Hőmérő alkalmazás készítése

Feladatunk egy olyan alkalmazás készítése, amely a TelosB mote-okon található Sensirion SHT11 hőmérséklet és páratartalom érzékelő szenzor segítségével fél másodpercenként méri a környezet hőmérsékletét, majd a TelosB mote-on található rádió segítségével továbbítja a nyers hőmérséklet adatokat a számítógéphez csatlakoztatott basestation mote felé. Minden egyes mérés elvégzésekor a 0-ás led-et, kapcsolgassa a mote. Ha a hőmérséklet 30 $^{\circ}C$ fok alatt van, akkor a 2-es led-et, ha 30 $^{\circ}C$ fok felett, akkor az 1-es led-et kapcsolja be a mote. A basestation mote a beérkező nyers hőmérséklet adatokat a soros port-on keresztül továbbítja a számítógépnek, ahol ezeket az adatokat a *Homero.java* alkalmazás segítségével konvertáljuk át és jelenítjük meg a konzolon.

Az alkalmazás elkészítésének első lépékén hozzunk létre egy *HomeroC.nc* file-t mely a következő konfigurációt tartalmazza:

```
#include "Homero.h"

configuration HomeroC
{}
implementation
{
	components MainC;
	components LedsC;
	components new TimerMilliC();
	components new SensirionSht11C();
	components ActiveMessageC;
	components new AMSenderC(AM_HOMEROMSG);
	components HomeroP;

	HomeroP.Boot-> MainC;
	HomeroP.Leds -> LedsC;
	HomeroP.Timer -> TimerMilliC;
	HomeroP.Read -> SensirionSht11C.Temperature;
	HomeroP.SplitControl-> ActiveMessageC;
	HomeroP.AMSend->AMSenderC;
}
```

Az itt látható komponensekkel, egy kivételével a *BlinkToRadio* alkalmazás esetén már megismerkedhettünk. Az egyetlen még nem használt komponens a *SensirionSht11C*, mely egy generikus komponens, és a Sensirion Sht11-es hőmérséklet és páratartalom szenzorról történő adatkiolvasást valósítja meg. A komponens által biztosított interface-ek közül a *Temperature* interface-t használjuk, mely egy *Read* interface. A Read interface a *(TOSROOT)/tos/interfaces* mappában található. Ha megnyitjuk a *Read.nc* file-t akkor láthatjuk, hogy tartalmaz egy *read()* parancsot, illetve egy *readDone(error_t result, uint16_t val)* event-et. A *read()* parancs segítségével tudunk elindítani az adatok kiolvasását a szenzorról. Az adatok kiolvasásának végét a *readDone()* event jelzi.

Második lépésként hozzuk létre a *Homero.h* header file-t, melyben definiáljuk a rádión elküldendő csomag AM típusát (*AM_HOMEROMSG*), és a csomag felépítését.

```
#ifndef HOMERO_H
#define HOMERO_H

enum{
	AM_HOMEROMSG = 6,
};

typedef nx_struct HomeroMsg{
	nx_uint16_t Celsius;
}HomeroMsg;
```

```
#endif
```

Harmadik lépésként hozzuk létre a *HomeroP.nc* file-t, mely az alkalmazásunkat fogja tartalmazni. Ennek a file-nak a következőeket kell tartalmaznia:

```
#include "Homero.h"

module HomeroP
{
	uses interface Boot;
	uses interface Leds;
	uses interface Timer<TMilli>;
	uses interface Read<uint16_t>;
	uses interface SplitControl;
	uses interface AMSend;
}
implementation
{
	bool busy = FALSE;
	message_t pkt;

	event void Boot.booted()
	{
		call SplitControl.start();
	}

	event void SplitControl.startDone(error_t err){
		if(err==SUCCESS){
			call Timer.startPeriodic(500);
		}else
		{
			call SplitControl.start();
		}

	}

	event void SplitControl.stopDone(error_t err){}

	event void Timer.fired(){
		call Leds.led0Toggle();
		call Read.read();
	}

	event void Read.readDone(error_t result, uint16_t val ){
		int32_t Temp;
		if(result==SUCCESS){
			Temp=-3960+(int32_t)val;
			if(Temp>=3000){
				call Leds.led1On();
				call Leds.led2Off();
			}else{
				call Leds.led2On();
				call Leds.led1Off();
			}
			if(!busy){
				HomeroMsg* btrpkt = (HomeroMsg*)(call
AMSend.getPayload(&pkt, sizeof(HomeroMsg)));
				btrpkt->Celsius = val;
				if(call AMSend.send(AM_BROADCAST_ADDR, &pkt,
sizeof(HomeroMsg))==SUCCESS){
				busy = TRUE;
				}
			}
```

```
		}
	}

	event void AMSend.sendDone(message_t* msg, error_t error) {
		busy = FALSE;
	}
}
```

Ebben az alkalmazásban a már megszokott módon, ha a mote elindult, akkor bekapcsoljuk a rádiót. Ha a rádió sikeresen elindult, akkor elindítunk egy timer-t a *Timer.startrPeriodic()* utasítással, úgy hogy 500 milliszekundumonként generálódjon egy fired event. Minden alkalommal, amikor egy *Timer.fired()* event keletkezik, elindítunk egy kiolvasást a hőmérséklet szenzorról a *read()* parancs segítségével. Ha a kiolvasás megtörtént, azaz egy *readDone()* even-t generálódott, és nincs semmilyen hiba, akkor elvégezzük a nyers hőmérséklet adatok konvertálását a $Temerature = -39.6 + 0.01 \cdot RawTemerature$ képlet alapján, és megvizsgáljuk, hogy a kapott érték kisebb, vagy nagyobb, mint 30 $^{\circ}C$. Az összehasonlítás eredményének megfelelően bekapcsolunk egy led-et, és elküldjük a mért nyers hőmérséklet értéket a rádión keresztül.

Negyedik lépésként készítsük el a *Makefile*-t mely tartalmazni fogja a *mig* eszköz meghívását is.

```
COMPONENT=HomeroC
BUILD_EXTRA_DEPS = HomeroMsg.class
CLEAN_EXTRA = HomeroMsg.class HomeroMsg.java

HomeroMsg.class: HomeroMsg.java
	javac HomeroMsg.java

HomeroMsg.java: Homero.h
	mig java -target=null -java-classname=HomeroMsg Homero.h HomeroMsg -o
$@
include $(MAKERULES)
```

Végül a Homero.java kódot kell elkészítsük az alábbiaknak megfelelően:

```
import static java.lang.System.out;
import net.tinyos.packet.*;
import net.tinyos.message.*;
import net.tinyos.util.PrintStreamMessenger;
import java.io.*;

class Homero implements MessageListener{

	private MoteIF moteIF;

	public Homero(MoteIF moteIF){
		this.moteIF=moteIF;
		this.moteIF.registerListener(new HomeroMsg(),this);
	}

	public void messageReceived(int dest_addr,Message msg){
		if (msg instanceof HomeroMsg) {
			HomeroMsg homeroData = (HomeroMsg)msg;
			out.println("A	homersekelt:	"	+	(-
39.4+(0.01*(double)homeroData.get_Celsius())));
		}
	}

	public static void main(String[] args) throws Exception
	{
		PhoenixSource phoenix = null;
		MoteIF mif = null;
```

```
            if( args.length == 0 ){
                  phoenix                                                    =
BuildSource.makePhoenix(PrintStreamMessenger.err);
            } else if( args.length == 2 && args[0].equals("-comm") ) {
                  phoenix        =         BuildSource.makePhoenix(args[1],
PrintStreamMessenger.err);
            } else {
                  System.err.println("usage:     java     Homero     [-comm
<source>]");
                  System.exit(1);
            }
            mif = new MoteIF(phoenix);
            Homero app= new Homero(mif);
      }
}
```

Utolsó lépésként programozzuk fel a TelosB mote-ot a *Homero* alkalmazással, és egy másik TelosB mote-ot pedig a *BaseStation* alkalmazással a `make telosb install` utasítás segítségével. Indítsuk el a java alkalmazásunkat is a `java Homero –comm serial@/dev/ttyUSB0` paranccsal.