**Wireless Energy Harvesting for Internet of Things**

**Software:** NetSim Standard v14.3, 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>

# 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 (PH) is determined by the Friis equation. It is directly proportional to Transmitted power (PT), Path loss (PL), Transmitter antenna gain (GT), Receiver antenna gain (GR), the Power conversion efficiency of the converter (PCEH), 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 (ELS)
* Receiver energy (ERX)
* Transmitter energy (ETX)
* Processing energy (EPR)
* Sensing energy (ESN)

To capture the energy distribution among the aforementioned energy consumers, weighting coefficients, αLS> αTX> αRX > αPR> αSN are assigned to them. The total average energy consumption ED= αLSELS+ αTXETX+ αRXERX+ αPREPR+ αSNESN 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 EB. On the other hand, the available harvested energy per active-duty cycle is represented by EH.

**Topology Considerations:**

We assume constant energy consumption for the receiver, processor, and sensor. However, the energy consumption of the transmitter (ETX) is directly proportional to the square of the distance (r2ij) ,where rij is the distance between the originating device (j) and the sink node (i), particularly in a ring topology or multihop topology. The harvested energy (EH) is inversely proportional to r2ij (here j is the sink node and rij = rji).

A computer screen shot of a network

Description automatically generated

**Figure 1:** Scenario for Smart agriculture system Without Energy harvesting

A screen shot of a computer screen

Description automatically generated

**Figure 2:** Scenario for Smart agriculture system With Energy harvesting

# COMPARATIVE ANALYSIS

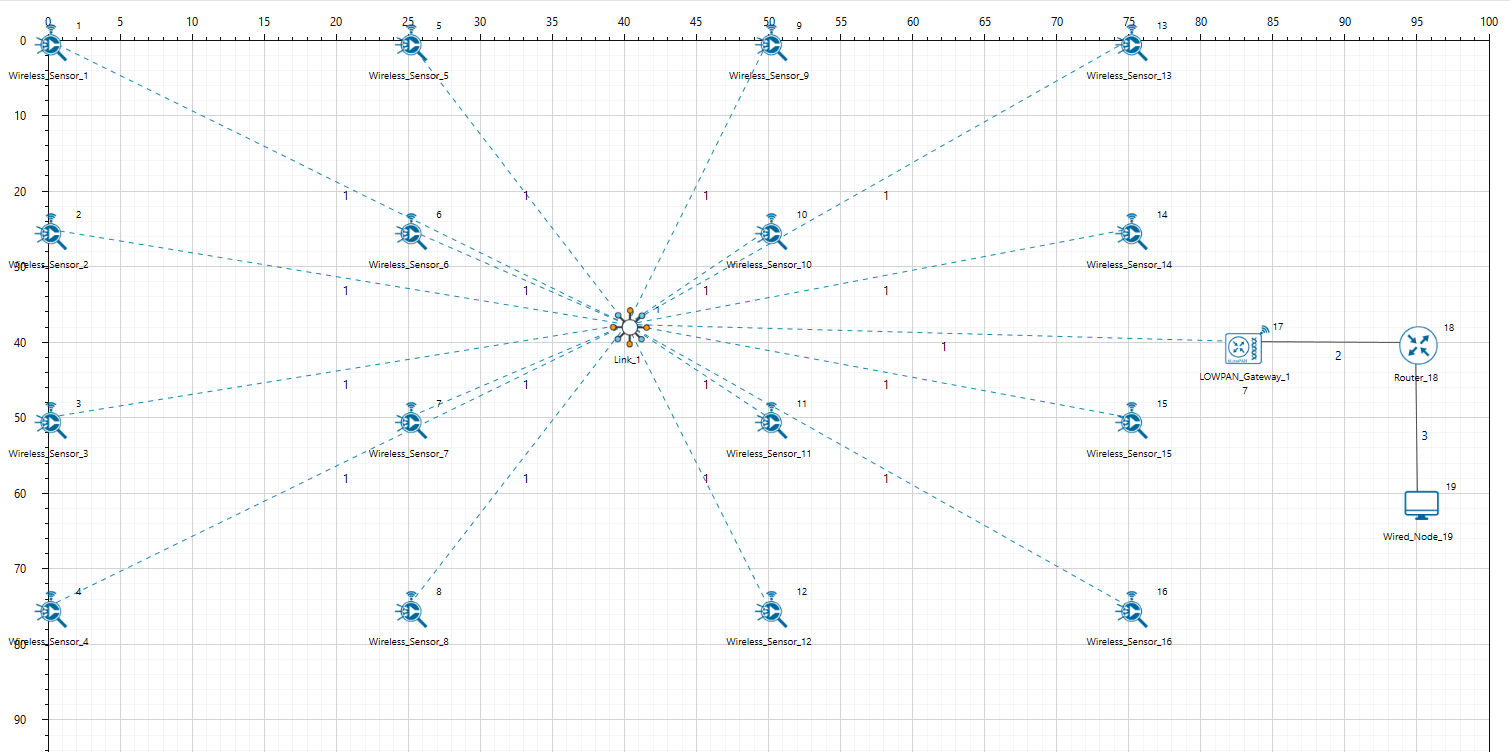
1. The **Energy\_Harvesting\_IOT\_Workspace** includes sample network configuration files, namely Without-energy-harvesting and With-energy-harvesting, which are pre-saved.

A close-up of a computer screen

Description automatically generated

**Figure 3:** Sample configuration files

1. We will now open the **“Without-Energy-Harvesting”** sample.
2. 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

1. 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

A screenshot of a computer

Description automatically generated

**Figure 5:**Energy Harvesting parameter

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

# 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.

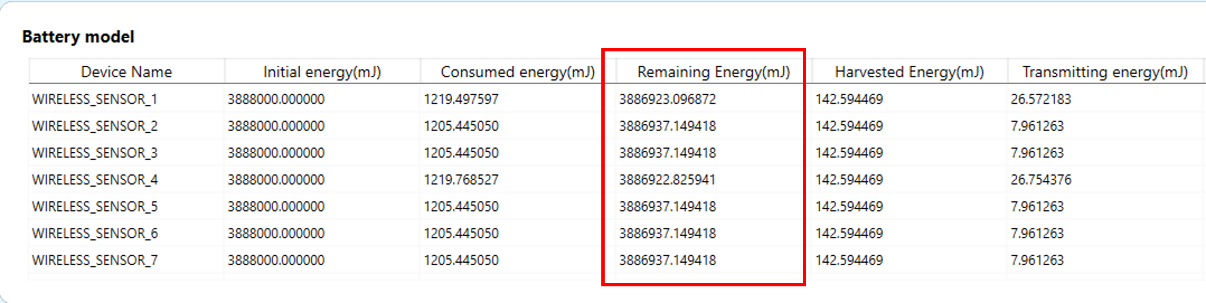
A screenshot of a computer

Description automatically generated

**Figure 6:** Battery model table from Netsim result dashboard

**WITHOUT ENERGY HARVESTING:**

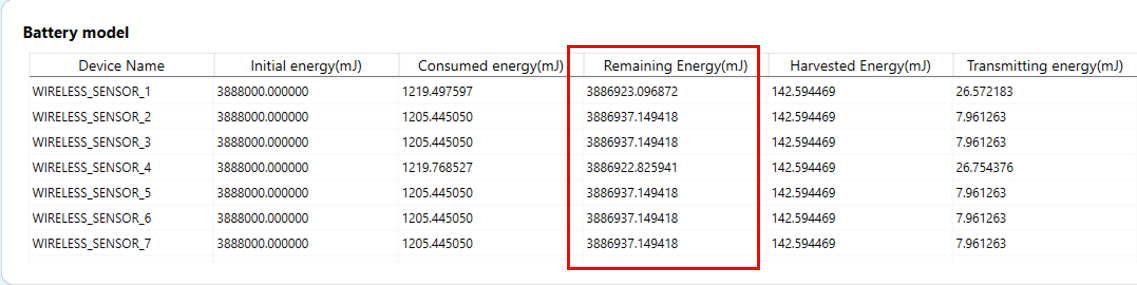
In the Battery model table, you can observe the results for scenario without energy harvesting. This table provides detailed insights into the energy consumption of each sensor node



**Figure 7:** 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



**Figure 8** : Battery Model table with energy harvesting

Simulations can be performed for different values of EH 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 Battery 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