Wireless Energy Harvesting for the Internet of Things

Software Used: NetSim Standard v12.1 (32/64-bit), Visual Studio 2019

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.

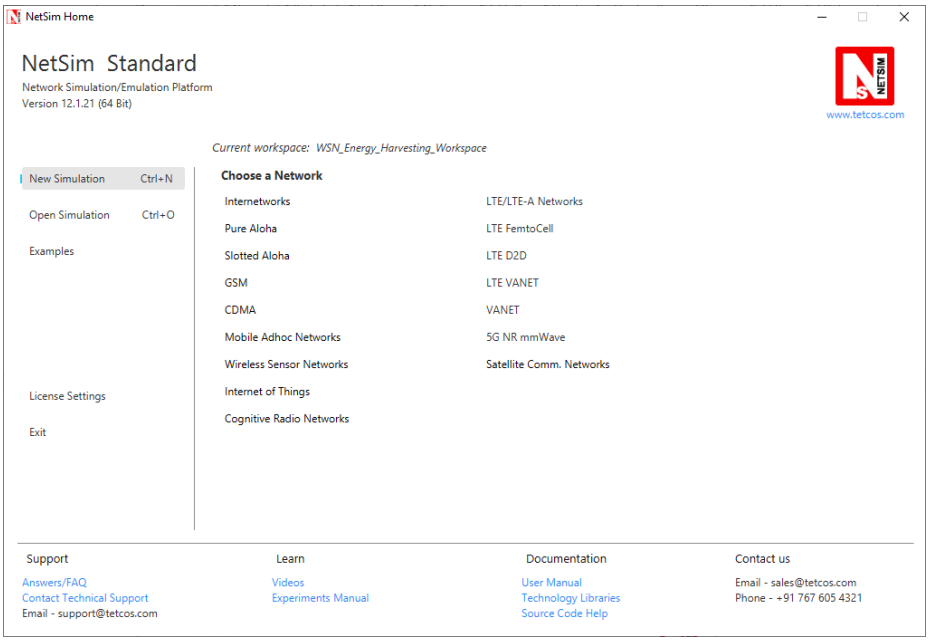
The communication energy consists of *ELS* (listening energy), *ERX* (receiver energy), and *ETX* (transmitter energy). The computation energy includes *EPR* (processing energy) and *ESN* (sensing energy). To capture the energy distribution among the aforementioned energy consumers, weighting coefficients 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 $r_{ij}^{\ 2}$, where r_{ij} 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 $r_{ij}^{\ 2}$ (here *j* is the sink node and $r_{ij} = r_{ji}$).

IEEE Ref Paper:

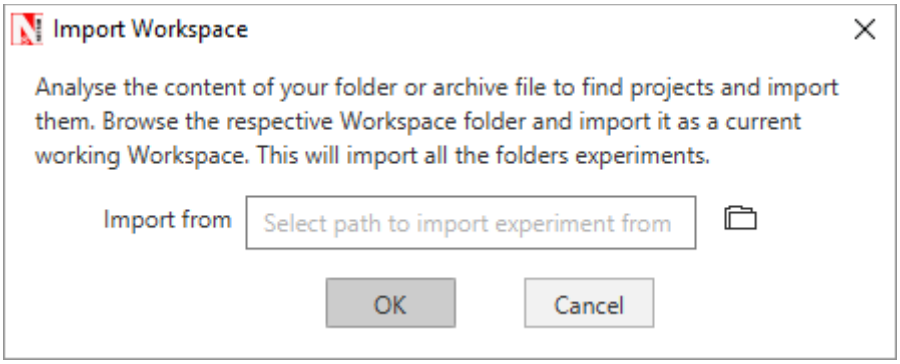
Wireless Energy Harvesting for the Internet of Things
P. Kamalinejad C. Mahapatra ; Z. Sheng ; S. Mirabbasi ; V. C. M. Leung ; Y. L. Guan
IEEE COMMUNICATIONS MAGAZINE · JUNE 2015

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:

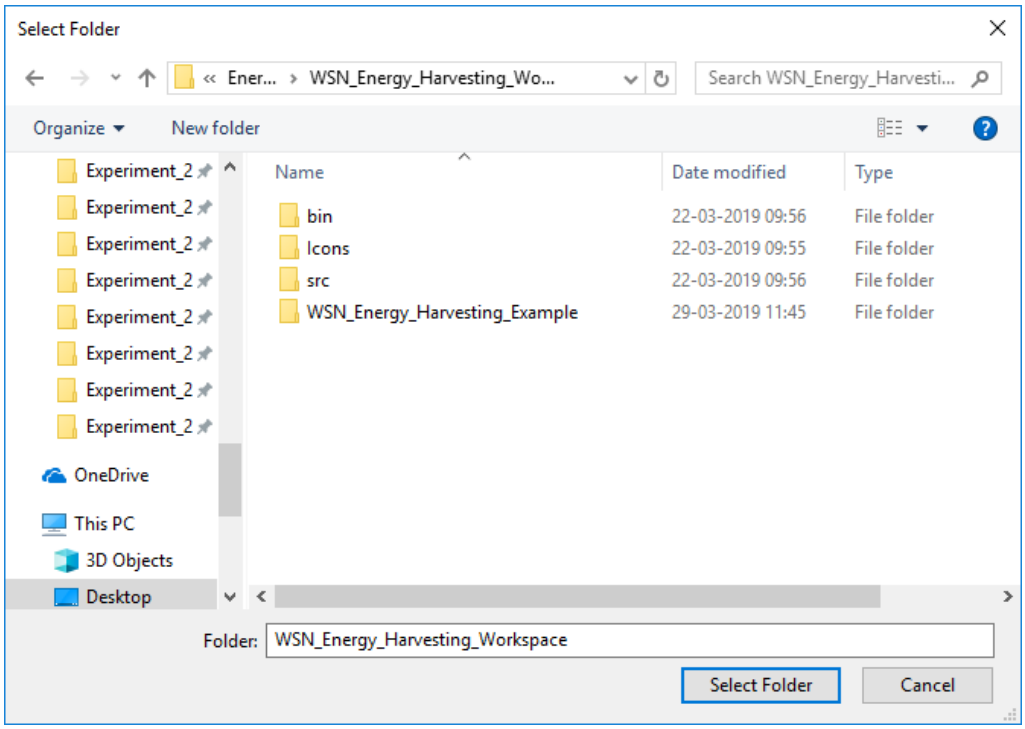
1. Import WSN_Energy_Harvesting_Workspace by going to Open Simulation->Workspace Options->More Options in NetSim Home window. Then select Import as shown below:



2. It displays a window where users need to give the path of the workspace folder and click on OK as shown below:

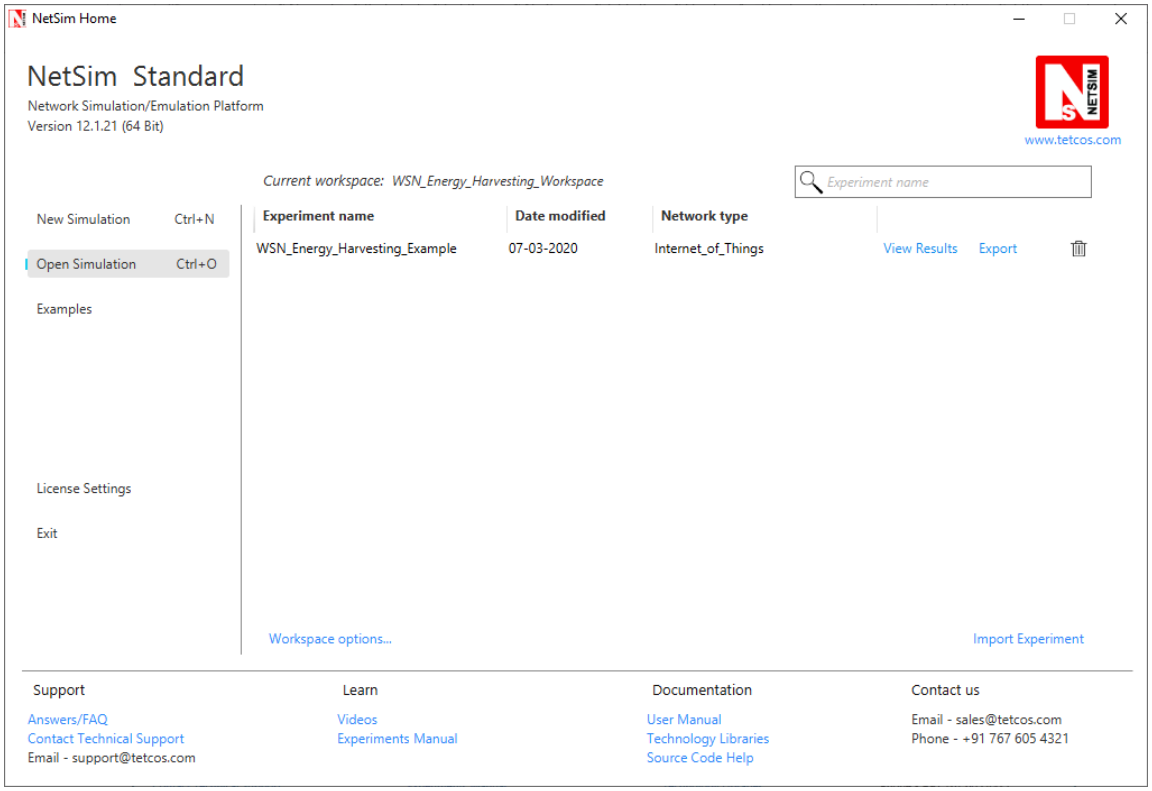


3. Browse to the WSN_Energy_Harvesting_Workspace folder and click on select folder as shown below:

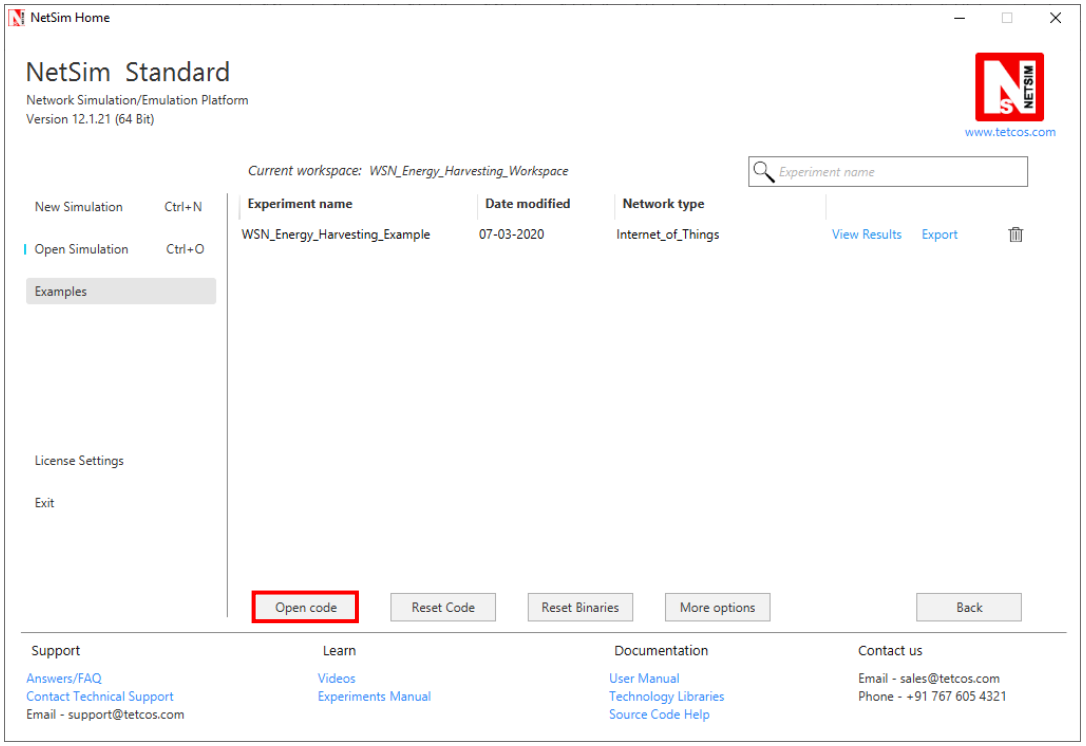


4. After this click on OK button in the Import Workspace window.

5. The Imported workspace will be set as the current workspace automatically. To see the imported workspace, click on Open Simulation->Workspace Options->More Options as shown below:



6. Open the Source codes in Visual Studio by going to Open Simulation-> Workspace Options and Clicking on Open code button as shown below:



7. Expand BatteryModel Project and double click on the BatteryModel.h file to open it. The following changes(highlighted in red) were done to the code:

```
_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);
```

8. Now double click on the BatteryModel.c file to open the file. The following changes(highlighted in red) were done to the code:

```
_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;

}
```

9. Expand ZigBee project and double click on the ChangeRadioState.c file to open it. At the end of ChangeRadioState.c file the following lines of code are added:

```
#define EH_FRACTION 0.1
// EH_FRACTION is the fraction of the received signal energy that can be
// captured and harvested by the sensor.
```

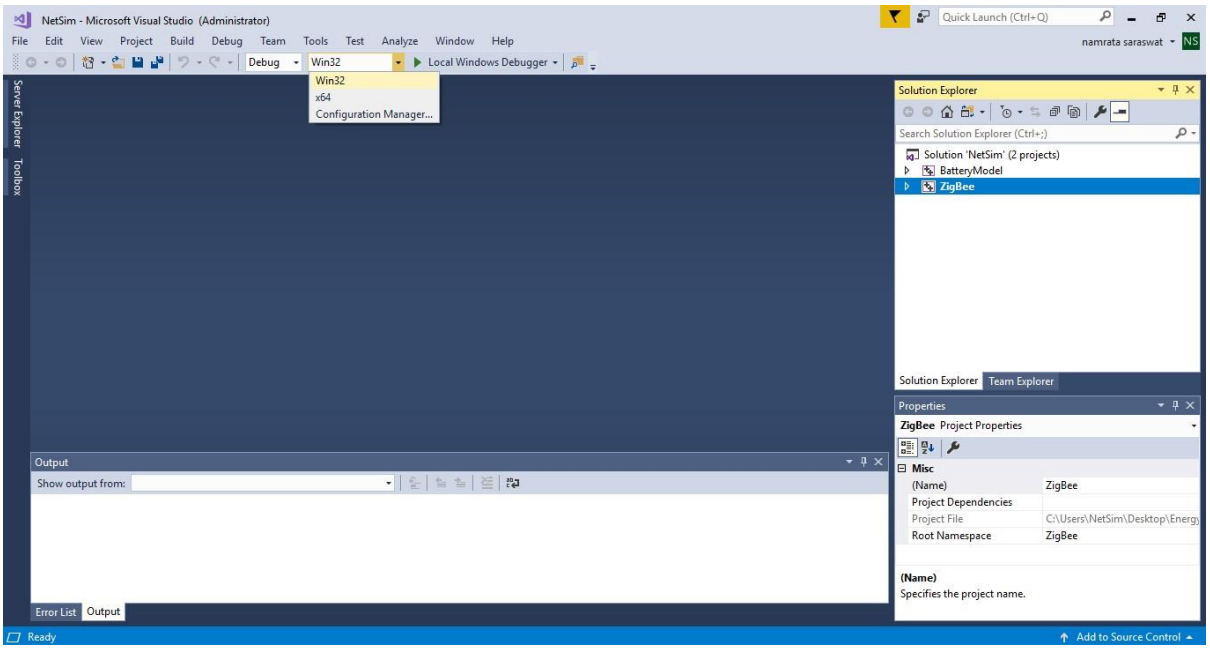
```
int calculate_eh(NETSIM_ID dev1, NETSIM_ID dev2)
{
        double rx_pwr = GET_RX_POWER_mw(dev1, dev2, pstruEventDetails-
>dEventTime);
        double eh_energy= EH_FRACTION * rx_pwr;
        ptrBATTERY battery = WSN_PHY(dev2)->battery;
        if(battery)
        battery_energy_harvesting(battery, eh_energy);
}
```

10. The function call shown in red is added to 802_15_4.c file

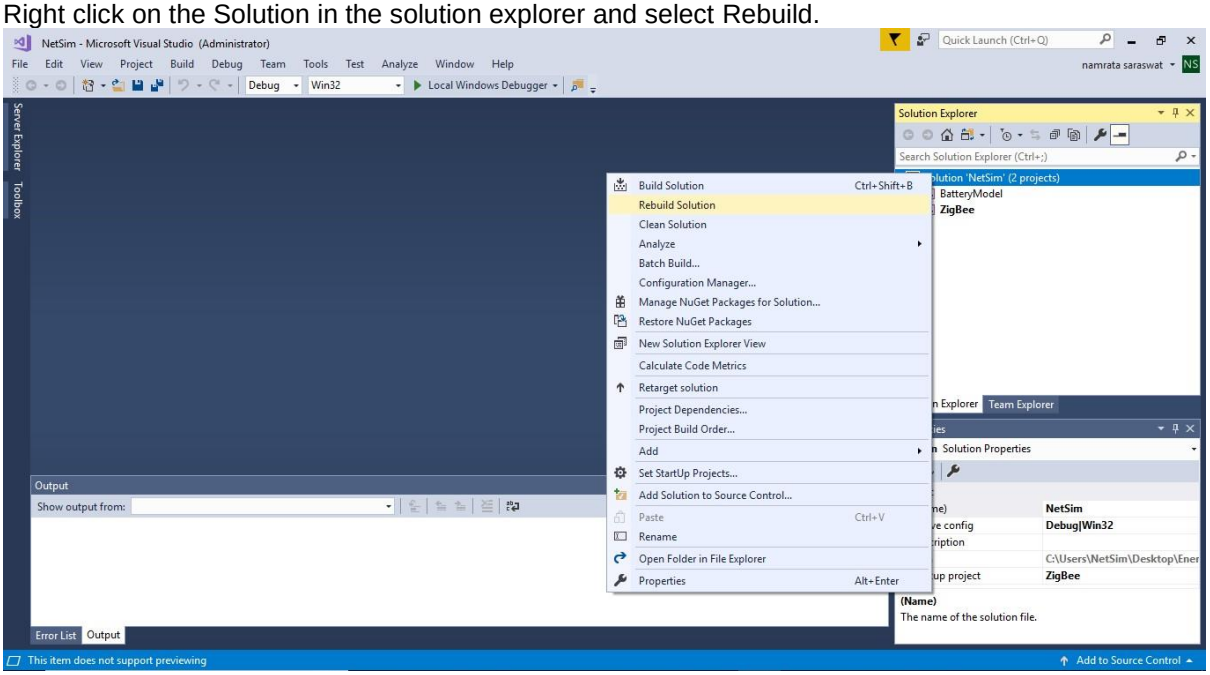
```
case UPDATE_MEDIUM:
    {
        double dtime=pstruEventDetails->dEventTime;
        NETSIM_ID nLink_Id, nConnectionID, nConnectionPortID, nLoop;
        NETSIM_ID nTransmitterID;
        nTransmitterID = pstruEventDetails->nDeviceId;
        ZIGBEE_CHANGERADIOSTATE(nTransmitterID, WSN_PHY(nTransmitterID)-
    >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 modifications for energy harvesting.

11. Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit Dll files respectively as shown below:

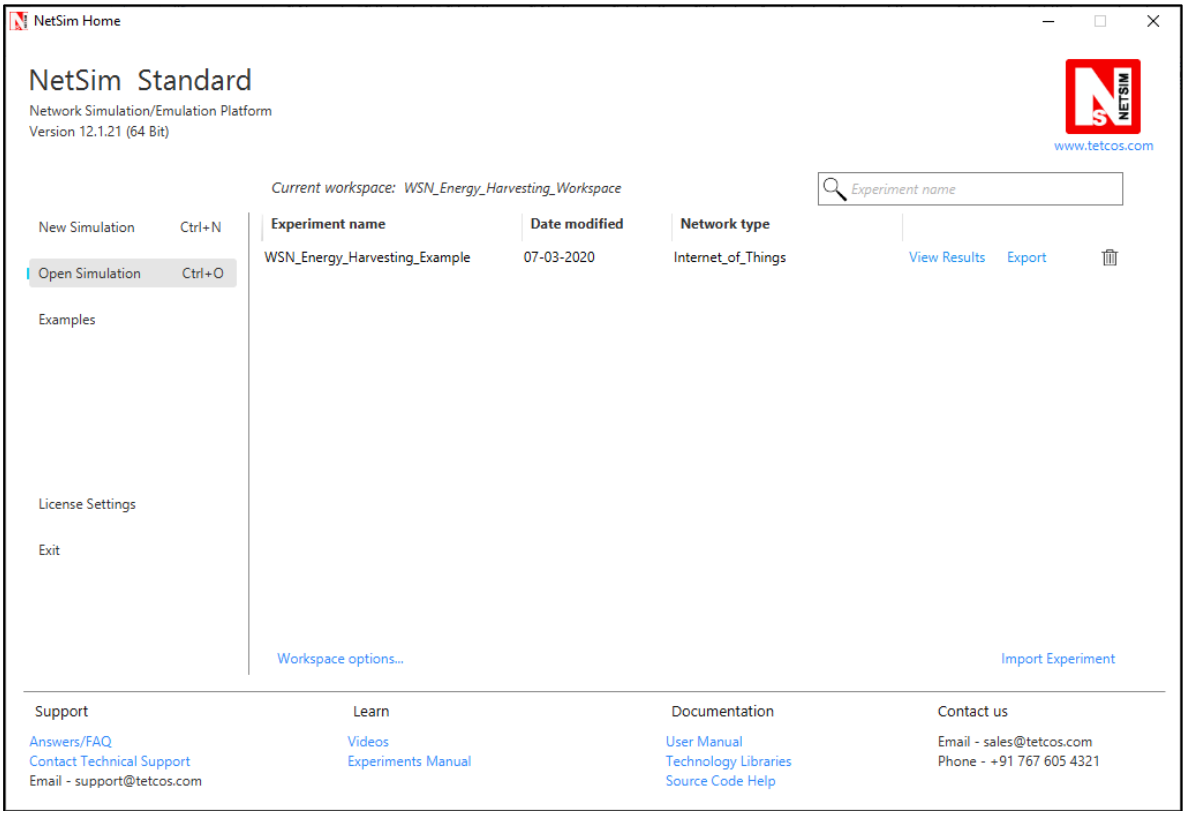


12. Right click on the Solution in the solution explorer and select Rebuild.



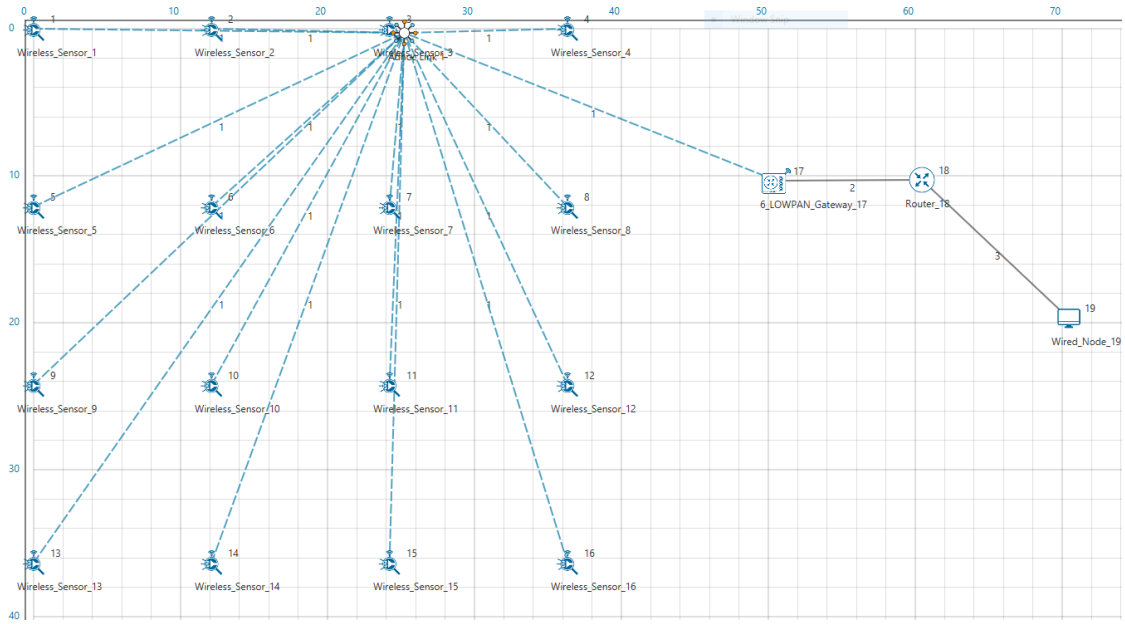
13. Upon successful build modified libZigBee.dll and BatteryModel.dll file gets automatically updated in the directory containing NetSim binaries.

14. Then WSN_Energy_Harvesting_Workspace comes with a sample configuration that is already saved. To open this example, go to Open Simulation and click on the WSN_Energy_Harvesting_Example that is present under the list of experiments as shown below:

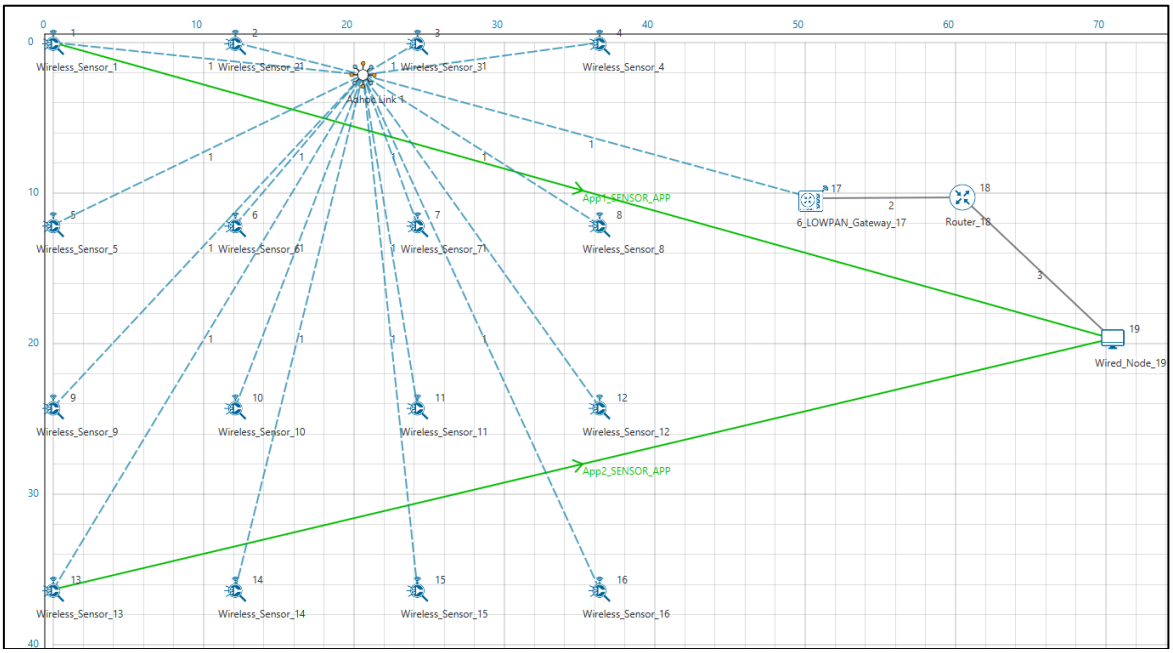


COMPARATIVE ANALYSIS:

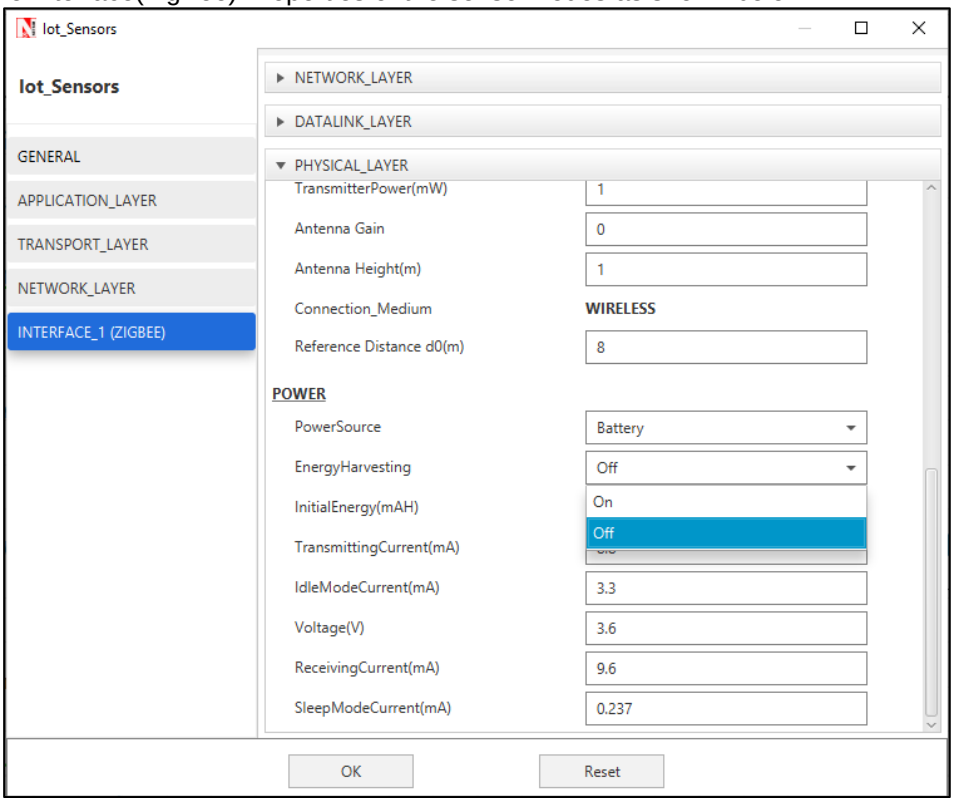
1. Create a network scenario in IoT with say 16 sensors, a 6LoWPAN Gateway, a Router and a Wired Node as shown below:



2. 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:



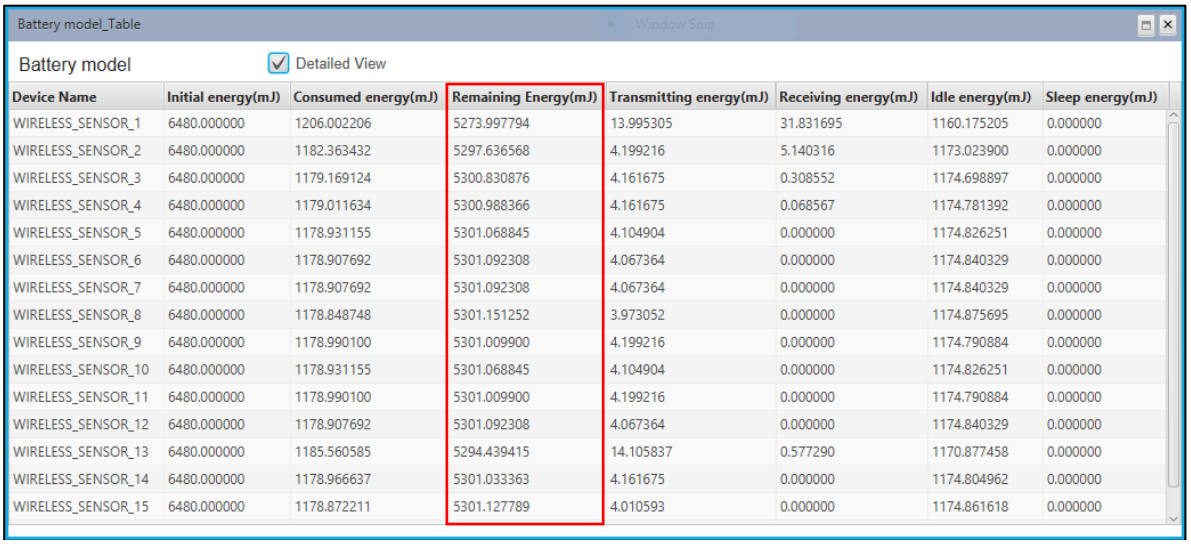
3. Disable Energy Harvesting in the Sensor nodes by setting the EnergyHarvesting parameter to OFF in the Interface(ZigBee) Properties of the sensor nodes as shown below:



4. 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	5273.997794	13.995305	31.831695	1160.175205	0.000000
WIRELESS_SENSOR_2	6480.000000	1182.363432	5297.636568	4.199216	5.140316	1173.023900	0.000000
WIRELESS_SENSOR_3	6480.000000	1179.169124	5300.830876	4.161675	0.308552	1174.698897	0.000000
WIRELESS_SENSOR_4	6480.000000	1179.011634	5300.988366	4.161675	0.068567	1174.781392	0.000000
WIRELESS_SENSOR_5	6480.000000	1178.931155	5301.068845	4.104904	0.000000	1174.826251	0.000000
WIRELESS_SENSOR_6	6480.000000	1178.907692	5301.092308	4.067364	0.000000	1174.840329	0.000000
WIRELESS_SENSOR_7	6480.000000	1178.907692	5301.092308	4.067364	0.000000	1174.840329	0.000000
WIRELESS_SENSOR_8	6480.000000	1178.848748	5301.151252	3.973052	0.000000	1174.875695	0.000000
WIRELESS_SENSOR_9	6480.000000	1178.990100	5301.009900	4.199216	0.000000	1174.790884	0.000000
WIRELESS_SENSOR_10	6480.000000	1178.931155	5301.068845	4.104904	0.000000	1174.826251	0.000000
WIRELESS_SENSOR_11	6480.000000	1178.990100	5301.009900	4.199216	0.000000	1174.790884	0.000000
WIRELESS_SENSOR_12	6480.000000	1178.907692	5301.092308	4.067364	0.000000	1174.840329	0.000000
WIRELESS_SENSOR_13	6480.000000	1185.560585	5294.439415	14.105837	0.577290	1170.877458	0.000000
WIRELESS_SENSOR_14	6480.000000	1178.966637	5301.033363	4.161675	0.000000	1174.804962	0.000000
WIRELESS_SENSOR_15	6480.000000	1178.872211	5301.127789	4.010593	0.000000	1174.861618	0.000000

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.