**Go and C**

**System programming**

System programming languages are designed for writing system software, which requires a different development approach when compared with application software.

Smartphone Apps are separate third-party applications written after the Smartphone was introduced. Writing it would have been applications programing however for the display driver of the smart phone, writing it would have been systems programming. Graphics display are often implemented in the kernel or even to the hardware as circuit design.

As technology evolves circuit design will become less and less important as processors become faster. The challenge will be at the system level, how to build systems that scale up or out to improve performance.

http://www.cisco.com/c/en/us/solutions/internet-of-things/overview.html

**Internet of Things (IOT)**

IOT-> computers that are embedded in machines with microcontrollers that run on software

Vertical IT —> Technology solutions that are specially tailored for individual sectors, eg automotive

Horizontal IT —> suited to many sectors at the same time, for example Microsoft Office or software for purchasing, sales, resource scheduling and database management.

Vertical IT builds on embedded systems and in the process offers users additional features. For instance, embedded intelligence, a doctor using a modern CT scanner can view the images created by the machine on a local monitor, or the data can be transferred to a diagnostic software package by means of an integrated external interface.

Customers benefit from vertical IT’s advantages. For example, one automaker has succeeded in cutting its development time almost in half by using end-to-end engineering solutions.

IOT -> expect that by 2020 there will be **50 billion** connected devices.

IOT --> heading for the smart grid

IOT --> cars able to communicate with one another and with their environment in order to avoid accidents and optimize traffic flow.

IOT --> revolution in manufacturing.

GoLang with its evolving and sophisticated [**one language toolchain**](https://golang.org/doc/go1.5)  is poised to become the IOT language of the future

In this set of videos Go language bindings, we will show how , like Java as the glue that binds the enterprise application space, Golang will become the glue that binds the IOT space.

**Install gcc**

**sudo apt-get update && sudo apt-get install -y build-essential**

# Eduonix Build Tools Series

**https://github.com/nsavageJVM/cbuildtools**

This is the basic pattern.

* C source code process.c -> object file
* C main application entry point source code main.c
* Link the object file to the main application and create an executable or library

This is the minimal set of commands to illustrate this

Object

gcc process.c -c

Link

gcc process.o main.c -o myApp

Run

./myApp

**A linker links several object (and library) files to generate an executable.**

an important concept to impart here is the gcc command runs a binary the acts like a driver, invoking binutils to do the work to create the machine code and execute various other optimisations depending on the options we input to gcc from the command line.

**Now let's extend this understanding with some go code**

**go build -buildmode=c-shared -o cbinding.so cbinding.go**

**gcc -Wall -o main main.c ./cbinding.so**

**Go Linker** https://golang.org/doc/go1.5#link

it now supports situations such as building shared libraries and allowing other

languages to call into Go libraries.