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

**Software Used:** NetSim Standard v11, Visual Studio 2015/2017

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. This implementation involves modifications to BatteryModel and ZigBee Projects present in <NetSim\_Install\_Directory>/src/Simulation directory.
2. As a best practice create a new directory for your project. From <NetSim\_Install\_Directory>/src/Simulation path copy the folders BatteryModel, ZigBee, Include, lib along with NetSim.sln file to your new folder.
3. Load the source codes in Visual studio by double clicking on NetSim.sln file.
4. Remove all unloaded projects leaving only ZigBee and BatteryModel in the Solution explorer to avoid confusion.
5. Expand BatteryModel Project and double click on the BatteryModel.h file to load the open the file and make the following changes(highlighted in red):

**\_declspec(dllexport) void battery\_animation();**

**\_declspec(dllexport) void battery\_metrics(PMETRICSWRITER metricsWriter);**

**\_declspec(dllexport) double battery\_get\_remaining\_energy(ptrBATTERY battery);**

**\_declspec(dllexport) int battery\_energy\_harvesting(ptrBATTERY battery, double eh\_energy);**

**\_declspec(dllexport) double battery\_get\_consumed\_energy(ptrBATTERY battery, int mode);**

1. Now double click on the BatteryModel.c file to load the open the file and make the following changes(highlighted in red):

**\_declspec(dllexport) double battery\_get\_remaining\_energy(ptrBATTERY battery)**

**{**

**return battery->remainingEnergy;**

**}**

**\_declspec(dllexport) int battery\_energy\_harvesting(ptrBATTERY battery, double eh\_energy)**

**{**

**battery->remainingEnergy += eh\_energy;**

**}**

1. Expand ZigBee project and double click on the ChangeRadioState.c file to open it. At the end of ChangeRadioState.c file add the following lines of 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)**

**{**

**double rx\_pwr = GET\_RX\_POWER\_mw(dev1, dev2, pstruEventDetails->dEventTime);**

**double eh\_energy= EH\_FRACTION \* rx\_pwr;**

**ptrBATTERY battery = WSN\_PHY(dev2)->battery;**

**if(battery)**

**battery\_energy\_harvesting(battery, eh\_energy);**

**}**

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;**

**}**

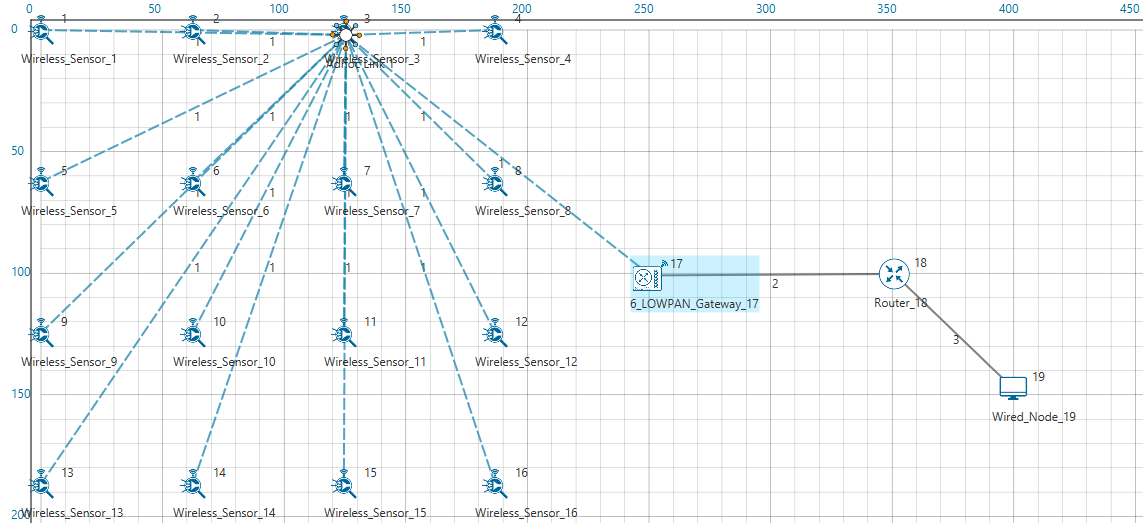
**}**

This completes the code for energy harvesting.

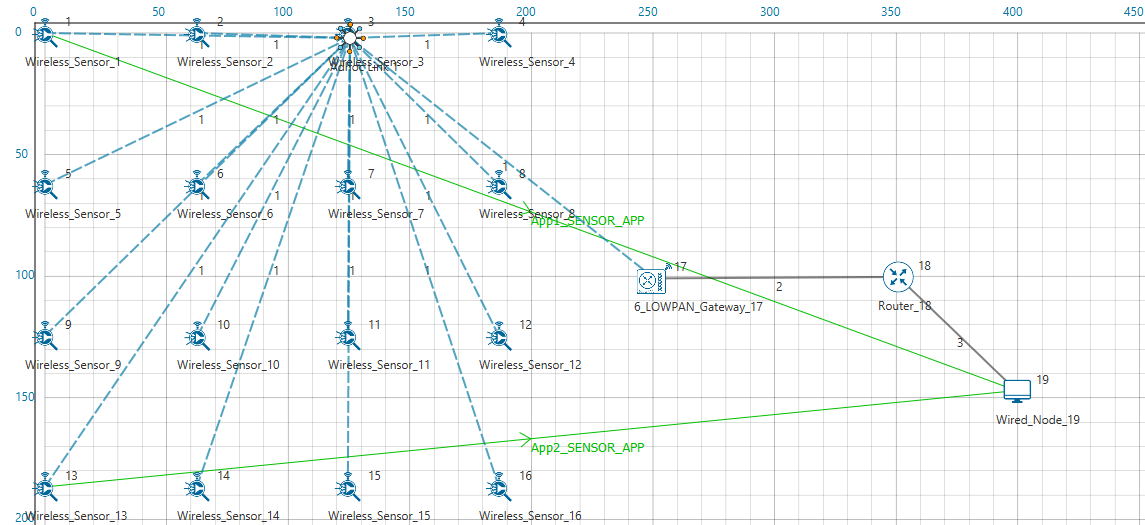
1. Right click on the BatteryModel Project in Solution Explorer and select Rebuild. Upon Successful build go to the DLL folder that has been created inside your project directory (created in Step 2). Copy the BatteryModel.lib file and place it in the lib folder which is present in your project folder.
2. Right click on the ZigBee project in Solution Explorer and select Rebuild. Upon successful build a new libZigBee.dll file will get created in the DLL folder present in your project directory (created in Step 2).

**COMPARATIVE ANALYSIS:**

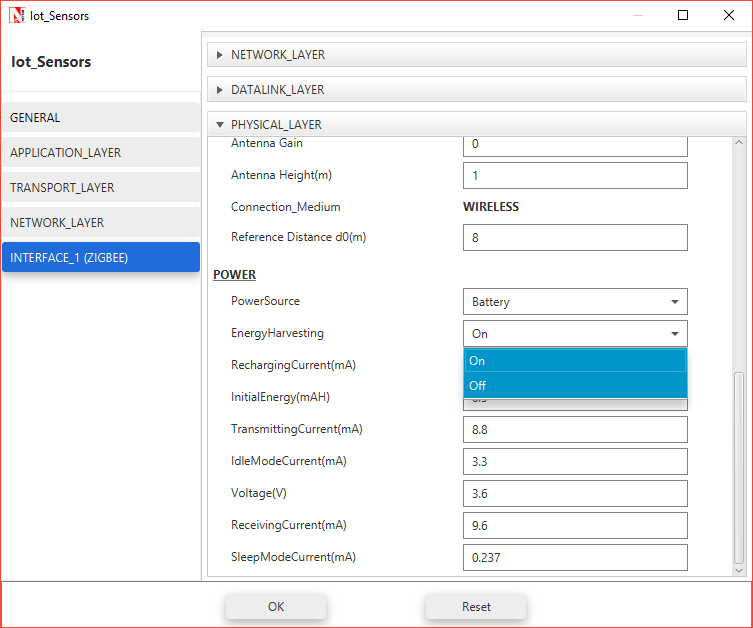
1. Create a network scenario in IoT with say 16 sensors, a 6LoWPAN Gateway, a Router and a Wired Node as shown below:



1. Configure traffic in the network by setting a few applications between some of the sensor nodes to the Wired Node using the Application Icon, as shown below:



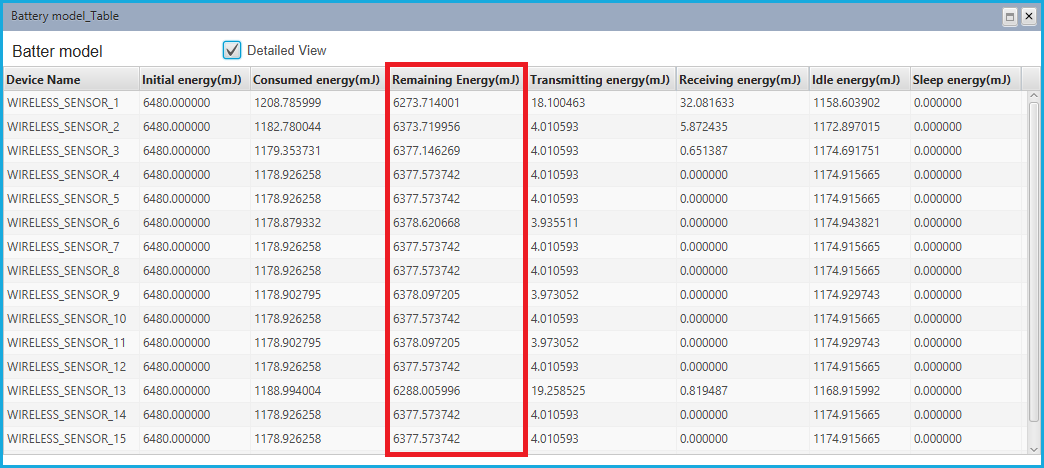
1. Disable Energy Harvesting in the Sensor nodes by setting the EnergyHarvesting parameter to OFF in the Interface(ZigBee) Properties of the sensor nodes as shown below:



1. Run Simulation for say 300 seconds and save the simulation results. In NetSim Simulation Results Window, the Battery model table provides detailed metrics related to energy consumed by each sensor node. The column Remaining energy can be used to compare simulations with and without energy harvesting code modification.

**WITH ENERGY HARVESTNG**

1. Copy the previously built BatteryModel.dll and libZigBee.dll files present in the DLL folder of your project directory (created in the beginning).
2. Go to <NetSim\_Install\_Directory>/bin and rename the already existing BatteryModel.dll and libZigBee.dll files to say BatteryModel\_original.dll and libZigBee\_original.dll and paste the copied dll files.
3. Now re-run the network simulation for 300 seconds and save the simulation results. You can use the table remaining energy column present in the Battery model table which is part of NetSim Simulation Results window to compare simulations with and without wireless energy harvesting.



Now on comparing the custom IOT metrics we can observe that Energy Harvesting increases sensors’ working capability. Simulations can be performed for different values of EH Fraction which may vary as per the efficiency of the Energy Harvesting unit.