Portierung von TinyOS CoAP auf die UCMotes Hardware

1. Folgende Verzeichnisse mit den entsprechenden aus dem Development Release ersetzen:

    /apps/CoapBlip
    /support/make/coap.extra
    /support/sdk/c/coap
    /tos/interfaces/CoAPClient.nc
    /tos/interfaces/CoapResource.nc
    /tos/interfaces/CoAPServer.nc
    /tos/interfaces/LibCoAP.nc
    /tos/lib/net/coap

2. Folgende Zeilen auskommentieren, da sie Fehler beim Kompilieren hervorrufen:

    /support/sdk/c/coap/**bits.h**
    - #include <sys/types.h>
    + //#include <sys/types.h>

    /support/sdk/c/coap/**net.c**
    - #include <sys/types.h>
    + //#include <sys/types.h>

3. Die Komponente /tos/lib/timer/LocalTimeSecondC.nc bindet Komponente CounterSecond32C ein, welche für den Atm128rfa1 Chip nicht existiert. Daher folgende Datei neu anlegen: /tos/chips/atmrfa1/timer/**CounterSecond32C.nc**

```
configuration CounterSecond32C
{
    provides interface Counter<TSecond,uint32_t>;
 }
 implementation
 {
    components new TransformCounterC(TSecond, uint32_t, T62khz, uint32_t, 6, uint8_t);
    Counter = TransformCounterC;

    components Counter62khz32C;
    TransformCounterC.CounterFrom -> Counter62khz32C;
  }
```

4. Zum Auslesen der Sensoren, die Komponente MeasurementC.nc unter /tos/platforms/ucmini/**MeasurementC.nc** angelegt.

```
/**
 * MeasurementC is a top-level access component for the getting sensor data
 * of the ucmini platform, including the SHT21 humidity and temperature sensor,
 * air pressure, light and voltage. Because this component represents one physical
 * device, simultaneous calls to read temperature and humidity will be
 * arbitrated and executed in sequential order. Feel free to read both
 * at the same time, just be aware that they'll come back
 * sequentially.
 *
 * @author Sebastian Scheibe <Sebastian.Scheibe@imms.de>
 * @version $Revision: 1.0 $ $Date: 2013-12-09 15:46:18 $
 */

generic configuration MeasurementC() {

  provides interface ReadNow<uint16_t> as BatteryVoltage;

  provides interface Read<uint16_t> as Temperature;
  provides interface Read<uint16_t> as Humidity;
  provides interface Read<uint16_t> as Light;
}
implementation {

   components new TemperatureC() as Sht21Temp, new HumidityC() as Sht21Hum;
  Temperature = Sht21Temp.Read;
  Humidity = Sht21Hum.Read;

  components new AtmegaVoltageNowC() as VoltageNow;

  BatteryVoltage = VoltageNow;

  components new LightC() as PhotoSensor;
  Light = PhotoSensor.Read;

}
```

5. Anpassen der Coap-Server-Applikation "CoapBlipC.nc" in /apps/CoapBlip/**CoapBlipC.nc**

```
#if defined (COAP_RESOURCE_TEMP)  || defined (COAP_RESOURCE_HUM) || defined
(COAP_RESOURCE_ALL)
 #if defined PLATFORM_TELOSB
        components new SensirionSht11C() as HumTempSensor;
 #else if defined PLATFORM_UCMINI
        components new MeasurementC() as HumTempSensor;
 #endif
#endif
…
#ifdef COAP_RESOURCE_ALL
 components new CoapReadResourceC(val_all_t, INDEX_ALL) as CoapReadAllResource;
 #if defined PLATFORM_TELOSB
        components new SensirionSht11C() as HumTempSensorAll;
 #else if defined PLATFORM_UCMINI
        components new MeasurementC() as HumTempSensorAll;
 #endif
```

6. Umwandeln der Sensor-Rohwerte im CoapBuffer für SHT21. Im Verzeichnis /tos/lib/net/coap/translate die folgenden Komponenten anpassen:

CoapBufferTempTranslateP.nc:

```
generic module CoapBufferTempTranslateP() {
 provides interface Read<uint16_t> as ReadTemp;
 uses interface Read<uint16_t>;
} implementation {

 command error_t ReadTemp.read() {
  call Read.read();
  return SUCCESS;
 }

 event void Read.readDone(error_t result, uint16_t val) {


 #if defined PLATFORM_TELOSB

 /*
   The calculation of the temperature for TelosB nodes is done according to the datasheet SHT1x
(www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT1x.pdf).
```

```
   T = d1 + d2 * val
   All values are multiplied with 100 to get a fixed-point representation and the final result is in
Kelvin (+ 273.15). In addition, an offset can be subtracted for battery or USB powered nodes */
  val =  23355 + val -200; //Offset: 500 for USB, 200 for Battery
  printf( "CoapBufferTempTranslateP.readDone: %hu \n", val);
  signal ReadTemp.readDone(result, val);

 #else if defined PLATFORM_UCMINI

 /*
  The calculation of the temperature for ucmini nodes is done according to the datasheet SHT2x

(http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensi
rion_Humidity_SHT21_Datasheet_V3.pdf).
  T = -46.85 + 175.72 * ST / 2^16
  All values are multiplied with 100 to get a fixed-point representation and the final result is in °C. */

  val =  ((((17572*((uint32_t)val*100/65536))-468500))/1000);   //Temp in °C*10 to get 1 decimal
after comma
  //val =  ((((17572*((uint32_t)val*100/65536))-468500))/10000); //Temp in °C

  printf( "CoapBufferTempTranslateP.readDone: %hu \n", val);
  signal ReadTemp.readDone(result, val);

 #endif

 }
}


generic module CoapBufferHumTranslateP() {
 provides interface Read<uint16_t> as ReadHum;
 uses interface Read<uint16_t>;
} implementation {

 command error_t ReadHum.read() {
  call Read.read();
  return SUCCESS;
 }

 event void Read.readDone(error_t result, uint16_t val) {


#if defined PLATFORM_TELOSB

 /*
```

```
     The calculation of the relative humidity for TelosB nodes is done according to the datasheet SHT1x (www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT1x.pdf).
     RH = c1 + c2 * val + c3 * val^2
     To avoid floating point calculations and to achieve a precision of 0.01 %, the values c1 and c2 are multiplied with 100 to get fixed point values and value c3 is transformed by 1/x. */
    val =  (-204 + val*4 -((uint32_t)val*(uint32_t)val)*100/628931);
    printf("CoapRead.readDone: %hu\n", val);
    signal ReadHum.readDone(result, val);

 #else if defined PLATFORM_UCMINI

 /*
    The calculation of the humidity for ucmini nodes is done according to the datasheet SHT2x

(http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT21_Datasheet_V3.pdf).
    H% = -6 + 125 * SH / 2^16           */
    //val =  (125*(uint32_t)(val/65536)-6);

    val = (((125*((uint32_t)val*10/65536))-60));                          //relative humidity in %*10 to get 1 decimal after comma
    //val = (((125*((uint32_t)val*100/65536))-600)/100);            //relative humidity in %

    printf( "CoapBufferTempTranslateP.readDone: %hu \n", val);
    signal ReadHum.readDone(result, val);

 #endif
  }
}
```