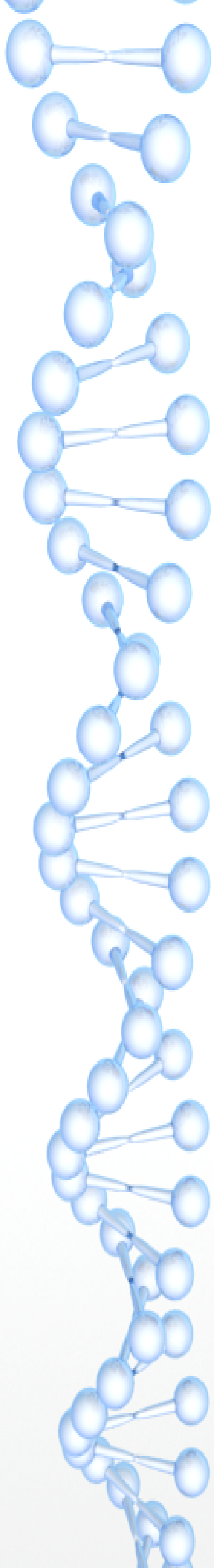
Using Raspberry Pi as Data Server for Embedded Systems

Course Schedule

- Session 1: Introduction
  - Raspberry Pi Development Board
  - Data Server Systems Overview
- Session 2: Programming Basics
  - Database access and SQL
  - File access

Course Schedule (cont.)

- Session 3: Server-side Programming
  - C-based (mongoose/my1goose)
  - PHP-based (my1apisrv)
- Session 4: Client-side Programming
  - HTML/CSS/Javascript
  - Graphs with D3js

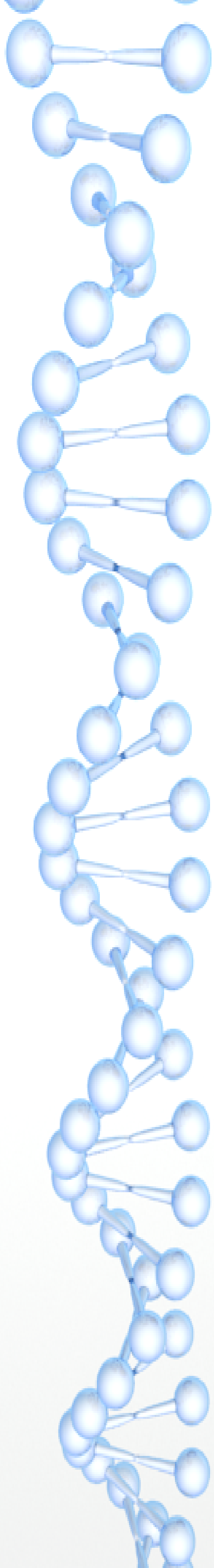


Reminder

- Compressed contents
  - Wide coverage with limited days
- Selected topic/coverage
  - Focusing on what is required
- Feel free to ask for more information!

Session 1: Introduction

Raspberry Pi Development Board



Session Overview

- Raspberry Pi Development
  - Platform Specifications
  - Bare-metal / OS Selection
  - Board Preparations
  - Simple GPIO access
- Data Server Systems
  - Embedded Data Server
  - Pure Data Server

Raspberry Pi

- SBC to promote computer science in schools
  - Raspberry Pi Foundation (http://raspberrypi.org)
  - “… computer to inspire children ...”
- Low cost, yet huge features
  - $25 - $35 SBC board with graphics engine
- Commercial Facts:
  - First batch (10,000 units) were sold before made!
  - (October 2014) 3.8 million boards sold

Raspberry Pi (cont.)

- Main models (as of 20160902)
  - Pi 1 (Model A,A+,B,B+), Pi 2, Pi 3
  - Pi Compute Module, Pi Zero!
- Pi 1 Model B+ with Broadcom’s BCM2835
  - SoC package with 512MB SDRAM
  - CPU: 700 MHz ARM1176JZF-S (ARM11 core, ARMv6 ISA)
  - GPU: 250 MHz Broadcom VideoCore IV (with OpenGL engine and MPEG codecs

Endless Possibilities?

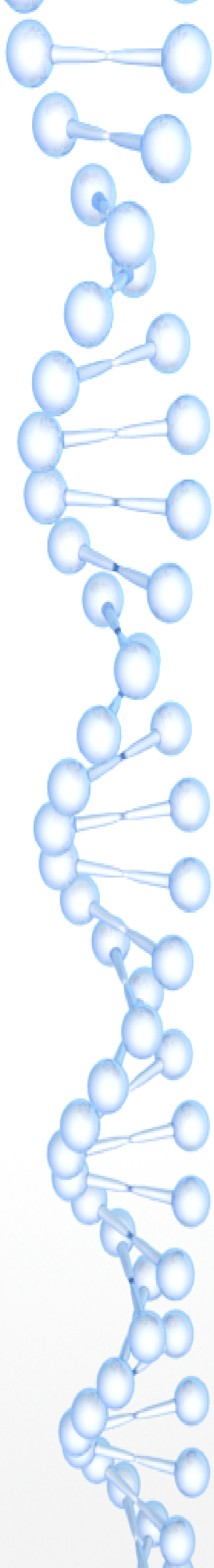
- What we have
  - SBC capable of running Linux
  - USB Hubs, Network capable, Graphics Engine
  - Hardware Floating-point Unit
  - Very portable/mobile (small sized)
- Supported Extensions
  - Camera – supported and configurable
  - Gertboard – educational I/O interface board
  - HAT (Hardware Attached on Top)

Bare-metal Codes

- First code to run on hardware
  - Low-level hardware access
  - No OS (bare-metal codes becomes the OS)
- Simple/common application
  - Single task, single threaded
  - Using (100%) assembly is still possible
  - Using C is sometimes an overkill (still.. it works)
- May implement multi-tasking
  - Static scheduling

Pi Operating System

- Targets school children
  - Naturally, bare-metal is not an option
  - Must be familiar, easy-to-use and feature-rich
  - Pre-installed memory card available
- Download options
  - NOOBS – New-Out-Of-Box-Software
  - Raspbian – Official Linux Distribution for Pi
  - Other 3rd-party OS (including Win10 IoT!)

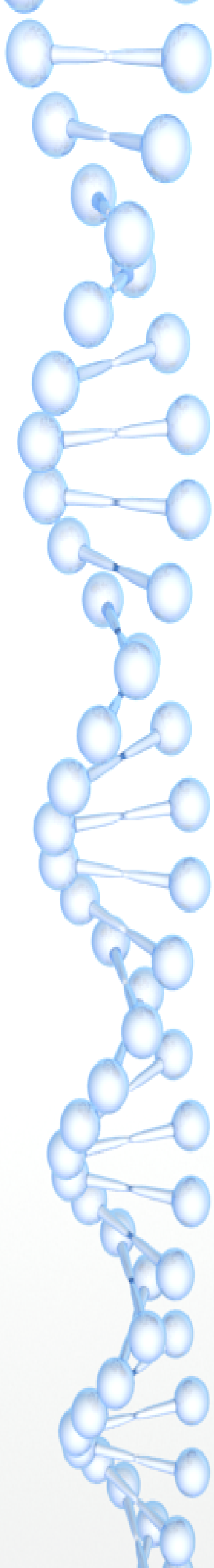


Raspbian OS

- Raspbian – based on Debian
  - ‘Official’ Distribution
- Normal user on login with sudo access
  - Commands with sudo has root privileges
- Same package management system
  - To install package: *sudo apt install <pkg>*
  - To update package list: *sudo apt update*
  - To upgrade software: *sudo apt upgrade*

OS Installation

- Downloads are disk images
  - Write (raw) to SD card (use dd on Linux)
- Can be done ‘manually’
  - Boot partition: FAT32 (FAT16 OK?)
  - Root partition: Linux EXT filesystem (Any?)



Pi Development

- Bare-metal Codes
  - Cross-compiler is a must
  - Development on host PC
- Running (Linux) OS
  - Native compiler available
  - Can be slow for many (better on Pi 3?)
  - Cross-compilers can be used

Playtime 1

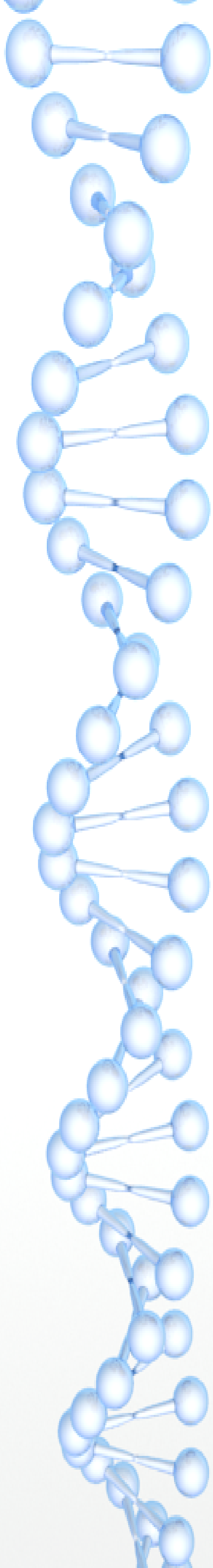
- What we need
  - Raspberry Pi board, HDMI monitor and cable
  - USB keyboard & mouse
  - MicroSD card with Raspbian pre-installed
- What we do
  - Switch on Raspberry Pi and boot Raspbian
  - Explore Raspbian Desktop on Pi
  - Change hostname to something unique

Playtime 2

- What we need
  - Stuffs from Playtime 1
  - Breadboard, LED, 1K resistor
  - Connection wires (2xM-F, 2xM-M)
- What we do
  - Access GPIO from shell (using sys fs)

Playtime 3 (optional)

- What we need
  - Stuffs from Playtime 2
  - Linux kernel module programming knowledge
- What we do
  - Compile/Load/Remove kernel module
  - Module to blink LED

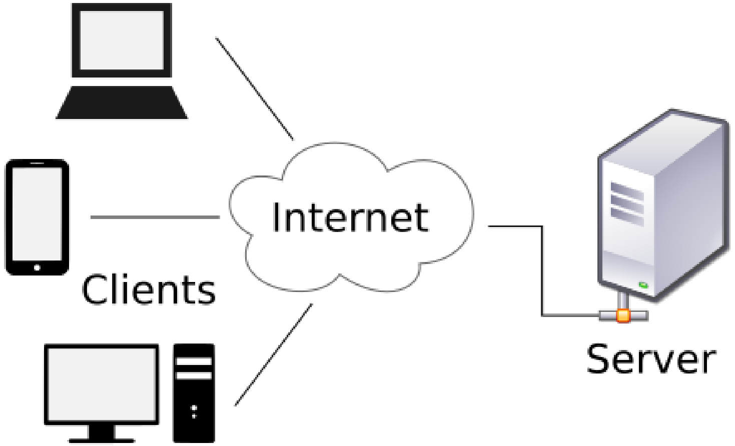


Session 1: Introduction

Data Server Systems Overview

Client-Server Model

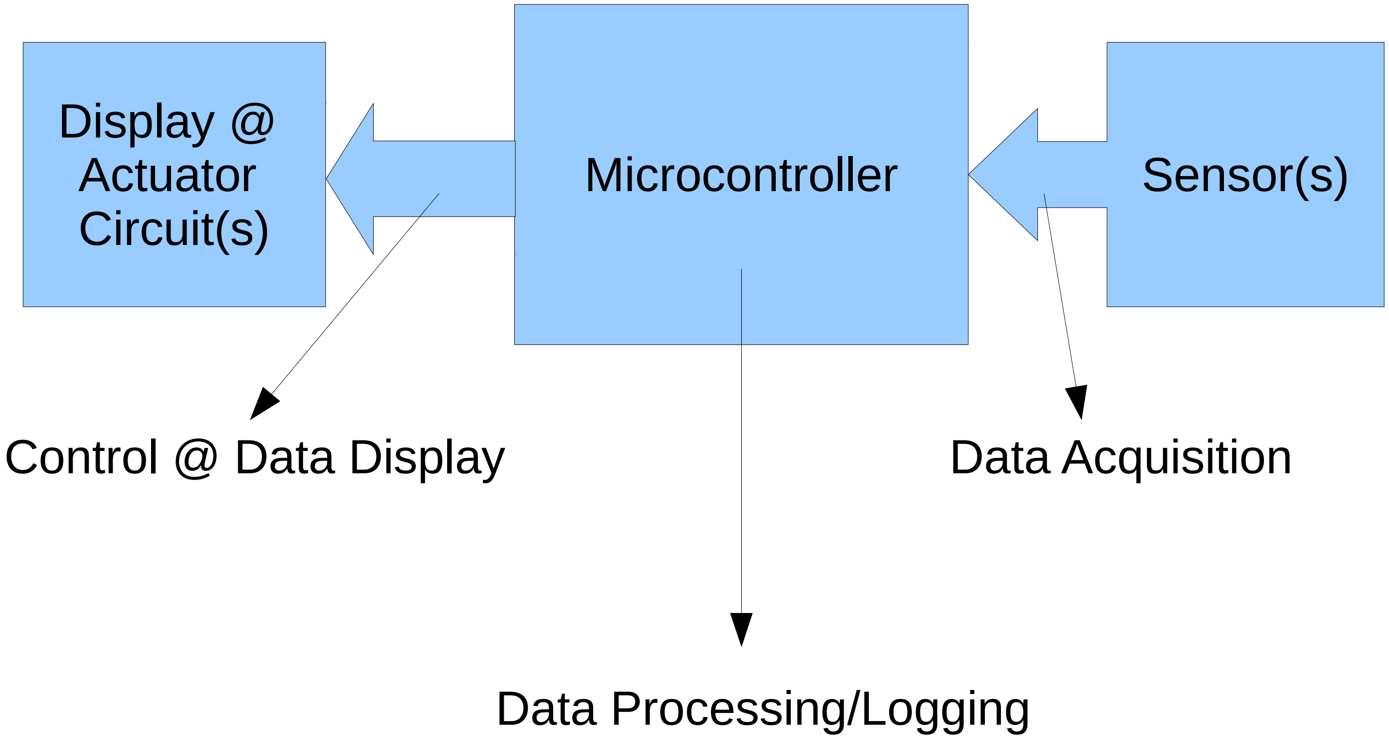
- All communications start with client → server requests
- OSI layers for flow
- Most common server is the Web Server (HTTP)
- Servers are the software – hardware can be the same!



OSI Layers (Side Track!)

- Physical – bit-level transmission (medium)
- Data Link – framing, MAC address
- Network – physical → logical, IP address
- Transport – flow control, TCP/datagram
- Session – manage connections (socket!)
- Presentation – format conversions
- Application – protocols like HTTP, FTP

Common Electronics Project



Display @ Actuator Circuit(s)

Microcontroller

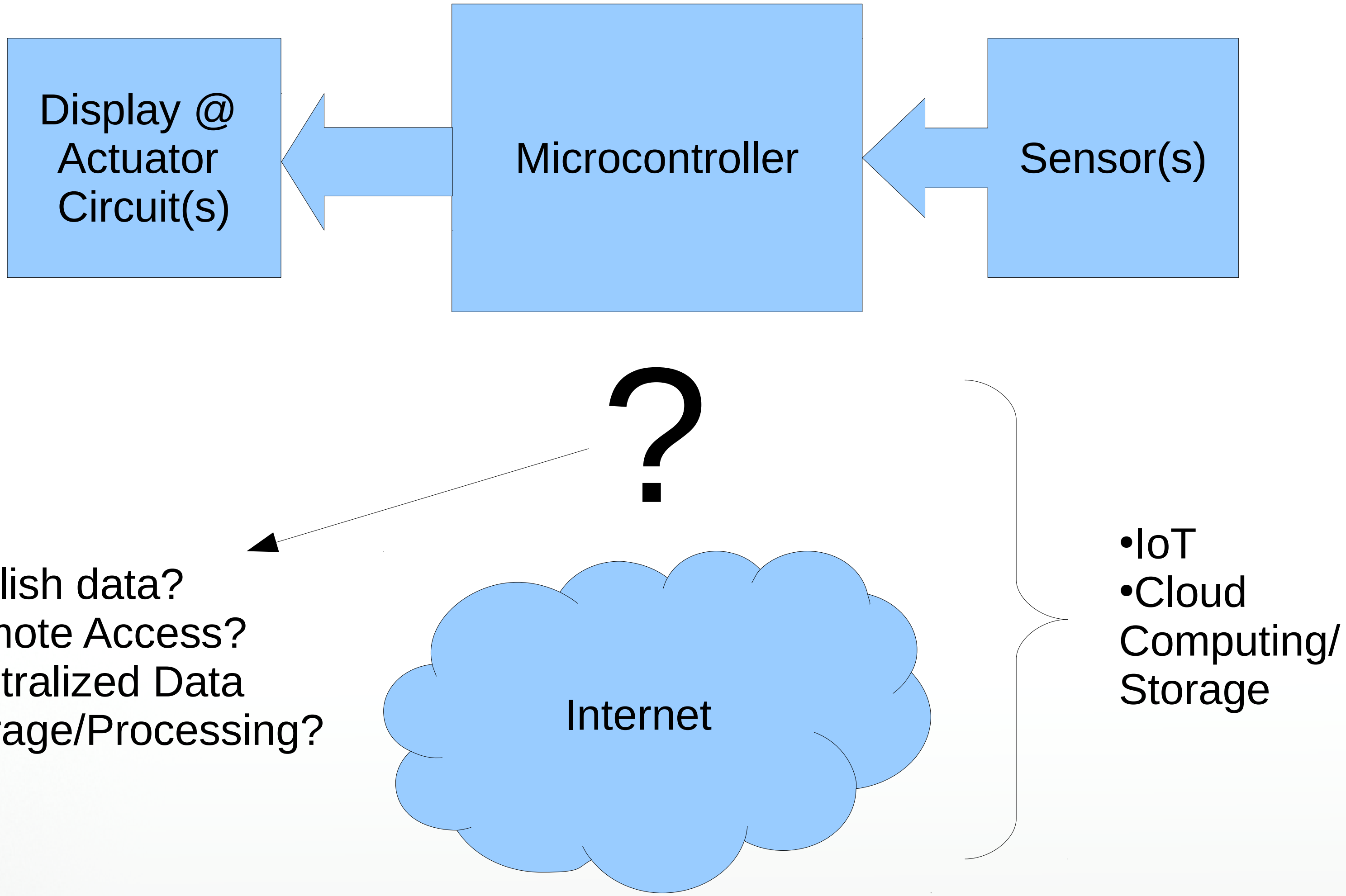
Sensor(s)

Control @ Data Display

Data Acquisition

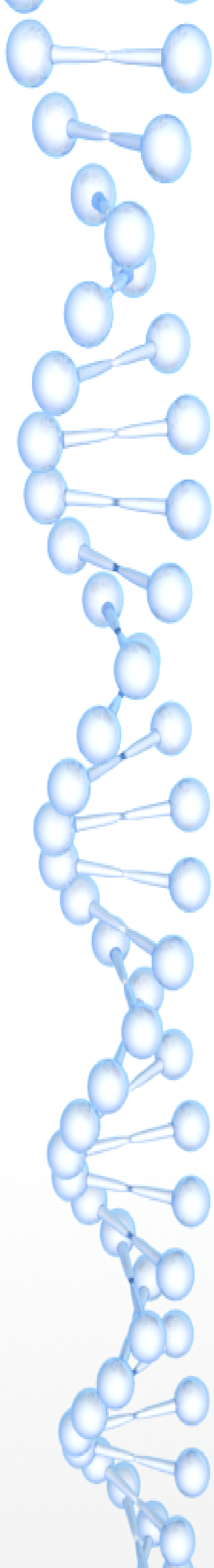
Data Processing/Logging

Common Electronics Project



Publish data?
Remote Access?
Centralized Data
Storage/Processing?

- IoT
- Cloud Computing/ Storage



Data Server for Embedded Systems

- Embedded Server
  - The same software that does data acquisition, serves data
  - C programing is more efficient (PHP possible)
  - Javascript (NodeJS/NWJS) getting popular
- Pure Data Server
  - Software only handles data transactions (no hardware interfacing)
  - PHP is great here
  - Javascript (NodeJS) also capable

Playtime 4

- What we need
  - Stuffs from Playtime 2
  - But, a switch instead of LED (or temp sensor?)
  - Codes from my1codelib
- What we do
  - See how socket works... can be web server!
  - Get data and serve!
- Note:
  - Servers need fixed IP (or hostname)