**Report**

**Original application**

Functionality

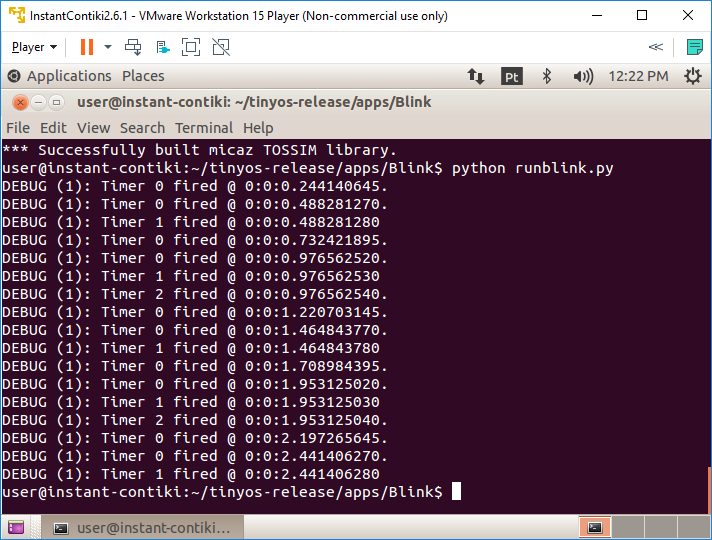
In the module defined in the ​BlinkC.nc file are being specified the interfaces of the components to be used. The Blink module uses three timers *(*Timer0, Timer1, Timer2) with different frequencies; Leds (which operations can be accessible using dot notation); and Boot (related to the booting phase of the device). The original applications sets three timers with 250ms, 500ms and 1000ms periods. Every time each of these are fired to start again, a function linked to this event is triggered. For each timer there is a function that writes the number of the timer it is, and the time it has passed since the simulation started. Also it toggles a led. For timer 0, it is led0; for timer 1, led 1; and for timer 2, led 2.

There is the actual module implementation, which consists of a few events to be handled:

i. **Boot.booted()** handles setup operations to be carried out after booting. In particular,  
the timers are initialized at different frequencies.

ii. ​**TimerX.fired()** defines what to do every time the timer goes off, in this case printing a debug message to mark the event firing and then toggling the corresponding LED.

In ​**BlinkAppC.nc** we can find the declaration and initialization of the ​**Timer** components - provided through the OSKI TimerMilli service - to be used by the ​**Blink** module, as well as the booting and LEDs components. These components have to satisfy the interfaces expected by the ​**Blink**​ module for the application to work properly. Finally, the ​**runblink.py** file specifies how the simulation is going to be conducted, the simulated booting time, and the number of events to be fired during the simulation.



**Modified application**

**BlinkApp.nc**

i.Add a new timer component:

components new TimerMilliC() as Timer3;

ii. Assign it to the Blink module:

BlinkC.Timer3 -> Timer3;

**BlinkC.nc**

i. Specify a new interface for the timer to be added:

uses interface Timer<TMilli> as Timer3;

ii. Start the timer after boot with a period of 100 ms:

call Timer3.startPeriodic( 100 );

iii. Define the firing event callback with a debug message:

event void Timer3.fired()

{

dbg("BlinkC", "I am Timer 3 and I have the shortest period!.

Fired @ %s.\n", sim\_time\_string());

}

**runblink.py**

Increase the number of the simulated events to 2000:

for i in range(0, 2000):

    t.runNextEvent()

