SOC_1202_T4_120221

Diseñar software prototipo

Versión:20120710

Autor: Víctor

Inicio	14/03/2012
Final	10/07/2012
Personas	Víctor Sánchez Rafel Mormeneo

Predictive
Control
Systems

Thinking Forward
www.tfxxi.com

VARIACIONES RESPECTO LA VERSIÓN ANTERIOR:

Versión anterior	Ninguna
Variación	1- Ninguna

1- Autor: Vacio. Descripción: Vacio.

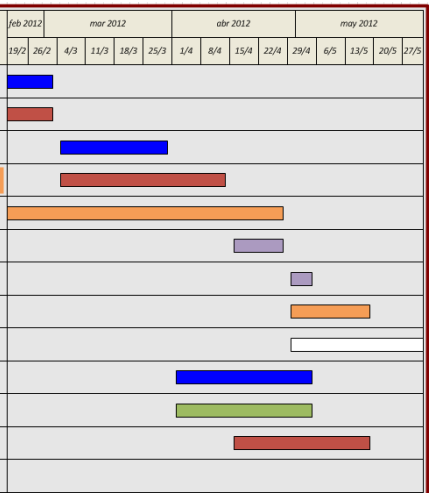
1 Diseñar software prototipo

2 Objetivos

El objetivo principal de la tarea consiste en desarrollar el código de visión para el microcontrolador Atmel escogido para el desarrollo. Este código deberá basarse en el software ya desarrollado en C++ durante la tarea T5. Deberemos implementar una librería con funciones que permitan, a partir de una imagen, calcular la holgura de comprobación en el micro Atmel. Además del desarrollo de la librería deberemos finalmente, con los algoritmos reales, calcular la velocidad máxima que podremos alcanzar en el cálculo de la holgura de comprobación.

3 Plan de desarrollo de la tarea

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
13						



Dentro del proyecto SOC, esta tarea (Marcada en marrón en el diagrama original) ocupa un total de 5 semanas y media que van desde 05/03/2012 hasta 13/04/2012. La razón por la que el inicio de la tarea haya sido desplazado hasta el día 14/03/2012 ha sido básicamente que Fran, quien era responsable de esta tarea, dejó la empresa antes de poder empezarla. Por esta razón he tenido que acabar primero la tarea T5 para poder hacerme responsable de la tarea T4.

4 Desarrollo de la tarea

4.1 Subtareas

Para el desarrollo de esta tarea se han identificado 9 subtareas:

- Instalación y configuración del entorno de programación Atmel
- Migrar el código desarrollado en T5 para que pueda funcionar en el Atmel
- Pruebas del código desarrollado
- Cálculo de la velocidad máxima en el procesado de imágenes
- Integración de la cámara final en la placa de desarrollo Atmel
- Implementar software de gestión de la cámara final
- Implementar software de procesamiento de imagen. Captura y calibración
- Búsqueda de los parámetros de configuración óptimos
- Aplicación de los algoritmos de cálculo de la holgura usando la cámara final

- Desarrollo de software para controlar periféricos

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

4.1.1 Instalación y configuración del entorno de programación Atmel

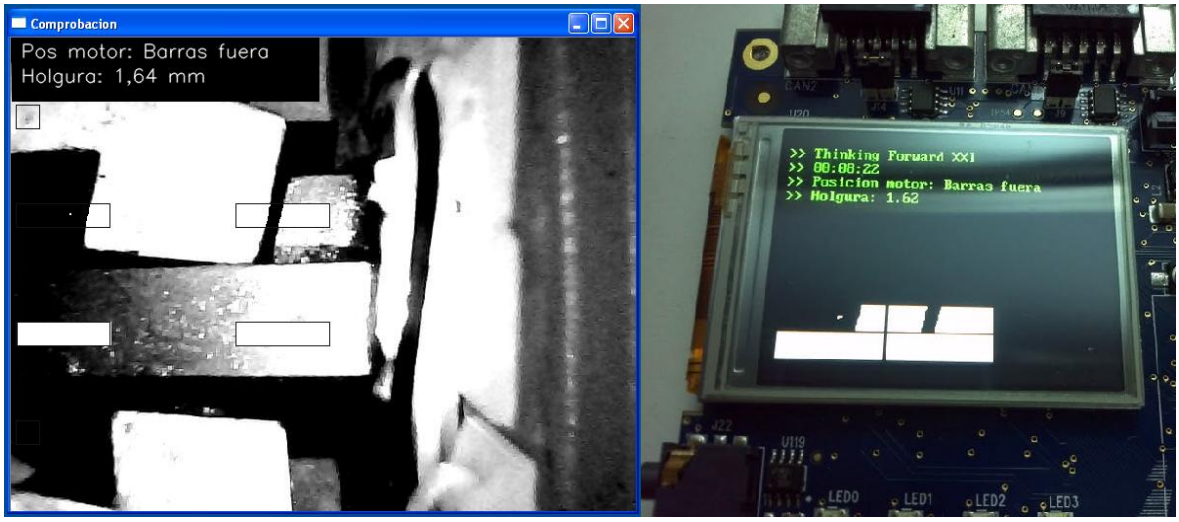
Esta subtarea básicamente consistía en instalar el entorno en mi PC para poder realizar los desarrollos necesarios. Fue muy simple debido a que todo ya se había hecho anteriormente y documentado perfectamente en la tarea T2 del proyecto.

4.1.2 Migrar el código desarrollado en T5 para que pueda funcionar en el Atmel

La migración del código se realizó en dos días. Fue muy sencillo ya que en ambos casos se trataba de C y prácticamente fue hacer copy/paste con algunas pequeñas diferencias.

4.1.3 Pruebas del código desarrollado

Tras implementar el código en el Atmel se realizaron numerosas pruebas en las que se comparaban los resultados obtenidos en el PC con los que se obtenían en el Microcontrolador siendo el resultado muy satisfactorio. En la siguiente figura se observa el resultado de una de estas pruebas.



4.1.4 Cálculo de la velocidad máxima en el procesado de imágenes

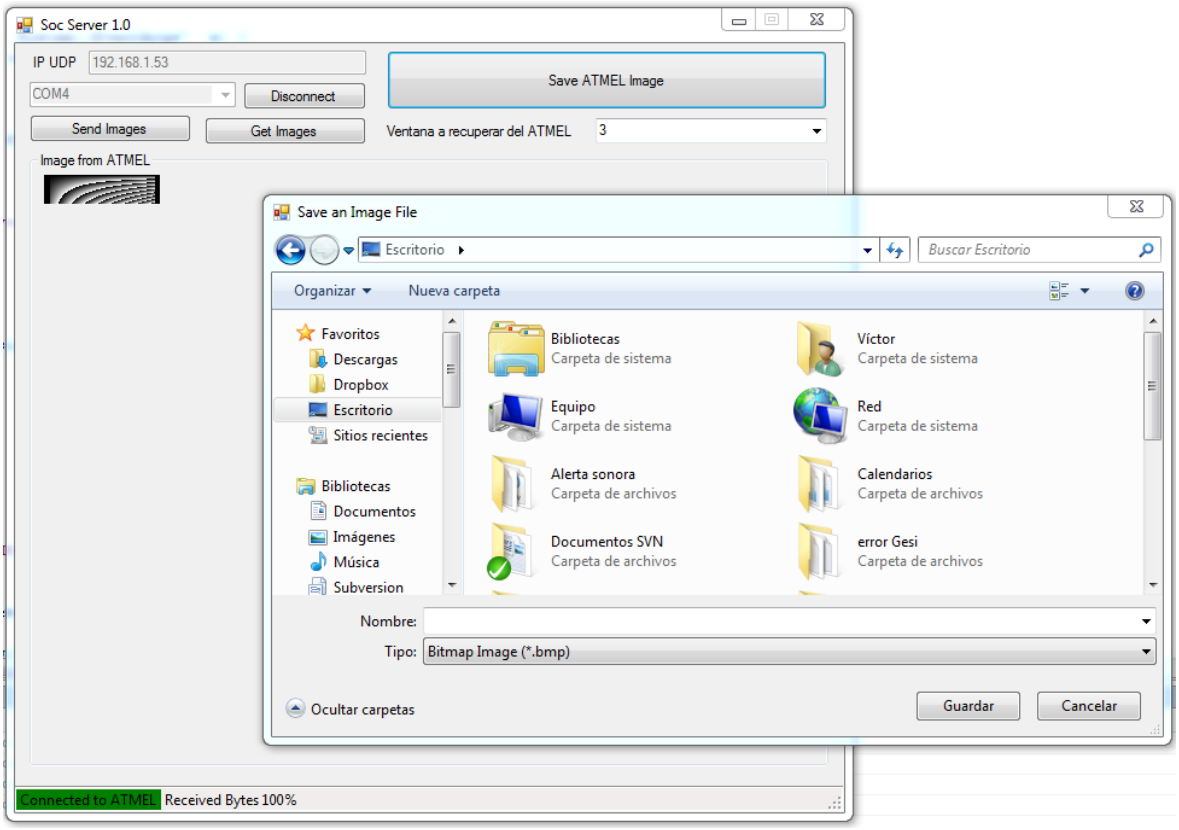
Una vez teníamos la certeza de que los algoritmos estaban funcionando exactamente igual que en el PC implementamos una porción de código para calcular el tiempo que tardaba el Atmel en calcular la holgura. Este tiempo es contabilizado a partir de que se recibe la imagen por lo que al tiempo de procesado habrá que sumarle el tiempo de adquisición una vez se sepa. Tras numerosas pruebas se concluye con que el tiempo de procesado de los algoritmos propuestos (extremadamente simples) no supera 1 milisegundo. Este dato es un resultado muy positivo ya que esto significa que, si fuese necesario, podremos en un futuro realizar cálculos más complejos sin que esto suponga un aumento de tiempo de procesado que pueda comprometer la viabilidad del proyecto.

4.1.5 Integración de la cámara final en la placa de desarrollo Atmel

Para esta subtarea se tuvo que esperar hasta que se finalizó el hardware, ya que al acabar la subtarea anterior el hardware de adquisición todavía no estaba listo por lo que se paró el desarrollo de la tarea T4 hasta finalizar dicho punto. La subtarea anterior se acabó el día 19/03/2012 y esta subtarea se empieza el día XXXXX una vez el hardware de la cámara está funcionando.

….

Durante la espera a recibir las placas de la cámara y a configurarlo todo para poder capturar imágenes de esta, se desarrolló un software tanto en el PC como en el Atmel para poder pedir imágenes al Atmel y poder guardarlas en el PC, de forma que podamos aplicar los algoritmos de cálculo de holgura en el PC y ver si obtenemos los mismo resultados que se obtienen en el Atmel. A continuación se muestra un screenshot del software de PC desarrollado.



Las conclusiones que obtuvimos durante el desarrollo de este software es que USART es un protocolo extremadamente inapropiado para transmitir grandes volúmenes de datos. Tanto es así que el envío de una ventana como la mostrada en la imagen anterior tarda 2 minutos y una imagen completa de 640 x 480 tardaría unas 4 horas. Se barajó la posibilidad de estudiar el protocolo USB e implementar una versión del código de envío de imágenes en USB, pero se complicaba bastante y enviando las 4 ventanas (8 minutos) tenemos suficiente en caso de ser necesario, por lo que se decidió no implementar la versión USB a menos de que en fases futuras del proyecto se vea que es imprescindible.

Una vez llego la placa de la cámara se empezó a integrar en la placa de desarrollo Atmel para poder seguir con los algoritmos básicos de cálculo de holgura.

4.1.6 Implementar software de gestión de la cámara final

Para empezar esta tarea era necesario tener antes el prototipo hardware funcionando correctamente. Debido a que surgieron varios problemas a la hora de testear este hardware, explicado en la tarea SOC_1202_T3_120221, esta tarea ha quedado bloqueada temporalmente hasta el día 09/05/2010 en que se reprende.

Se ha desarrollado una librería de gestión de la cámara. Esta librería, llamada APTINA, debe incluirse en el proyecto debajo de la carpeta SOFTWARE_FRAMEWORK/COMPONENTS. Básicamente consta de tres ficheros:

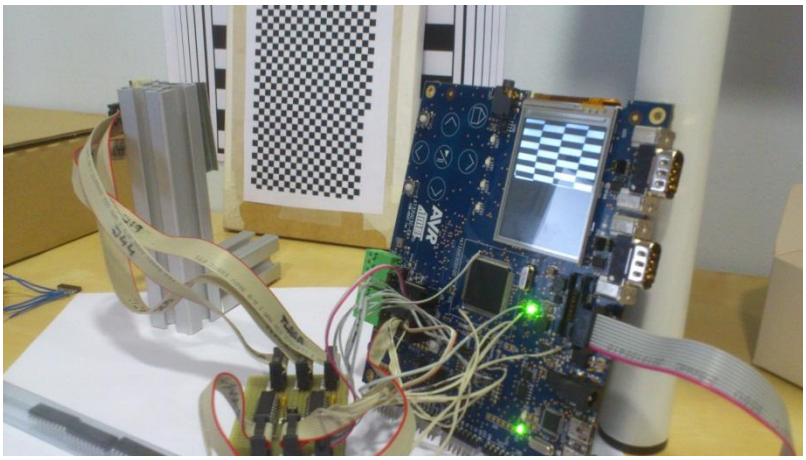
- aptina_i2c.c y aptina_i2c.h: Funciones de escritura y lectura de los principales registros del sensor de imagen.
- conf_aptina.h: Declaración de constantes para la configuración del sensor de imagen.

4.1.7 Implementar software de procesamiento de imagen. Captura y calibración

En este punto se han realizado las funciones de captura de imágenes y posterior calibración. Para adquirir las imágenes en primer lugar se estudió la posibilidad de hacerlo por interrupciones. La idea más lógica es programar las interrupciones con las señales hardware que nos da el sensor de imagen (Frame valid, Line valid y Pixel Clock). Debido a la latencia de las interrupciones es imposible realizarlo de esta forma, porqué des de cuando salta la interrupción hasta que leemos el valor del pixel ya ha pasado demasiado tiempo y el bus de datos ha cambiado. Por lo tanto, la captura de imágenes se hará en polling, consultando reiteradamente las entradas digitales del Atmel. Para mejorar el tiempo de ejecución se entraran todas las señales y datos por el mismo bus, de forma que sólo tendremos que leer un registro del procesador.

Para el tratamiento de imágenes se ha desarrollado una librería con las estructuras necesarias para guardar imágenes, procesarlas y visualizarlas por el display de la placa de desarrollo. También se han desarrollado las funciones de calibración de imágenes. Esta librería se llama image_toolbox y contiene los siguientes ficheros:

- image.c e image.h: Varias estructuras que definen las imágenes así como su inicialización y tratamiento.
- calibration.c y calibration.h: Funciones de calibración de los parámetros intrínsecos de la cámara.



También se ha desarrollado en este punto una función que nos proporciona un índice directamente proporcional al enfoque de la imagen. Cuando esta se esta enfocando correctamente el índice crece, mientras que cuando se gira la óptica al revés, el índice de enfoque disminuye. Esta funcionalidad debería ayudar a la hora de instalar las cámaras en los motores.

Se observa que de vez en cuando la imagen que capturamos no es correcta debido a que no cogemos el canal correcto. Esto es debido a que no sincronizamos el contador divisor del pixel clock con el inicio de línea. Para solucionar esto se ha tenido que revisar la tarea SOC_1202_T3.1_120522 añadiendo un transistor para hacer la señal de reset del contador mediante la señal de Line Valid del sensor de imagen.

Para la correcta captura de las imágenes hay que situar la cámara correctamente, enfocarla y ubicar las ventanas correctamente. Para estas tareas será necesario el uso de un dispositivo con display a la hora de instalar. Se han desarrollado funciones para realizar estas tareas con mayor facilidad y precisión.

4.1.8 Búsqueda de los parámetros de configuración óptimos

Se han probado muchas configuraciones ya que el datasheet del sensor no especifica como configurar la cámara. También se ha intentado contactar con Aptina para que nos proporcionen información detallada sobre la configuración y no han respondido.

Para trabajar adecuadamente con la cámara es necesario utilizar la función de exposición manual. La función de auto exposición no funciona en nuestro entorno ya que queremos sobrexponer la imagen para facilitar la tarea de binarización.

El registro que cambia a exposición manual es el R0x106. Concretamente tenemos que poner el bit 14 a 0. Entonces con el registro R0x009 cambiamos el tiempo de integración que es directamente proporcional al tiempo de exposición. Por lo tanto, aumentando el valor en este registro se comprueba que la exposición es más larga. Los valores adecuados para este registro dependen de la iluminación.

Se ha probado con dos LEDs situados a la pared delantera del motor y lo más cerca posible a las barras. Otro LED situado a la altura de la cámara. Para la barra de arriba esta iluminación es perfecta mientras que para la barra de bajo es insuficiente.

Se han comprado LEDs blancos de mayor intensidad (400mcd) …

4.1.9 Aplicación de los algoritmos de cálculo de la holgura usando la cámara final

En este punto se han implementado las funciones desarrolladas en la primera parte de la tarea. Algunas de estas funciones se han tenido que modificar para mejorar el funcionamiento del algoritmo.

Las funciones de búsqueda de líneas se han modificado. La función desarrollada inicialmente daba un valor entero para la posición de la línea y comparaba los valores de cada línea para calcular la holgura. Este algoritmo sólo es válido cuando las líneas son completamente horizontales. Para tratar con líneas no horizontales se ha modificado la función teniendo en cuenta todos los puntos de la línea. Una vez encontrada la nube de puntos que forman la línea se ajusta una línea de regresión con ellos. Se calcula la distancia entre líneas haciendo de la distancia entre el inicio y el

final de las líneas. La siguiente figura muestra gráficamente el algoritmo de cálculo. A la izquierda tenemos la nube de puntos que se ha encontrado. Estos puntos se buscan sobre una imagen binarizada y corresponden a las transiciones entre las regiones negras y blancas en cada columna de la imagen. En el medio vemos la recta de regresión con los puntos encontrados. A la derecha se ha representado las distancias que se tienen en cuenta para determinar la distancia entre líneas.

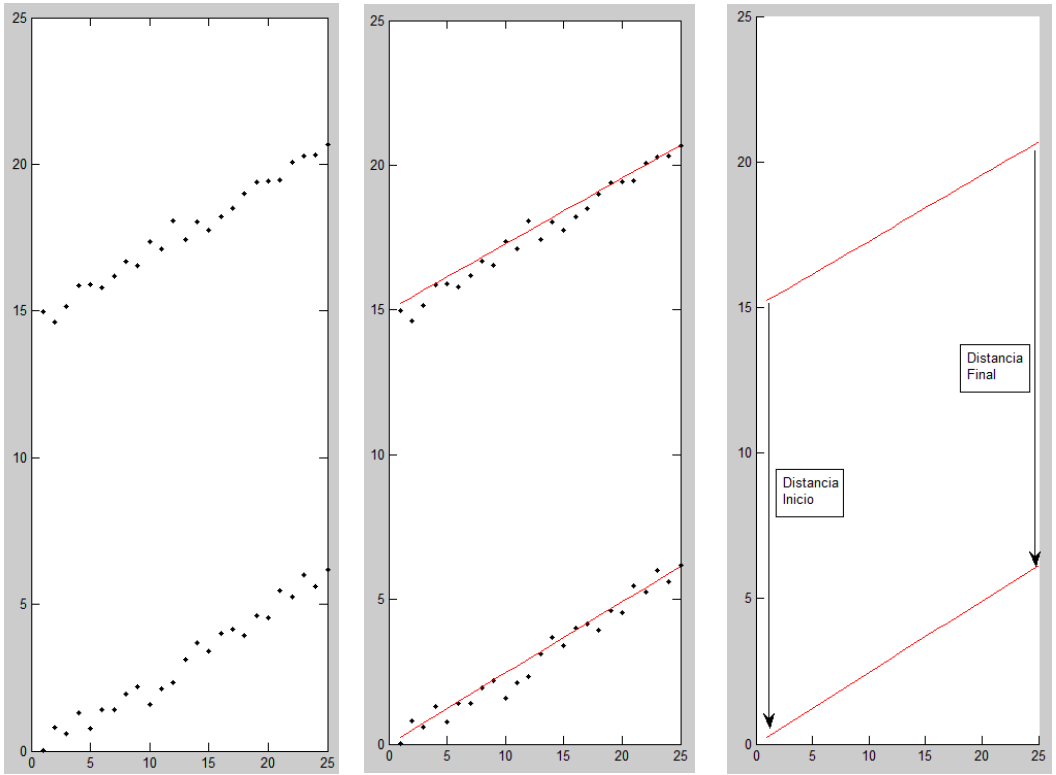
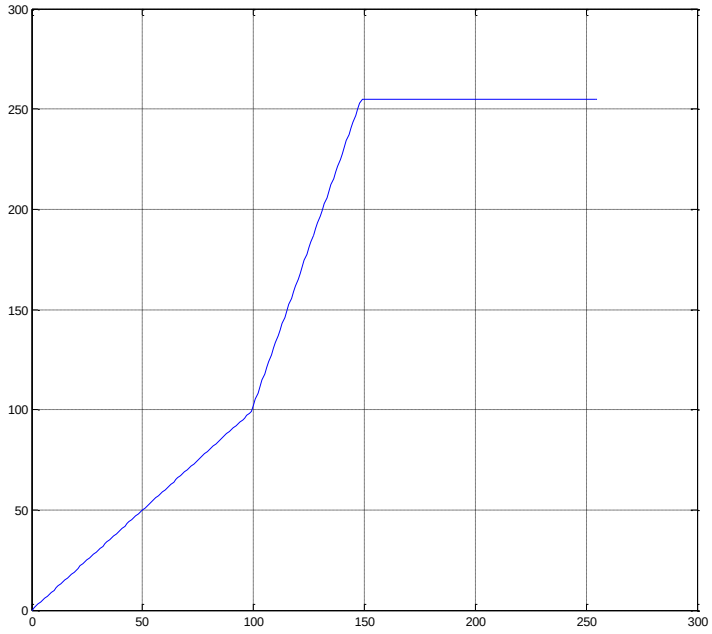


Figura. Distancia entre líneas con el nuevo algoritmo. La distancia es la media entre distancia Inicio y Distancia Final

Se ha observado que para los cálculos que estamos haciendo no es necesario calibrar las imágenes. Por otra parte han tenido que desarrollarse filtros de imagen para eliminar el ruido presente en las imágenes. Con el fin de preservar los bordes de las imágenes se utilizan filtros morfológicos. Principalmente combinaciones de openings y closings con un elemente estructural que se adapte a la imagen que capturamos de la holgura. Estas imágenes siempre presentan líneas diagonales, por lo tanto el elemento estructural deberá ser una línea diagonal para no tocar la forma de estos bordes.

En segundo lugar se ha observado que la iluminación es fundamental para el correcto funcionamiento de los algoritmos. Con tal de disminuir esta dependencia se ha implementado una función de compresión de rango dinámico. La transformación es la que se muestra a continuación. Esta función tiene 3 zonas, una lineal dónde no se modifican los píxeles, una de compresión y una de saturación. Se ha implementado la función con dos parámetros para poder modificar estas tres regiones. Después de hacer varias pruebas se ha observado que poniendo la región lineal hasta 100 y la de saturación a partir de 150 el comportamiento del algoritmo de binarización es mejor.



4.1.10 Desarrollo de software para controlar periféricos

4.1.10.1 Memoria Flash

Los parámetros de configuración de la cámara (posición de las ventanas, shutter width…) deberán configurarse en el momento de la instalación. Estos parámetros deben guardase en la cámara de forma permanente. Debido a que el procesador no dispone de memoria EPROM utilizaremos la memoria FLASH para este propósito. La FLASH dispone de una zona reservada para el usuario, la User Page, de 4048 bytes. Se ha desarrollado funciones de lectura y escritura de esta página para guardar los valores de los parámetros de configuración.

4.1.10.2 Pantalla táctil

Durante la instalación deberá conectarse un display al SOC para poder configurar los distintos parámetros. Se utilizará un display táctil como el que hay en la placa de desarrollo AT32UC3C-EK. Por lo tanto el dispositivo final deberá tener un conector para el display.

Se ha desarrollado el código necesario para controlar el display táctil mediante el microcontrolador. El dispositivo verificador de instalaciones no dispondrá de microcontrolador, sino que deberán implementarse las funcionalidades de instalación en el propio SOC. Se han desarrollado funciones para este propósito.

Se han buscado pantallas táctiles y la mas parecida a la de la placa de desarrollo es la NHD-2.4-240320SF-CTXI#-FT1. Se puede comprar directamente en digi-key. Venden por separado la placa NHD-FFC40-ND con el conector FFC Molex# 54132-4097 de cable plano para el display.

4.1.10.3 CAN

El SOC se comunicará con el DAEN y con el ECON mediante bus CAN. En este punto se ha realizado la integración del microcontrolador con el bus CAN. Se han realizado test de funcionamiento con el ECON. También se ha desarrollado una librería para el CAN. En la carpeta

de recursos de la tarea existe un documento explicando los pasos a seguir para incluir CAN en un proyecto.

4.2 Conclusiones

Se corrobora que la portabilidad del código de C++ (PC) a C en Atmel es muy elevada, haciendo el desarrollo muy cómodo en el PC y luego pudiendo portar los códigos desarrollados al Atmel sin que suponga un problema mayor.

El tiempo de cálculo de la holgura en el Atmel con los algoritmos propuestos es inferior a un milisegundo. Es muy posible que una vez tengamos la cámara final funcionando debamos realizar cambios en los algoritmos o usar soluciones más complejas para aumentar la fiabilidad, pero esto no debería suponer un aumento de tiempo de cálculo no tolerable.

Sera necesario trabajar mucho con la iluminación como se comentó ya en la T5 ya que afecta de forma DRÁSTICA a los resultados obtenidos.

Será necesario el uso de un dispositivo con display para realizar las instalaciones correctamente. Se deberá dejar un conector para el display en en SOC. El dispositivo de verificación de instalaciones no tendrá microcontrolador, por lo que las funciones de instalación se implementarán en el SOC.

Se utilizará una región de la FLASH para guardar los parámetros de configuración.

No es necesario calibrar las imágenes para tener lecturas de holgura correctas.

Se calculan las primeras holguras con el prototipo. Las medidas en la barra superior son muy buenas. En la barrar inferior…. Debido a la baja velocidad de adquisición de imágenes procesamos 1 imagen cada 4 segundos. El algoritmo de cálculo es muy rápido, tardando unos pocos milisegundos en procesar las imágenes.

Fecha de finalización de la tarea 11/07/2012