International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 07 | July -2016 www.irjet.net p-ISSN: 2395-0072

Simulation of Wireless Network Using TrueTime Toolbox

Vipul Rajyaguru[1], Dhwanit Chotaliya[2]

[1,2] Assistant Professor, Instrumentation and Control Department, Government Engineering College, Rajkot, Gujarat, India

---***---

Abstract - *Research on Wireless Sensor Networks (WSN) has received considerable attention in the last few years due to their unique characteristics, including, flexibility, self-organization, easy deployment, which make them an ideal candidate for low-cost monitoring. In order to help WSN planning and to enable the design of new protocols and applications and assess their performance, TrueTime simulation platform can be used. This paper presents performance assessment of ZigBee & WiFi protocol considering power control scheme and analysis of effect of interference in network.*

Key Words: **ZigBee, Wi-Fi, TrueTime, WSN, Matlab, Sensor**

1.INTRODUCTION

Wireless Sensor Networks (WSN) consists of a number of battery powered sensor nodes, endowed with physical sensing capabilities, limited processing and memory, and short-range radio communication. These nodes collectively form a network and forward gathered information to a data sink or gateway.

In general, a sensor node includes a sensing device for data acquisition from the physical environment, a processing subsystem for local data processing and storage, and a wireless communication module. Additionally, a power source supplies the energy needed by the device to perform all the programmed tasks. This power source is in most cases battery-driven with limited available energy. Therefore, maximizing the network lifetime is also an important challenge and should be tackled either in the planning stage or by means of a recursive optimization of the network lifetime under minimal coverage constraint. Thus, in order to support planning and deployment as well as to enable testing of new protocols and applications, simulation platforms have been used to simulate the performance of WSN protocols. A comparative analysis of several WSN simulation platforms can be found in [1].

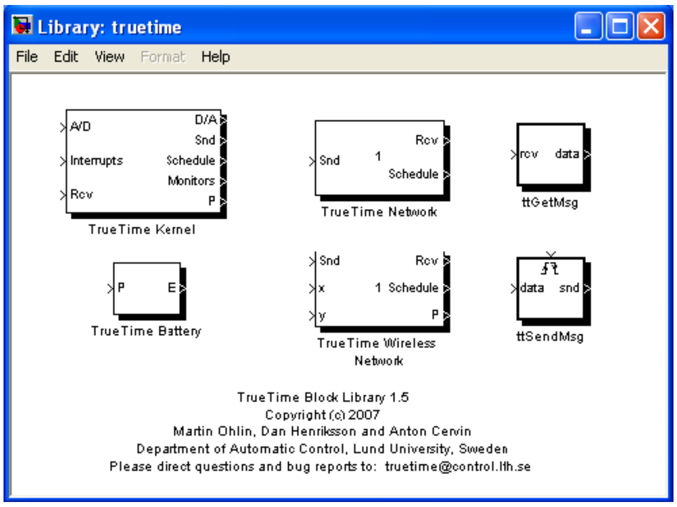
The approach followed here makes use of the TrueTime toolbox [2], [4], which is a freeware Matlab® library for simulating networked and embedded real-time control systems.

The organization of this paper is as follows. Section II gives an overview of the simulation platform, pointing out the enhanced features and flexibility. Control loop based on ZigBee protocol is presented in Section III, while in Section IV some results are drawn. Section V contains conclusion & Section VI gives references used.

2. THE SIMULATION PLATFORM

TrueTime [2], [4], [5] is a MATLAB/Simulink-based tool that facilitates simulation of the temporal behaviour of a multitasking real-time kernel executing controller tasks. The tasks are controlling plants that are modelled as ordinary continuous-time Simulink blocks. TrueTime also makes it possible to simulate simple models of communication networks and their influence on networked control loops.

TrueTime provides a number of Simulink blocks, which are shown in Fig. 1. The kernel block is event-driven and executes code that models, e.g., I/O tasks, control algorithms, and network interfaces. The scheduling policy of the individual kernel blocks is arbitrary and can be decided by the user. Likewise, in the network, messages are sent and received according to a chosen network model.



TrueTime Block Library 1.5
Copyright (c) 2007
Martin Ohlin, Dan Henriksson and Anton Cervin
Department of Automatic Control, Lund University, Sweden
Please direct questions and bug reports to: truetime@control.lth.se

Fig. 1 TrueTime Library

2.1 The Kernel Block

The kernel block is a Simulink S-function that simulates a computer with a simple but flexible real-time kernel, A/D and D/A converters, a network interface, and external interrupt channels. The kernel executes user-defined tasks and interrupts handlers. Internally, the kernel maintains several data structures that are commonly found in a real-time kernel: a ready queue, a time queue, and records for tasks, interrupt handlers, monitors and timers that have been created for the simulation.

An arbitrary number of tasks can be created to run in the TrueTime kernel. Tasks may also be created dynamically as the simulation progresses. Tasks are used to simulate both periodic activities, such as controller and I/O tasks, and aperiodic activities, such as communication tasks and event-driven controllers. Aperiodic tasks are executed by the creation of task instances (jobs). Each task is characterized by a number of static (e.g., relative deadline, period, and priority) and dynamic (e.g., absolute deadline and release time) attributes.

2.2 The Network Block

TrueTime has two types of network blocks: for wired networks and for wireless networks. The network blocks are event-driven and executes when messages enter or leave the network. When a node tries to transmit a message, a triggering signal is sent to the network block on the corresponding input channel. When the simulated transmission of the message is finished, the network block sends a new triggering signal on the output channel corresponding to the receiving node. The transmitted message is put in a buffer at the receiving computer node.

Configuring the network blocks involve specifying a number of general parameters, such as transmission rate, network model, and probability for packet loss. Protocol-specific parameters that need to be supplied include the time slot and cyclic schedule in the case of TDMA.

3. WIRELESS CONTROL LOOP

The control loop based on Wireless protocols proposed here is as shown in Fig 2.

Fig 2 is the schematic diagram of entire network system with 1 sensor, an actuator, and a controller and interference node. Diagram is made in matlab / Simulink environment. Node 1 represents actuator node. Node 2 represents sensor node. Node 3 represents controller node. Node 4 represents interference node. All four nodes contains TrueTime kernel Block inside as a sub system. Block with labelled 'Wireless Network' is defined more in its low level which is shown in below Fig 3.

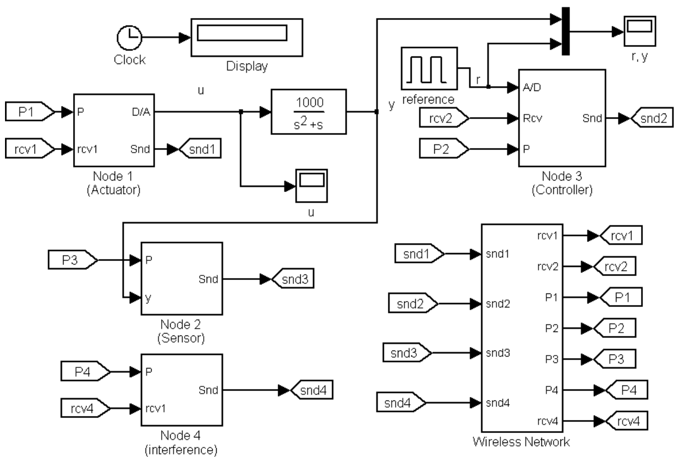


Fig. 2 Proposed Control Loop

The TrueTime wireless Network block is the block from the TrueTime Simulink library. The diagram represents that there are 4 nodes in the network. Input signal to the 'Snd' represents, wireless signals which are being transmitting. In the figure 3 x and y can be used to represent the location of each node in 2-D. the TrueTime wireless Network also allows us to define the amount of noise from the each device in the wireless network. 'Rcv' output indicates number of signals which has to reach at its destination node through the wireless network. Output of the scheduling shows the scheduling of the entire Wireless network. Through the Loss signal we can find out the number of lost signal during wireless communication.

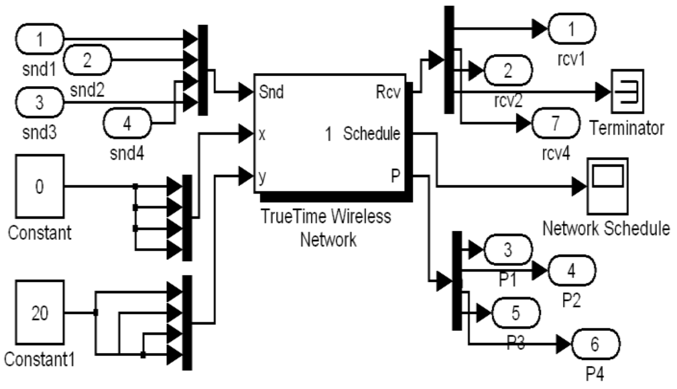


Fig.3 Wireless Control Loop

Drilling down the TrueTime Wireless Network block we get the User interface which allows us to choose different network model i.e. IEEE 802.11 b/g or ZigBee. After selecting the network model we can set the different parameters for the network model as shown in the below Fig 4.

3.1 Sensor Node

To initialize the sensor node in the network, we need to define number of analog inputs and outputs for the node and scheduling policy of the function i.e. fixed priority, deadline monotonic or earliest deadline first. This can be achieved by calling and initializing the function 'ttInitKernel' e.g. writing ttInitKernel(nbrOfInputs, nbrOfOutputs, scheduling policy). Create periodic task which wakes up after every 0.01s by calling the function 'ttCreatePeriodicTask'. To call this function we need to define different parameters which are shown below:

ttCreatePeriodicTask ('task name', offset, period, prio, 'function name', data);

To use the interrupt handler function we need to create interrupt handler task, define the priority of the function and function name which should be executed when interrupt occurs. This can be achieved as following:

ttCreateInterruptHandler(handler name, priority, function name);

Code for sensor_init

```
function sensor_init
```

```
ttInitKernel(1, 0, 'prioFP');
data.y = 0;
offset = 0;
period = 0.010;
prio = 1;
ttCreatePeriodicTask('sens_task', offset,
 period, prio, 'senscode', data);
ttCreateInterruptHandler('nw_handler',
 prio, 'msgRcvSensor');
ttInitNetwork(4, 'nw_handler');
```

3.2 Actuator Node

Initialization of actuator is same as the sensor node, except actuator has a mailbox and an event driven task to receive data from the controller. Interrupt handler functions same as for the controller as receiver task.

Actuator task is segmented in the three case segments. In the first segment actuator task fetches data from the mailbox. In the next segment actuator writes data to the process plant in the analog form. The process plant performs control operation depending on the received control signal. In the third segment actuator task quits.

Code for autuator_init

```
function actuator_init
ttInitKernel(0, 1, 'prioFP');
deadline = 100;
prio = 1;
ttCreateTask('act_task', deadline, prio,
 'actcode');
ttCreateInterruptHandler('nw_handler',
 prio, 'msgRcvActuator');
ttInitNetwork(2, 'nw_handler');
```

3.3 Controller Node

Initialization of the controller is same as the actuator node.
The Controller task is also segmented in the three case segments. The controller task contains PID algorithm to calculate the controller output.

Code for controller_init

```
function controller_init
ttInitKernel(1, 0, 'prioFP');
h = 0.010;
N = 100000;
Td = 0.035;
K = 1.5;
data.u = 0.0;
data.K = K;
data.ad = Td/(N*h+Td);
data.bd = N*K*Td/(N*h+Td);
data.Dold = 0.0;
data.yold = 0.0;
deadline = h;
prio = 1;
ttCreateTask('pid_task', deadline, prio,
 'ctrlcode', data);
ttCreateInterruptHandler('nw_handler',
 prio, 'msgRcvCtrl');
ttInitNetwork(3, 'nw_handler');
```

3.4 Interference Node

Initialization of interference node is same as sensor node.Interference node generates disturbing network traffic in wireless network. To add interference in wireless network interference node transmits and receives frame to itself. The level of interference can be changed by changing the bandwidth occupation provided to interference node.

Code for interference_init

```
function interference_init
ttInitKernel(0, 0, 'prioFP');
offset = 0;
period = 0.001;
prio = 1;
ttCreatePeriodicTask('interf_task',
 offset, period, prio, 'interfcode');
ttCreateInterruptHandler('nw_handler',
 prio, 'msgRcvInterf');
ttInitNetwork(1, 'nw_handler');
```

TABLE
SUMMARY TO INITIALIZE THE NETWORK DEVICES AND CONTROLLER

	Sensor	Actuator	Controller	interference
No. Analog I/O	Yes	Yes	Yes	Yes
Create Mailbox	No	Yes	No	Yes
Tasks	Periodic	Aperiodic	Periodic	Periodic
Interrupt Handler	Yes	Yes	Yes	Yes
Assign node number	Yes	Yes	Yes	Yes

3.5 Power Control Scheme

Power controller scheme consumes battery power according to the variation in signal strength. If signal strength at receiver is low then power controller scheme consumes more battery to receive the signal.

Code for power controller scheme:

```
%Create power controller task
offset = 0;
period = 0.025;
prio = 3;
power_data.transmitPower = 20;
power_data.name = 1;
power_data.receiver = 2;
power_data.haverun = 0;
```

```
ttCreatePeriodicTask('power_controller_ta
 sk', offset, period, prio,
 'powctrlcode', power_data);
%Create power response task
deadline = 100;
prio = 4;
ttCreateTask('power_response_task',
deadline, prio, 'powrespcode');
```

4 RESULTS

4.1 Wi-Fi

1. With power control scheme with interference:

By adding power control scheme in actuator node and adding interference in the network by setting bandwidth share = 0.025 we have the results as shown in Fig 4.

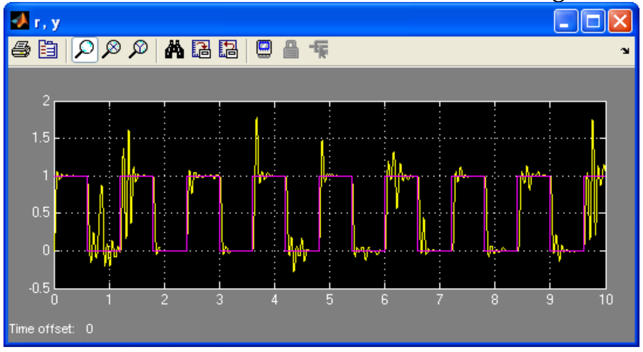


Fig. 4

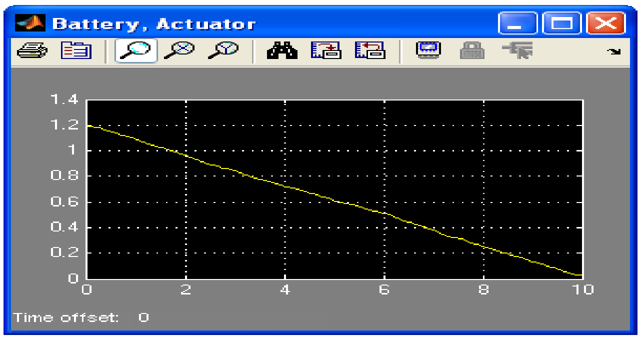


Fig. 5

2. Without power control scheme Without interference:

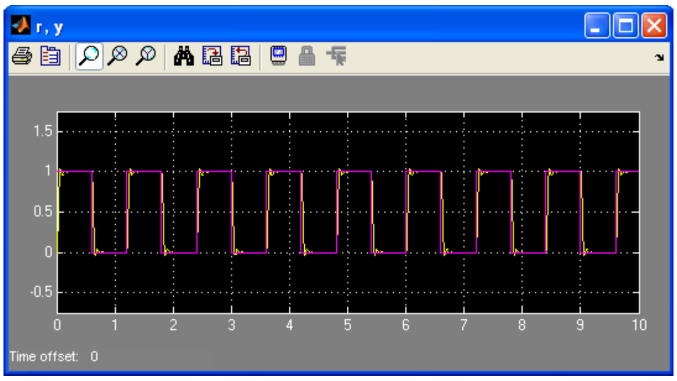


Fig. 6

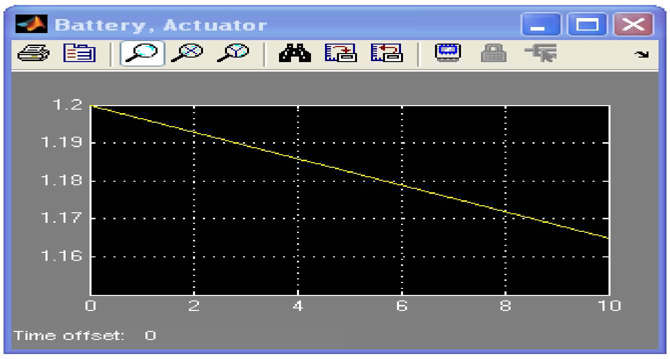


Fig. 7

4.2 Zig-Bee

1. With power control scheme with interference

By adding power control scheme in actuator node and adding interference in the network by setting bandwidth share = 0.025 we have the results as shown in Fig 8.

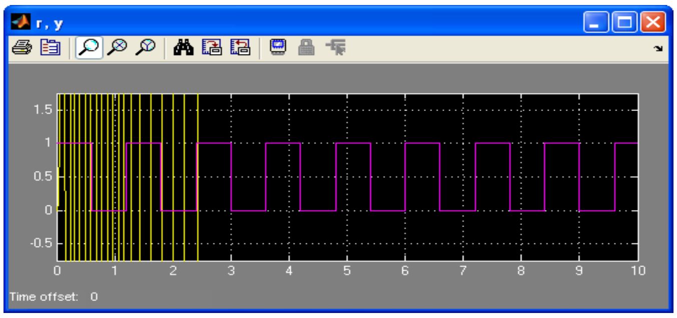


Fig. 8

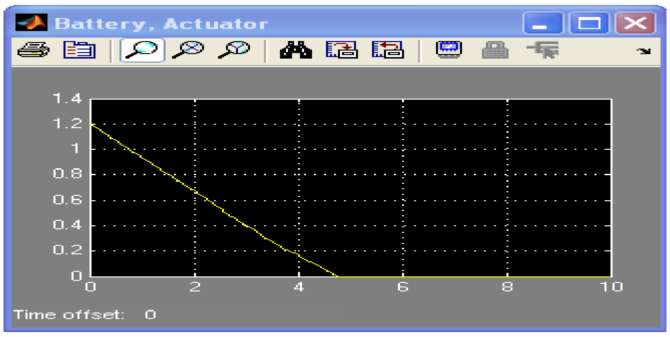


Fig. 9

2. Without power control scheme without interference

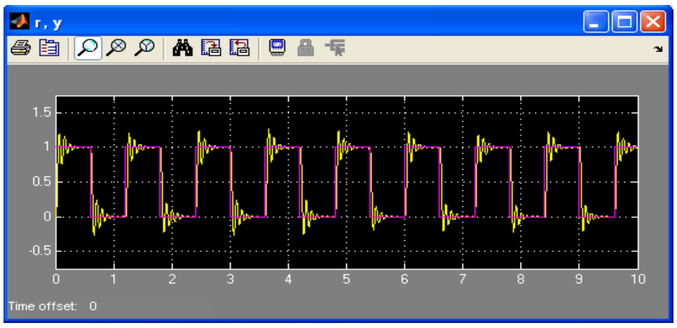


Fig. 10

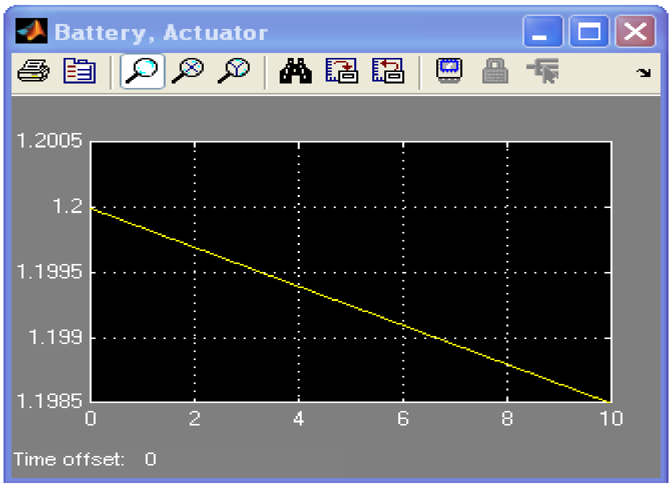


Fig. 12

5. CONCLUSION

From the results we can conclude that as we use the power control scheme the performance of wireless control loop is improved but power consumed from battery is also increased. Wi-Fi uses more power than ZigBee. Wi-Fi has high signal reach capabilities then Zigbee. As ZigBee has low signal reach capabilities, use of power control scheme in wireless control loop increases power consumption, which causes the battery to run out of energy and subsequently lost of control.

While running simulation without power control scheme we can see that power drain is constant throughout the simulation.

We can also observe that Wi-Fi has less effect of interference compared to ZigBee, as a fact that it has high signal reach capabilities.

REFERENCES

[1] Mikael Bjorkbom, "Wireless control system simulation and network adaptive control", Helsinki University of Technology, Control Engineering, 2010

[2] Alberto Cardoso1, Sérgio Santos, Amâncio Santos, "Simulation Platform for Wireless Sensor Networks based on the TrueTime Toolbox" CISUC, Department of Informatics Engineering, University of Coimbra, Portugal, IEEE Conference 2009

[3] Kunjesh Shah, "Design and Implementation of a simulator in support of WirelessHART-based control systems development", Embedded Systems (Electrical Engineering),2009

[4] Glen Stone, Bruce Fairman, Jos Laake, "TrueTime Software Architecture", Interconnect Architecture Division, Network Software & Technology Center of America, SONY, IEEE, 2002

[5] M. Ohlin, D. Henriksson, and A. Cervin. TrueTime1.5-Reference Manual. Department of Automatic Control Lund University, 2007

[6] Dan Henriksson, Anton Cervin, Martin Andersson, Karl-Erik Årzén ,"TrueTime: Simulation of Networked Computer Control Systems", Department of Automatic Control, Lund University, Sweden, 2nd IFAC Conf. on Analysis and Design of Hybrid Systems (Alghero, Italy), 7-9 June 2006

[7] T. Chvostek, A. Kratky, M. Foltin "Simulation of Network Using TrueTime ToolBox", Institute of Control and Industrial Informatics, Faculty of Informatics and Information Technologies, Ilkovičova 3, 812 19 Bratislava, Slovak Republic, 2007

[8] A. Cervin, K.E.Arzen, D. Henriksson, "Control Loop Timing Analysis Using TrueTime and Jitterbug," Department of Automatic Control, LTH., Lund University, Sweden, IEEE Conference on Computer Aided Control Systems Design Munich, Germany, October 4-6, 2006

BIOGRAPHIES

Name: Dhwanit K. Chotaliya
Assistant Professor,
Instrumentation and Control Department,
Government Engineering College,
Rajkot, Gujarat, India

Name: Vipul C. Rajyaguru
Assistant Professor,
Instrumentation and Control Department, Government Engineering College, Rajkot, Gujarat, India