**Wireless Energy Harvesting for the Internet of Things**

**Software Used:** NetSim Standard v10.1 (32-bit), Visual Studio 2015

Among different energy harvesting methods, such as vibration, light, and thermal energy extraction, wireless energy harvesting (WEH) has proven to be one of the most promising solutions by virtue of its simplicity, ease of implementation, and availability. This recent technology trend in energy harvesting provides a fundamental method to prolong battery longevity. While harvesting from the aforementioned environmental sources is dependent on the presence of the corresponding energy source, RF energy harvesting provides key benefits in terms of being wireless, readily available in the form of transmitted energy (TV/radio broadcasters, mobile base stations and handheld radios), low cost, and small form factor implementation.

A WEH-enabled sensor device usually consists of an antenna, a transceiver, a WEH unit, a power management unit (PMU), a sensor/processor unit, and possibly an onboard battery. The available harvested power, *PH*, is given by a Friis equation and is directly proportional to the transmitted power, *PT*, path loss, *PL*, transmitter antenna gain, *GT*, receiver antenna gain, *GR*, power conversion efficiency of the converter, *PCEH*, and the square of the wavelength, l, and is inversely proportional to the square of the communication distance, *r*, between the source and the device.

The communication energy consists of *ELS* (listening energy), *ERX* (receiver energy), and *ETX* (transmitter energy). The computation energy includes *EPR* (processing energy) and *ESN* (sensing energy). To capture the energy distribution among the aforementioned energy consumers, weighting coefficients a*LS* > a*TX* > a*RX* > a*PR* > a*SN* are assigned to them. The total  
average energy consumption *ED* = a*LS ELS* + a*TX ETX* + a*RX ERX* + a*PR* E*PR* + a*SN ESN*. *EB* is the total energy stored in the battery, and *EH* is the available harvested energy per active duty cycle. We assume constant energy consumptions for receiver, processor, and sensor. However, the energy consumption of the transmitter (*ETX*) is directly proportional to *rij* 2, where *rij* is the distance between the originating device *j* and the sink node *i* (in ring topology) or the sink node/sensor device (in multihop topology).The harvested energy *EH* is inversely proportional to *rij* 2 (here *j* is the sink node and *rij* = *rji*).

**IEEE Ref Paper:**

Wireless Energy Harvesting for the Internet of Things  
P. Kamalinejad C. Mahapatra ; Z. Sheng ; S. Mirabbasi ; V. C. M. Leung ; Y. L. Guan

*IEEE* COMMUNICATIONS MAGAZINE · JUNE 2015

**The code given below is for an example implementation of WEH whereby energy is harvested based on the received signal power. The Steps to be followed for Implementation in NetSim are:**

1. At the end of ChangeRadioState.c file inside Zigbee project folder add the following code

#define EH\_FRACTION 0.1

// EH\_FRACTION is the fraction of the received signal energy that can be captured and harvested // by the sensor.

int calculate\_eh(NETSIM\_ID dev1,NETSIM\_ID dev2)

{

pstruDevicePower[dev2-1]->dRemainingPower+= EH\_FRACTION\*GET\_RX\_POWER\_mw(dev1,dev2,pstruEventDetails->dEventTime);

}

1. Add the function call shown in red in 802\_15\_4.c file

case UPDATE\_MEDIUM:

{

double dtime=pstruEventDetails->dEventTime;

NETSIM\_ID nLink\_Id, nConnectionID, nConnectionPortID, nLoop;

NETSIM\_ID nTransmitterID;

nTransmitterID = pstruEventDetails->nDeviceId;

ZIGBEE\_CHANGERADIOSTATE(nTransmitterID, WSN\_PHY(nTransmitterID)->nRadioState, RX\_ON\_IDLE);

if(WSN\_PHY(nTransmitterID)->nRadioState != RX\_OFF)

WSN\_MAC(nTransmitterID)->nNodeStatus = IDLE;

nLink\_Id = fn\_NetSim\_Stack\_GetConnectedDevice(pstruEventDetails->nDeviceId,pstruEventDetails->nInterfaceId,&nConnectionID,&nConnectionPortID);

for(nLoop=1; nLoop<=NETWORK->ppstruNetSimLinks[nLink\_Id-1]->puniDevList.pstruMP2MP.nConnectedDeviceCount; nLoop++)

{

NETSIM\_ID ncon = NETWORK->ppstruNetSimLinks[nLink\_Id-1]->puniDevList.pstruMP2MP.anDevIds[nLoop-1];

if(ncon != pstruEventDetails->nDeviceId)

{

calculate\_eh(nTransmitterID,nLoop);

WSN\_PHY(ncon)->dTotalReceivedPower -= GET\_RX\_POWER\_mw(nTransmitterID,ncon,pstruEventDetails->dEventTime);

if(WSN\_PHY(ncon)->dTotalReceivedPower < WSN\_PHY(ncon)->dReceiverSensivity)

WSN\_PHY(ncon)->dTotalReceivedPower = 0;

}

}

This completes the code for energy harvesting.

Now we add code for getting network lifetime as an output metric from NetSim

1. Copy the provided code at the top in 802\_15\_4.h file

#include "string.h"

double NetSim\_Off\_Time[100];

string NetSim\_Node\_state[100];

double NetSim\_Residual\_Energy[100];

1. Copy the below code (in red colour) in 802\_15\_4.c file (inside fn\_NetSim\_Zigbee\_Metrics() function)

\_declspec(dllexport) int fn\_NetSim\_Zigbee\_Metrics(PMETRICSWRITER metricsWriter)

{

//CUSTOM METRICS

PMETRICSNODE table = init\_metrics\_node(MetricsNode\_Table,

"IOT Metrics for Energy Harvesting", NULL);

NETSIM\_ID nDeviceCount = NETWORK->nDeviceCount;

add\_table\_heading(table, "Node Name", true, 0);

add\_table\_heading(table, "Status", true, 0);

add\_table\_heading(table, "Time", true, 0);

add\_table\_heading(table, "Residual Energy", true, 0);

int i;

for (i = 1; i <= nDeviceCount; i++)

{

if(DEVICE\_MACLAYER(i, 1)->nMacProtocolId== MAC\_PROTOCOL\_IEEE802\_15\_4)

NetSim\_Residual\_Energy[i - 1] = pstruDevicePower[i - 1]->dRemainingPower;

else

NetSim\_Residual\_Energy[i - 1] = 0;

add\_table\_row\_formatted(false, table, "%s,%s,%.2lf,%lf,",

NETWORK->ppstruDeviceList[i - 1]->szDeviceName, NetSim\_Node\_state[i - 1],

NetSim\_Off\_Time[i - 1], NetSim\_Residual\_Energy[i - 1]);

}

PMETRICSNODE menu = init\_metrics\_node(MetricsNode\_Menu,

"IOT Metrics for Energy Harvesting", NULL);

add\_node\_to\_menu(menu, table);

write\_metrics\_node(metricsWriter, WriterPosition\_Current, NULL, menu);

delete\_metrics\_node(menu);

//CUSTOM METRICS

return fn\_NetSim\_Zigbee\_Metrics\_F(metricsWriter);

}

5. Copy the below code (in red colour) at the end of ChangeRadioState.c file (inside if(nStatus) loop)

if(nStatus)

{

WSN\_PHY(nDeviceId)->nOldState = nOldState;

WSN\_PHY(nDeviceId)->nRadioState = nNewState;

NetSim\_Node\_state[nDeviceId-1]= "ON";

return nStatus;

}

else

{

WSN\_PHY(nDeviceId)->nRadioState = RX\_OFF;

WSN\_MAC(nDeviceId)->nNodeStatus = OFF;

NetSim\_Off\_Time[nDeviceId-1] = ldEventTime;

NetSim\_Node\_state[nDeviceId-1]= "OFF";

return nStatus;

}

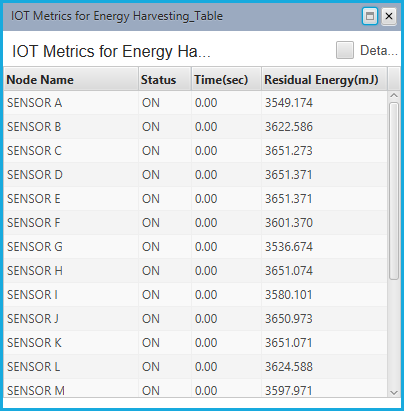
**COMPARATIVE ANALYSIS:**

**WITH MODIFIED EH\_FRACTION VALUE**

In the ChangeRadioState.c file modify the value of EH\_FRACTION to 0.1

#define EH\_FRACTION 0.1

Build Zigbee Project with the modified code and replace the obtained libZigbee.dll file in the bin path (“C:\Program Files (x86)\NetSim Standard\bin”) and run the Network Scenario (Turn off Energy Harvesting in the sensor nodes) for a Simulation Time of 300 Seconds.



**WITHOUT ENERGY HARVESTNG**

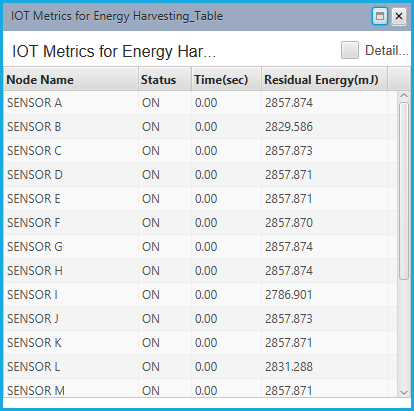
Comment the function call made in 802\_15\_4.c to run without RF energy harvesting

if(ncon != pstruEventDetails->nDeviceId)

{

//calculate\_eh(nTransmitterID,nLoop);

Build Zigbee Project with the modified code and replace the obtained libZigbee.dll file in the bin path (“C:\Program Files (x86)\NetSim Standard\bin”) and run the Network Scenario (Turn off Energy Harvesting in the sensor nodes) for a Simulation Time of 300 Seconds.



Now on comparing the custom IOT metrics we can observe that Energy Harvesting increases sensors’ working capability.