Wireless Energy Harvesting for the Internet of Things

Software: NetSim Standard v13.0 (32/64-bit), Visual Studio 2019

Project Download Link:
https://github.com/NetSim-TETCOS/Energy_Harvesting_in_IOT_v13_0/archive/refs/heads/main.zip

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

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

Introduction

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 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 aLS > 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

Steps to simulate

1. Open the Source codes in Visual Studio by going to Your work-> Workspace Options and Clicking on Open code button in NetSim Home Screen window.
2. Right click on the solution in the solution explorer and select Rebuild.

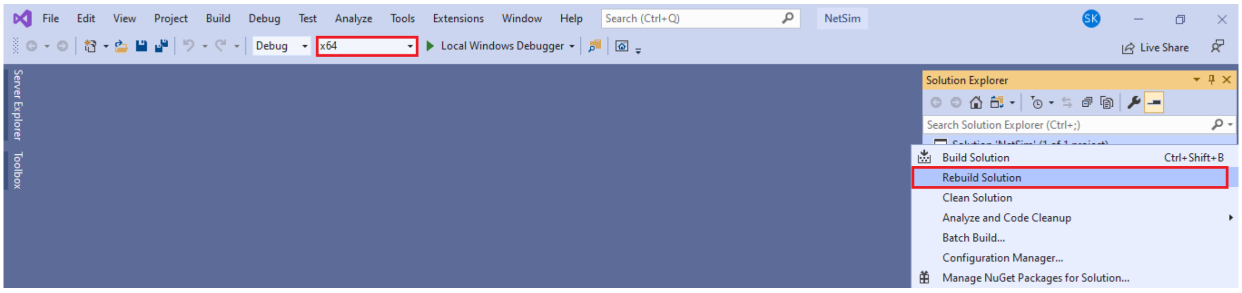


Figure 1: Screen shot of NetSim project source code in Visual Studio

3. Upon successful build modified libZigBee.dll and BatteryModel.dll file gets automatically updated in the directory containing NetSim binaries.

COMPARATIVE ANALYSIS:

1. The **WSN_Energy_Harvesting_Workspace** comes with a sample network configuration that are already saved. To open this example, go to Your work in the Home screen of NetSim and click on the **WSN_Energy_Harvesting_Example** from the list of experiments.
2. Create a network scenario in IoT with say 16 sensors, a 6LoWPAN Gateway, a Router and a Wired Node as shown below.

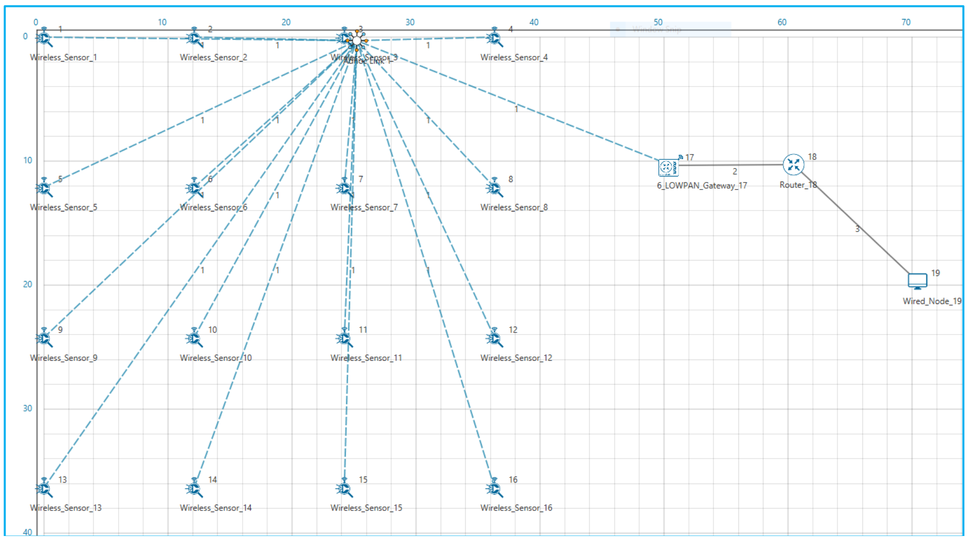


Figure 2: Energy Harvesting Network Topology

3. 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.

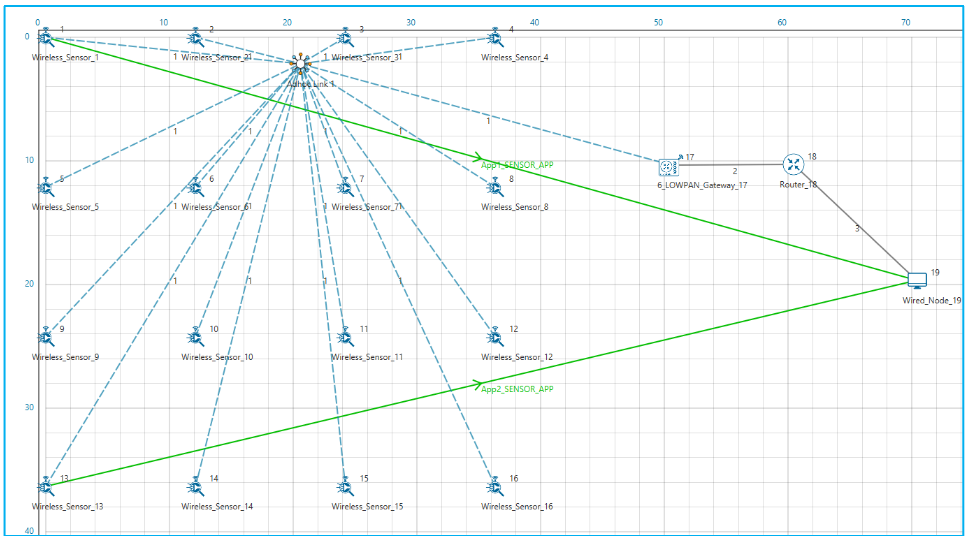
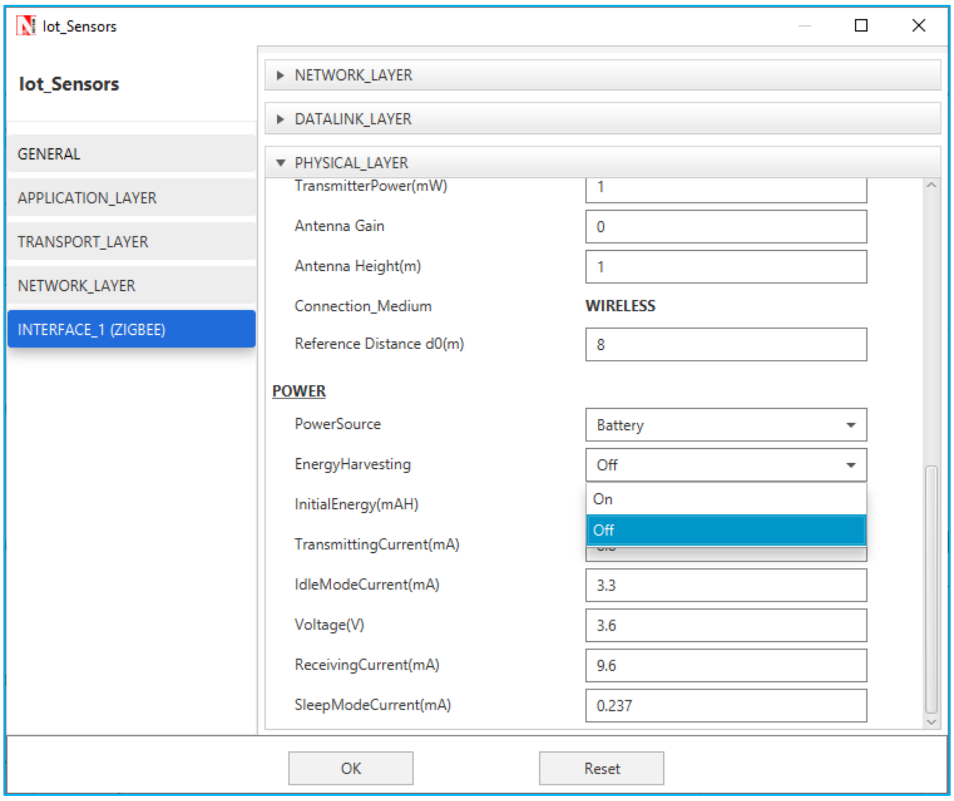


Figure 3: Configured traffic wireless sensor to Cloud server

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



TransmitterPower(mW)	1
Antenna Gain	0
Antenna Height(m)	1
Connection_Medium	WIRELESS
Reference Distance d0(m)	8
POWER	
PowerSource	Battery
EnergyHarvesting	Off (On / [✓ Off])
InitialEnergy(mAH)	
TransmittingCurrent(mA)	
IdleModeCurrent(mA)	3.3
Voltage(V)	3.6
ReceivingCurrent(mA)	9.6
SleepModeCurrent(mA)	0.237

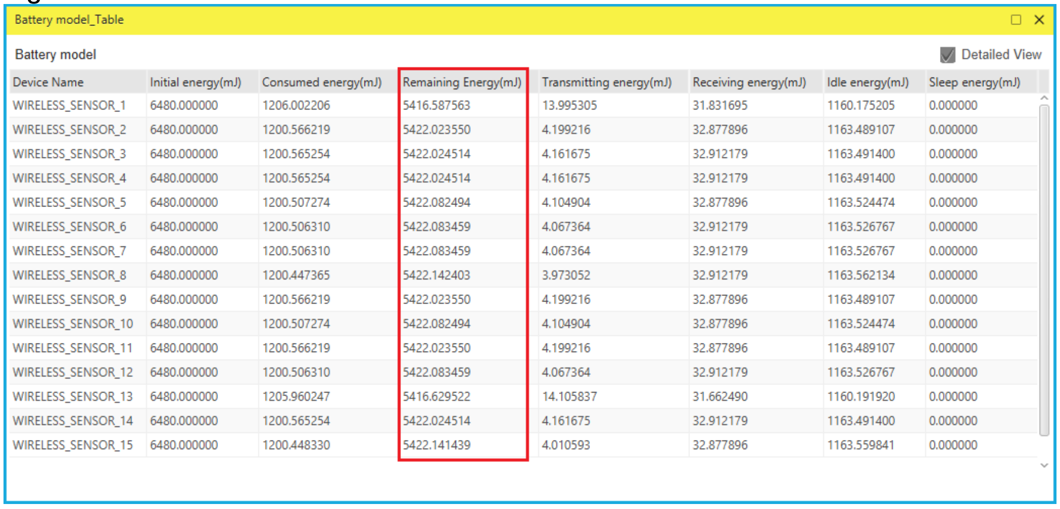
Figure 4: Disable Energy Harvesting in all Sensor nodes

5. Run Simulation for say 100 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. Now re-run the network simulation for 100 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.



Battery model_Table

Battery model

Detailed View

Device Name	Initial energy(mJ)	Consumed energy(mJ)	Remaining Energy(mJ)	Transmitting energy(mJ)	Receiving energy(mJ)	Idle energy(mJ)	Sleep energy(mJ)
WIRELESS_SENSOR_1	6480.000000	1206.002206	5416.587563	13.995305	31.831695	1160.175205	0.000000
WIRELESS_SENSOR_2	6480.000000	1200.566219	5422.023550	4.199216	32.877896	1163.489107	0.000000
WIRELESS_SENSOR_3	6480.000000	1200.565254	5422.024514	4.161675	32.912179	1163.491400	0.000000
WIRELESS_SENSOR_4	6480.000000	1200.565254	5422.024514	4.161675	32.912179	1163.491400	0.000000
WIRELESS_SENSOR_5	6480.000000	1200.507274	5422.082494	4.104904	32.877896	1163.524474	0.000000
WIRELESS_SENSOR_6	6480.000000	1200.506310	5422.083459	4.067364	32.912179	1163.526767	0.000000
WIRELESS_SENSOR_7	6480.000000	1200.506310	5422.083459	4.067364	32.912179	1163.526767	0.000000
WIRELESS_SENSOR_8	6480.000000	1200.447365	5422.142403	3.973052	32.912179	1163.562134	0.000000
WIRELESS_SENSOR_9	6480.000000	1200.566219	5422.023550	4.199216	32.877896	1163.489107	0.000000
WIRELESS_SENSOR_10	6480.000000	1200.507274	5422.082494	4.104904	32.877896	1163.524474	0.000000
WIRELESS_SENSOR_11	6480.000000	1200.566219	5422.023550	4.199216	32.877896	1163.489107	0.000000
WIRELESS_SENSOR_12	6480.000000	1200.506310	5422.083459	4.067364	32.912179	1163.526767	0.000000
WIRELESS_SENSOR_13	6480.000000	1205.960247	5416.629522	14.105837	31.662490	1160.191920	0.000000
WIRELESS_SENSOR_14	6480.000000	1200.565254	5422.024514	4.161675	32.912179	1163.491400	0.000000
WIRELESS_SENSOR_15	6480.000000	1200.448330	5422.141439	4.010593	32.877896	1163.559841	0.000000

Figure 5: Battery model table from result dashboard

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.

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;
}
}
```

This completes the code modifications for energy harvesting.