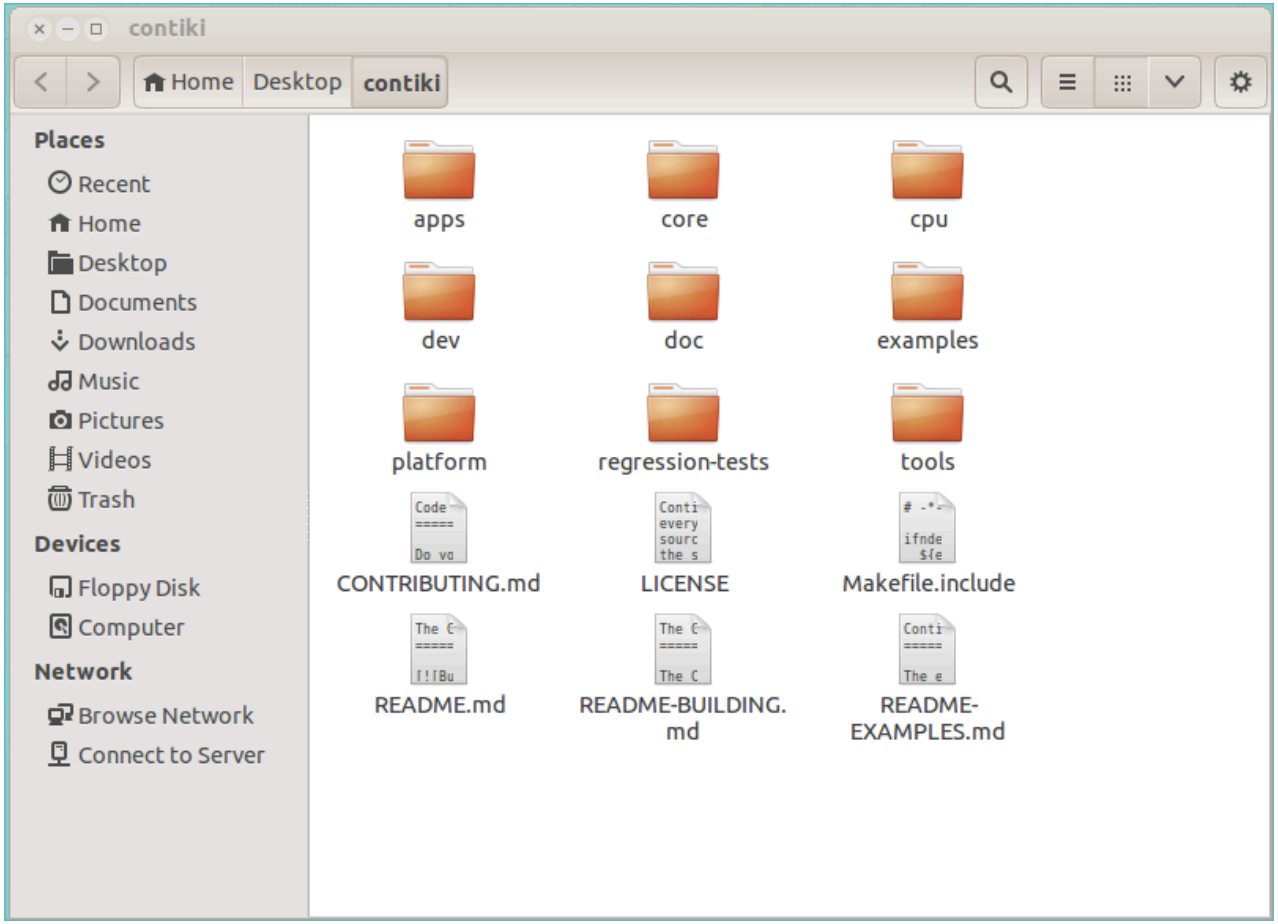
Contiki Tutorial

Contiki Overview

- Based on the introduction tutorial, you already have a basic understanding of Contiki, as follows
  - Complete environment for programming sensor nodes
  - Has everything to get started in making applications
  - Includes built-in simulator named Cooja
  - Large range of sensor compatibility, e.g., T-mote Sky, Z1
- To use Contiki for hands-on practice, it is important to understand the structure of its source code files

Source Code Structure in Contiki

- Folders and files in Contiki



Contiki Folders

- apps/: Contiki applications, such as web browser, shell, etc.
- core/: Contiki filesystem and core components
  - core/dev/: Source code for devices, such as LED, sensors, buttons
  - core/net/: Source code for MAC, RPL routing protocol, IPv6 and IPv4, queuing packets and buffers, etc.
- cpu/: Source code for computation units of sensor nodes
- dev/: Source code files for some hardware platforms, such as cc1200, sht11, etc.
- examples/: Example source code files
- platform/: Specific device files and drivers
- tools/: Variety of tools for debugging, simulating, and testing applications, including Cooja

Hello World Example

Hello World Simulation

- The simplest way to open the simulation is to select “Hello World Simulation” in the IoTrain-Sim interface
- Alternatively, you can open it manually as follows
  - Open Cooja
  - Click File > Open simulation > Browse...
  - Go to the folder “iotrain-sim/database/fundamental_training/single_node/basics_contiki/simulation”
  - Select the file “hello-world.csc”
- Once the simulation control window appears, click the “Start” button to begin the simulation
  - The logical running time for this simulation is set to 5 minutes
  - After the simulation ends, you can use the “Reload” button to reload it, and then the “Start” button again to restart it

Source Code File “hello-world.c”

```
#include "contiki.h"
#include <stdio.h>
/*---------------------------------------------------------------------*/
PROCESS(hello_world_process, "Hello world process");
AUTOSTART_PROCESSES(&hello_world_process);
/*---------------------------------------------------------------------*/
PROCESS_THREAD(hello_world_process, ev, data)
{
  PROCESS_BEGIN();
  printf("Hello, world\n");
  PROCESS_END();
}
/*---------------------------------------------------------------------*/
```

Part I

Part II

Source Code Commentary: Part I

```
#include "contiki.h" /* For Contiki applications */
#include <stdio.h> /* For the printf() function*/
/*---------------------------------------------------------------------------*/
/* Contiki application: Declare the main PROCESS */
PROCESS(hello_world_process, " Hello world process "); ❶
/* Make the process start when the module is loaded */
AUTOSTART_PROCESSES(& hello_world_process); ❷
/*---------------------------------------------------------------------------*/
```

❶ hello_world_process is the actual name of the process, and "Hello world process" is the readable name of the process when printed to the terminal

❷ The AUTOSTART_PROCESSES(&hello_world_process) call tells Contiki to start that process when it finishes booting

Source Code Commentary: Part II

```
/*---------------------------------------------------------------------*/
PROCESS_THREAD(hello_world_process, ev, data) ❶
{
  PROCESS_BEGIN(); ❷
  printf("Hello, world\n"); ❸ /* Do what you want here */
  PROCESS_END(); ❹
}
/*---------------------------------------------------------------------*/
```

❶ Declare the content of the process in the process thread, including the name of the process and callback functions (event handler and data handler)

❷ Begin executing the main processing thread, PROCESS; must always come first

❸ Put the source code corresponding to your logic here

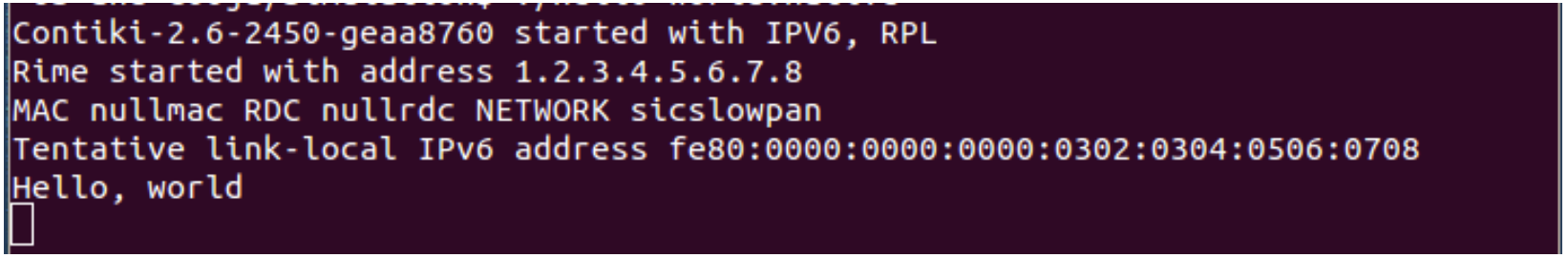
❹ Ends the execution of PROCESS; must always come last

Command-Line Execution

- To run the example via command-line, open a terminal and change to the source code directory “iotrain-sim/database/fundamental_training/single_node/basics_contiki_cooja/simulation”
- The included files are as follows
  - hello-world.c: Source program in Contiki OS
  - hello-world.csc: Simulation file using Cooja simulator
  - Makefile: File that controls the compilation process
  - README.md: Basic instructions for the example

Command-Line Execution (cont.)

- To execute the program, do as follows
  - Compile the program by running the command “make”
  - After processing finishes, run “./hello-world.native”
- The program will output the message “Hello, world” as shown below



- To end the program execution, press Ctrl+C

Makefile Explanation

- A “Makefile” contains instructions on how to compile source code into appropriate object files for execution
- The Makefile for the Hello World example is as follows

```
CONTIKI_PROJECT = hello-world ❶
all: $(CONTIKI_PROJECT) ❷

CONTIKI = /home/user/contiki
include $(CONTIKI)/Makefile.include ❸
```

❶ “hello-world” is the source code file name (more names are OK)
❷ The files defined above should be used to compile the project
❸ The additional file “Makefile.include” contains Contiki-specific compilation information