ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE
TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

DETECTOR DE MOVIMIENTO INALÁMBRICO CON DETERMINACIÓN DE VELOCIDAD Y DISTANCIA.

AUTOR: Álvaro Jiménez Fernández
DIRECTORES: Francisco J. Fernández Luque
Juan F. Zapata Pérez
Junio de 2016

Agradecimientos

El trabajo que conforma este proyecto fin de carrera llevado a cabo durante, aproximadamente, año y medio ha sido fruto de un esfuerzo que no habría sido posible sin los apoyos recibidos a los que dedico mi gratitud en estas líneas.

Para empezar, sin duda alguna, mi familia. Gracias a mis padres y mi hermana por el apoyo sin condiciones recibido y lo que les he hecho sufrir cada vez que me enfrentaba a algún reto durante toda mi etapa como estudiante. Silvia, que se convirtió, hace más de cuatro años, en mi acompañante en todo este viaje y que es uno de los pilares fundamentales de mi vida.
Cómo olvidar a mis amigos de la infancia, cuya compañía en los ratos más difíciles me ayudaba a superar mis preocupaciones. Mis compañeros de carrera y ahora algunos de ellos compañeros en mi nueva etapa profesional, que hacen que el día a día en el trabajo sea más ameno y divertido.

Por último, pero no menos importante, agradecer a mis mentores en este proyecto su compromiso y su paciencia conmigo. A Paco por todas esas tardes que le he robado de poder estar con sus niñas y cuya ayuda y guía ha sido fundamental a lo largo de este tiempo. A Juan, cuyos sabios consejos me han ayudado a mejorar mucho, no sólo en este trabajo, sino en todos los venideros.
Una vez más, GRACIAS A TODOS.

I

Índice de contenido

II

Índice de figuras

II

Índice de tablas

Uno

Introducción

1.1 Motivaciones. Marco del proyecto

Desde finales de los años 90, debido al aumento de la esperanza de vida y a la reducción de la natalidad, las sociedades occidentales se enfrentan a un nuevo reto: el envejecimiento de la población. Uno de los problemas que plantea esta realidad es el incremento de personas de avanzada edad que viven solas en casa. Éstas desean conservar su independencia creyendo, en muchos casos, que pueden realizar tareas que han hecho toda su vida sin riesgo alguno y negándose a recibir ayuda del exterior. Tristemente es frecuente encontrar noticias de ancianos que fallecen solos en sus hogares y que su ausencia no es percibida hasta varios días después por vecinos o familiares.

Con este problema en mente se propuso crear un sistema inteligente y no intrusivo, basado en una red inalámbrica de sensores, que sea capaz de detectar situaciones o acciones de emergencia y alertar para que se atienda al anciano si fuera necesario. Una red de sensores es una infraestructura formada por dispositivos sensores, de procesado y de comunicación que proporciona al administrador la capacidad de configurar, observar y reaccionar a eventos sucedidos en un entorno específico. El proyecto Pro-DIA [1] (PROtotipo de Dispositivo Inteligente de Alerta) comenzó a desarrollar dispositivos que detectasen por sí mismos pautas [2] de conducta de sus usuarios y las utilizaran para llevar a cabo acciones de alerta cuando se produjesen alteraciones significativas de esas pautas.

Para poder detectar una posible situación de emergencia es necesario tener algún conocimiento de lo que sucede en la casa en cada momento pero esta vigilancia no debe interferir en la privacidad del atendido, por ello se eliminó desde el principio la posibilidad de la instalación de videocámaras o micrófonos. Se decidió instalar tres tipos de sensores: movimiento (infrarrojos), presión (contacto) y apertura de puerta (magnético). El sensor de movimiento que se utilizó en el prototipo era un sensor de

movimiento digital que informaba del grado de actividad que hay en la casa y localizaba al usuario pero no se podían obtener datos tales como la velocidad o distancia de la persona, mediante el sensor de presión se podía detectar cuando, el atendido, se encuentra acostado o sentado en su sillón y mediante el sensor de la puerta se detectaba cuando el usuario salía de casa y, por tanto, ya no era necesario que se activasen las alarmas.

En resumen, la idea principal es que con un sistema mínimo de sensores y una pequeña unidad de procesamiento que desempeña el tratamiento inteligente de la información obtenida de esos dispositivos sensores debería de poder mantener asistido al anciano en su casa de forma no intrusiva.

1.2 Planteamiento del Proyecto Fin de Carrera

El objetivo de este trabajo es el desarrollo de un sensor analógico que complemente la información obtenida por la capa de captación sensorial ya implementada y analizar los datos que se obtengan en tiempo real. Este nuevo sensor tomará muestras cuando el sensor de movimiento digital se active, permitiendo así un ahorro de energía y el aumento de la vida de las baterías conectadas al nodo, y nos permitirá conocer datos adicionales acerca de la situación del atendido tales como la velocidad y posición del mismo.

Al añadir este nuevo sensor se plantea un problema. Y es que para procesar los datos que se obtienen se utilizarán funciones como la Transformada de Fourier que necesitan más memoria RAM de la que incluye el microcontrolador de Atmel, utilizado tanto para la versión anterior del prototipo como para la que se usará en este proyecto, por lo que será necesario incluir en el nuevo diseño de la placa una memoria RAM externa. Bajo estas condiciones las fases de desarrollo del proyecto pueden ser resumidas de la siguiente forma:

1. Estudio del Estado del Arte en redes de sensores inalámbricos.

2. Estudio del lenguaje de programación NesC, del sistema operativo TinyOS, del software de diseño de circuitos Altium Designer y comprensión del código utilizado en la última versión del nodo sensor (*mote*).

3. Adaptación de la aplicación para incluir al nuevo sensor analógico.

4. Diseño del nuevo prototipo de *mote* en Altium, su posterior fabricación en la microfresadora y soldado de los componentes.

5. Implementación del código que trate los datos obtenidos y activación de la RAM externa.

6. Validación y depuración de los resultados obtenidos mediante el análisis de logs de las tramas recibidas.

1.3 Estructura de la memoria

La estructura de la memoria del proyecto fin de carrera es la siguiente: en este primer capítulo se indican las motivaciones y el planteamiento del trabajo. En el siguiente capítulo se desarrolla el Estado del Arte en redes de sensores inalámbricas, donde se comentan las diferentes arquitecturas, protocolos, plataformas y aplicaciones. A continuación, se presentan las herramientas que permiten el desarrollo e implementación del detector de movimiento: TinyOS, Altium Designer y Protomat C60. En el capítulo cuarto, se procede a describir el desarrollo del detector de movimiento inalámbrico. En el quinto se resumen los resultados que se han ido obteniendo durante el desarrollo así como una serie de conclusiones y propuestas de líneas de investigación que podrían llevarse en un futuro.

Dos

Estado del Arte en Redes de Sensores Inalámbricos (Wireless Sensor Networks,WSN)

2.1 Introducción a las WNS

Las redes de sensores inalámbricas están formadas por un grupo de dispositivos con ciertas capacidades sensoriales, de cómputo y de comunicación sin cables que permite la formación de redes ad-hoc sin infraestructura física preestablecida ni administración central. Este tipo de redes es un concepto relativamente nuevo en el campo de la adquisición y tratamiento de datos con múltiples aplicaciones en distintos ámbitos como son los entornos industriales, domótica, entornos militares o detección ambiental. Se caracterizan por su facilidad de despliegue y por ser autoconfigurables, pudiendo convertirse en todo momento en emisor, receptor, ofrecer servicios de encaminamiento entre nodos sin visión directa, así como registrar datos referentes a los sensores locales de cada nodo. Otra de sus características es su gestión eficiente de la energía, que les permite obtener una alta tasa de autonomía que las hacen plenamente operativas.

Tanto la miniaturización, cada vez mayor, de los componentes electrónicos como su precio, cada vez más económico, nos proporcionan la habilidad de combinar la captación de datos, cómputo y comunicación en un único y pequeño dispositivo. A través de avanzados protocolos de red, estos dispositivos forman un mar de nodos interconectados que extiende los límites del ciberespacio al mundo físico. Tal como el agua en un barco que se hunde busca todos los caminos hasta llenar los resquicios de la nave, la red de sensores buscará y usará cualquier posible ruta de comunicación realizando los datos saltos de un nodo a otro hasta encontrar su destino. Mientras que las capacidades de un solo dispositivo son muy pequeñas, el conjunto de cientos de ellos podría ofrecer un potencial que hasta ahora era difícil de imaginar.

2 Estado del Arte

La ventaja de las WSN radica en la facilidad de instalar un gran número de pequeños nodos sensores y que estos son autoconfigurables, pudiendo convertirse en todo momento en emisor, receptor, ofrecer servicios de encaminamiento entre nodos sin visión directa, así como registrar datos referentes a los sensores locales de cada nodo. Otra de sus características es su gestión eficiente de la energía, que les permite obtener una alta tasa de autonomía que las hacen plenamente operativas.

La idea general de estas redes es repartir aleatoriamente estos nodos en un territorio grande, el cual será observado por los nodos hasta que sus recursos energéticos se agoten. En contraste, las redes de sensores cableadas no son nuevas y sus funciones incluyen medir niveles de temperatura, líquido, humedad etc. Muchos sensores en fábricas o coches, por ejemplo, tienen su propia red que se conecta con un ordenador o una caja de controles a través de un cable y, al detectar una anomalía, envían un aviso a la caja de controles. La diferencia entre los sensores que todos conocemos y la nueva generación de redes de sensores inalámbricas es que estos últimos son inteligentes, es decir, capaces de poner en marcha una acción según la información que vayan acumulando, y no están limitados geográficamente por un cable fijo.

Los nuevos avances en la fabricación y miniaturización de circuitos integrados de radio y microcontroladores, y los desarrollos experimentados en la implementación de nuevos programas informáticos que permiten sustentar este tipo de sistemas, están logrando eliminar los enlaces cableados de los nodos de las redes de sensores, multiplicando así su potencial. Como ejemplo de nodo sensor se muestra en la Figura 2.1 uno de los dispositivos con los que se trabajará en este PFC. El ámbito de aplicación de este tipo de sistemas, como se verá, es muy amplio, por nombrar unos pocos podemos destacar: monitorización de entornos naturales, aplicaciones para defensa, aplicaciones médicas en observación de pacientes, teleasistencia, etc. El principal motivo de su éxito se debe a sus especiales características. Tal vez, una muy importante es que a los nodos de las redes se les imponen unas restricciones de consumo severas. El motivo de la imposición de estas restricciones es la necesidad de que los nodos sean capaces de operar, por sí mismos, durante periodos largos de tiempo, en lugares donde las fuentes de alimentación son si no inexistentes, de baja potencia. El tamaño es otra restricción que cada vez se hace más necesaria para la mayoría de las aplicaciones, de manera que las tarjetas o nodos que forman las redes de sensores sean cada vez de menor tamaño.

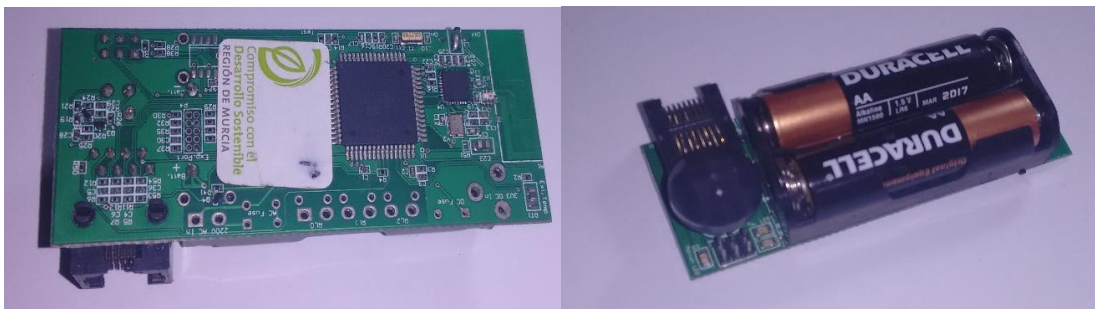


Figura 2.1: Ejemplo de nodo sensor DAS220.4

Además de la plataforma hardware que se ha comentado, también se desarrolló un software específico para las aplicaciones de las WSN. La universidad de Berkeley e Intel crearon un sistema operativo llamado TinyOS, una plataforma específica para este tipo de sistemas que tiene en cuenta las restricciones de los nodos. La característica principal reside en que al ser modular resulta ideal para instalarse en sistemas con restricciones de memoria.

Junto al sistema operativo se desarrolló también un lenguaje de programación, llamado NesC, de sintaxis muy parecida a C, basado en componentes, y a partir del cual se rediseñó una primera versión de TinyOS de modo que actualmente está íntegramente implementado sobre NesC. Tanto NesC como TinyOS están basados en componentes e interfaces bidireccionales. Además, actualmente, Berkeley e Intel han desarrollado diversas aplicaciones a modo de ejemplo, simuladores de ejecución, y varias universidades internacionales están dedicando esfuerzos al desarrollo de aplicaciones usando esta emergente tecnología.

Existen otras empresas que son proveedores de esta tecnología, el mayor de estos es *Crossbow. Technology*, que han desarrollado redes de sensores inalámbricos a gran escala para su uso comercial. Las últimas investigaciones apuntan hacia una eventual proliferación de redes de sensores inteligentes, redes que recogerán enormes cantidades de información hasta ahora nunca registrada.

2.2 Características de una WSN

Los recientes avances en microelectrónica, comunicación por radio y electrónica digital han permitido el desarrollo de nodos sensores de bajo coste, reducido tamaño, bajo consumo y que se comunican de forma inalámbrica.

Como ya se ha comentado, las WSN se componen de miles de dispositivos pequeños, autónomos, distribuidos geográficamente, llamados nodos sensores con capacidad de cómputo, almacenamiento y comunicación en una red conectada sin cables, e instalados alrededor de un fenómeno objeto para monitorizarlo. Una vez se produzcan eventos, toma de medidas o cualquier actividad programada con el fenómeno en cuestión los nodos enviaran información a través de la red, hasta llegar a un sistema central de control que recogerá los datos y los evaluara, ejecutando las acciones pertinentes en comunicación con otros sistemas o en la propia red de sensores. Se pueden distinguir una serie de elementos que componen de forma general una WSN, tal y como se observa en la Figura 2.2:

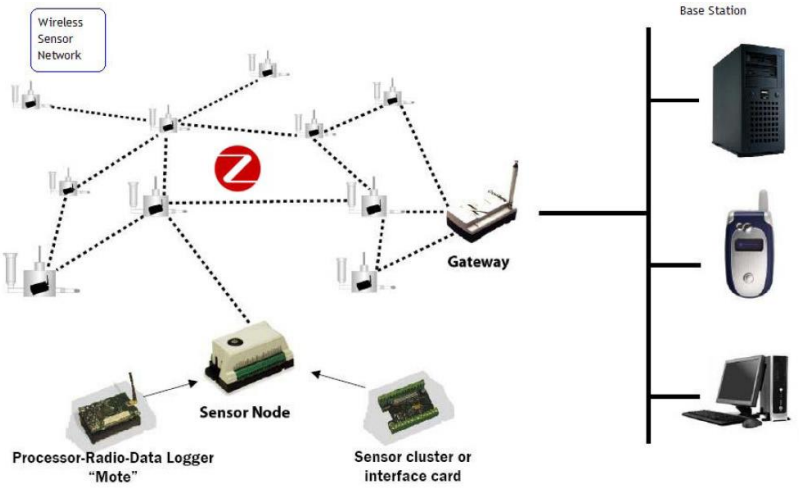


Figura 2.2: Elementos de una WSN

- Transductores: De distinta naturaleza y tecnología, toman del medio la información y la convierten en señales eléctricas.
- Nodos sensores: Son los procesadores de radio, que toman los datos del sensor a través de sus puertas de datos y envían la información a la estación base.
- Pasarelas o *Gateways*: Elementos para la interconexión entre la red de sensores y una red TCP/IP.
- Estaciones base: Recolector de datos basado en un ordenador común o sistema integrado.
- Red inalámbrica: Típicamente basada en el estándar IEEE802.15.4 (*Zig-Bee*).

Las redes de sensores tienen una serie de características propias y otras adaptadas de las redes Ad-Hoc:

- *Topología Dinámica:* En una red de sensores, la topología siempre es cambiante y éstos tienen que adaptarse para poder comunicar nuevos datos adquiridos.

- *Variabilidad del canal:* El canal radio es un canal muy variable en el que existen una serie de fenómenos como pueden ser la atenuación, desvanecimientos rápidos, desvanecimientos lentos e interferencias que puede producir errores en los datos.

- *Ausencia de infraestructura de red:* Una red de sensores no tiene necesidad alguna de infraestructura para poder operar, ya que sus nodos pueden actuar de emisores, receptores o enrutadores de la información. Sin embargo, hay que destacar, en el concepto de red de sensores, la figura del nodo recolector (también denominados sumidero o *sink node*), que es el nodo que recolecta la información y por el cual se recoge la información generada normalmente en tiempo discreto. Esta información generalmente es adquirida por un ordenador conectado a este nodo y es sobre el ordenador que recae la posibilidad de transmitir los datos por tecnologías inalámbricas o cableadas según sea el caso.

- *Tolerancia a errores:* Un dispositivo sensor dentro de una red de sensores tiene que ser capaz de seguir funcionando a pesar de tener errores en el sistema propio.

- *Comunicaciones multisalto o broadcast:* En aplicaciones de redes de sensores siempre es característico el uso de algún protocolo que permita comunicaciones multi-hop, léase AODV, DSDV, EWMA u otras, aunque también es muy común utilizar mensajería basada en broadcast.

- *Consumo energético:* Es uno de los factores más sensibles debido a que tienen que conjugar autonomía con capacidad de proceso, ya que actualmente cuentan con una unidad de energía limitada. Un nodo sensor tiene que contar con un procesador de consumo ultra bajo así como de un transceptor radio con la misma característica, a esto hay que agregar un software que también conjugue esta característica haciendo el consumo aún más restrictivo.

- *Limitaciones hardware:* Para poder conseguir un consumo ajustado, se hace indispensable que el hardware sea lo más sencillo posible, así como su transceptor radio, esto nos deja una capacidad de proceso limitada.

- *Costes de producción:* Dada la naturaleza de una red de sensores, el número de nodos debe ser muy elevado para poder obtener datos con fiabilidad. Los nodos sensores, una vez definida su aplicación, resultan económicos si son fabricados en grandes cantidades.

2.3 Requisitos para una WSN

Para que una red pueda funcionar de acuerdo con las anteriores características hay una serie de condiciones que se deben tener en cuenta. Estos, que se detallan a continuación, son los requisitos no funcionales del sistema:

- *Eficiencia energética:* Es uno de los asuntos más importantes en redes de sensores inalámbricas. Cuanto más se consiga reducir el consumo de un nodo mayor será el tiempo durante el cual pueda operar y, por tanto, mayor tiempo de vida tendrá la red. La aplicación tiene la capacidad de bajar este consumo de potencia restringiendo el uso de la CPU y la radio FM. Esto se consigue desactivándolos cuando no se utilizan y, sobre todo, disminuyendo el número de mensajes que generan y retransmiten los nodos.

- *Autoorganización:* Es fundamental que los nodos sean capaces de formar la topología deseada sin ayuda del exterior de la red. Este proceso no sólo debe

ejecutarse cuando la red comienza su funcionamiento, sino que debe permitir que en cada momento la red se adapte a los cambios que pueda haber en ella.

- *Escalabilidad:* Puesto que las aplicaciones van creciendo con el tiempo y el despliegue de la red es progresivo, es necesario que la solución elegida para la red permita su crecimiento sin que las prestaciones caigan drásticamente.

- *Tolerancia a fallos:* Los sensores son dispositivos propensos a fallar. Por todos los medios se debe evitar que un fallo en un nodo individual provoque el mal funcionamiento del conjunto de la red.

- *Tiempo real:* Los datos llegan a su destino con cierto retraso. Pero algunos datos deben entregarse dentro de un intervalo de tiempo conocido. Pasado éste dejan de ser válidos, como puede pasar con datos que impliquen una reacción inmediata del sistema, o se pueden originar problemas serios como ocurriría si se ignora una alarma crítica. En caso de que una aplicación tenga estas restricciones debe tomar las medidas que garanticen la llegada a tiempo de los datos.

- *Seguridad:* Las comunicaciones inalámbricas viajan por un medio fácilmente accesible a personas ajenas a la red de sensores. Se deben establecer mecanismos que permitan tanto proteger los datos de estos intrusos, como protegerse de los datos que estos puedan inyectar en la red.

Dependiendo de la función que tenga nuestra aplicación algunos de los requisitos cobrarán mayor importancia. Por ejemplo, en un parque natural que se desee controlar a los animales, la autoorganización y la eficiencia energética serán fundamentales tanto por el tiempo que puede estar un animal sin ser visto como por el cambio constante de posición de éste. No sería así en el caso de la domótica dónde los nodos permanecerán estáticos la mayor parte de su vida útil.

2.4 Arquitectura software y hardware de un nodo sensor

El elemento fundamental en una red de sensores es el nodo sensor o mote. Existen variadas versiones de nodo sensor sin embargo todos deben incluir los siguientes componentes: microcontrolador, radio, memoria RAM y Flash (que puede estar integrada en el microcontrolador) y fuente de alimentación (que suelen ser baterías). En la Figura 2.3 se muestran los componentes hardware y software que conforman un nodo sensor.

Ciertos diseños de motes incluyen sensores integrados mientras que otros incorporan un conector de expansión al que se puede conectar una tarjeta con sensores a conveniencia, incrementando así la versatilidad a costa de incrementar el precio del producto final. La versión modular es la versión preferible para las primeras

fases de prototipado de un sistema, mientras que al finalizar el proyecto se tenderá a diseñar y fabricar un modelo integrado. Éste incorporará los sensores y las prestaciones cuya definición hayan sido el resultado de esa primera fase de prototipado.

Otra característica que define y diferencia a un nodo sensor es la tecnología de comunicaciones que emplea. Prácticamente la totalidad operan en RF, destacando una amplia mayoría que lo hace en la banda de 2.4 GHz, al tratarse ésta de una banda ISM globalizada.

En toda red de sensores debe existir un sumidero de datos, estación base o *sink*. Éste suele ser un dispositivo muy similar a un nodo sensor, con la característica de ofrecer conectividad hacia el exterior de la red, bien hacia una puerta de enlace que encamine los datos hacia otra red como Internet, o bien hacia un equipo con una potencia de cálculo y almacenamiento superiores a las de un nodo sensor donde se procesará la información recabada por la red. Este nodo sumidero también suele actuar como coordinador de la red, es decir, genera de forma dinámica las rutas multi-hop a partir de la información recabada de los sensores acerca de su vecindad. Debido a esta funcionalidad adicional, el nodo sumidero suele ser conocido como estación base de la red de sensores.

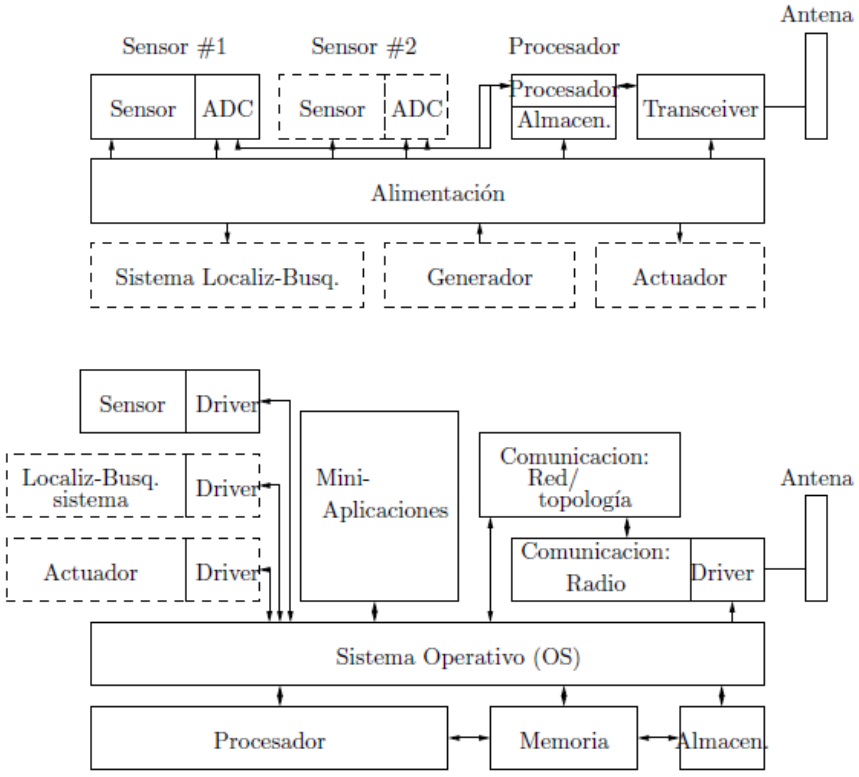


Figura 2.3: Componentes hardware y software de un nodo sensor.

En el caso de los motes con bus de expansión (que no integran sensores), la estación base está típicamente integrada por uno nodo sensor conectado a una tarjeta

de comunicaciones, en lugar de a una tarjeta sensora. Existen diferentes versiones de tarjetas de comunicaciones; resultando las más comunes las que operan mediante los protocolos RS-232, puerto serie virtual sobre USB (VSP over USB) o Ethernet. Esta última está especialmente indicada cuando los datos van a ser encaminados hacia redes IP para su recepción y tratamiento remotos. Por otro lado, las dos primeras opciones son las más adecuadas cuando los datos son tratados en local por un sistema tipo PC. No obstante, resulta evidente que cualquier combinación es válida.

2.5 Arquitecturas de las WSN

Tomando como elementos principales de la red a los nodos sensores, las pasarelas (*gateways*) y las estaciones base, podemos distinguir dos tipos principales de arquitecturas: centralizada y distribuida.

2.5.1 Arquitectura Centralizada

En este tipo de arquitectura los nodos de una red que estudian un fenómeno enviarán sus datos directamente a la pasarela más cercana, que dirige el tráfico de esa red como se muestra en la Figura 2.4. Si se tiene en cuenta que el ciclo de vida de un nodo consiste en despertarse, adquirir, procesar, transmitir y dormirse, y que cada vez que se transmita un mensaje irá a la pasarela, se están produciendo dos grandes problemas en la red. El primero, es que se generará un gran cuello de botella de mensajes no procesados en las pasarelas; y segundo, y no menos importante, se generará un mayor consumo de energía por el reenvío de los mensajes. Como resultado, el tiempo de vida de la red será relativamente corto.

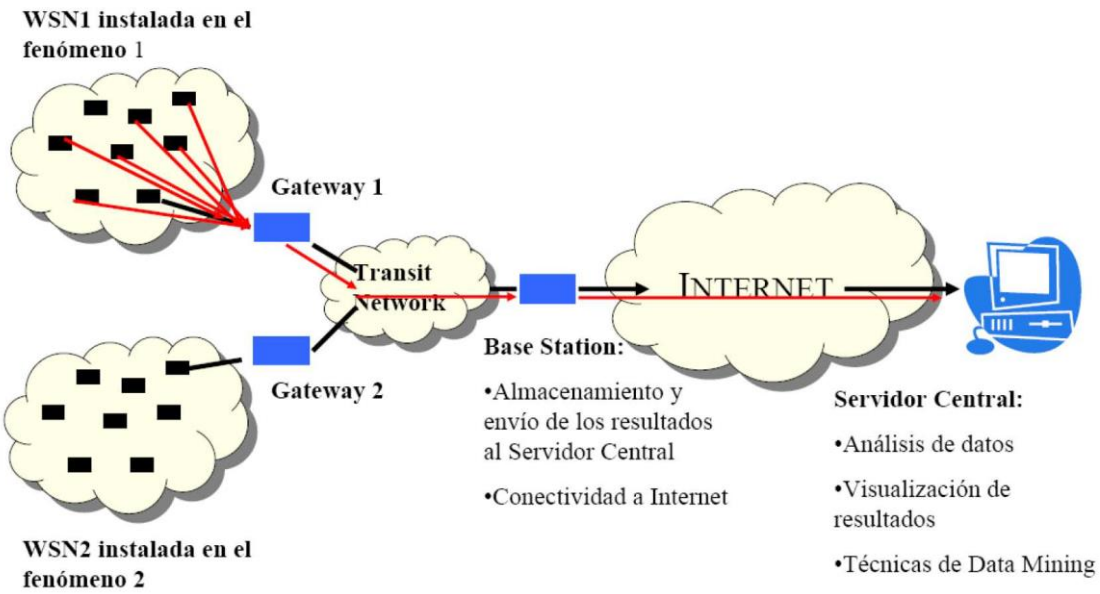


Figura 2.4: WSN de arquitectura centralizada.

2.5.2 Arquitectura Distribuida

Dada la naturaleza intrínseca de las redes de sensores, normalmente se tiende a un tipo de arquitectura que permita computación distribuida. De hecho, las WSN son redes basadas en la cooperación de sus nodos. Los nodos sensores se comunican entre nodos vecinos y cooperan entre ellos, ejecutando algoritmos distribuidos para obtener una única respuesta global que un nodo (*cluster head*) o estación base enviará a través de su pasarela. La Figura 2.5 muestra una típica red distribuida. Operando de esta forma se evitan los problemas de cuellos de botella y mayor consumo energético que surgían en la arquitectura centralizada.

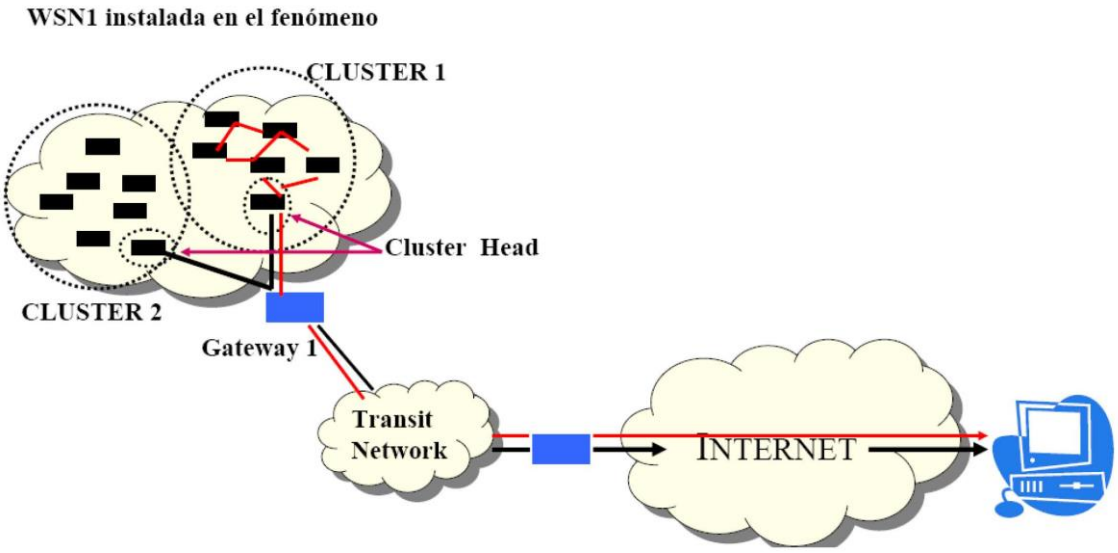


Figura 2.5: WSN de arquitectura distribuida.

2.6 Protocolos de las WSN

Las WSN utilizan una comunicación inalámbrica y, obviamente, sus protocolos de comunicación son diferentes de las redes cableadas. También son diferentes de las tradicionales redes inalámbricas como las redes celulares, las redes móviles ad hoc (*mobile ad-hoc networks*, MANETS). En las WSN, el objetivo principal es optimizar el rendimiento y el retardo. Aunque las MANETS comparten las características de desarrollo ad hoc y autoconfiguración de los nodos, el consumo de potencia no es una prioridad. Además, los nodos sensores son frecuentemente expuestos a extremas condiciones ambientales, haciéndolos propensos a frecuentes fallos en los nodos. Esto conlleva unas restricciones estrictas en las WSN, no como en las otras redes. A continuación se describen los protocolos empleados, desglosados según las distintas capas del modelo *Open Software Initiative*, (OSI). Así mismo, en la Figura 2.6 se muestra un diagrama ilustrativo del modelo.

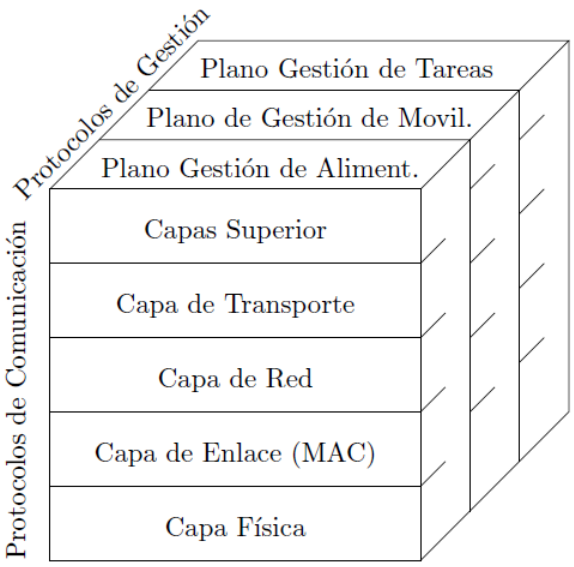


Figura 2.6: Modelo de capas para las comunicaciones en WSN.

2.6.1 Protocolos de Nivel Físico

Aunque la transmisión en las WSN se puede realizar por infrarrojos, radio o medio óptico; la banda industrial, científica y médica (*Industrial, Scientific and Medical*, ISM) se ha hecho muy popular en las redes de sensores. Las frecuencias ISM más empleadas son 915MHz en EEUU, 433.39MHz en Europa y 2.4 GHz de forma global. El inconveniente de usar infrarrojos o medios ópticos para la trasmisión es que se requiere que los nodos transmisor y receptor mantengan una línea de contacto visible. Algunas especificaciones inalámbricas como Bluetooth, HomeRF, y las redes LAN Wireless especificadas por el IEEE 802.11b/g/n operan todas en la frecuencia de los 2.4GHz.

Precisamente la misma frecuencia donde opera el estándar por excelencia en WSN: IEEE 802.15.4 [3]. Sobre este estándar se ha construido una solución más compleja que resuelve las comunicaciones hasta el nivel de red. Esta solución se conoce como ZigBee [4] y es la especificación de un conjunto de protocolos de comunicaciones desarrollados específicamente para las WSNs.

2.6.2 Protocolos de Nivel de Enlace: (Medium Access Control, MAC)

La responsabilidad de la capa de enlace es establecer un enlace fiable o una infraestructura de red sobre la que los datos puedan ser encaminados. Existen especificaciones como Bluetooth que utilizan protocolos de control de acceso al medio como la multiplexación por división en el tiempo (*Time Division Multiple Access*, TDMA) con saltos de frecuencia mientras que las redes LAN inalámbricas especificadas

por el 802.11b utilizan un método de acceso al medio con detección de portadora y evitando colisiones (*Carrier Sense Multiple Access/Collision Avoidance*, CSMA/CA).

La necesidad de un nuevo protocolo de capa MAC para WSN radica en que las dificultades de las WSN son muy distintas de los problemas que tenían las especificaciones existentes.
Un protocolo de capa MAC para las WSN es el llamado MAC autoorganizado para redes de sensores (*Sensor MAC*, SMAC), que configura la capa de enlace. El algoritmo de escucha y registro (*Ear And Register*, EAR) permite a los nodos sensores móviles interconectar nodos estacionarios. SMAC actúa al crear la red detectando los nodos vecinos usando transferencia de mensajes. En SMAC, un canal se define con un par de intervalos de tiempo.

La detección de vecinos y la asignación de canales se combinan en una fase, para que cuando los nodos vayan a escuchar a sus vecinos, ya hayan formado una red conectada. No hay jerarquías asumidas en SMAC y por esto se forma una topología llana. El algoritmo EAR tiene el problema del control de la movilidad cuando se introducen nodos móviles en la red.

En la práctica, la mayoría de sistemas de WSN se basan en el estándar IEEE 802.15.4 [3] para resolver las comunicaciones hasta el nivel de enlace. Las principales características de este estándar son:

- Tasas de transferencia de 250 kbps, 40 kbps y 20kbps.
- Dos modos de direccionamiento: 16-bit short y 64-bit IEEE-addressing.
- Soporte para dispositivos críticos en latencia.
- Acceso a canal por CSMA-CA.
- Establecimiento automático de la red por el coordinador.
- Gestión de energía para asegurar bajo consumo.
- 16 canales en la banda ISM de 2.4 GHz, 10 canales en la banda 915MHz y un canal en la banda 868 MHz.

2.6.3 Protocolos de Nivel de Red

El encaminamiento en las WSN es bastante similar al de los protocolos ad hoc en las redes MANETS. La diferencia es que en los algoritmos de enrutamiento ad hoc, el consumo de potencia es secundario. Los requisitos de potencia de los nodos sensores son mucho menores que para Bluetooth. Varios algoritmos se han propuesto para el

encaminamiento de WSN. El principal objetivo de los algoritmos es el bajo consumo. Se pueden clasificar como aquellos que determinan:

1. Rutas con los nodos de mayor potencia a lo largo de la ruta.
2. Rutas que consuman la mínima energía.
3. Rutas con el mínimo número de saltos.
4. Rutas en las que la mínima potencia disponible es la máxima entre todos los demás caminos.

2.6.4 Protocolos de Nivel de Transporte

La necesidad de una capa de transporte radica en que las WSN necesitan ser conectadas a una red más grande, como Internet. Las WSN se conectan a Internet por medio de pasarelas. El protocolo de la capa de transporte que conecta el usuario con la pasarela podría ser TCP (*Transmission Control Protocol*) o UDP (*User Datagram Protocol*), ya existentes.

Sin embargo, el protocolo que conecta la pasarela y los nodos sensores tendría que ser diferente ya que no hay un esquema de direccionamiento global en una red de sensores. La limitación de memoria en los nodos sensores conlleva una cuestión importante en tanto que se prefieren protocolos que requieran un bajo almacenamiento de información de estado antes que otros del tipo TCP.

2.6.5 Protocolos de Nivel de Aplicación

Los nodos sensores tienen muchas aplicaciones distintas. Diseñar una capa de aplicación tiene el mérito de que las WSN pueden ser conectadas a grandes redes como Internet. El direccionamiento de nodos es una cuestión importante aquí ya que, no como en otras redes, los nodos sensores no tienen un identificativo global.

Los protocolos de la capa de aplicación como el Protocolo de Administración de Sensores (*Sensor Management Protocol*, SMP) y el Protocolo de Petición de Sensores y Entrega de Datos (*Sensor Query and Data Dissemination Protocol*, SQDDP) están actualmente en investigación. El SQDDP introduce una interfaz de peticiones para emitir peticiones, responderlas y recopilar las respuestas de las peticiones. Aunque las aplicaciones de las WSN son diversas, las últimas investigaciones indican que se van a tener que realizar más desarrollos en el direccionamiento de varios protocolos en todas las capas. Además, se necesita una plataforma común donde se puedan probar los algoritmos propuestos. Las simulaciones de WSN son difíciles y normalmente están vinculadas a una aplicación en particular. El área de las WSN es un área en expansión y

en los próximos años se percibirán los resultados de los esfuerzos que se están realizando en estas aplicaciones.

2.7 Aplicaciones de las WSN

Una WSN puede consistir en muchos tipos diferentes de sensores, como pueden ser sísmicos, magnéticos, térmicos, acústicos, radar, IR, etc. [5] Los distintos tipos de sensores existentes pueden monitorizar una gran variedad de condiciones ambientales, que incluyen:

- Temperatura, humedad, presión.
- Condiciones de luz, movimiento de vehículos, niveles de ruido.
- Composición del suelo.
- Presencia o ausencia de cierto tipo de objetos.
- Niveles de estrés mecánico en objetos (maquinaria, estructuras, etc.).
- Características de velocidad, dirección y tamaño de un objeto.

Los nodos sensores pueden adoptar diversas formas de trabajo, pueden actuar en modo continuo, por detección de eventos, por identificación de eventos, toma de datos localizados o como control local de actuadores (idóneo para aplicaciones domóticas). Habiendo determinado el tipo de monitorización que van a realizar los sensores, se puede hacer una primera clasificación de aplicaciones, en tres tipos distintos que tendrían las siguientes propiedades:

- *Monitorización del entorno*: Este tipo de aplicaciones se caracterizan por un gran número de nodos sincronizados que estarán midiendo y transmitiendo periódicamente en entornos puede que inaccesibles, para detectar cambios y tendencias. La topología es estable y no se requieren datos en tiempo real, sino para análisis futuros. Ejemplos: control de agricultura, microclimas, etc.

- *Monitorización de seguridad*: Son aplicaciones para detectar anomalías o ataques en entornos monitorizados continuamente por sensores. No están continuamente enviando datos, consumen menos y lo que importa es el estado del nodo y la latencia de la comunicación: se debe informar en tiempo real. Como ejemplos tenemos el control de edificios inteligentes, detección de incendios, aplicaciones militares, seguridad, etc.

- *Seguimiento (Tracking)*: Aplicaciones para controlar objetos que están etiquetados con nodos sensores en una región determinada. La topología va a ser muy dinámica debido al continuo movimiento de los nodos: se descubrirán nuevos nodos y se formaran nuevas topologías.

- *Redes híbridas*: Son aquellas en las que los escenarios de aplicación reúnen aspectos de las tres categorías anteriores.

El concepto de microsensores comunicados de forma inalámbrica promete muchas nuevas áreas de aplicación. De momento las vamos a clasificar en: militares, entorno, salud, hogar y otras áreas comerciales. Por supuesto es posible ampliar esta clasificación.

2.7.1 Apicaciones Militares

Las WSNs pueden ser parte integral de sistemas militares C4ISRT (command, control, communications, computing, intelligence, surveillance, reconnaissance and targeting) que son los que llevan las órdenes, el control, comunicaciones, procesamiento, inteligencia, vigilancia, reconocimientos y objetivos militares. El rápido y denso despliegue de las redes de sensores, su autoorganización y tolerancia a fallos las hace una buena solución para aplicaciones militares. Ofrecen una solución de bajo coste y fiable para éstas ya que la pérdida de un nodo no pone en riesgo el éxito de las operaciones. Ejemplos de aplicación en esta área son:

- *Monitorización de fuerzas aliadas, equipamiento y munición*: Cada equipo, tropa, vehículo o arma crítica lleva integrado un sensor para informar de su estado a líderes o niveles superiores. Se usan nodos recolectores donde se recopila toda la información para luego transmitirla a niveles superiores.

- *Reconocimiento del terreno y fuerzas enemigas*: Se pueden desplegar y obtener información valiosa antes de que el enemigo los intercepte sobre rutas de acceso, posibles caminos e incluso movimientos del enemigo.

- *Adquisición de blancos*: Se pueden incorporar los nodos a sistemas de guiado de armas inteligentes.

- *Valoración de daños*: Antes o después de los ataques.

- *Reconocimiento de ataques*: Nucleares, biológicos y químicos, desplegándolos en la región y usándolos como sistema de aviso. También para reconocimiento después del ataque sin tener que exponer a equipos de reconocimiento a radiaciones o agentes químicos.

2.7.2 Aplicaciones Medioambientales

En este campo se encuentran aplicaciones como el seguimiento de aves, animales e insectos; monitorización de condiciones ambientales que afectan al ganado y las cosechas; irrigación; macro instrumentos para la monitorización planetaria de gran escala; detección química o biológica; agricultura de precisión (monitorización de niveles de pesticidas, polución y erosión del terreno); detección de incendios; investigación meteorológica o geofísica; detección de inundaciones; mapeado de la biocomplejidad del entorno; y estudios de la polución. Podemos destacar dos de estas aplicaciones:

- *Detección de fuego en bosques*: Se podrían desplegar miles de sensores de manera estratégica, pseudo-aleatoria y densa en el bosque que informarían del origen exacto de un incendio antes de que se haga incontrolable. Los sensores podrían tener métodos de obtención de energía como placas solares, ya que serían abandonados durante meses o años sin mantenimiento. Además, debido a la densidad de su despliegue serían capaces de cooperar para realizar una recolección de datos cooperativa y evitar obstáculos en las transmisiones, como árboles y rocas que pueden bloquear la línea de visión entre sensores.

- *Detección de Inundaciones*: De forma muy similar a la detección de incendios, las WSNs pueden emplearse para la detección de inundaciones; como de muestran los estudios realizados por investigadores del Instituto Tecnológico de Massachusetts [6].

2.7.3 Aplicaciones Sanitarias

Mediante el uso de WSNs se puede proveer de interfaces para los discapacitados; monitorización integral de pacientes; diagnósticos; administración de medicamentos en hospitales; monitorización de los movimientos y procesos internos de insectos u otros pequeños animales; telemonitorización de datos fisiológicos humanos; y seguimiento y monitorización de pacientes en un hospital.

- *Telemonitorización de datos fisiológicos humanos*: Los datos recolectados se pueden almacenar durante períodos largos de tiempo, usados para exploración médica. La WSN instalada puede monitorizar y detectar el comportamiento de personas mayores, como por ejemplo una caída [2]. Los sensores, por su reducido tamaño, dan al usuario una mayor libertad de movimiento y permiten a los médicos identificar antes síntomas predefinidos. Además, permiten una mayor calidad de vida a los usuarios comparados con los centros de tratamiento.

- *Seguimiento y monitorización de médicos y pacientes*: Cada paciente lleva un sensor pequeño y ligero. Cada sensor tiene una tarea específica, por ejemplo monitorizar la presión arterial y la frecuencia cardiaca. Los médicos también llevan sensores, lo que permite a otros médicos localizarlos en el hospital.

- *Administración de medicación en hospitales*: Colocar un sensor en la medicación reduce el riesgo de errores, porque el paciente también podría llevar un sensor que identifique sus alergias y la medicación requerida.

2.7.4 Aplicaciones Domóticas

Los nodos sensores pueden ser introducidos en aparatos domésticos como aspiradoras, microondas, hornos, frigoríficos y VCRs. Esto permite que sean manejados remotamente por los usuarios finales mediante una comunicación que se realizaría vía satélite o Internet.

A través de las redes de sensores pueden crear hogares inteligentes donde los nodos se integran en muebles y electrodomésticos. Los nodos dentro de una habitación se comunican entre ellos y con el servidor de la habitación. Estos servidores de habitaciones se comunican también entre ellos dando así conectividad entre distintas habitaciones. Se crea lo que se conoce como entorno inteligente, cuyo diseño puede tener dos enfoques: desde el punto de vista humano, el entorno se adapta a necesidades del usuario final en términos de capacidades de entrada-salida; desde el punto de vista tecnológico, hay que desarrollar nuevas tecnologías hardware, soluciones de redes y servicios middleware.

Los sensores pueden integrarse en muebles y aparatos. Se comunican entre ellos y con servidores o actuadores de una habitación, que a su vez se comunican con otros servidores de otras habitaciones. Todos ellos se integran y organizan con los dispositivos integrados existentes para autoorganizarse, autorregularse y autoadaptarse basándose en modelos de control.

En este Proyecto de Fin de Carrera la ubicación de la WSN es un domicilio, por lo que la topología, así como otras características, serían las de una red aplicada a la domótica. Sin embargo, el servicio prestado tiene un cierto carácter sanitario; por lo que otras características, como la fiabilidad, mostrarán parámetros propios de estas aplicaciones.

2.8 Plataformas WSN Comerciales

Existen diferentes plataformas hardware sobre las que desarrollar aplicaciones basadas en redes de sensores. El principal fabricante especializado en kits de

desarrollo de WSNs fue Crossbow, una spin-off de la Universidad de Berkely [7], pero en enero de 2010 la división no militar de Crossbow junto con los derechos y patentes de dispositivos de WSN fueron adquiridos por MEMSIC, Inc. además de la incorporación de varios ingenieros de la compañía Crossbow, por 18$ millones. Ahora es esta compañía la que fabrica los kits de desarrollo.

A continuación se muestra una breve descripción de sus principales productos [8] ya que parte del desarrollo de este proyecto se ha basado en las ideas proporcionadas por este fabricante.

2.8.1 Tipos de nodos sensores

Micaz

La plataforma Micaz de MEMSIC emplea el chip de radio CC2420 que cumple con la normativa IEEE 802.15.4 [3] e integra el microcontrolador de 8 bits ATMEGA128L. Esta familia dispone de un único modelo, el MPR2400, que dispone de un conector de expansión de 51 pines así como de memoria flash extendida. La tasa de transferencia de datos es de 250 kbps, el consumo medio estimado es de 8mA en activo y 15 µA en modo pasivo y el alcance máximo es de unos 150 metros.

El microcontrolador puede ser programado mediante la conexión al bus de expansión de alguna de las placas de interfaz (comunicaciones/programación) que MEMSIC oferta. En concreto, Micaz es compatible con los modelos MIB510, MIB520 y MIB600. La combinación de este módulo de radio con una de las tarjetas indicadas anteriormente se puede emplear como estación base siempre que el mote esté programado convenientemente.

En la Figura 2.7 se puede contemplar una fotografía del módulo mientras que en la Figura 2.8 se muestra su diagrama de bloques. Más información online en la página web de MEMSIC [8].

Figura 2.7: Fotografía del módulo Micaz

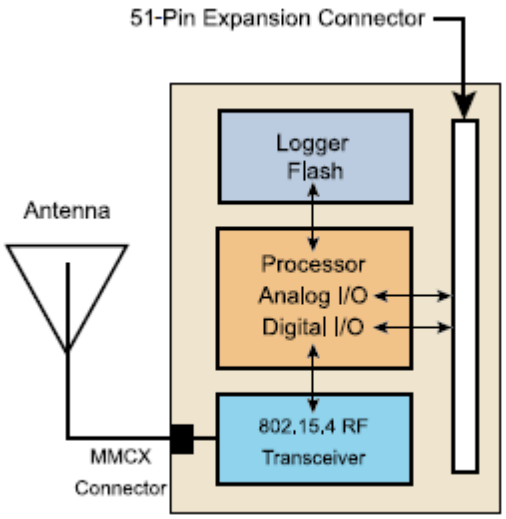


Figura 2.8: Diagrama de bloques del módulo Micaz

TelosB

La familia TelosB fue una de las últimas plataformas lanzadas por Crossbow. En esta ocasión no se trata de un módulo de radio con bus de expansión sino de un modelo cerrado. Al contrario que los demás modelos, el TelosB no necesita de una tarjeta adicional para ser programado. Este modelo incorpora una interfaz USB que le dota de conectividad directa con un PC. De esta forma, la interfaz USB se puede emplear para programar el microcontrolador o como enlace de datos dotando al dispositivo del hardware necesario para actuar como estación base de la red de sensores.

Incorpora el módulo de Radio CC2420 que opera en la banda de 2.4 GHz e implementa el estándar IEEE 802.15.4. Por otro lado, el microcontrolador empleado en este modelo es el TI MSP430, con arquitectura de 16 bits y una de las mejores prestaciones en consumo. La tarjeta incorpora también memoria flash externa y un identificador serie único (*unique-ID*). La tasa de transferencia de datos es de 250 kbps, el consumo medio estimado es de 23mA en activo y 1µA en modo pasivo y el alcance máximo es de unos 100 metros.

Existen dos modelos de TelosB que definen los sensores integrados en el dispositivo. El modelo TPR2420 incorpora sensores de iluminación, temperatura y humedad; mientras que el modelo TPR2400 no incorpora sensores pudiendo actuar exclusivamente como nodo de red.

En la Figura 2.9 se puede contemplar una fotografía del módulo mientras que en la Figura 2.10 se muestra su diagrama de bloques. Más información online en la página web MEMSIC.

Figura 2.9: Fotografía del módulo TelosB

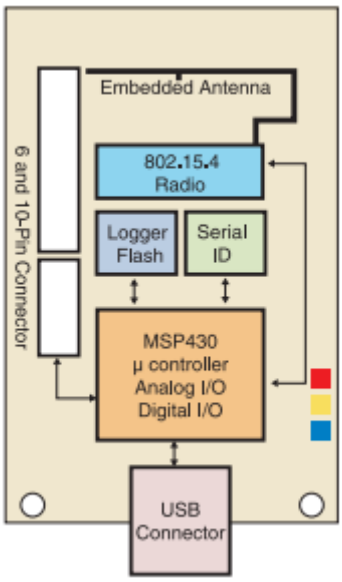


Figura 2.10: Diagrama de bloques del módulo TelosB

Lotus

Lotus es una plataforma desarrollada sobre la CPU de bajo consumo Cortex M3 e incorpora lo mejor de Iris y TelosB en un único dispositivo. El diseño es modular con interfaces de conexión para añadir expansiones con otras placas y presenta nuevas capacidades que mejoran la funcionalidad de los productos destinados a WSNs de MEMSIC.

Incorpora el módulo de radio RF231 que implementa el estándar 802.15.4. Soporta 250kbps con 16 canales a 2.4GHz. La plataforma Lotus tiene integrada una antena con un rango de 100 metros. Para obtener más rango es posible soldarle a la placa un conector SMA para conectar una antena con un aplificador.

Por otro lado, el microcontrolador está basado en un LPC1758, esta serie de microcontroladores son de 32 bits y están diseñados específicamente para aplicaciones empotradas (*embedded applications*) con un alto nivel de integración y un consumo de potencia muy bajo.

Gracias a su conector de 51 pines, MEMSIC ofrece una gran variedad de sensores y tarjetas de adquisición de datos compatibles con esta interfaz. En la Figura 2.11 se puede contemplar una fotografía del módulo mientras que en la Figura 2.12 se muestra su diagrama de bloques. Más información online en la página web MEMSIC [8].

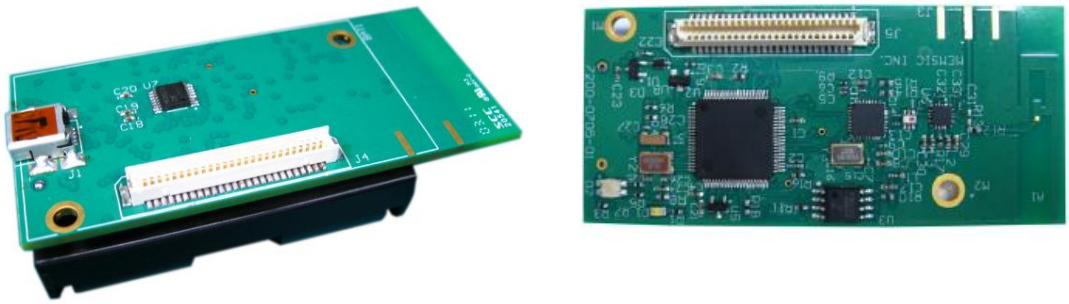


Figura 2.11: Fotografía de las partes anterior y posterior del módulo Lotus

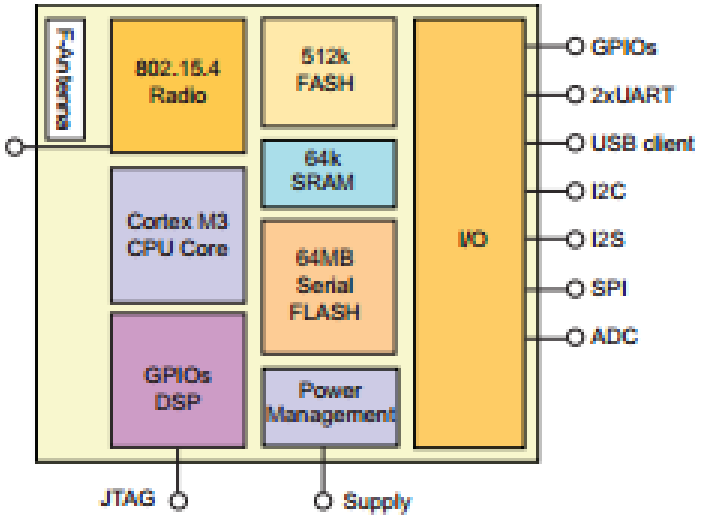


Figura 2.12: Diagrama de bloques del módulo Lotus

Iris

La plataforma Iris es la evolución de la paltaforma Micaz, de hecho también se la conoce como Micaz. Presenta un único modelo, el XM2110, que incorpora el microcontrolador de 8 bits ATmega1281 [9], optimizado para bajo consumo, y el chip de radio AT86RF230 [10] que opera en la banda de 2.4 GHz cumpliendo con el estándar IEEE 802.15.4 [3] y con una potencia de transmisión y sensibilidad en recepción superiores a los empleados en las demás familias. Al igual que la familia TelosB, este modelo incorpora una memoria flash externa y un identificador serie único. La tasa de transferencia de datos es de 250 kbps, el consumo medio estimado es de 8mA en activo y 15µA en modo pasivo y el alcance máximo es de unos 150 metros.

Este modelo se puede combinar con las tarjetas de interfaz de la misma manera que se indica para el modelo Micaz en el apartado anterior. En la Figura 2.13 se puede contemplar una fotografía del módulo mientras que en la Figura 2.14 se muestra su diagrama de bloques.

En la realización de este proyecto se usarán varios de los componentes de esta plataforma para la fabricación del prototipo. Por este motivo se exponen, a continuación, una descripción algo más detallada de cada uno de sus componentes fundamentales.

Figura 2.13: Fotografía del módulo Iris

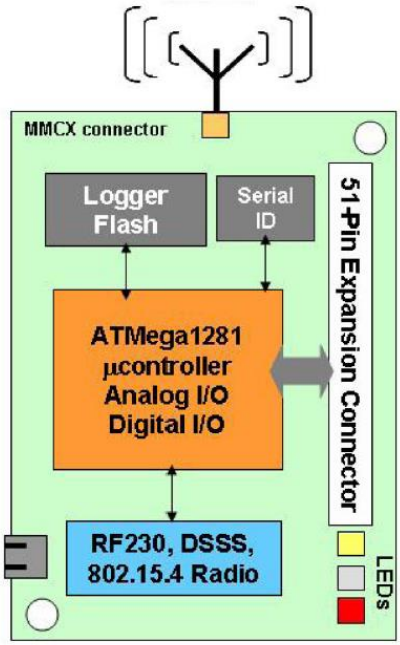


Figura 2.14: Diagrama de bloques del módulo Iris

Microcontrolador ATMEL ATmega1281: El ATmega1281 es un microcontrolador CMOS de bajo consumo basado en la arquitectura *Enhanced-RISC* de 8-bits de AVR [9]. El AVR es una CPU de arquitectura Harvard. Tiene 32 registros de 8 bits. Algunas instrucciones sólo operan en un subconjunto de estos registros. La concatenación de los 32 registros, los registros de entrada/salida y la memoria de datos conforman un espacio de direcciones unificado, al cual se accede a través de operaciones de carga/almacenamiento. A diferencia de los microcontroladores PIC (su competencia), el stack se ubica en este espacio de memoria unificado, y no está limitado a un tamaño fijo. Ejecutando instrucciones potentes en un ciclo de reloj, el ATmega640/120/1281/2560/2561 consigue *throughputs* de aproximadamente 1 MIPs por MHz; permitiendo al diseñador del sistema optimizar el consumo de energía frente a la velocidad de procesamiento.

Este microcontrolador posee la capacidad para añadirle una memoria RAM externa tal y como nos indica su hoja de características, dato muy importante debido que es uno de los cambios que se introducirán en el desarrollo de este proyecto.
En el Cuadro se resumen las principales características del ATmega1281.
Independientemente, se puede obtener más información on-line en la página web de Atmel.

Flash	EEPROM	RAM	GPIO	USARTs	SPI	ADC	Ext. Int.
128 kB	4 kB	8 kB	54	2	1	8	8

Tabla 2.1: Características del microcontrolador ATmega1281

Transceptor de Radio AT86RF230: La radio usada en el módulo IRIS es un transceptor conforme al estándar IEEE 802.15.4 [3] diseñado para aplicaciones inalámbricas de bajo voltaje y bajo consumo. El AT86RF230 [10] es una solución altamente integrada para comunicaciones inalámbricas en la banda ISM de 2.4GHz. Cumple los reglamentos establecidos a nivel mundial por la ETSI EN 300 328 y EB 300 440 class 2 (en Europa), FCC cFR47 Part 15 (en EEUU) y ARIB STD-T66 (en Japón).
Este transceptor de radio se puede sintonizar a través de los canales establecidos en el estándar IEEE 802.15.4 que están numerados desde el 11 (2.405 GHz) hasta el 26 (2.480 GHz); separados cada uno por 5 MHz. Además, la potencia de transmisión es programable desde los 3dBm hasta los -17.2dBm. El empleo de potencias de transmisión más débiles puede resultar ventajoso al reducir las interferencias y el consumo de la radio.

Antena: El módulo Iris incorpora una antena monopolo en $\lambda/4$, conectada mediante un conector MMCX. Esta antena puede ser sustituida por otro modelo a conveniencia.

Conector de Expansión: El conector de expansión proporciona conectividad para tarjetas sensoras y tarjetas de interfaz. El conector incluye interfaces para alimentación, masa, control de alimentación de sensores periféricos, entradas a ADC para lecturas de sensores, GPIOs, así como interfaces UART y I2C.

Baterías: El módulo iris se alimenta típicamente mediante dos baterías formato LR6 (AA). La tarjeta presenta un conector de dos pines para la conexión de un porta-pilas externo y también dos terminales para soldar un porta-pilas *on-board*.

Tres

Herramientas de Desarrollo

3.1 TinyOS

3.1.1 Introducción a TinyOS

TinyOS es un sistema operativo para trabajar con redes de sensores desarrollado en la Universidad de Berkeley [11]. TinyOS puede ser visto como un conjunto de programas avanzados, que cuenta con un amplio uso por parte de comunidades de desarrollo, dada sus características de ser un proyecto de código abierto (Open Source). Este conjunto de programas contiene numerosos procedimientos, que permiten desde la generación tablas de enrutamiento en malla hasta el manejo transparente de los periféricos de un microcontrolador. Además, soporta diferentes plataformas de nodos de sensores, arquitectura base para el desarrollo de aplicaciones.

Esta aplicación se ha programado sobre TinyOS, es por ello que se dedica una sección de este documento a introducir los conceptos fundamentales de TinyOS así como el modelo de programación para las aplicaciones.

El lenguaje en el que se encuentra programado TinyOS es un metalenguaje que deriva de C; cuyo nombre es NesC. Además, existen varias herramientas que ayudan el estudio y desarrollo de aplicaciones para las redes de sensores; que van desde aplicaciones para la obtención y manejo de datos hasta sistemas completos de simulación.

El diseño de TinyOS está basado en responder a las características y necesidades de las redes de sensores, tales como reducido tamaño de memoria, bajo consumo de energía, operaciones de concurrencia intensiva, diversidad en diseños y usos, y finalmente operaciones robustas para facilitar el desarrollo confiable de

aplicaciones. Además, se encuentra optimizado en términos de uso de memoria y eficiencia de energía.

El diseño del núcleo (Kernel) de TinyOS está basado en una estructura de dos niveles de planificación:

- *Eventos:* Pensados para realizar un proceso pequeño (por ejemplo la interrupción del contador del temporizador o de un conversor analógico-digital). Los eventos tienen capacidad para interrumpir la ejecución de las tareas en curso.

- *Tareas:* Las tareas están pensadas para realizar una cantidad mayor de procesamiento y no son críticas en tiempo. Las tareas se ejecutan en su totalidad, pero la solicitud de iniciar una tarea, y el término de ella son funciones separadas. Esta característica es propia de la programación orientada a componentes, la cual se presentará en detalle en la sección siguiente. Con este diseño se permite que los eventos (que son rápidamente ejecutables), puedan ser realizados inmediatamente, pudiendo interrumpir a las tareas (que tienen mayor complejidad en comparación con los eventos).

El enfoque basado en eventos es la solución ideal para alcanzar un alto rendimiento en aplicaciones de concurrencia intensiva. Adicionalmente, este enfoque usa las capacidades de la CPU de manera eficiente y de esta forma gasta el mínimo de energía. TinyOS está programado en NesC, un lenguaje diseñado para reflejar las ideas propias del enfoque de componentes, incorporando además un modelo de programación que soporta concurrencia, manejo de comunicaciones y fácil interacción con el medio (manejo de hardware).

TinyOS y NesC se encuentran profundamente relacionados, es por eso que a continuación se presenta un pequeño resumen con algunas de las características de este lenguaje.

3.1.2 Instalación de TinyOS y sus componentes

Para el desarrollo de la parte software de este proyecto se necesitará instalar TinyOS. Se opta por la versión 2.x de este SO por tener una comunidad activa y estar más actualizado que su versión 1.x. Es posible instalarlo en varios sistemas operativos, por ejemplo Windows. Para ello es necesario el software Cygwin, una aplicación que permite utilizar herramientas *Open Source* y GNU en un entorno similar a una distribución de Linux. El proceso de instalación se encuentra detallado en la web de TinyOS y básicamente se puede resumir en estos sencillos pasos:

- Instalación de Java Development Kit (JDK).

- Instalación de Cygwin con los paquetes rpm, make, perl y python.
- Instalación de un compilador adecuado a tu hardware y sistema operativo.
- Instalación de las herramientas de TinyOS (En este paso se instala el compilador de NesC, el lenguaje de programación del que se hablará en la Sección 3.1.3 y que TinyOS utiliza, además de otra serie de herramientas).
- Por último, una vez instaladas todas las herramientas, se procede a descargar el código fuente de TinyOS 2.x y a establecer una serie de variables de entorno necesarias para el funcionamiento del software.

Otra opción es instalar TinyOS en una distribución de Linux basada en Debian. Para este PFC se escoge la versión más actualizada de Ubuntu y se siguen los pasos de la guía de instalación incluida en la web de TinyOS:

1. Instalar desde el repositorio de tinyprod las herramientas que se usan en el desarrollo de aplicaciones con TinyOS siguiendo la guía de Eric Decker. El principal de los productos a instalar sería wheezy, pues incluye los paquetes más importantes para TinyOS (NesC, tinyos-tools, compiladores para los productos de atmel, etc.) Todo esto fácilmente instalable con unas pocas líneas de comandos.

2. Descargar y descomprimir TinyOS 2.x directamente desde el repositorio de github mediante los siguientes comandos:

```
wget http://github.com/tinyos/tinyos-release/archive/tinyos-
2_1_2.tar.gz
```

3. Añadir algunas variables de entorno al Shell. Para esto se crea un archivo (tinyos.env por ejemplo) con el contenido que se muestra en la Figura:

```
# Here we setup the environment
# variables needed by the tinyos
# make system

export TOSROOT="<local-tinyos-path>"
export TOSDIR="$TOSROOT/tos"
export CLASSPATH=$CLASSPATH:$TOSROOT/support/sdk/java
export MAKERULES="$TOSROOT/support/make/Makerules"
export
PYTHONPATH=$PYTHONPATH:$TOSROOT/support/sdk/python
```

4. Añadir al .bashrc estas variables es tan simple como escribir el siguiente comando en el terminal de Ubuntu:

```
source <local-tinyos-path>/tinyos.env
```

Por la facilidad para trabajar con un entorno Linux como es Ubuntu y la sencillez del proceso de instalación de TinyOS en dicho entorno, se elige éste para el desarrollo software de la aplicación.

3.1.3 NesC

NesC es un meta-lenguaje de programación basado en C, orientado a sistemas empotrados que incorporan el manejo de red [12]. Además, soporta un modelo de programación que integra el manejo de comunicaciones, las concurrencias que provocan las tareas y eventos y la capacidad de reaccionar frente a sucesos que puedan ocurrir en los ambientes donde se desempeña (por ejemplo, monitorizar). Básicamente, NesC ofrece:

- *Separación entre la construcción y la composición:* Hay dos tipos de componentes en NesC: módulos y configuraciones. Los módulos proveen el código de la aplicación, implementando una o más interfaces. Estas interfaces son los únicos puntos de acceso al componente. Las configuraciones son usadas para unir los componentes entre sí, conectando las interfaces que algunas componentes proveen con las interfaces que otras usan.
- *Interfaces bidireccionales:* Tal como ya se explicaba anteriormente, las interfaces son los accesos a los componentes, conteniendo comandos y eventos, los cuales implementan las funciones. El componente proveedor de una interfaz implementa los comandos, mientras que el usuario de la misma implementa eventos.
- *Unión estática de componentes, vía sus interfaces:* Esto aumenta la eficiencia en tiempos de ejecución, incrementa la robustez del diseño, y permite un mejor análisis del programa. NesC, también presenta herramientas que optimizan la generación de códigos. Finalmente y a modo de resumen, se mencionan los principales aspectos que el modelo de programación NesC ofrece y que deben ser entendidos para el entendimiento del diseño con TinyOS:

NesC, también presenta herramientas que optimizan la generación de códigos. Finalmente y a modo de resumen, se mencionan los principales aspectos que el modelo de programación NesC ofrece y que deben ser entendidos para el entendimiento del diseño con TinyOS:

- Tal como se ha mencionado, el modelo de NesC está formado por interfaces y componentes.

- Una interfaz puede ser usada o puede ser provista, los componentes son módulos o configuraciones.

- Una aplicación se verá representada como un conjunto de componentes, agrupados y relacionados entre sí, tal como se observa en la Figura 3.1.

- Las interfaces se utilizan para operaciones que describen la interacción bidireccional; el proveedor de la interfaz debe implementar comandos, mientras que el usuario de la interfaz debe implementar eventos.

- Existen dos tipos de componentes: Módulos, que implementan especificaciones de un componente; configuraciones, que se encargarán de unir (wire) diferentes componentes en función de sus interfaces, ya sean comandos o eventos.

En la Figura 3.2 se muestra un diagrama de bloques en el que se describe el proceso de compilación para una aplicación TinyOS. Mayor información acerca de NesC, remítase al "Manual de Referencia del Lenguaje" [12].

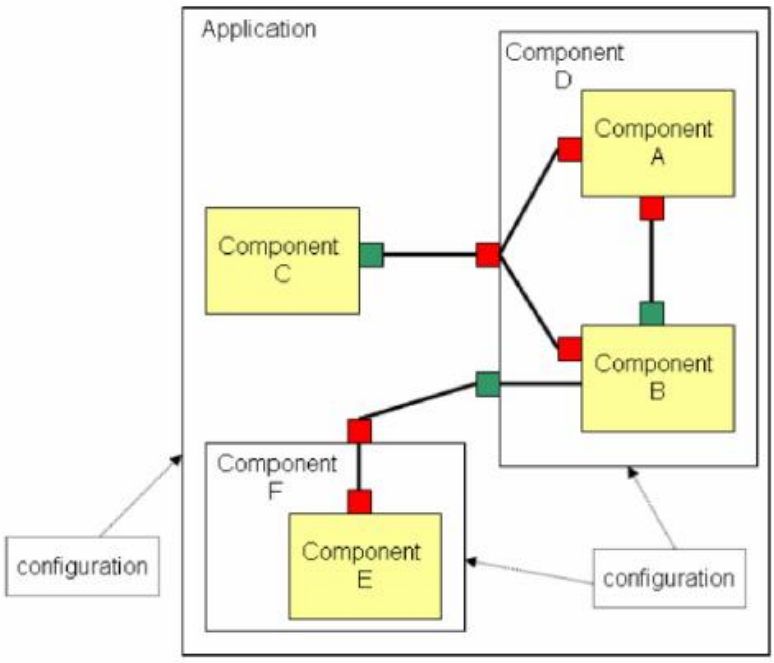


Figura 3.1: Ejemplo de Diagrama de una Aplicación en TinyOS.

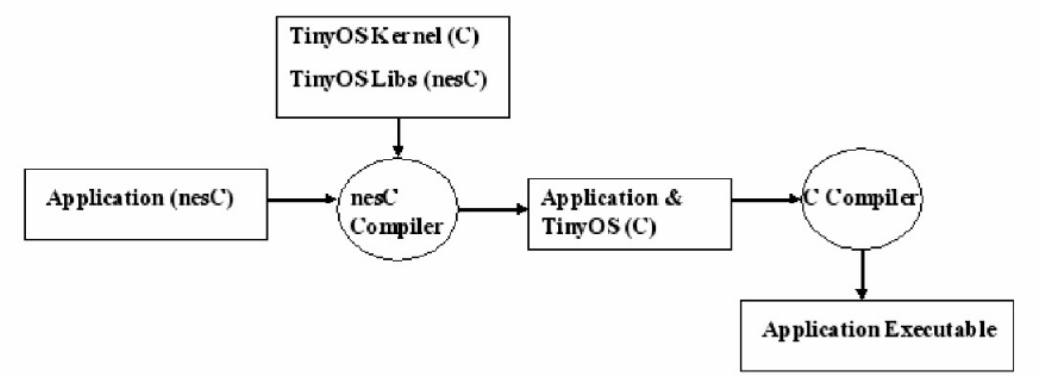


Figura 3.2: Esquema de Compilación de una Aplicación en TinyOS.

3.1.4 Modelo de Programación de TinyOS

Descripción del Modelo

TinyOS posee un modelo de programación característico de los sistemas empotrados: el modelo de componentes [13]. Tal como se ha mencionado, TinyOS está escrito en NesC, lenguaje que surge para facilitar el desarrollo de este tipo de aplicaciones. A continuación se presentan las ideas de este modelo.

Arquitectura basada en componentes: TinyOS se encuentra construido sobre un conjunto de componentes de sistema, las cuales proveen la base para la creación de aplicaciones. En la Figura 3.3 se muestra el modelo de un componente.

La interconexión de estos componentes, usando un conjunto de especificaciones detalladas, es lo que finalmente definirá una aplicación. Este modelo es esencial en sistemas empotrados para incrementar la confiabilidad, sin sacrificar desempeño.

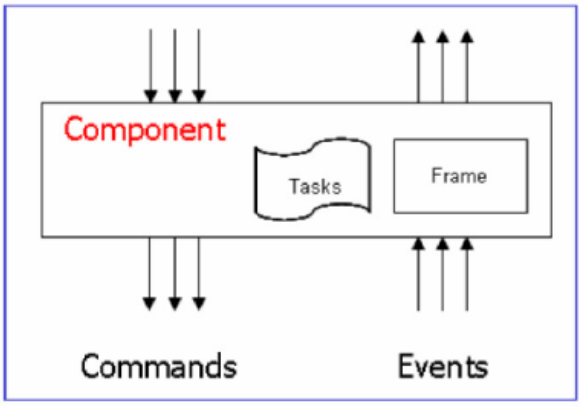


Figura 3.3: Modelo de Componente de TinyOS y su interfaz.

Concurrencia en tareas y en eventos: Las dos fuentes de concurrencia en TinyOS son las tareas y los eventos. Los componentes entregan tareas al planificador, siendo el retorno de éste de forma inmediata, aplazando el cómputo hasta que el planificador ejecute la tarea. Los componentes pueden realizar tareas siempre y cuando los requerimientos de tiempo no sean críticos. Para asegurar que el tiempo de espera no sea muy largo está recomendado programar tareas cortas; en caso de necesitar procesamientos mayores se recomienda dividirlo en múltiples tareas. Las tareas se ejecutan en su totalidad, y no tiene prioridad sobre otras tareas o eventos. Así también los eventos se ejecutan hasta completarse, pero estos sí pueden interrumpir a otros eventos o tareas; con el objetivo de cumplir de la mejor forma los requerimientos de tiempo real.

Todas las operaciones de larga duración deben ser divididas en dos estados: la solicitud de la operación y la ejecución de esta. Específicamente si un comando solicita la ejecución de una operación éste debe obtener respuesta inmediatamente; mientras que la ejecución queda en mano del planificador, el cual deberá señalizar a través de un evento el éxito de la operación.

Una ventaja secundaria de elegir este modelo de programación, es que propaga las abstracciones del hardware en el software. Tal como el hardware responde a cambios de estado en sus pines de entrada/salida, los componentes responden a eventos y a los comandos en sus interfaces.

Ahora se tiene claro que las aplicaciones TinyOS son construidas por componentes. Un componente provee y usa interfaces. Estas interfaces son el único punto de acceso al componente. Además, un componente estará compuesto por un espacio de memoria y un conjunto de tareas.

Cuatro son los elementos que conforman un componente: (1) manejador de comandos; (2) manejador de eventos, (3) un *frame* de tamaño fijo y estáticamente asignado, en el cual se representa el estado interno del componente; y (4) un bloque con tareas simples.

Los comandos son peticiones hechas a componentes de capas inferiores. Éstos generalmente son solicitados para ejecutar alguna operación. Existen dos posibles casos de uso de los comandos. El primero es para operaciones bifase, donde los comandos retornan inmediatamente, no generando bloqueos por la espera de la ejecución, es decir, una vez que realizan la petición, el planificador será el encargado de ejecutar lo solicitado, generando un evento que indique el fin de la operación. El otro es el caso de realizar una operación no bifase, donde ésta se realizará completamente, y por tanto no habrá evento retornado.

Los manejadores de eventos son invocados por eventos de componentes de capas inferiores, o por interrupciones cuando se está directamente conectado al

hardware. Similar a los comandos, el *frame* será modificado y las tareas agregadas. Los eventos son capaces de interrumpir tareas, pero no al contrario.

Las tareas se ejecutan hasta terminar y pueden ser sólo interrumpidas por eventos (podría decirse que eventos tienen mayor prioridad). Las tareas son entregadas a un planificador (*task scheduler*) que en este caso está implementado con método FIFO. Debido a esta implementación, las tareas se ejecutan secuencialmente y no deben ser excesivamente largas. Como alternativa al planificador de tareas FIFO, éste puede ser reemplazado por planificadores basados en prioridades (*priority-based*) o por planificadores basados en plazos (*deadline-based*), los cuales pueden ser implementados sobre TinyOS.

Tipos de Componentes

En general, los componentes TinyOS se pueden clasificar en una de las siguientes categorías:

- *Abstracciones de Hardware:* Proyectan el hardware físico en el modelo de componentes de TinyOS. Por ejemplo, en el módulo que se observa en la Figura 3.4, el componente de radio RFM (*Radio Frequency Module*), es representativo de esta clase, ya que exporta comandos para manipular los pines de entrada-salida conectados al RFM. También, entrega eventos informando a otros componentes acerca del bit de transmisión o recepción. Su *frame* contiene información acerca del estado actual del componente (si se encuentra transmitiendo o recibiendo, la tasa de transferencia de bits, etc.)
  El RFM detecta las interrupciones del hardware que se transforman en el bit de evento de RX o en el bit TX, dependiendo del modo de operación.

- *Hardware Sintético*: Los componentes de hardware sintéticos simulan el comportamiento del hardware avanzado. Un buen ejemplo de tal componente es el Radio Byte (véase la Figura 3.4). Intercambia datos de entrada o salida del módulo RFM y señaliza cuando un byte ha sido completado. Las tareas internas realizan la codificación y decodificación de los datos. Conceptualmente, este componente es una máquina de estados que podría ser directamente modelada en el hardware. Desde el punto de vista de los niveles superiores, este componente provee una interfaz, funcionalmente muy similar al componente de abstracción de hardware UART1: proporcionan los mismos comandos y señalan los mismos eventos, se ocupan de datos del mismo tamaño, e internamente realizan tareas similares (buscando un bit o un símbolo de inicio, realizando una codificación simple, etc.).

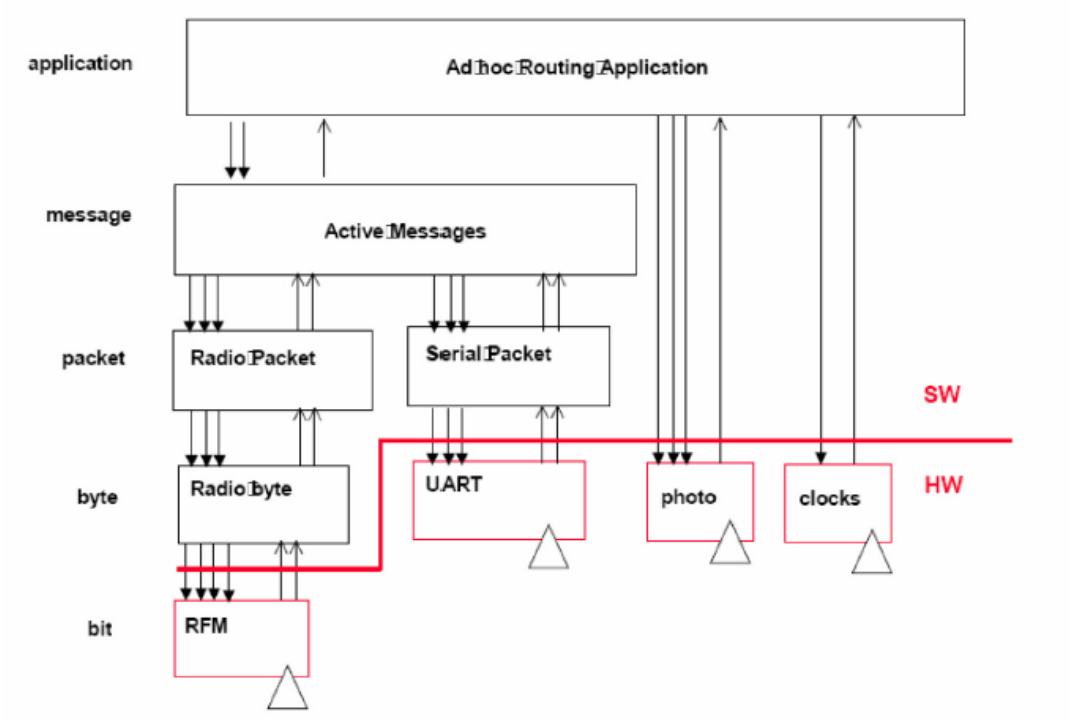


Figura 3.4: Modelo de Componentes para un Sistema Multi-Hop.

- *Componente de alto nivel*: Realizan el control, enrutamientos y toda la transferencia de datos. Un representante de esta clase es el módulo de mensajes (*messaging module*), presentado en la Figura 3.4. Éste realiza la función de llenar el buffer de los paquetes antes de la transmisión y envía mensajes recibidos a su lugar apropiado. Además, los componentes que realizan cálculos sobre los datos o su agregación, entran en esta categoría.

Otra de las características de TinyOS es que el modelo de componentes permite la fácil migración a otro hardware, dada la abstracción de hardware que se logra con el modelo de manejo de eventos. Además, la migración es particularmente importante en las redes de sensores, ya que constantemente aparecen nuevas tecnologías, donde de seguro los diseñadores de sistemas querrán explorar el compromiso entre la integración a nivel físico, los requerimientos de energía, y el costo del sistema, requerimientos claves para la optimización.

Compilación y Programación del Dispositivo

Instalar TinyOS en un PC significa instalar Linux o Cygwin junto con una serie de librerías y compiladores de código. Como se muestra en la Figura 3.2, una aplicación ejecutable en microcontrolador (código máquina) basada en TinyOS se obtiene realizando una secuencia que implica una doble compilación. En primer lugar se compila la aplicación del programador escrita en NesC, junto con las librerías de

TinyOS, para obtener un código C apenas ininteligible que posteriormente será compilado para el microcontrolador en cuestión. En primer lugar se hace uso, por tanto, de un primer compilador de código NesC que vendrá determinado por la versión de TinyOS empleada. En segundo lugar se hace uso de un compilador específico (normalmente Gnu) del microcontrolador en el que se desea compilar la aplicación (gcc para AVR, gcc para MSP430, etc.).

Al compilar el código NesC es necesario también disponer de las librerías adecuadas para el microcontrolador que se desea emplear. TinyOS permite abstraerse del empleo de los periféricos del micro, por lo que es necesario disponer de las librerías (componentes) que manejan dichos periféricos para poder enlazarlas con el código de la aplicación.

La secuencia de compilación está predefinida en un makefile específico para cada plataforma. Este archivo especifica el microcontrolador, chip de radio y otros componentes típicos de un nodo sensor. Por tanto, en la práctica siempre que se desee trabajar con una plataforma soportada por TinyOS, simplemente hay que llamar a la herramienta 'make' indicando la plataforma para la cual se desea compilar así como una serie de parámetros opcionales (como la potencia de trasmisión, el canal de radio-frecuencia o el modo de operación: LP-HP).

Un ejemplo de compilación se muestra en la Figura 3.5. En este caso se ha compilado para la plataforma IRIS que, como se describe en la Sección 2.8.1, está compuesta por un microcontrolador ATMEGA1281 y un transceptor de radio ATRF230.

El resultado de la operación es un fichero .exe. Este fichero puede será descargado en la memoria del microcontrolador a través de un script que usa la herramienta *avrdude* para generar el fichero imagen (.hex) con código máquina y que posteriormente se usa directamente para programar el microcontrolador a través del dispositivo *AVRISPmkII* [14], el cual permite descargar este fichero .hex en la flash a través de un puerto USB tal y como muestra la Figura 3.6 en la que se puede observar el dispositivo programador convenientemente conectado tanto al PC como al mote.

```
alvaro@alvaro-VirtualBox: ~/TinyOS2/apps/VersionesT2/DAS220/versionfft
alvaro@alvaro-VirtualBox:~/TinyOS2/apps/VersionesT2/DAS220/versionfft$ make iris
 route,lp
mkdir -p build/iris
    compiling TDAS220C to a iris binary
ncc -o build/iris/main.exe  -Os -fnesc-separator=__ -Wall -Wshadow -Wnesc-all -t
arget=iris -fnesc-cfile=build/iris/app.c -board=micasb -DDEFINED_TOS_AM_GROUP=1
--param max-inline-insns-single=100000 -DLOW_POWER_LISTENING -DLPL_DEF_LOCAL_WAK
EUP=512  -DLPL_DEF_REMOTE_WAKEUP=512 -DDELAY_AFTER_RECEIVE=30   -DTASKLET_IS_TAS
K  -DTOSH_DATA_LENGTH=60 -DRF230_DEF_CHANNEL=26 -DRF230_DEF_RFPOWER=0 -I../../..
/../tos/lib/net/tymo/dymo/ -I../../../../tos/lib/net/tymo/mh   -I../../../tos/li
b/tosboot/avr/ -I Sensors -I SensorDoor -I SensorMov -I SensorPress -I Wire_Chec
k  -I AnSensorMov -I Weight -I Led      -I ExtTemperature   -I IntTemperature -I
 ExtLight  -I Light -I Voltage  -I UserButton -I../../../tos/chips/rf2xx/rf230 -
I../../../tos/chips/rf2xx/layers -I../../../tos/chips/rf2xx/rf230/RF230ActiveMes
sageC   -DIDENT_APPNAME=\"TDAS220C\" -DIDENT_USERNAME=\"alvaro\" -DIDENT_HOSTNAM
E=\"alvaro-VirtualB\" -DIDENT_USERHASH=0xa4faeef8L -DIDENT_TIMESTAMP=0x56fd65eeL
 -DIDENT_UIDHASH=0x43e32b03L -fnesc-dump=wiring -fnesc-dump='interfaces(!abstrac
t())' -fnesc-dump='referenced(interfacedefs, components)' -fnesc-dumpfile=build/
iris/wiring-check.xml TDAS220C.nc -lm
TDAS220M.nc:1924:4: warning: "/*" within comment
/home/alvaro/tinyos2/tos/chips/rf230/RF230RadioC.nc:206:3: warning: #warning "**
* USING LOW POWER LISTENING LAYER"
TDAS220M.nc: In function 'TDAS220M__NotifyMov__notify':
TDAS220M.nc:2968: warning: unused variable 'sensorNumber'
    compiled TDAS220C to build/iris/main.exe
          38610 bytes in ROM
          18308 bytes in RAM
avr-objcopy --output-target=srec build/iris/main.exe build/iris/main.srec
avr-objcopy --output-target=ihex build/iris/main.exe build/iris/main.ihex
    writing TOS image
alvaro@alvaro-VirtualBox:~/TinyOS2/apps/VersionesT2/DAS220/versionfft$
```

Figura 3.5: Ejemplo de compilación para el módulo Iris.

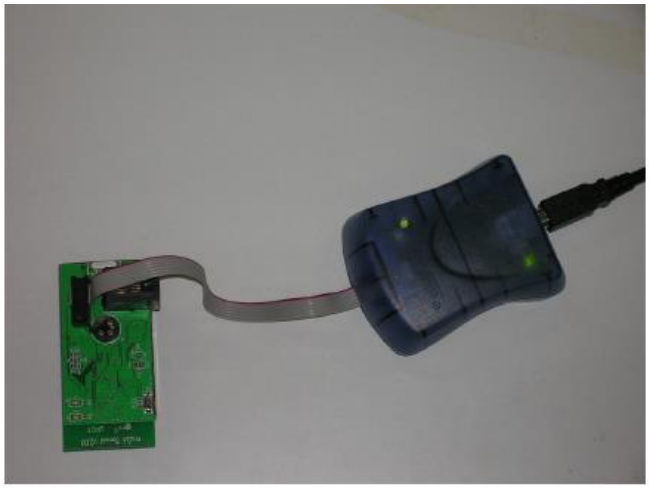


Figura 3.6: Dispositivo AVRISPmkII listo para programar un mote.

3.2 Altium Designer

Existen en el mercado diversas herramientas CAD para afrontar el diseño de dispositivos electrónicos. Entre ellas cabe destacar la herramienta OrCAD, de Cadence Design Systems, y la herramienta Altium Designer, de Altium Ltd.

OrCAD ofrece un conjunto de aplicaciones que permiten afrontar diversas fases del diseño. Sin embargo, Altium, con su última versión de su herramienta CAD, Altium Designer, ha unificado en una sola aplicación la funcionalidad necesaria para afrontar todo el proceso. La interacción directa entre herramientas que se consigue con la aplicación de Altium acelera considerablemente el desarrollo al convertir en nativos los procesos de diseño con bucles de depuración; por ejemplo, modificaciones a nivel de esquemático una vez que el diseño PCB ya ha sido realizado se pueden proyectar sobre la solución anterior con un mínimo impacto y esfuerzo.

La generación de librerías que integran los diferentes niveles de diseño también es una tarea cómoda de realizar sobre la plataforma de Altium, contando además con traductores de librerías que permiten importar _cheros de otras plataformas, como OrCAD.

Dadas las ventajas que presenta, Altium Designer ha sido la herramienta CAD empleada para realizar los diseños electrónicos en este proyecto. Por ello se resumen a continuación algunas características generales de la aplicación, así como una pequeña introducción a su uso.

3.2.1 Características de Altium Designer

Diseño de PCB

Altium Designer ha unificado el diseño de la plataforma física con el diseño de tarjeta de circuito impreso, con soporte para lógica programable. Esto proporciona un sistema de desarrollo totalmente unificado que puede desplegarse a través de todos los elementos del proceso de diseño de productos electrónicos.

Gestión de librerias

Elegir un componente obsoleto o fuera de stock de puede dar lugar a la producción de largos retrasos y sobrecostos. Altium Designer ofrece una amplia gama de herramientas de gestión de datos y recursos de información que le permiten mantener el control sobre el uso de partes.

Diseño para fabricación

Altium Designer ayuda a reducir la brecha entre el diseño y fabricación, y le permite administrar de forma activa la generación y verificación de todos los datos de fabricación, ahorrando tiempo y minimizando los costosos errores durante el flujo de diseño.

Dispositivos programables

Altium Designer es la única herramienta que permite explotar plenamente el potencial que ofrecen hoy los dispositivos programables, al unificar el diseño de FPGAs, el desarrollo de software y el diseño de PCBs, en una sola plataforma.

Diseño Unificado de FPGA/PCB

Los dispositivos programables son cada vez más usados en desarrollo electrónico. Este tipo de dispositivos abren nuevas posibilidades de diseño y aportan beneficios significativos para el proceso de diseño, lo que permite que la complejidad funcional se traslade de dispositivos discretos cableados al ámbito de dispositivos programables.

Gestión de todo el proceso de desarrollo

Altium Designer unifica todo el proceso de diseño y permite gestionar todos los aspectos del desarrollo dentro de un único entorno de diseño, y ofrece una infraestructura de administración de proyectos y documentos unificada que soporta la convergencia de diseños de tradicionalmente disciplinas separadas.

3.3.2 Indicaciones de Uso

Las herramientas de Altium Designer de las que se hace uso en este proyecto van desde el diseño CAD a nivel de esquemático hasta la generación de ficheros de fabricación, o gerbers. Esto implica el diseño de hojas de esquemático (Schematic Sheets), diseño de PCB (PCB Document) y el uso de los generadores de documentación de salida (Output Files); así como el empleo de las librerías de componentes incorporadas en la aplicación y el editor de librerías para los componentes no incluidos.

Tal y como se explica con detalle en la guía de Altium [15], el proceso habitual para la realización de un diseño completo comienza con la elaboración de un boceto del esquema del circuito sobre el papel. Una vez que te tiene una idea aproximada del

esquemático, se procede a su realización sobre una hoja de esquemático de Altium Designer. En primer lugar se añaden los componentes haciendo uso de las librerías de altium o generando librerías propias si es necesario. Una vez localizados los componentes se procede a su distribución y colocación. Para diseños de cierta complejidad, no necesariamente mucha, es recomendable agrupar los componentes por grupos funcionales; realizando las conexiones entre componentes del mismo grupo haciendo uso de cables y las conexiones entre distintos grupos haciendo uso de etiquetas. En la Figura 3.7 se muestra como ejemplo el esquemático de un circuito sencillo con un temporizador que hará parpadear un led.

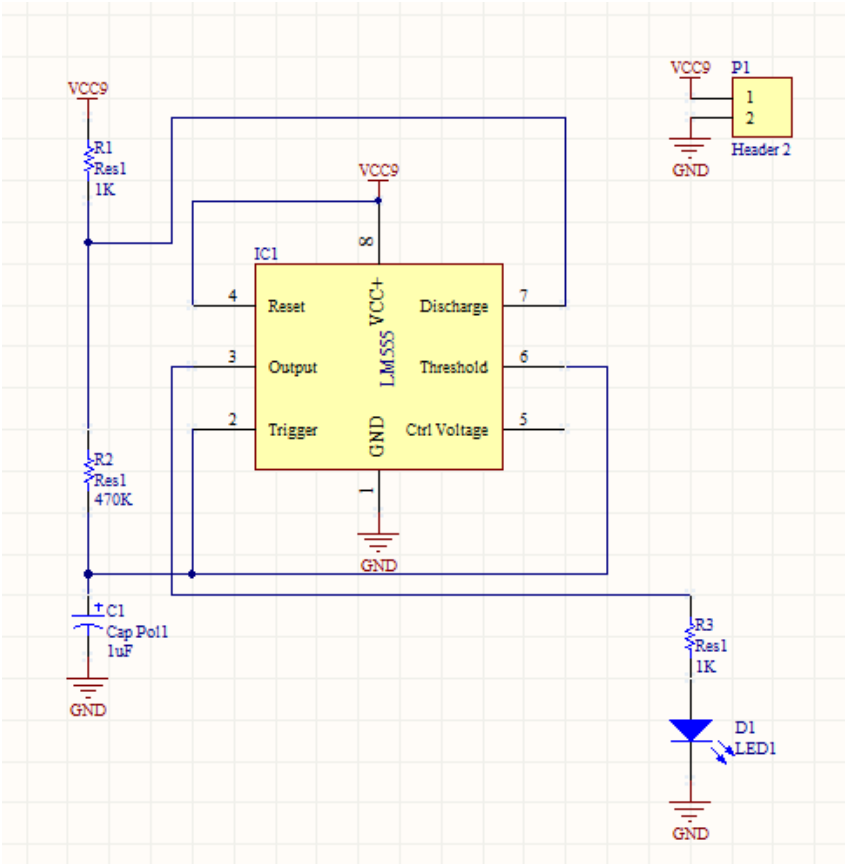


Figura 3.7: Ejemplo de esquemático diseñado en Altium Designer

El siguiente paso podría ser la realización de simulaciones sobre el circuito. Los componentes de librería de Altium Designer incorporan ficheros de simulación. En este proyecto no se han realizado simulaciones dado que la mayor parte de procesos están gobernados por un microcontrolador y, además, los bloques funcionales añadidos resultaban suficientemente sencillos como para no ser necesaria la simulación.

Una vez terminado el diseño a nivel de esquemático, o primera aproximación a éste, se procede a la creación de un documento PCB (*PCB Document*) sobre el que proyectar el circuito. Antes de poder importar el diseño desde el nivel de esquemático hasta el nivel PCB hay que seleccionar los componentes reales (fabricante, modelo) que se pretenden emplear en el montaje final. Cada componente de librería a nivel de

esquemático tiene asociada una huella o *footprint* a nivel de PCB. Una vez seleccionados hay que verificar que la huella o *footprint* asociada a cada componente coincide con la huella del modelo real que se pretende utilizar. En caso contrario, será necesario buscar la huella adecuada o, en la mayoría de casos, realizar el diseño de la misma mediante la herramienta de edición de librerías. Es muy común que los diseñadores tengan una librería personal en la que incorporan las huellas de los componentes que emplean y que, generalmente, reutilizan en sucesivos diseños.

Cuando el esquemático está terminado y todos los componentes tienen asociada la huella apropiada se procede a importar el diseño desde el nivel de esquemático hasta el de PCB. Este proceso de importación puede repetirse más adelante para importar cambios incluso aunque el diseño PCB esté avanzado; suele ser inmediato cuando la hoja PCB está en blanco y puede requerir de indicaciones del diseñador cuando se trata de importar cambios.

El importador sitúa los componentes alineados en fila sobre la placa. El diseñador debe prestar especial atención a la hora de colocar los componentes. Una buena colocación facilitará enormemente el trazado de pistas que se realizará a posteriori, mientras que una colocación ineficiente puede desembocar en un circuito imposible de enrutar. Por lo general, la mejor forma de afrontar esta tarea es agrupar los componentes, fuera de la zona de placa, por grupos funcionales. La mayor parte de conexiones se realizan entre componentes dentro de cada grupo funcional; una organización a nivel de PCB que respeta los grupos minimiza la longitud de las pistas y el entramado de éstas. Una vez que los componentes de cada grupo se han dispuesto de forma óptima, se procede a la distribución dentro de la placa. Esta colocación se realizará atendiendo a diversos factores. Se suele considerar la disposición de los conectores en la periferia de la tarjeta, la simplificación de conexiones entre grupos y el ruido generado por señales de RF; especialmente en este tipo de dispositivos que incorporan comunicación a 2.4GHz en tarjetas del menor tamaño posible.

Sólo cuando los componentes están óptimamente ubicados en la tarjeta es recomendable comenzar con el trazado de pistas, o enrutado. La herramienta CAD facilita esta labor mediante el enrutado interactivo; cuyas características se pueden configurar al gusto del diseñador. Por norma general, se deja sin enrutar la red de masa (GND) para conectarla al final mediante un plano en cada cara. En la Figura 3.8 se muestra un ejemplo de diseño a nivel de PCB haciendo uso de esta herramienta.

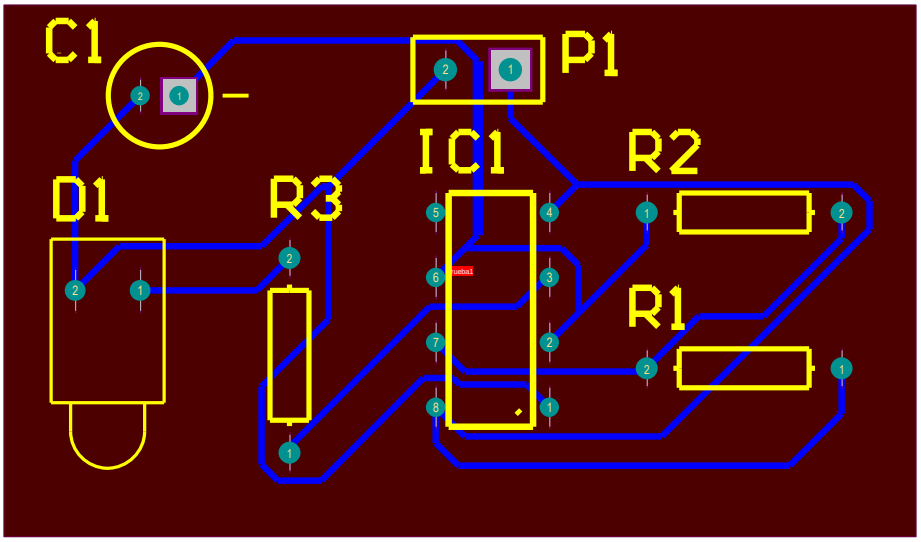


Figura 3.8: Ejemplo de PCB diseñado en Altium Designer

Una vez que parece que el diseño está terminado es necesario ejecutar el comprobador de reglas *(Rule Check*) que nos indicara si hemos dejado pistas sin conectar o si existe alguna otra violación de reglas como proximidad de pistas y componentes o incluso cortocircuitos. Habiendo realizado el enrutado mediante la herramienta interactiva lo más probable es que el diseño sea correcto. No obstante, es común que a la hora de generar el plano de masa se hayan formado islas inconexas. Esta situación se resuelve colocando vías que conecten estas islas con el plano de masa de la otra cara del circuito. En ocasiones resulta necesario modificar el trazado de alguna pista o incluso desplazar ligeramente un componente para permitir la expansión del plano y resolver estas situaciones.

Un diseño que supera el *Rule Check* se puede considerar listo para su fabricación. En ese momento se hace uso de los generadores de ficheros de salida para obtener los ficheros de fabricación, principalmente *gerbers* y descriptores de taladros.

No tiene cabida en este documento una explicación más detallada del uso de la aplicación Altium Designer. Los pasos descritos anteriormente, a excepción de algunas recomendaciones que autor ha intercalado, están convenientemente detallados e ilustrados en la guía on-line ofrecida por Altium [15].

3.3 Protomat C60 y Boardmaster

Tras el diseño a ordenador del circuito será necesaria la implementación física del prototipo. Para ello se utilizará el plotter fresador Protomat C60 que nos proporciona la empresa LPKF Laser & Electronics AG.

Fundada a mediados de los 70, LPKF se forjó rápidamente un nombre en Garbsen, cerca de Hannover, con los procedimientos no convencionales de producción de prototipos de placas de circuito impreso. La tecnología de fresado dirigida por CAD se estableció en el mercado como una alternativa, eficiente y respetuosa con el medio ambiente, a la corrosión por ácido. Entonces era una revolución, hoy día es un estándar aceptado.

3.3.1 Pasos previos a la impresión

El fabricante recomienda [16] encarecidamente que, antes de realizar cualquier trabajo con la fresadora, se haga un estudio sobre su mantenimiento, funcionamiento y que haya una familiarización con el software que controla la máquina: el BoardMaster, proporcionado también por LPKF.

Este software de control automático requiere unos archivos específicos llamados LMD, creados a partir de los *gerbers* generados en Altium por el software CircuitCam (proporcionado también por LPKF). Estos archivos contienen la información

necesaria que usará el BoardMaster para identificar vías, pistas y su tamaño así como la fresa o broca recomendada en cada fase de la producción. Las principales fases a seguir en CircuitCam para obtener el fichero LMD serían las siguientes:

- *Importar gerbers*: Se importan los ficheros de fabricación necesarios para realizar la impresión de la placa. Es posible que no se necesiten todos, eso dependerá de cada diseño, pero se puede decir que los básicos tendrían que ser los relativos a las pistas de la capa top, las pistas de la capa bottom, los taladros y los límites del prototipo para recortarlo y separarlo de la plancha de cobre en la que se imprime. Si se quisiera también se podrían importar archivos relativos al rotulado de la placa, por ejemplo los nombres de los componentes o sus límites, siempre teniendo en cuenta que si están encima de alguna pista o vía, éstas podrían ser cortadas.

- *Insulate (aislar):* En esta fase el software procesa las capas top y bottom para que el cobre sobrante sea eliminado por la fresadora.

- *Guardar:* Se guarda el proyecto como un archivo .CAM.

- *Exportar LMD*: El fichero .CAM puede ser exportado con el formato LMD para ser procesado por el software BoardMaster y controlar el proceso de impresión del prototipo.

3.3.2 Indicaciones de uso de BoardMaster

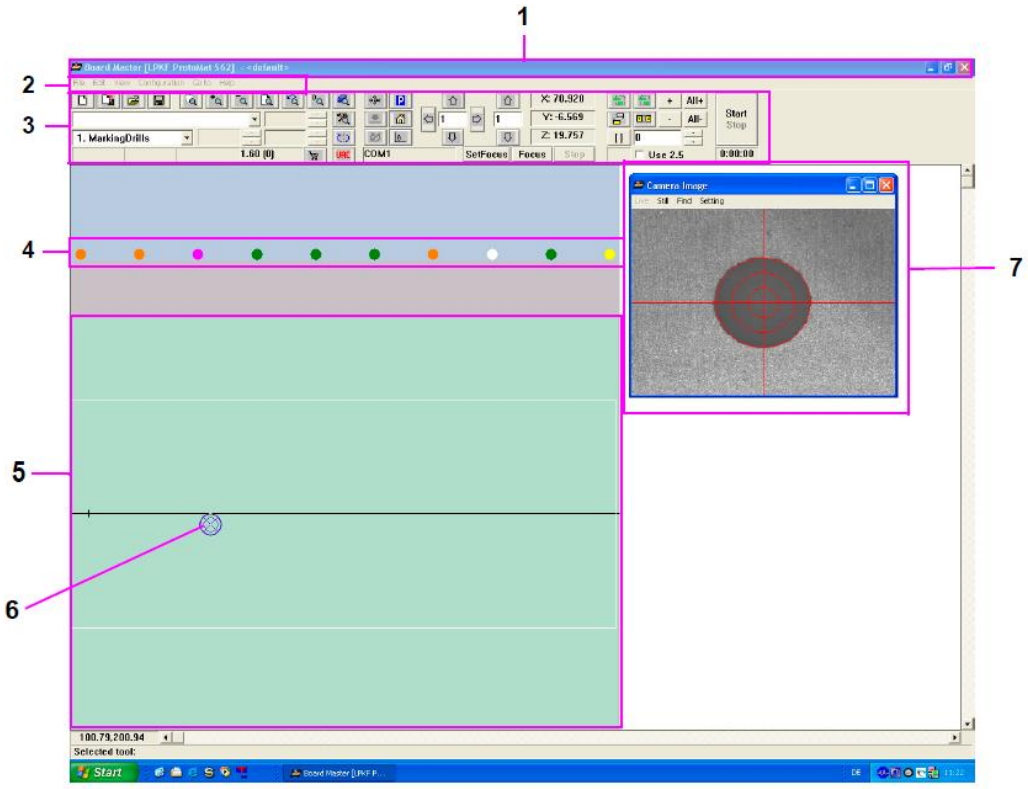
Al abrir el software aparece la interfaz de operaciones, en ella se distinguen varias áreas tal y cómo se aprecia en la Figura 3.9. De las áreas que se ven, la principal en el proceso de impresión, es la de *workspace*. En ella se mostrará el circuito a fabricar, la posición del mismo respecto a unas coordenadas y la posición del cabezal de la fresadora.

Si la fresadora no dispone de cámara para obtener coordenadas fiduciales habrá que ajustar la plancha a un eje de simetría para que al voltearla y que realice el proceso por la otra cara sea totalmente simétrica. En el caso de que sí tuviera, se realizan unos taladros de referencia para las coordenadas y no haría falta tener en cuenta el eje de simetría.

Al igual que con la cámara, el Protomat C60 no consta de cambio automático de herramienta pero existen modelos que sí lo tienen. Tan sólo habría que indicarle al BoardMaster la posición y el tipo de herramienta y la fresadora escogerá de forma automática cada taladro o fresa en función de la fase en la que se encuentre.

Estos taladros son muy delicados y tienen un tiempo de vida determinado. Se recomienda no tocar nunca con los dedos las puntas de las herramientas y que se conserven en sus blíster para que no se dañen y realicen el trabajo con precisión.

Éstos son algunos de los conceptos básicos que hay que tener en cuenta para trabajar con este tipo de máquinas. Para más detalles y más información, en la web se pueden ver los manuales de los softwares BoardMaster y CircuitCAM [17, 18] así como el manual de instrucciones de la fresadora [16].



1- title bar
2- menu bar
3- function bar
4- tool change position (optional, only *ProtoMat* S62/S100)
5- available workspace
6- blue crosslines indicates the actual position of the drill / mill head
7- camera window (optional, *ProtoMat* with camera system)

Figura 3.9: Apariencia del software BoardMaster en ejecución.

Cuatro

Diseño e implementación del prototipo de nodo sensor

4.1 Estudio previo del método de clasificación de señales propuesto

Antes de comenzar con el diseño del nuevo nodo sensor, se debe comprender el método que se propuso en la tesis Wireless Sensor Networks for Ambient Intelligence [20]. Este método propone clasificar las señales del sensor PIR mediante dos parámetros: distancia y velocidad del sujeto con el sensor como punto de referencia. Las señales de varios sensores PIR y varios sujetos sirvieron para generar un patrón. Los sujetos se desplazaban a diferentes velocidades y a diferentes distancias del sensor y se muestreaban las señales a una frecuencia de 10Hz, así que se puede concluir que los patrones están indexados mediante estos dos parámetros y sirven para clasificar otras señales que se obtengan de los sensores PIR.

Los datos generados para los patrones fueron tomados por 5 sensores, haciendo 4 repeticiones, a 7 distancias (de 0.5m a 3.5m con intervalos de 0.5m) y 4 velocidades distintas (de 200mm/s hasta 800mm/s en intervalos de 200mm/s). En total se obtuvieron 560 señales que fueron fusionadas quedando entonces 28 señales distintas en el conjunto patrón.

Una vez se tiene el patrón es posible clasificar las señales generadas por otros sujetos comparándolas con él y observando su grado de similitud gracias al siguiente proceso:

1. Se elimina la componente DC de ambas señales.

2. Las señales se alinean usando el desfase para obtener la máxima correlación.

3. Esta correlación se multiplica por el resultado de la correlación de las FFT (Fast Fourier Transform) de las señales.

4. El resultado es multiplicado por la correlación resultante de las derivadas de las señales.

5. El resultado se multiplica por la correlación bidimensional de la distribución Wigner-Ville de las señales.

6. El valor resultante se guarda como el que mayor grado de similitud tiene con una de las señales patrón.

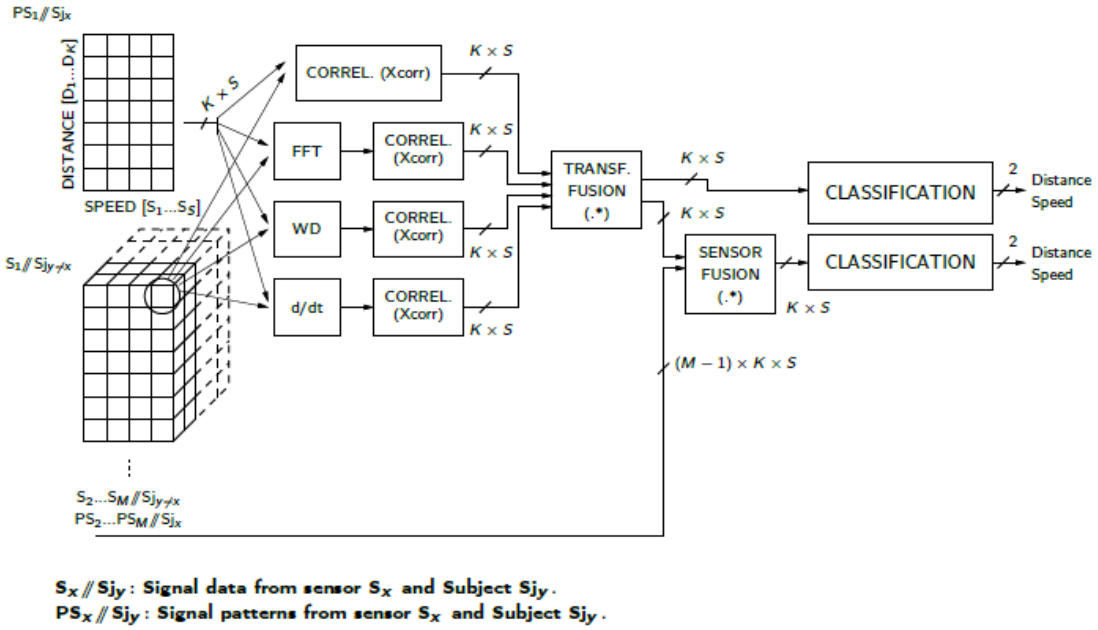


Figura 4.1: Fases del proceso de clasificación de señales

En la Figura 4.1 se muestra en detalle el método de clasificación recién explicado atendiendo al tamaño y cantidad de señales involucradas en él. También se añaden al algoritmo los resultados de una futura malla de nodos sensores, que hayan capturado el mismo evento, fusionándolos entre sí para obtener como salida una clasificación de los parámetros velocidad y distancia mucho más precisa. Es durante el estudio de este algoritmo cuando se detecta el problema al que este proyecto de fin de carrera pretende dar solución. El microcontrolador no dispone de suficiente memoria RAM para atender las necesidades de cálculo de las operaciones y a la vez las del resto de la aplicación. Es fácil de comprobar realizando los siguientes cálculos:

El tamaño de la memoria RAM interna es de 8KB (8Kb x 8), la señal analógica capturada contiene 128 muestras, de 32 bits cada una, a la que hay que aplicar la operación FFT. El resultado de una FFT produce números complejos lo que quiere decir

que por cada muestra se generarán dos números de tipo flotante, de 32 bits, ya que los resultados contienen números decimales que serán guardados en una tabla. Si calculamos el tamaño de esa tabla se obtiene un total de 8192 bits (8Kb).

A estos cálculos de memoria habría que sumar la tabla del conjunto patrón. Este está formado por 28 señales guardadas en tablas del mismo tamaño que la de la señal muestreada. Una vez aplicada la FFT se tendrá como resultado un consumo añadido de memoria de unos 230Kb. El tamaño necesario llegado este punto ya es de 238Kb, superando con creces la capacidad de la memoria RAM interna. Es muy probable que durante la programación del algoritmo se creen más tablas pero no tan grandes como las que se exponen en estos cálculos por lo que las necesidades del algoritmo no serán mucho más altas. Además, el espacio ocupado en la memoria se iría liberando para poder ir realizando las sucesivas iteraciones del algoritmo de clasificación, explicado en la Sección 4.5.

A la hora de programar las operaciones habrá que controlar siempre los escasos recursos de memoria de los que se dispone que, realizando una estimación, se resumen en que la RAM externa que se adquiera debe tener un tamaño de 512Kb, (64Kb x 8).

4.2 Selección de los componentes electrónicos

4.2.1 Integrados

Una vez se conoce qué algoritmos se van a implementar en el microcontrolador se puede hacer una estimación de las necesidades hardware. Tomando como base el prototipo DAS220.4 se escoge el mismo microcontrolador, el Atmega 1281 mostrado en la Figura 4.2, por ser el que mejor se adapta a las necesidades de consumo energético y potencia. Debido a la cantidad de señales, y a las operaciones realizadas con ellas, se consumiría toda la memoria RAM interna del microcontrolador, tal y como se demuestra en la sección anterior, por lo que hay que añadir una memoria RAM externa para suplir esta carencia.

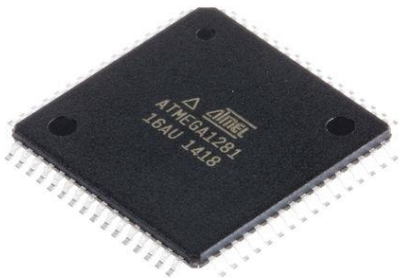


Figura 4.2: Microcontrolador ATMEGA 1281

Según la hoja de características [9], el Atmega 1281 consta de 8Kbytes de RAM interna pero tiene la capacidad de ampliar su capacidad gracias a su interfaz de memoria externa. Cuando esta interfaz es activada el microcontrolador genera las señales adecuadas para controlar una RAM externa que puede llegar a ser de 512Kbits de memoria debido a los 16 pines de direccionamiento que el microcontrolador usa con este único propósito al activar la interfaz y a los 8 bits de la arquitectura del mismo.

La nueva memoria es mapeada a continuación de la memoria RAM interna y ambas forman un único bloque de RAM como se ve en la Figura 4.3. El acceso a la memoria externa se produce cuando se alcanza la última dirección de la RAM interna y la única diferencia en cuanto a operatividad es que el microcontrolador necesita de un ciclo de reloj adicional para leer o escribir en esta memoria.

Una vez se conocen cómo funciona la interfaz de memoria externa y las necesidades para el proyecto se busca en el mercado una RAM acorde a estas características. Hoy en día es difícil encontrar una RAM económica para diseño de circuitos de un tamaño tan reducido como el que el microcontrolador es capaz de direccionar, pero eso no es problema. Se escoge una RAM de 1Mbit de capacidad (128Kx8) en la que el bit más significativo se capará conectándolo a masa quedando así los 16 bits restantes de direccionamiento para ser conectados a los pines del microcontrolador. Esta RAM es la CY62128EV30 de Cypress Perform que, además de cumplir con los requisitos de memoria, también cumple con las especificaciones de consumo y velocidad que la hacen compatible con el Atmega1281.

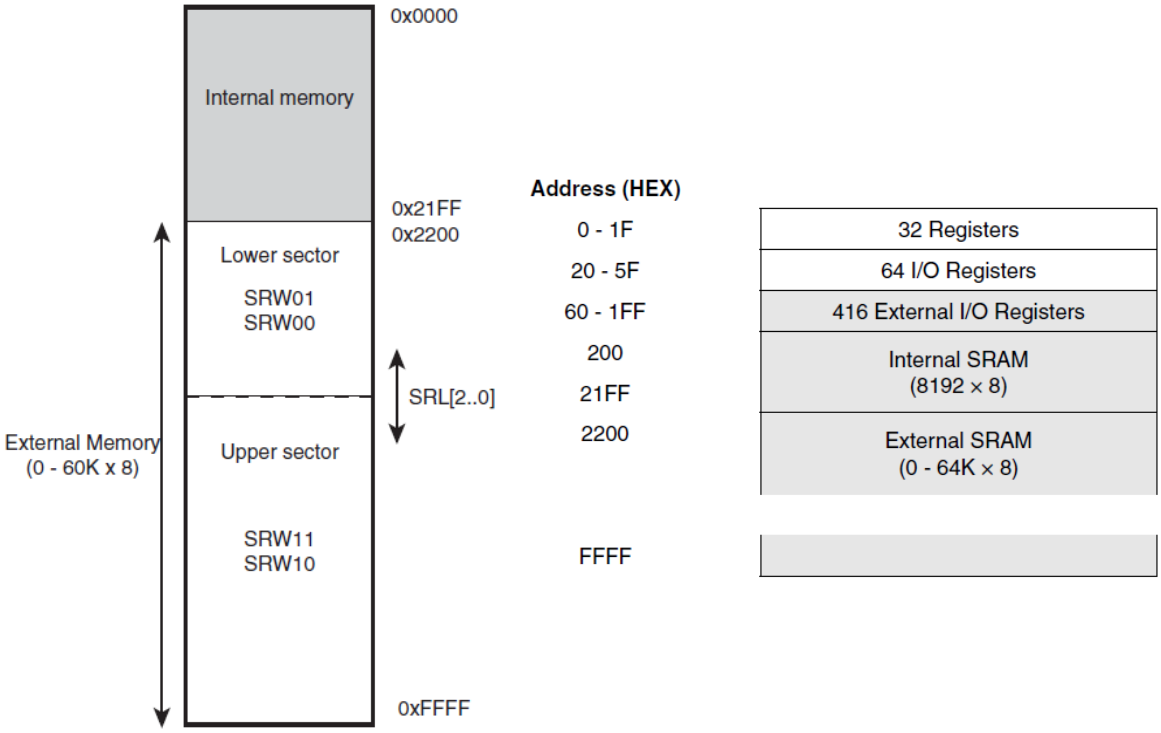


Figura 4.3: Mapa de memoria RAM y mapa de memoria completa del dispositivo diferenciando entre interna y externa.

Si se atiende a las especificaciones recogidas en la hoja de características del Atmega1281 [9], se puede ver que es necesario añadir un latch D debido a que 8 pines del microcontrolador se usarán tanto de bus de direcciones como de bus de datos para la RAM como se detalla en la Figura 4.4. Debido a las operaciones a alta velocidad que se realizan en la RAM, el fabricante recomienda usar un latch con unas frecuencias superiores a 8MHz a 4V y 4MHz a 2.7V. La serie de latches 74AHC es perfecta para este tipo de sistemas por lo que para este proyecto se elige específicamente el Octal latch de tipo D 74AHC573.

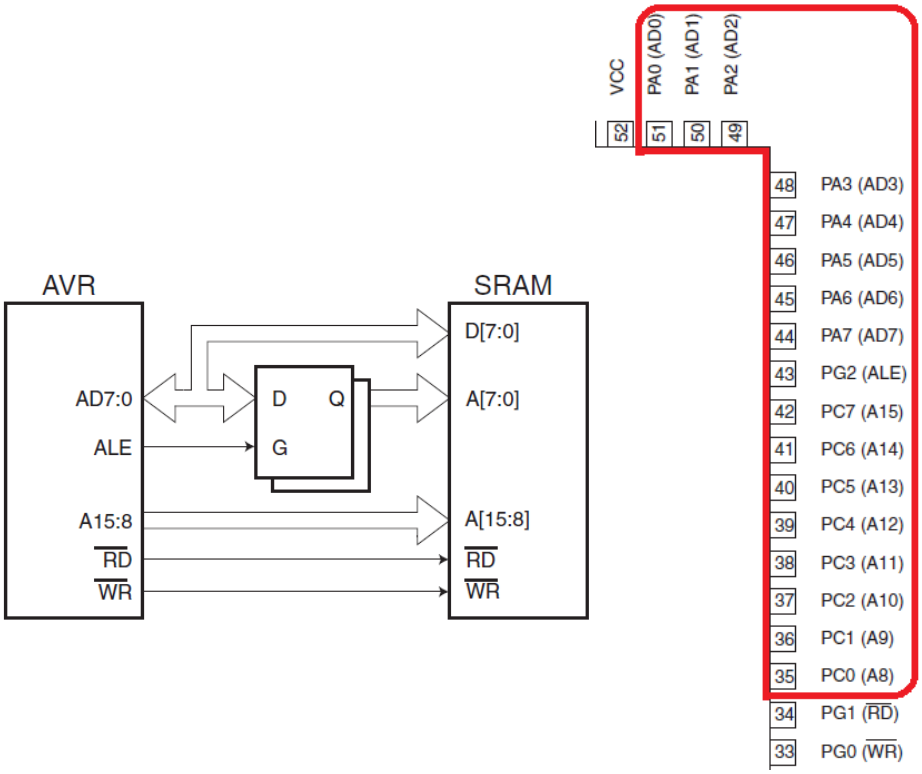


Figura 4.4: Esquema de montaje del módulo de RAM externa y pineado específico del microcontrolador para direccionamiento y datos de la memoria externa.

4.2.2 Dispositivos Sensores de movimiento

Para detectar movimiento hay una gran variedad de dispositivos y tecnologías diferentes. Fundamentalmente se dividen en sensores pasivos y sensores activos. Entre los activos destacan los sensores por láser y por ultrasonido. Los sensores pasivos se reducen básicamente a los sensores de tipo PIR (Pasivo Infra-Rojo), que detectan variaciones en el infrarrojo térmico emitido por los objetos en su campo de visión. Teniendo en cuenta las restricciones energéticas del dispositivo final, el mínimo consumo es una necesidad fundamental, por lo que se considera inaceptable el consumo de los dispositivos de tipo activo. La elección se reduce, por tanto, a los dispositivos PIR ofrecidos por el mercado. Es conveniente destacar que a pesar de

catalogarse como pasivo, ya que la magnitud ambiental se mide de forma pasiva, el sensor de movimiento es un dispositivo activo.

Este dispositivo debe estar diseñado para operar a tensiones entre 2V y 3 V; ya que este es el rango de operación de las baterías. Con estas restricciones se encontró un único fabricante que ofrece sensores de movimiento PIR para voltajes inferiores a 5V. Los dispositivos seleccionados son los sensores PIR digitales de la familia AMN de Panasonic. Estos dispositivos tienen un alto nivel de integración unos índices de consumo realmente bajos.

Es importante tener en cuenta que este tipo de dispositivo es compatible con el planteamiento de monitorización por eventos. Un sensor analógico requerirá de una constante monitorización, necesitando que el microcontrolador estuviese permanentemente activo pero se solucionará mediante software como se explicará en la Sección 4.4.7. El sensor de movimiento digital seleccionado puede activar las interrupciones externas del microcontrolador en los momentos en que se ha producido un evento por movimiento. Esta idea se utilizará para poder disparar la monitorización del sensor analógico cuando sea necesario y permitiendo entrar al microcontrolador en reposo cuando no se produzcan estos disparos. En definitiva, los sensores seleccionados son, como sensor digital, el AMN44121 y, como sensor analógico, el AMN22112 [19].

Figura 4.5: Fotografía de los sensores de movimiento analógico y digital

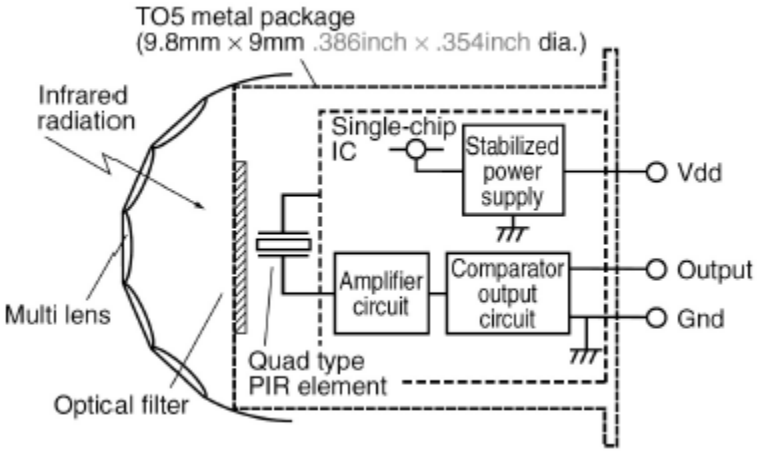


Figura 4.6: Diagrama de un sensor de movimiento analógico

4.3 Pasos previos al prototipado del dispositivo Nodo Sensor

El objetivo final del trabajo descrito en esta sección es la obtención de un prototipo de nodo sensor que cumpla con los requisitos indicados anteriormente. Para ello se parte de la solución ofrecida en la versión del modelo DAS220.4 del hardware y de la versión TDAS220.2.0.5 del software realizando modificaciones que van aproximando el modelo al prototipo final que cumplirá con las especificaciones mencionadas en la definición del proyecto.

4.3.1 Adaptación del software al nuevo módulo de sensor de movimiento

Antes del desarrollo del nuevo hardware y una vez se dispone de los sensores de movimiento, se procede a comprobar, sobre el dispositivo DAS220.4, el nuevo sensor de movimiento analógico. Para ello se desarrolla el nuevo módulo software de la aplicación relativo al sensor de movimiento analógico. Los datos de este sensor se deben capturar a través del ADC (*Analogic to Digital Conversor*). El microcontrolador dispone de varios pines dedicados a entradas de datos analógicos que pasan por el conversor analógico-digital para ser almacenados y/o tratados por el microcontrolador.

El ADC convierte una señal analógica de voltaje en un valor digital de 10 bits de aproximación, es decir, los valores de las amplitudes de voltaje de la señal analógica se representarán en una escala que va de 0 a 1023. Este valor digital se conserva en uno de los registros del ADC hasta que es sustituido por otra muestra después del siguiente ciclo de conversión. El microcontrolador puede acceder a estos registros e ir almacenando en su memoria los valores que en ellos se guardan para así, por ejemplo,

obtener una serie de muestras de una señal analógica que podremos procesar según se requiera.

El dispositivo DAS220.4 cuenta con unos puertos auxiliares, dos de ellos conectados a los pines específicos del ADC del microcontrolador. Se usa el puerto PF6 (o ADC6), en la conexión denominada como AUX_AN0, y se configura como entrada del módulo del sensor analógico en TinyOS. Se fabrica entonces una rudimentaria interfaz entre el sensor y el circuito con un portasensor y unos cables de cobre soldados a sus patillas y se coloca el sensor analógico como muestra la Figura 4.7.

Los cambios realizados en el código de la aplicación son los básicos en esta ocasión para introducir, en la fase de lectura de los sensores, el tratamiento del evento producido por la captura con éxito de un dato por el módulo sensor analógico e insertando ese dato dentro de la trama que se envía a la estación base.

Durante la fase de lectura de sensores, que se compone por una serie de funciones y eventos que se ejecutan cuando se lanza la tarea *sensingStart()*, los diferentes sensores que se incluyen en el DAS220.4 toman muestras de forma secuencial y son procesadas para su análisis.

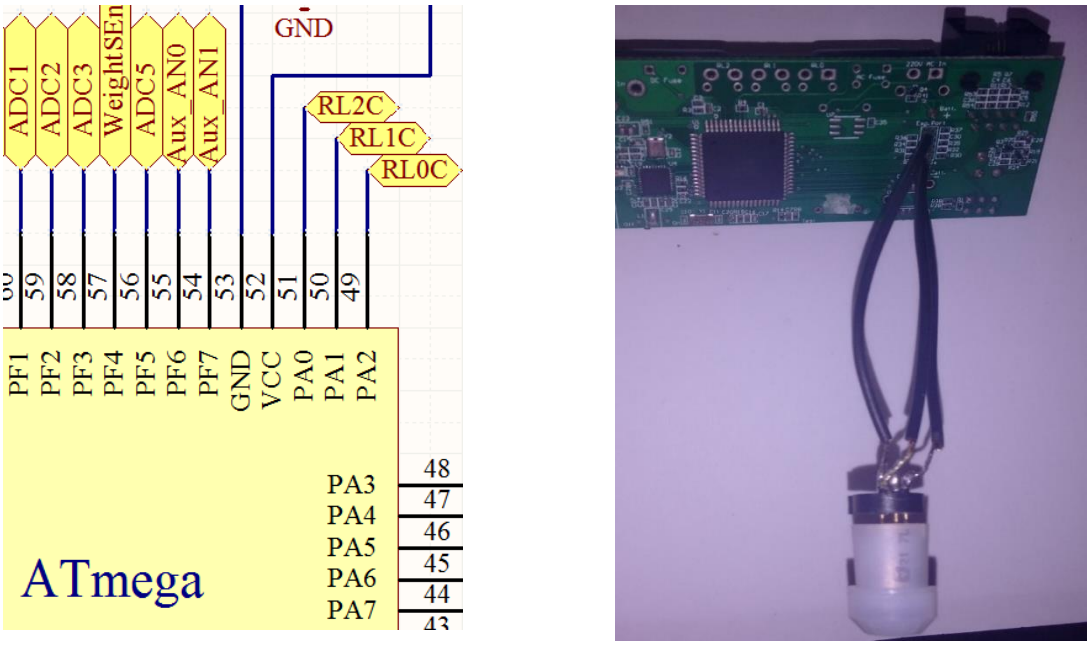


Figura 4.7: Puertos auxiliares e interfaz auxiliar para el sensor de movimiento analógico.

Comprobaciones realizadas sobre el nuevo módulo de sensor de movimiento analógico

Como se comentó en la Sección 4.1, las señales se muestrean a una frecuencia de 10Hz por lo que se debe comprobar si el ADC incluido en módulo del sensor analógico es capaz de capturar muestras a esa velocidad. Para ello se usa una de las salidas del componente auxiliar del DAS220.4 y en la aplicación se programa para que el nivel que haya en el pin del microcontrolador conectado a esa salida vaya alternando de alto a bajo y viceversa conforme se vaya adquiriendo cada muestra. Esto se consigue gracias al evento que lanza la interfaz de lectura del ADC que permite ejecutar el comando *toogle()* cambiando el estado en el que se encuentre una salida cada vez que se produce una captura de forma exitosa.

Se puede observar en la siguiente captura de osciloscopio cómo se producen flancos cada 100ms por lo que podemos concluir que las muestras se toman con una frecuencia de 10Hz tal y como requieren las especificaciones del algoritmo de clasificación. Como apunte, añadir que para esta prueba se programó a la aplicación para que se tomasen seis muestras, de ahí que tan sólo se aprecien seis flancos en la Figura 4.8.

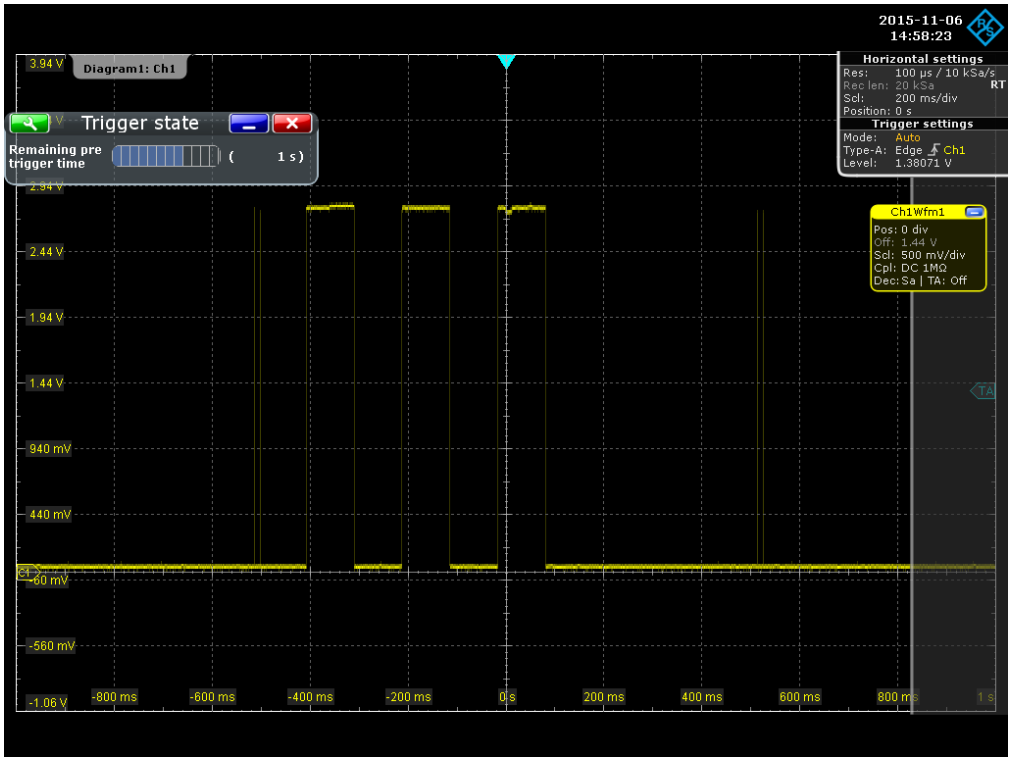


Figura 4.8: Captura comprobación de la frecuencia de muestreo.

La segunda comprobación que se realiza es para asegurar que los datos muestreados son coherentes. Se añade un array formado por las muestras que se capturan del sensor analógico y es añadido a la trama donde se envían los datos del resto de sensores codificados en hexadecimal y se realizan los siguientes pasos:

1 Se conecta la salida del sensor analógico a masa y se comprueba que las muestras recibidas son todas 0x0000.

2 Se conecta la salida del sensor analógico a VCC y se comprueba que las muestras recibidas son todas 0x03FF, es decir, 1023. Dato coherente ya que, como se explica en la sección anterior, el ADC que incluye el microcontrolador dispone de una precisión de 10 bits.

4.3.2 Adaptación del hardware al nuevo prototipo

Al igual que con la adaptación software, se parte de la solución DAS220.4. Para el desarrollo de este PFC no es necesario incluir todos los módulos con los que cuenta la solución previa por lo que se procede a hacer un estudio del esquemático, para una toma de decisiones adecuada, que se incluye en el Apéndice, en la Figura 6.1.

- *Módulo de radio*: Se decide que el módulo de radio no irá incorporado en la misma placa que el resto del circuito. La principal razón es que el integrado AT86RF230 tiene unas dimensiones tan reducidas que sería muy costoso tener que soldarlo a mano varias veces si la fabricación del circuito resultase defectuosa. Así pues, se creará un circuito aparte con el módulo de radiofrecuencia y su antena con un conector macho que se acoplará a la placa principal la cual será rediseñada para poder contener un conector hembra específicamente para este propósito. Como resultado se podrá reutilizar el módulo de radio en todos los prototipos futuros ahorrando recursos tanto económicos como temporales.

- *Sensores externos*: No son necesarios en el desarrollo del proyecto por lo que se pueden obviar en el diseño así como la interfaz externa. Estos sensores son los de presión, peso y los de las puertas entre otros.

- *Serial ID*: Tampoco es necesario este módulo. Al no estar destinado el prototipo a una producción comercial no necesita un número de serie único.

- *Actuadores*: Según la funcionalidad que se busca en el circuito, los relés no son necesarios pues no conectaremos ningún dispositivo que los requiera.

- *Módulos Dataflash*: No serán utilizados por lo que se eliminan del diseño.

- *Fuente de alimentación*: Se modifica para que quede lo más simplificada posible. Tan sólo se usarán baterías AA, y no una fuente externa, para alimentar al circuito así que se eliminan los componentes no necesarios y quedan el conector de las baterías y el interruptor.

- *Sensores integrados en el circuito*: El módulo del sensor de movimiento digital sí que será necesario para la finalidad de este proyecto por lo que se mantiene en el diseño. También se decide mantener otros sensores, como los de temperatura, por si fueran necesarios en un futuro.

- *Módulo auxiliar*: Se deja en el diseño por si hiciese falta en un futuro utilizar algún pin del microcontrolador para alguna tarea auxiliar.

- *Osciladores*: Son indispensables para el correcto funcionamiento del microcontrolador.

- *Conector de programación*: Como su nombre indica, es la interfaz por la que se programará el microcontrolador, por lo que se deja incluida en el diseño.

En resumen, el esquemático que queda después de estos cambios, que se puede ver en el Apéndice, Figura 6.5, está pensado para cumplir los requisitos propuestos para este PFC, aunque no es definitivo puesto que aún falta introducir en él el módulo de memoria RAM externa y los cambios en las conexiones y pineado del microcontrolador.

4.3.3 Diseño del módulo de memoria RAM

Tal y como se explica en la Sección 4.1, es necesaria una RAM externa para el propósito que se plantea en este proyecto, el de poder realizar una serie de operaciones en las que se involucra una gran cantidad de datos. En esa sección también se explica que existen unos pines del microcontrolador destinados exclusivamente al intercambio de datos entre la RAM y el propio microcontrolador y varias señales de control para la memoria y el latch que requiere. En este apartado se detalla cómo crear una nueva librería en el software Altium que contenga los dispositivos introducidos en el módulo de RAM externa.

Cuando se active la interfaz de memoria externa dentro del microcontrolador, los pines que se muestran en la Figura 4.4 dejarán de realizar la función para la que estaban programados en la versión anterior y pasan a ser usados por la RAM. Es por esto que se necesitará cambiar esas entradas y salidas a otros puertos que queden libres en el microcontrolador.

Una vez programados los cambios en los puertos tanto a nivel de aplicación como a nivel de diseño en Altium, el siguiente paso es implementar el esquemático y el *footprint* (o huella) de la memoria RAM y del octal latch e incluirlos en una librería de componentes de Altium para que estén disponibles en el diseño del prototipo.

La primera parte en la creación de una nueva librería es a nivel de PCB (Printed Circuit Board). En el nivel PCB se diseña el *footprint* del componente, es decir, los pads

del integrado que se dibujarán en el circuito impreso como muestra la Figura 4.11. Para ello se deben tener muy en cuenta las medidas que proporciona el fabricante en la hoja de características y en las unidades en las que se encuentra mostradas en la Figura 4.9, normalmente en unidades del sistema métrico internacional o pulgadas. Después de tener ya definido el *footprint* se le asocia la figura del esquemático que representará a este componente cuando se diseñe el circuito en este nivel. Los diseños resultantes de esquemático y PCB de la RAM y el Latch se pueden ver en las Figuras 4.10 y 4.14 respectivamente.

Para la memoria RAM se proporcionan las medidas en pulgadas y en milímetros. En Altium se puede diseñar en ambas unidades, para mayor comodidad se hará con esta última opción.

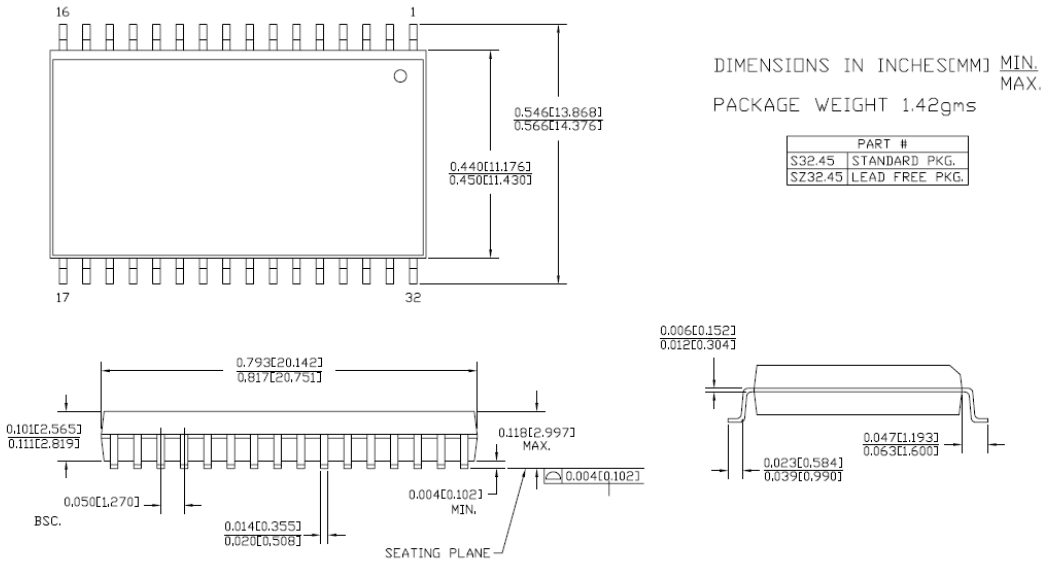


Figura 4.9: Medidas del circuito integrado de la memoria RAM externa

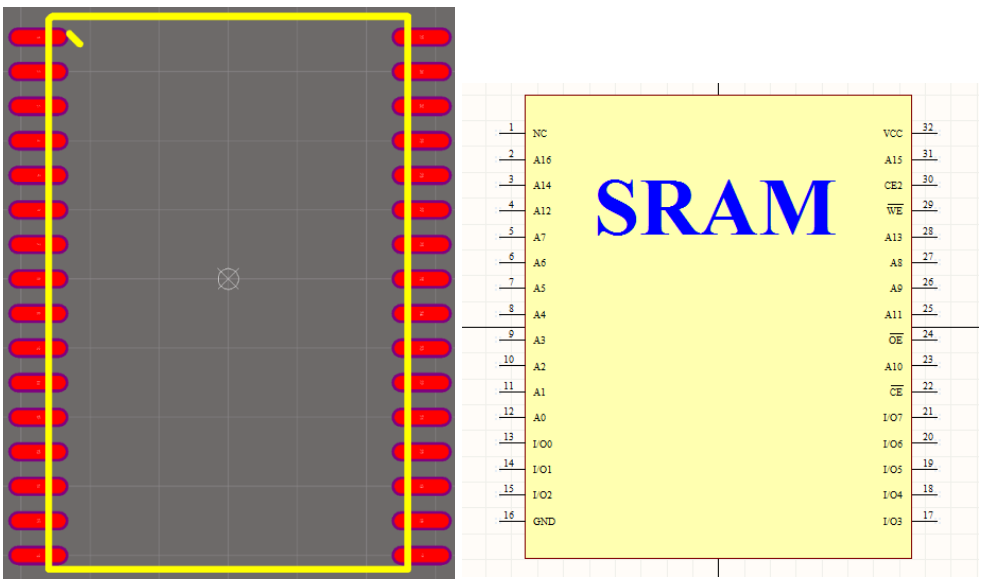


Figura 4.10: Diseño PCB y esquemático de la librería de la memoria RAM CY62128EV30.

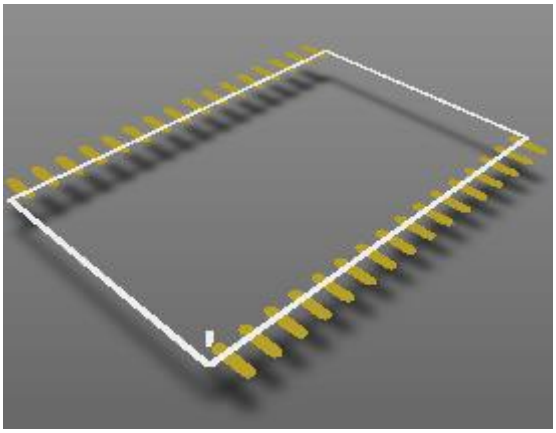
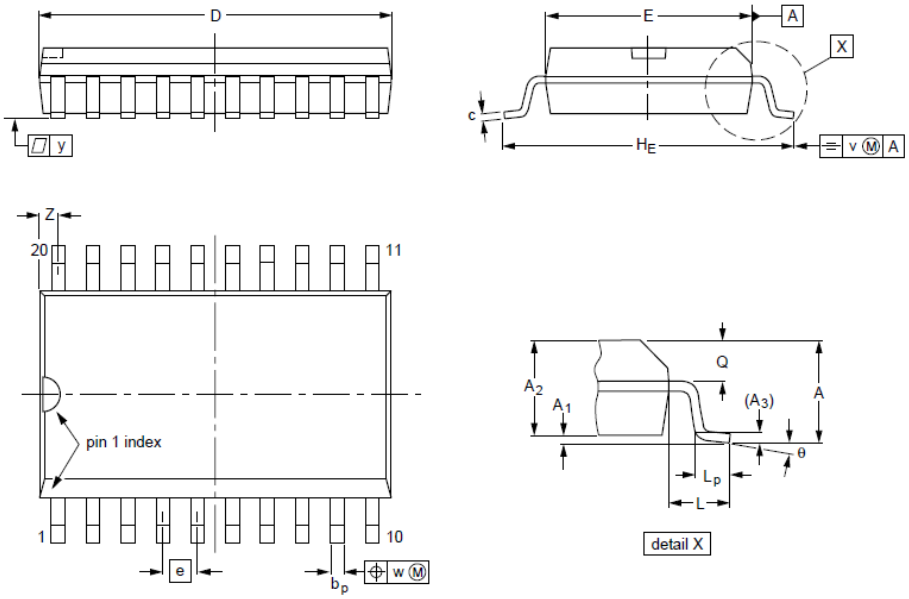


Figura 4.11: Footprint en 3D de la memoria RAM

Para el latch se proporcionan las medidas en ambos tipos de unidad como se aprecia en la Figura 4.12 aunque se especifica que las dimensiones en pulgadas han sido obtenidas de la conversión en milímetros. Tanto en el diseño del *footprint* de la RAM como del latch se han alargado un poco más los *pads*, como se muestra en la Figura 4.13, para que sea más sencillo soldar los integrados a mano.



UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(1)}$	e	H_E	L	L_p	Q	v	w	y	$Z^{(1)}$	θ
mm	2.65	0.3 0.1	2.45 2.25	0.25	0.49 0.36	0.32 0.23	13.0 12.6	7.6 7.4	1.27	10.65 10.00	1.4	1.1 0.4	1.1 1.0	0.25	0.25	0.1	0.9 0.4	8° 0°
inches	0.1	0.012 0.004	0.096 0.089	0.01	0.019 0.014	0.013 0.009	0.51 0.49	0.30 0.29	0.05	0.419 0.394	0.055	0.043 0.016	0.043 0.039	0.01	0.01	0.004	0.035 0.016	

Figura 4.12: Medidas del circuito integrado del Octal Latch

Figura 4.13: Medidas de prolongación de los pads en la RAM y Octal Latch

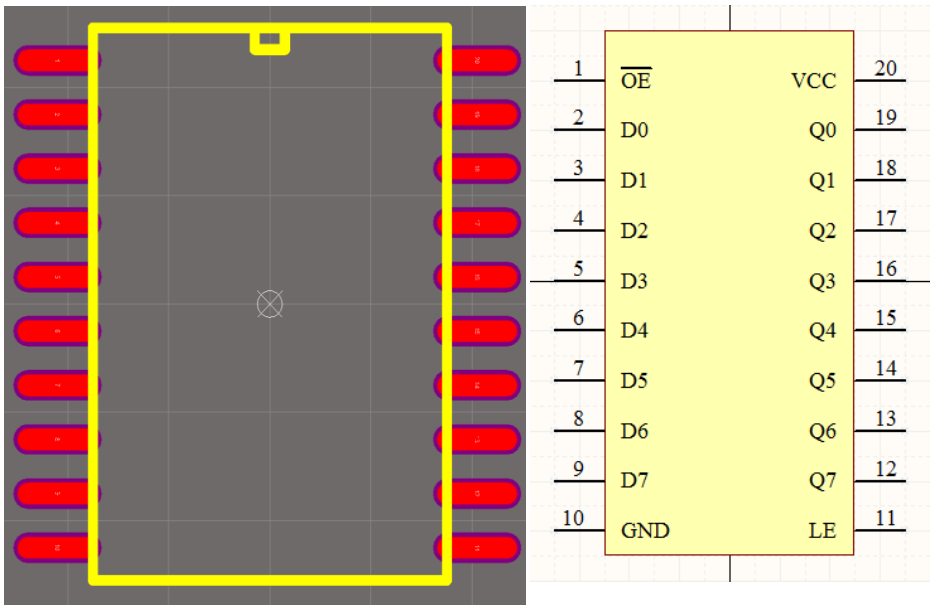


Figura 4.14: Diseño PCB y esquemático de la librería del Octal Latch 74HC

4.4 Diseño final del nuevo prototipo de nodo sensor

Una vez construidas las librerías de los nuevos componentes, éstos ya están disponibles para ser utilizados en el nuevo diseño de nodo sensor. Se crea un nuevo proyecto en Altium con el nombre de DAS220.5.proto y se parte de la solución obtenida en la Sección 4.3.2.

4.4.1 Diseño del esquemático DAS220.5

En este apartado se procede a detallar los cambios que se realizan en el diseño para que la nueva memoria externa tenga cabida en el prototipo. Tal y como se señala en la sección anterior, se deben realizar una serie de cambios en el pineado del microcontrolador ya que será necesario usar una serie de entradas y salidas exclusivamente para la memoria RAM además de eliminar las conexiones que ya no se

utilizan debido a las modificaciones producidas en el modelo DAS220.4, resumidas en la Tabla 4.1.

ID puerto	Puerto antiguo	Puerto nuevo	Observaciones
RL2C	PA0	No existe	Relé eliminado.
RL1C	PA1	No existe	Relé eliminado
RL0C	PA2	No existe	Relé eliminado
SERIAL_ID	PA4	No existe	Componente eliminado.
PA6	PA6	PE3	Pin de la radio, cambia de nombre a PE3.
FLOOD	PA7	No existe	Componente eliminado.
MV_LED_ENABLE	PC3	No existe	Enable eliminado.
WEIGHTSLED	PC2	No existe	Componente eliminado.
M25P_HOLD	PC1	No existe	Componente eliminado
M25P_CS	PC0	No existe	Componente eliminado.
AT45_CS	PG1	No existe	Componente eliminado.
TX_LED	PG0	PB4	Led activado durante la transmisión por radio.
PD5	PD5	No existe	DataFlash eliminado
PD3	PD3	No existe	Componente eliminado.
PD2	PD2	No existe	Componente eliminado.
SENSOR_PWR	PB6	No existe	Componente eliminado
PRESION	PE4	No existe	Componente eliminado.
WAVEFORM	PE3	No existe	Componente eliminado.

Tabla 4.1: Cambios en el pineado de DAS220.4

Una vez están despejados los puertos de la parte del microcontrolador donde se conecta la interfaz de RAM externa añadimos al esquemático el Octal Latch D y la SRAM. Realizamos las conexiones tal y como se aprecia en la Figura 4.15 y siguiendo las indicaciones de la hoja de características del microcontrolador y como se muestra en la Sección 4.3.2.

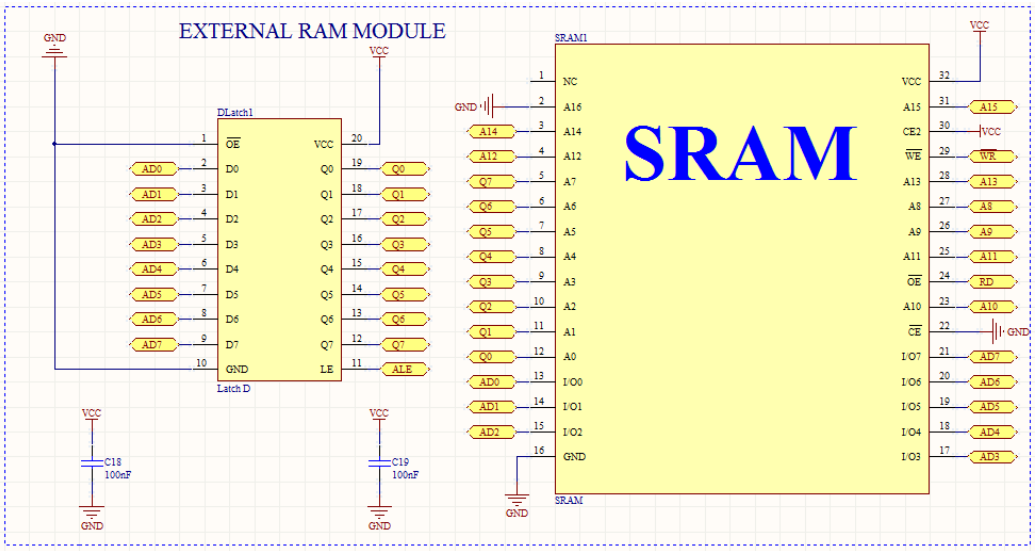


Figura 4.15: Conexiones en el módulo de RAM externa

La configuración específica que debe tener la RAM para las necesidades del proyecto se deduce de la tabla de verdad de ésta. Los puertos CE y CE2 se establecen a masa y a VCC respectivamente y el puerto A16, al no ser necesario ya que el microcontrolador no puede direccionar más de 16 bit de direcciones, se establece a masa.

En cuanto a la configuración del Octal latch D, se establece el puerto OE a masa ya que, con las señales de control del micro en el Latch enable (ALE) y las habilitadoras de lectura y escritura de la RAM, es suficiente para controlar el proceso de lectura y escritura de datos. Un vistazo rápido a la Tabla 4.2, en la que se explican los distintos estados de la RAM, demuestra que no es necesario usar la señal habilitadora de la salida del Octal Latch para manejar qué hay a la entrada de la RAM.

Por otro lado, se introducen en el módulo unos condensadores de desacoplo por seguridad. La ausencia de los capacitores de desacoplo, o la ubicación incorrecta de ellos, puede provocar un funcionamiento errático del circuito. Las causas del funcionamiento errático son:

- Los circuitos digitales conmutan en tiempos muy breves, generando una gran velocidad de cambio (di/dt) en la corriente consumida desde la fuente.

- Las líneas de conexión de fuente y de tierra contienen inductancias parásitas, en las cuales se pueden generar caídas de tensión significativas como consecuencia de la gran di/dt.

- La mayoría de los circuitos integrados digitales (registros, contadores, memorias estáticas, microcontroladores) contienen *flip-flop*, ya que son los elementos básicos requeridos para almacenar cada bit de información.

- Los *flip-flop* pueden cambiar de estado si el voltaje de polarización fluctúa demasiado.

Es por ello que durante el diseño del PCB estos condensadores se colocarán muy cercanos a la alimentación de los integrados.

Truth Table

$\overline{CE_1}$	CE_2	$\overline{WE}$	$\overline{OE}$	Inputs/Outputs	Mode	Power
H	X [32]	X	X	High Z	Deselect/power-down	Standby (I_{SB})
X [32]	L	X	X	High Z	Deselect/power-down	Standby (I_{SB})
L	H	H	L	Data out	Read	Active (I_{CC})
L	H	L	X	Data in	Write	Active (I_{CC})
L	H	H	H	High Z	Selected, outputs disabled	Active (I_{CC})

Tabla 4.2: Tabla de verdad de la RAM

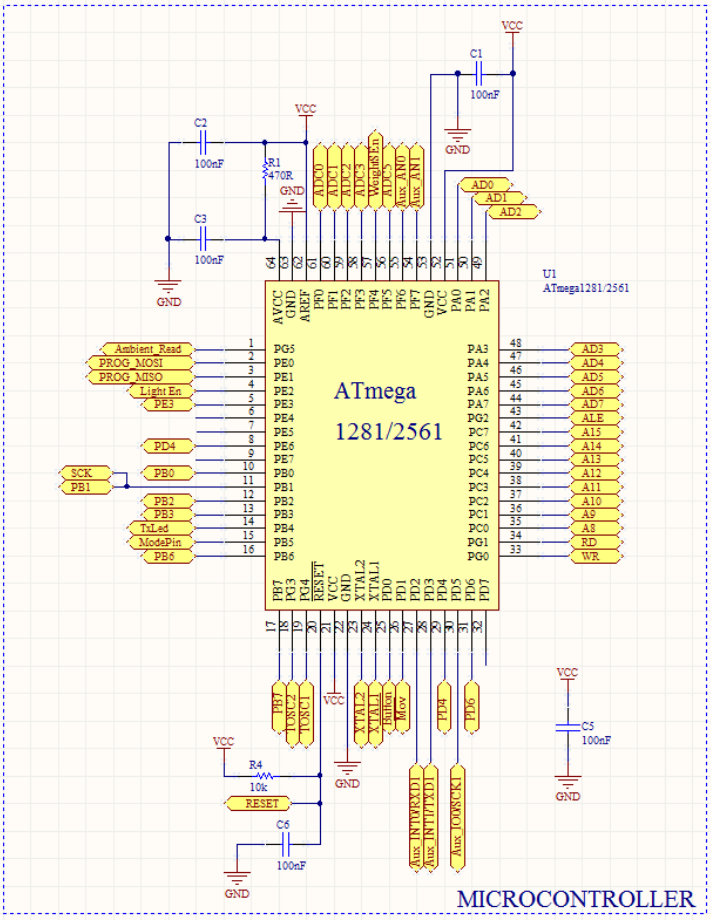


Figura 4.16: Esquemático de las conexiones del microcontrolador

El otro módulo que se añade al diseño es el del sensor de movimiento analógico, dibujado en la Figura 4.17. Se agrega el portasensor y se conecta al puerto ADC5 del microcontrolador. En la Figura 4.16 se pueden ver los puertos del microcontrolador conectados ahora al módulo de la memoria RAM externa y al módulo del sensor de movimiento analógico.

Estos son los cambios producidos a nivel de esquemático. El diseño al completo de este nivel se muestra en el Apéndice, más concretamente en la Figura 6.2.

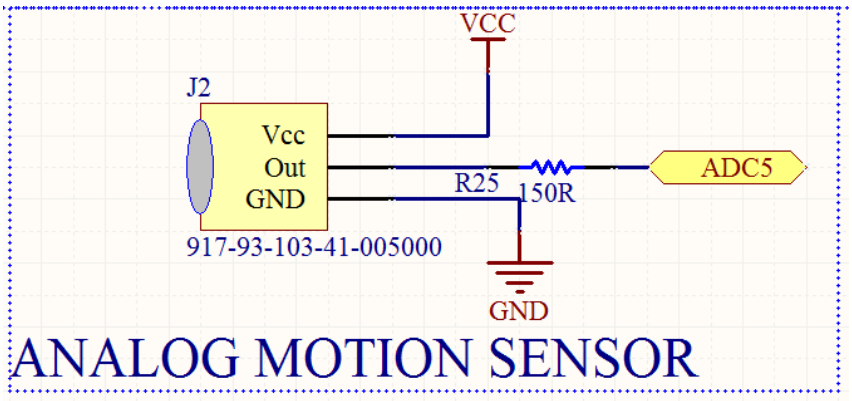


Figura 4.17: Módulo del sensor analógico de movimiento

4.4.2 Diseño PCB del DAS220.5

Una vez terminada la construcción del esquemático, es hora de diseñar cómo quedará físicamente el circuito cuando se fabrique en la plancha de cobre, en otras palabras, se va a diseñar el circuito impreso.

En primer lugar se crea el archivo PCB y se importan los cambios desde el esquemático. Si existiese algún error en las conexiones o alguna incompatibilidad, el software Altium daría un aviso para que el desarrollador lo solucione además de proponer el posible cambio. Una vez importados todos los cambios aparecerá el entorno de trabajo con los componentes en fila unidos por líneas de conexiones a realizar mediante la creación de pistas como se ve en la Figura 4.18.

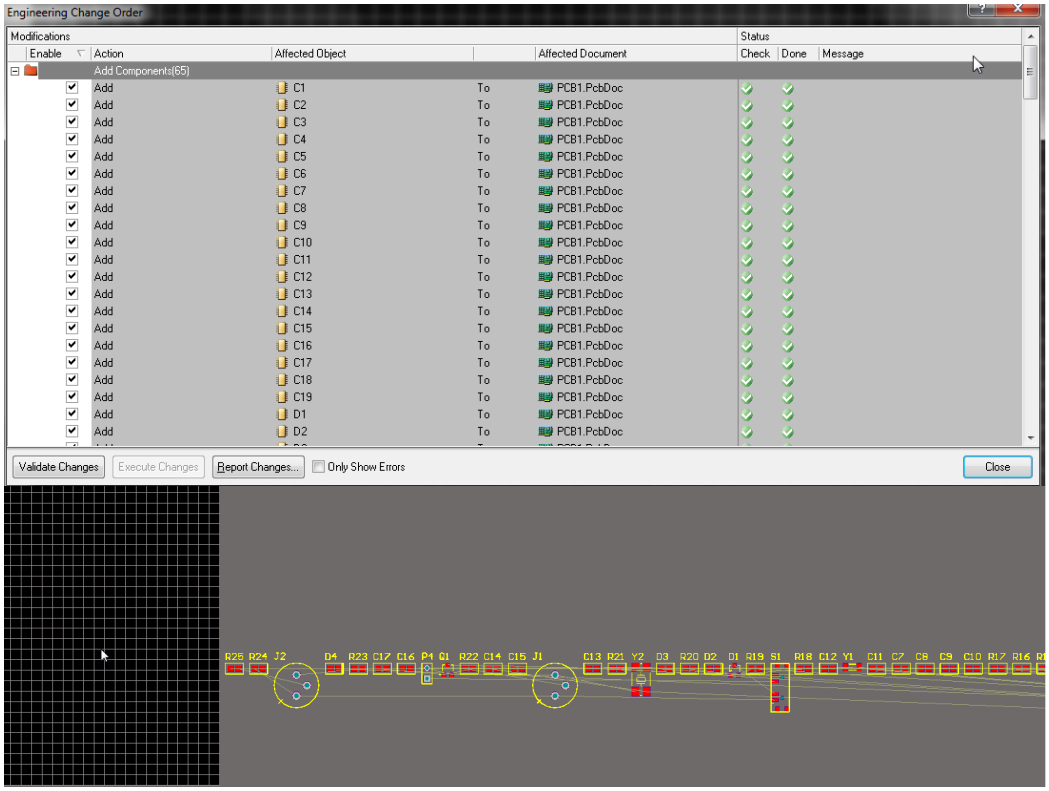


Figura 4.18: Lista de componentes importados al diseño PCB desde el esquemático

El procedimiento que se sigue a la hora de colocar los componentes es hacerlo por módulos, por ejemplo, el módulo del sensor de movimiento digital tendrá todos los componentes que lo forman cercanos entre sí. Estos módulos se colocarán alrededor del microcontrolador, que será el eje central del diseño ya que casi todas las conexiones irán dirigidas a él.

Componentes que por obligación tengan que estar en el borde de la placa, como el conector para el transmisor o el interruptor, tendrán la preferencia en esas posiciones. Otras restricciones que se tienen en cuenta son:

- No hacer ninguna vía debajo de los componentes ya que sería muy complicado metalizarla y colocar el componente encima.

- Los giros de las pistas no serán de más de 45º.

- Componentes voluminosos como pueden ser los sensores de movimiento o el puerto para el programador se colocarán en la misma capa del circuito.

- Las conexiones de componentes cuyos pines deban ser insertados en perforaciones de la placa deberán hacerse justo en la capa opuesta del circuito.

- Se extienden las pistas conectadas a los pines de los integrados un poco para facilitar el estañado. Un ejemplo de este paso se puede ver en la Figura 4.19

donde podría parecer que existen pistas que salen de los pads que no llevan a ninguna parte cuando en realidad está hecho a propósito para facilitar el soldado.

Cuando se termina de realizar todo el proceso se comprueba que se han seguido las reglas de diseño tales como la separación entre pistas, tamaño de las vías, etc. Y se preparan los archivos *gerber* para la fabricación del prototipo, cuyo diseño final se encuentra en el Apéndice, concretamente en la Figura 6.3.

4.4.3 Diseño del módulo de radio RF230

Tal y cómo se explica en la Sección 4.3.2, el módulo de radio se fabrica aparte para que pueda ser utilizado en futuros prototipos ahorrando recursos. El diseño del esquemático es similar al del módulo incluido en el modelo DAS220.4, se introduce como nuevo elemento la interfaz necesaria para poder conectar la radio al resto del dispositivo. El diseño completo se incluye en el Apéndice, en la Figura 6.4.

En la Figura 4.20 se muestra el resultado del desarrollo del módulo como circuito impreso. Se destaca la forma que tiene la antena, adecuada para transmitir y recibir a 2.4GHz.

4.4.4 Impresión del prototipo

En esta fase del proyecto el prototipo ya se encuentra listo para ser fabricado. Se obtienen los archivos *gerber* necesarios y el archivo *NC Drill* a través de la opción *Fabrication outputs* obteniendo un esquema de lo que se dibujará en la plancha de cobre. En la Figura 4.21 se representan en rojo las pistas y pads de la top layer, en azul las de la bottom layer y en fucsia, alrededor del circuito, el keep out layer que delimita las dimensiones del prototipo. En la parte inferior de esta figura se muestran las perforaciones que se harán, contenidas en un archivo txt con las coordenadas y tamaños. También se muestran resaltados en la Figura 4.22 los archivos gerber que se utilizarán en esta ocasión.

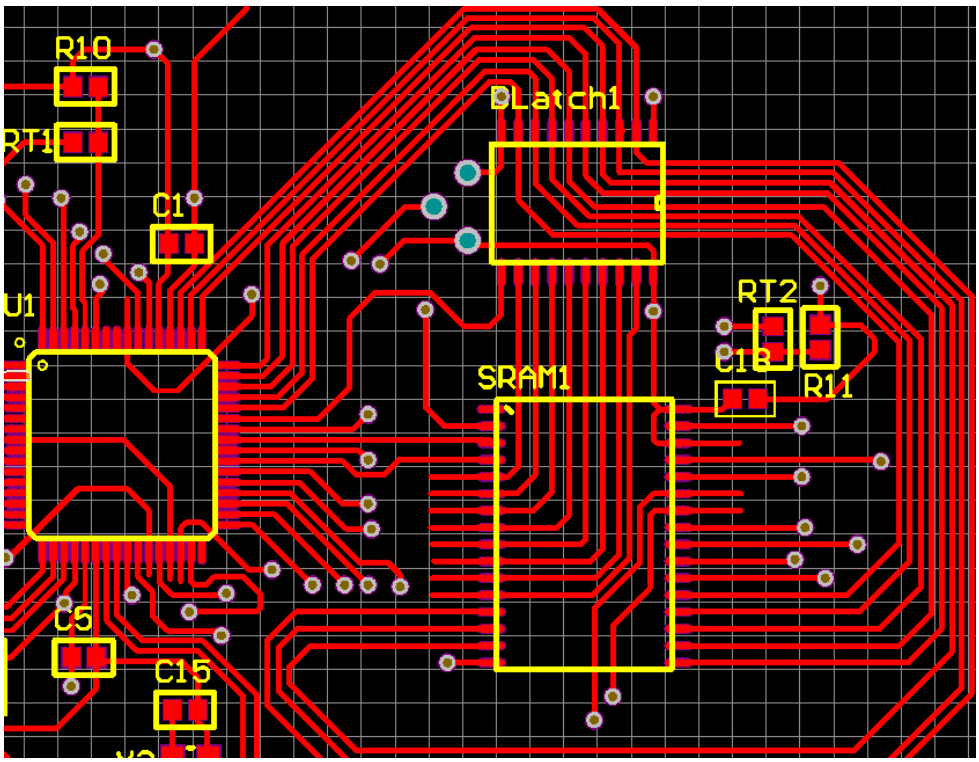


Figura 4.19: Captura PCB de la posición del módulo RAM y el microcontrolador con los pads extendidos

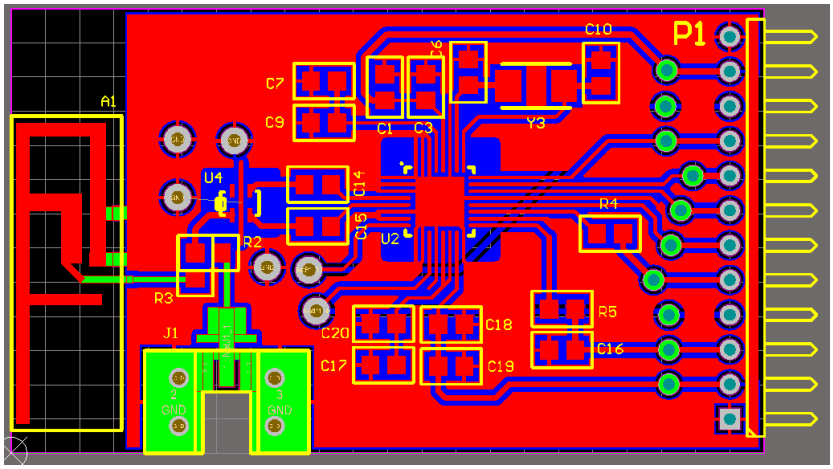


Figura 4.20: Diseño PCB del módulo de Radiofrecuencia

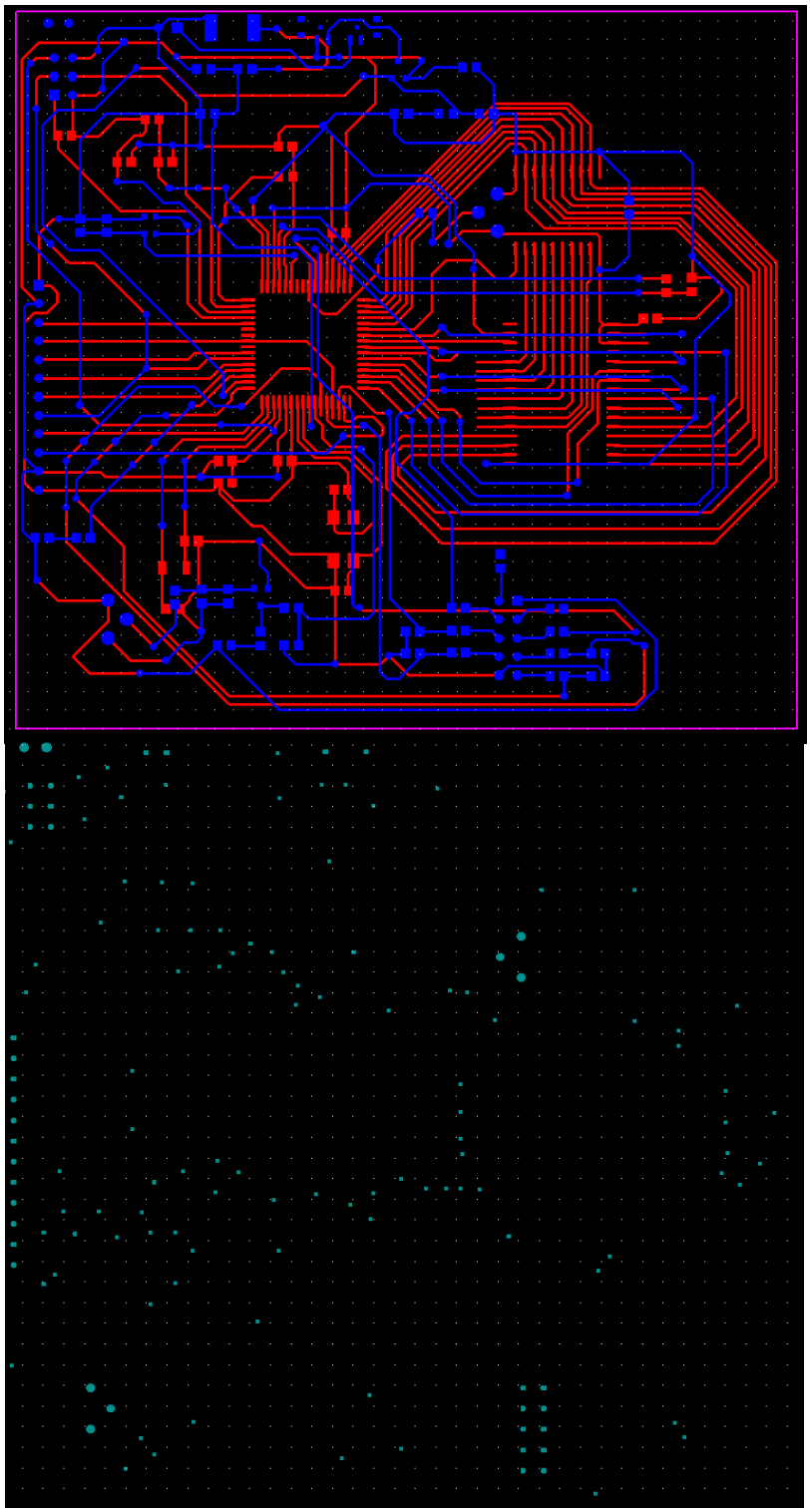


Figura 4.21: Representación de pistas, pads y perforaciones contenidas en los archivos para fabricación.

Nombre	Fecha de modifica...	Tipo	Tamaño
DAS220.5.RAM.proto_b.apr	07/05/2016 19:22	CAMtastic Apertu...	2 KB
DAS220.5.RAM.proto_b.DRL	07/05/2016 19:26	CAMtastic NC Dril...	2 KB
DAS220.5.RAM.proto_b.DRR	07/05/2016 19:26	Altium NC Drill Re...	2 KB
DAS220.5.RAM.proto_b.EXTREP	07/05/2016 19:22	Archivo EXTREP	1 KB
DAS220.5.RAM.proto_b.GBL	07/05/2016 19:22	CAMtastic Botto...	15 KB
DAS220.5.RAM.proto_b.GKO	07/05/2016 19:22	CAMtastic Keepo...	1 KB
DAS220.5.RAM.proto_b.GTL	07/05/2016 19:22	CAMtastic Top La...	24 KB
DAS220.5.RAM.proto_b.LDP	07/05/2016 19:26	Archivo LDP	1 KB
DAS220.5.RAM.proto_b.REP	07/05/2016 19:22	Report File	2 KB
DAS220.5.RAM.proto_b.RUL	07/05/2016 19:22	Archivo RUL	1 KB
DAS220.TXT	07/05/2016 19:26	Documento de tex...	2 KB
DAS220-macro.APR_LIB	07/05/2016 19:22	Archivo APR_LIB	0 KB
Status Report.Txt	07/05/2016 19:26	Documento de tex...	1 KB

Figura 4.22: Archivos gerber

En la Sección 3.3 se explican los pasos a seguir con CircuitCam para convertir estos archivos a LMD, fichero que se carga en el software BoardMaster para controlar a la microfresadora. El resultado en la interfaz del programa se puede ver en la Figura 4.23 donde se aprecian, entre otras cosas, los movimientos horizontales que hará el cabezal de la microfresadora para vaciar de cobre ciertas regiones.

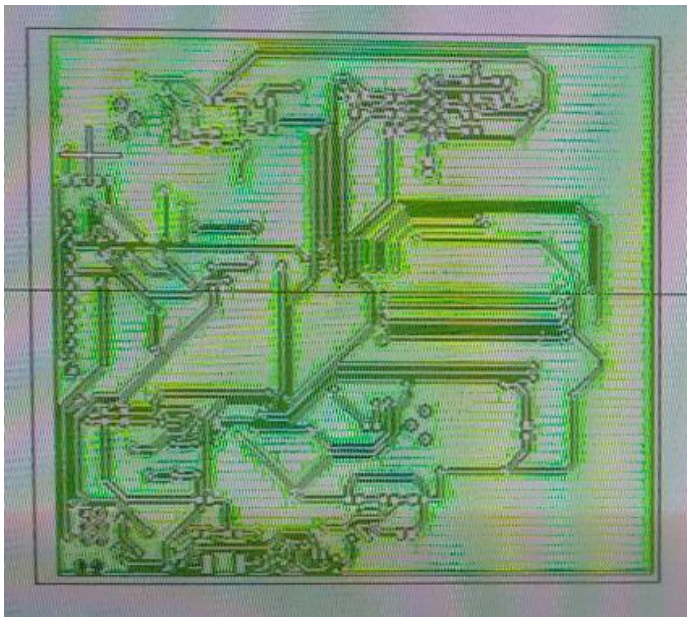


Figura 4.23: Esquema de movimientos de la microfresadora

Para comenzar el proceso, se coloca la plancha de cobre en el soporte de la máquina y se ancla con los tetones situados a ambos extremos de la plancha sobre el eje de simetría de la misma. Para una comprobación rápida de que el eje de simetría es

el correcto se realiza un pequeño taladro de forma manual utilizando los controles que proporciona BoardMaster. Seguidamente se le da la vuelta a la plancha, se posiciona el taladro en las mismas coordenadas y se le da la orden al cabezal para que realice otro taladro. Si coincide con el agujero hecho la primera vez es que el eje de simetría es correcto y que la top y la bottom layer se fresarán sin que haya pistas que no lleguen a las vías que les corresponden o más problemas que pueden causar este tipo de configuraciones erróneas.

Realizar esta tarea de comprobación, antes de empezar a fabricar, es importante ya que la máquina fresadora puede tardar entre 2 y 3 horas en terminar todo el proceso de impresión y se malgastaría mucho tiempo si desde el mismo inicio se están cometiendo errores.

4.4.5 Fases de la impresión del circuito

La máquina fresadora consta de varias fases diferenciadas en la creación del circuito impreso tal como se muestra en la Figura 4.24. Es importante seguir el orden que marca el software aunque está la opción de elegir la fase que se desee por si hubiera que volver a repasar ciertas regiones del circuito porque los taladros no hayan perforado completamente la plancha o por cualquier otro problema que pudiera surgir.

1. La primera fase consiste en marcar en la plancha de cobre los puntos donde se van a realizar los taladros. Después de escoger la fase ***Marking drills*** se selecciona todo el circuito y se presiona Start. Estos pasos son comunes a todas las fases y, si no se dispone de una fresadora con cambio de herramienta automático, la máquina primero llevará el cabezal al punto de cambio de herramienta donde instalaremos el taladro o fresa adecuado tal y como se ve en la Figura 4.27.

2. Durante la segunda fase, ***Drilling plated***, se efectúan los taladros. El cabezal irá al punto de cambio de herramienta cada vez que haya que instalar una broca de distinto grosor ya que no todos los taladros tienen el mismo tamaño.

3. Es en la tercera fase, ***Milling bottom***, cuando se realiza el fresado de la capa bottom del circuito. Primero se instala una fresa de 0,2mm para tareas de fresado de precisión, luego otra fresa más grande para ir retirando cobre alrededor de las pistas y, por último, una fresa aún más grande para vaciar las regiones de cobre que no contengan ninguna pista o vía cercanas.

4. Una vez terminado el milling bottom, se voltea la plancha de izquierda a derecha colocándola de nuevo con los tetones en el soporte de la microfresadora. Es entonces cuando comienza la cuarta fase, ***Milling top***, ejecutando los mismos pasos que en la fase anterior pero esta vez dibujando en

la top layer. De estas últimas dos fases se muestran en las Figuras 4.25 y 4.26 el estado del circuito con el proceso a medio terminar.

5. Por último, se instala la herramienta *universal cutter* y comienza la quinta fase, ***Cutting outside***, recortando el circuito por el perímetro que delimita al prototipo.

Figura 4.24: Fases de impresión del circuito

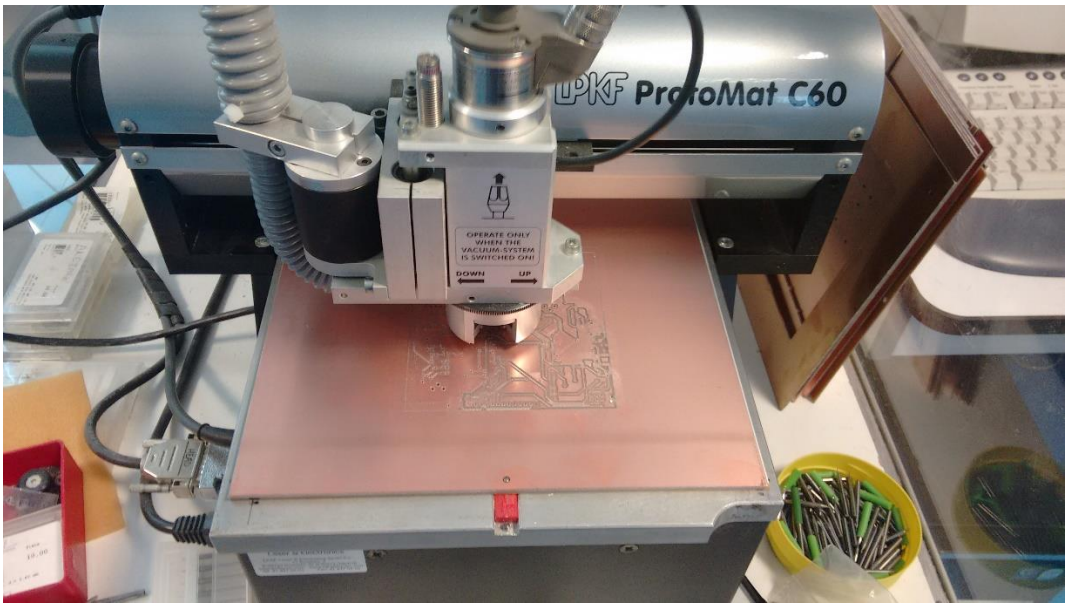


Figura 4.25: Vista de la microfresadora durante la fabricación

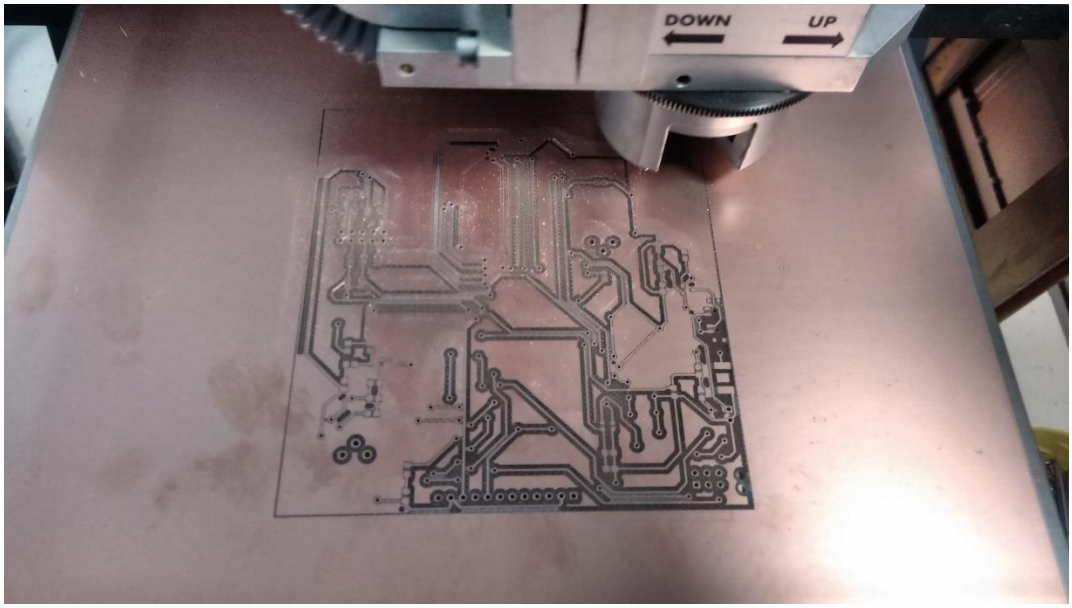


Figura 4.26: Circuito impreso durante su fabricación

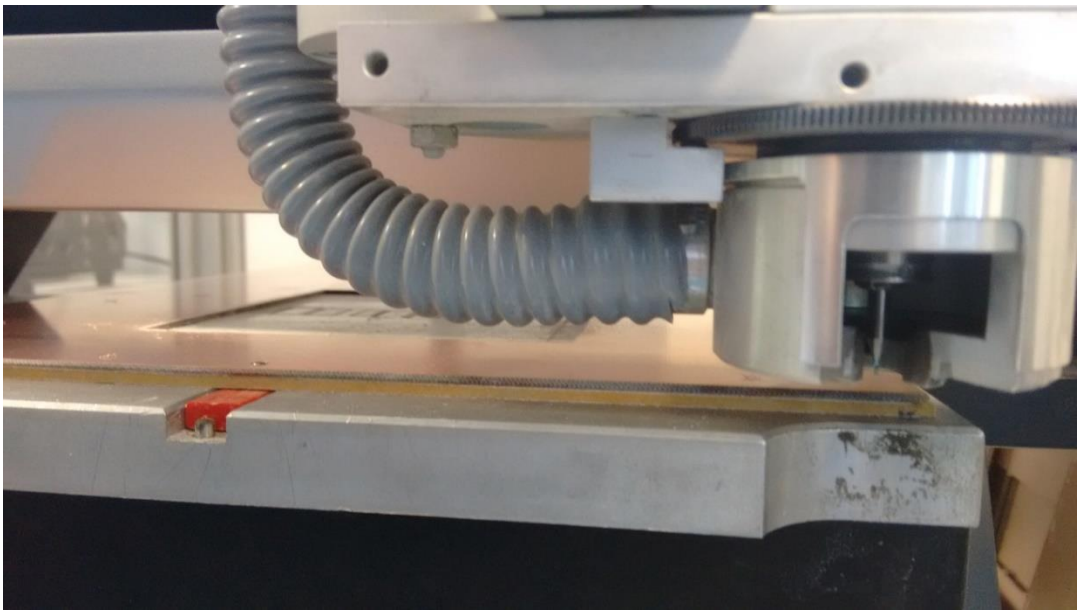


Figura 4.27: Posición de cambio de herramienta del cabezal

4.4.6 Soldado de los componentes al circuito

Antes de empezar a soldar los componentes electrónicos con estaño, se revisa la placa con una lupa para asegurar que no haya pistas que estén cortocircuitadas. Las impurezas de cobre que hayan quedado se retiran con cuidado de no dañar el circuito. Una vez hecho esto se le aplica una laca como la de la Figura 4.28 para evitar la degradación del circuito debido a agentes externos tales como luz del sol, oxidación, arañazos…Y se deja secar.

Figura 4.28: Lacado del circuito impreso

Ya se tiene todo listo para soldar los componentes electrónicos. Se usa una estación de soldadura con temperatura regulable y multitud de herramientas para un trabajo más cómodo. Los primeros componentes que se colocan son los circuitos integrados (Atmel1281, SRAM y Octal Latch) seguidos de la metalización de las vías que, al no disponer de una remachadora, se ha de hacer de forma manual. A continuación se sueldan el resto de componentes: resistencias, condensadores, osciladores, portasensores, el interruptor, diodos led, etc. Tal y como se puede ver en las Figuras 4.29 y 4.30 en las que se incluyen fotografías tomadas en distintas etapas de soldado La lista de materiales se recoge en el Apéndice, en la Tabla 6.1.

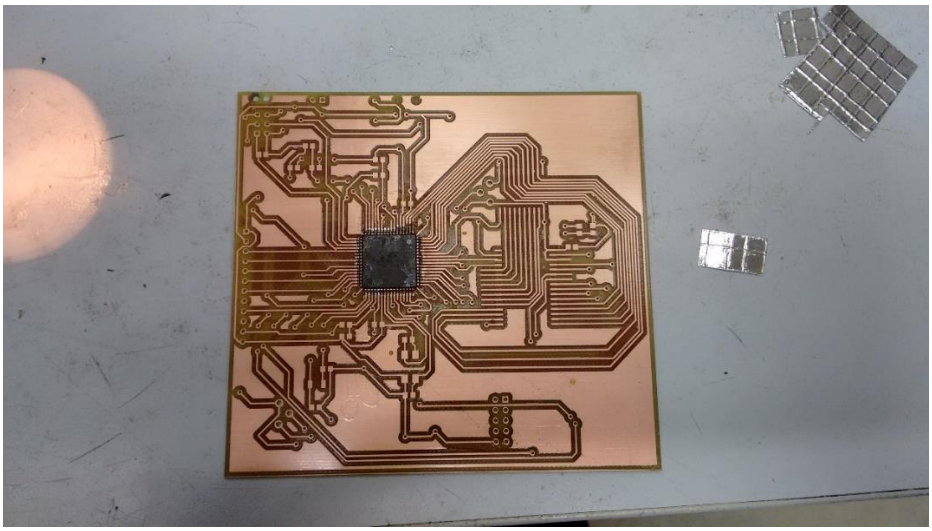


Figura 4.29: Soldado del microcontrolador

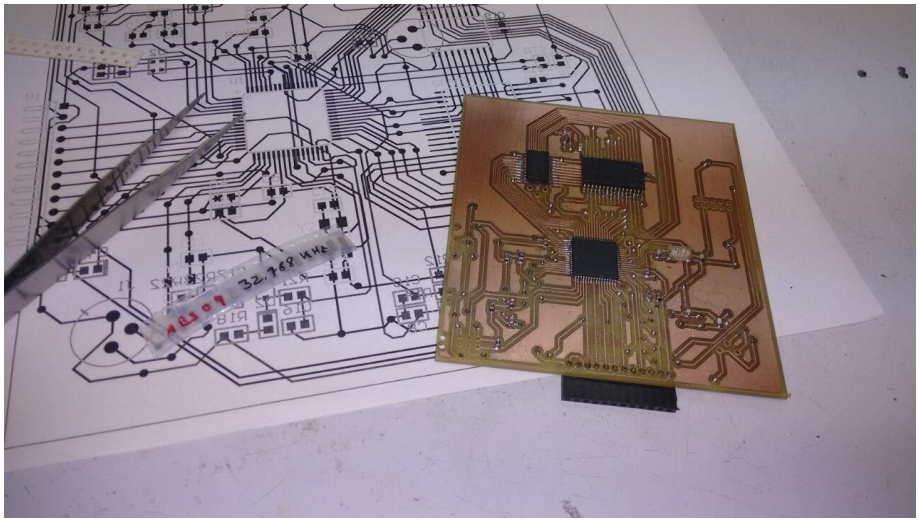


Figura 4.30: Vista del circuito con la mayoría de sus componentes ya soldados

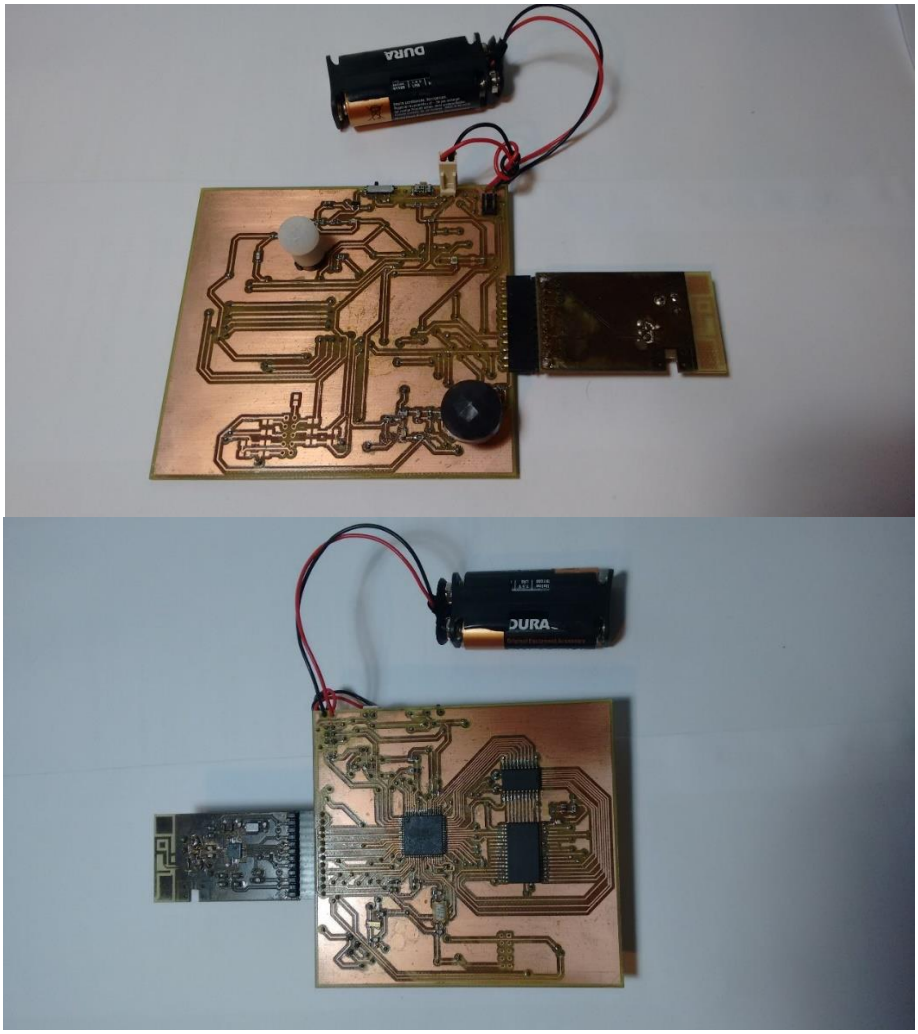


Figura 4.31: Vista anterior y posterior del circuito una vez acabado el montaje

El resultado final después de todo el proceso de impresión y montaje del prototipo se puede ver en la Figura 4.31 donde se muestran tanto la capa bottom como la capa top del circuito, con el módulo de radio conectado y los sensores de movimiento colocados en sus bases.

4.4.7 Comprobaciones realizadas sobre el prototipo

Antes de proceder a programar el dispositivo se comprueba que no haya cortocircuitos o circuitos abiertos provocados por alguna soldadura mal hecha. Para ello se utiliza un multímetro con opciones para comprobar la continuidad, leds y medir la resistencia total del circuito. Estas pruebas resultan exitosas así que se avanza al siguiente paso: Intentar programar el microcontrolador. Como se puede ver en la Figura 4.32, el script se ejecuta con normalidad y la programación se completa satisfactoriamente.

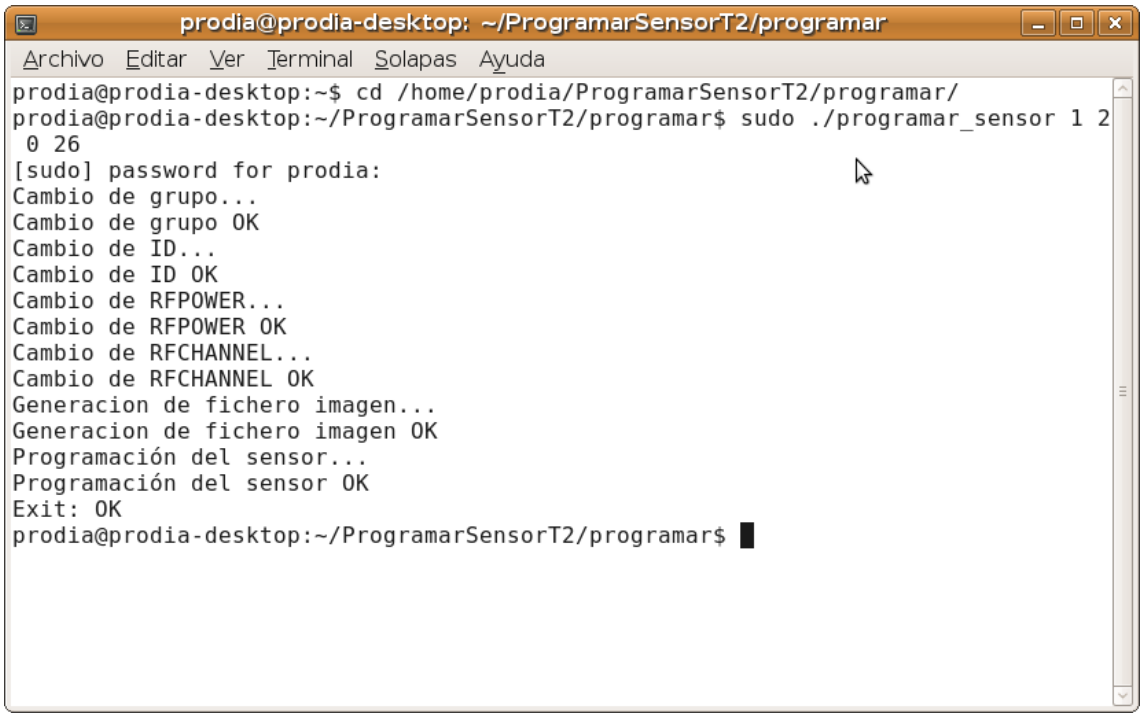


Figura 4.32: Ejecución del script del programador

La aplicación cargada al microcontrolador es la misma versión que había hecha hasta ahora salvo por los cambios introducidos para adaptarla a las nuevas posiciones de los pines, a los módulos que se descartaron y a los nuevos módulos introducidos. Se realiza pues la comprobación de conectividad por radiofrecuencia con la estación base acoplando primero el módulo de radiofrecuencia al circuito. Como se aprecia en la Figura 4.33, la trama se recibe correctamente y sin pérdidas ya que los datos mostrados en ella son conocidos de antemano conectando VCC, y masa posteriormente, a la salida del portasensor.

```
[driverLog : Day 21 - 05 - 2016 , 11 : 50 : 35] :Frame: -> {7E 45 00 FF FF 00 02 37 01 09 31 33 00 02 00 01 07 50 00 01 00 00
00 02 00 03 FF 03 FF 03 FF 03 FF 03 FF 03 FF 03 FB 03 FF 03 FF 03 FF 03 FF 03 FF 03 FF 03 FF 03 FF 03 FF 03 FF 03 FF 03 FF
FF CRC 7E},
[driverLog : Day 21 - 05 - 2016 , 11 : 51 : 17] :Frame: -> {7E 45 00 FF FF 00 02 37 01 09 31 33 00 02 00 01 07 50 00 01 00 00
00 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 CRC 7E},
```

Figura 4.33: Tramas con la salida del portasensor conectada a VCC y masa respectivamente

La siguiente comprobación es sobre el correcto funcionamiento de los sensores de movimiento. Antes de nada se va a implementar un algoritmo para reducir el consumo de energía del prototipo, resumido en el diagrama de la Figura 4.34. Como se ha explicado en secciones anteriores en este proyecto, uno de los objetivos es que el nodo sensor funcione con la mínima cantidad de energía sin que afecte a su funcionalidad. Por ejemplo, el protocolo que se ha estado utilizando para la comunicación con la estación base es el protocolo Tymo. Este protocolo está basado en el Dymo pero implementado para TinyOS y está diseñado para crear rutas de forma dinámica bajo demanda en redes inalámbricas que requieran un consumo moderado. Continuando con el requisito del bajo consumo, el algoritmo introducido se basa en activar el muestreo del sensor analógico únicamente cuando el sensor digital de movimiento detecte un flanco levantando una *flag* para impedir que la interrupción vuelva a saltar mientras el sensor de movimiento analógico esté muestreando.

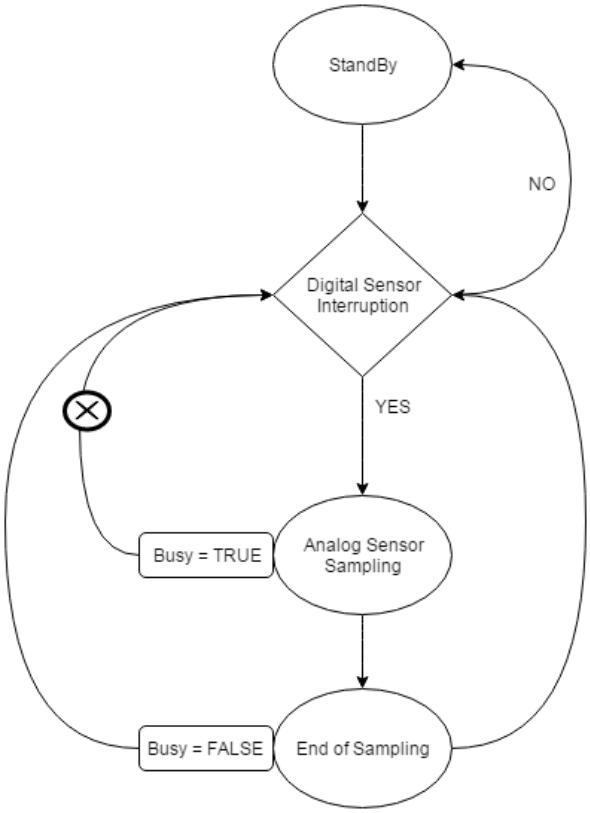


Figura 4.34: Diagrama del algoritmo de interrupciones

La última comprobación que se va a realizar es para conocer si el módulo del sensor analógico de movimiento captura las muestras correctamente. Para ello se obtiene la forma de onda de este sensor conectado directamente al osciloscopio. En la Figura 4.35 se aprecia la forma típica de las señales que emite un sensor analógico, la señal capturada variará dependiendo de la posición y velocidad del movimiento detectado.

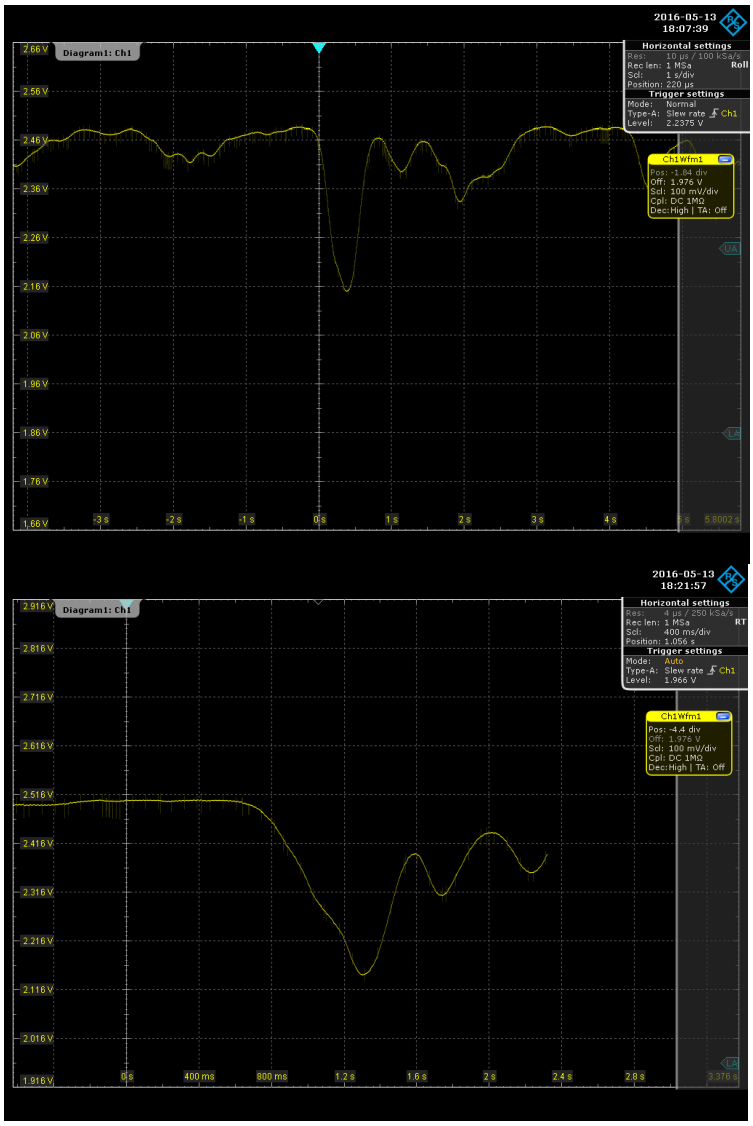


Figura 4.35: Capturas de osciloscopio de formas típicas de onda en un sensor analógico

Ahora es el prototipo el que toma las muestras de este sensor y las envía por radiofrecuencia obteniendo los siguientes resultados en función de la posición. Las señales generadas constan de 20 muestras tomadas en un intervalo de tiempo de 0.1 segundos entre cada una, es decir, 2 segundos de tiempo de muestreo con una frecuencia de 10Hz, la misma frecuencia que marcan los requisitos del proyecto. En las gráficas se representan los niveles de voltaje de la señal cuantificados y codificados con una precisión de 10 bits (de 0 a 1023) en función del tiempo.

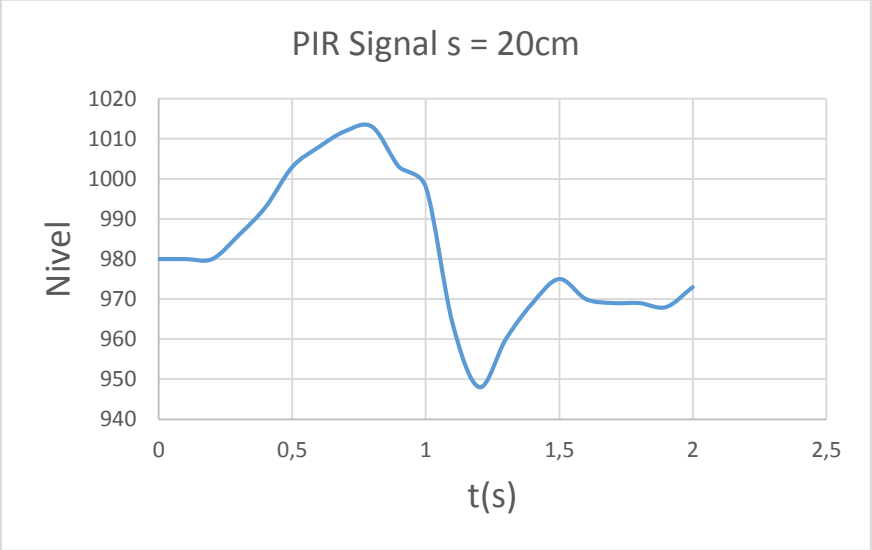


Figura 4.36: Señal muestreada del sensor analógico a 20cm

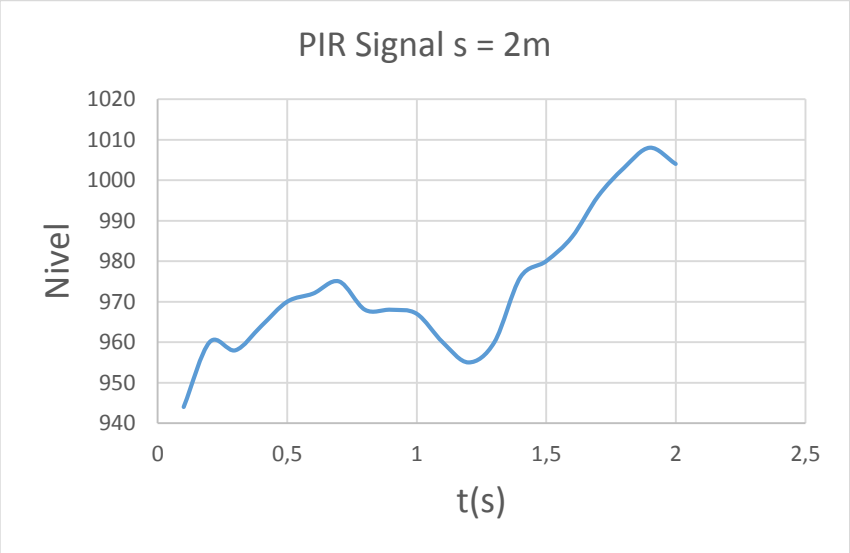


Figura 4.37: Señal muestreada del sensor analógico a 2m

Si se comparan las señales de las Figuras 4.36, 4.37, 4.38, 4.39 y 4.40 con la Figura 4.35, en la que se muestran capturas de osciloscopio, se puede concluir que el hardware y la parte de software realizada hasta ahora funcionan de forma correcta ya que las señales capturadas a través del nodo tienen la forma característica de las de un sensor analógico. Este resultado permite continuar con la última fase del PFC, la implementación del método de clasificación de señales.

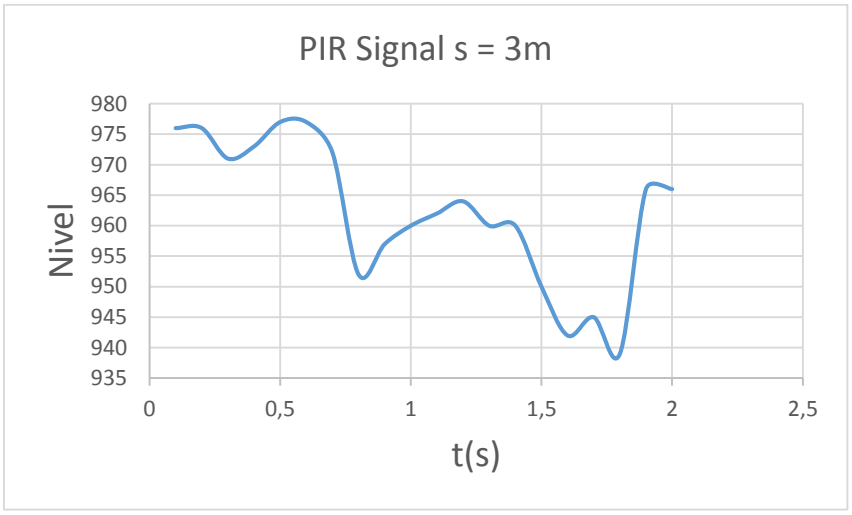


Figura 4.38: Señal muestreada del sensor analógico a 3m

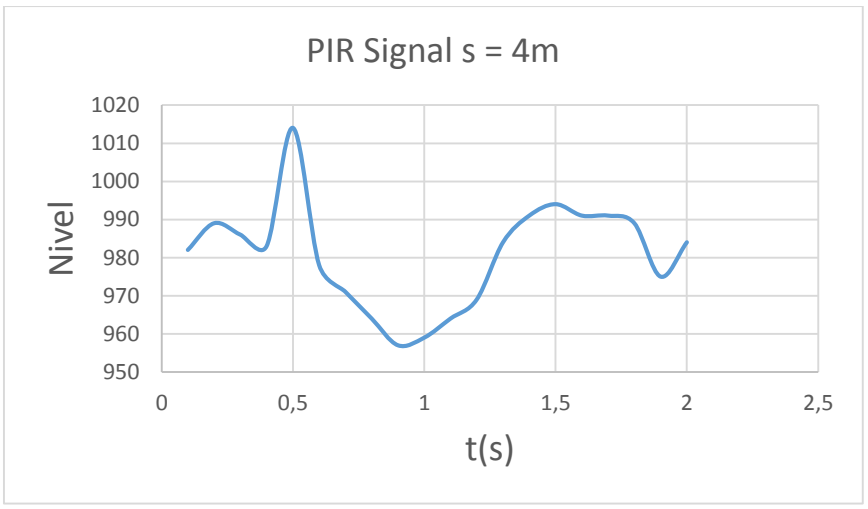


Figura 4.39: Señal muestreada del sensor analógico a 4m

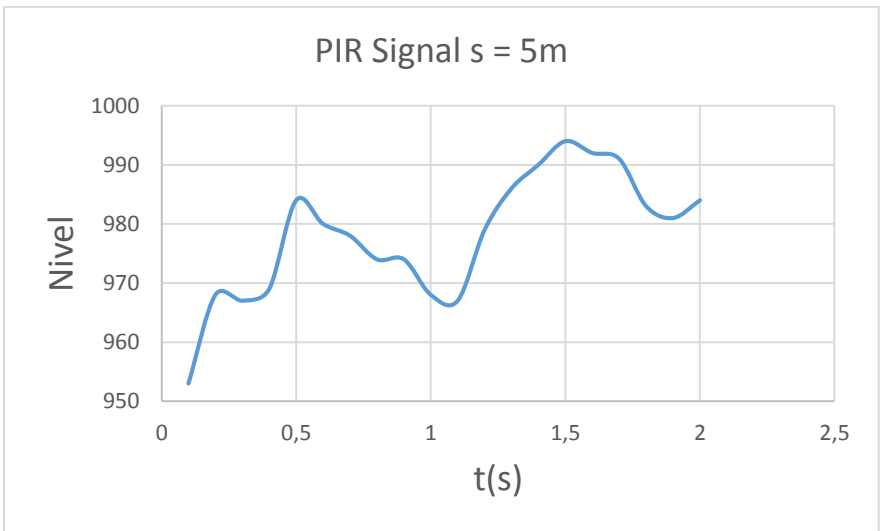


Figura 4.40: Señal muestreada del sensor analógico a 5m

4.5 Desarrollo del algoritmo de clasificación de señales

En el estudio previo del método de clasificación se explicó qué algoritmo se va a implementar. Para realizar las operaciones se necesitan dos conjuntos de datos, el conjunto patrón y el conjunto de señales. El patrón, como se estudió en la Sección 4.1, está formado por 28 señales, mostradas en la Figura 4.41, de distintos tamaños cada una. La velocidad es indexada por columnas (de la A a la D), desde 200mm/s hasta 800mm/s en pasos de 200mm/s mientras que la distancia es indexada por filas (de la 1 a la 7), desde 0.5m hasta 3.5m en pasos de 0.5m.

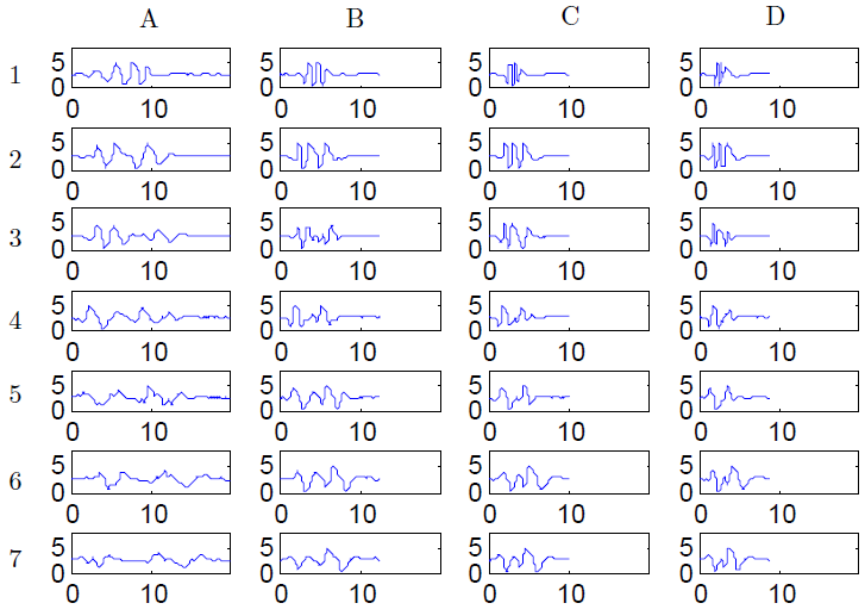


Figura 4.41: Conjunto de señales patrón

Las señales patrón no tienen todas el mismo tamaño, algo lógico por otra parte ya que no todas las muestras serían objeto de análisis debido a las distintas velocidades del sujeto, en otras palabras, si el sujeto avanza con mayor velocidad saldrá del rango de detección del sensor o llegará a su posición final antes y, aunque este siga muestreando, las muestras obtenidas no contendrán información acerca de los parámetros velocidad y distancia. La señal que se capture por el sensor analógico constará de 128 muestras, cantidad de datos lo suficientemente significativa como para poder obtener un resultado, obtenidas a una frecuencia de 10Hz, esto significa que el dispositivo se encontrará ocupado en la terea de muestreo durante algo más de 12 segundos en los que no podrá saltar ninguna interrupción para que los datos tomados sean fiables, problema que se soluciona con el algoritmo resumido en la Figura 4.34.

Volviendo al asunto tratado en este apartado, se crean las siguientes funciones en la aplicación en las que se realizarán las operaciones matemáticas necesarias para conseguir el objetivo propuesto por el algoritmo:

- *four1()*: Función que calcula la FFT o iFFT de una tabla data[].Esta función requiere una tabla con números complejos lo que deriva en un tamaño doble.
- *Xcorr()*: Realiza la correlación cruzada de las señales patrón y adquirida.
- *corr()*: realiza la correlación de dos señales.
- *getPirDiff()*: efectúa y obtiene el resultado de la derivada discreta de la señal que se le pase como argumento.
- *readPirPat()*: Esta función se ejecutará sólo una vez y sirve para ordenar todas las muestras de todas las señales patrón para que puedan ser tratadas.
- *readPirData()*: Hace lo mismo que la función anterior pero con las muestras obtenidas por el sensor. Ésta se ejecutará cada vez que se termine la rutina de muestreo.

También se crean funciones auxiliares para ayudar en el cálculo:

- *getAC():* obtiene un vector de medias.
- *getSumQ()*: obtiene la suma de cuadrados de un vector.
- *getFFTmod()*: Obtiene el módulo de la FFT de una señal que se le pase como argumento.

La función que controla el proceso de todos los cálculos, *operaciones(),* se programa de tal forma que se lance cuando la aplicación termina de tomar las muestras del sensor analógico y no se vuelva a tratar una interrupción provocada por el sensor digital que daría inicio de nuevo a la rutina de muestreo hasta que *operaciones()* hubiera acabado de realizar los cálculos.

El proceso del algoritmo sería el siguiente:

1. Se termina de obtener las muestras por parte de la rutina provocada por la interrupción del sensor digital y que controla al sensor analógico, este proceso se explicó en la Sección 4.4.7. Las muestras son almacenadas en una tabla y se ejecuta la función *operaciones()*.

2. Si es la primera vez que se entra en *operaciones()* se realiza la función *readPirPat()*. Los patrones, guardados en un fichero de constantes en una tabla unidimensional se van introduciendo en otra tabla de dimensiones 28x128 teniendo en cuenta el tamaño de cada señal, especificado a su vez en el fichero de constantes y rellenando el tamaño sobrante de la tabla bidimensional de patrones con ceros. También se calcula la FFT y las derivadas de esta tabla bidimensional para no tener que gastar capacidad de cómputo durante la ejecución de las operaciones. Si no fuera la primera vez significaría que los datos del conjunto patrón ya están en las tablas para ser utilizados en los cálculos que van a continuación.

3. Comienza un bucle de 28 iteraciones para ir haciendo las distintas operaciones a las muestras obtenidas con cada una de las señales del patrón.

4. La función *readPirData()* prepara la tabla de los datos obtenidos y la deja con el tamaño adecuado para operar con ella, rellenando de ceros el espacio sobrante.

5. Se realizan los cálculos de la correlación cruzada de la señal patrón de la iteración actual y la señal muestreada, la FFT y la derivada de la señal capturada y la correlación de las mismas con las tablas de FFT y derivada de la señal patrón.

6. El resultado de esta correlación es el que indicará a qué patrón correspondería clasificar la señal obtenida, es decir, a qué señal del conjunto patrón se asemeja más.

7. El último paso antes de acabar la iteración es fusionar, efectuando el producto, las tres correlaciones resultado de los tres métodos anteriores para obtener un grado de similitud entre los dos conjuntos de datos más preciso.

En cada iteración se comprobará, comparando con los resultados de las correlaciones obtenidos en la iteración anterior para cada una de las correlaciones, cuál es la correlación máxima.

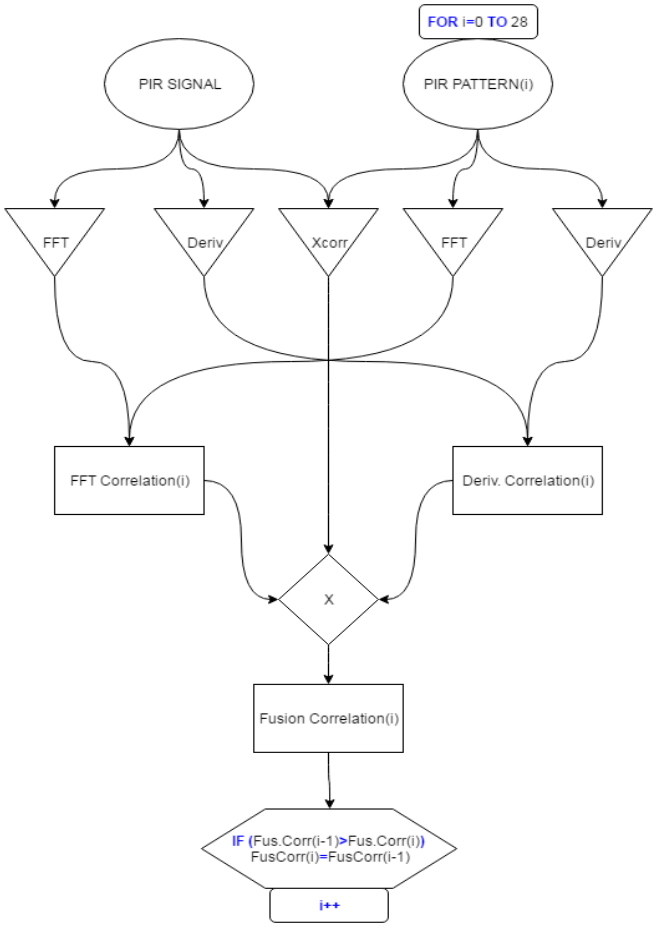


Figura 4.42: Diagrama explicativo del algoritmo de clasificación de señales

El resultado del algoritmo, resumido en el diagrama de la Figura 4.42, será la máxima de las correlaciones obtenidas en todas las iteraciones, que corresponderá con una de las señales del conjunto patrón y, a su vez, ésta corresponde a una posición y velocidad concreta del sujeto. Datos que se enviarán a la estación base y que podrán ser fruto de análisis para detectar situaciones de peligro en las que se pudiera ver envuelto el sujeto.

A continuación se va a demostrar que el algoritmo funciona siendo ejecutado en un PC. Se utilizan 28 señales de las que se conoce el resultado de su clasificación y se procesan junto con el conjunto patrón. Los resultados son los siguientes:

- Señal 1, correspondiente a la posición 1 del conjunto patrón:

Se muestran los resultados de aplicar la correlación a las distintas operaciones en cada una de las 28 iteraciones que se realizan para obtener, finalmente, una clasificación de la señal capturada. En el resto de señales se obvian los resultados intermedios de cada iteración y tan sólo se muestra el resultado final de la clasificación.

corMax[1] = 0.908743	corMax[2] = 0.574414	corMax[3] = 0.628169
corDiff[1] = 0.859578	corDiff[2] = 0.290569	corDiff[3] = 0.405472
corFFT[1] = 0.962550	corFFT[2] = 0.851660	corFFT[3] = 0.813714
corFus[1] = 0.751882	corFus[2] = 0.142148	corFus[3] = 0.207257
corMax[4] = 0.591630	corMax[5] = 0.709061	corMax[6] = 0.585492
corDiff[4] = 0.336890	corDiff[5] = 0.596262	corDiff[6] = 0.138808
corFFT[4] = 0.844890	corFFT[5] = 0.848987	corFFT[6] = 0.810249
corFus[4] = 0.168399	corFus[5] = 0.358940	corFus[6] = 0.065850
corMax[7] = 0.478982	corMax[8] = 0.422638	corMax[9] = 0.763085
corDiff[7] = 0.270099	corDiff[8] = 0.239764	corDiff[9] = 0.554625
corFFT[7] = 0.914703	corFFT[8] = 0.934611	corFFT[9] = 0.935112
corFus[7] = 0.118337	corFus[8] = 0.094707	corFus[9] = 0.395764
corMax[10] = 0.469922	corMax[11] = 0.387949	corMax[12] = 0.391426
corDiff[10] = 0.452324	corDiff[11] = 0.253916	corDiff[12] = 0.420624
corFFT[10] = 0.929037	corFFT[11] = 0.930528	corFFT[12] = 0.945418
corFus[10] = 0.197473	corFus[11] = 0.091663	corFus[12] = 0.155657
corMax[13] = 0.458685	corMax[14] = 0.509396	corMax[15] = 0.505079
corDiff[13] = 0.311269	corDiff[14] = 0.580555	corDiff[15] = 0.366505
corFFT[13] = 0.874408	corFFT[14] = 0.878699	corFFT[15] = 0.923391
corFus[13] = 0.124843	corFus[14] = 0.259860	corFus[15] = 0.170932

corMax[16] = 0.480361
corDiff[16] = 0.235833
corFFT[16] = 0.923940
corFus[16] = 0.104669

corMax[17] = 0.463756
corDiff[17] = 0.554647
corFFT[17] = 0.776343
corFus[17] = 0.199692

corMax[18] = 0.644248
corDiff[18] = 0.298945
corFFT[18] = 0.793648
corFus[18] = 0.152852

corMax[19] = 0.502585
corDiff[19] = 0.298132
corFFT[19] = 0.917083
corFus[19] = 0.137412

corMax[20] = 0.467781
corDiff[20] = 0.337081
corFFT[20] = 0.848614
corFus[20] = 0.133810

corMax[21] = 0.813828
corDiff[21] = 0.691458
corFFT[21] = 0.853882
corFus[21] = 0.480503

corMax[22] = 0.444899
corDiff[22] = 0.430789
corFFT[22] = 0.913867
corFus[22] = 0.175149

corMax[23] = 0.447540
corDiff[23] = 0.387666
corFFT[23] = 0.855840
corFus[23] = 0.148485

corMax[24] = 0.407614
corDiff[24] = 0.326406
corFFT[24] = 0.686929
corFus[24] = 0.091394

corMax[25] = 0.767227
corDiff[25] = 0.611759
corFFT[25] = 0.816217
corFus[25] = 0.383098

corMax[26] = 0.492272
corDiff[26] = 0.443977
corFFT[26] = 0.895729
corFus[26] = 0.195768

corMax[27] = 0.474075
corDiff[27] = 0.519282
corFFT[27] = 0.922231
corFus[27] = 0.227033

corMax[28] = 0.486702
corDiff[28] = 0.463455
corFFT[28] = 0.831771
corFus[28] = 0.187618

Tras las 28 iteraciones el algoritmo finaliza guardando el resultado de la máxima correlación de cada operación a lo largo del bucle y luego la fusión de las 3. Más concretamente, después de ejecutar la operación que realiza la correlación cruzada, la operación que aplica el diferencial a las señales y las correlaciona, la operación que realiza la FFT a ambas señales y las correlaciona y la fusión de esas 3 correlaciones realizando el producto sobre ellas para obtener un grado de similitud más preciso, el algoritmo se queda con la correlación mayor de cada operación. Para finalizar, el dato que indica con qué patrón guarda más similitud la señal capturada es el número de iteración, o posición dentro de la tabla de patrones, donde se ha calculado la correlación más alta. Es este dato el que se enviará como resultado final de la clasificación.

Las variables MaxCorPos (posición de la máxima correlación cruzada), MaxCorDiffPos (posición de la máxima correlación entre derivadas), MaxCorFFTPos (posición de la máxima correlación entre FFTs) y MaxCorFusPos (posición del producto de las correlaciones máximas anteriores) se han ido actualizando a lo largo del bucle siempre que se obtenía una correlación más alta que la anterior, quedando al final la que cada operación calcula que es la posición correcta para la clasificación.

MaxCorPos = 1
MaxCorDiffPos = 1
MaxCorFFTPos = 1
MaxCorFusPos = 1
El criterio que se va a utilizar en estas pruebas para definir como correcta o no una clasificación es el resultado que proporcione MaxCorFusPos. **Por lo que la señal 1 es clasificada de forma correcta.**

- Señal 2, correspondiente a la posición 2del conjunto patrón:

MaxCorPos = 2
MaxCorDiffPos = 2
MaxCorFFTPos = 2
MaxCorFusPos = 2
Por lo que la señal 2 es clasificada de forma correcta.

- Señal 3, correspondiente a la clasificación del conjunto patrón 3:

MaxCorPos = 2
MaxCorDiffPos = 3
MaxCorFFTPos = 2
MaxCorFusPos = 3

Aquí se observan discrepancias entre los resultados de las operaciones, pero la que más importancia tiene, la que es la salida principal del algoritmo (MaxCorFusPos) **clasifica la señal correctamente**.

- Señal 4, correspondiente a la posición 4 del conjunto patrón:

MaxCorPos = 4
MaxCorDiffPos = 4
MaxCorFFTPos = 4
MaxCorFusPos = 4
Por lo que la señal 4 es clasificada de forma correcta.

- Señal 5, correspondiente a la posición 5 del conjunto patrón:

MaxCorPos = 5
MaxCorDiffPos = 5
MaxCorFFTPos = 5
MaxCorFusPos = 5
Por lo que la señal 5 es clasificada de forma correcta.

- Señal 6, correspondiente a la posición 6 del conjunto patrón:

MaxCorPos = 6
MaxCorDiffPos = 6
MaxCorFFTPos = 6
MaxCorFusPos = 6
Por lo que la señal 6 es clasificada de forma correcta.

- Señal 7, correspondiente a la posición 7 del conjunto patrón:

MaxCorPos = 7
MaxCorDiffPos = 7
MaxCorFFTPos = 7
MaxCorFusPos = 7
Por lo que la señal 7 es clasificada de forma correcta.

- Señal 8, correspondiente a la posición 8 del conjunto patrón:

MaxCorPos = 8
MaxCorDiffPos = 8
MaxCorFFTPos = 8
MaxCorFusPos = 8
Por lo que la señal 8 es clasificada de forma correcta.

- Señal 9, correspondiente a la posición 9 del conjunto patrón:

MaxCorPos = 10
MaxCorDiffPos = 9
MaxCorFFTPos = 22
MaxCorFusPos = 10
Esta señal no es clasificada de forma correcta, ni siquiera con la fusión de las correlaciones.

- Señal 10, correspondiente a la posición 10 del conjunto patrón:

MaxCorPos = 10
MaxCorDiffPos = 22
MaxCorFFTPos = 10
MaxCorFusPos = 10
Por lo que la señal 10 es clasificada de forma correcta.

- Señal 11 correspondiente a la posición 11 del conjunto patrón:

MaxCorPos = 11
MaxCorDiffPos = 11
MaxCorFFTPos = 11
MaxCorFusPos = 11
Por lo que la señal 11 es clasificada de forma correcta.

- Señal 12, correspondiente a la posición 12 del conjunto patrón:

MaxCorPos = 12
MaxCorDiffPos = 28
MaxCorFFTPos = 12
MaxCorFusPos = 28
Por lo que la señal 12 es clasificada de forma incorrecta.

- Señal 13, correspondiente a la posición 13 del conjunto patrón:

MaxCorPos = 13
MaxCorDiffPos = 28
MaxCorFFTPos = 14
MaxCorFusPos = 13
Por lo que la señal 13 es clasificada de forma correcta.

- Señal 14, correspondiente a la posición 14 del conjunto patrón:

MaxCorPos = 14
MaxCorDiffPos = 14
MaxCorFFTPos = 26
MaxCorFusPos = 14
Por lo que la señal 14 es clasificada de forma correcta.

- Señal 15, correspondiente a la posición 15 del conjunto patrón:

MaxCorPos = 15
MaxCorDiffPos = 8
MaxCorFFTPos = 15
MaxCorFusPos = 4
Por lo que la señal 15 es clasificada de forma incorrecta.

- Señal 16, correspondiente a la posición 16 del conjunto patrón:

MaxCorPos = 23
MaxCorDiffPos = 12
MaxCorFFTPos = 24
MaxCorFusPos = 24
Por lo que la señal 16 es clasificada de forma incorrecta.

- Señal 17, correspondiente a la posición 17 del conjunto patrón:

MaxCorPos = 10
MaxCorDiffPos = 9
MaxCorFFTPos = 22
MaxCorFusPos = 10
Por lo que la señal 17 es clasificada de forma incorrecta.

- Señal 18, correspondiente a la posición 18 del conjunto patrón:

MaxCorPos = 18
MaxCorDiffPos = 28
MaxCorFFTPos = 14
MaxCorFusPos = 28
Por lo que la señal 18 es clasificada de forma incorrecta.

- Señal 19, correspondiente a la posición 19 del conjunto patrón:

MaxCorPos = 12
MaxCorDiffPos = 12
MaxCorFFTPos = 6
MaxCorFusPos = 12
Por lo que la señal 19 es clasificada de forma incorrecta.

- Señal 20, correspondiente a la posición 20 del conjunto patrón:

MaxCorPos = 20
MaxCorDiffPos = 20
MaxCorFFTPos = 24
MaxCorFusPos = 20
Por lo que la señal 20 es clasificada de forma correcta.

- Señal 21, correspondiente a la posición 21 del conjunto patrón:

MaxCorPos = 21
MaxCorDiffPos = 21
MaxCorFFTPos = 21
MaxCorFusPos = 21
Por lo que la señal 21 es clasificada de forma correcta.

- Señal 22, correspondiente a la posición 22 del conjunto patrón:

MaxCorPos = 10
MaxCorDiffPos = 10
MaxCorFFTPos = 22
MaxCorFusPos = 10
Por lo que la señal 22 es clasificada de forma incorrecta.

- Señal 23, correspondiente a la posición 23 del conjunto patrón:

MaxCorPos = 23
MaxCorDiffPos = 23
MaxCorFFTPos = 23
MaxCorFusPos = 23
Por lo que la señal 23 es clasificada de forma correcta.

- Señal 24, correspondiente a la posición 24 del conjunto patrón:

MaxCorPos = 24
MaxCorDiffPos = 23
MaxCorFFTPos = 24
MaxCorFusPos = 24
Por lo que la señal 24 es clasificada de forma correcta.

- Señal 25, correspondiente a la posición 25 del conjunto patrón:

MaxCorPos = 21
MaxCorDiffPos = 1
MaxCorFFTPos = 25
MaxCorFusPos = 21
Por lo que la señal 25 es clasificada de forma incorrecta.

- Señal 26, correspondiente a la posición 26 del conjunto patrón:

MaxCorPos = 26
MaxCorDiffPos = 26
MaxCorFFTPos = 26
MaxCorFusPos = 26
Por lo que la señal 26 es clasificada de forma correcta.

- Señal 27, correspondiente a la posición 27 del conjunto patrón:

MaxCorPos = 23
MaxCorDiffPos = 23
MaxCorFFTPos = 11
MaxCorFusPos = 23
Por lo que la señal 27 es clasificada de forma incorrecta.

- Señal 28, correspondiente a la posición 28 del conjunto patrón:

MaxCorPos = 24
MaxCorDiffPos = 28
MaxCorFFTPos = 24
MaxCorFusPos = 24
Por lo que la señal 28 es clasificada de forma incorrecta.

Atendiendo solamente a los resultados de la fusión de las correlaciones se obtiene un porcentaje de éxito del 60%. Resultado que se puede considerar satisfactorio si se tiene en cuenta que será obtenido por un sistema de monitorización no intrusivo y que los datos resultantes son producto de aplicar el algoritmo a las señales recibidas por un solo sensor y no por una malla de ellos. Para mejorar estos resultados se podría continuar entrenando al algoritmo para obtener un conjunto patrón mejor definido.

Cinco

Resultados, Conclusiones y Líneas Futuras

5.1 Resultados y principales aportaciones

Los resultados del trabajo cumplen con los objetivos establecidos en la Sección 1.2. En resumen, se ha desarrollado un nuevo prototipo, basándose en el proyecto Pro-DIA, ampliando tanto la aplicación como el dispositivo y dando como resultado las nuevas versiones DAS220.5 de hardware y TDAS220.2.0.6 de software. A continuación se describen los resultados parciales obtenidos.

Se ha llevado a cabo el estudio del Estado del Arte en redes de sensores inalámbricas y el estudio previo de las tecnologías y algoritmos que han permitido tomar las decisiones sobre la implementación y los componentes del sistema desarrollado en este Proyecto de Fin de Carrera.

Se ha realizado una primera aproximación, en la Sección 4.3.1, a la aplicación final utilizando el dispositivo DAS220.4 y las entradas y salidas auxiliares de las que dispone. Se ha podido introducir un sensor de movimiento analógico y se ha verificado que la aplicación es capaz de manejarlo de forma satisfactoria a través de una serie de pruebas como la comprobación de la frecuencia de muestreo y el envío de tramas de datos habiendo conectado el pin de salida del portasensor a VCC y, después, a masa.

Se han implementado los cambios en la aplicación para añadir el nuevo módulo de memoria RAM, como se vio en la Sección 4.3.3, el módulo del sensor analógico y se ha introducido el algoritmo necesario para cumplir con el requisito de bajo consumo como se muestra en la Sección 4.4.7.

Se ha desarrollado un nuevo prototipo de nodo sensor, proceso que se explica a lo largo de la Sección 4.4, con la incorporación del módulo de memoria RAM externa, necesaria como se pudo comprobar en la Sección 4.1, y del módulo del sensor de movimiento analógico. Este desarrollo ha incluido tanto el diseño a ordenador del circuito impreso del prototipo con el software CAD Altium como su fabricación,

utilizando las herramientas software de control de la impresión del circuito, la microfresadora LPKF C60 y una estación de soldadura, obteniendo como resultado un prototipo, mostrado en la Figura 4.31, totalmente funcional y habiendo realizado las comprobaciones pertinentes enumeradas en la Sección 4.4.7. Algunas de estas verificaciones han consistido en comprobar la continuidad de las pistas y vías, demostrar que es posible cargar la aplicación, ejecutarla y enviar datos a la estación base con los que se construyeron las formas típicas del sensor analógico.

Por último, se ha implementado el algoritmo de clasificación de señales propuesto en la Sección 4.1 demostrando su funcionamiento con 28 señales de ejemplo en la Sección 4.5. Aunque se haya dejado integrado en la aplicación TDAS220.2.0.6 para que se ejecute sin que ninguna interrupción interfiera con el procesado de los datos, aún no se ha conseguido que realice el manejo de las operaciones con las señales y la clasificación de las mismas en el nodo sensor. Esta tarea queda pendiente para futuros desarrollos de la aplicación.

5.2 Conclusiones

A la vista de los resultados obtenidos se han extraído las siguientes conclusiones:

1. El microcontrolador Atmel1281 y el lenguaje NesC junto con TinyOS son las herramientas adecuadas para desarrollar una red de sensores inalámbrica por sus características, destinadas a una mayor eficiencia en el consumo. Las librerías del SO TinyOS son de mucha ayuda a la hora de realizar una programación concurrente para este tipo de aplicaciones, que funcionan ejecutando varios hilos a la vez y necesitan de un sistema de interrupciones y contadores para gestionar las tareas.

2. El compilador de TinyOS presenta problemas a la hora de manejar operaciones matemáticas complejas. Por tanto, resulta necesario que se siguiera desarrollando y solventando los fallos que presenta. Tal vez, incluso, considerar la posibilidad de que estas operaciones se realicen en el PC al que esté conectada la estación base para liberar al nodo sensor de esta carga.

3. Los sensores de movimiento analógico y digital han resultado ser los adecuados para realizar el muestreo del movimiento del sujeto, aunque no hay mucha oferta en el mercado donde poder elegir.

4. La memoria RAM cumple las necesidades de capacidad que se requerían. Que el microcontrolador la añada a su mapa de memoria y la gestione de forma casi automática facilita el trabajo con este tipo de tecnología, donde normalmente se disponen de capacidades limitadas. Uno de los problemas que se ha encontrado para seleccionar el dispositivo ha sido que en el mercado es difícil encontrar memorias RAM inferiores a 1Mb que resulten viables a nivel

económico, es por esto que se decidió adquirir una RAM de 1Mb y capar el bit más significativo con la finalidad de usar sólo la mitad de su capacidad (512Kb), justo la que el microcontrolador es capaz de manejar al disponer de 16 bits para el direccionamiento.

5. La microfresadora ha sido una herramienta muy útil permitiendo construir circuitos impresos sin depender de una empresa externa que realice la fabricación de un simple prototipo que aún no está destinado a su producción en masa. Uno de los problemas encontrados fue que no se disponía de una herramienta para metalizar las vías del circuito y, en algún proyecto más complejo donde haya un número importante de ellas, se perdería demasiado tiempo en metalizarlas a mano.

5.3 Líneas futuras

En la Sección 5.2 se han expuesto algunas conclusiones que dan pie a la apertura de nuevas líneas de trabajo e investigación que podrían resultar en la elaboración de un nuevo proyecto.

1. Con respecto al algoritmo de clasificación de señales implementado en este proyecto, las operaciones utilizadas para encontrar un grado de similitud entre las muestras son satisfactorias e incluso aún más si se fusionan pero quizá haya otro algoritmo más eficiente tanto para el entrenamiento del controlador como para el proceso de clasificación. Hablamos de las redes neuronales artificiales.
   Con este sistema el aprendizaje del controlador sería similar al realizado en el algoritmo de clasificación que se utiliza en este proyecto pero sería interesante para futuras ampliaciones de la investigación el comprobar si se obtendrían mejores resultados

2. Durante la realización de este PFC se ha podido comprobar que el compilador presenta ciertos problemas para manejar operaciones complejas al tiempo que controlaba al resto de la aplicación. Es posible que, debido al uso de un lenguaje de programación tan específico y que, naturalmente, no dispone de una comunidad de usuarios tan amplia como otros lenguajes más extendidos, le falte aún madurez en el desarrollo para poder realizar ciertos cómputos matemáticos. Hay que tener en cuenta que un nodo sensor está ideado para trabajar junto con muchos otros formando una red cuyos datos son trasladados a una estación base con más potencia de procesado.
   Es por esto que una alternativa a implementar en un futuro fuera el envío de las señales capturadas por los sensores sin procesar para que fueran recibidas por un dispositivo u ordenador más potente que realizara las operaciones complejas. Por un lado esta decisión podría plantear un problema energético ya que el envío de muchas más tramas provocaría un aumento del gasto de las baterías, pero por otra parte se ahorraría tiempo en el procesado de datos. La

dificultad de este método estaría entonces en buscar un compromiso entre gasto de energía y velocidad repartiendo las obligaciones de cálculo entre nodo sensor y el ordenador receptor de los datos.

Seis

Apéndice

6.1 Esquemáticos

En este apéndice se muestran los diversos esquemáticos que se han realizado en el proceso de diseño.

6 Apéndice

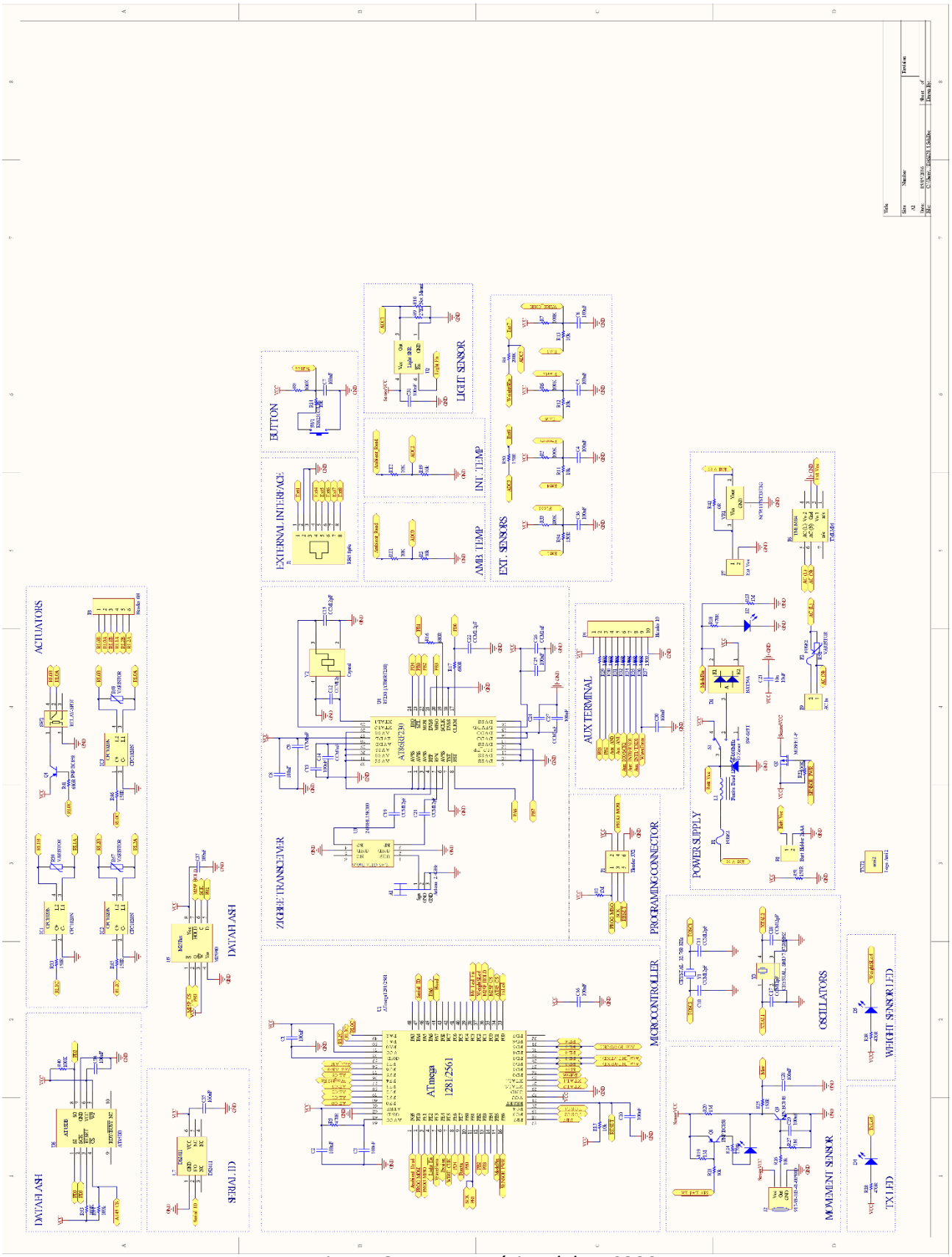


Figura 6.1: Esquemático del DAS220.4

6 Apéndice

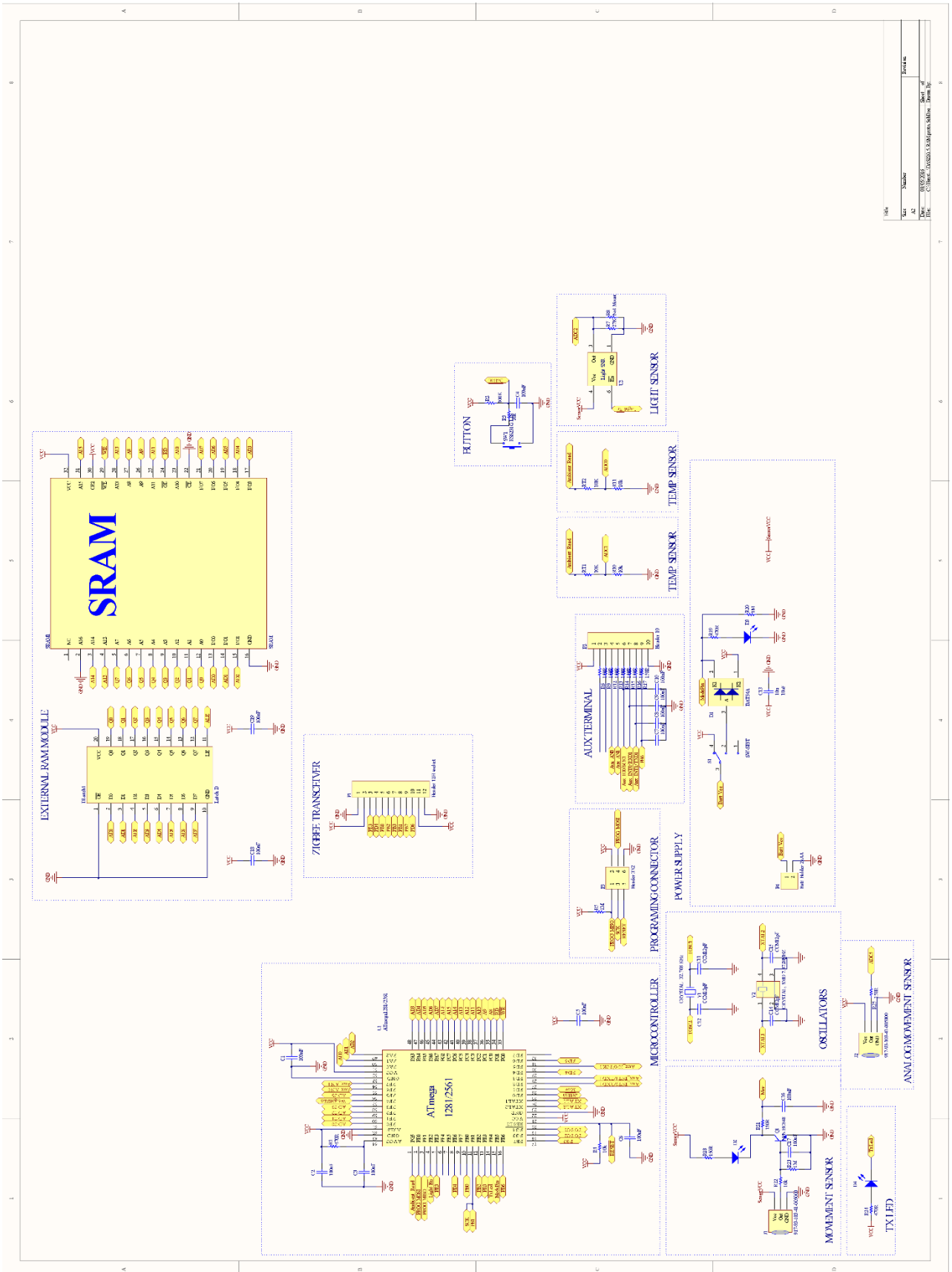


Figura 6.2: Esquemático del DAS220.5.proto

6 Apéndice

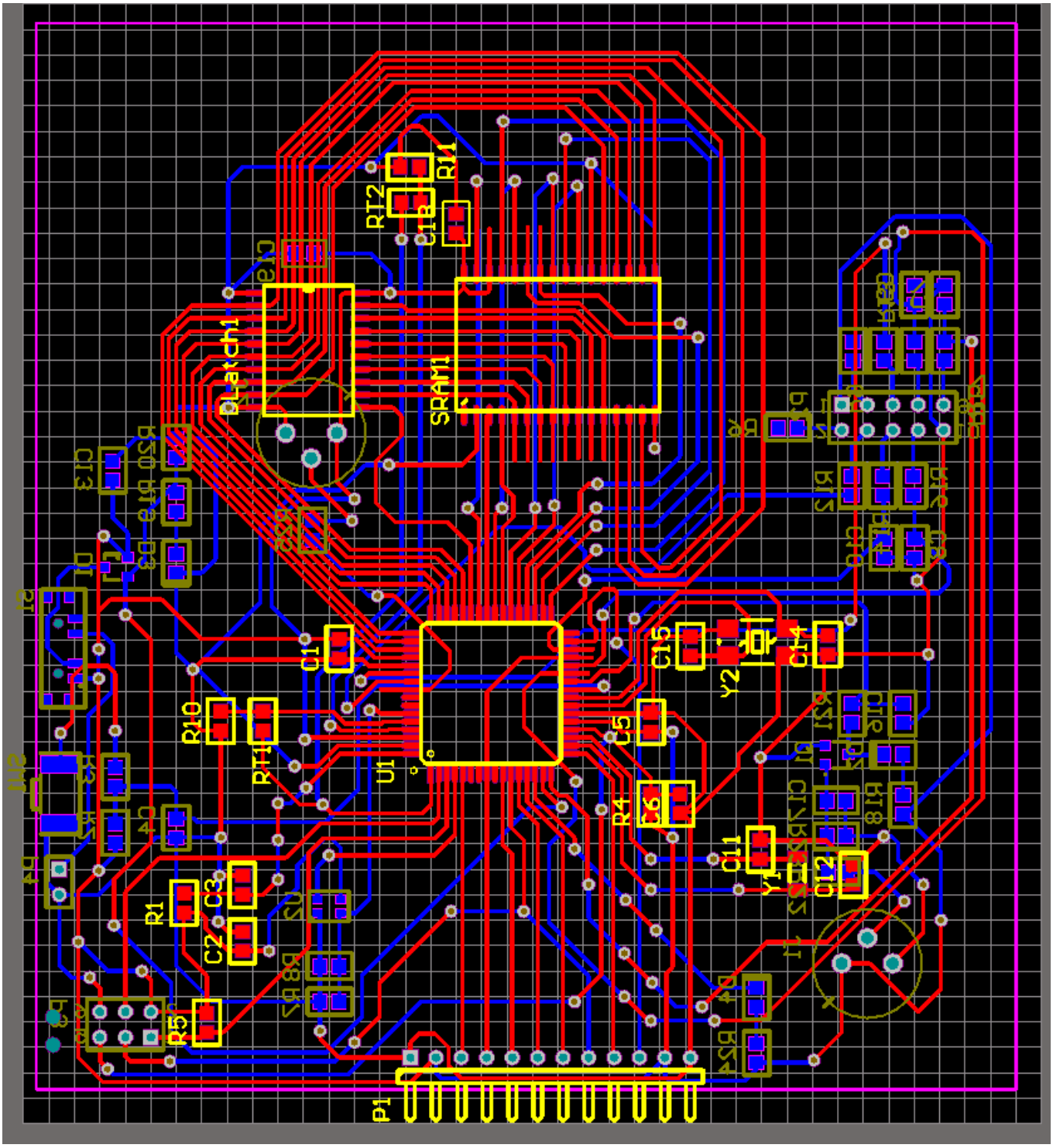


Figura 6.3: Diseño PCB del dispositivo DAS220.5

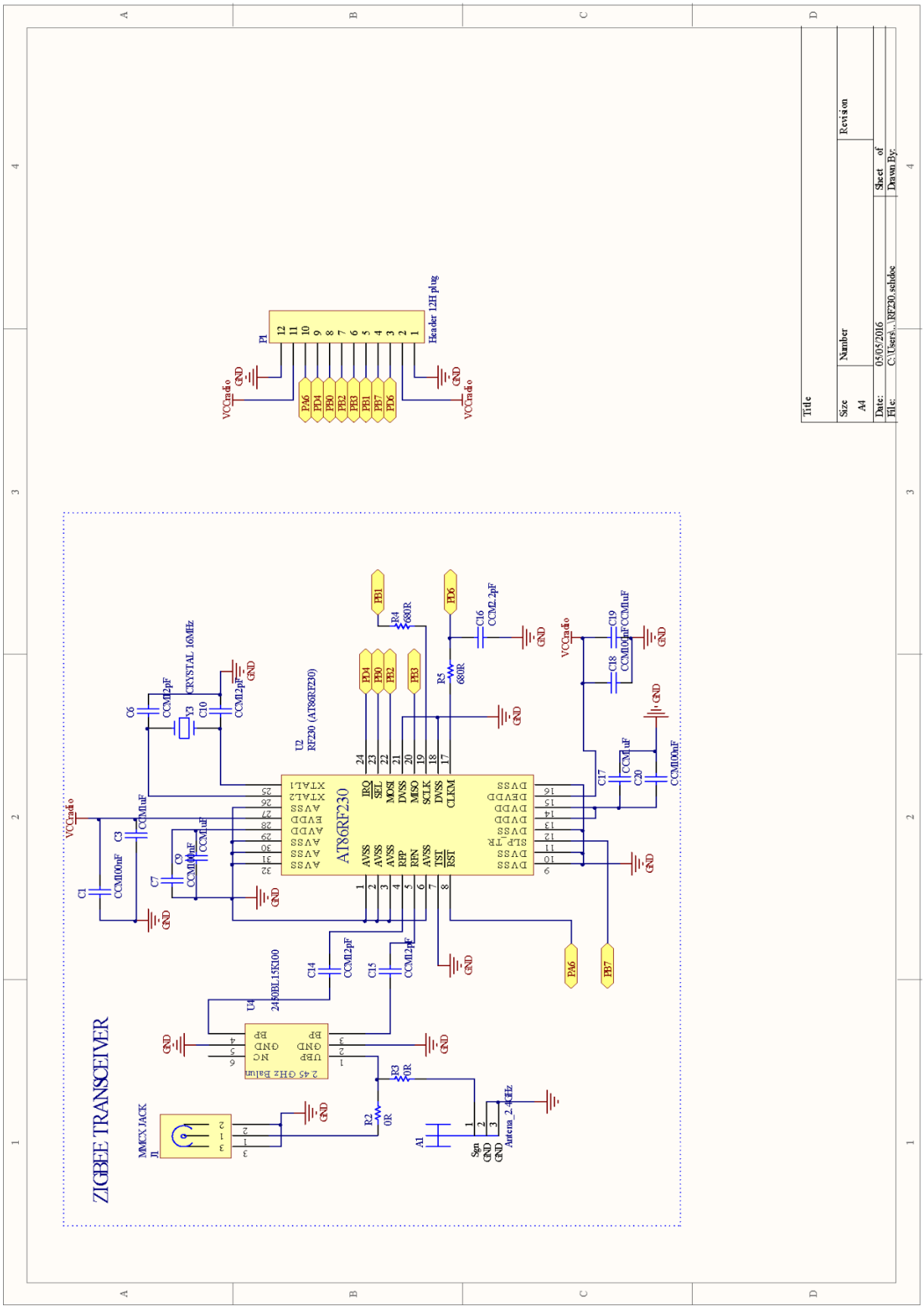


Figura 6.4: Esquemático del módulo de radiofrecuencia

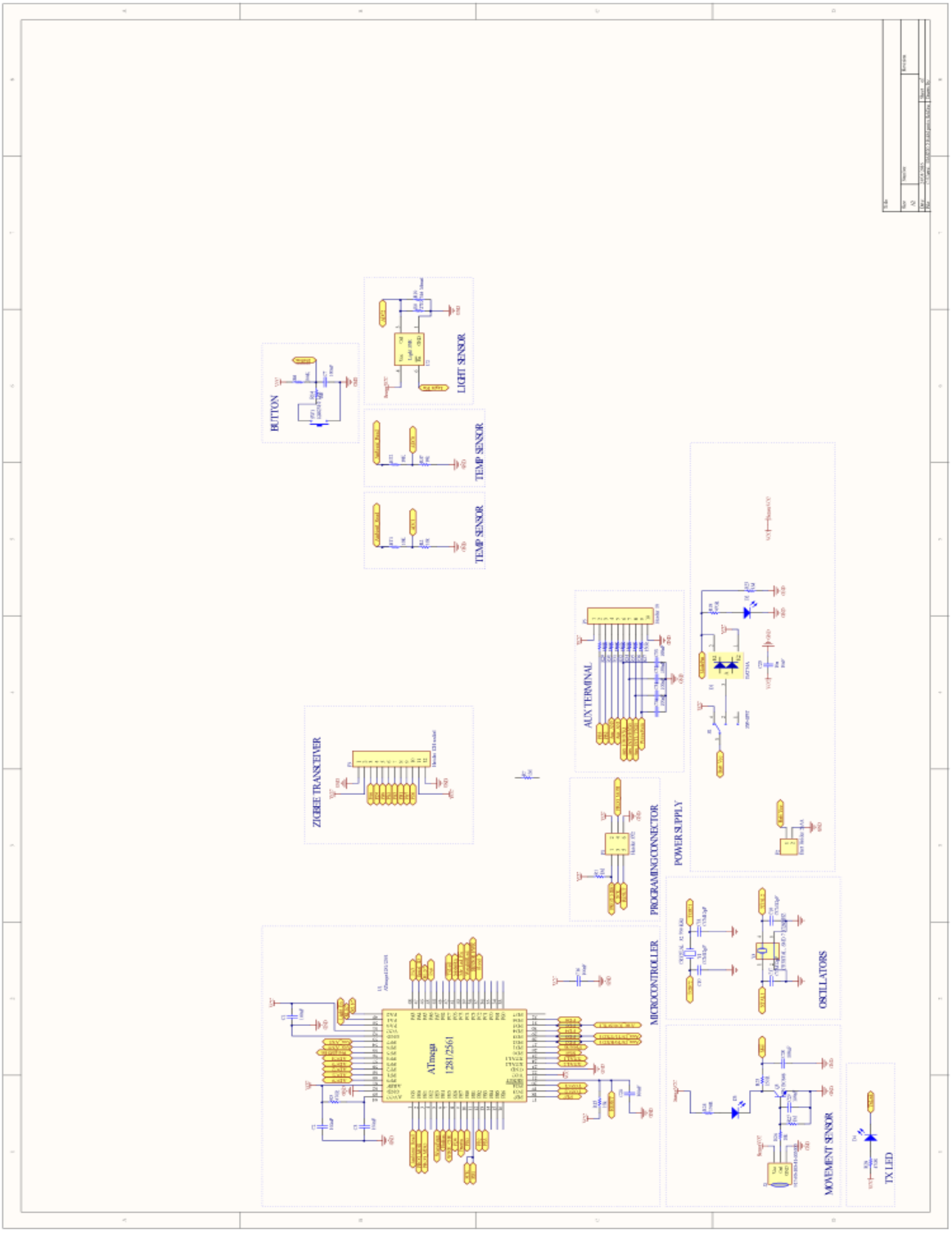


Figura 6.5: Esquemático resultante después del estudio del DAS220.4 para la adición de la RAM

6 Apéndice

Comment	Description	Designator	Footprint	LibRef	Quantity
100nF		C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C16, C17, C18, C19	0805 (2012)	CAP	14
CCM12pF		C11, C12, C14, C15	0805 (2012)	CAP	4
10u	Capacitor	C13	0805 (2012)	Cap	1
BAT54A	SMALL SIGNAL SCHOTTKY DIODE (DUAL)	D1	SOT-23C	BAT54A	1
RED_LED	Typical RED GaAs LED	D2, D3	0805 (2012) LED	LED1	2
YELLOW_LED	Typical YELLOW GaAs LED	D4	0805 (2012) LED	LED1	1
Latch D	74AHC 573 Octal Latch	DLatch1	Latch 74AHC 573	Latch D	1
917-93-103-41-005000	SOCKET TRANSISTOR 3 PIN TO-5	J1, J2	Socket Transistor TO-5	AMN41121	2
Header 12H socket	Header, socket 12-Pin, Right Angle	P1	HDR1X12H	Header 12H	1
Header 10	Header, 10-Pin	P2	HDR2X5	Header 10	1
Header 3X2	Header, 3-Pin, Dual row	P3	HDR2X3	Header 3X2	1
Batt Holder 2xAA	Header, 2-Pin	P4	HDR1X2	Header 2	1
NPN BC848	BC848 NPN	Q1	SOT23A	2N3904	1
470R	Resistor	R1, R19, R24	0805 (2012)	Res1	3
100K	Resistor	R2	0805 (2012)	Res1	1
10k	Resistor	R3, R4, R10, R11, R22	0805 (2012)	Res1	5
1M	Resistor	R5, R20, R23	0805 (2012)	Res1	3
150R	Resistor	R6, R9, R12, R13, R14, R15, R16, R17, R18, R21, R25	0805 (2012)	Res1	11
56K	Resistor	R7, R8	0805 (2012)	Res1	2
THERMISTOR 10K	THERMISTOR R25=10K 0805	RT1, RT2	0805 (2012)	Res1	2
SW-SP3T	SP3T Subminiature Slide Switch, Right Angle Mounting, Vertical Actuation	S1	SP3T PCM13SMTR	SW-SP3T	1
SRAM	1-Mbit (128 K × 8) Static RAM	SRAM1	SRAM	SRAM	1
KSS231G LFS	TACTILE SWITCH, SPST, SMD, 2.5N	SW1	KSS231G LFS	SW2 KSS231G LFS	1
ATmega1281/2561	Microcontroller	U1	TQFP-64 LP=0.8	ATmega1281/2561	1
APDS-9007	Ambient Light Photo Sensor with Logarithmic Current Output	U2	APDS-9007	APDS-9007	1
CRYSTAL 32.768 KHz	CRYSTAL, SMD 32.768 KHz	Y1	CRYSTAL ABS09	CRYSTAL	1
CRYSTAL, SMD 7.3	Surface Mount Quart	Y2	CRYSTAL FQ7050B	85SMX	1

Tabla 6.1: Lista de materiales del DAS220.5

Bibliografía

[1] Ambiental Intelligence & Interaction S.L.L. Proyecto dia, 2007. http: //www.ami2.net/secciones/proyectos.htm.

[2] F.J. Fernández-Luque and J. Zapata. Wireless sensor network for assisted living at home. In *Bioinspired Applications in Aritificial and Natural Computation*, pages 65-74. International Work-Conference on the Interpalay Between Natural and Artificial Computation, IWINAC, Junio 2009.

[3] IEEE 802.14 WPAN Task Group 4 (TG4). Standard ieee 802.14.4-2006. http://www.ieee802.org/15/pub/TG4.html, 2006.

[4] Web oficial de la zigbee alliance, 2009. http://www.zigbee.org/.

[5] Antonio Rosa Rodríguez. Desarrollo de una herramienta para la generación de interfaces gráficas con redes de sensores inalámbricas. Master's thesis, Escuela Técnica Superior de Ingeniería de Telecomunicación. Universidad Politécnica de Cartagena, 2008. http://repositorio.bib.upct.es/dspace/bitstream/10317/805/1/pfc2842.pdf.

[6] Daniela Rus Elizabeth Basha, Sai Ravela. Model-based monitoring for early warning flood detection. *In Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Raleigh, NC, Noviembre 2008.

[7] Memsic, Inc. MEMSIC Completes Crossbow Technology Acquisition, Enero 2010.http://www.memsic.com/about-memsic/news-room.cfm?nid=MEMSIC%20Completes%20Crossbow%20Technology%20Acquisition

[8] Memsic, Inc. Wireless Sensor Networks. http://www.memsic.com/wireless-sensor-networks/

[9] Atmel. *8-bit Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash ATmega640/V ATmega1280/V ATmega1281/V ATmega2560/V ATmega2561/V*, Agosto 2007. http://www.atmel.com/ dyn/resources/prod_documents/2549S.pdf.

[10] Atmel. *AT86RF230: Low Power 2.4GHz Transceiver for ZigBee, IEEE 802.15.4, 6LoWPAN, RF4CE and ISM Applications*, Febrero 2009. http://www.atmel.com/dyn/resources/prod_documents/doc5131.pdf.

[11] Tinyos website, Septiembre 2009. http://tinyos.stanford.edu/tinyos-wiki/index.php/Main_Page.

[12] Philip Levis David Gay, David Culler. *NesC 1.3 Language Reference Manual*, Julio 2009.

[13] Philip Levis. *TinyOS Programming*, rev. 1.3 edition, Octubre 2006. http://www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf.

[14] Atmel. Avrisp mkii in-system programmer. http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3808.

[15] Phil Loughhead. *Tutorial - Getting Started with PCB Design*, Septiembre 2009. http://wiki.altium.com/display/ADOH/Tutorial+-+Getting+Started+with+PCB+Design.

[16] LPKF. *LPKF Protomat C60 manual* http://studio.pataky.hu/edu/2014_2015/angolhoz/lpkf_c60_man.pdf

[17] CircuitCAM. *CircuitCAM 7.2 User guide and reference manual*. 2013. http://www.circuitcam.com/sites/circuitcam.com/files/2013-12-30_CircuitCAM_7-2.pdf

[18] LPKF Electronics. *BoardMaster 5.1 Manual.* 2008. https://www.ece.ubc.ca/~leos/pdf/tools/lpkf/BoardMaster51.pdf

[19] MP Motion Sensor (AMN2, 3, 4). http://pewa.panasonic.com/pcsd/product/sens/pdf/amn.pdf.

[20] F.J. Fernández-Luque. Wireless Sensor Networks for Ambient Intelligence. In *PIR-based motion patterns classification for AmI systems* pages 168-177. Abril 2013.