**Structure d’un programme en Contiki :**

Programmer et concevoir des réseaux de capteurs sans fil sur Contiki repose sur la maitrise des bases de la programmation de ce système d’exploitation, qui propose une bibliothèque spécifique de fonctions et un environnement de développement complet et fonctionnel.

Ainsi, apprendre les bases et les principes de la programmation sur Contiki est indispensable pour arriver par la suite, à rédiger des programmes plus compliqués et sophistiqués.

Voici la structure d’un programme C basique en Contiki qui permet d’afficher une chaîne de caractères sur le terminal :

**1: #include “contiki.h”**

**2:   #include <stdio.h>**

**3:  PROCESS(my\_first\_process, “Hello World Process”);**

**4:  AUTOSTART\_PROCESSES(&my\_first\_process);**

**5:   PROCESS\_THREAD(my\_first\_process, ev, data)**

**6:  {**

**7:   PROCESS\_BEGIN();**

**8:   printf(“Hello WSN World!\n”);**

**9:   PROCESS\_END();**

**10:   }**

L’entête  **#include “contiki.h”** contient toutes les déclarations nécessaires aux abstractions de Contiki.

**#include <stdio.h>** est déclarée pour pouvoir appeler la fonction *printf().*

**PROCESS (my\_first\_process, “Hello World Process”);** permet de déclarer un nouveau processus Contiki nommé « *Hello World Process* » et ayant comme variable de process *my\_first\_process*.

**AUTOSTART\_PROCESSES(&my\_first\_process);** permet de lancer automatiquement le process *my\_first\_process* au démarrage du système d’exploitation. Cette méthode peut avoir comme paramètres une liste de process qui vont être lancés automatiquement dans l’ordre.

**PROCESS\_THREAD (my\_first\_process, ev, data) :** c’est la définition de notre process. Le premier paramètre est la variable *my\_first\_process*, le deuxième *ev* est un évènement, qu’on peut utiliser pour permettre au programme de répondre à des évènements relatifs au système. Le troisième paramètre *data* est une variable de donnée qu’on peut recevoir avec l’évènement.

**PROCESS\_BEGIN();** déclare le commencement du process’thread.

**printf(“Hello WSN World!\n”);** affiche la chaîne de caractère *Hello WSN World !*

**PROCESS\_END();** déclare la fin du process’thread.

Pour compiler le programme, on a besoin d’un fichier *Makefile*, qui est un fichier sans extension, contenant les instructions nécessaires à la compilation. Dans ce fichier, nous devons spécifier la version de Contiki utilisée, les applications (programmes C) prises en charge par ce fichier et inclure le *Makefile* qui existe dans le répertoire du root.

**1:  CONTIKI=../..**

**2 : all : app-name**

**2:  include $(CONTIKI)/Makefile.include**

**CONTIKI=../..**: spécifie l’emplacement le root (l’administrateur) de Contiki

**all : app-name :** spécifie les applications (programmes C) à compiler

**include $(CONTIKI)/Makefile.include :** permet d’inclure le *Makefile* qui existe dans le répertoire du root

**Compilation du programme :**

Sur le terminal, la compilation et l’implémentation du programme sur un module Zolertia z1 consiste en 3 commandes :

**make Target=z1 application.c :** spécifier z1 comme plateforme d’exécution du programme.

**make Target=z1 savetarget**: enregistrer la plateforme d’exécution pour les prochaines applications prises en charge par le même fichier *Makefile*.

**make application.upload :** compiler et implémenter le programme sur z1 pour l’exécuter.