

Sensor Network for Animal Husbandry

Aplicações para Sistemas Embebidos 2013/2014

João Filipe Garrett Paixão Florêncio – 58508
Guilherme de Sousa Aranha – 64766

2 de Maio de 2014

Conteúdo

1	Introdução	2
2	Descrição Geral	2
3	Arquitectura do Sistema	3
4	Arquitectura da Rede	4
5	Manual do Utilizador	5
6	Problemas encontrados no desenvolvimento do projecto	6
7	Especificações não cumpridas	6

1 Introdução

“The challenges faced by modern agriculture have never been greater. There has always been a need for livestock producers to be able to observe their animals as often as possible. But managing a large quantity of animals while at the same time letting them be as loose as possible is not an easy task. Recently, we have seen the rise in the use of wireless sensor networks to help in the management of animals.”

Conforme é mencionado no enunciado que serviu de base para a construção deste projecto, existe uma necessidade crescente de colocar os avanços tecnológicos ao serviço das actividades mais tradicionais da nossa economia. A agricultura e a pecuária sempre foram sectores importantes no desenvolvimento do país e como não podia deixar de ser, podem ser positivamente influenciadas pelo uso de sistemas computadorizados que ajudam na sua manutenção e desenvolvimento. Este projecto baseia-se nessa ideia. Em termos mais técnicos o objectivo era modelar e implementar em software, os nós de uma rede. Esta poderá posteriormente vir a ser montada com sensores nos próprios animais para mais agilmente gerir a sua localização, interacção e saúde.

2 Descrição Geral

O sistema é composto por:

1. Sensores (SensorNodes)
2. Feeding Spots
3. Emissores Rádio
4. Computador Portátil (Simulador)

O funcionamento normal do sistema pressupõe a existência dos quatro componentes importantes mencionados em cima.

Os **sensores** são uma parte fundamental do sistema. O objectivo final é que possam ser *deployed* num dispositivo que esteja em cada animal. Em termos de software, estão programados para comunicar com os feeding spots e com os emissores rádio. A comunicação com os “locais de alimentação” (feeding spots) serve para controlar a quantidade de alimento que o animal necessita e a quantidade que permanece no local.

Os **feeding spots** são os locais onde os animais se vão alimentar, guardam a informação de quanta comida existe ainda e dispensam comida aos animais. Acabam por ser eles mesmos também nós do sistema.

Os **emissores de rádio** são a ferramenta que é utilizada para obter a localização de cada animal. A localização não é a mais exacta mas para os efeitos e requisitos do sistema é suficiente.

Os **computadores portáteis** permitem aos utilizadores interagir com os outros componentes através da aplicação. Alterar a quantidade de comida nos locais, saber a localização dos animais, saber a quantidade de comida disponível, são tudo funções importantes do sistema que acabam por só poder ser activadas por este componente.

3 Arquitectura do Sistema

Os sensor nodes em termos de Hardware são dispositivos simples. São motes que servem apenas o propósito de controlar a quantidade de comida que os animais ingerem e saber a sua localização. Comunicam por radiofrequência com os locais de alimentação e com outros sensores.

Os motes são modelos MicaZ cujas características estão descritas na tabela abaixo mais informação consultar o seguinte documento:

http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf

O modelo do rádio especificamente é o predefinido no TOSSIM, que é baseado no modelo CC2420 da “Chipcom Products from Texas Instruments”. Mais informações sobre o modelo podem ser encontradas no link que se segue onde são descritas as suas características:

<http://www.ti.com/lit/ds/symlink/cc2420.pdf>

Os **feeding spots** são igualmente simples, não têm qualquer memória apenas têm que comunicar com os sensor nodes. Os emissores de rádio emitem um sinal na frequência e segundo o protocolo que consegue ser lido pelos sensores.

Processor/Radio Board	MPR2400CA	Remarks
Processor Performance		
Program Flash Memory	128K bytes	
Measurement (Serial) Flash	512K bytes	> 100,000 Measurements
Configuration EEPROM	4K bytes	
Serial Communications	UART	0-3V transmission levels
Analog to Digital Converter	10 bit ADC	8 channel, 0-3V input
Other Interfaces	Digital I/O,I ² C,SPI	
Current Draw	8 mA	Active mode
	< 15 μ A	Sleep mode
RF Transceiver		
Frequency band ¹	2400 MHz to 2483.5 MHz	ISM band, programmable in 1 MHz steps
Transmit (TX) data rate	250 kbps	
RF power	-24 dBm to 0 dBm	
Receive Sensitivity	-90 dBm (min), -94 dBm (typ)	
Adjacent channel rejection	47 dB	+ 5 MHz channel spacing
	38 dB	- 5 MHz channel spacing
Outdoor Range	75 m to 100 m	1/2 wave dipole antenna, LOS
Indoor Range	20 m to 30 m	1/2 wave dipole antenna
Current Draw	19.7 mA	Receive mode
	11 mA	TX, -10 dBm
	14 mA	TX, -5 dBm
	17.4 mA	TX, 0 dBm
	20 μ A	Idle mode, voltage regular on
	1 μ A	Sleep mode, voltage regulator off
Electromechanical		
Battery	2X AA batteries	Attached pack
External Power	2.7 V - 3.3 V	Molex connector provided
User Interface	3 LEDs	Red, green and yellow
Size (in)	2.25 x 1.25 x 0.25	Excluding battery pack
(mm)	58 x 32 x 7	Excluding battery pack
Weight (oz)	0.7	Excluding batteries
(grams)	18	Excluding batteries
Expansion Connector	51-pin	All major I/O signals

Figura 1: Especificações do MicaZ

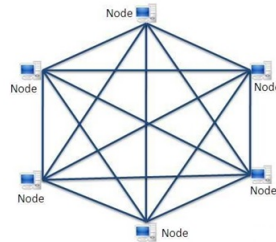
4 Arquitectura da Rede

Em termos de rede, os **sensor nodes** comunicam entre si para difundir a sua localização. Na base da rede está uma topologia que é carregada a partir de um ficheiro pré-existente. Neste ficheiro existem entradas que especificam ligações entre um nó origem e um nó destino e especificam a potência do sinal que chega.

Relacionando com a figura aqui mostrada, cada aresta na figura representa duas linhas no ficheiro da topologia da rede. A razão pela qual duas linhas e não apenas uma, reflecte o facto de cada linha modelar uma ligação unidireccional.

Os nós interagem uns com os outros para disseminar a informação da localização. A localização foi modelada com base em inteiros difundidos pelos emissores de rádio. Cada emissor envia um inteiro específico que no fundo identifica um local físico distinto. Os nós vão recebendo essa informação e guardam-na, assumindo que a sua localização é junto ao “emissor com o id: x”. Quando é feita uma interrogação ao sistema sobre a localização de um animal que não está dentro do raio de acção do laptop, o nó que recebe o pedido responde ao fazer ECO desse pedido para os nós vizinhos até o

pedido chegar ao destino. Antes de fazer ECO, o nó guarda o identificador do pedido para não voltar a fazer eco daquela mensagem.



5 Manual do Utilizador

Para colocar a aplicação a funcionar é necessário que a pasta AnimalHusbandry se encontre na pasta onde estão as outras aplicações que originalmente vêm com o TOSSIM. Seguidamente é preciso compila-la executando o comando “make micaz sim”. Finalmente para executar o código tem que se correr o ficheiro laptop.py executando “python laptop.py”. Irá surgir o seguinte ecrã:

```

joao@ubuntu: /opt/tinyos-2.1.2/apps/AnimalHusbandry
joao@ubuntu: /opt/tinyos-2.1.2/apps/AnimalHusbandry$ python laptop.py
#####
#--Aplicacoes para Sistemas Embebidos 2013/2014--#
#-----Welcome to Animal Husbandry-----#
#-----Grupo 5-----#
#---64766 - Guilherme de Sousa Aranha-----#
#---58508 - Joao Filipe Garrett Paixao Florencio-#
#####

Topology file [topo.txt]:
Number of animals [7]:
Number of feeding spots [7]:
Quantity of food in feeding spots[7]:

Menu
1: Get animals location
2: Get animals location (dissemination)
3: Get amount of consumed food
4: Change max daily food
5: Change amount of food in feeding spot
6: Check food left in feeding spots
7: Change animal position
8: Simulate animal approaching feeding spot
9: Simulate new day (resets Food for animals and feeding spots)
10: Quit
option: [

```

Figura 2: Menu inicial

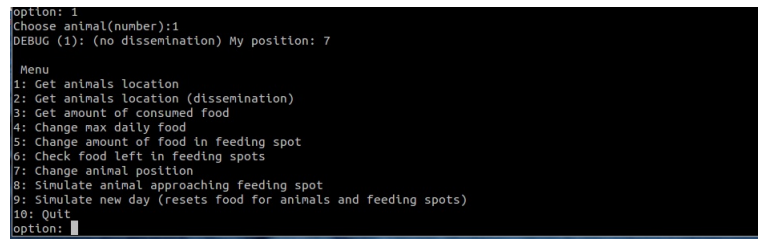
A aplicação inicia e pede ao utilizador para especificar o ficheiro onde se encontra a topologia da rede, qual o número de animais, qual o número de locais de alimentação e a que quantidade de alimento existe inicialmente nesses mesmos locais.

Se o utilizador não especificar estes parâmetros (apenas carregar Enter), a aplicação assume os valores default que estão entre parênteses rectos ([topo.txt] por exemplo).

Logo de seguida é apresentado o menu principal que mostra todas as funções do sistema. O utilizador deve pressionar a tecla (entre 1 e 10) correspondente à tarefa que pretende executar. Se, eventualmente pressionasse a tecla 1, iria surgir a informação que vem na figura abaixo.

Conforme podemos ver no exemplo da figura 3, a aplicação pede o identificador do animal e em seguida apresenta uma mensagem com a resposta (neste caso, posição 7).

As interações relativas às outras funções são basicamente semelhantes a esta, sendo que em algumas opções em vez ou para além do identificador do animal é também pedido o identificador do local



```
option: 1
Choose animal(number):1
DEBUG (1): (no dissemination) My position: 7

Menu
1: Get animals location
2: Get animals location (dissemination)
3: Get amount of consumed food
4: Change max daily food
5: Change amount of food in feeding spot
6: Check food left in feeding spots
7: Change animal position
8: Simulate animal approaching feeding spot
9: Simulate new day (resets food for animals and feeding spots)
10: Quit
option: █
```

Figura 3: Interação

de alimentação ou o identificador da posição (um inteiro).

Cada uma destas opções em termos específicos, executa respectivamente:

1. Consulta de localização previamente conhecida
2. Consulta de localização actual (com disseminação)
3. Consulta de alimento já ingerido
4. Alteração na quantidade máxima de alimento de um animal
5. Alteração na quantidade de alimento num local
6. Consulta de quantidade restante de alimento num local
7. Alteração da posição de um animal
8. Simulação de aproximação de animal de local de alimentação
9. Simulação de novo dia (reinicia contadores de alimento)
10. Sair da aplicação

6 Problemas encontrados no desenvolvimento do projecto

Ao longo do desenvolvimento do projecto deparamo-nos com alguns problemas, maioritariamente devido ao facto de não estarmos muito familiarizados com o TinyOS e com o TOSSIM. A forma como o código se estrutura e algumas particularidades da sua implementação suscitaram algumas dúvidas, e a documentação, especialmente do TOSSIM é muito reduzida.

O TOSSIM permite por exemplo a injeção de pacotes, mas que tenhamos conhecimento, não permite receber uma resposta, o que por vezes seria útil.

7 Especificações não cumpridas

Relativamente a especificações não cumpridas, tivemos algumas dúvidas na parte da disseminação. No enunciado é referido que poderiam haver vários computadores portáteis, ainda que na realidade fôssemos só testar com um (o simulador); por uma questão de facilidade de implementação, não consideramos essa hipótese na disseminação, em que utilizamos um identificador único gerado no simulador para identificar os pedidos.

No caso de haverem realmente vários computadores portáteis, seria necessário assegurar que os identificadores eram realmente únicos e não haveria computadores portáteis a gerar identificadores iguais. Retirando este aspecto, acreditamos que cumprimos com o pedido.

8 Conclusão

Para primeiro contacto com o desenvolvimento de aplicações para sistemas embebidos, acreditamos que foi uma experiência positiva. Inicialmente existiram os obstáculos comuns de qualquer curva de aprendizagem, mas que foram ultrapassados.

O TOSSIM como simulador é sem dúvida uma mais valia para o teste deste tipo de aplicações, dado que efectuar directamente o deployment destes equipamentos seria muito difícil para debug. Ainda assim, acreditamos que existam alguns pontos em que este possa evoluir como por exemplo na recepção de respostas RF e não só injeção.

A sua documentação também deveria ser mais formal, pois na realidade, pouco mais se encontra do que o conteúdo que está na wiki oficial, o que acaba por ser insuficiente para uma aprendizagem correcta e mais suave.