Using Raspberry Pi as Data Server for Embedded Systems

This page is intended for the participants of '*Using Raspberry Pi as Data Server for Embedded Systems*' training course (April 3-4, 2018).

Updated20180402: The course has been postponed to April 4-5, 2018.

Raspberry Pi has gained considerable interest among researchers, specifically in using it as a platform for systems development. Most research projects are currently leaning towards having some kind of data collection and processing system, that is also preferably capable of serving the information on the internet. A Raspberry Pi board has the perfect set of features that fits into that category - it has sufficient general purpose I/O (GPIO) pins for interfacing, 32-bit ARM processor for processing data, and capable of running server applications.

This is actually an intermediate-level course that requires participants to have some basic understanding of various system-level hardware and software knowledge of a practical application. However, this course will try to present the absolute necessary knowledge that will hopefully allow a project (that requires such setup) to get started.

This course covers fundamental ideas related to the following topics:

1. basic knowledge on raspberry pi
    - bare-metal vs operating systems
2. raspberry pi on Linux
    - plethora of possibilities...
3. programming stuffs
    - database access using sqlite
    - socket programming (my1sock)
4. server-side programming
    - using php (e.g. my1apisrv)
    - using c (e.g. mongoose/my1goose)
5. client-side programming
    - html/css (web page development)
    - javascript

Course Objective

This course is designed to provide some basic insights on how a Raspberry Pi can be used as a data server.

Learning Outcomes

Upon completion of this course, the participants should be able to:

1. Understand the basic requirements in implementing a basic data server
2. Implement a simple peripheral interfacing for collecting data
3. Implement a simple database for data storage
4. Implement a simple server-side code to serve data on a computer network
5. Implement a simple client-side code to display data on a client computer

Who Will Benefit From This Course

This course is designed for engineers, researchers, system designers, technical specialists, graduate students and individuals who are interested in developing fundamental skills on embedded systems development, especially towards IoT compliance.

Related keywords: Raspberry Pi, Data server, API server, server-side programming, Web Server, client-side programming.

Course Content

Sessions	Details	Materials	Notes

Session 1	• Introduction
	  ◦ Raspberry Pi
	  ◦ Bare-Metal vs Using OS
	  ◦ Running Linux OS
	• Data Server Systems Overview
	  ◦ Architecture 1: Embedded Server
	  ◦ Architecture 2: Pure Data Server
		Session 1 Slides	Playtimes 1 - 4
			Some basic socket programming here

Session 2	• Simple database access using SQL
	  ◦ using sqlite
	• File access
	  ◦ device access
	  ◦ alternative database
		lecture slides preparation in progress...	Playtimes 5 - 8
			Some basic SQL coding here

Session 3	• Server-side Programming
	  ◦ PHP API server
	  ◦ C web server (using mongoose/my1goose)
		lecture slides preparation in progress...	

Sessions	Details	Materials	Notes

Session 4	• Client-side Programming
	  ◦ HTML/CSS/Javascript
	• Graphical Display
	  ◦ D3js
		lecture slides preparation in progress...	

Course Resource

Most of the things here will be prepared by the instructor. This is basically for your reference in case you would like to fully experience building the system.

Preparing the SD Card

Note: The term SD card mentioned here generally covers/means the microSD card.

We will be using Raspbian (the official Linux distribution for Raspberry Pi). This enables us to run web servers and other network-related stuffs.

[201804011654] *Note: I just noticed there is now an option to use Windows10 IOT Core (which is prepared by Microsoft as a third party option), but I will not be using that here. Maybe in the future?*

Checkout this page for other options

- Get the image from https://www.raspberrypi.org/downloads/raspbian/
  - latest is Raspbian Stretch
  - this is a ZIP file containing an image file - currently 2018-03-13-raspbian-stretch.zip
- Extract (unzip) the image (file with *.img extension)
  - e.g. 2018-03-13-raspbian-stretch.img
- Write the image to SD card
  - Insert the SD card to your SD card reader
  - (Linux) assuming the device is at /dev/sdb
  - (Linux) use `dd if=2018-03-13-raspbian-stretch.img of=/dev/sdb` and wait...
  - (Windows) use Win32DiskImager tool (or any other tool that can raw-write to the SD card)
  - currently, at least 8GB SD card is required...

Some notes:

- I used the 2017-09-07-raspbian-stretch.img and got an error while trying to update
  - problems seem to be with storage space (98% usage)
  - turns out the partition for root fs was only 4.9GB
- to resize the partition, use fidsk
  - assume card is /dev/sdb (i use USB card reader)
  - `fdisk /dev/sdb`
  - delete partition 2 and recreate (make sure use the same start sector!)

    - save and exit
    - clean the fs `e2fsck -f /dev/sdb2`
    - resize `resize2fs /dev/sdb2`

Booting Raspbian

- Simply insert the SD card and power up
    - It will take some time, but a Linux Desktop will appear on screen.
- Initial Configuration:
    - locale set to US
    - timezone changed to Asia/Kuala_Lumpur
    - keyboard layout set to US (English)
- Run update `sudo apt update` and upgrade `sudo apt upgrade` if necessary
    - if error occurs, most probably problem with the mirror (change to another mirror)

Getting the tools

- Get kernel headers (compiling loadable kernel module)
    - *`sudo apt install`* `raspberrypi-kernel-headers`
- To be able to compile codes that require sqlite
    - *`sudo apt install`* `sqlite3 libsqlite3-dev`
- get web server with php
    - *`sudo apt install`* `apache2`
        - default path for web is `/var/www/html`
    - *`sudo apt-get install`* `php libapache2-mod-php`
- (already there?) To make Pi hostname visible on local network, get avahi-daemon
    - *`sudo apt install`* `avahi-daemon`
    - this is part of Bonjour/Rendezvous/ZeroConf multicast DNS (aka mDNS) services
    - change hostname as desired
        - use raspi-configuration tool
    - clients need to support mDNS as well:
        - windows need this
        - linux require avahi (for slackers, slackbuilds.org got this)

Get some codes

- Get my source codes (e.g. download everything @/home/pi/Work)

```
mkdir -pv /home/pi/Work && cd /home/pi/Work
git clone git://github.com/azman/my1codelib.git
git clone git://github.com/azman/my1webapp.git
git clone git://github.com/azman/my1apisrv.git
git clone git://github.com/azman/my1linuxpi.git
git clone git://github.com/my1matrix/my1goose.git
```

Course Playtime Sessions

These are the code snippets / instructions for the practical (playtime) sessions!.

Playtime 2

```
Note: sysfs interface for GPIO is marked DEPRECATED!

Useful variable - assume we want to access GPIO26 (26 is NOT pin number)

# export GPIO=/sys/class/gpio
# export GPIO_NUM=26
# export GPIO_PATH=$GPIO/gpio$GPIO_NUM

To capture/enable GPIO access (IF still disabled)

# [ ! -d $GPIO_PATH ] && echo $GPIO_NUM > $GPIO/export

To check if the interface has been created

# ls -l $GPIO_PATH

By default, a GPIO pin acts an input.
To configure a GPIO pin as output

# echo out > $GPIO_PATH/direction

To set a GPIO pin to logic HI (Vdd)

# echo 1 > $GPIO_PATH/value

To set a GPIO pin to logic LO (GND)

# echo 0 > $GPIO_PATH/value

To blink it (press CTRL-C to stop)

# while true ; do echo 1 > $GPIO_PATH/value ; sleep 1 ; echo 0 >
$GPIO_PATH/value ; sleep 1 ; done

To configure a GPIO pin back as input

# echo in > $GPIO_PATH/direction

To read current value

# cat $GPIO_PATH/value
```

```
To release/disable GPIO access (IF enabled)

# [ -d $GPIO_PATH ] && echo $GPIO_NUM > $GPIO/unexport
```

Playtime 4

```
Get to my1codelib folder

# cd /home/pi/Work/my1codelib

Compile test socket program

# make my1sock_test

To run the server test program

# ./test_my1sock --server

At the browser, type "http://localhost:8080/test/this/path"

To run the client test program (in another terminal)

# ./test_my1sock --port 8080 --host localhost --path /test/that/path

To show the client can control the server,

# ./test_my1sock --port 8080 --host localhost --path /exit

We can also do that at the browser, type "http://localhost:8080/exit"
```