf](#) and [low cost](#) hardware
  - Low power consumption
  - Access to the camera’s [real-time video stream](#) via VLC client
  - [Face detection](#) and store recognized faces in a database for 30 days

## Idea HW/SW

- ❑ Use NSLU2 with Debian-ARM as implementation platform
  - [NSLU2](#) is off-the-shelf:  
low cost (< 60 €) & low power (ARM chip)
  - USB-Webcam: [Quickcam](#) Pro 9000
- ❑ Use PHP in education:
  - Scripting language ([easy to learn](#))
  - Professional software tools are available ([e.g. symfony with MVC, ORM etc.](#))
  - Available on Debian-ARM as package
- ❑ Image processing library with PHP interface:
  - Intel's [OpenCV](#) (image processing library)

## Using PHP to solve the following tasks

- ❑ Task 1: Last activities
  - Website showing the last recognized faces
- ❑ Task 2: Detailed view of activity
  - Captured picture with data
- ❑ Task 3: Webcam posts recognized face
  - Webcam captures a picture and start a face detection algorithm. If a face is recognized, this is stored in the database.
- ❑ Task 4: Video Live Stream Server
  - Live access to the webcam via VLC

## PHP Task 1 – Setting up PHP

- ❑ Starting up the project
  - Installing PHP, MySQL and Apache on the NSLU2
  - Installing [symfony](#) with doctrine as ORM
  - Setting up the data model and fixture data i.e. activity and category (e.g. category could be the living room or the office hallway, activity describes the detected faces)

## PHP and Image Processing

- ❑ OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real time computer vision.
- ❑ Example applications of the OpenCV library:
  - Human-Computer Interaction (HCI)
  - Object Identification
  - Segmentation and Recognition
  - [Face Recognition](#)
  - Gesture Recognition
  - Motion Tracking
  - Motion Understanding
  - Mobile Robotics
  - Stereo and Multi-Camera Calibration

## PHP and Face Detection #1

### □ Face detection in C

- Cross-compiling OpenCV libs
  - `configure --without-x --without-python`
  - Roughly 8 hours on NSLU2
- Use `facetect.c` from the `opencv/samples/c/` directory
- Send events via `cURL` to the webserver
  - **cURL** is a command line tool for transferring files with URL syntax
  - Just 20 LOCs

## PHP and external C scripts

### □ Face detection in C/PHP wrapper script

- Use facedetect.c and add

```
fprintf(stdout, "%d\n", num_faces);  
return 0;
```

- Execution of an external C program:

```
$num_faces =  
exec("facedetect ...", $dummy, $rcode);
```

- Does not work in 'safe mode'.
- Length of command line give critical restrictions

Windows cmd.exe	→ 8k
Windows cygwin bash	→ 32k
Linux bash	→ 128k (may vary)

## PHP Facedetect

### □ Face detection in PHP

[www.xarg.org/project/php-facedetect/](http://www.xarg.org/project/php-facedetect/)

- Install OpenCV
- Install PHP Facedetect  
[www.xarg.org/download/facedetect-1.0.0.tar.gz](http://www.xarg.org/download/facedetect-1.0.0.tar.gz)
- Recognize faces in any PHP script by the following command:

```
$num_faces =  
face_detect($image,$path_to_haarcascade)
```

## PHP and Video Streaming

- ❑ VideoLAN is a complete software solution for video streaming and playback (GPL)
- ❑ Originally designed to stream MPEG videos on high bandwidth networks
- ❑ Full-featured, cross-platform media player
- ❑ VLC works on many platforms: Linux, Windows, Mac OS X, ...
- ❑ VLC can also be used as a [streaming server](#).

## Video Streaming in PHP

- ❑ Create a socket for communication with VLC client (port 1234)
  - AF\_INET family
  - SOCK\_STREAM for TCP connection
- ❑ Pictures are loaded from a local directory (e.g. captured via gstreamer)
- ❑ Loaded pictures are written to the socket
- ❑ 10 LOCs for implementing a VLC server in PHP 😊

```
#!/usr/bin/php -q //quiet mode

$ip = "..."; $port = "1234";

$sock = socket_create(AF_INET,
    SOCK_STREAM, SOL_TCP);

socket_bind($sock, $ip, $port);
socket_listen($mysock, 5);
$c = socket_accept($sock);

socket_write($c, "HTTP/1.1 200 OK\r\n");
socket_write($c,
    "Content-Type: image/jpeg\r\n");
socket_write($c, "\r\n");

while (true) {
    //capture/load pic & write to socket
}
socket_close($mysock);
```

## ❑ Cross-Compiling of PHP

1. Statically on HW (NSLU2)
2. Statically in Virtualizer  
(Debian ARM in QEMU)
3. Dynamically in Scratchbox  
(for Nokia N810)

## ❑ Using PHP for

1. SOAP clients/server  
~3 - 4 LOC
2. Wrapper for external C programs  
~1 LOC
3. Implementing face detection  
(OpenCV)  
~1 LOC
4. Implementing a VLC server  
~ 10 LOCs

- ❑ Execution of external programs has to be improved  
e.g. tricky workflow ;)

Starting webserver script and facedetection by one PHP wrapper script => I failed

- ❑ CPU and memory utilization of PHP should be reduced
- ❑ Standard libraries for PHP development have to be ported to embedded architectures (e.g. ARM, MIPS etc.)

...



This is not a Smart  
Camera!

Thank you for your attention!