Diseño de una Plataforma de Desarrollo para IoT

Acquarone Cesar[1]; Osio Jorge R[1,2]; Hromek Erik[1]; Salvatore Juan[1]; Montezanti Diego[1]; Morales D. Martín[1,3]

[1]*Universidad Nacional Arturo Jauretche, Instituto de Ingeniería,*
Florencio Varela, Av. Calchaquí 6200

[2] *Universidad Nacional de La Plata – Facultad de Ingeniería – CeTAD*

[3] *Universidad Tecnológica Nacional FRLP - Proyecto de Investigación:Codiseño Hw Sw para Aplicaciones en Tiempo Real*

josio@unaj.edu.ar, jsalvatore@unaj.edu.ar, dmontezanti@unaj.edu.ar, martin.morales@unaj.edu.ar

Abstract

El presente trabajo tiene como objetivo mostrar el diseño y desarrollo de una plataforma IoT (internet de las cosas) como una importante herramienta para la implementación de algoritmos mediante software embebido, que posibilita resolver una amplia variedad de problemáticas mediante la conectividad wifi y el acceso remoto.

La plataforma desarrollada consta de tres bloques importantes; un sistema embebido programable basado en un procesador Cortex M3, un módulo wifi esp8266 que provee la conectividad wifi y un software servidor programado en java mediante programación orientada a objetos que permite enviar y recibir datos, comandos y la medición de tiempos de transmisión de paquetes. Estas características le dan una gran potencialidad para el desarrollo de todo tipo de sistemas que interactúan con sensores, actuadores y requieren control y acceso de manera remota mediante IoT.

Como aplicación inicial se implementó un sistema de control automático de temperatura e iluminación en un sistema domótico.

Palabras Clave: *IoT, Programación de alto nivel, Software embebido, Plataforma de Desarrollo IoT.*

1. Introducción

La red LoRa es una tecnología inalámbrica Open Source de comunicación especialmente diseñada para el IoT, competidora de Sigfox. Estas redes están especialmente diseñadas para comunicar objetos conectados a larga distancia, con un coste muy ajustado y reduciendo enormemente el consumo energético de los dispositivos, permitiendo a éstos transmitir sus datos a mucha distancia sin depender de otras redes o conexiones. En esta propuesta se pretende mostrar el diseño de una plataforma cuyo funcionamiento es independiente de estas redes y cuyo objetivo principal es dar el puntapié inicial para el desarrollo de una nueva red personalizada para aplicaciones específicas.

Para la programación del sistema embebido, se utilizó el entorno integrado de desarrollo LPCXpresso que permite Programar un microcontrolador mediante el programador LPC-link [1], de la Figura 1 y la placa base, diseñadas en la UNAJ, que contiene los periféricos y sensores a incluir. La aplicación desarrollada utiliza los siguientes periféricos con las funcionalidades que se detallan continuación:

- **Display OLED de la placa base:** Se programó un menú que permite seleccionar opciones de configuración. También muestra los valores devueltos por los sensores e información para manejar la aplicación (por ejemplo, volver al inicio).
- **Sensores de la placa:** SE configuraron los sensores para medir la temperatura ambiente (en ºC) y tomar el nivel de iluminancia del ambiente (en Lux).
- **Módulo Wifi:** Se configuró el módulo para la conexión a la red, la implementación de un socket y la transmisión de datos.

En primera instancia se utilizaron las librerías en lenguaje C, desarrolladas para manejar el display oled mediante el protocolo I2C y mostrar mensajes. Además,

se programaron las funciones para la lectura de sensores de iluminación y temperatura mediante entradas digitales. Para la interfaz wifi, se configuró el módulo mediante el protocolo UART (serie asincrónico) y se realizó una librería que utiliza los comandos AT para la conexión de red y la comunicación TCP/IP.

La aplicación servidor fue programada en Java, mediante programación orientada a objetos en el entorno Eclipse. La misma permite configurar una conexión TCP mediante la configuración de un puerto y la dirección IP. Adicionalmente, permite medir los tiempos de transmisión de paquetes y la recepción de paquetes datos de interés. Dicha plataforma tiene características básicas, pero muy potentes para la implementación de todo tipo de algoritmos de procesamiento, posibilitando la obtención de resultados y la determinación de los parámetros de interés.

2. Metodología

La metodología para crear la plataforma consiste en la programación en C y modificación de las bibliotecas que permiten controlar los periféricos. Luego, se desarrolla un proyecto base que permita al usuario implementar la aplicación deseada que disponga del llamado a todos los periféricos. Para compilar el proyecto base y programar la placa se utilizará el software LPCxpresso que provee la posibilidad de hacer debugging en tiempo real y verificar el funcionamiento del sistema seleccionando la opción paso a paso o continua desde el software.

La metodología para crear la plataforma servidor, utiliza programación en java y programación orientada a objetos. Para esto se configuró el compilador java sobre eclipse y se aplicaron los conceptos de objetos para crear la ventana de comando y el botón para envío por tcp/ip

El método combina varias herramientas de desarrollo para la creación de la plataforma, a su vez, esta permitirá resolver todas las problemáticas de HW que pueden surgir al realizar una aplicación IoT.

Esta plataforma permite implementar una amplia variedad de algoritmos orientados a temáticas de eficiencia energética, telemedicina, procesamiento de imágenes, cómputo de altas prestaciones, entre otros. Mediante el método de programación de software embebido y la conexión mediante el protocolo TPC/IP.

3. Implementación de la Plataforma

El diseño de la plataforma consiste en desarrollar el conjunto de etapas del diagrama que se muestra en la Figura 1.

Para utilizar la plataforma se requiere programar el procesador cortex M3 mediante la placa que se muestra en la Figura 2. Para la programación del sistema microcontrolador que contiene el procesador se desarrolló un proyecto base que contiene las bibliotecas de funciones para controlar los periféricos de la plataforma (sensores de iluminación, movimiento y temperatura; acelerómetro; entrada de micrófono; módulo wifi: display oled y teclado).

Desde el punto de vista del servidor se deberán configurar las características de ip, máscara de subred y puerta de enlace.

3.1. Sistema Embebido

El sistema embebido se puede programar y depurar mediante el programador se muestra en la placa izquierda de la Figura 2.

El sistema embebido está formado por el Microcontrolador LPC1769 que se encuentran en la placa derecha de la figura 2. Este está formado por un Procesador Cortex M3, memoria y un conjunto de módulo que tienen las siguientes características:

- 64 kB de SRAM
- 512 kB de Flash
- 4 módulos UART
- 3 módulos I2C
- 1 módulo SPI
- 2 módulos SSP
- 2 módulos CAN
- 1 módulo PWM
- 1 módulo USB 2.0 Device/Host/OTG
- 1 módulo RTC (reloj de tiempo real)
- 1 módulo Ethernet

Mediante estos módulos se conectan los diferentes periféricos y sensores a utilizar en la plataforma. A continuación se describe la interfaz IoT que se comunica al LPC1769 mediante el módulo UART.

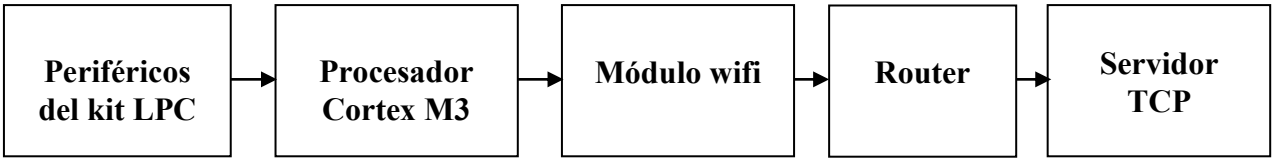

Figura 1. Diagrama en bloques de la plataforma

Además, se describirán los diferentes periféricos que forman parte de la plataforma; como el display, sensores, etc.

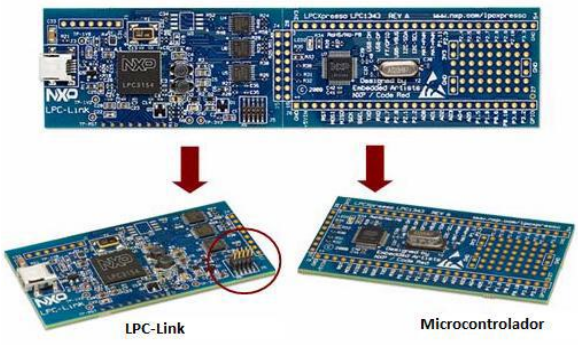

Figura 2. Procesador Cortex M3

3.2. Interfaz IoT

El módulo wifi ESP6288 que se muestra en la figura 3, se conecta al sistema embebido para posibilitar la comunicación TCP/IP con el servidor. Para que la plataforma sea versátil, el módulo se configuró de manera de posibilitar la configuración para la conexión con un router o con el servidor directamente. Para esto se utilizan comandos que listan las redes disponibles y permiten la configuración de usuario y contraseña de red (ver tabla 1).

Figura 3. Módulo wifi ESP6288

Adicionalmente, en la Tabla 1 se listan los comandos AT para iniciar una conexión TPC, enviar datos de manera transparente y cerrar la conexión TCP. Con este grupo de comandos se pueden realizar todas las tareas de conexión y transmisión de datos con el servidor.

Tabla 1. Comandos AT que soporta el ESP6288

COMANDOS AT MODULO WIFI	
COMANDOS	MODO ESTACION
AT+CWMODE=1<CR><	REDES DISPONIBLES
LF>	
AT+CWLAP<CR><LF>	CONECTARSE A TP-LIN_LAB
AT+CWJAP="TP-LINK_LAB","67851614"<CR><LF>	DHCP ESTACION HABILITADO
AT+CWDHCP=1,1	MUESTRA IP Y MAC
AT+CIFSR<CR><LF>	ABRE CONEXION TCP
AT+CIPSTART="TCP","192.168.1.101",6789<CR><LF>	MODO ESTACION
AT+CIPMODE=1<CR><LF>	ENVIO DE DATOS TRANSPARENTE
AT+CIPSEND<CR><LF>	**PASAJE DE DATOS**
AT+CIPCLOSE<CR><LF>	CIERRA CONEXION TCP
AT+CWQAP<CR><LF>	DESCONEXION DEL AP

3.3. Periféricos disponibles en el sistema embebido

La placa base del sistema embebido contiene un conjunto de periféricos, que permiten multiplicidad de aplicaciones para el sistema IoT. Entre los periféricos que contiene la plata base se dispone de un sensor de iluminación (fotoresistor), sensor de temperatura (lm335), acelerómetro, sensor de movimiento (Infrarrojo), relés actuadores, entrada de micrófono (electrect), interfaz serie asincrónica con la PC (uart), interfaz wifi, un display gráfico oled y un conjunto de pulsadores [6].

Como etapa de adquisición de la señal de voz se utilizó un micrófono tipo electret, un filtro de Butterworth de segundo orden y una etapa preamplificadora con ganancia 10 basada en un amplificador operacional. La señal de salida contiene una componente de continua de aproximadamente 1,6V con la finalidad de aprovechar todo el rango dinámico del conversor, el cual va desde 0 a 3,3V.

Parte de la configuración de los periféricos, se efectúa durante las rutinas de inicialización implementadas en el Firmware del proyecto base.

Para el periférico ADC (conversor analógico digital) se invoca a la función ADC_Init() de la capa driver, y su configuración resultante por defecto, establece:

- ADC_MAX_SAMPLE_RATE en 400kHz (frecuencia de muestreo)
- ADC_10BITS (tamaño de dato digitalizado de 10 bits)
- Modo burst deshabilitado (conversión continua)

- Para leer las muestras de la señal de voz se utiliza ADC_Read(1)

Para la lectura del sensor de temperatura e iluminación se usó otra entrada analógica al conversor AD, pero la temperatura se leyó desde el canal 2 y la iluminación desde el canal 3:

- Lectura del valor de temperatura en ADC_Read(2). La lectura es proporcional a un valor de temperatura que va de -10 a 40 C°.
- La lectura del valor de luminancia se realizó mediante el llamado a ADC_Read(3).

Respecto al display Oled, el mismo se controla mediante el protocol I2C desde el procesador cortex m3. Adicionalmente se desarrollaron funciones para la escritura de mensajes en el display.

- init_i2c(): llama a las funciones que inician las interfaces de comunicación I2C, mediante los módulos del microcontrolador I2C.
- Las funciones oled_init(), oled_putstring(), oled_clearscreen(), oled_fillrect(): se desarrollaron para controlar el display oled.

Para la interfaz con la PC se utilzó la interfaz uart mediante las funciones:

- Init_uart(baudrate): inicializa el módulo uart
- Uart_read(): lee el registro de datos recibidos en el módulo uart
- Uart_send(): envía datos por la interfaz uart

3.4. Programación del Sistema Servidor

El Programa servidor, fue desarrollado en java mediante programación orientada a objetos y provee la posibilidad de seleccionar el puerto a utilizar para iniciar la comunicación TCP y una ventana para mostrar los paquetes recibidos, (ver figura 4).
Se implementaron 4 archivos mediante en java:

- MySQLDB.java: implementa la base de datos que almacena los datos entrantes según la clasificación realizada
- MainSERVIDOR.java: implementa el servidor del socket mediante el puerto seleccionado.
- Conc.java: Clase que permite conectar con la base de datos.
- Principal.java: Llama a las clases definidas en los archivos anteriores para implementar el servidor TCP

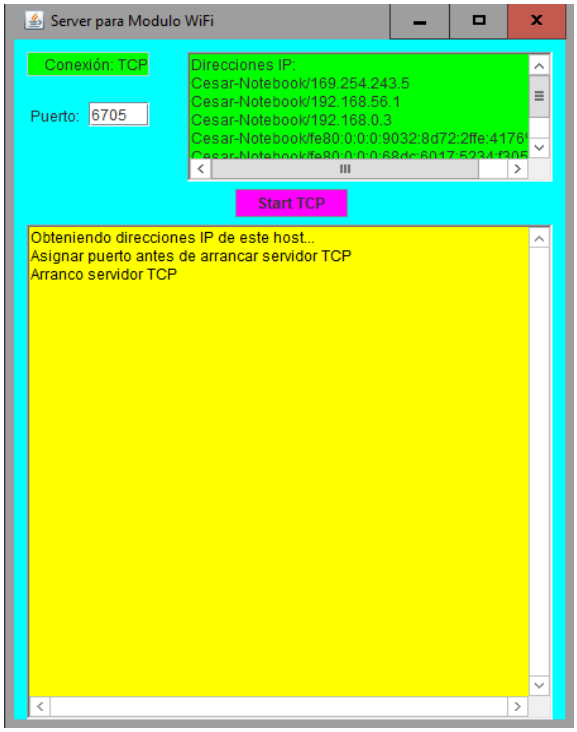

Figura 4. Aplicación Servidor

4. Resultados

Para las pruebas de funcionamiento de la plataforma se implementó una aplicación que lee datos de voz desde un micrófono conectado al canal 1 del conversor Analógico/Digital; además lee datos de iluminación desde el canal 2 y adicionalmente se leen valores de temperatura. Los datos leídos de los diferentes periféricos se guardan en un buffer de 32 bytes y son transmitidos al servidor. En cuanto a los resultados obtenidos, se pudo comprobar la correcta recepción de los datos desde el servidor y los tiempos de transferencia logrados en diferentes momentos (con distintos niveles de tráfico de red).

En cuanto a los tiempos de solicitud y recepción de paquetes se obtuvieron los resultados que se muestran en la tabla 2. Donde se considera que los tiempos son más que aceptables y varían en función del tráfico de la red. Se obtuvieron tiempos mínimos del orden de los 20 ms y máximos en el orden de 106ms. En función de estos datos se obtuvo una velocidad de transferencia promedio del orden de los 62ms.

Tabla 2. Tiempos de transferencia cliente servidor

Tamaño de paquetes (Bytes)	Tiempos TX/RX (ms)
32	88
32	106
32	.
32	.
32	49
32	20
32	35

En la Figura 5, se muestra la ventana de interfaz entre el sistema embebido y una PC, mediante puerto serie. Dicha interfaz permite al sistema IOT informar los datos de tiempos de transferencia, la correcta transmisión y recepción de datos y el estado actual de la comunicación.

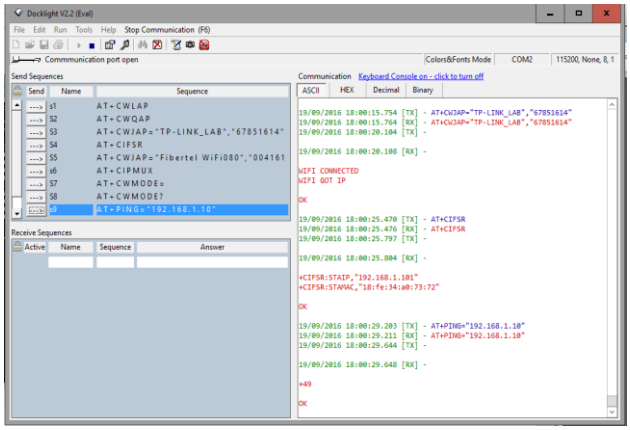

Figura 5. Interfaz sistema embebidos - PC

5. Conclusiones

El sistema implementado funcionó de acuerdo a los requerimientos y posibilitó comprobar el correcto funcionamiento de la plataforma IoT.

Aunque actualmente existen plataformas IoT de desarrollo, la potencialidad de esta plataforma se centra en la posibilidad de desarrollar un protocolo personalizado de IoT, debido a que las plataformas tradicionales se basan en Lora y Sigfox.

En cuanto a la tasa de transferencia promedio, se considera que 1967 Byte/seg es más que aceptable para los datos utilizados en las aplicaciones de IoT

Por último, se determinó que la plataforma es ideal para la implementación de sistemas basados en internet de las cosas; especialmente para aplicaciones de eficiencia energética, telemedicina, red de sensores y todo tipo de aplicación que utilice los periféricos disponibles en el sistema embebido.

Referencias

[1] NXP semiconductors, “LPC1769 - 32-bit ARM Cortex-M3 microcontroller”, rev. 9, Agosto 2012

[2] Embedded Artist, “LPCXpresso Base Board - User’s Guide”, rev. B, 2011

[3] NXP semiconductors, “LPCXpresso - Getting started with NXP LPCXpresso”, rev. 11, Diciembre 2011.

[4] ARM DUI, “ARM® DS-5 - Using Eclipse”, v5.3, 2010