Open Ensis FSM

Aparato Señalador de Esgrima basado en LPC1769

Memoria TFC – SistemesEncastats

Autor: Julio Fortis Urbano
Enginyeria tècnica en Informàtica de Sistemes

Consultor: Jordi Bécares Ferrés

16/01/2013

Agradecimientos

A Emanuela por su paciencia.
A mis profesores y consultores por su tiempo.
A mi familia y amigos por su apoyo.

Resumen

Este documento busca presentar de una manera ordenada, clara y concisa los procesos seguidos y los resultados obtenidos en el desarrollo de un proyecto realizado dentro del marco del área de “sistemes encastats”.

El proyecto consiste en el desarrollo de un aparato señalizador de tocados de esgrima (Fencing Score Machine o FSM) que asista al arbitraje de un match de esgrima, de acuerdo al reglamento técnico de dicho deporte, utilizando para ello una placa de desarrollo LPC1769.

Para la programación de esta placa se utilizará FreeRTOS, un sistema operativo en tiempo real para sistemas empotrados, que proporcionará una manera sencilla de trabajar muchas de las funcionalidades necesarias para este sistema, como puede ser la gestión de tareas, colas de proceso, semáforos o temporizadores.

Además, será necesario un montaje electrónico mínimo de los componentes para la señalización visual y acústica, así como para le entrada proveniente de las armas de esgrima, que permitirá que el aparato pueda utilizarse de forma autónoma en una pista real de esgrima como asistencia al arbitraje.

De forma adicional, se dotara al sistema de comunicación inalámbrica para permitir su control remoto y para el seguimiento de las variables de estado del aparato mediante la electrónica de salida, tales como la puntuación, el arma seleccionada o gestión del tiempo del match de esgrima.

Para dicho control remoto, se planteará una pequeña aplicación en PHP que servirá de ejemplo de cómo puede ser gestionado el envío de comandos remotos al aparato señalizador y como representar la información reportada por este, de forma que se pueda incorporar en software ya existente (como las aplicaciones de gestión de la competición de esgrima) o a futuros desarrollos que se construyan sobre esta base propuesta.

Índice de Contenidos

Tabla de contenido

Índice de Imágenes

1. Introducción

El objeto de este proyecto es el desarrollo de un aparato señalizador de esgrima (FSM, de las siglas en inglés de “Fencing Score Machine”) basado en la placa LPC1769 que permita la integración con un software de control remoto mediante un módulo Wi-Fi (WiFly RN-XV).

Un aparato señalizador de esgrima es un dispositivo electrónico capaz de determinar si el arma de un deportista ha tocado la superficie válida del otro deportista, en las condiciones de tiempo preestablecidas por el reglamento de esgrima. Estas condiciones son:

- Un tiempo mínimo de contacto del arma con el blanco válido del adversario
- Un tiempo de tocado doble durante el cual después de tocar al adversario, el aparato registra el tocado dado simultáneamente por este

De forma adicional, los aparatos señalizadores, pueden disponer de las funciones básicas de control de un asalto que puede utilizar un árbitro de forma remota durante una competición:

- Selección del tipo de asalto (que determinará la duración de este)
- Inicio y pausa del tiempo del asalto
- Control de la puntuación
- Determinación de la prioridad del asalto

Debido a la complejidad de estas tareas, en los inicios de la esgrima como deporte, el principal problema para el árbitro era determinar precisamente si el arma había tocado al deportista o no.La solución a este problema fue el desarrollo (y posterior obligatoriedad de uso) del aparato de señalización de tocados en 1936.

Por tanto, el objetivo de este proyecto es desarrollar un prototipo que, disponiendo de las funciones comentados, permita en un futuro seguir construyendo funcionalidades no presentes en los aparatos comerciales actuales.

Para la construcción de dicho prototipo, he realizado las siguientes tareas:

- Desarrollo de la funcionalidad del aparato señalizador
- Desarrollo de una librería para la comunicación inalámbrica
- Implementación de un circuito electrónico para la entrada y salida del aparato señalizador
- Desarrollo de un software de control remoto de ejemplo que permita el envío comandos y la recepción del estado del aparato señalizador.

Las iré exponiendo con mayor profundidad en los siguientes capítulos de esta memoria.

1.1 Justificación

Para una persona ajena al mundo de la esgrima, la importancia del aparato señalizador de tocados puede pasar desapercibida, pero este aparto no sólo es una herramienta de ayuda al arbitraje, si no que bien utilizado puede ser una gran ayuda para el maestro de esgrima como herramienta de entrenamiento.

Los aparatos señalizadores de esgrima comerciales actuales, además de tener un precio exorbitante que los deja fuera del alcance no sólo de los deportistas particulares y de los clubs de esgrima más modestos, presentan ciertas limitaciones.

Por un lado en su uso en la competición de esgrima, la mayoría de los aparatos no se pueden integrar con el software de gestión de la competición, por lo que requieren que el árbitro mantenga el resultado de un match en papel y al finalizar lo lleve al directorio técnico de la competición, donde será introducido manualmente en el software.

Por otro lado, estos aparatos no permiten personalizar o configurar opciones técnicas como el tiempo de contacto o la tolerancia del tocado doble, lo que sería muy interesante tanto para diseñar entrenamientos especializados, como para la actualización de dichos valores cuando los tiempos cambian en el reglamento. Actualmentecuando esto ocurre, un aparato comercial se debe enviar al servicio técnico de su fabricante para la actualización del firmware.

Así, sería de esperar que un aparato señalizador de bajo coste que permitiese la personalización y ajuste de los valores expuestos por parte del usuario y que pudiese integrarse con un software de gestión de la competición y/o aplicaciones de control remoto, fuese bien recibido por los practicantes de este deporte.

1.2 Descripción

El aparato señalizador es un pequeño sistema empotrado autónomo que permite comunicarse con un sistema remoto de control utilizando un sistema de comunicación inalámbrico.

Los requisitos básicos del proyecto son que, partiendo de un hardware ya diseñado y seleccionado (LPC1769, WiFly…), se deberá programar este dispositivo para realizar las tareas necesarias de asistencia al arbitraje y cumplir los objetivos del proyecto.

1.3 Objetivos

El objetivo es conseguir una plataforma que permita experimentar con los conceptos desarrollados durante la ingeniería y que además sirva como prototipo sobre el que desarrollar posteriormente un sistema completo de asistencia al arbitraje.

El sistema debe ser capaz de:

- Detectar cuando se ha producido un tocado
- Señalizar visual y acústicamente dicho tocado
- Permitir o bloquear un tocado doble de acuerdo al reglamento de esgrima
- Recibir comandos remotos de un software de control
- Enviar el estado del aparato (puntuación, tiempo, etc.) a un software externo

Es decir, de forma general, los objetivos son dotar mediante el firmware desarrollado a la placa de la funcionalidad de un aparato señalizador de tocados que pueda utilizarse con material de esgrima estándar.

1.4 Enfoque y Método seguido

A grandes rasgos, he utilizado un ciclo de desarrollo iterativo e incremental basado en el ciclo clásico (en cascada), separando cada módulo en las etapas de Análisis, Diseño, Codificación y Prueba. Aunque en algunos momentos puntuales he optado por un desarrollo más cercano a la programación exploratoria.

El proyecto se puede dividir en cuatro grandes bloques, que se han desarrollado de manera secuencial con la salvedad del bloque de documentación, que ha sido transversal a los otros tres.

1.4.1 Fase de desarrollo de la librería y aprendizaje

Durante esta fase he desarrollado una serie de librerías/drivers y pequeños programas de prueba (de acuerdo a las PACs) con el objetivo de familiarizarme con el entorno y producir código reusable en las fases siguientes del desarrollo.

Los productos de esta fase que se han reutilizado en las fases posteriores del proyecto están englobados en la librería TFC_UART_IO, que contiene básicamente la funcionalidad para el manejo de la WiFly, UART, arp@lab, etc.

1.4.2 Primera fase del desarrollo

A partir de esta fase he comenzado el desarrollo del proyecto propiamente dicho, es decir, del aparato señalizador de tocados.

En esta primera fase, el desarrollo ha estado enfocado a la programación de un sistema autónomo capaz de detectar un tocado simple y/o un tocado doble producido dentro de las normas de tiempo establecidas por el reglamento de esgrima.

He incorporado a la librería TCF_UART_IO la funcionalidad para el manejo de GPIO.

1.4.3 Segunda fase del desarrollo

En esta fase, he ido incorporando el resto de la funcionalidad al proyecto. Es decir, se he añadido al susodicho sistema autónomo las funciones para la recepción y procesado de comandos y el envío de información mediante la WiFly.

He programado un servidor de control remoto de ejemplo y he incorporado la funcionalidad a la librería TFC_UART_IO.

1.4.4 Documentación

Durante todas las fases de desarrollo, he producido una serie de documentación técnica (pruebas, búsquedas de información, diagramas…) que ha cumplido tanto una función de control (realización de las PACs), como una labor preparativa para la memoria final (este documento).

Obviamente, según han cambiado los objetivos y planificaciones del proyecto, ha sido necesario modificar y ampliar esta documentación.

Por último, la última fase del proyecto ha sido la redacción y construcción final de esta memoria, para la que he intentado aprovechartoda la documentación apropiada previamente generada.

1.5 Planificación

Presentaré la planificación mediante dos diagramas de Gantt, uno con la planificación inicial, la que se esperaba inicialmente, y otro con la temporización real.

1.5.1 Planificación Inicial

	Name	Duration	Start	Finish
1	**Fase previa**	63,5 days	**19/09/12 18.00**	**24/11/12 16.00**
2	Instalacion del entorno	18,75 days	19/09/12 18.00	08/10/12 0.00
3	Conectividad mediante WiFly	18,75 days	19/09/12 18.00	08/10/12 0.00
4	Driver UART	12 days	10/10/12 18.00	22/10/12 0.00
5	Driver WiFly	12,5 days	22/10/12 18.00	04/11/12 16.00
6	Sistema productor/consumidor	12 days	11/11/12 16.00	24/11/12 16.00
7	**Primera Fase**	10,5 days	**10/12/12 18.00**	**22/12/12 0.00**
8	**Desarrollo bloque de tocado**	3 days	**10/12/12 18.00**	**14/12/12 0.00**
9	Documentacion sobre GPIO	2 days	10/12/12 18.00	12/12/12 22.00
10	Conexiones de Entrada	1 day	12/12/12 22.00	14/12/12 0.00
11	**Implementacion del bloque**	2,5 days	**10/12/12 18.00**	**13/12/12 20.00**
16	**Desarrollo bloque determinació**	2,5 days	**14/12/12 18.00**	**16/12/12 11.00**
17	Analisis	0,5 days	14/12/12 18.00	14/12/12 22.00
18	Diseño	0,5 days	14/12/12 22.00	15/12/12 11.00
19	Codificación	1 day	15/12/12 11.00	15/12/12 22.00
20	Prueba	0,5 days	15/12/12 22.00	16/12/12 11.00
21	**Bloque de salida**	3 days	**16/12/12 11.00**	**19/12/12 20.00**
22	Documentacion sobre GPIO	2 days	16/12/12 11.00	18/12/12 0.00
23	Conexiones de salida	1 day	18/12/12 18.00	19/12/12 20.00
24	**Implementacion del bloque**	2,5 days	**16/12/12 11.00**	**18/12/12 22.00**
29	Implementacion, correcciones y prue	2 days	19/12/12 20.00	22/12/12 0.00
30	**Segunda Fase**	11 days	**22/12/12 9.00**	**01/01/13 22.00**
31	**Desarrollo bloque control del as**	5 days	**22/12/12 9.00**	**26/12/12 22.00**
32	Establecer comandos y configuraci	1 day	22/12/12 9.00	22/12/12 20.00
33	**Implementacion del bloque**	4 days	**22/12/12 20.00**	**26/12/12 22.00**
38	**Desarrollo bloque comunicacion**	4 days	**26/12/12 22.00**	**30/12/12 18.00**
39	Incorporar WiFly al proyecto	1 day	26/12/12 22.00	28/12/12 0.00
40	**Implementacion del bloque**	4 days	**26/12/12 22.00**	**30/12/12 18.00**
45	Desarrollo bloque servidor	1 day	30/12/12 18.00	31/12/12 20.00
46	Implementacion, correcciones y prue	1 day	31/12/12 20.00	01/01/13 22.00
47	**Documentacion**	98,75 da...	**08/10/12 20.00**	**19/01/13 18.00**
48	PAC1	1 day	08/10/12 20.00	09/10/12 22.00
49	PAC2	1 day	09/10/12 22.00	11/10/12 0.00
50	PAC3	2 days	04/11/12 16.00	06/11/12 20.00
51	Propuesta de proyecto	7 days	24/11/12 16.00	01/12/12 18.00
52	Entrega final codigo	0,5 days	02/01/13 8.00	02/01/13 22.00
53	Memoria	25,5 days	20/12/12 8.00	15/01/13 0.00
54	Presentacion	3 days	16/01/13 8.00	19/01/13 18.00

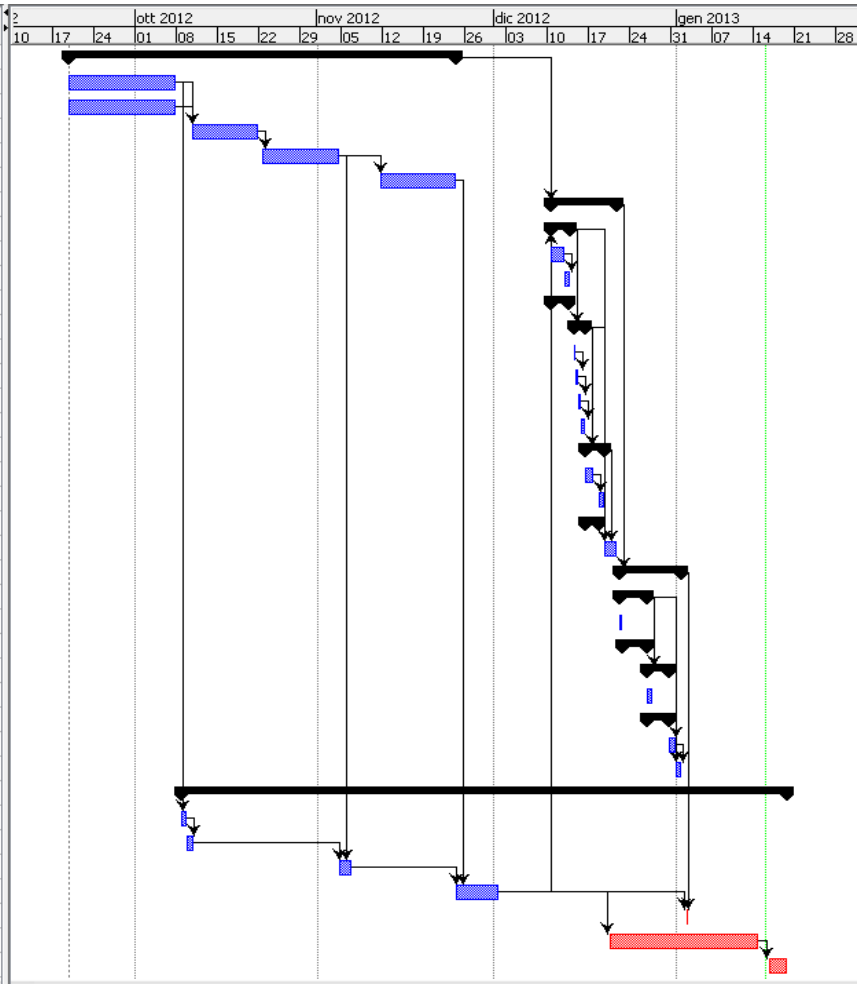

Figura 1. Diagrama de Gantt Previo

1.5.2 Temporización Real

	Name	Duration	Start	Finish
1	**Fase previa**	58,375 d...	**10/10/12 18.00**	**09/12/12 17.00**
2	Instalacion del entorno	1 day	10/10/12 18.00	11/10/12 20.00
3	Conectividad mediante WiFly	1 day	10/10/12 18.00	11/10/12 20.00
4	Driver UART	10 days	13/10/12 20.00	23/10/12 22.00
5	Driver WiFly	39,625 days	23/10/12 22.00	03/12/12 21.00
6	Sistema productor/consumidor	5,5 days	03/12/12 21.00	09/12/12 17.00
7	**Primera Fase**	8,5 days	**10/12/12 20.00**	**19/12/12 22.00**
8	**Desarrollo bloque de tocado**	3 days	**10/12/12 20.00**	**14/12/12 20.00**
9	Documentacion sobre GPIO	1 day	10/12/12 20.00	11/12/12 22.00
10	Conexiones de Entrada	1 day	11/12/12 22.00	13/12/12 0.00
11	**Implementacion del bloque**	1 day	**13/12/12 18.00**	**14/12/12 20.00**
16	**que determinación del tocado**	1 day	**14/12/12 20.00**	**15/12/12 16.00**
17	Analisis	0,25 days	14/12/12 20.00	14/12/12 22.00
18	Diseño	0,25 days	14/12/12 22.00	15/12/12 0.00
19	Codificación	0,25 days	15/12/12 9.00	15/12/12 11.00
20	Prueba	0,25 days	15/12/12 11.00	15/12/12 16.00
21	**Bloque de salida**	3,5 days	**15/12/12 16.00**	**18/12/12 20.00**
22	Documentacion sobre GPIO	0,5 days	15/12/12 16.00	15/12/12 20.00
23	Conexiones de salida	0,5 days	15/12/12 20.00	16/12/12 0.00
24	**Implementacion del bloque**	1 day	**17/12/12 18.00**	**18/12/12 20.00**
29	Implementacion, correcciones y prue	1 day	18/12/12 20.00	19/12/12 22.00
30	**Segunda Fase**	13,5 days	**20/12/12 18.00**	**03/01/13 0.00**
31	**Desarrollo bloque control del as**	10,5 days	**20/12/12 18.00**	**30/12/12 18.00**
32	Establecer comandos y configuraci	2 days	20/12/12 18.00	22/12/12 16.00
33	**Implementacion del bloque**	8,5 days	**22/12/12 16.00**	**30/12/12 18.00**
38	**Desarrollo bloque comunicacion**	1 day	**30/12/12 20.00**	**31/12/12 22.00**
39	Incorporar WiFly al proyecto	1 day	30/12/12 20.00	31/12/12 22.00
40	**Implementacion del bloque**	1 day	**30/12/12 20.00**	**31/12/12 22.00**
45	Desarrollo bloque servidor	1 day	31/12/12 22.00	02/01/13 0.00
46	Implementacion, correcciones y prue	0,75 days	02/01/13 18.00	03/01/13 0.00
47	**Documentacion**	96,5 days	**11/10/12 20.00**	**19/01/13 18.00**
48	PAC1	1 day	11/10/12 20.00	12/10/12 22.00
49	PAC2	1,75 days	01/12/12 22.00	03/12/12 0.00
50	PAC3	1 day	09/12/12 21.00	10/12/12 23.00
51	Propuesta de proyecto	7 days	03/12/12 16.00	10/12/12 20.00
52	Entrega final codigo	0,5 days	03/01/13 18.00	03/01/13 22.00
53	Memoria	12 days	03/01/13 8.00	15/01/13 0.00
54	Presentacion	3 days	16/01/13 8.00	19/01/13 18.00

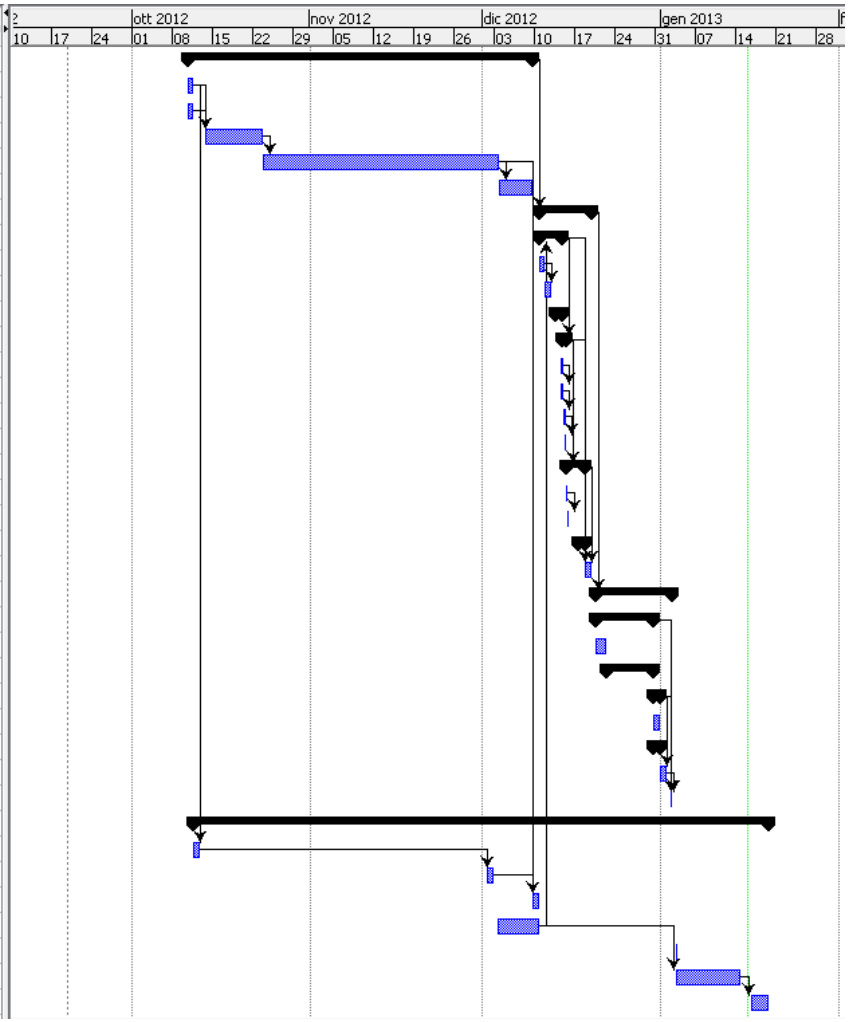

Figura 2. Diagrama de Gantt Posterior

1.5.3 Explicación de la planificación

Como se puede observar en los dos diagramas anteriores, se han producido tres grandes alteraciones entre la planificación inicial y la temporización real.

El primer problema se presentó directamente al inicio de la fase previa. Debido a un retraso en la llegada del material, no sólo se redujeron drásticamente los tiempos para la realización de las tareas, si no que dicho retraso se fue acumulando en todas las tareas posteriores.

La segunda gran diferencia, probablemente la más significativa, fue una extensión anormalmente grande del tiempo requerido para el desarrollo del driver la WiFly, debido a causas personales imprevistas que me afectaron como desarrollador impidiéndome avanzar durante casi un mes.

La última diferencia a destacar se encuentra durante la fase final del desarrollo, cuando el tiempo real disponible resultó menos del estimado provocando que se hayan reducido los tiempos para el testeo y últimas correcciones en el código del proyecto.

1.6 Recursos Utilizados

A continuación presento la lista de los recursos que se han utilizado en el desarrollo del proyecto, divididos en categorías.

1.6.1 Recursos materiales

LPC1769:

Placa principal del sistema

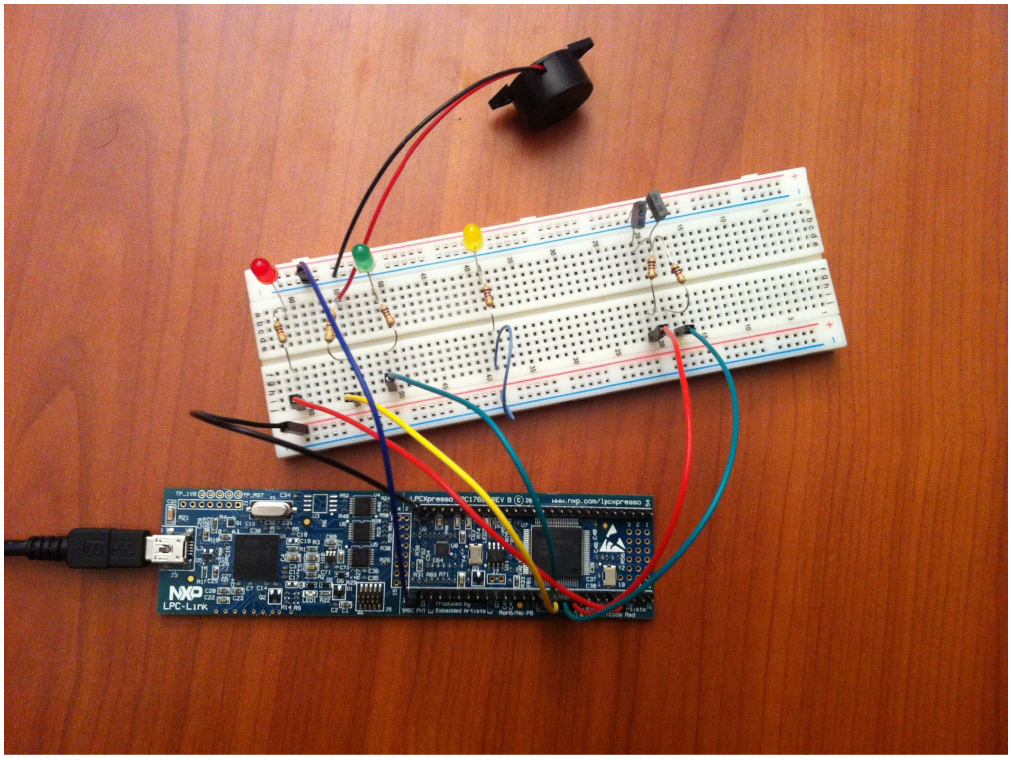

Figura 3. LPC1769 conectada a la breadboard

WiFly RN-XV:

Utilizada para la comunicación inalámbrica

Figura 4. WiFly

Adaptador Serial CP2102:

Para el debug mediante conexión UART

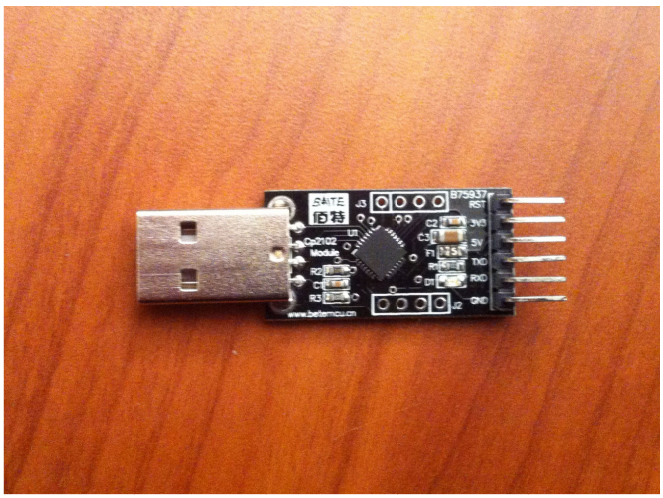

Figura 5. CP2102

Otro material electrónico:

Breadboard, Buzzer, LEDs, resistencias, condensadores, cables tipo jumper…

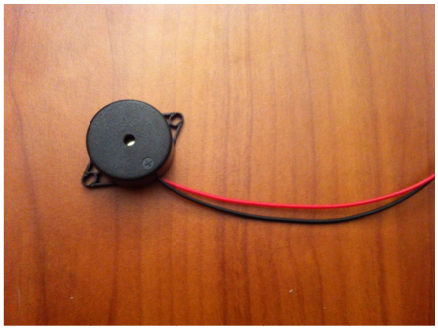

Figura 6. Buzzer

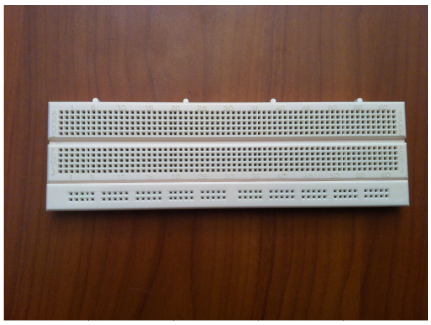

Figura 7. Breadboard

Figura 8. Cables tipo jumper

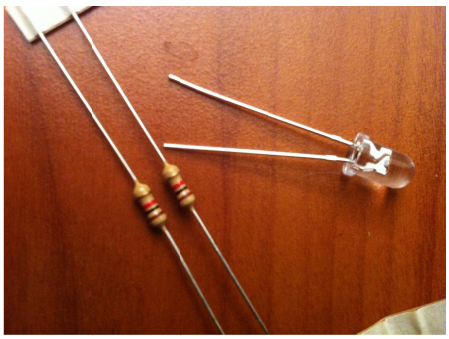

Figura 9. Resistencias y LED

Figura 10. Polimetro y pinzas

1.6.2 Lenguajes de programación

Para el desarrollo del firmware, he optado por programar en un lenguaje de alto nivel, C, ya que me facilitaba la gestión y manejo del código respecto a desarrollar directamente en un lenguaje ensamblador.

Así, en el desarrollo del firmware, he empleado el IDE LPCXpresso v4.3.0. Este IDE, además de facilitar la tarea de codificación, genera el archivo .HEX y permite cargarlo de forma transparente al usuario en la memoria flash del microcontrolador.

Por otra parte, la aplicación de control remoto de ejemplo la he desarrollado en PHP. Debido a la sencillez del código, no ha sido necesario utilizar nada mas que un editor de texto simple para realizar las 4 paginas que lo componen.

1.6.3 Material de consulta

Durante el desarrollo de este proyecto, he utilizado diverso material de consulta.

Se puede consultar una lista en el capítulo "Bibliografía" de esta memoria.

1.6.4 Otros recursos utilizados

Aunque no forman parte específica del proyecto, para la prueba y testeo de las funcionalidades, he utilizado el siguiente material de esgrima: pasantes, enchufes de pasante, espada, sable, chaquetilla eléctrica de sable, máscara de sable...

1.7 Productos obtenidos

Durante el desarrollo de este proyecto, se han obtenido tres productos:

- **EnsisFSM:** Es el código que permite programar una placa LPC1769 para realizar las funciones de aparato señalizador de tocados. Para su correcto funcionamiento es necesario también el montaje de la parte electrónica del prototipo (LEDs, Buzzer, puertos de entrada…).

- **EnsisServer:** Es un código de ejemplo en PHP que demuestra como se pueden enviar comandos remotos al FSM y como se pueden mostrar los datos que este devuelve.

- **Librería TFC_UART_IO:** Es una librería que contiene las funciones reutilizables independientes del proyecto final.

Conestos productos, se puede realizar una instalación "real" modificando los datos de configuración del EnsisServer presentes en EnsisFSM, o partiendo de la base del código de ejemplo, se podría desarrollar un servidor propio o incorporar la funcionalidad en una aplicación de gestión de competiciones propia ya existente.

1.8 Descripción del resto de capítulos

En el resto de capítulos haré un análisis detallado de las tecnologías, procesos y funcionalidades finales.

- En el capitulo 2 se mostrará el estado de la tecnología actualmente en el mercado y se expondrá un pequeño estudio de mercado para nuestro FSM.
- En el capitulo 3 se hará una descripción funcional del sistema.
- En el capitulo 4 se profundizará de forma técnica en el funcionamiento del sistema
- En el capitulo 5 se hará un breve resumen de los conceptos esgrimísticos utilizados en el marco de nuestro sistema
- En los capítulos 6 y 7 se tratará la viabilidad técnica y la valoración técnica del proyecto
- En el capitulo 8 se presentarán las conclusiones finales
- En los capítulos 9, 10 y 11 se encuentra un glosario, la bibliografía utilizada y diversos anexos

2. Antecedentes

2.1 Estado del arte

Un sistema empotrado es un sistema informático (normalmente rodeado de un hardware, como puertos de entrada/salida, memoria, etc.) dedicado a una tarea específica concreta, como el control de pequeñas máquinas o dispositivos.

Muchas veces encontraremos sistemas empotrados en dispositivos con exigencias de computación en tiempo real, es decir aquellos sistemas cuya respuesta no puede ser relativa al flujo del programa, ya que se encuentran sometidos a limitaciones de tiempo absolutas.

La caída del precio de los microcontroladores en los últimos años ha favorecido que hoy en día se utilicen sistemas empotrados en áreas donde antes la electrónica analógica era común, o incluso en áreas donde antes nunca se habría planteado utilizar un sistema informático.

Así, hoy en día esta tecnología es muy común en áreas como:

- Equipo médico
- Sistemas de telecomunicaciones
- Electrónica de consumo
- Domótica
- Automoción
- Redes de sensores
- Robótica y electrónica industrial

Y como veremos en el siguiente apartado, también se ha utilizado en el mundo de la señalización electrónica de tocados en la esgrima.

2.2 Estudio de mercado

Los primeros aparatos señalizadores de esgrima que se desarrollaron utilizaban relés telefónicos para detectar circuitos eléctricos abiertos o cerrados. Posteriormente, en los años 70 se sustituyeron estos relés por transistores, mucho más robustos y fiables y con tiempos de calibración más exactos. Finalmente, a partir de los 80 se pasó a la tecnología de microcontroladores que permitían algunas funciones adicionales de gestión del asalto.

Hasta el desarrollo de los primeros aparatos con microcontrolador, los aparatos señalizadores de tocado se podían utilizar sólo con un arma. Es decir, existían 3 versiones distintas del aparato incompatibles entre sí.

El primer FSM con un microcontrolador (y que además permitía su actualización cuando cambiaban los tiempos) fue comercializado por la empresa Allstar en 1981 y el primer FSM universal (utilizable con las tres armas) vio la luz en 1987, comercializado también por Allstar.

En contra de lo que podría pensarse de un producto que va dirigido a un publico relativamente pequeño, en este caso a los practicantes de un deporte con una base reducida de participantes, actualmente existen en el mercado bastantes productos con la misma finalidad que el que se presenta en este proyecto.

Estos productos normalmente son adquiridos por clubes locales o federaciones deportivas nacionales/regionales para ser utilizados en los circuitos de competición y en los centros de tecnificación.

Así, el primer grupo de alternativas que encontramos a nuestro producto son los aparatos señalizadores comerciales, que incluyen obviamente los aparatos fabricados por los principales productores de material para la esgrima. Casas como Leon Paul, PBT o Allstar distribuyen dichos aparatos (a fecha de la elaboración de esta memoria) por cantidades

comprendidas entre los 300€ y los 5000€ (siendo obviamente los aparatos por encima de los 1000€ los dirigidos a la alta competición).

En los últimos años, han comenzado a aparecer nuevos aparatos, generalmente no homologados por la FIE, dirigidos a pequeños clubs y particulares, que se sitúan en un rango entre los 200 y 300€, quedando como ejemplo los fabricados por Favero.

Entre las alternativas a los FSM tradicionales, podemos encontrar implementaciones por software (utilizando un PC equipado con un dispositivo de conexión) que se presenta como una alternativa novedosa en el momento de la redacción de esta memoria.

Más cerca de la temática de este proyecto, encontramos la implementación mediante un microcontrolador de Eric Schlaepfer. Esta implementación, publicada con licencia GPL, ha dado lugar a dos familias de productos. La familia comercial Eigertek Eclipse, y la implementación libre llamada GNU-GPL-3WT.

Por ultimo, cabe destacar una iniciativa muy similar a este proyecto, de implementar un FSM mediante una placa arduino, que aun no ha producido ningún tipo de producto o información reseñable.

3. Descripción Funcional

Comenzaré por una exposición general del sistema para después entrar en el detalle de cada una de las partes. Es decir, iremos de lo general a lo específico.

3.1 Sistema Total

El sistema se divide en dos partes: EnsisFSM, que corresponde al aparato señalador de tocados, y EnsisServer, que corresponde con la visualización remota y el envío de comandos .

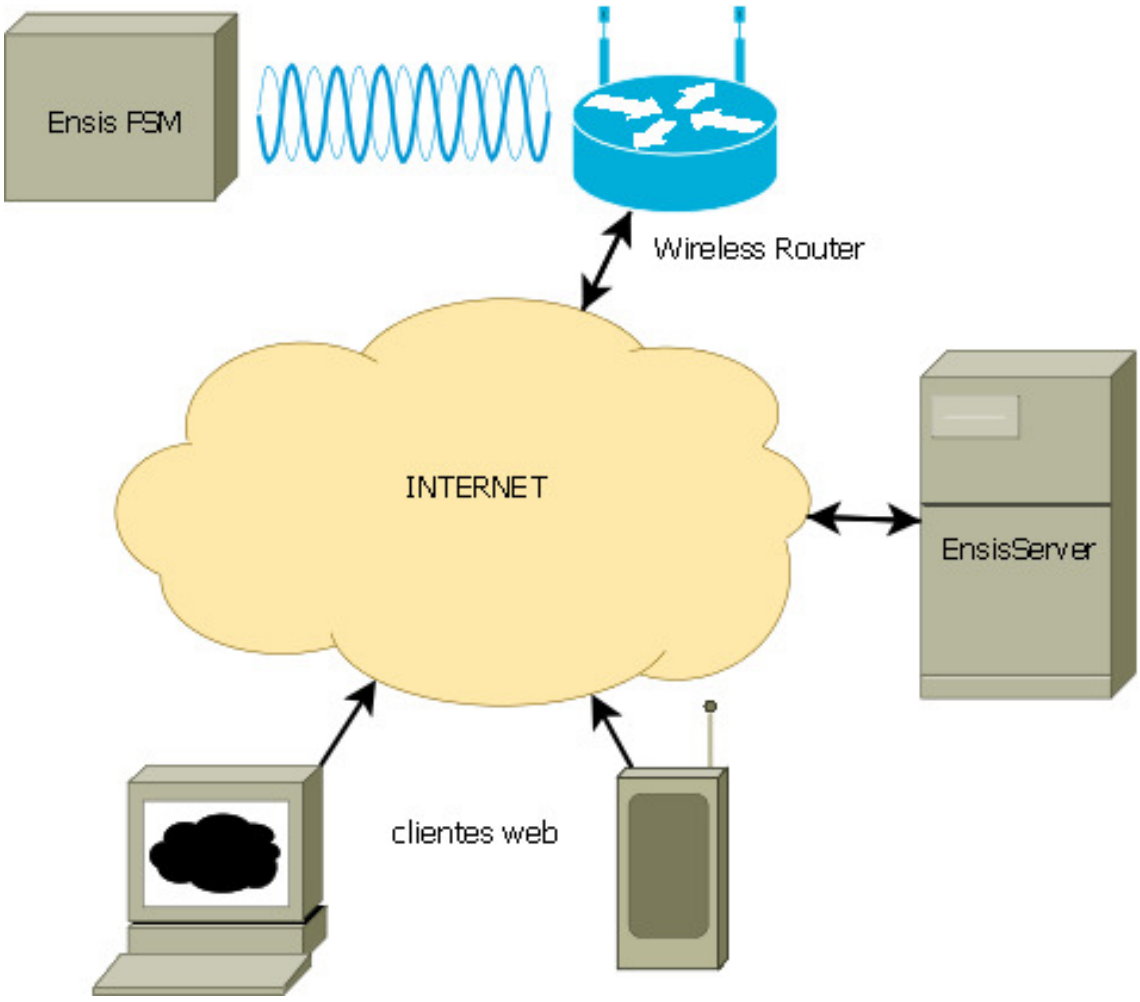

Figura 11. Sistema

EnsisServer es una aplicación web y, por tanto, puede ser accedida desde distintos dispositivos tales como ordenadores o smartphones.

EnsisFSM puede funcionar de manera autónoma a EnsisServer, pero sin este sólo podrá funcionar en modo libre (sin control de puntos ni de tiempo). Este modo de funcionamiento autónomo, es habitual en un aparato señalizador de tocados (de hecho, muchos FSM no cuentan con otro tipo de funcionalidad), queda justificado ya que en muchas ocasiones se utiliza sin árbitro en sesiones de entrenamiento, donde dos tiradores necesitan saber solamente quién ha tocado a quién.

Para la comunicación entre EnsisFSM y EnsisServer, es necesario que exista una red inalámbrica (por tanto un router wi-fi o un punto de acceso wireless son necesarios) para que EnsisFSM pueda conectarse. Una vez cumplido este requisito, EnsisServer puede encontrarse en cualquier punto accesible mediante HTTP desde dicha red, ya sea en la misma, en internet, etc. Igualmente, los clientes que se conecten a EnsisServer pueden encontrarse en cualquier punto desde donde alcancen al servidor. Esto permitiría por ejemplo a múltiples clientes remotos seguir el desarrollo de un match por internet (algo similar a lo que hacen algunos programas de gestión de competiciones).

El sistema interactúa de la siguiente forma. EnsisFSM se activa y comienza a enviar su estado a EnsisServer. EnsisServer puede preparar un comando para el FSM que este leerá y procesará.

3.2 Parte PC

Ya que EnsisServer es sólo un ejemplo de cómo puede mostrarse la información de estado de EnsisFSM, su funcionamiento es muy sencillo.

EnsisServer se compone de una página principal (ver interface en las figuras 12 y 13) donde se muestra el estado de EnsisFSM y un formulario que permite enviar los comandos (tipo de asalto, arma, puntos, tiempo, etc.…). La estructura sería, por tanto, en cuatro bloques nombrados desde el punto de vista de la placa (los nombres pueden parecer al revés).

Figura 12. View - Figura 13. Enviar comando

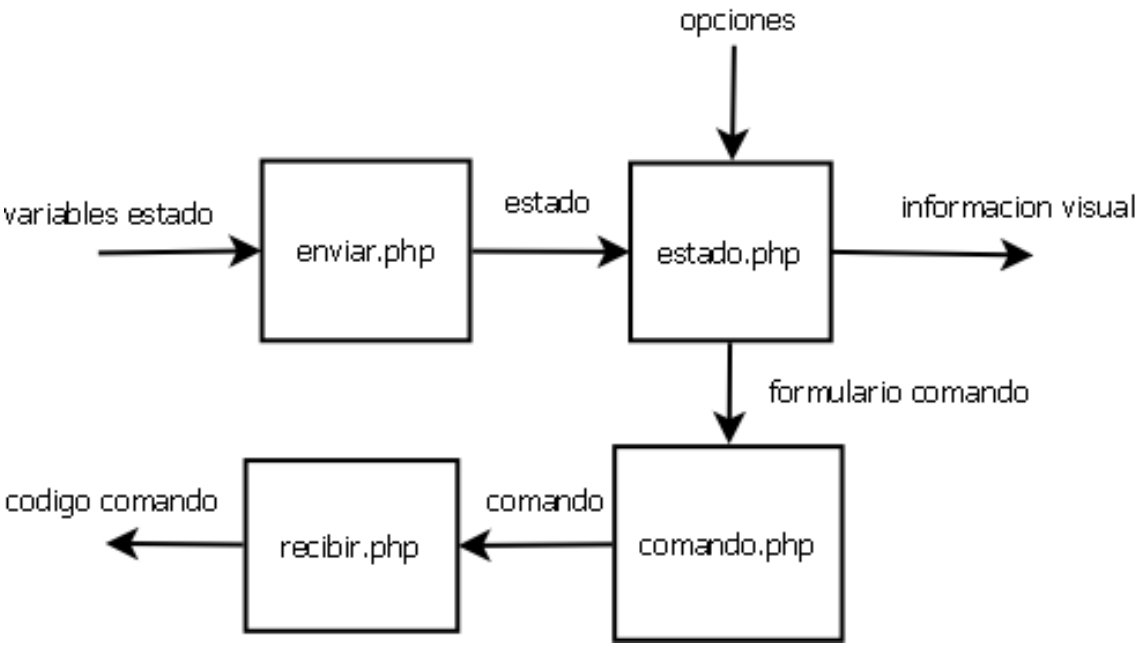

Figura 14. Diagrama de bloques

Como podemos observar, el bloque enviar recibe las variables de estado del FSM y las pasa al bloque estado (que es el único con el que interactúa el usuario). Si el usuario selecciona alguna opción, esta se pasa al bloque comando que genera el comando para el FSM. Finalmente el bloque recibir se ocupa de enviar el código del comando al FSM.

Los nombres aparentemente invertidos de enviar.php y recibir.php se justifican ya que el usuario no interactúa con ellos, sólo el FSM lo hace… ¡y desde el punto de vista de este, los nombres son correctos!

3.3 Parte Placa

EnsisFSM es el dispositivo señalador de tocados, es decir se ocupa de detectar cuando el arma de un tirador se ha puesto en contacto con el contrario y si este contacto es suficiente para considerarse un tocado en base al reglamento vigente de esgrima.

Si el tocado es válido, EnsisFSM se ocupa de notificarlo a los tiradores mediante unas luces y un sonido (Bloques de tocado y "Salida"). Este proceso, compuesto de varias tareas, estaría englobado en el bloque "Control del Asalto".

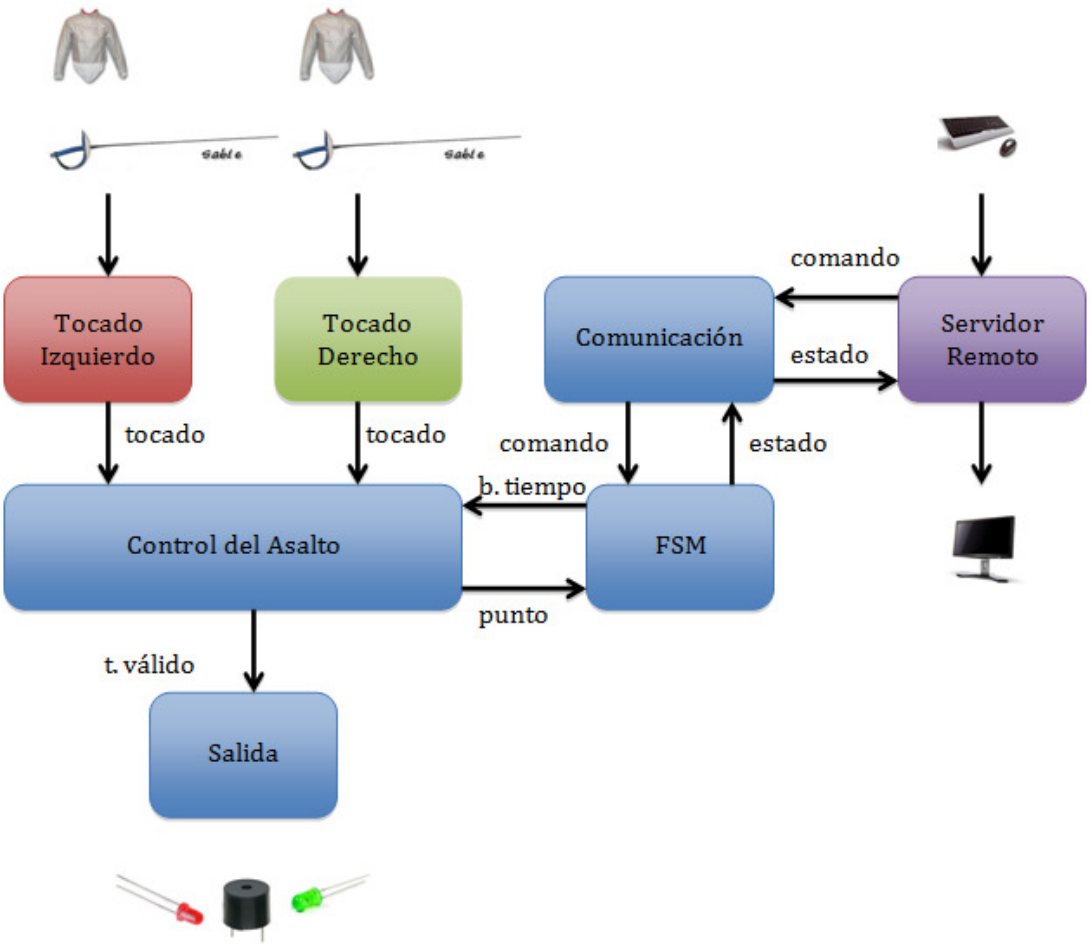

Figura 15. Diagrama de bloques

Adicionalmente, dos tareas (englobadas en el diagrama en el bloque “Comunicación”) se ocupan la recepción de comandos y envío del estado a EnsisServer.

El bloque “FSM” se ocupa de procesar los comandos y de almacenar las variables de estado que se enviarán al servidor remoto.

Por último, destacar que EnsisFSM utiliza FreeRTOS para la gestión de tareas y tiempos necesaria para realizar su cometido.

4. Descripción detallada

4.1 Breve explicación de TFC_UART_IO

Esta librería, contiene las funciones que he desarrollado durante la fase previa del proyecto con el objetivo de reutilizarlas en el FSM.

La librería contiene algunas funciones que se han utilizado en los programas de prueba durante la fase previa y que no son necesarias para el proyecto final.

Las partes de la librería relacionadas con este proyecto son:

- **ensisServer:** Contiene las funciones para recibir comandos desde el servidor remoto y la función para enviar el estado
- **GPIO:** Contiene las funciones para el manejo de los pines de la placa
- **logTerminal:** Contiene las funciones para la salida de debug por el UART3
- **UARTprintf:** Contiene las funciones para escribir y leer texto por UART
- **WiFly:** Contiene las funciones para el manejo de la WiFly

4.2 EnsisFSM

Como ya hemos visto en el capítulo anterior, EnsisFSM se compone de diversos módulos. Ahora entraré en el detalle del funcionamiento de cada uno de ellos.

4.2.1 Módulo comunicación

El módulo de comunicación se compone de dos tareas independientes.

La primera tarea se ocupa de la recepción de comandos desde la WiFly. Comparte el acceso a esta mediante un mutex (mutua exclusión) con la tarea de envío del estado y utiliza una cola para enviar los comandos procesados a la tarea que los procesará.

El diagrama de flujo que representa su comportamiento sería el siguiente:

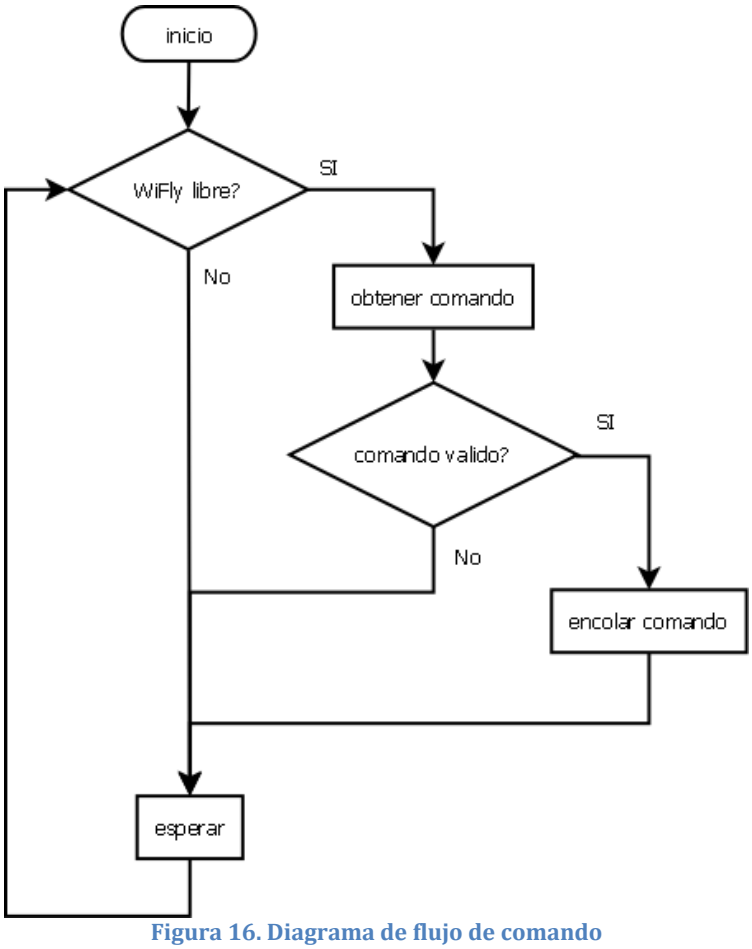

Figura 16. Diagrama de flujo de comando

Como podemos ver, una vez se ha finalizado la inicialización del sistema, se trata de un bucle infinito que alterna esperas con el intento de recepción de comandos.

El actor principal sería el propio EnsisFSM (ya que es un proceso automático), la precondición que la WiFly estuviese conectada y la postcondición que el comando viene encolado si es válido.

Los comandos que se reciben desde el ensisServer son un valor numérico uint8_t y se corresponden con la siguiente lista de comandos:

Código	Comando
101	SET WEAPON EPEE
103	SET WEAPON SABRE
201	START CLOCK
202	PAUSE CLOCK
301	SCORE LEFT +1
302	SCORE LEFT -1
303	SCORE RIGHT +1
304	SCORE RIGHT -1
401	FREE MODE
402	START NEW POULE
403	START NEW DIRECT ELIMINATION
0	NULL COMMAND / ERROR

La segunda tarea se ocupa del envío del estado del FSM.

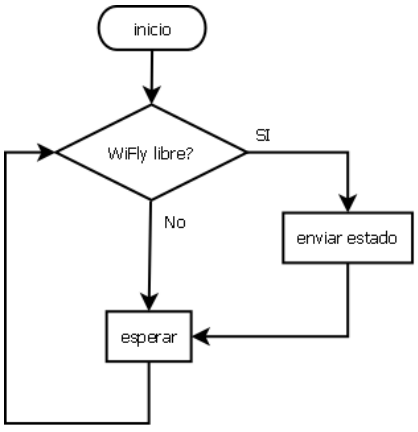

Figura 17. Diagrama de estado de Envio

Comparte el mutex con la otra tarea de comunicación y como podemos ver en su diagrama de flujo, consiste sencillamente en el envío de las variables de estado al ensisServer cuando la WiFly está libre.

Estas variables, que se almacenan siempre como uint8_t, son:

- El arma seleccionada actualmente
- La puntuación del tirador izquierdo
- La puntuación del tirador derecho
- Los minutos transcurridos de match
- Los segundos transcurridos de match
- El tirador con la prioridad
- El periodo de tiempo actual del match

El actor principal sería el propio EnsisFSM (ya que es un proceso automático), la precondición que la WiFly estuviese conectada y la postcondición que el estado viene actualizado en ensisServer.

Las armas se clasifican como:

Código	Comando
1	EPEE
3	SABRE

El periodo se clasificaría de la siguiente manera:

Código	Comando
0	FREE MODE
1	POULE
2	FIRST TIME
3	FIRST BREAK
4	SECOND TIME
5	SECOND BREAK
6	THIRD TIME
7	THIRD BREAK
8	EXTRA TIME

4.2.2 Módulo control del asalto

El módulo de control del asaltose implementa en una tarea que realiza dos funciones. Primero, es el encargado de inicializar el restode las tareasdel sistema. En segundo lugar,se ocupa de procesar los comandos que se han encolado desde el módulo de comunicaciones.

Cuando el FSM está funcionando en un modo con tiempo, es el encargado de activar y detener el reloj (timer por software), así como de rearmar el aparato.

El diagrama de flujo que representa su comportamiento sería el siguiente:

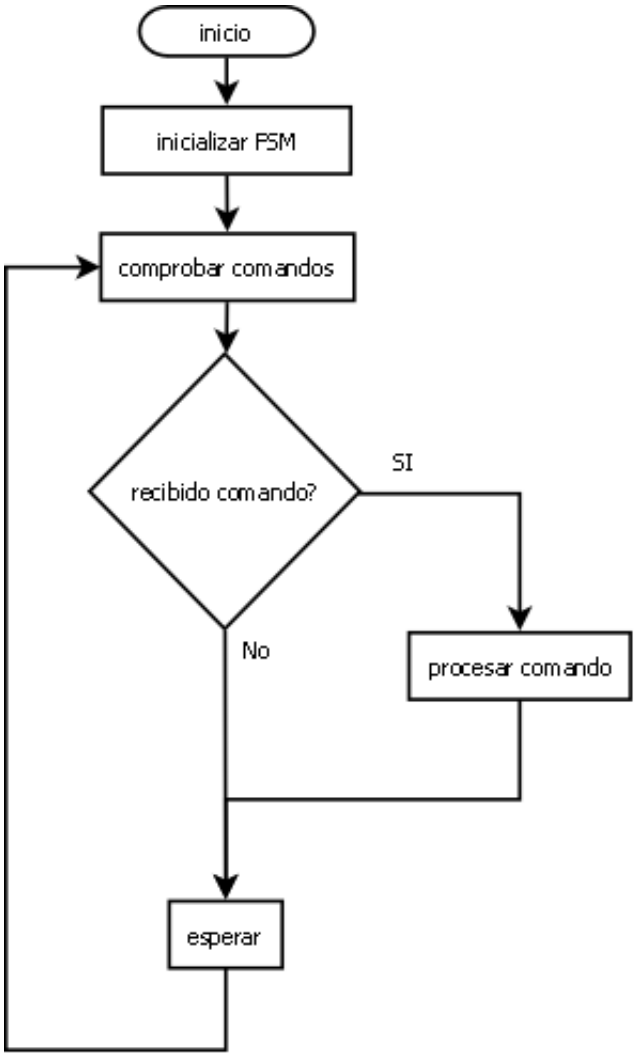

Figura 18. Diagrama de flujo Control del Asalto

4.2.3 Módulos de tocado

Los módulos de tocado son dos tareas (una para cada tirador) que se ocupan de controlar cuando se ha producido un tocado y de notificarlo mediante las señales luminosas y acústicas establecidas.

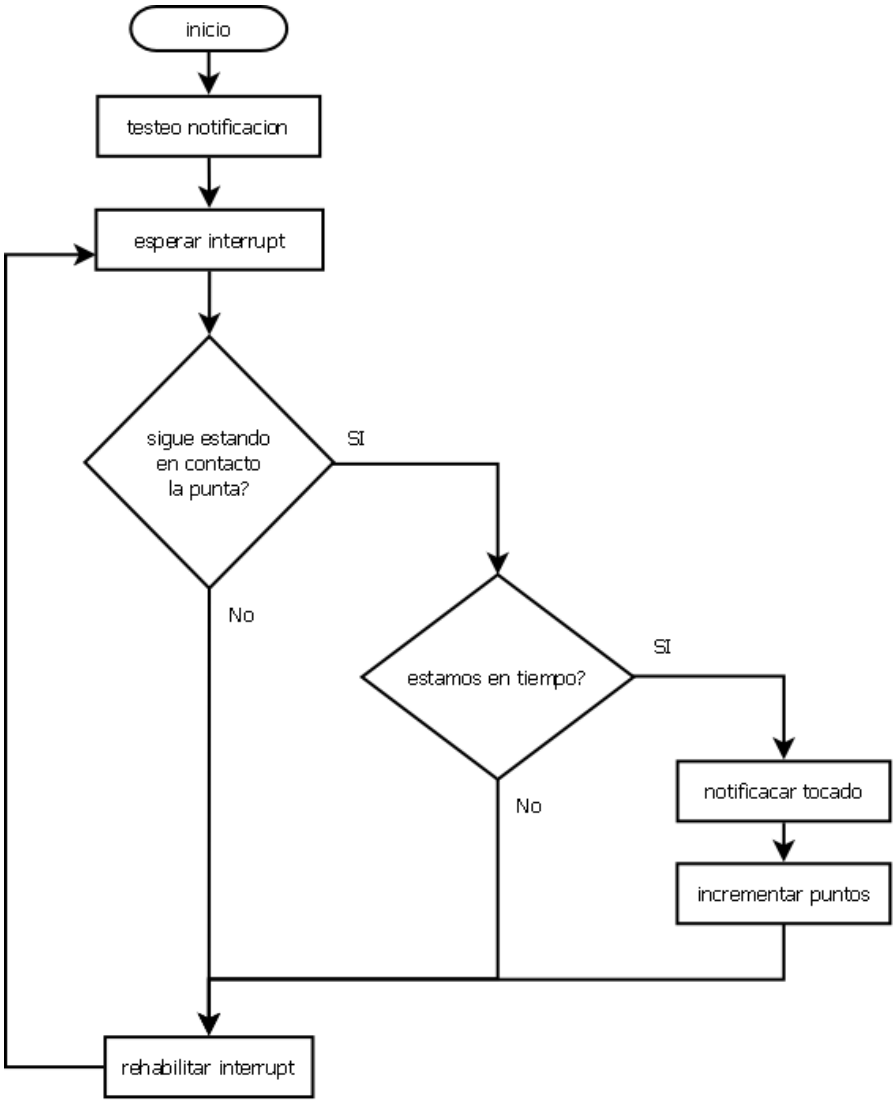

Figura 19. Diagrama de flujo Tocado

Como se puede observar en el diagrama de flujo, una vez se ha testeado el funcionamiento de la notificación, el módulo esperará a que se produzca una interrupción proveniente del pin de entrada del arma conectada.

Si una vez pasados los segundos establecidos del tiempo mínimo de contacto, la punta sigue estando presionada y no se ha producido otro tocado, o el tocado se está produciendo en tiempo reglamentario para el tocado doble, entonces la tarea notificará dicho tocado e incrementará los puntos del match.

Por último, el módulo rehabilitará las interrupciones en el caso de que sea apropiado.

El actor principal sería el tirador que lanza el tocado, la precondición que el aparato estuviese armado (interrupciones habilitadas) y que el tocado sea en tiempo reglamentario, y la postcondición que la puntuación del tirador viene incrementada y el tocado notificado.

4.3 EnsisServer

Como ya hemos visto en el capítulo anterior, EnsisServer se compone de cuatro páginas web. Ahora entraré en el detalle del funcionamiento de cada uno de ellos.

No hay que olvidar que EnsisServer es sólo un código de ejemplo y, por tanto, se ha implementado la funcionalidad mínima para demostrar el funcionamiento de los comandos remotos.

4.3.1 Enviar

La funcionalidad para la recepción de los datos de estado de EnsisFSM se encuentra codificada en la página enviar.php, que es llamada por el Módulo de comunicación de EnsisFSM cuando este envía sus variables.

Este pequeño script, almacena las variables ya mencionadas en archivos de texto que estarán disponibles para la página estado.php.

Como los archivos de texto se sobrescriben, sólo se podrá ver el último estado conocido del FSM.

El actor principal es EnsisFSM, no hay ninguna precondición y la postcondición es que se almacenarán las variables enviadas en archivos de texto.

Los archivos de texto que utiliza esta página para el intercambio son:

- weapon.var para el arma seleccionada
- score1.var para la puntuación del tirador izquierda
- score2.var para la puntuación del tirador derecha
- mins.var para los minutos que se mostrarán en el reloj del asalto
- secs.var para los segundos que se mostrarán en el reloj del asalto
- priority.var para la prioridad de los tiradores
- match_period.var para el modo de tiempo seleccionado

4.3.2 Recibir

La funcionalidad para el envío de comandos a EnsisFSM se encuentra codificada en la página recibir.php, que es llamada por el Módulo de comunicación de EnsisFSM cuando solicita un comando.

Este pequeño script, lee el último comando seleccionado de un archivo de texto (comando_pendiente.var) y sobrescribe su contenido con un 0 (comando nulo) para evitar que el mismo comando se ejecute más de una vez.

Como el archivo de texto se sobrescribe sólo se podrá recibir el último comando enviado desde el formulario de estado.php.

El actor principal es EnsisFSM, no hay ninguna precondición y la postcondición es que se sustituirá el código del comando enviado por el código del comando nulo.

4.3.3 Comando

Esta página recibe la opción seleccionada el formulario de estado.php y almacena el valor del comando apropiado en la variable de texto “comando_pendiente.var”, para que recibir.php pueda pasarlo a EnsisFSM cuando este lo solicite.

Como ya se ha dicho antes, el archivo de texto se sobrescribe sólo se podrá recibir el último comando enviado desde el formulario de estado.php.

El actor principal es el usuario que ha seleccionado el comando en el formulario, la precondición es que se haya seleccionado una opción válida y la postcondición es que se sustituirá el código del presente en el archivo de texto por el nuevo código de comando.

4.3.4 Estado

Esta página en la que se encuentra codificada tanto la visualización del estado como el formulario de envío de comandos.

Estado.php comparte los mismos archivos de texto que enviar.php para los datos que muestra por pantalla (véanse en el apartado 4.3.1)

Adicionalmente, el usuario tendrá disponible un formulario para seleccionar el comando a enviar a EnsisFSM

El actor principal es el usuario que ha seleccionado el comando en el formulario, la precondición es que se haya seleccionado una opción válida y la postcondición es que se sustituirá el código del presente en el archivo de texto por el nuevo código de comando.

5. Desarrollo de los conceptos esgrimísticos utilizados

En esta sección se recogen y explican los conceptos esgrimísticos necesarios para comprender la operación del FSM.

La esgrima es un deporte de oposición (combate) en la que dos participantes, llamados tiradores,deben intentar tocarse utilizando un arma deportiva.

Las armas (o modalidades) de la esgrima son 3: espada, florete y sable. Cada una cuenta con sus propias reglas y características.

Cuando un tirador toca al otro con su arma durante el tiempo mínimo de contacto establecido en el reglamento, se dice que ha “tocado” al otro. Si se trata de un match donde se contabilizan los puntos, el tirador que ha tocado incrementa su puntuación en 1.

Una vez un tirador ha sido tocado, el aparato señalador de tocados no debe registrar ningún tocado más, hasta que se produzca el rearme (automático o manual) de este.

Si dos golpes se dan simultáneamente, separados por el tiempo de tocado doble establecido en el reglamento del arma, el aparato debe señalar ambos tocados. Dependiendo del arma, ambos puntos serán contabilizados o sólo será contabilizado el del tirador que determine el árbitro.

Cuando al final de un match los dos tiradores se encuentran en igualdad de puntos, se sortea la “prioridad” entre ellos y se procede a tirar durante un minuto extra. Si al final de ese minuto de tiempo adicional los marcadores están aun empatados, gana el match el tirador que había ganado la prioridad en el sorteo.

El sorteo de la prioridad debe ser totalmente aleatorio, por ejemplo tirando una moneda o similar.

6. Viabilidad técnica

El producto final de este proyecto, podrá ser comercializado una vez se cumplan los requisitos mínimos de estabilidad y calidad necesarios. No obstante, mi idea sería liberar el código para que cualquiera pueda implementar el sistema en base a su propio criterio, quizás limitando la comercialización a kits ya ensamblados para quien no tenga la capacidad de montar su propio aparato señalizador.

Lamentablemente, el resultado del proyecto no deja de ser un prototipo. El tiempo que se ha dedicado no basta para obtener un producto final "pulido" que pueda ser directamente comercializado. Así, para poder comercializar este producto, el primer paso será implementar el resto de la funcionalidad que se encuentra en un FSM comercial y corregir los errores y carencias presentes.

Durante el transcurso del desarrollo y en las pruebas realizadas posteriormente, se han detectado dichos errores y carencias y habrá que resolverlos antes de obtener un producto final utilizable.

Una vez se hayan resuelto, estaremos en condiciones de obtener un producto 100% funcional, que se podrá enviar al SEMI (la comisión técnica de la Federación Internacional de Esgrima) en forma de prototipo para que sea aprobado (homologado). Este paso, si bien no es obligatorio, permitirá que nuestro producto pueda ser utilizado en competiciones oficiales de esgrima, lo que si bien no es imprescindible para su comercialización, si mejoraría mucho las posibilidades de adopción del aparato.

Como punto final, quedaría pendiente ultimar detalles como el diseño de una carcasa para proteger el aparato y la incorporación de toda la electrónica en una placa fabricada en serie.

Como conclusión, las ventajas de este sistema terminado frente a las alternativas presentes en el mercado serían su precio y su escalabilidad, mientras que su principal problema sería su inestabilidad actual y la falta de homologación por parte del SEMI.

7. Valoración económica

Una vez que hemos estudiado la viabilidad técnica, analicemos de forma orientativa los costes que tiene asociado el desarrollo del prototipo.

Por un lado tenemos los costes materiales (sin incluir el material ya adquirido, como el ordenador de desarrollo) y por otro lado tendremos en cuenta los costes de desarrollo, donde no incluiremos las jornadas de formación de la fase previa del proyecto.

Esto correspondería a nuestra inversión inicial y se desglosaría de la siguiente manera:

Concepto	Coste aproximado
Placas (LPC1769, WiFly)	35€
Material electrónico (LEDs, resistencias, breadboard…)	15€
Tiempo de desarrollo (22 jornadas)	2000€
Total aproximado	**2050€**

Una vez analizados los costes de desarrollo, analizaremos el costo del producto final:

Concepto	Coste aproximado
Placas (LPC1769, WiFly)	35€
Material electrónico (LEDs, resistencias…)	1€
Tiempo de montaje y configuración	50€
Total aproximado	**86€**

En este coste aproximado habría que incluir un retorno de la inversión inicial (que se debe recuperar), pero aun así podemos observar como los precios son muy inferiores a las alternativas comerciales que hemos analizado en capítulos anteriores.

Si el producto se comercializase, habría que tener en cuenta que los costes unitarios no se corresponderían a los que hemos contemplado hasta ahora, si no que se reducirían bastante.

No obstante, sería conveniente en el caso de proceder a una comercialización, realizar un nuevo estudio en busca de alternativas más económicas a los componentes utilizados en el proyecto (por ejemplo, posiblemente la LPC1768 disponga de todas las funcionalidades que necesitamos de la LPC1769 a un precio más reducido).

8. Conclusiones

8.1 Conclusiones

Si bien el software desarrollado cubre a grandes rasgos todos los objetivos propuestos, la versión final del prototipo presenta una serie de problemas que no he podido arreglar en el tiempo establecido para este proyecto.

La falta de tiempo en la última fase ha dejado el prototipo en un estado parcialmente inestable, tal como ya se ha comentado. Lo cual provoca situaciones erróneas cuando se utiliza para el arbitraje en un asalto real. Especialmente si lo comparamos con el funcionamiento total (aunque no óptimo) en el que se encontraba al final de la primera fase de desarrollo.

El método que he utilizado para el envío remoto de comandos, tiene unos tiempos de espera demasiado grandes (a veces de más de 2 segundos). Si se utiliza el prototipo para arbitrar un asalto real y se manda, por ejemplo, un comando de parada del reloj, pueden pasar varios segundos antes de que este se haga efectivo, cosa que no sería aceptable en un aparato apto para competición.

Si ahora volviese al principio del proyecto, con todo lo aprendido probablemente enfocaría muchas cosas de manera diferente.

8.2 Propuesta de mejora y líneas de futuro

Obviamente la primera propuesta de mejora es la corrección de los problemas existentes en el software que he podido depurar por falta de tiempo.

Además, la segunda prioridad, para mi, sería la optimización del código existente.

Igualmente, por los motivos que se han expuesto antes, sería conveniente cambiar el sistema de recepción de comandos de una conexión a un servidor HTTP a un socket TCP para comunicar con una aplicación de control, con el objetivo de reducir los tiempos necesarios para la ejecución de un comando. Sin descartar que la funcionalidad actual de envío de estado pudiese quedar como complementaria a esta nueva propuesta.

Además, faltaría por incorporar el florete a las armas compatibles con el sistema. El florete es el arma más compleja de las tres, ya que utiliza tres circuitos distintos para distinguir los tocados dados en blanco válido (chaquetilla eléctrica), blanco no valido (el resto del tirador) y en las partes aisladas del arma del contrario.

Una vez solucionados todos los problemas y realizadas todas las mejoras, sería conveniente solicitar la homologación al SEMI de la versión final del aparato señalizador.

Por último, mi idea es liberar el código en el futuro con una licencia GPL (o similar) para que sea realmente una plataforma abierta sobre la que se puedan desarrollar nuevas funcionalidades o basar nuevos aparatos.

8.3 Autoevaluación

Personalmente, he encontrado interesantísimo todo el TFC. Desde hace ya mucho tiempo me llamaba la atención el mundo de la programación de microcontroladores y hasta ahora no había tenido oportunidad de experimentarlo.

También ha sido la primera vez que he debido trabajar con un sistema operativo en tiempo real, en este caso el ya mencionado FreeRTOS, y me ha hecho cambiar totalmente el enfoque a la hora de diseñar una aplicación.

Me considero una persona perfeccionista, y por tanto no puedo estar satisfecho del resultado de este proyecto. He sido quizás muy optimista y esperaba que el proceso fuese mucho más simple, así que cuando me he encontrado con todos los problemas inherentes a la programación de sistemas embebidos he descubierto que mi proyecto era muy ambicioso para el tiempo del que disponía.

He descubierto que cuanto más tiempo le dedicaba al proyecto, más tiempo necesitaba dedicarle… una aparente contradicción. Y aunque el tiempo que le he dedicado ha sido grato, debo reconocer que también ha sido bastante frustrante a veces.

Quiero recalcar también que, al margen de los resultados, considero que este proyecto es viable y útil si se le dedica el tiempo y el trabajo necesarios. Mi intención es seguir trabajándolo como proyecto personal al margen de la finalización del TFC.

Por último, quiero comentar también que, independientemente del producto final obtenido, he aprendido muchísimo durante la realización del proyecto y estoy deseando seguir adelante descubriendo que más me puede ofrecer el mundo de los sistemas empotrados.

Resumiendo, gracias a este proyecto me he introducido en el mundo de la programación de microcontroladores, he comprendido sus limitaciones y he comenzado a vislumbrar su potencial. Además, he tenido un primer contacto con los sistemas operativos en tiempo real y el nuevo enfoque que eso supone.

9. Glosario

Asalto: Un match amistoso entre dos tiradores de esgrima

FIE: Federación Internacional de Esgrima. El máximo órgano regulador de este deporte.

FreeRTOS: Sistema operativo en tiempo real para sistemas empotrados. Soporta gran cantidad de microcontroladores.

FSM: Fencing Score Machine. Aparato señalizador de tocados de esgrima. Análogo a otros términos como ScoreBox o marcador electrónico.

Match: Un asalto de esgrima en el que se tiene en cuenta el resultado (por ejemplo en competición)

Placa: Del inglés “board”. Un sistema empotrado compuesto por un microcontrolador y sus periféricos.

Prioridad: En caso de empate al final de un match, indica el tirador que se considerará vencedor.

SEMI: Comisión técnica de la FIE. Entre otras cosas se ocupa de la homologación de los aparatos señalizadores de tocados.

10. Bibliografía

- Using the FreeRTOS real time kernel – *Richard Barry*
- Reglamento de material de la FIE
- Material Nivel III – Escuela de Entrenadores de Esgrima (RFEE)
- Manuales de referencia de la LPC1769 y de la WiFly RN-XV
- UOC (foro y consultor de la asignatura):http://www.uoc.edu
- Wiki de la asignatura: http://cv.uoc.edu/app/mediawiki14/wiki/P%C3%A0gina_principal
- Página de la Federación Internacional de Esgrima (FIE):http://www.fie.ch
- NXP Knowledgebase:http://knowledgebase.nxp.com
- Sitio web de Eric Schlaepfer: http://www.sonic.net/~schlae/gplscoremach.html
- GNU-GPL-3WT: http://ca.groups.yahoo.com/group/GNU-GPL-3WT/
- Fencing-arduino: http://code.google.com/p/fencing-arduino/

11.Anexos

11.1 Manual del usuario

11.1.1 EnsisFSM

EnsisFSM es un sistema autónomo. Para encenderlo, basta con alimentar la placa y esta comenzará el proceso de inicialización. Para apagarlo, retire la alimentación. Mientras la placa está encendida, se encenderá un LED amarillo de funcionamiento.

EnsisFSM por defecto se inicia en modo espada. Para cambiar el arma utilice EnsisServer.

11.1.2 EnsisServer

EnsisServer se accede a través de un servidor web, introduciendo la URL en un navegador web desde un dispositivo con conectividad (ej.: se ha establecido un servidor de prueba para el proyecto en http://encastatstfc.comli.com/estado.php). Una vez cargada la página, veremos el actual estado de EnsisFSM.

Clicar la opción deseada o seleccionar la opción a modificar. Nótese que sólo se puede enviar un comando cada vez. Si se envían dos comandos seguidos (o antes de que EnsisFSM pueda procesar el primero), sólo se procesará el último enviado.

11.2 Ejecución y compilación

11.2.1 EnsisFSM

La ejecución del firmware del aparato señalizador de tocados viene realizada automáticamente cuando se enciende la placa.

La compilación y carga del firmware en la placa se realiza directamente desde el IDE.

Antes de compilar el código del FSM, será necesario compilar el de FreeRTOS y el de la librería TFC_UART_IO. Para ello, simplemente seleccionar el proyecto y hacer clic en el botón “Build” del menú.

Una vez compilados ambos proyectos, asegurarse que la placa está conectada por el puerto USB, seleccionar el proyecto de EnsisFSM y hacer clic en el botón “Debug”para cargar el firmware en la placa. Una vez se haya finalizado el proceso el firmware se encontrará en la memoria flash de la placa.

11.2.2 EnsisServer

Al tratarse de una página web, será necesario utilizar un navegador web para dirigirse a la dirección donde se pueda acceder la página “estado.php” (nótese que debe ser la misma dirección especificada en la librería TFC_UART_IO).

Por ejemplo: http://encastatstfc.comli.com/estado.php

Al ser un código en PHP, no será necesario compilarlo, pero será necesario disponer de un servidor web (por ejemplo Apache) con el intérprete de PHP.

11.3 Diagramas de montaje

En este anexo incluyo los diagramas que explican como se realiza el montaje físico (cableado) de las placas, las entradas y las salidas. Para facilitar la claridad, los diagramas se han separado por categorías.

Aquí se puede observar el prototipo completamente montado.

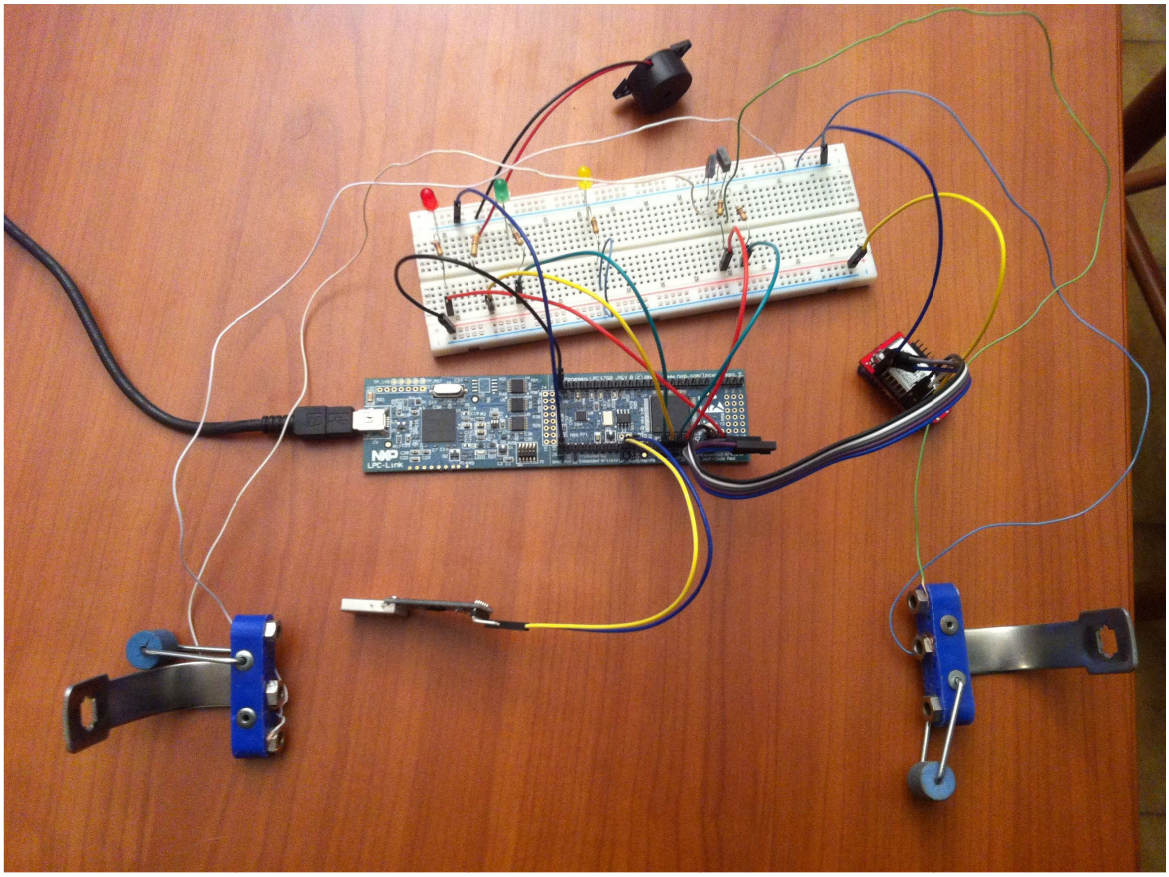

Figura 20. Prototipo completo montado

11.3.1 Comunicación entre placas

Este diagrama muestra como se conecta la LPC1769 a la WiFly

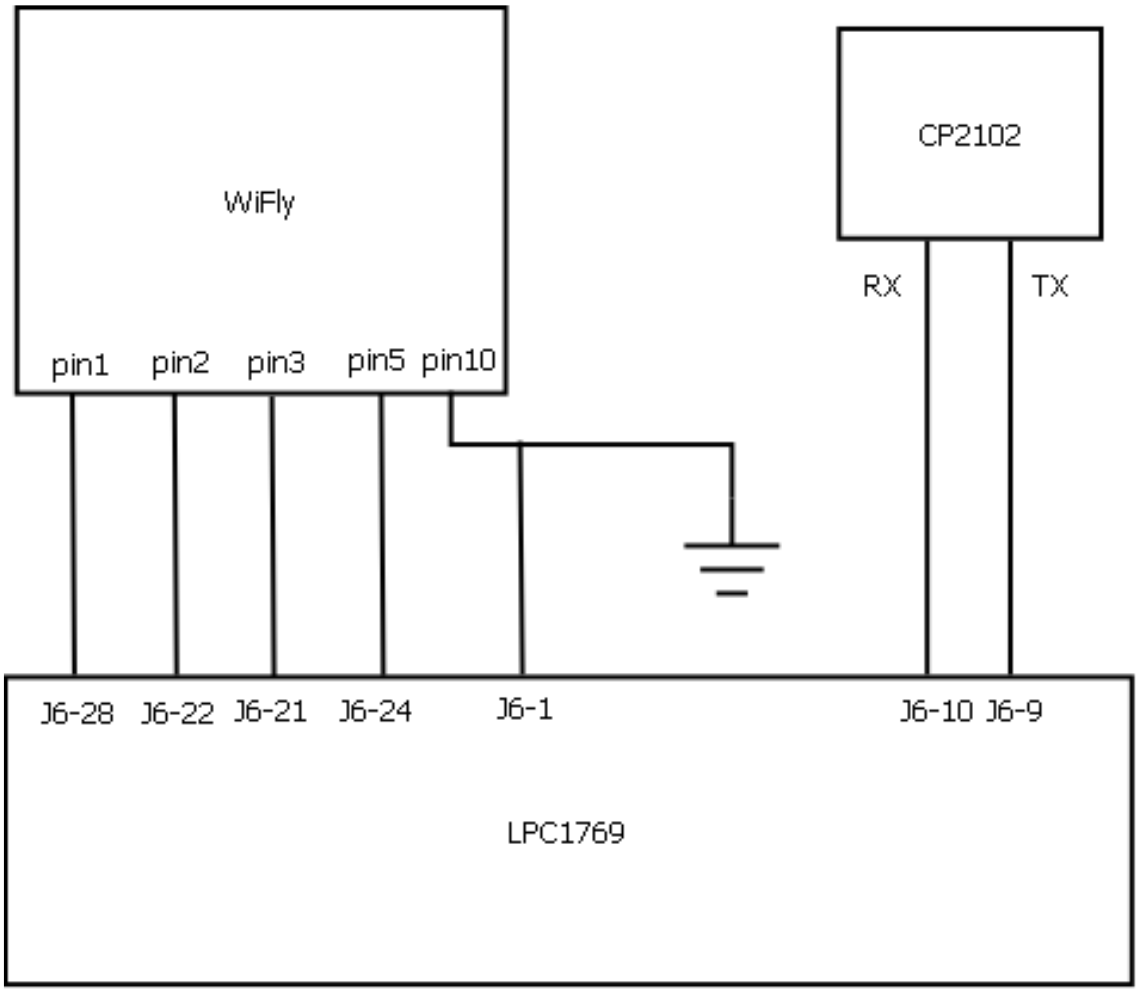

Figura 21. Com. entre placas

11.3.2 Outputs

Este diagrama muestra la conexión de los LEDs y el buzzer a la placa

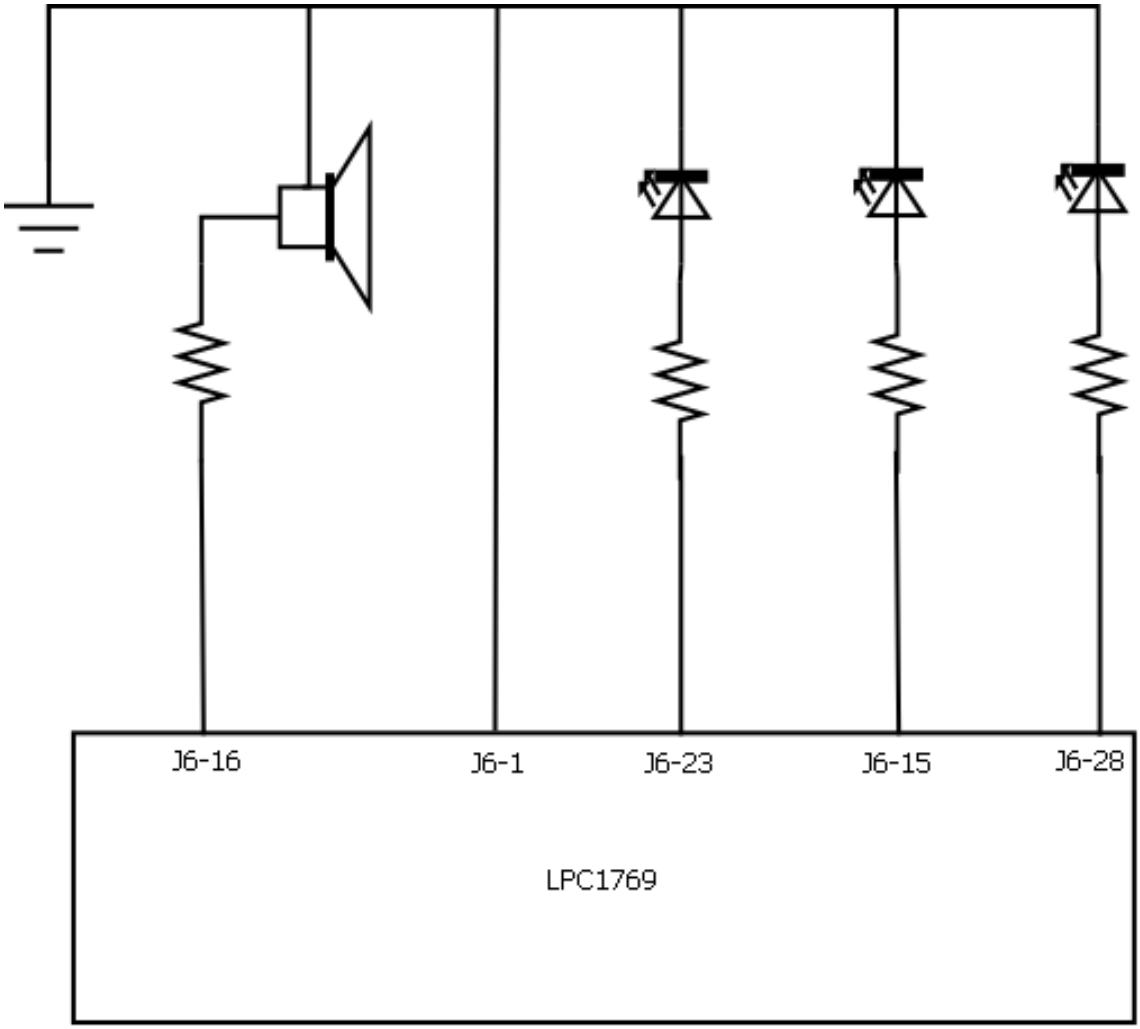

Figura 22. Pinout

11.3.3 Inputs

Este diagrama muestra como se conectan las entradas a la placa

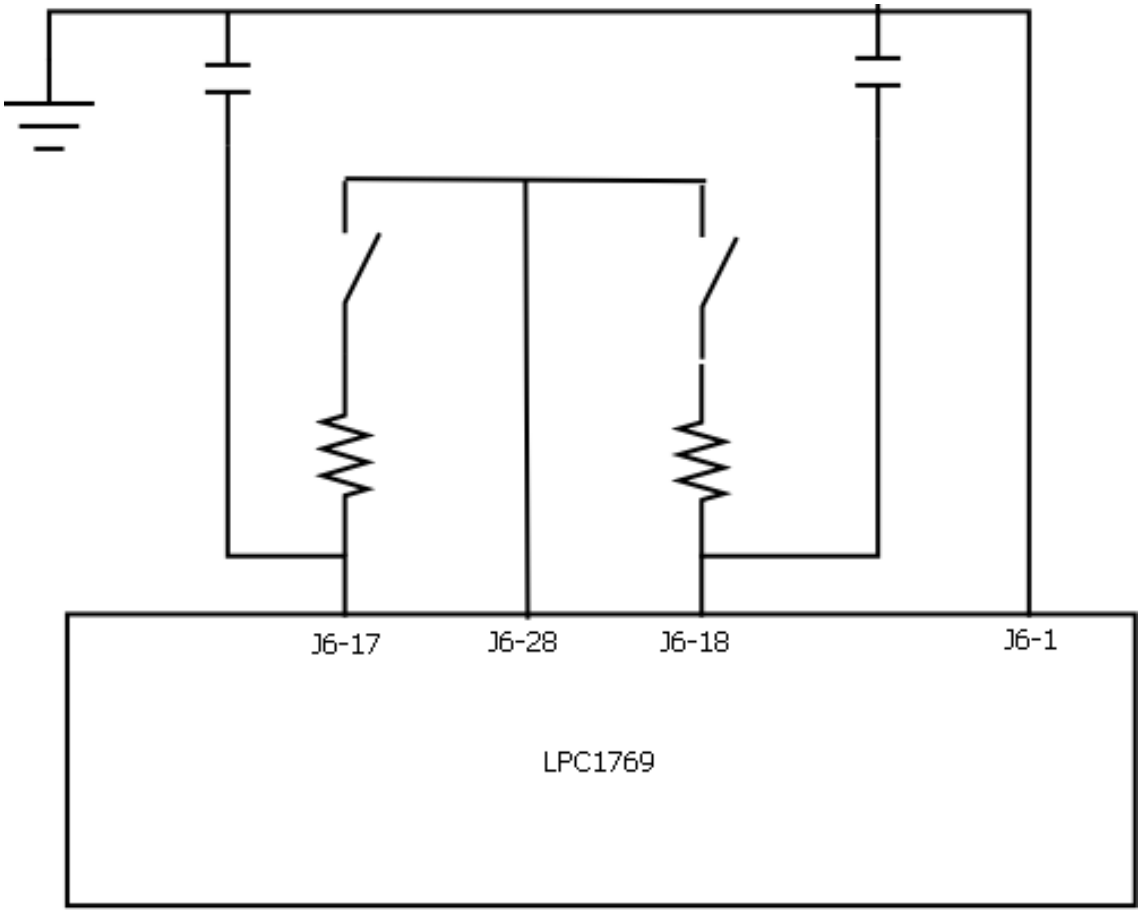

Figura 23. Pinin

11.4 Diagramas de estado

En este anexo se presentan los diagramas de estado de EnsisFSM correspondientes a los bloques definidos en el capítulo 4

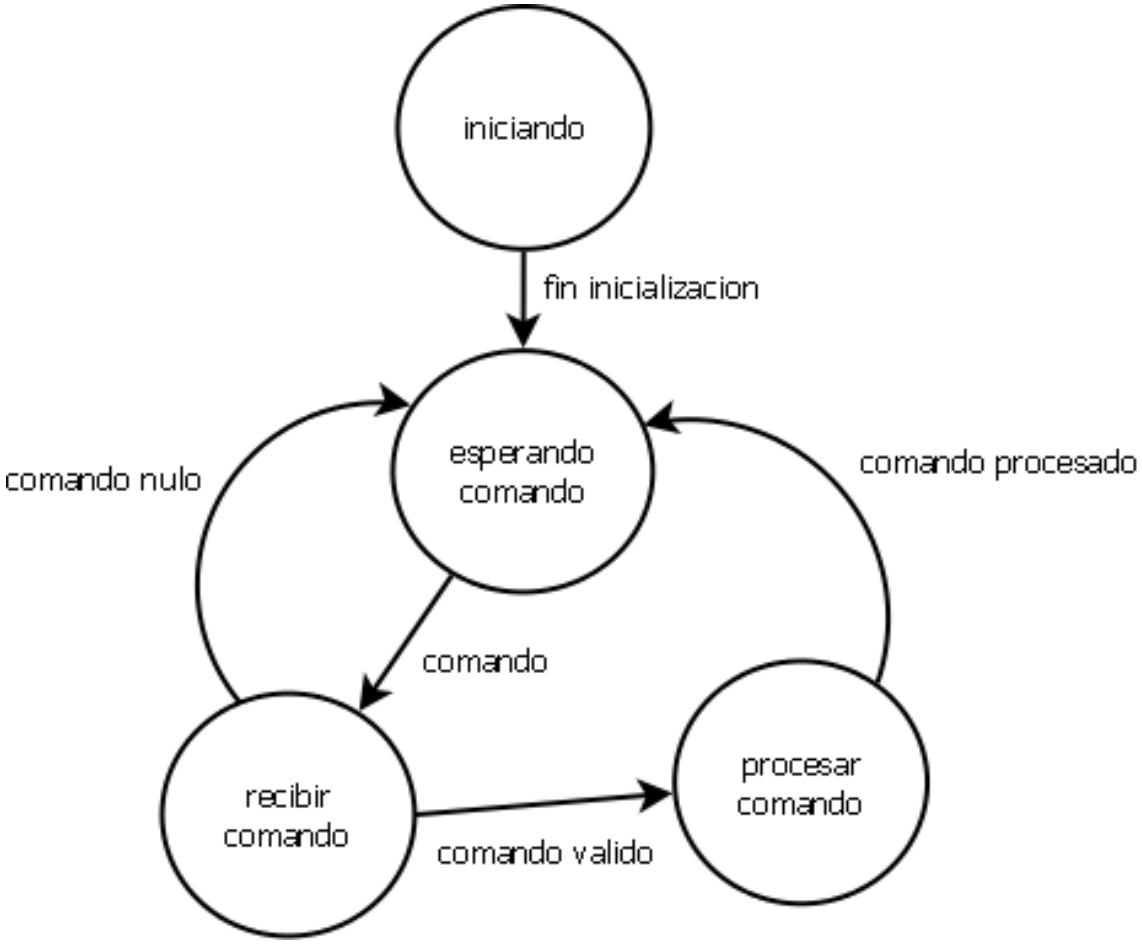

Figura 24. Diagrama de estados Control Asalto

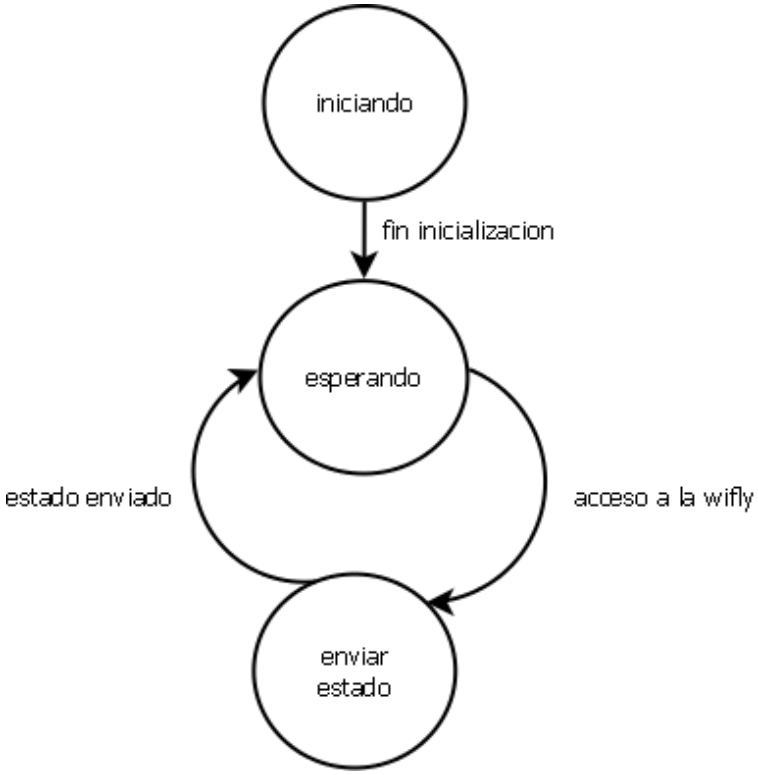

Figura 25. Diagrama de Estados Enviar Estado

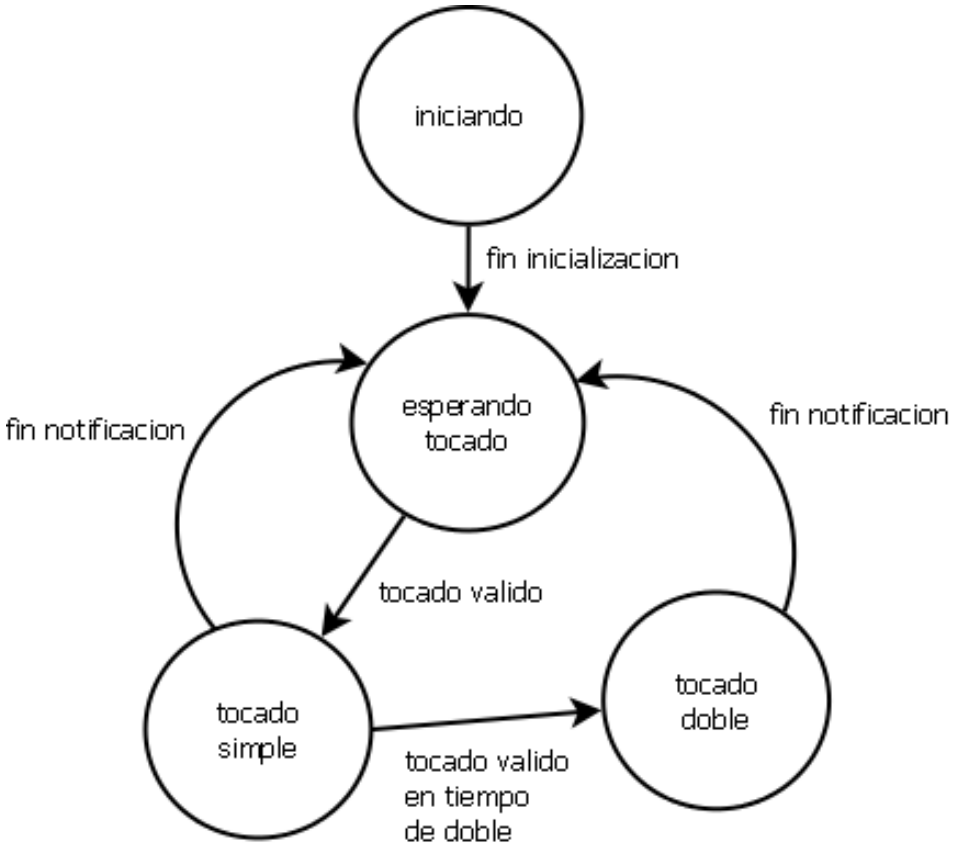

Figura 26. Diagrama de Estados Tocado