Wireless Energy Harvesting for Internet of Things

Software: NetSim Standard v14.2, Visual Studio 2022

Project Download Link:

https://github.com/NetSim-TETCOS/Wireless-Energy-Harvesting-_-v14.2/archive/refs/heads/main.zip

Follow the instructions specified in the following link to download and set up the Project in NetSim:

https://support.tetcos.com/en/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects

1 Introduction to Energy Harvesting:

Among various methods like vibration, light, and thermal energy extraction, wireless energy harvesting (WEH) has proven to be one of the most promising solutions due to its simplicity, ease of implementation, and wide availability. This technology trend is gaining attention as it offers a fundamental approach to extending the lifespan of batteries. While harvesting from environmental sources depends on the presence of the corresponding energy source, RF (radio frequency) energy harvesting provides unique advantages, being wireless and easily accessible from transmitted energy sources like TV/radio broadcasters, mobile base stations, and handheld radios. It's cost-effective and allows for compact implementations.

Components of a WEH-Enabled Sensor Device:

A typical WEH-enabled sensor device comprises key components: an antenna, a transceiver, a wireless energy harvesting (WEH) unit, a power management unit (PMU), a sensor/processor unit, and optionally, an onboard battery.

Calculation of Harvested Power (PH):

The available Harvested power (P_H) is determined by the Friis equation. It is directly proportional to Transmitted power (P_T), Path loss (P_L), Transmitter antenna gain (G_T), Receiver antenna gain (G_R), the Power conversion efficiency of the converter (PCE_H), and the square of the wavelength (λ), and is inversely proportional to the square of the communication distance (r) between the source and the device.

Energy Components and Distribution:

The energy consumed by the device can be categorized into communication energy (listening, receiving, and transmitting) and computation energy (processing and sensing).

- Listening energy (E_{LS})
- Receiver energy (E_{RX})
- Transmitter energy (E_{TX})
- Processing energy (E_{PR})

- Sensing energy (E_{SN})

To capture the energy distribution among the aforementioned energy consumers, weighting coefficients, $\alpha_{LS} > \alpha_{TX} > \alpha_{RX} > \alpha_{PR} > \alpha_{SN}$ are assigned to them. The total average energy consumption $E_D = \alpha_{LS}E_{LS} + \alpha_{TX}E_{TX} + \alpha_{RX}E_{RX} + \alpha_{PR}E_{PR} + \alpha_{SN}E_{SN}$ is then calculated as the sum of the product of these coefficients and their respective energy components.

Battery Storage and Harvested Energy:

The total energy stored in the device's battery is denoted as E_B. On the other hand, the available harvested energy per active-duty cycle is represented by E_H.

Topology Considerations:

We assume constant energy consumption for the receiver, processor, and sensor. However, the energy consumption of the transmitter (E_{TX}) is directly proportional to the square of the distance (r^2_{ij}) ,where r_{ij} is the distance between the originating device (j) and the sink node (i), particularly in a ring topology or multihop topology. The harvested energy (E_H) is inversely proportional to r^2_{ij} (here j is the sink node and $r_{ij} = r_{ji}$).

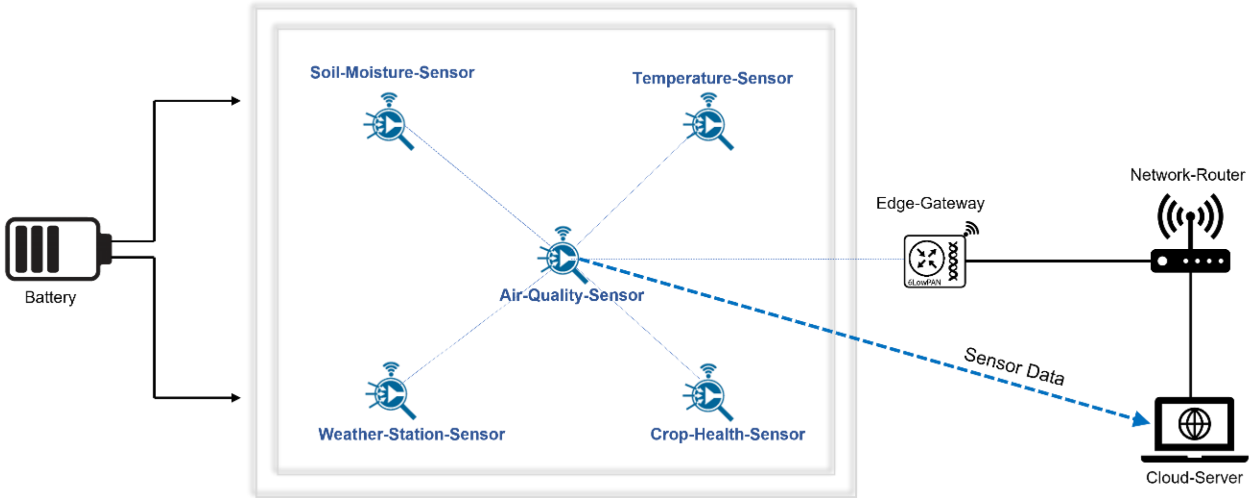


Figure 1: Scenario for Smart agriculture system Without Energy harvesting

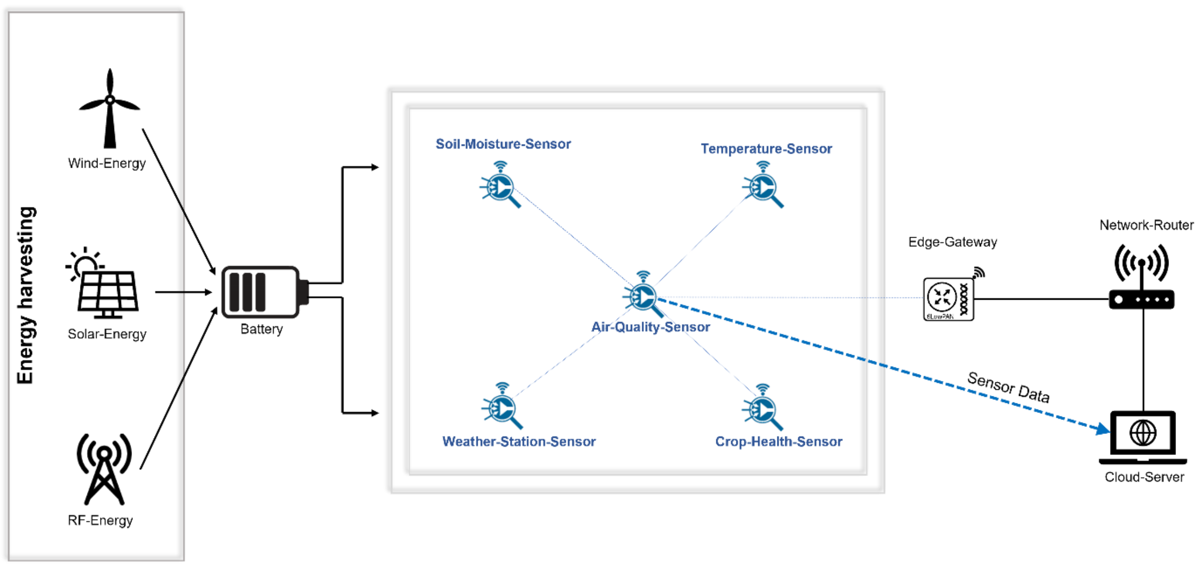


Figure 2: Scenario for Smart agriculture system With Energy harvesting

2 COMPARATIVE ANALYSIS

1. The **Energy_Harvesting_IOT_Workspace** includes sample network configuration files, namely Without-energy-harvesting and With-energy-harvesting, which are pre-saved.

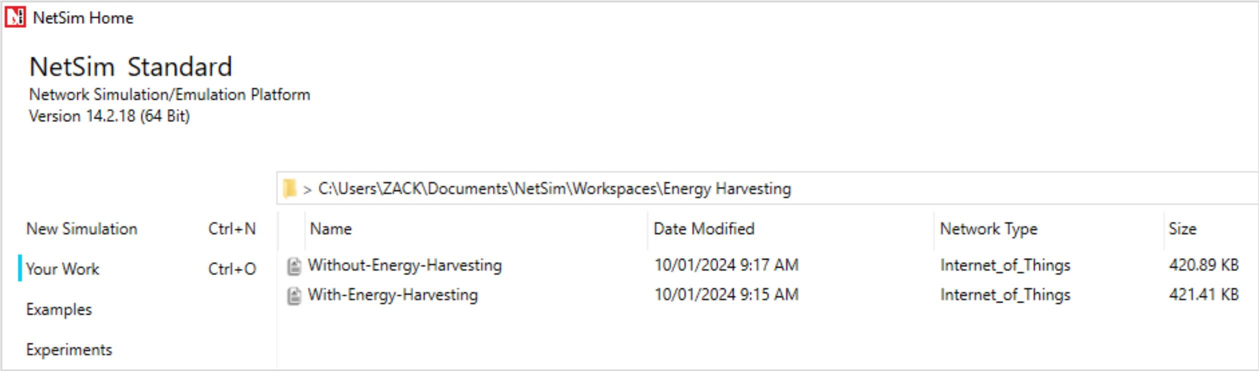


Figure 3: Sample configuration files

2. We will now open the **"Without-Energy-Harvesting"** sample.
3. The network scenario consists of 16 sensors, a LoWPAN Gateway, a Router and a Wired Node as shown below.

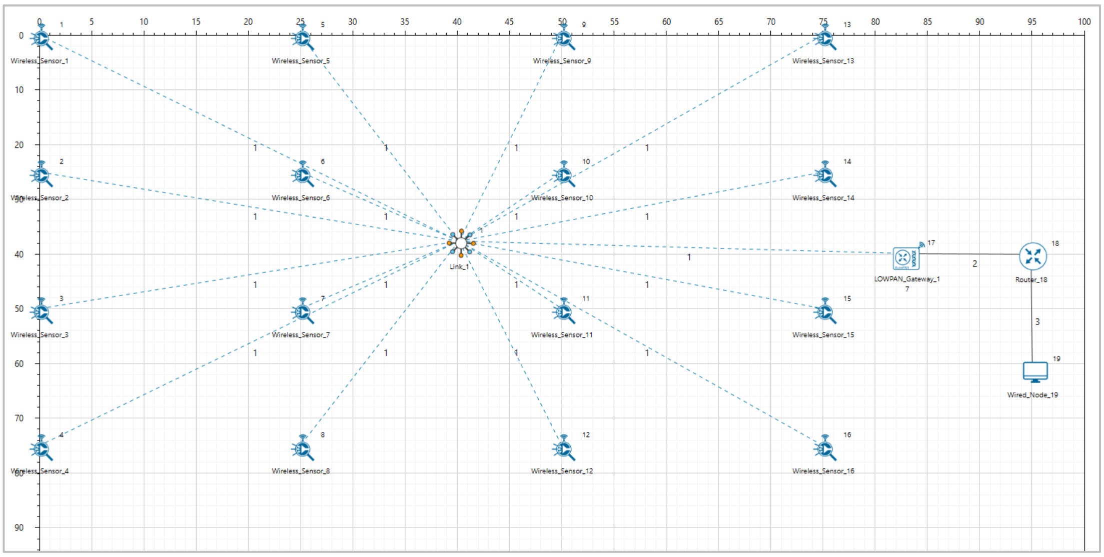


Figure 4: Energy Harvesting Network Topology

4. Here, the Energy Harvesting parameter has been set to OFF in all Sensor nodes. To enable or disable the energy harvesting setting, users can navigate to Interface(ZigBee)->Physical layer -> IEE802.15.4 ->Power ->Set Power source to Battery of the sensor nodes, as shown below

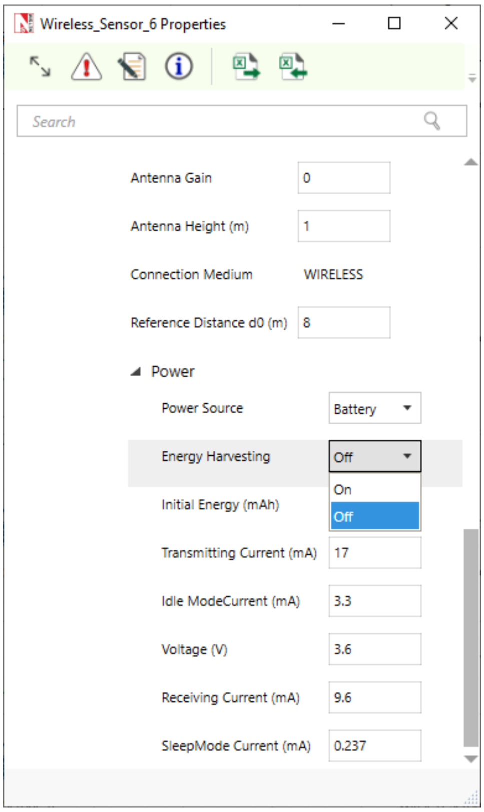
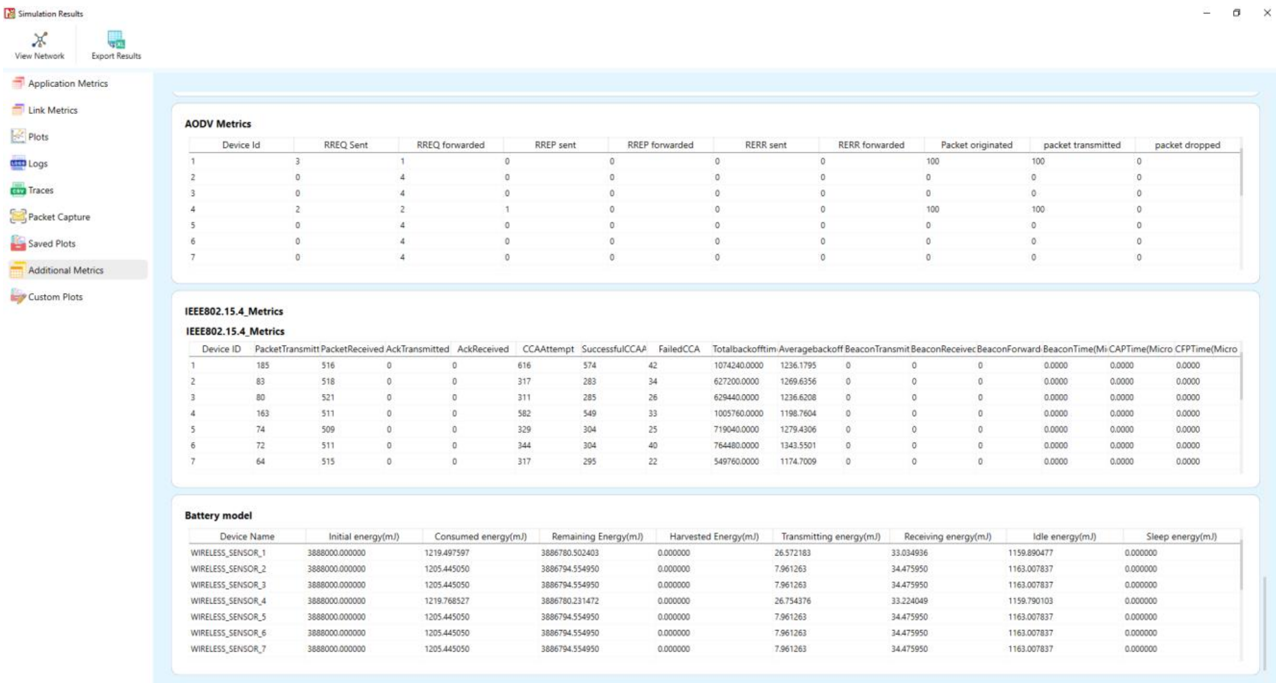


Figure 4: Energy Harvesting parameter

5. Run Simulation for 100 seconds and save the simulation results.

3 Results and Discussion

Once the simulation is complete, the NetSim result dashboard will open. From there, navigate to the additional metrics section and select the Battery model under IEEE802.15.4_Metrics. This will display the Battery model table.



AODV Metrics

Device Id	RREQ Sent	RREQ forwarded	RREP sent	RREP forwarded	RERR sent	RERR forwarded	Packet originated	packet transmitted	packet dropped
1	3	1	0	0	0	0	100	100	0
2	0	4	0	0	0	0	0	0	0
3	0	4	0	0	0	0	0	0	0
4	2	2	1	0	0	0	100	100	0
5	0	4	0	0	0	0	0	0	0
6	0	4	0	0	0	0	0	0	0
7	0	4	0	0	0	0	0	0	0

IEEE802.15.4_Metrics

IEEE802.15.4_Metrics

Device ID	PacketTransmitt	PacketReceived	AckTransmitted	AckReceived	CCAAttempt	SuccessfulCCAA	FailedCCA	Totalbackofftim	Averagebackoff	BeaconTransmit	BeaconReceivec	BeaconForward	BeaconTime(Mi	CAPTime(Micro	CFPTime(Micro
1	185	516	0	0	616	574	42	1074240.0000	1236.1795	0	0	0	0.0000	0.0000	0.0000
2	83	518	0	0	317	283	34	627200.0000	1269.6356	0	0	0	0.0000	0.0000	0.0000
3	80	521	0	0	311	285	26	629440.0000	1236.6208	0	0	0	0.0000	0.0000	0.0000
4	163	511	0	0	582	549	33	1005760.0000	1198.7604	0	0	0	0.0000	0.0000	0.0000
5	74	509	0	0	329	304	25	719040.0000	1279.4306	0	0	0	0.0000	0.0000	0.0000
6	72	511	0	0	344	304	40	764480.0000	1343.5501	0	0	0	0.0000	0.0000	0.0000
7	64	515	0	0	317	295	22	549760.0000	1174.7009	0	0	0	0.0000	0.0000	0.0000

Battery model

Device Name	Initial energy(mJ)	Consumed energy(mJ)	Remaining Energy(mJ)	Harvested Energy(mJ)	Transmitting energy(mJ)	Receiving energy(mJ)	Idle energy(mJ)	Sleep energy(mJ)
WIRELESS_SENSOR_1	3888000.000000	1219.497597	3886780.502403	0.000000	26.572183	33.034936	1159.890477	0.000000
WIRELESS_SENSOR_2	3888000.000000	1205.445050	3886794.554950	0.000000	7.961263	34.475950	1163.007837	0.000000
WIRELESS_SENSOR_3	3888000.000000	1205.445050	3886794.554950	0.000000	7.961263	34.475950	1163.007837	0.000000
WIRELESS_SENSOR_4	3888000.000000	1219.768527	3886780.231472	0.000000	26.754376	33.224049	1159.790103	0.000000
WIRELESS_SENSOR_5	3888000.000000	1205.445050	3886794.554950	0.000000	7.961263	34.475950	1163.007837	0.000000
WIRELESS_SENSOR_6	3888000.000000	1205.445050	3886794.554950	0.000000	7.961263	34.475950	1163.007837	0.000000
WIRELESS_SENSOR_7	3888000.000000	1205.445050	3886794.554950	0.000000	7.961263	34.475950	1163.007837	0.000000

Figure 5: Battery model table from Netsim result dashboard

WITHOUT ENERGY HARVESTING:

In the Battery model table, you can observe the results for scenario without energy harvesting. This table provides detailed insights into the energy consumption of each sensor node

Battery model

Device Name	Initial energy(mJ)	Consumed energy(mJ)	Remaining Energy(mJ)	Harvested Energy(mJ)	Transmitting energy(mJ)	Receiving energy(mJ)	Idle energy(mJ)	Sleep energy(mJ)
WIRELESS_SENSOR_1	3888000.000000	1219.497597	3886780.502403	0.000000	26.572183	33.034936	1159.890477	0.000000
WIRELESS_SENSOR_2	3888000.000000	1205.445050	3886794.554950	0.000000	7.961263	34.475950	1163.007837	0.000000
WIRELESS_SENSOR_3	3888000.000000	1205.445050	3886794.554950	0.000000	7.961263	34.475950	1163.007837	0.000000
WIRELESS_SENSOR_4	3888000.000000	1219.768527	3886780.231472	0.000000	26.754376	33.224049	1159.790103	0.000000
WIRELESS_SENSOR_5	3888000.000000	1205.445050	3886794.554950	0.000000	7.961263	34.475950	1163.007837	0.000000
WIRELESS_SENSOR_6	3888000.000000	1205.445050	3886794.554950	0.000000	7.961263	34.475950	1163.007837	0.000000
WIRELESS_SENSOR_7	3888000.000000	1205.445050	3886794.554950	0.000000	7.961263	34.475950	1163.007837	0.000000

Figure 6: Battery model table without energy harvesting

WITH ENERGY HARVESTING:

Now, open and run the 'With-Energy-Harvesting' sample, where Energy Harvesting is enabled for all sensor nodes. Upon comparing the remaining energy levels with the "without-energy-harvesting" Scenario, you will observe increases in the working capability of sensors

Battery model

Device Name	Initial energy(mJ)	Consumed energy(mJ)	Remaining Energy(mJ)	Harvested Energy(mJ)	Transmitting energy(mJ)	Receiving energy(mJ)	Idle energy(mJ)	Sleep energy(mJ)
WIRELESS_SENSOR_1	3888000.000000	1219.497597	3886923.096872	142.594469	26.572183	33.034936	1159.890477	0.000000
WIRELESS_SENSOR_2	3888000.000000	1205.445050	3886937.149418	142.594469	7.961263	34.475950	1163.007837	0.000000
WIRELESS_SENSOR_3	3888000.000000	1205.445050	3886937.149418	142.594469	7.961263	34.475950	1163.007837	0.000000
WIRELESS_SENSOR_4	3888000.000000	1219.768527	3886922.825941	142.594469	26.754376	33.224049	1159.790103	0.000000
WIRELESS_SENSOR_5	3888000.000000	1205.445050	3886937.149418	142.594469	7.961263	34.475950	1163.007837	0.000000
WIRELESS_SENSOR_6	3888000.000000	1205.445050	3886937.149418	142.594469	7.961263	34.475950	1163.007837	0.000000
WIRELESS_SENSOR_7	3888000.000000	1205.445050	3886937.149418	142.594469	7.961263	34.475950	1163.007837	0.000000

Figure 7 : Battery Model table with energy harvesting

Simulations can be performed for different values of E_H Fraction which may vary as per the efficiency of the Energy Harvesting unit.

Note: You can observe slight variation in the remaining energy with and without energy harvesting as the simulation time is in seconds.

Appendix: NetSim source code modifications

Changes to Battery Model.h within Battery Model project

/* We implemented the Batter Model*/

```
#ifndef _NETSIM_BATTERY_MODEL_H_
#define _NETSIM_BATTERY_MODEL_H_
#ifdef  __cplusplus
extern "C" {
#endif

#ifndef _BATTERY_MODEL_CODE_
#pragma comment(lib,"BatteryModel.lib")
	typedef void* ptrBATTERY;
#endif

	_declspec(dllexport) ptrBATTERY battery_find(NETSIM_ID d,

NETSIM_ID in);
	_declspec(dllexport) void battery_add_new_mode(ptrBATTERY battery, int mode, double current,
char* heading);
	_declspec(dllexport) ptrBATTERY battery_init_new(NETSIM_ID deviceId, NETSIM_ID interfaceId,
double initialEnergy, double voltage, double dRechargingCurrent);

	_declspec(dllexport) bool battery_set_mode(ptrBATTERY battery, int mode, double time);
	_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);

#ifdef  __cplusplus
}
#endif
#endif //_NETSIM_BATTERY_MODEL_H_
```

Changes to double battery_get_remaining_energy (), Battery Model.c within Battery Model project

```
_declspec(dllexport) double battery_get_remaining_energy(ptrBATTERY battery)
{
	return battery->remainingEnergy;
```

```
}
_declspec(dllexport) int battery_energy_harvesting(ptrBATTERY battery, double eh_energy)
{
	double eh_energy_mJ = eh_energy * ((pstruEventDetails->dEventTime - battery->modeChangedTime) / 1000000);
	battery->remainingEnergy += eh_energy_mJ;
}
```

Changes code to ChangeRadioState.c, within Zigbee project at the end of the file

```
#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);
}
```

Changes code to int fn_NetSim_Zigbee_Run(), 802_15_4.c file**, within Zigbee project**

```
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;
}
}
```

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

