IoT on Dragonboard410c – Terminology and basics

This cheat sheet documents the necessary terminology and other useful information which can be referred while developing an Internet of Things solution using the SnapDragon based Dragonboard 410c. Its organized in four broad sections namely

- Key Terms and Definitions related to the Internet of Things, the Dragonboard 410c,Development environment and Programming languages
- Useful Linux OS commands which would be used typically while developing an IoT solution using Dragonboard 410c
- Programming languages for real world problems related to Internet of Things
- Broad rules to be followed for IoT Development

I) Key terms and definitions

General Terminology:

IoT - Internet of Things refers to a world in which objects, humans and other life forms communicate and send information without human to human and human to computer interaction.

IDE - Software Application which provides comprehensive tools for software development

SDK - Software Development Kit used to create applications for certain software/hardware platforms such as Android, iOS.

SoC - An integrated circuit chip with all components of a computer on a single chip. Eg: Dragonboard 410c

Microcontroller - A small computer on a single integrated chip with programmable I/O Peripherals. Does not contain a full fledged OS like a SoC. Eg: Adruino

Dragonboard-410 Terminology:

GPIO - 40 Pin General Purpose Input/Output can be used for user specified purposes. Sends/receives digital (1/0 or on/off) signals.

MPP - Multi Purpose pins can send wide range of signals in analog format.

UART - Universal Asynchronous Receiver/Transmitter for serial (bit by bit) and parallel (all at once) communication

SPI - Serial Peripheral Interface for short distance serial communication

USB port - Two USB ports, normally used to attach a keyboard and mouse to the board

microUSB port - One MicroUSB port, used mainly for the Android Debug Bridge

HDMI port - One HDMI port, allows you to connect the board to a monitor/screen

SD Card - Used for external storage or can contain an OS to boot off. Board has a micro SD card

slot for usage.

eMMC - Embedded Multimedia Card, flash memory on the board

ARM - Advanced RISC Machine, instruction set used in dragonboard's processor

Dip Switches - Boot switches on back of board for deciding the boot sequence such as boot of embedded memory, boot of sd card

Development terminology:

Fastboot - A method to install an OS using a USB and a computer
Flashing - Overwriting existing OS or firmware
Terminal - A shell on Unix systems where commands can be executed
Script - An executable file which automates a process

II) Useful Linux OS commands

ls -lt lists the contents of the current directory sorted by modification time
cd changes directory; move to a different place on your system
cat outputs contents of a file to somewhere else (terminal default)
echo prints/calls something out to the terminal
export used to create a path variable
sudo execute a command as a superuser
(sudo) su enter superuser mode and become root (putting sudo in front of everything)
chmod changes file permissions (read, write, execute)
exit - exit current mode (exit root or quits terminal)
mkdir - Create a new folder
mv move a file to another location
cp copy a file to another location
pwd prints out the current (working) directory you are in
rm remove a file
rmdr removes an empty directory (will not allow you to delete a folder with items in it)
rmdir removes a directory (folder) along with its contents
clear - clears the terminal
man displays a manual of a command (i.e. man ls will explain how ls is used/works)
!! repeat a command (can also use arrow keys to cycle through previous commands)
history Lists the commands executed in the shell
!n Refer to command line n
grep Search file(s) for lines that match a given pattern
egrep Search file(s) for lines that match an extended expression
file Determine file type
find Search for files that meet a desired criteria
file Determine file type
find Search for files that meet a desired criteria
gzip Compress or decompress named file(s)
lsof List open files
mount Mount a file system
nice Set the priority of a command or job
reboot Reboot the system
ulimit Limit user resources

unrar Extract files from a rar archive
unzip Extract files from a zip archive
tar store and extract files from a disk archive
vi Text Editor
xargs Execute utility, passing constructed argument list(s)
ssh Secure Shell client (remote login program)
scp remote secure copy of a file
exit Exit the shell
top List processes running on the system
free Display memory usage
~ home directory
. current directory
.. parent directory
./executes files

III) Criteria for programming language choice related to IoT solution

- Low level language
  Assembly - Use only when optimization for speed and optimization for size is required in sensor modules where provisions to write C or high level coding languages are not available. Mostly it must be code that interacts directly with the hardware

- Mid level language
  C - Conducive for programming on systems and hardware. Since C is in between low and high level programming, it gives you more control over your memory and hardware like a low level language while still maintaining many of the components of a high level language. More efficient on space and size compared to C++ and other high level languages

- High level language
  C++/Java - Highly extensible provides for inheritance with a longer planning phase. Can write programs that can scale and reuse code easily. Mostly choose this for any IoT solution on a hub such as dragon board, where data from different sensors is manipulated and interconnections arise.

IV) Other broad rules for developing an IoT solution

a) Choose a IoT framework such as alljoyn which would enable devices to interconnect in a generic manner. Do a case studies of the current frameworks such as alljoyn,IoTivity, Brillo and choose the framework which can extend easily and can be supported on a number of devices.

b) Design a development framework with a pluggable architecture so that adding new sensors or removing sensors for a particular solution is easier

c) Use versioning system such as git for maintaining source code and documents. Revert to snapshots if any problem arises.

d) Use review systems such as geritt while working in teams

e) Use project management systems such as Asana to monitor due dates for tasks and task management