Redes de sensores inalámbricos (RSI)

Plataforma de software: Contiki-NG (parte 2)

Leonardo Steinfeld

Inst. de Ingeniería Eléctrica, Fac. de Ingeniería
Universidad de la República (Uruguay)

NEON
Network of Competence on Internet of Things

Co-funded by the Erasmus+ Programme of the European Union

FACULTAD DE INGENIERÍA

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Objetivos

- Comprender el proceso de compilación
- Comprender la jerarquía de makefiles
- Describir y configurar una aplicación (incluyendo com)
- Usar e incorporar el logging para *debug*.
- Incorporar el servicio de shell en una aplicación y describir su uso.

Agenda

- Estructura de directorios
- Proceso de compilación
- Makefiles
- Configuración de la aplicación
- Configuración de la pila de comunicación
- Funcionalidades útiles: *logging* y *shell*

Referencias

- Contiki-NG Documentation
  - Programming Contiki-NG
    - Repository structure
  - Getting Started
    - The Contiki-NG build system
    - The Contiki-NG configuration system
    - The Contiki-NG logging system
  - Tutorials
    - Logging
    - Shell
- Contiki-NG cheat sheet

Contiki-NG: estructura de directorios

- **os**
  - primitivas del sistema , pila de comunicación, etc.
- **arch**
  - dependiente de hardware (incluye CPU, drivers de plataformas)
- **examples**
  - proyectos prontos para usar (empezar basándose en un ejemplo)
- **tools**
  - no incluidos en el firmware del dispositivo, sino para PC
- **tests**
  - para integración continua (se ejecuta con cada PR / merge)

Contiki-NG: estructura de directorios

- **os**
  - **sys**
    - protothreads, procesos, timers, logs, etc.
  - **dev**
    - drivers generales (restantes en arch).
  - **lib**
    - bibliotecas: listas, colas, etc.
  - **net**
    - implementación de los protocolos de cada capa
  - **services**
    - módulos con procesos, usados por Conitiki-NG o usuarios
  - **storage**
    - sistema de archivos coffee

Contiki-NG: estructura de directorios

- Notas
  - archivos .h y .c en misma carpeta
  - módulos (.h, .c) con procesos o pasivos (funciones)

Proceso de compilación (build system)

- Proceso: crea “imagen de ejecutable“ (elf)
  - incluye: aplicación y Contiki-NG
- Makefile (mínimo, ejercicio clase pasada)

```
# archivo: Makefile
CONTIKI_PROJECT = hola
all: $(CONTIKI_PROJECT)

CONTIKI = /home/leo/work/contiki-ng/
include $(CONTIKI)/Makefile.include
```

archivo .c (proyecto)

target

ruta a **contiki-ng**

makefile “master”

Proceso de compilación

- Salida de compilación
  - hola.$(TARGET)
    - Ejemplo: hola.native, hola.cc26x0-cc13x0
  - build/$(TARGET)
    - Ejemplo: /build/native, build/cc26x0-cc13x0
  - build/$(TARGET)/$(BOARD)/ ($(BOARD) si especificado)
  - build/$(TARGET)/$(BOARD)/hello-world.elf
  - build/$(TARGET)/$(BOARD)/obj

    archivos: .o (object file)
  - build/$(TARGET)/$(BOARD)/obj/.dep

    archivos .d (dependency file)

Proceso de compilación

- Cleaning
  - clean:
    - Removes all compiled files for TARGET
  - distclean:
    - Removes all compiled files for all TARGETs
- Help
  - make help

Proceso de compilación: ejemplo

- **hello-world** para **nativo** (repaso)
  - open file: Makefile, hello-world.c
  - open in integrated terminal (example/hello-world)
  - make
  - Ver carpetas y archivos creados
    - hello-world.native
    - build/native/hello-world.native
    - build/native/*.o,
    - build/native/obj/.deps/*.d
  - Ver contenido de:
    - build/native/obj/.deps/hello-world.d

Proceso de compilación: ejemplo

- **hello-world** para **LP-CC2650**
  - En la terminal
    - make help
    - make targets
    - make TARGET=cc26x0-cc13x0 boards
    - make TARGET=cc26x0-cc13x0 BOARDS=launchpad/cc2650
    - make usage
  - Ver carpetas y archivos creados
  - Opcional (verbose): make clean && make V=1

Makefile

- **Makefile** (proyecto):
  - makefile del proyecto, ubicado en la carpeta del proyecto
  - Permite:
    - Agregar módulos (MODULES)
    - configurar la pila de comunicación, etc.
- Makefiles (top-level)
  - **Makefiles.include**
    - makefile “master”, incluye
      - Makefile.$(TARGET)
      - makefiles para modules en la lista `MODULES` list.
  - **Makefile.help**
    - definiciones de targets `help` y `usage`.

Makefile

- **Makefile.$(TARGET)**
  - $(TARGET) es el nombre de la plataforma
  - reglas específicas para la plataforma (ubicada en arch/platform)
  - lista a archivos CONTIKI_TARGET_SOURCEFILES
  - incluye Makefile.$(CPU) de arch/cpu/$(CPU).
- **Makefile.$(CPU)**
  - $(CPU) nombre del CPU/MCU
  - reglas específicas para el CPU (ubicado en arch/cpu)
- **Makefile.$(MODULE)**
  - $(MODULE) nombre del modulo en carpeta os
- Recordamos: cadena de inclusión desde Makefile.include

Makefile

- Recorrida por Makefile.include

- Atención:
  - No pretendemos seguir todos los detalles

Configuración

- Contiki-NG permite mucha configuración

- Métodos
  - Makefile
  - project-conf.h
    - Makefile chequea si existe y lo incluye

Configuración

- Módulos
  - Variable MODULE en Makefile
  - Ejemplos
    - MODULES += os/services/shell
    - MODULES += super_proyecto/lib/sensor
- Variables útiles
  - PROJECT_SOURCEFILES:
    - otros archivos .c que no son *target*
  - MODULES_REL:
    - módulos en carpeta relativas al proyecto

Configuración: orden de inclusión

- module-macros.h
  - macros de cada módulo
  - para definir constantes (no configurar)
- **project-conf.h**
  - configuración específica del proyecto y para configurar Contiki-NG
- **contiki-conf.h**
  - específico de cada plataforma que incluye CPU
- **contiki-default-conf.h**
  - valores por defecto de parámetros de Contiki-NG parameters.
- .h específicos de los módulos
  - setear valores por defecto no configurado previamente

Configuración

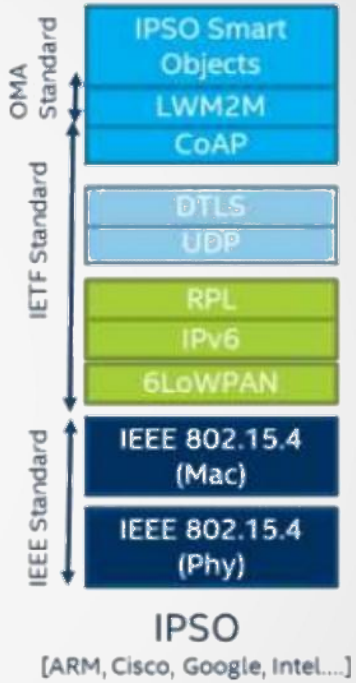
- Ejemplo:
  - Configuración por defecto arch/cpu/cc26x0-cc13x0/cc13xx-cc26xx-conf.h (fragmento)

```
/* Channels count from 0 upwards in IEEE 802.15.4g */
#ifndef IEEE802154_CONF_DEFAULT_CHANNEL
#define IEEE802154_CONF_DEFAULT_CHANNEL          0
#endif /* IEEE802154_CONF_DEFAULT_CHANNEL */
```

  - Se puede *override* desde project-conf.h

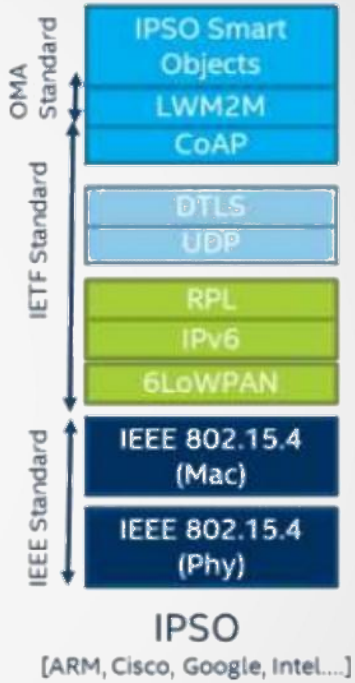
Pila de comunicación: configuración

- Dos capas principales
  - MAC (Medium Access Control)
  - NET (Network) layer
- Otras capas
  - APP
    - MODULES = os/net/app-layer/coap
    - MODULES += os/services/lwm2m
    - MODULES += os/services/lwm2m/ipso-objects
    - MODULES += os/net/app-layer/snmp



Pila de comunicación

- MAC (Medium Access Control)
  - MAKE_MAC: variable
    - MAKE_MAC_NULLMAC
    - MAKE_MAC_CSMA (default)
    - MAKE_MAC_TSCH
    - MAKE_MAC_BLE
- NET (Network) layer
  - MAKE_NET
    - MAKE_NET_NULLNET
    - MAKE_NET_IPV6 (default)
- Routing protocol
  - si IPv6 se configura de forma similar a las anteriores



Logging

- Sistema de logs (printf)
  - niveles diferenciados por módulo
  - modificables en tiempo de ejecución (via shell)
- Niveles
  - LOG_LEVEL_NONE: Logs disabled
  - LOG_LEVEL_ERR: Errors
  - LOG_LEVEL_WARN: Errors and warnings
  - LOG_LEVEL_INFO: Errors, warnings, and information logs
  - LOG_LEVEL_DBG: All the above and debug messages

Logging

- Ejemplo: hello-world

```
leo@carbon:~/work/contiki-ng/examples/hello-world$ ./hello-world.native
[WARN: Tun6      ] Failed to open tun device (you may be lacking permission). Running without network.
[INFO: Main      ] Starting Contiki-NG-develop/v4.9-51-gd11c1f0cf-dirty
[INFO: Main      ] - Routing: RPL Lite
[INFO: Main      ] - Net: tun6
[INFO: Main      ] - MAC: nullmac
[INFO: Main      ] - 802.15.4 PANID: 0xabcd
[INFO: Main      ] - 802.15.4 Default channel: 26
[INFO: Main      ] Node ID: 1800
[INFO: Main      ] Link-layer address: 0102.0304.0506.0708
[INFO: Main      ] Tentative link-local IPv6 address: fe80::302:304:506:708
[INFO: Native    ] Added global IPv6 address fd00::302:304:506:708
Hello, world
```

Logging

- Módulos (ya) soportados
  - MAIN
  - RPL
  - TCPIP
  - IPV6
  - 6LOWPAN
  - NULLNET
  - MAC
  - FRAMER
  - 6TOP
  - COAP
  - SNMP,
  - LWM2M
- Uso en proyectos propios
  - Ejercicio 1)
    - Agregar logs a hello-world siguiendo

      Getting Started » Logging
  - Ejercicio 2)
    - Modificar nivel de logs de hello-world siguiendo

      Tutorials » Logging

Shell

- Intérprete de comandos
  - inspección y mantenimiento interactivo
  - funciona vía puerto serie (virtual com vía USB)
- Incluir en un proyecto
  - MODULES += os/services/shell

Shell

```
> help
Available commands:
'> help': Shows this help
'> reboot': Reboot the board by watchdog_reboot()
'> ip-addr': Shows all IPv6 addresses
'> ip-nbr': Shows all IPv6 neighbors
'> log module level': Sets log level (0--4) for a given module (or "all"). For module "mac",
level 4 also enables per-slot logging.
'> ping addr': Pings the IPv6 address 'addr'
'> rpl-set-root 0/1 [prefix]': Sets node as root (1) or not (0). A /64 prefix can be
optionally specified.
'> rpl-local-repair': Triggers a RPL local repair
'> rpl-refresh-routes': Refreshes all routes through a DTSN increment
'> rpl-global-repair': Triggers a RPL global repair
'> rpl-status': Shows a summary of the current RPL state
'> rpl-nbr': Shows the RPL neighbor table
'> routes': Shows the route entries
'> radio help': Shows radio command usage.
'> leds [on/off led]': Controls the leds.
```

Referencias

- Contiki-NG Documentation
  - Programming Contiki-NG
    - Repository structure
  - Getting Started
    - The Contiki-NG build system
    - The Contiki-NG configuration system
    - The Contiki-NG logging system
  - Tutorials
    - Logging
    - Shell
- Contiki-NG cheat sheet

Planificación clases

1) Introducción RSI
2) Plataformas de hardware
3) Arquitectura 6LoWPAN (IPv6)
4) Plataforma de software: Contiki-NG (parte 1)
5) **Plataforma de software: Contiki-NG (parte 2)**
6) Capa de aplicación: CoAP / MQTT
7) Capa de red: RPL
8) MAC
9) IEEE 802.15.4 / 6lowpan
10) Capa Fisica & antenas
11) IoT y las RSI

¿más preguntas?