Taller de sistemas embebidos (TA134)

Trabajo Práctico Final

Grupo N°2

2°C 2024

Integrantes del Grupo		
Nombre y apellido	Padrón	Correo
Juan Ignacio Biancuzzo	106005	jbiancuzzo@fi.uba.ar
Manuel Lalia	107979	mlalia@fi.uba.ar
Rocio Nicole Heredia Piñon	107621	rherediap@fi.uba.ar

Repositorio de Github

Índice

1. Descripción del sistema de control

El sistema de control se encarga de contabilizar la cantidad de autos entrantes a un estacionamiento, donde se tiene en cuenta las limitaciones físicas del mismo proporcionando dos parámetros configurables, los cuales son una advertencia de llenado y la máxima capacidad del estacionamiento.

En su modo normal, se usa el display para obtener la información esencial del sistema, como la cantidad de autos y los parámetros mencionados anteriormente. En su modo set up, el sistema usa el display para permitirle al usuario configurar los parámetros.

- app.c (app.h)
  - Endless loop, ejecuta las tareas de sensores, temperatura, sistema, actuadores y display, activando la secuencia cuando un evento es transmitido de los sensores.
- task_sensor.c (task_sensor.h, task_sensor_attribute.h)
  - Controlando los botones y Dip Switch
- task_temperatura.c (task_temperatura.h, task_temperatura_attribute.h)
  - Lectura de los sensores de temperatura interno y externo mediante un ADC
- task_system.c (task_system.h, task_system_attribute.h)
  - Intermediario de los sistemas normal y set up, controlando el estado del sistema
- task_normal.c (task_normal.h, task_normal_attribute.h)
  - Estado normal de ejecución, contando el ingreso y egreso de autos al estacionamiento, limitado por su máxima capacidad
- task_set_up.c (task_set_up.h, task_set_up_attribute.h)
  - Estado set up de ejecución, permitiendo modificar los parámetros de capacidad máxima y advertencia de capacidad
- task_system_interface.c (task_system_interface.h)
  - Permite la comunicación entre los sensores y los ADC de temperatura, con el sistema intermediario, de forma desacoplada y modularizada
- task_actuator.c (task_actuator.h, task_actuator_attribute.h)
  - Límitador y Baliza de Sistema con Sirena
- task_actuator_interface.c (task_actuator_interface.h)
  - Permite la comunicación entre los sistemas (normal y set up) con los actuadores de forma desacoplada y modularizada
- task_display.c (task_display.h)
  - LCD Display
- task_display_interface.c (task_display_interface.h)
  - Permite la comunicación entre los sistemas (normal y set up) con el display de forma desacoplada y modularizada

- logger.h (logger.c)
  - Utilities for Retarget "printf"to Console
- dwt.h
  - Utilities for Mesure çlock cycle.ªnd .exªecution time.ºf code

2. Esquema eléctrico y cableado

Veamos el esquemático del sistema general.

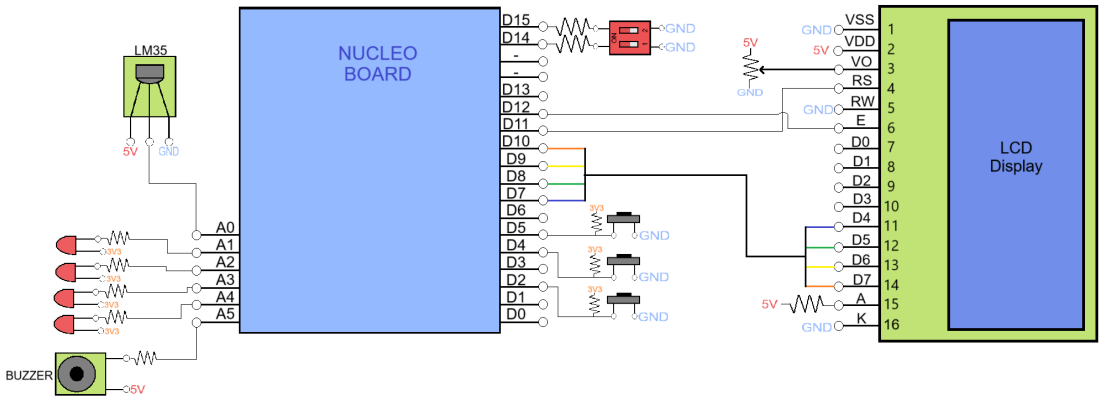

Figura 1: Esquemático eléctrico

3. Calculo y medición del consumo

Veamos el calculo del consumo, para eso hagamos un recuento de los componentes que tenemos:

- 4 LED's
- 2 Dip Switch's
- Un Display
- Un Buzzer KY-012
- Un sensor de temperatura LM35

El sensor de temperatura consume 60 μA, por lo que vamos a despreciar su consumo. El Display en su uso típico es de 1,5 mA.

En el uso normal, se tiene 3 led's en uso y los dos dip switch, a la par del display, por lo que tendremos que calcular el consumo de un led y un dip switch.

Para el led los estamos alimentando con 3,3 V por lo que podemos conseguir la corriente de la siguiente forma:

$$I_{\text{LED}} = \frac{V_{DD} - 0{,}7V}{1\ k\Omega} = 2{,}6\ mA$$

Para los dip switch's tenemos

$$I_{\text{DipSwitch}} = \frac{V_{DD}}{10\ k\Omega} = 0{,}33\ mA$$

Por último para el buzzer

$$I_{\text{Buzzer}} = \frac{V_{DD}}{1\ k\Omega} = 3{,}3\ mA$$

	Corriente calculada $[mA]$	Corriente medida $[mA]$
Uso normal ($I_{\text{Display}} + 3\ I_{\text{LED}} + 2\ I_{\text{DipSwitch}}$)	$\approx$ 9,9	7,9
Alimentando el buzzer ($+I_{\text{Buzzer}}$)	$\approx$ 13,2	11

4. Items implementados

Vamos a enunciar los items, de este trabajo práctico final, que fueron implementados como el item que no se pudo terminar de implementar.

Se pudo implementar

- La task de sensores para medir correctamente si los botones fueron pulsados o los dip switches fueron modificados.
- La task de temperatura para poder transformar la medición de tensión dada por los sensores de temperatura y transformarlos en información manejable en el sistema.
- La task de sistema normal, manejando el conteo de autos y teniendo en cuenta el máximo posible.
- La task de sistema set up, donde podemos configurar los valores de advertencia y el máximo posible de autos.
- La task de actuadores, donde podemos indicar el estado del sistema.
- La task de display, donde podemos mandar información al display de forma correcta.

No se pudo implementar la transformación de la lectura del ADC a la temperatura en Celsius.

5. WCET y Factor de uso

Veamos primero los WCET (Worst-case execution time) en microsegundos:

	tiempo $[\mu s]$
Task Sensor	35023
Task Temperatura	65206
Task Sistema	6044
Task Actuador	26558
Task Display	344707

Recordemos que estamos midiendo el tiempo de cada tarea y nos quedamos con el peor tiempo.

El factor de uso, nos parece importante mostrarlo en diferentes momentos de la ejecución del sistema, ya que como esta dada por:

$$U = \sum C_{\text{TASK}} + \sum C_{\text{ISR}}$$

Dado que solo usamos una interrupción dada cada 1 ms donde se ejecutan 6 instrucciones, podemos despreciar su tiempo de ejecución. Eso nos da los siguientes valores:

	tiempo $[\mu s]$
Mínimo	44
Uso normal	300
Máximo	345000

Notemos que en el uso normal estamos 3 veces por debajo del milisegundo, por lo que no estamos sobrecargados, pero en los picos donde llegamos al máximo de 345 ms estamos ocupando 345 ciclos extra. Esto sería un problema si el sistema llega a estos máximos en una cantidad menor a $345ms$, pero pudimos ver que esto ocurre cuando se tiene que manejar un evento y como esperamos que un usuario del sistema no pueda generar un evento en un tiempo menor a 345 ms podemos decir que el sistema no se sobrecarga.