Bizkaiko Foru Aldundia
Diputación Foral de Bizkaia
Berrikuntza eta Ekonomi Sustapen Saila
Departamento de Innovación y Promoción Económica

Programa Ekinberri 2007

SmartMotes

Nodos inalámbricos de redes de sensores con inteligencia semántica

D4.3 Documentación del Programa

RFIDGlove

Tecnológico Fundación Deusto
Teknologikoa Deustu Fundazioa
robotiker tecnalia

RESUMEN

En este documento se explican las partes más relevantes del código implementado para desarrollar el dispositivo RFIDGlove.

Se comentan las diferentes decisiones que se han tomado, como el protocolo de comunicaciones que se utiliza, la cola implementada, los tipos de paquete que se envían entre las motas etc.

Se analiza por un lado las decisiones tomadas para desarrollar el programa del dispositivo RFIDGlove, y por otro la aplicación que se implementa en el ordenador al que se conecta la mota base y la comunicación entre ambas motas.

HISTORIAL DE CAMBIOS

Versión	Descripción	Autor	Fecha	Comentarios
V1.0	Primer borrador del documento	Leire Muguira	11/03/2008	
V1.1	Más información sobre el programa, y diagrama de clases añadido	Leire Muguira	01/04/2008	
V1.2	Imágenes añadidas	Leire Muguira	04/04/2008	

TABLA DE CONTENIDOS

1 INTRODUCCIÓN

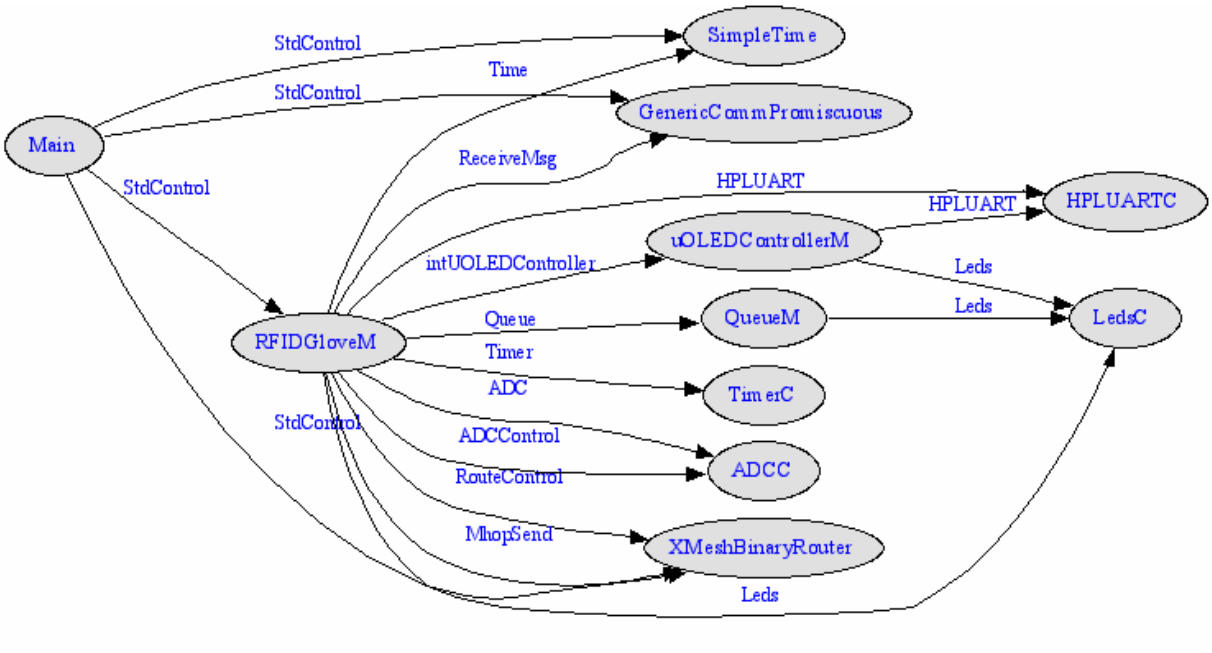
SmartGlove es un proyecto en el cual un guante dotado de un lector RFID y de una mota MICAz es capaz de comunicarse con otra mota base para que el sistema logístico conozca las actividades de manipulación de materiales que lleva a cabo el usuario y pueda optimizar el funcionamiento de la organización.

En este documento se explican las decisiones que se han tomado a la hora de desarrollar el código que se ha programado en las motas MICAz y el código java necesario en el ordenador al que se conecta la mota base.

2 PROGRAMA DE LA MOTA INTEGRADA EN EL SMARTGLOVE

La mota tiene que ser capaz de llevar a cabo diferentes funciones. Por un lado tiene que comunicarse con el lector RFID para capturar lo que éste ha leído. A continuación tiene que enviar el paquete con la información deseada a la mota base, a través de la red Mesh que estará formada por varias motas MICAz. Además, tiene que actuar en función de lo que la mota base le conteste. Por ejemplo, en la v.2 de SMARTGLOVE, la mota es capaz de visualizar en una pantalla uOLED de 4D Systems el resultado de la lectura, así como notificaciones que se envían desde la estación base en forma de texto.

El programa implementado en la mota integrada en el guante tiene la siguiente configuración:



Commponent RFIDGlove

Tal y como se muestra en la figura, el programa hace uso de diferentes componentes. A continuación se explican los detalles más importantes que hay que tener en cuenta.

2.1 Configuración del puerto UART:

El lector RFID que hemos empleado transmite los datos a 9600 baudios y las motas MICAz, por defecto trabajan a 57600 baudios. Es necesario reducir la tasa de transmisión de las motas.

Una vez que hemos instalado cywing, por defecto, dentro de la carpeta C:\Crossbow\cygwin\opt\MoteWorks\tos\platform\mica2 encontramos dos ficheros HPLUAR0M.nc y HPLUARTC.nc. Para poder modificar el *baud rate*, será necesario copiar ambos ficheros en el siguiente directorio:

C:\Crossbow\cygwin\opt\MoteWorks\tos\platform\mica2

Si nos fijamos detenidamente, vemos como en el fichero HPLUARTM.nc, dentro del método init(), existe una línea de código en el que se especifica la tasa de transmisión en baudios.

Call Setbaud(TOS_UART0_BAUDRATE);

Por defecto está configurado a 57600 baudios.

Para nuestra aplicación, nos interesa que funcione a 9600 baudios, de modo que sustituiremos la línea anterior por la siguiente:

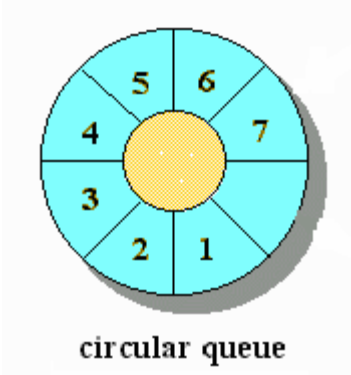
Call Setbaud(9600u);

Una vez realizadas estas modificaciones, lo único que hay que hacer es especificar en nuestro fichero RFIDGloveM.nc que vamos a utilizar el interfaz HPLUART e iniciarlo dentro del método init().

2.2 Implementación de la cola:

Puede haber situaciones en las que el RFIDGlove no pueda comunicarse con la mota base, como por ejemplo la falta de cobertura. En caso de que no se pueda realizar la comunicación, el guante tiene que ser capaz de encolar la información para transmitirlo más adelante, y perder la menor información posible.

La solución que se ha adoptado es la implementación de una cola fifo circular. Tal y como su definición específica, el primer elemento almacenado en la cola será el primero en ser enviado cuando sea posible y se tratará de forma circular.



El tamaño máximo de la cola está limitado por el hardwar. Las motas MICAz disponen de una memoria RAM de 4KB. Se han realizado diferentes pruebas para limitar el tamaño máximo de la cola que se puede implementar.

Por un lado se ha elaborado un programa que únicamente cree una cola del tamaño máximo posible. En cada posición de la cola se almacenará un paquete XDataMsg. Se ha comprobado que en este caso el tamaño máximo es de 189 posiciones. Al añadir el código necesario para desarrollar el RFIDGlove, el tamaño máximo de la cola se reduce a 97 posiciones. Cuando la cola se llena, no se almacenan más datos en ella, se descartan todos los datos recibidos.

- result_t **init**(void)
- uint8_t **enqueue**(XDataMsg msg)
- result_t **dequeue**(void)
- XDataMsg **get**(void)

Métodos del interfaz Queue

2.3 Control del reloj de la mota:

En esta aplicación nos interesa saber la hora en la que se ha realizado cada lectura. Para ello hemos utilizado el interfaz Time del componente SimpleTime.

Realizando la siguiente llamada capturamos los 32 bits de menor peso del tiempo en milisegundos binarios en formato *little endian*.

```
call Time.getLow32();
```

Para convertir milisegundos binarios en segundos hay que dividirlos entre 1024.

A lo largo del programa para cumplir los requisitos del RFIDGlove como por ejemplo el

parpadeo de los leds, se ha tenido que implementar un método *wait(uint16_t ds)* que realiza una llamada a TOSH_uwait. De este modo se puede, por ejemplo, encender un led, esperar un tiempo determinado y apagarlo.

Se ha comprobado que ejecutando el programa sin utilizar este método, el tiempo que transcurre entre la captura consecutiva de dos muestras es la esperada. En cambio, si se emplea el método *wait* se produce un retraso que depende del tiempo que pretendemos esperar en el wait.

Para entender mejor lo que sucede vamos a suponer que en el programa capturamos dos muestras con un intervalo de 5000ms entre ambos y que realizamos una única llamada al método wait para que espere 2000ms. Si el tiempo en que se realizó la primera muestra es 1000ms (desde el instante en que se encendió la mota), la segunda muestra debería dar un tiempo de 6000ms, pero en lugar de eso el tiempo que se obtiene es de 8000ms.

Para solucionar este problema se ha empleado una variable llamada offset, que se encarga de ir acumulando el sumatorio de todos los tiempos que se le pasan al método wait a lo largo de las sucesivas llamadas realizadas al método entre dos capturas consecutivas. De este modo, cuando se recupera el tiempo del reloj interno de la mota, se calcula el tiempo transcurrido desde la captura anterior y se le suma el offset antes de enviar el tiempo a la mota base.

3 COMUNICACIÓN ENTRE RFIDGLOVE Y MOTA BASE:

La comunicación entre el dispositivo RFIDGlove y la mota base se realiza a través del servicio de enrutado multihop XMesh. Para ello se han añadido los componentes GenericCommPromiscuous y MULTIHOPROUTER.

El protocolo de comunicaciones que se ha implementado para la comunicación entre RFIDGlove y la mota base es la siguiente:

- Cada vez que el guante recibe datos, los almacena en la cola y los envía a la mota base. Estos paquetes son de tipo XDataMsg.

- La mota base procesa el paquete recibido y envía un paquete de confirmación al RFIDGlove. Este paquete contiene el código asociado al tag leído por el lector RFID para que el RFIDGlove pueda visualizar la imagen correspondiente. Estos paquetes son de tipo DisplayImage.

- El RFIDGlove, cuando recibe el paquete que viene desde la mota base, comprueba si es la contestación al que él ha enviado anteriormente. En caso afirmativo, borra el paquete de la cola porque ya ha sido tratado.

- Si no recibe la contestación correspondiente, el paquete no se elimina de la cola y el RFIDGlove sigue enviando el paquete hasta que no reciba la contestación desde la base.

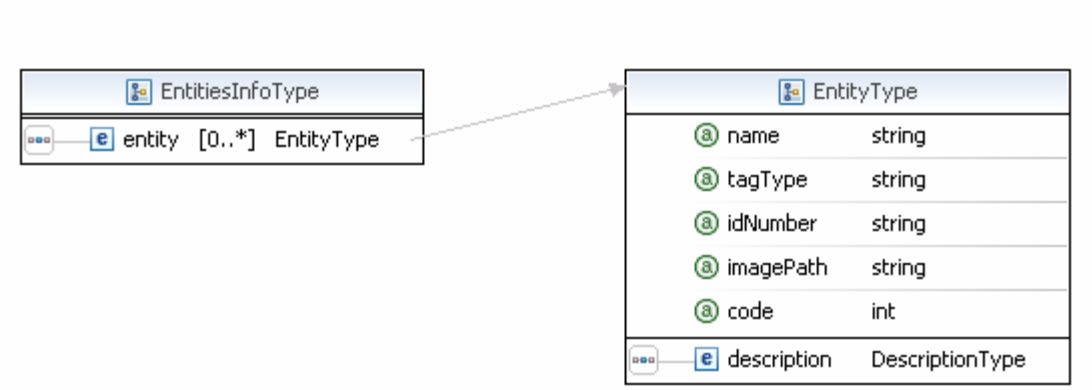
- Además la mota base también puede enviar otro tipo de paquetes que sirven para enviar texto desde la estación base hasta el RFIDGlove. Estos paquetes son de tipo DisplayText y se emplean cada vez que desde la estación base se solicita un envío de texto pulsando el botón de la GUI correspondiente.

3.1 Mota base:

El código implementado en las motas MICAz está programado en nesC sobre Tinyos.

La mota base, interactúa con java a través de las librerías tinyos.jar que se encargan de gestionar las operaciones de bajo nivel. Desde java se realiza el tratado de los paquetes recibidos.

Cuando la mota base recibe un paquete desde el RFIDGlove, coge el identificador de la tarjeta que se ha leído y comprueba en un fichero xml si reconoce esa tarjeta. Cada tarjeta además de un número de identificación (idNumber), tiene un código asociado (code).



Diseño del esquema xml

La mota base, coge el código asociado a la tarjeta del fichero xml, y envía un mensaje de contestación al RFIDGlove, con el código correspondiente, para que este pueda buscar la posición de memoria con la que se corresponde en el Uoled y lo pueda visualizar.

Por otro lado, para tener el historial de todos los paquetes enviados por el RFIDGlove desde que se puso en marcha la aplicación, éstos se van guardando en un fichero csv.

En el fichero csv se guarda la siguiente información:

- El identificador del tag leído
- La hora en la que se produjo la lectura

4 ANEXOS:

Formato de los paquetes que se utilizan en la comunicación entre el RFIDGlove y la mota base.

4.1 XDataMsg:

Éste es el tipo de paquete que envía el RFIDGlove a la mota base:

```
typedef struct XDataMsg {
        XMeshHeader xMeshHeader;
        union {
                RFIDGlove data;
         }xData;
} __attribute__ ((packed)) XDataMsg;
typedef struct RFIDGlove {
        uint8_t byte1;
        uint8_t byte2;
        uint8_t byte3;
        uint8_t byte4;
        uint8_t byte5;
        uint8_t byte6;
        uint8_t byte7;
        uint8_t byte8;
        uint8_t byte9;
```

uint8_t byte10;

uint8_t byte11;

uint32_t timelow; // it is in binary miliseconds (1024bonary miliseconds=1second) and in little endian format.

} __attribute__ ((packed)) RFIDGlove ;

4.2 DisplayImage:

Éste es el tipo de paquete que envía la mota base al RFIDGlove para confirmar que la lectura ha sido realizada correctamente y pueda visualizar la imagen correspondiente en la pantalla:

typedef struct **DisplayImage**{

XMeshHeader xMeshHeader;

struct{

int16_t code;//The corresponding code of the tag to know the display memory position

int8_t messageDisplTime;//The time to display the image on the screen

int32_t timelow;

}xData;

} __attribute__ ((packed)) DisplayImage;

4.3 DisplayText:

Éste es el tipo de paquete que envía la mota base al RFIDGlove cuando desde la mota base se solicita enviar texto al RFIDGlove pulsando el botón correspondiente en la GUI de la aplicación local:

typedef struct **DisplayText**{

```
        XMeshHeader xMeshHeader;
        struct{
                uint8_t messageDisplTime;
                char data[15];
        }xData;
} __attribute__ ((packed)) DisplayText;
```

4.4 XMeshHeader:

Todos los paquetes que se han mostrado anteriormente incluyen la siguiente cabecera porque la comunicación entre el dispositivo RFIDGlove y la mota base se realiza a través del servicio de enrutado multihop XMesh.

```
typedef struct XMeshHeader{
        uint8_t  board_id;
        uint8_t  packet_id;
        uint8_t  node_id;
        uint16_t parent;
}__attribute__ ((packed)) XMeshHeader;
```

4.5 Aplicación local del ordenador al que se conecta la mota base:

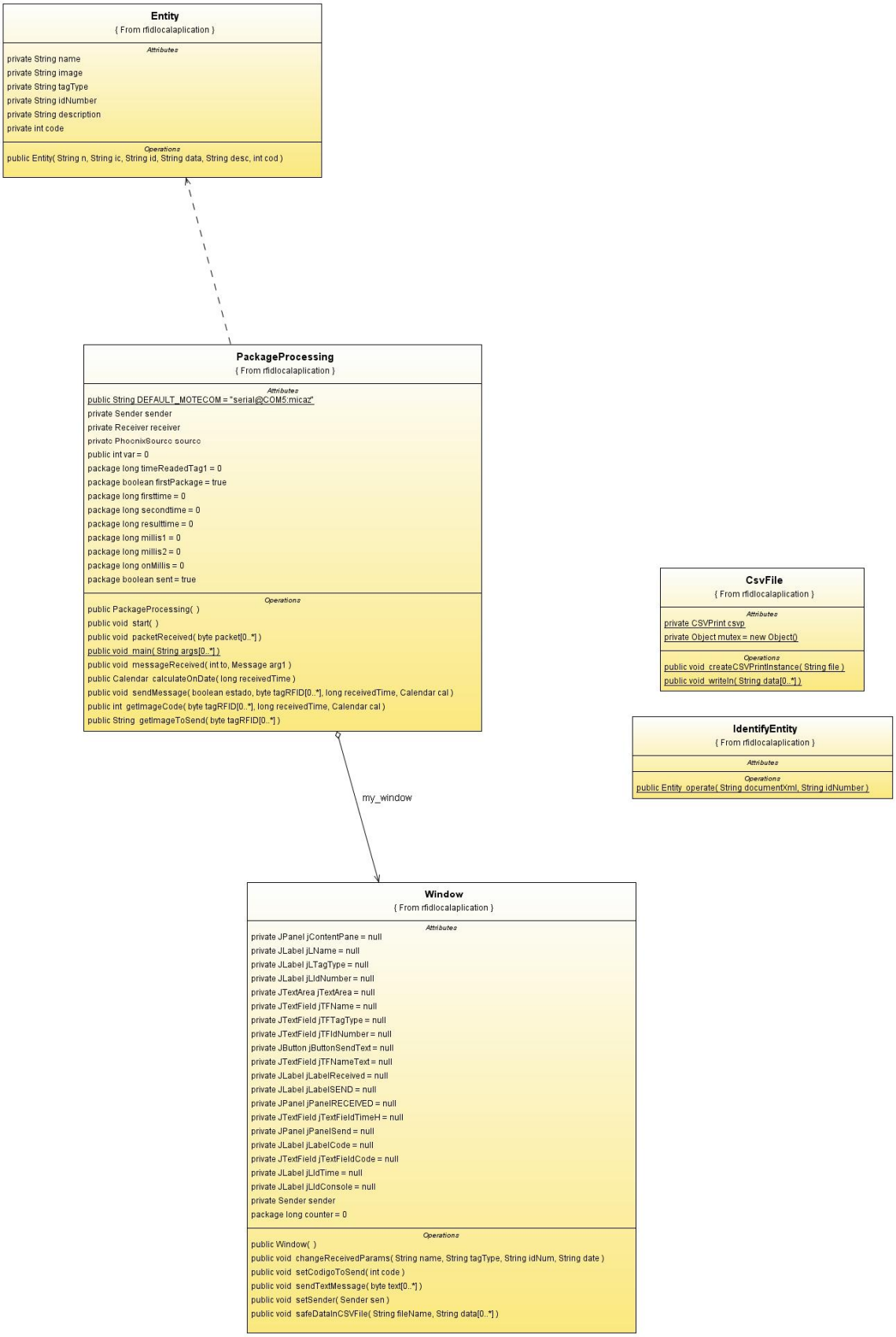
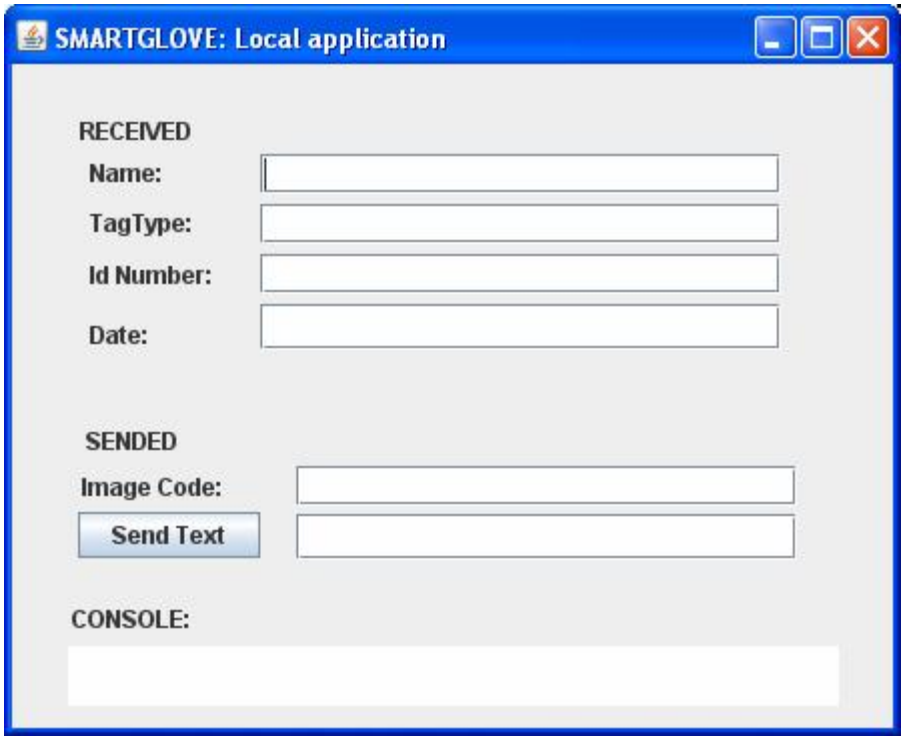


Diagrama de clases de la aplicación local



GUI de la aplicación local