# SOC_0212_T2_210212

Definición del microcontrolador y familiarización con el entorno de desarrollo

## Versión: 20120309

Autor: Rafel Mormeneo Melich

| | |
|---|---|
| Inicio | 22/02/2012 |
| Final | 09/03/2012 |
| Personas | Fran Mañez<br>Rafel Mormeneo Melich<br>Víctor Sánchez |

Predictive Control Systems

Thinking Forward

www.tfxxi.com

## VARIACIONES RESPECTO LA VERSIÓN ANTERIOR:

| Versión anterior | Ninguna |
|---|---|
| Variación | 1- Ninguna |

1- Autor: Vacio. Descripción: Vacio.

# 1 Definición del Microcontrolador y familiarización con el entorno

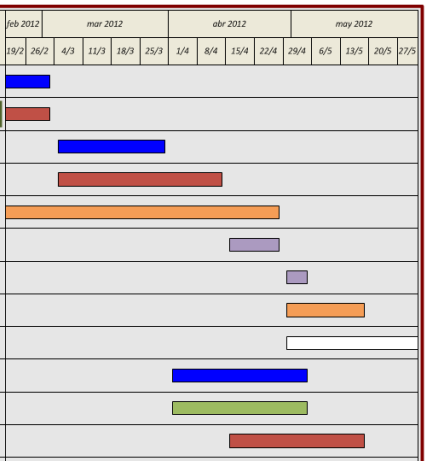
# 1. Objetivos

El objetico de esta tarea consiste en definir el microcontrolador que se va a usar en el proyecto y definición de unas pautas a seguir para desarrollar software con este microcontrolador.

# 2 Plan de desarrollo de la tarea

Dentro del proyecto SOC, la planificación inicial de esta tarea ocupa un total de 2 semanas que van desde 21/02/2012 hasta 02/03/2012.

| ID | Tarea | Descripción tarea | Responsable tarea | Inicio | Final | Duración |
|---|---|---|---|---|---|---|
| 1 | SOC_1202_T1_120221 | Definir hardware cámara | Rafel | 21/02/2012 | 02/03/2012 | 9d |
| 2 | SOC_1202_T2_120221 | Definir el microcontrolador, familirizarse con el entorno de desarroyo, can, etc. | Fran | 21/02/2012 | 02/03/2012 | 9d |
| 3 | SOC_1202_T3_120221 | Diseñar prototipo | Rafel | 05/03/2012 | 30/03/2012 | 20d |
| 4 | SOC_1202_T4_120221 | Diseñar software prototipo | Fran/Ezio | 05/03/2012 | 13/04/2012 | 30d |
| 5 | SOC_1202_T5_120221 | Pruebas de algoritmos básicos | Víctor | 21/02/2012 | 27/04/2012 | 49d |
| 6 | SOC_1202_T6_120221 | Definición protocolo de comunicación Servidor/Cliente/SOC | Ezio/Victor/Fran | 16/04/2012 | 27/04/2012 | 10d |
| 7 | SOC_1202_T7_120221 | Implementación del protocolo de comunicación Servidor/Cliente/SOC | Ezio/Fran | 30/04/2012 | 04/05/2012 | 5d |
| 8 | SOC_1202_T8_120221 | Cambios en TFClient derivados de la aplicación de SOC | Victor/Fran | 30/04/2012 | 18/05/2012 | 15d |
| 9 | SOC_1202_T9_120221 | Diseño de algoritmos alternativos mejorados de visión. | Todos | 30/04/2012 | 01/06/2012 | 25d |
| 10 | SOC_1202_T10_120221 | Rediseño de hardware SOC | Rafel/Ezio | 02/04/2012 | 04/05/2012 | 25d |
| 11 | SOC_1202_T11_120221 | Diseño industrial SOC | Marc/Rafel/Oriol | 02/04/2012 | 04/05/2012 | 25d |
| 12 | SOC_1202_T12_120221 | Diseño del software de visión final | Fran/Ezio/Victor | 16/04/2012 | 18/05/2012 | 25d |




La tarea se ha desviado una semana respecto la estimación inicial debido a varios motivos:

- La tarea se inició con un día de retraso ya que la persona asignada, Fran, tenía que terminar otras cosas.
- El responsable de la tarea, no estaba asignado 100% del tiempo a ella.
- El programador (DRAGON) no ha llegado hasta mitades de la segunda semana (28/02/2012).
- Han aparecido problemas de compatibilidad entre el entorno de desarrollo y la placa de desarrollo (hardware) de que disponemos. En el directorio ***How to*** de la tarea se explica detalladamente como instalar los controladores del programador DRAGON.

Además inicialmente en la tarea esta previsto que sólo trabajara Fran pero en la tercera semana se ha tenido que añadir recursos para cumplir con la planificación del proyecto.

# 3 Desarrollo de la tarea

La parte principal del proyecto SOC consiste en la captura y procesado de imágenes de las barras de comprobación. Esta tarea requiere una gran capacidad de cálculo si se quiere realizar de forma precisa, robusta y con una velocidad de procesamiento aceptable.

Se ha desestimado la realización de este proyecto con un microcontrolador DSPIC de Microchip. Aunque se dispone de muchas librerías de software para este dispositivo, la experiencia nos demuestra que es incapaz de realizar tareas de cálculo complejas.

Disponemos de la placa de desarrollo AT32UC3C-EK. Esta placa va equipada con el procesador de Atmel AT32UC3C0512C. Es por ello que vamos a realizar unos pequeños test para ver si este microcontrolador es adecuado para el procesamiento de imágenes.

## 3.1 Subtareas

Para el desarrollo de esta tarea se han identificado 5 subtareas:

- Instalación del entorno de desarrollo
- Creación de un proyecto de ejemplo
- Instalación de los drivers del programador
- Pruebas de capacidad de cálculo del microcontrolador
- Creación de un programa template con las funcionalidades básicas.

El detalle del desarrollo de cada una de las subtareas se lleva a cabo en los siguientes apartados.

### 3.1.1 Instalación del entorno de desarrollo

#### *AVR Studio 5*

Investigar entorno de desarrollo a utilizar. Se empezó con AVR Studio 5, IDE con simulador, emulador y debug de código basado en Visual Studio 2010. Necesita otro programa externo AVRDude para quemar el código hex en la MCU. Este programa funciona a través de una GUI (Free ISP) que se ha de usar con programadores como USBtinyISP. Es un entorno de desarrollo muy pesado y sin soporte nativo para la MCU AT32UC3C-EK. A parte es difícil de configurar y trabajar ya que necesita muchos plugins (los programas comentados anteriormente).

Se estudio cómo funciona el entorno y se siguió un tutorial para hacer un ejemplo de un programa que hacía un “blinding LEDs“. El código se seguía correctamente (aunque era de un micro y una placa diferente de las nuestras). A la hora de seleccionar nuestra placa no había soporte completo por parte del IDE. Se tendría que realizar una configuración muy pesada importando a mano muchas librerías para poder trabajar con nuestra placa y MCU.

Se continua trabajando con AVR Studio 5 y el ejemplo de “blinding LEDs” para nuestra placa, sin mucho éxito. Se revisa documentación para nuestra placa y alternativas en cuanto a entornos de desarrollo que se integren perfectamente con nuestra placa (AVR32 Studio es el recomendado para trabajar con proyectos AVR32)

Se necesita herramienta de para programar el MCU. Se ha optado por comprar el AVR Dragon.

#### *AVR32 Studio*

AVR32 Studio es un IDE basado en eclipse, mucho más ligero que el anterior y con un soporte completo para nuestra placa de AVR32. Se instala el entorno de desarrollo y a parte el avr32-gnu-toolchain, que es un paquete de herramientas con utilidades para la creación de programas bajo microcontroladores AVR32 . Posee el compilador, el assembler, el linker, el debugger y las librerías C para el desarrollo de programas C/C++ para AVR32. El entorno es mucho más “amigable” al tratarse de un eclipse modificado y mucho más fácil de integrar.

Se estudia cómo funciona el IDE siguiendo tutoriales para crear nuevos proyectos. Importante, a diferencia del anterior, es que tiene un framework donde puedes seleccionar los componentes,

drivers y servicios que quieres usar en tu aplicación y te genera automáticamente en tu proyecto todas las clases necesarias para trabajar con los componentes seleccionados (Ej: USART, GPIO, CANIF, ADCIFA, etc.). Se generan ejemplos simples y compilan correctamente.

Se abre en el IDE el programa de ejemplo que viene cargado en la placa de desarrollo y se estudia el código para ver como se trabaja. Se investiga el código para ver como se inicializan los puertos utilizados, que es cada pata de la MCU, las interfaces de la placa, etc. Como ejemplo es muy completo pero para partir de ahí para desarrollar un proyecto nuevo es algo complicado. El siguiente paso es estudiar cómo hacer un proyecto nuevo desde 0 y tener como base para hacer las cosas este proyecto de ejemplo.

Se intenta conectar la placa al PC e intentar enviar datos pero no se consigue. Este paso se queda pendiente hasta que se compre el programador AVR Dragon. Se pedirá el lunes y el martes o miércoles debería llegar.

**Subtarea Completada**: 2 días

### 3.1.2 Creación de un proyecto de ejemplo

Se intenta crear proyecto nuevo desde cero con AVR32 Studio. Debido a los problemas de linkaje de librerías y configuración de proyecto se ha optado por crear un proyecto base a partir de un proyecto ejemplo que proporciona el propio IDE.

Se ha creado un proyecto base a partir de varios proyectos ejemplo. Se ha estudiado el código fuente de cómo funcionan ejemplos varios y finalmente se ha partido de uno de estos proyectos para crear nuestro proyecto base. Se ha creado la configuración necesaria del proyecto para acceder al USART y se ha hecho un print por el puerto UART. Se ha creado la configuración necesaria para inicializar las interrupciones. Se ha creado la configuración necesaria para inicializar el display y un pequeño ejemplo que pinta la pantalla de color rojo y dibuja una línea azul.

En cuanto al IDE, funciona perfectamente para el desarrollo de este tipo de aplicaciones. Se pueden añadir fácilmente al proyecto drivers, componentes y servicios, a partir de la opción framework del eclipse. Con esto se añade todo lo el código necesario (drivers c) para poder trabajar con los componentes seleccionados. A partir de aquí ya se pueden incluir las cabeceras .h de las clases generadas en nuestros archivos para poder utilizar los componentes. De la misma manera que se pueden añadir, se pueden eliminar. El código generado no es nada intrusivo para nuestras aplicaciones.

Se continúa a la espera del AVR Dragon para poder hacer pruebas del código de prueba implementado.

**Subtarea Completada**: 1 día.

### 3.1.3 Instalación de los drivers del programador

Llega el programador AVR Dragon y se empiezan a hacer las primeras pruebas de integración con el IDE y el Dragon para programar la placa.

Se tienen bastantes problemas ya que a la hora de programar, el entorno de desarrollo nos da un conflicto entre su versión de MCU y la nuestra. (Nuestra MCU es la AT32UC3C0512C, y en el IDE

estas MCU aparecen como AT32UC3C0512CRevC). El problema es que en realidad son el mismo tipo de device pero el IDE no deja programar ya que detecta que el nombre no es el mismo.

Se pierde mucho tiempo buscando como solucionar el problema y finalmente se encuentra dentro de la página de Atmel una solución al problema que reside en cambiar un fichero XML. Pero este fichero solo se encuentra en la versión AVR32 Studio 2.6 y nosotros trabajábamos con la 2.4.

Por la tanto se procedió a cambiar el IDE a la versión 2.6, y se actualizó la AVR toolchain a la versión 3.2.3.

El cambio a realizar dentro del nuevo entorno es el siguiente:

En la ruta:

C:\Program Files (x86)\Atmel\AVR Tools\AVR32 Studio\plugins\com.atmel.avr.utilities.win32.x86_3.0.0.201009140848\os\win32\x86\share\avr32

Modificar el fichero ***uc3c0512c.xml***. Donde pone:

***<part_name>UC3C0512C</part_name>***

Se tiene que sustituir por:

***<part_name>UC3C0512CREVC</part_name>***

Una vez funcionando el entorno correctamente y hecha la primera programación, se empieza la primera fase de pruebas en "entorno real". Se realiza una pequeña prueba de escritura en el display de la placa.

#### *Instalar los drivers de la placa*

Con la placa conectada, ir a "Administrador de dispositivos". Aquí aparecerá el dispositivo USB con el símbolo de exclamación indicando que no tiene el driver. Instalar el driver e indicar que busque el driver en la siguiente ruta (debes tener instalado el ARV32 Studio):

C:\Program Files (x86)\Atmel\AVR Tools\AVR32 Studio\plugins\com.atmel.avr32.sf.uc3_1.7.0.201009140900\framework\

**Subtarea Completada**: 2 días.

### 3.1.4 Pruebas de capacidad de cálculo del microcontrolador

A partir de este momento Víctor Sánchez y Rafel Mormeneo entran a formar parte de la tarea.

En este punto se va a realizar un pequeño programa de prueba para decidir si el microcontrolador es adecuado para el proyecto.

La tarea más pesada consistirá en aplicar filtros sobre una imagen. Aunque la imagen tiene una resolución de 640x480 se definirán unas regiones de interés mucho más pequeñas. La prueba se realizará con una matriz de tamaño 165x24. Este es el tamaño que, en principio tendrán las regiones de interés.

El programa de test realizará un filtro de media sobre la matriz. El kernel del filtro se ha definido de 3x3 ya que estos serán el tipo de filtro que se va a utilizar en el procesado de las imágenes de comprobación. El filtro se va a pasar 16 veces por la imagen para representar que se pasa 4 veces por cada una de las 4 imágenes.

Con este código se ha comprobado que el microcontrolador puede hacer esta tarea aproximadamente con un segundo. Por lo tanto podemos procesar 1 imagen de comprobación por segundo, lo cual es más que suficiente.

Nota: En este punto aún no se ha descubierto como configurar el microcontrolador para que trabaje a la máxima velocidad. Para realizar esta prueba se ha configurado para que trabaje con el oscilador externo a 16MHz. Se tiene que investigar como utilizar el PLL para que trabaje a 66 MHz.

**Subtarea Completada**: 1 días.

### 3.1.5 Creación de un programa template con las funcionalidades básicas.

#### *Pruebas con los distintos componentes*

Se hacen pruebas de USART, ADC, DISPLAY. Se accede al potenciómetro y al micro de la placa mediante UART y se muestran por el display los valores obtenidos.

#### *Recepción de imágenes por UART y visualización en display LCD*

Se realiza un programa que recibe imágenes por UART y las muestra en el display.

Para ello se crean las estructuras básicas de imagen que se utilizaran en el futuro. Todo ello se hace de forma que el código sea reutilizable. Entre ellas se definen las estructuras siguientes:

Hemos tenido muchos problemas para la recepción por UART y hemos descubierto lo siguiente:

- Cuando salta la interrupción del UART se debe enviar por UART para reactivarla. Hay que investigar si se puede hacer de otra forma.
- Cuando se realiza un programa que envía datos de un PC al microcontrolador los datos se deben enviar byte a byte e introducir un retraso entre ellos para evitar que se pierdan.

En esta tarea se ha desarrollado código para trabajar con las interrupciones del UART y del Timer del microcontrolador.

#### *Aplicación Cliente-Servidor SOC*

Se ha desarrollado un software con Visual Studio que captura imágenes de una webcam y las envia al microcontrolador Atmel por UART. Esta aplicación se ha desarrollado con arquitectura cliente servidor para que nos sea útil para desarrollar software para el proyecto. Con esta herramiente se puede instalar la cámara en el motor del taller y procesar las imágenes en los sitios de trabajo. De esta forma podemos desarrollar software de procesado de imagen con imágenes reales.

**Subtarea Completada**: 7 días.

## 4 Conclusiones de la tarea

Para desarrollar software para AVR32 es recomendable utilizar AVR32 Studio.

El microcontrolador AT32UC3C0512 se puede utilizar para el procesado de imágenes de comprobación a una velocidad de 1 imagen por segundo aproximadamente.

Se va a utilizar el procesador ***AT32UC3C2512***. Este procesador tiene las mismas funcionalidades y prestaciones que el procesador analizado con la diferencia que en lugar de 144 pines tiene 64.

Se tiene que investigar como recibir datos por UART sin tener que enviar para reactivar la interrupción.

Se ha creado el image toolbox que inicialmente contiene los ficheros image.h e image.c que contienen las definiciones y funciones básicas para trabajar con imágenes.

Se ha desarrollado un cliente-servidor que captura imágenes en un PC y las envía al Atmel que puede estar conectado en otro PC de la red.

Tarea completada: 13 días