



Inteligência Distribuída usando o módulo LabVIEW WSN

ni.com/wsn

Renato Fernandes
Engenheiro de Vendas

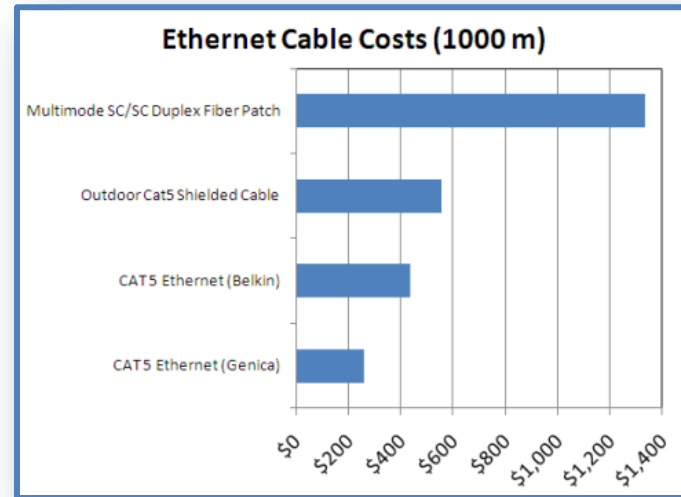


Abel Souza
Engenheiro de aplicação



Necessidade de medições sem fio

Corte de fios, corte de gastos



Eliminação de fios simplifica a instalação

Atende as necessidades de novas aplicações



Áreas de aplicação



Eficiência energética

Smart grid, monitoramento de energia



Monitoramento ambiental

Emissão de CO₂ , mudança climática



Monitoramento estrutural (SHM)

Infraestrutura de pontes, monitoramento de edifício



Medição industriais

Monitoramento de máquina, medições em área de risco

Usando o LabVIEW para construir uma WSN

Adicionando inteligência usando uma linguagem gráfica intuitiva

- Conecte aos nós de medição NI e componentes de 3rd
- Integre medições wireless com NI PXI
- dispositivos de rede inteligente e laptops
- Centenas de funções integradas
- Interface de usuário integrada

Gateways

1. Gateway Ethernet
2. Gateway Programável
3. Gateway Série C

Integração com o CompactRIO



Nós

1. Nó de entrada analógica
2. Nó de Termopar
3. Nó de RTD/tensão/resistência
4. Nó deformação/ponte
5. Nó RS232
6. Nó RS485

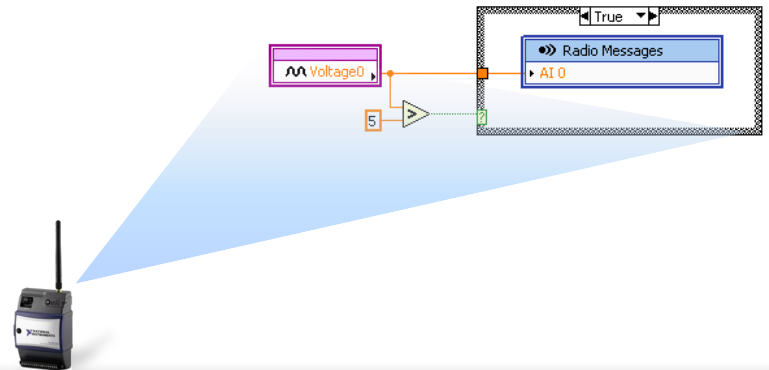
* Todos os nós WSN contem E/S digitais de propósito geral

**Programável e não programável

Os nós de medição WSN permitem:

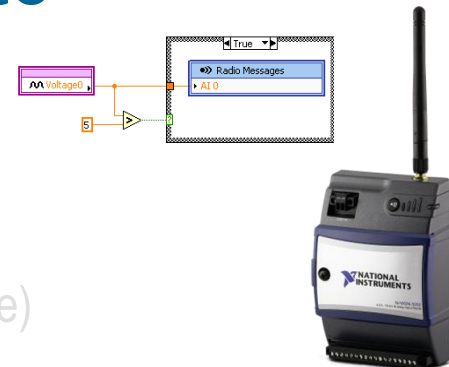
- Transferir dados de um nó ao gateway a uma distância de até 300 metros.
- Intervalo de amostras de aproximadamente **uma** amostra por segundo.
- Controle de um aplicativo host
- Vida útil da bateria.

Porquê programar no nó de medição?



Benefícios de usar programáticamente um nó WSN de medição

- Extenso tempo de vida da bateria
 - Transmite somente os dados mais significativos (ex. abaixo do limite)
 - Transmite com menos frequência que você adquire
 - Intervalos de transmissão e amostragem adaptada de acordo com as condições de operação



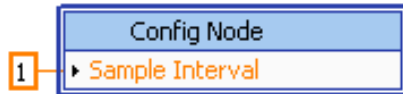
- **NI LabVIEW**
 - Converte linha de dado em unidades de engenharia
 - Calcula a média dos dados
- **Modulo Wireless Sensor Network (WSN)**

- Realiza o controle embarcado
 - Liga e desliga reles e atuadores sem a interação do host
- Armazena dados na memória flash local
- Responde a eventos digitais ou mudança de status na reue
- Realiza atualização de firmware enviados por via aérea



LabVIEW WSN versus linguagem de programação baseada em texto alternativo

1) Define o intervalo para 1Hz



2) Alternar um LED

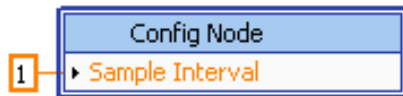


```
1 //Contents configuration file, Blink.c
2
3
4 configuration Blink {
5 }
6 implementation {
7     components Main, BlinkM, SingleTimer, LedsC;
8
9     Main.StdControl -> BlinkM.StdControl;
10    Main.StdControl -> SingleTimer.StdControl;
11    BlinkM.Timer -> SingleTimer.Timer;
12    BlinkM.Leds -> LedsC;
13 }
14
15 //Contents of the module BlinkM.nc
16 module BlinkM {
17     provides {
18         interface StdControl;
19     }
20     uses {
21         interface Timer;
22         interface Leds;
23     }
24 }
25
26 implementation {
27
28     command result_t StdControl.init() {
29         call Leds.init();
30         return SUCCESS;
31     }
32
33     command result_t StdControl.start() {
34         return call Timer.start(TIMER_REPEAT, 1000);
35     }
36
37     command result_t StdControl.stop() {
38         return call Timer.stop();
39     }
40
41     event result_t Timer.fired()
42     {
43         call Leds.redToggle();
44         return SUCCESS;
45     }
46 }
```

NesC no TinyOS

LabVIEW WSN versus linguagem de programação baseada em texto alternativo

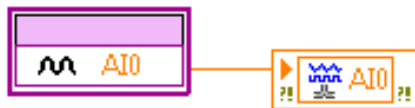
1) Define o intervalo para 1Hz



2) Alterna um LED



3) Adquiri e envia uma amostra

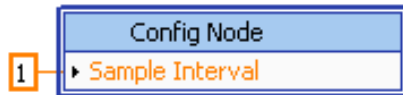


```
1 implementation {
2   components MyApp, Timer, LEDC, Photo, GenericComm as Comm;
3
4   main:stccontrol -> Timer.stccontrol;
5   main:stccontrol -> MyApp.stccontrol;
6   main:stccontrol -> Comm.stccontrol;
7
8   MyApp.Timer -> Timer.Timer[unique("Timer")];
9
10  MyApp.LED -> LEDC.LED1;
11  MyApp.PhotoControl -> Photo.PhotoControl;
12  MyApp.Light -> Photo.ExternalPhotoAcc;
13
14  MyApp.Sending -> Comm.Sending[unique("Comm")];
15
16 }
17
18 module MyApp {
19   provides {
20     interface stccontrol;
21   }
22   uses {
23     interface Timer;
24     interface LEDC;
25     interface stccontrol as PhotoControl;
26     interface ACC as Light;
27     interface Sending;
28   }
29 }
30
31 implementation {
32   bool sending_packet = FALSE;
33   bool msg_ready;
34   xdata msg *pack;
35
36   /*
37    * Initialize the component.
38    *
39    * Return Always returns <code>SUCCESS</code>
40    */
41   command result_t stccontrol_init() {
42     call Ledc_init();
43     call PhotoControl_init();
44
45     // initialize the message packet with default values
46     pack = (xdata msg *)calloc(sizeof(msg), 1);
47     pack->sensorheader.user_id = SENSOR_BOARD_ID;
48     pack->sensorheader.packet_id = 1;
49     pack->sensorheader.mode_id = TSL_LOCAL_ADDRESS;
50     pack->sensorheader.rsvd = 0;
51   }
52 }
53
54 return;
```

```
1 command result_t PhotoControl_start()
2 {
3   atomic photodensor = statefalse;
4   TOSH_SET_PHOTO_CTL_PIN();
5   return SUCCESS;
6 }
7
8 command result_t PhotoControl_stop()
9 {
10  atomic photodensor = statefalse;
11  TOSH_CLR_PHOTO_CTL_PIN();
12  return SUCCESS;
13 }
14
15 // Gets the next sample, deals with which sample to get now
16 task void getSample()
17 {
18   TOSH_CLR_TOSH_CTL_PIN();
19   TOSH_SET_TOSH_CTL_PIN();
20   TOSH_CLR_PHOTO_CTL_PIN();
21   TOSH_SET_PHOTO_CTL_PIN();
22   call Phototimer_start(); // fail in case
23   if (call Phototimer_start(TIMEOUT_WAIT, 1)) != SUCCESS)
24   {
25     post getSample();
26   }
27   return;
28 }
29
30 // after waiting a little we can take a reading
31 event result_t Phototimer_fired()
32 {
33   if (call InternalPhotoAcc_getData() == SUCCESS)
34   {
35     // trigger the read which will post a new sample
36     TOSH_CLR_PHOTO_CTL_PIN();
37     return SUCCESS;
38   }
39   return SUCCESS;
40 }
41
42 async command result_t ExternalPhotoAcc_getData()
43 {
44   atomic photodensor = statefalse;
45   post getSample();
46   return SUCCESS;
47 }
```

LabVIEW WSN versus linguagem de programação baseada em texto alternativo

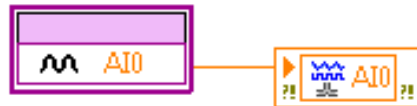
1) Define o intervalo para 1Hz



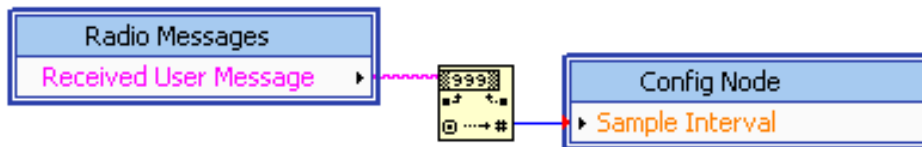
2) Alterna um LED



3) Adquiri e envia uma amostra



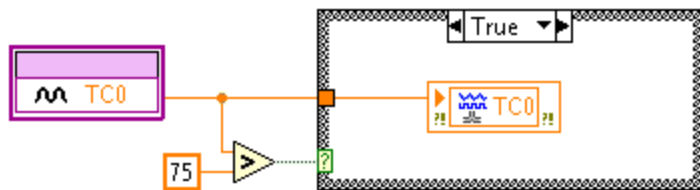
4) Atualiza a taxa de amostragem do host



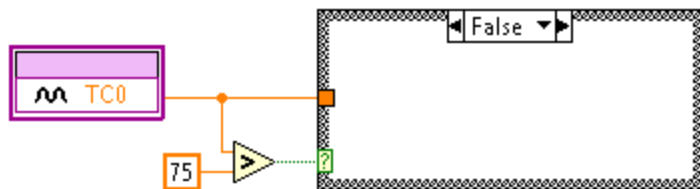
```
1 provides interface stControl as PhotostControl;
2 provides interface AIO as ExternalPhotost;
3 uses
4
5 interface AIOControl;
6 interface AIO as ExternalPhotost;
7 interface stControl as PhotostControl;
8 interface stControl as PhotostControl;
9
10 implementation
11
12 // Loop what the hardware is set up to do.
13 enum {
14     sensorReady = 0,
15     sensorPhotostReady,
16     sensorPhotostReady,
17     sensorPhotostReady,
18     sensorPhotostReady,
19 } hardwareReady;
20
21 // Loop what a particular sensor is trying to do when a single
22 // read completes the value reverts to false.
23 typedef enum {
24     stateIdle = 0,
25     statePhotostReady,
26     statePhotostReady,
27     statePhotostReady,
28 } sensorReady;
29
30 SensorReady = PhotostReady;
31 SensorReady = PhotostReady;
32
33 // Note when waiting for a sample to be read and another sample can
34 // not start, getSample will always be triggered again when this is
35 // true.
36 bool waitToGetSample;
37
38 // Command result_L PhotostControl, Get()
39 {
40     call AIOControl, PhotostReady, AIO, PhotostReady, PhotostReady;
41     call PhotostControl, Get();
42     atomic photostReady = stateIdle;
43     return call AIOControl, Get();
44 }
45
46 atomic photostReady = stateIdle;
47 bool sendSample = FALSE;
48 bool sendSample = FALSE;
49 bool sendSample = FALSE;
50
51 // Initialize the component.
52 // Return Always returns code=SUCCESS/NOOP
53 // Command result_L PhotostControl, Get()
54 {
55     call AIOControl, Get();
56     call PhotostControl, Get();
57 }
58
59 // Initialize the message packet with default values
60 atomic {
61     packetId = 0;
62     packetId = 0;
63     packetId = 0;
64 }
65
66 // Start things up. This just sets the rate for the clock component.
67 // Return Always returns code=SUCCESS/NOOP
68 // Command result_L PhotostControl, Start()
69 {
70     // Start a repeating timer that fires every 1000ms
71     return call Timer, Start(1000, 1000);
72 }
73
74 // Main execution of the application.
75 // This just disables the clock component.
76 // Return Always returns code=SUCCESS/NOOP
77 // Command result_L PhotostControl, Stop()
78 {
79     return call Timer, Stop();
80 }
```

Otimizar o consumo de energia com LabVIEW WSN

- O rádio é o maior consumidor de energia
- Transmitir apenas os dados significativos ou a média para aumentar a vida útil da bateria



Se a temperatura for superior 75 graus, enviar dados.



Quando abaixo de 75 graus, não enviar dados.

Benchmark na análise da vida útil da bateria (3212)

Intervalo de amostragem (segundos)	Intervalo de transmissão (segundos)	Vida útil da bateria (meses)
1	1	1.36
5	5	6.33

Utilizando o LabVIEW WSN nos nós podemos aumentar a vida útil da bateria

NOVOS PRODUTOS E APLICAÇÕES

Gateways para rede de sensores wireless

Especificações

- 2.4 GHz, IEEE 802.15.4
- Até 36 nós por gateway
- Distância de até 300 m
- Montado em painel ou trilho DIN
- Classificação industrial de temperatura
- 50 g_{rms} de choque, 5 g de vibração



Gateway	Programação	Processador	RAM (MB)	Ethernet	Onboard Storage	Porta Serial RS232	Porta USB	Alimentação	Alimentação Backup	Web/ FTP Server	WSN Radio	Temp. de operação
WSN-9791	--	266 MHz PPC	64	1: 10/100	--	--		9 to 30 VDC	--	--	✓	-30 to 70 °C
NI 9792	LabVIEW Real-Time	533 MHz PPC	256	1: 10/100/1000, 1: 10/100	2 GB	✓	Hi-Speed	9 to 35 VDC	✓	✓	✓	-40 to 70 °C
NI 9795	LabVIEW Real-Time	Os mesmos recursos disponíveis no chassi CompactRIO/Controladora									✓	-40 to 70 °C

NI 9795: Gateway WSN Series C

"Expansão de E/S wireless para CompactRIO"

- Integração com o CompactRIO
 - API compatível
- Combina facilmente medições com fio e sem fio
 - Aumento das soluções com o WSN
- Um Gateway por chassis suporta até 36 nós
- O chassis deve suportar o RSI (RIO scan interface)



Nós de medições WSN

Especificações

- 2.4 GHz IEEE 802.15.4
- Distância de até 300 m
- Vida útil da bateria de até 3 anos / 4 pilhas AA
 - Entrada de 9-30 V (opcional)
- Configurável como um roteador mesh
- Classificações Industriais
 - Operating temperature -40 to 70 °C
 - 50 g_{rms} shock 5 g vibration



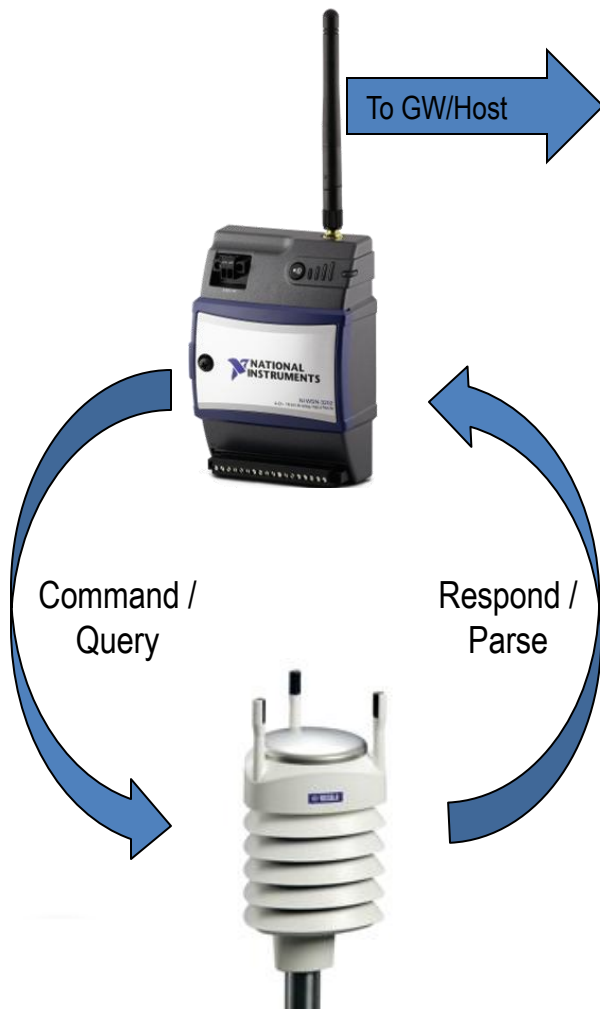
Nó	Canais de entrada analógica	Canais de E/S digital	Intervalo min. de amostras	Resolução	Ferramentas adicionais
WSN-3202 Nó de entrada analógica	4	4	1 segundo	16	Alimentação para sensores; faixa de entrada selecionável.
WSN-3212 Nó de entrada termopar	4	4	2 segundos	24	Tipos: J,K,R,S,T,N,B, E; suporta uma faixa de entrada analógica de +/- 73mV.
WSN-3226 Nó com RTD/Tensão	4	2	1 segundo	20	Alimentação para sensores; filtros de 50/60 Hz; Suporte para bateria de backup
WSN-3214 Nó complemento de ponte para strain gage	4	2	1 segundo	20	¼, ½ ou Ponte Completa para Strain Gage ou ratiometric; hardware temporizado aquisição de forma de onda
WSN-3230 Nó Serial RS232	0	2	1 segundo	-	Porta RS232, permite selecionar o baud rates, bit de paridade, stop bit, e fluxo de controle.
WSN-3231 Nó Serial RS485	0	2	1 segundo	-	Porta RS485, permite selecionar o baud rates, bit de paridade, stop bit, e fluxo de controle.

WSN-3214: Nó de ponte completa para straining gage



- Solução perfeita para SHM wireless
 - Pontes, edifícios, túneis, barragens, equipamentos
 - Engenheiros civis, técnicos de manutenção, construção, petróleo & gás
 - 4 canais analógicos com $\frac{1}{4}$, $\frac{1}{2}$ e ponte completa.
 - 350Ω e $1k\Omega$
 - Dados de forma de onda de até 10kS/s/ch
 - Excitação interna
 - Modo de alta velocidade e alta resolução
- 2 canais digitais E/S
- Realiza análise onboard e a redução de dados com LabVIEW WSN

WSN-3230 (RS-232) & WSN-3231 (RS-485)



- Interface wireless para sensores com comunicação serial:
 - “Sensor Programável/Controle de Instrumento”
 - Suportado para diversos dispositivos.
- Diversos tipos de aplicações
 - Monitoramento Ambiental
 - Control board interface
 - Monitoramento Solar Inverter
- **Único** Programável
 - Casos usando a plataforma WSN

Monitoramento de Estacionamento com WSN.



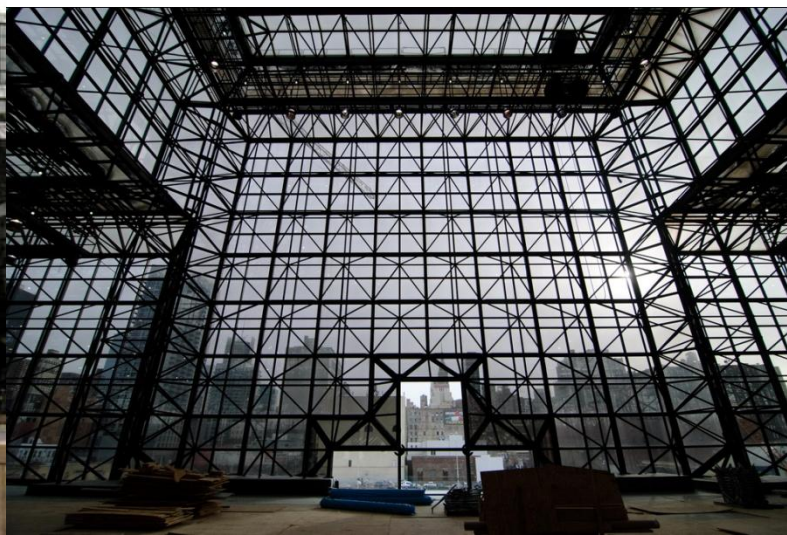
- Monitora números de vagas em estacionamentos
- Utilizando o LabVIEW WSN para detectar quando os veículos entram e saem do estacionamento
 - Sensores digitais para contagem dos veículos
 - Transmite os dados somente quando o veículo sai, ao invés de enviar os dados o tempo todo
 - Aumentando a vida útil da bateria do nó.
- Envia dados para o CompactRIO para processamento de atualização web e registro de banco de dados.



Monitoramento do Centro de Convenções com WSN

Solução do usuário: Monitorar o centro de convenção de Jacob Javits no centro de NY

- Mais de 50 dispositivos WSN montado em todas as salas de exposição
- Monitoramento de temperatura, umidade, deformação e deslocamento
- Modems celulares conectado ao gateway WSN NI 9792 programado para disponibilizar os dados
- Web Services hospedado no portal



Monitoramento com WSN do deslocamento longitudinal da ponte em suspensão

- **Challenge:** Continuously monitoring the longitudinal displacement of a suspension bridge deck using sensors located approximately **450 m** from the Web-accessible data acquisition PC.
- **Solution:** Using NI WSN-3202 analogue input nodes to transmit data wirelessly from the sensors to an NI WSN-9791 gateway node attached to the PC.
- **Product:** NI WSN-3202, NI 9791, LabVIEW

[Case study on ni.com](#)



"The NI WSN products provided a full-featured, reliable wireless platform that allowed us to connect any analogue sensor with little set up."-Nicky de Battista- [University of Sheffield](#)

Sistema de monitoramento com WSN para monitoramento e prevenção de colapso na construção de túneis.

- **Challenge:** Performing tunnel-digging construction safely by monitoring information such as temperature, humidity, bedrock side stress, and bedrock side water in a constantly unstable environment, which requires quickly taking measurements and regularly detecting abnormalities to prevent accidents from occurring.
- **Solution:** Using the NI wireless sensor network (WSN) platform to successfully build an in-tunnel collapse prevention monitoring system.
- **Products:** Wireless Sensor Network Module, NI WSN-3202, NI 9791, LabVIEW

[Case study on ni.com](http://ni.com)



"Using the NI Wireless Sensor Network platform, we successfully built an in-tunnel collapse prevention monitoring system. The NI WSN devices are programmable, very low cost, and can operate on battery power for a prolonged period of time." 鰐部 巧哉 氏 - [株式会社イー・アイ・ソル](http://www.e-i-sol.co.jp)

Sistema de monitoramento estrutural com WSN do estádio Fonte Nova – Salvador para Copa do Mundo de 2014

Application: Structural health monitoring (SHM) to determine stability, reliability, and livability of megastructures.

Challenge: Developing a reliable SHM system with continuous monitoring, rugged enclosure and remote access.

Products: LabVIEW and WSN 3214 - strain

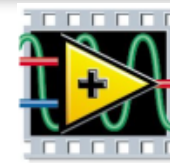


[Turbina Eólica – Quanto mais vento melhor?](#)

[Vídeo LabVIEW WSN](#)

DEMO – LABVIEW WSN



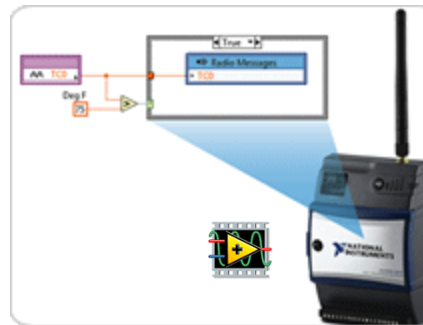


NATIONAL INSTRUMENTS

LabVIEW™

LabVIEW e NI WSN

- Use NI LabVIEW para construir sistemas de rede de sensores wireless
- Adicione medições sem fio para PACs da NI para criar uma solução completa com e sem fio
- Incorpore inteligência em nós de medição NI WSN com programação gráfica



Obrigado!

Renato Fernandes
Engenheiro de Vendas - ES
renato.fernandes@ni.com, 11 98603-0694

Abel Souza
Engenheiro de Aplicação

